Appendix C

Lessons to be learned by comparing integrated fisheries stock assessment models (SAMs) with integrated population models (IPMs) https://doi.org/10.1016/j.fishres.2023.106925

- Supplementary figures
- Script and data to run the SAM for Australasian snappers
- Script and data to run the IPM for woodchat shrikes

1. Supplementary figures

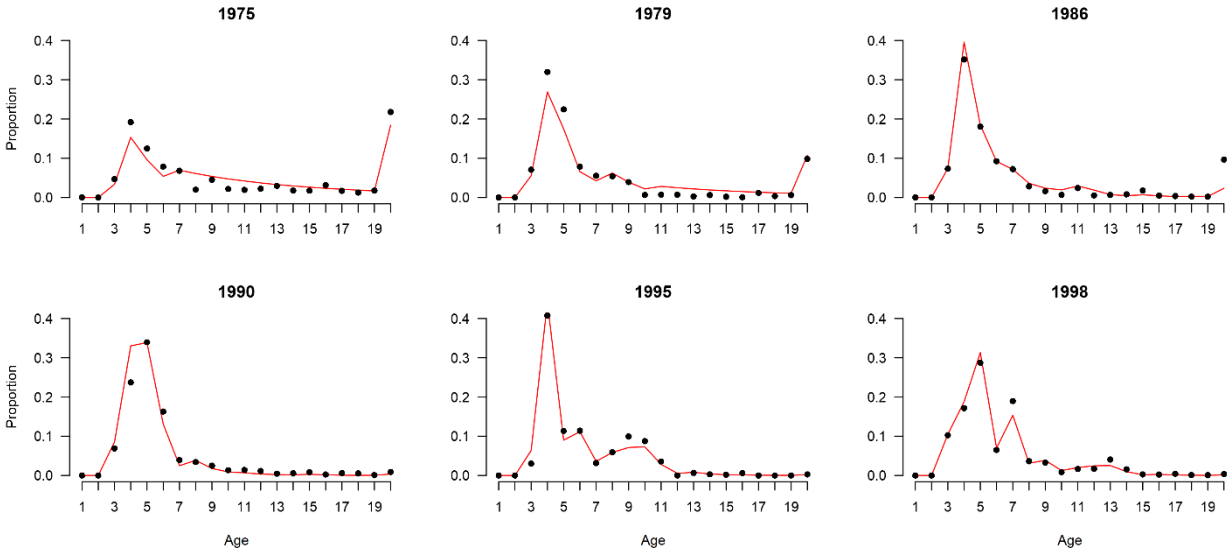

Figure S1. Example fits (red lines) to the age-composition data for the New Zealand snapper application. The dots are the observed age-composition (data).

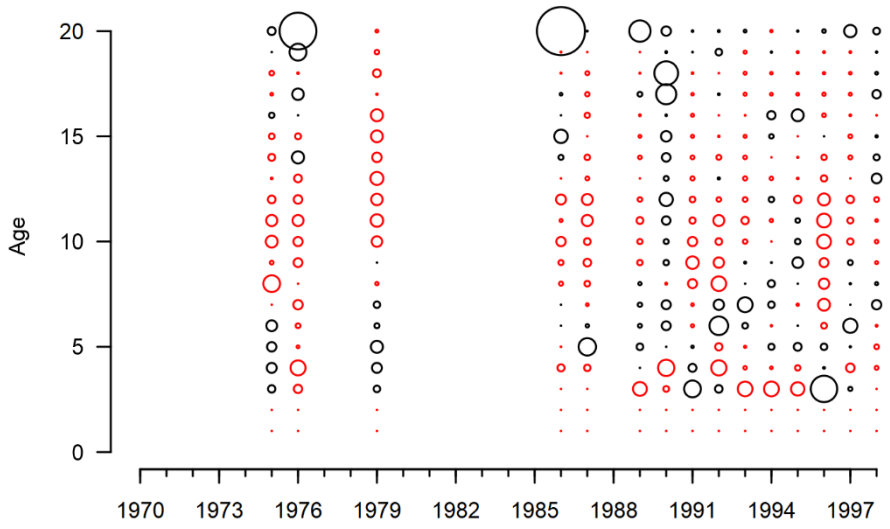

Figure S2. Pearson residuals for the fit to the age-composition data (black is positive and red is negative) for the New Zealand snapper application.

2. Script and data to run the SAM for Australasian snappers

For a recent assessment and data and the references therein for historical information see Langley (2021).

Langley, A.D. 2021. Stock assessment of snapper in SNA 8 for 2021. New Zealand Fisheries Assessment Report 2021/38 https://www.mpi.govt.nz/dmsdocument/45931-FAR-202138-Stock-assessment-of-snapper-in-SNA-8-for-2021

 # Set up the file paths, load libraries, and read in scripts

```
setwd(...)

library(TMB)
library(Matrix)
library(scales)
compile("SAM.cpp")                  # for SAM.cpp, see below
dyn.load(dynlib("SAM"))
```

 # Read in data

```
Catch <- c(214.5, 241.5, 340.5, 402, 238.5, 216, 259.5, 246, 160.5, 160.5,
153, 114.4, 131.7, 114.4, 203.5, 339.2, 353.8, 311.2, 424.3, 506.7, 552,
553.3, 1358.3, 1327.8, 1398.9, 1480.3, 1905.2, 1450.9, 1921, 2003.5,
1983.2, 2127.8, 2251, 2019.3, 1948.1, 2582.1, 2267.9, 2070.2, 2584.3,
2405.8, 2524.4, 3591.1, 3986.6, 5320.7, 5946, 7991.6, 6364.4, 4779.2, 3921,
3737, 3560.7, 2978.3,2052.4,1924.7,1788.6,1998.2, 1075.2, 1604.5, 1730.1,
1788.9, 1898, 1704, 1783, 1780, 1674, 1798, 1853.47, 1588.08)
```

```
Wage <- c(0.0412144, 0.228313, 0.552429, 0.9361, 1.28073, 1.67467, 2.1008,
2.48168, 2.78687, 3.10869, 3.44702, 3.6817, 3.92366, 4.17285, 4.3508,
4.56018, 4.69288, 4.82737, 4.96366, 5.10171)

Index <- matrix(c(44, 1, 0.2,
45, 0.85, 0.2,
46, 0.86, 0.2,
47, 0.46, 0.2,
48, 0.59, 0.2,
49, 0.37, 0.2,
50, 0.25, 0.2,
51, 0.21, 0.2,
52, 0.2, 0.2,
53, 0.15, 0.2,
54, 0.11, 0.2,
55, 0.13, 0.2,
56, 0.07, 0.2,
57, 0.13, 0.2,
58, 0.16, 0.2,
59, 0.25, 0.2,
60, 0.67, 0.2,
61, 0.36, 0.2), ncol=3, byrow=TRUE)

CatchAge <- matrix(c(45, 50, 0, 0, 0.04699, 0.19179, 0.12476, 0.0787,
0.06807, 0.02002, 0.04519, 0.02174, 0.01935, 0.02241, 0.02951, 0.01787,
0.01726, 0.03143, 0.01702, 0.01262, 0.01753, 0.21775,
46, 50, 0, 0, 0.02703, 0.04785, 0.12634, 0.07193, 0.02767, 0.06084,
0.03418, 0.02797, 0.01979, 0.02107, 0.01917, 0.05002, 0.0168, 0.02366,
0.03737, 0.01631, 0.0384, 0.3336,
49, 50, 0, 0, 0.07072, 0.31981, 0.2246, 0.07848, 0.0555, 0.05422, 0.03929,
0.00651, 0.00742, 0.00694, 0.00279, 0.00619, 0.00186, 0.0005, 0.01165,
0.0035, 0.00607, 0.09824,
56, 50, 0, 0, 0.07335, 0.35161, 0.18071, 0.09235, 0.07236, 0.02853,
0.01602, 0.00653, 0.02414, 0.0054, 0.00657, 0.00794, 0.01774, 0.00461,
0.0037, 0.00237, 0.00208, 0.09653,
57, 50, 0, 0, 0.2596, 0.13577, 0.33308, 0.12652, 0.05094, 0.03301, 0.01115,
0.00675, 0, 0.00358, 0.00778, 0.00086, 0.00286, 0.00091, 0.00048, 0,
0.00161, 0.01766,
59, 50, 0, 0, 0.0965, 0.49622, 0.21914, 0.0439, 0.06608, 0.03126, 0.00699,
0.00501, 0.00034, 0.00078, 0.00249, 0.0018, 0.00122, 0.0004, 0.00184,
0.00054, 0.00041, 0.0186,
60, 50, 0, 0, 0.06879, 0.23712, 0.33944, 0.16281, 0.03927, 0.03428,
0.02501, 0.01351, 0.01391, 0.01131, 0.00458, 0.00557, 0.00837, 0.00259,
0.00594, 0.00541, 0.00116, 0.00898,
61, 50, 0, 0, 0.07629, 0.25682, 0.27952, 0.25738, 0.09698, 0.00756,
0.00915, 0.00347, 0.00204, 0.00145, 0.00071, 0.00025, 0.00034, 0.00086,
0.0005, 0.00012, 0.00054, 0.00432,
62, 50, 0, 0, 0.13174, 0.06019, 0.14905, 0.30392, 0.2717, 0.04371, 0.00329,
0.01488, 0, 0.0025, 0.00611, 0, 0.00091, 0.001, 0.0027, 0.00096, 0.00185,
0.00415,
63, 50, 0, 0, 0.02797, 0.24092, 0.07684, 0.15693, 0.22509, 0.17016,
0.07073, 0.00765, 0.00912, 0.00463, 0.00178, 0.00133, 0.0005, 0.00081, 0,
0.00029, 0, 0.00408,
64, 50, 0, 0, 0.14969, 0.13231, 0.20192, 0.05341, 0.11259, 0.13622,
0.11938, 0.04315, 0.00557, 0.01924, 0.00338, 0.00305, 0.00452, 0.00404, 0,
0, 0.00143, 0.00128,
65, 50, 0, 0, 0.03027, 0.40755, 0.11333, 0.11403, 0.03172, 0.05917,
0.09933, 0.08759, 0.03579, 0, 0.00669, 0.00302, 0.00181, 0.00607, 0, 0, 0,
0.00284,
66, 50, 0, 0, 0.3164, 0.1491, 0.3468, 0.049, 0.0455, 0.009, 0.0233, 0.0193,
0.0203, 0.0031, 0, 0.0018, 0.0028, 0, 0, 0, 0, 0.0033,
```

```
67, 50, 0, 0, 0.0889, 0.3538, 0.0956, 0.2631, 0.0407, 0.0537, 0.0225,
0.0166, 0.0218, 0.0197, 0.0123, 0.0005, 0.0014, 0.001, 0, 0, 0, 0.0059,
68, 50, 0, 0, 0.10257, 0.17146, 0.2873, 0.06498, 0.18967, 0.03654, 0.03281,
0.00854, 0.01661, 0.01723, 0.04073, 0.01565, 0.00284, 0.00235, 0.00438,
0.00138, 0.00087, 0.00371), nrow=15, byrow=TRUE)
```

 # Set up the TMB files

```
# Data
SAM_data <- list(
  Nyears = length(Catch),
  Nages = dim(CatchAge)[2]-2,
  C = Catch,
  Wage = Wage,
  Nindex = dim(Index)[1],
  Index = t(Index),        #Year, Value, SE
  NCA = dim (CatchAge)[1],
  CatchAge = t(CatchAge)   #Year, SS, age1, age2, ...
)

# Parameters
SAM_pars <- list(
  ln_R0 = log(4000),
  ln_M = log(0.075),
  ln_Csd = log(0.05),
  ln_Rsd = log(0.6),
  ln_A50 = log(3),
  ln_A95 = log(4),
  ln_q = log(0.4),
  ln_F = log(rep(0.1,SAM_data$Nyears)),
  Rdev = rep(0.5*0.6*0.6,SAM_data$Nyears+1),
  Rinitdev = rep(0.5*0.6*0.6,SAM_data$Nages-2)
)

# Control which parameters are estimated
SAM_map <- list(
  ln_M = factor(NA),
  ln_Csd = factor(NA),
  ln_Rsd = factor(NA),
  ln_A50 = factor(NA),
  ln_A95 = factor(NA),
  Rdev =
c(rep(factor(NA),39),factor(40:65),rep(factor(NA),(SAM_data$Nyears+1)-
66+1)),
  Rinitdev = rep(factor(NA),SAM_data$Nages-2)
)
```

 # Run model

```
# First phase fixed selectivity
obj <- MakeADFun(SAM_data,SAM_pars,DLL="SAM",map=SAM_map,silent=T)
opt <- nlminb(obj$par,obj$fn,obj$gr)    # estimate the parameters

# Second phase estimate selectivity
SAM_map$ln_A50 <- NULL
SAM_map$ln_A95 <- NULL
obj <-
MakeADFun(SAM_data,obj$env$parList(opt$par),DLL="SAM",map=SAM_map,silent=T)
opt <- nlminb(obj$par,obj$fn,obj$gr)    # estimate the parameters
```

```
opt$objective
opt$convergence
obj$gr(opt$par)

sdrep<-summary(sdreport(obj))
sdrep

rep <- obj$report()
rep$B
rep$N
```

 # Produce figures

```
# Figure 2
png("Fig.2.png", width = 2*1500, height = 2*1000, res=300, type='cairo')
cex.tif <- 1
colo <- c('grey', 'black')
lw <- 1.5
year <- 1931:1998
pick <- seq(from=10, to=length(year), by=10)
pick2 <- seq(from=5, to=length(year), by=10)

layout(matrix(1:4,2,2,byrow = TRUE), widths = c(1.5,1.5), heights = c(1,1),
TRUE)
par(las  = 1, cex = cex.tif, mar=c(4,4,2,1))

tt.x <- c(1:(SAM_data$Nyears+1), (SAM_data$Nyears+1):1)
tt.y <- c(sdrep[row.names(sdrep)=="B",1]-
1.96*sdrep[row.names(sdrep)=="B",2],
rev(sdrep[row.names(sdrep)=="B",1]+1.96*sdrep[row.names(sdrep)=="B",2]))
plot(1:(SAM_data$Nyears+1), rep$B, type="l", xlab=NA, ylab="Vulnerable
biomass (B) * 1000", ylim=c(0,max(tt.y)), axes=FALSE)
polygon(tt.x ,tt.y, col=alpha(colo[1], 0.4), border=alpha(colo[1], 0.4))
lines(1:(SAM_data$Nyears+1), rep$B, col=colo[2], lwd=lw)
axis(2, las=1, at=c(0, 20000, 40000, 60000, 80000), labels=c(0, 20, 40, 60,
80))
axis(1, at=pick2, tcl=-0.25, labels=NA)
axis(1, at=pick, labels=year[pick], tcl=-0.5)
axis(1, at=1:(length(year)+1), tcl=0, labels=NA)

tt.x <- c(1:(SAM_data$Nyears),(SAM_data$Nyears):1)
tt.y <- c(sdrep[row.names(sdrep)=="F",1]-
1.96*sdrep[row.names(sdrep)=="F",2],

rev(sdrep[row.names(sdrep)=="F",1]+1.96*sdrep[row.names(sdrep)=="F",2]))
plot(1:(SAM_data$Nyears), rep$F, type="l", xlab=NA, ylab="Fishing mortality
(F)", ylim=c(0,max(tt.y)), axes=FALSE)
polygon(tt.x ,tt.y, col=alpha(colo[1], 0.4), border=alpha(colo[1], 0.4))
lines(1:(SAM_data$Nyears), rep$F, col=colo[2], lwd=lw)
axis(2, las=1)
axis(1, at=pick2, tcl=-0.25, labels=NA)
axis(1, at=pick, labels=year[pick], tcl=-0.5)
axis(1, at=1:length(year), tcl=0, labels=NA)

tt.x <- c(1:(SAM_data$Nyears+1), (SAM_data$Nyears+1):1)
tt.y <- c(sdrep[row.names(sdrep)=="R",1]-
1.96*sdrep[row.names(sdrep)=="R",2],
rev(sdrep[row.names(sdrep)=="R",1]+1.96*sdrep[row.names(sdrep)=="R",2]))
```

```
plot(1:(SAM_data$Nyears+1), rep$N[,1], type="l", xlab=NA, ylab="Recruiment
(R)", axes=FALSE, ylim=c(0, max(tt.y)))
polygon(tt.x ,tt.y, col=alpha(colo[1], 0.4), border=alpha(colo[1], 0.4))
lines(1:(SAM_data$Nyears+1), rep$N[,1], col=colo[2], lwd=lw)
axis(2, las=1)
axis(1, at=pick2, tcl=-0.25, labels=NA)
axis(1, at=pick, labels=year[pick], tcl=-0.5)
axis(1, at=1:(length(year)+1), tcl=0, labels=NA)

plot(1:SAM_data$Nages, rep$s, type="l", xlab="Age", ylab="Selectivity",
axes=FALSE, lwd=lw)
axis(2, las=1)
axis(1, at=1:20)
dev.off()


# Figure 3
cex.tif <- 1
colo <- c('grey', 'black')
lw <- 1.5
year <- 1931:1998
pick <- seq(from=10, to=length(year), by=10)
pick2 <- seq(from=5, to=length(year), by=10)

png("Fig.3a.png", width = 1*1500, height = 2*1000, res=300, type='cairo')
lw <- 1.5
layout(matrix(1:2,2,1,byrow = TRUE), widths = 1.5, heights = c(1, 1), TRUE)
par(las  = 1, cex = cex.tif, mar=c(4,4,2,1))

plot(1:(SAM_data$Nyears+1), rep$Ipred, type="l", ylim=c(0,max(rep$Ipred,
SAM_data$Index[2,])), xlab=NA, ylab="Index of relative abundance",
col="black", axes=FALSE)
points(SAM_data$Index[1,], SAM_data$Index[2,], pch=16, col='red')
segments(SAM_data$Index[1,], SAM_data$Index[2,]-
1.96*SAM_data$Index[2,]*SAM_data$Index[3,], SAM_data$Index[1,],
SAM_data$Index[2,]+1.96*SAM_data$Index[2,]*SAM_data$Index[3,], col='red')
lines(1:(SAM_data$Nyears+1), rep$Ipred, col="black", lwd=lw)
axis(2, las=1)
axis(1, at=pick2, tcl=-0.25, labels=NA)
axis(1, at=pick, labels=year[pick], tcl=-0.5)
axis(1, at=1:(length(year)+1), tcl=0, labels=NA)

plot(1:(SAM_data$Nyears), rep$Cpred, type="l",
ylim=c(0,max(rep$Cpred,SAM_data$C)), xlab=NA, ylab="Catch", col="black",
axes=FALSE)
points(SAM_data$C, pch=16, col='red')
segments(1:SAM_data$Nyears, SAM_data$C-
1.96*SAM_data$C*exp(SAM_pars$ln_Csd), 1:SAM_data$Nyears,
SAM_data$C+1.96*SAM_data$C*exp(SAM_pars$ln_Csd), col='red')
lines(1:(SAM_data$Nyears), rep$Cpred, col="black", lwd=lw)
axis(2, las=1)
axis(1, at=pick2, tcl=-0.25, labels=NA)
axis(1, at=pick, labels=year[pick], tcl=-0.5)
axis(1, at=1:(length(year)+1), tcl=0, labels=NA)
dev.off()


# Figure S1
png("Fig.S1.png", width = 3*1200, height = 2*1000, res=300, type='cairo')
lw <- 1.5
cex.tif <- 0.8
```

```
layout(matrix(1:6,2,3,byrow = TRUE), widths = c(1.5, 1.5, 1.5), heights =
c(1, 1), TRUE)
par(las = 1, cex = cex.tif, mar=c(4,4,2.5,1))
year <- 1931:1998

plot(1:SAM_data$Nages, rep$CApred[SAM_data$CatchAge[1,1],], type="l",
ylim=c(0,0.4), xlab=NA, ylab="Proportion", main=year[45], axes=FALSE,
col='red')
points(1:SAM_data$Nages, SAM_data$CatchAge[3:(SAM_data$Nages+2),1], pch=16)
axis(2, las=1)
axis(1, at=1:20)

plot(1:SAM_data$Nages, rep$CApred[SAM_data$CatchAge[1,3],], type="l",
ylim=c(0,0.4), xlab=NA, ylab=NA, main=year[49], axes=FALSE, col='red')
points(1:SAM_data$Nages, SAM_data$CatchAge[3:(SAM_data$Nages+2),3], pch=16)
axis(2, las=1)
axis(1, at=1:20)

plot(1:SAM_data$Nages, rep$CApred[SAM_data$CatchAge[1,4],], type="l",
ylim=c(0,0.4), xlab=NA, ylab=NA, main=year[56], axes=FALSE, col='red')
points(1:SAM_data$Nages, SAM_data$CatchAge[3:(SAM_data$Nages+2),4], pch=16)
axis(2, las=1)
axis(1, at=1:20)

plot(1:SAM_data$Nages, rep$CApred[SAM_data$CatchAge[1,7],], type="l",
ylim=c(0,0.4), xlab="Age", ylab="Proportion", main=year[60], axes=FALSE,
col='red')
points(1:SAM_data$Nages, SAM_data$CatchAge[3:(SAM_data$Nages+2),7], pch=16)
axis(2, las=1)
axis(1, at=1:20)

plot(1:SAM_data$Nages, rep$CApred[SAM_data$CatchAge[1,12],], type="l",
ylim=c(0,0.4), xlab="Age", ylab=NA, main=year[65], axes=FALSE, col='red')
points(1:SAM_data$Nages, SAM_data$CatchAge[3:(SAM_data$Nages+2),12],
pch=16)
axis(2, las=1)
axis(1, at=1:20)

plot(1:SAM_data$Nages, rep$CApred[SAM_data$CatchAge[1,15],], type="l",
ylim=c(0,0.4), xlab="Age", ylab=NA, main=year[68], axes=FALSE, col='red')
points(1:SAM_data$Nages, SAM_data$CatchAge[3:(SAM_data$Nages+2),15],
pch=16)
axis(2, las=1)
axis(1, at=1:20)
dev.off()


# Figure S2
png("Fig.S2.png", width = 29/20*1000, height = 1000, res=300, type='cairo')
cex.tif <- 0.7
par(las = 1, cex = cex.tif, mar=c(4,4,2.5,1))
res <- matrix(data = NA, nrow = SAM_data$NCA, ncol = SAM_data$Nages)
year <- 1931:1998

plot(NULL, xlim=c(40,SAM_data$Nyears), ylim=c(0,SAM_data$Nages+0.5),
ylab="Age", xlab=NA, axes=FALSE)
for (i in 1:SAM_data$NCA) {
  for (a in 1:SAM_data$Nages) {
    res[i,a] <- (SAM_data$CatchAge[a+2,i]-
rep$CApred[SAM_data$CatchAge[1,i],a])/sqrt(rep$CApred[SAM_data$CatchAge[1,i
],a])
    if(res[i,a]<0) use.col="red" else use.col="black"
```

```
    points(SAM_data$CatchAge[1,i], a, cex=abs(res[i,a])*10, col=use.col)
  }
}
axis(2, las=1)
sel <- 40:SAM_data$Nyears
pick <- seq(from=1, to=length(sel), by=3)
axis(1, at=sel[pick], labels=year[sel[pick]])
axis(1, at=sel, labels=NA, tcl=-0.25)
dev.off()
```

#####################
#
 # SAM.cpp
#
#####################

All code below is to be copied into a text file and saved as `SAM.cpp`. This file is then loaded with the compile call (see above)

```
# include <TMB.hpp>
# include <iostream>

template<class Type>
Type objective_function<Type>::operator() ()
{
   /* Data section */
  DATA_INTEGER(Nyears)
  DATA_INTEGER(Nages)
  DATA_VECTOR(C)
  DATA_VECTOR(Wage)
  DATA_INTEGER(Nindex)
  DATA_ARRAY(Index) //Year, Value, SE
  DATA_INTEGER(NCA)
  DATA_ARRAY(CatchAge) //Year, SampleSize, values age1, age2, ....

  /* Parameter section */
  PARAMETER(ln_R0);
  PARAMETER(ln_M);
  PARAMETER(ln_Csd);
  PARAMETER(ln_Rsd);
  PARAMETER(ln_A50);
  PARAMETER(ln_A95);
  PARAMETER(ln_q);
  PARAMETER_VECTOR(ln_F);
  PARAMETER_VECTOR(Rdev);
  PARAMETER_VECTOR(Rinitdev);

  // ***** Set up parameters *****
  Type R0 = exp(ln_R0);
  Type M = exp(ln_M);
  Type Csd = exp(ln_Csd);
  Type Rsd = exp(ln_Rsd);
  Type A50 = exp(ln_A50);
  Type A95 = exp(ln_A95);
  Type q = exp(ln_q);

  vector<Type> F(Nyears);
  F = exp(ln_F);
```

```
  // ***** Set up variables *****
  //Note: c++ vectors and arrays start with index 0
  Type Rpen;
  array<Type> N(Nyears+1,Nages);
  vector<Type> B(Nyears+1);
  vector<Type> s(Nages);
  vector<Type> Cpred(Nyears);
  vector<Type> Clike(Nyears);
  vector<Type> Ipred(Nyears+1);
  vector<Type> Ilike(Nindex);
  array<Type> CApred(Nyears,Nages);
  vector<Type> CAsum(Nyears);
  vector<Type> CAlike(NCA);

    //Selectivity
  for(int age=0;age<Nages;age++) s(age) = 1/(1+exp(log(19)*((age+1)-
A50)/(A50-A95)));

  //initial conditions
  N(0,0) = R0*exp(Rdev(0)-0.5*Rsd*Rsd);
  for(int age=1;age<Nages-1;age++) {
    N(0,age) = R0*exp(-M*age)*exp(Rinitdev(age-1)-0.5*Rsd*Rsd);;
  }
  N(0,Nages-1) = R0*exp(-M*(Nages-1))/(1-exp(-M));
  B(0)=0;
  for(int age=0;age<Nages;age++) B(0)+=N(0,age)*s(age)*Wage(age);

  //Dynamics
  for(int year=0;year<Nyears;year++) {
    N(year+1,0) = R0*exp(Rdev(year+1)-0.5*Rsd*Rsd); //recruitment
    for(int age=0;age<Nages-1;age++) {
      N(year+1,age+1) = N(year,age)*exp(-(M+F(year)*s(age)));
    }
    N(year+1,Nages-1) += N(year,Nages-1)*exp(-(M+F(year)*s(Nages-1)));
//Plus group
    B(year+1)=0;
    for(int age=0;age<Nages;age++)
B(year+1)+=N(year+1,age)*s(age)*Wage(age);
  }

  //Catch likelihood
  for(int year=0;year<Nyears;year++) {
    Cpred(year)=0;
    CAsum(year)=0;
    for(int age=0;age<Nages;age++) {
      CApred(year,age) =
(F(year)*s(age)/(F(year)*s(age)+M))*N(year,age)*(1-exp(-
(M+F(year)*s(age))));
      CAsum(year)+=CApred(year,age);
      Cpred(year)+=CApred(year,age)*Wage(age);
    }
    Clike(year) = -dnorm(log(C(year)),log(Cpred(year)),Csd,true);
  }

  //Index likelihood
  Ipred=B*q;
  for(int i=0;i<Nindex;i++) Ilike(i) = -
dnorm(Index(1,i),Ipred(CppAD::Integer(Index(0,i))-1),Index(2,i),true);

  //C@A likleihood
  for(int i=0;i<NCA;i++) {
    CAlike(i) = 0;
```

```
    for(int age=0;age<Nages;age++)
    {
      CApred(CppAD::Integer(CatchAge(0,i))-
1,age)/=CAsum(CppAD::Integer(CatchAge(0,i))-1);
      CAlike(i) -=
CatchAge(1,i)*CatchAge(age+2,i)*log(CApred(CppAD::Integer(CatchAge(0,i))-
1,age));
    }
  }

  //Recruitment penalty
  Rpen = -sum(dnorm(Rdev,0,Rsd,true))-sum(dnorm(Rinitdev,0,Rsd,true));

  Type ans = sum(Clike) + sum(Ilike) + sum(CAlike) + Rpen;

  ADREPORT(B);
  ADREPORT(F);
  vector<Type> R(Nyears+1);
  for(int y=0;y<Nyears+1;y++) R(y) = N(y,0);
  ADREPORT(R);

  REPORT(B);
  REPORT(R);
  REPORT(Ipred);
  REPORT(Cpred);
  REPORT(CApred);
  REPORT(N);
  REPORT(F);
  REPORT(CAsum);
  REPORT(s);
  REPORT(Ilike);
  REPORT(Clike);
  REPORT(CAlike);
  REPORT(Rpen);
  REPORT(Rdev);
  REPORT(Rinitdev);

  return ans;
}
```

3. Script and data to run the IPM for woodchat shrikes

Based on data and code published in Schaub & Kéry (2022, ISBN: 9780323908108).

The differences in the fitted model compared to IPM3 of chapter 11 in Schaub & Kéry (2022) are:

- ignore the fact that for some successful broods the number of fledglings is not known exactly
- survival of both age classes is constant over time
- the number of immigrants is constant over time
- the number of fledglings of successful broods is constant over time
- no goodness-of-fit testing performed

 # Set path and load libraries

```
path <- '...'
setwd(path)

library(IPMbook)
library(jagsUI)
library(scales)
```

 # Load data and prepare them according to the needs

```
data(woodchat11)     # Data file available in package 'IPMbook'

str(woodchat11)
# List of 6
# $ ch   : int [1:1079, 1:29] 0 0 0 0 0 0 0 0 0 0 ...
# $ age  : num [1:1079] 1 1 1 1 1 1 1 1 1 1 ...
# $ count: num [1:29] 10 13 25 15 17 16 6 16 7 7 ...
# $ f    : int [1:365] 5 0 6 0 3 6 6 3 0 3 ...
# $ d    : int [1:365] 0 0 0 0 0 0 0 0 0 1 ...
# $ year : int [1:365] 1964 1964 1964 1964 1964 1964 1964 ...


# 1. Convert capture-recapture data into an age-dependent m-array
marr <- marrayAge(woodchat11$ch, woodchat11$age)
rel.j <- rowSums(marr[,,1])
rel.a <- rowSums(marr[,,2])

recap <- c(1, 2, 3, 4, 5, 6, 7, 8, 19, 9, 10, 11, 12, 19, 19, 19, 19, 13,
14, 19, 19, 19, 19, 19, 15, 16, 17, 18)   # label of the recapture
occasions (years)

# 2. Productivity data
# 2.1. For nest success
z <- woodchat11$f
z[z>0] <- 1

# 2.2. For number of fledglings of successful nests
q <- woodchat11$f
q <- q[-which(z==0)]
q.year=woodchat11$year[-which(z==0)]-1963
```

 # Load data to be used in JAGS

```
# Bundle data and produce data overview
jags.data <- list(n.years=dim(marr)[2], marr.j=marr[,,1], marr.a=marr[,,2],
rel.j=rel.j, rel.a=rel.a, recap=recap, n.recap=max(recap), z=z,
z.year=woodchat11$year-1963, q=q, q.year=q.year, C=woodchat11$count,
pNinit=dUnif(1, 20))

str(jags.data)
List of 13
 $ n.years: int 29
 $ marr.j : num [1:28, 1:29] 1 0 0 0 0 0 0 0 0 0 ...
 $ marr.a : num [1:28, 1:29] 3 0 0 0 0 0 0 0 0 0 ...
 $ rel.j  : Named num [1:28] 27 32 26 32 55 26 22 32 0 25 ...
 $ rel.a  : Named num [1:28] 9 14 18 16 15 13 8 17 2 0 ...
```

```
 $ recap  : num [1:28] 1 2 3 4 5 6 7 8 19 9 ...
 $ n.recap: num 19
 $ z      : num [1:365] 1 0 1 0 1 1 1 1 0 1 ...
 $ z.year : num [1:365] 1 1 1 1 1 1 1 1 2 2 ...
 $ q      : int [1:274] 5 6 3 6 6 3 3 6 3 4 ...
 $ q.year : num [1:274] 1 1 1 1 1 1 2 2 2 2 ...
 $ C      : num [1:29] 10 13 25 15 17 16 6 16 7 7 ...
 $ pNinit : num [1:20] 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 ...
```

 # Write JAGS model file

```
cat(file="model1.txt", "
model {
  # Priors and linear models
  for (t in 1:(n.years-1)){
    phij[t] <- mean.phij
    phia[t] <- mean.phia
    pj[t] <- p.j[recap[t]]
    pa[t] <- p.a[recap[t]]
  }
  for (t in 1:n.years){
    logit.nu[t] ~ dnorm(l.mean.nu, tau.nu)
    nu[t] <- ilogit(logit.nu[t])
    omega[t] <- mean.omega
    kappa[t] <- mean.kappa
  }
  for (t in 1:(n.recap-1)){
    p.j[t] <- mean.p[1]
    p.a[t] <- mean.p[2]
  }
  # Fix at 0 recapture probability in years without capture/resighting
  p.j[n.recap] <- 0
  p.a[n.recap] <- 0
  mean.phij ~ dunif(0, 1)
  mean.phia ~ dunif(0, 1)
  mean.nu ~ dunif(0, 1)
  l.mean.nu <- logit(mean.nu)
  mean.kappa ~ dunif(1, 10)
  mean.omega ~ dunif(0.01, 30)
  for (i in 1:2){
    mean.p[i] ~ dunif(0, 1)
  }
  sigma.nu ~ dunif(0, 5)
  tau.nu <- pow(sigma.nu, -2)
  sigma.q ~ dunif(0.001, 5)
  tau.q <- pow(sigma.q, -2)

  # Population count data (state-space model)
  # Model for the initial population size: uniform priors
  R[1] ~ dcat(pNinit)                    # Local recruits
  S[1] ~ dcat(pNinit)                    # Surviving adults
  I[1] ~ dpois(omega[1])                 # Immigrants
  # Process model over time: our model of population dynamics
  for (t in 2:n.years){
    R[t] ~ dpois(nu[t-1] * kappa[t-1]/2 * phij[t-1] * N[t-1]) # Local
recruits
    S[t] ~ dbin(phia[t-1], N[t-1])       # Surviving adults
    I[t] ~ dpois(omega[t])               # Immigrants
  }
  # Observation model
```

```
  for (t in 1:n.years){
    N[t] <- S[t] + R[t] + I[t]    # Total number of breeding females
    C[t] ~ dpois(N[t])
  }

  # Productivity data
  # Bernoulli likelihood (nest success)
  for (i in 1:length(z)){
    z[i] ~ dbern(nu[z.year[i]])
  }
  # Normal likelihood (number of fledglings of successful broods)
  for (i in 1:length(q)){
    q[i] ~ dnorm(kappa[q.year[i]], tau.q)T(1,)
  }

  # Capture-recapture data (CJS model with multinomial likelihood)
  # Define the multinomial likelihood
  for (t in 1: (n.years-1)){
    marr.j[t,1:n.years] ~ dmulti(pr.j[t,], rel.j[t])
    marr.a[t,1:n.years] ~ dmulti(pr.a[t,], rel.a[t])
  }
  # Define the cell probabilities of the m-arrays
  for (t in 1:(n.years-1)){
    # Main diagonal
    qj[t] <- 1-pj[t]
    qa[t] <- 1-pa[t]
    pr.j[t,t] <- phij[t] * pj[t]
    pr.a[t,t] <- phia[t] * pa[t]
    # Above main diagonal
    for (j in (t+1):(n.years-1)){
      pr.j[t,j] <- phij[t] * prod(phia[(t+1):j]) * qj[t] * prod(qa[t:(j-
1)]) * pa[j] / qa[t]
      pr.a[t,j] <-prod(phia[t:j]) * prod(qa[t:(j-1)]) * pa[j]
    } #j
    # Below main diagonal
    for (j in 1:(t-1)){
      pr.j[t,j] <- 0
      pr.a[t,j] <- 0
    } #j
  } #t
  # Last column: probability of non-recapture
  for (t in 1:(n.years-1)){
    pr.j[t,n.years] <- 1-sum(pr.j[t,1:(n.years-1)])
    pr.a[t,n.years] <- 1-sum(pr.a[t,1:(n.years-1)])
  }
  # Compute derived parameters: total productivity and immigration rate
  for (t in 1:n.years){
    rho[t] <- nu[t] * kappa[t]
  }
  mean.rho <- mean.nu * mean.kappa
  for (t in 1:(n.years-1)){
    omega.rate[t] <- I[t+1] / N[t]
  }
}
")
```

 # Run model in JAGS

```
# Initial values
inits <- function() {list(mean.phia=runif(1))}
```

```
# Parameters monitored
parameters <- c("mean.phij", "mean.phia", "mean.nu", "mean.kappa",
"mean.rho", "mean.omega", "sigma.nu", "sigma.q", "mean.p", "phij", "phia",
"nu", "kappa", "omega", "rho", "omega.rate", "R", "S", "I", "N")

# MCMC settings
ni <- 30000; nb <- 10000; nc <- 3; nt <- 10; na <- 1000

# Call JAGS from R
out1 <- jags(jags.data, inits, parameters, "model1.txt", n.iter=ni,
n.burnin=nb, n.chains=nc, n.thin=nt, n.adapt=na, parallel=TRUE)
```

 # Inspect results

```
traceplot(out1)   # traceplot to check convergence

print(out1, 3)
```

```
JAGS output for model 'model1.txt', generated by jagsUI.
Estimates based on 3 chains of 30000 iterations,
adaptation = 1000 iterations (sufficient),
burn-in = 10000 iterations and thin rate = 10,
yielding 6000 total samples from the joint posterior.
MCMC ran in parallel for 1.428 minutes at time 2023-06-21 16:59:06.

                 mean     sd     2.5%      50%    97.5% overlap0 f  Rhat n.eff
mean.phij       0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
mean.phia       0.388  0.054    0.289    0.386    0.500    FALSE 1 1.001  1250
mean.nu         0.766  0.033    0.699    0.766    0.830    FALSE 1 1.001  6000
mean.kappa      4.287  0.086    4.117    4.288    4.451    FALSE 1 1.000  3765
mean.rho        3.283  0.155    2.975    3.285    3.589    FALSE 1 1.001  6000
mean.omega      7.621  1.134    5.445    7.627    9.846    FALSE 1 1.001  2670
sigma.nu        0.677  0.216    0.276    0.667    1.131    FALSE 1 1.004  1995
sigma.q         1.368  0.066    1.248    1.365    1.507    FALSE 1 1.000  6000
mean.p[1]       0.154  0.083    0.042    0.138    0.358    FALSE 1 1.000  6000
mean.p[2]       0.486  0.103    0.299    0.481    0.695    FALSE 1 1.003   883
phij[1]         0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[2]         0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[3]         0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[4]         0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[5]         0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[6]         0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[7]         0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[8]         0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[9]         0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[10]        0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[11]        0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[12]        0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[13]        0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[14]        0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[15]        0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[16]        0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[17]        0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[18]        0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[19]        0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[20]        0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[21]        0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[22]        0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[23]        0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[24]        0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[25]        0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[26]        0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[27]        0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phij[28]        0.057  0.022    0.024    0.054    0.110    FALSE 1 1.000  4486
phia[1]         0.388  0.054    0.289    0.386    0.500    FALSE 1 1.001  1250
phia[2]         0.388  0.054    0.289    0.386    0.500    FALSE 1 1.001  1250
phia[3]         0.388  0.054    0.289    0.386    0.500    FALSE 1 1.001  1250
phia[4]         0.388  0.054    0.289    0.386    0.500    FALSE 1 1.001  1250
phia[5]         0.388  0.054    0.289    0.386    0.500    FALSE 1 1.001  1250
```

phia[6]	0.388	0.054	0.289	0.386	0.500	FALSE	1	1.001	1250
phia[7]	0.388	0.054	0.289	0.386	0.500	FALSE	1	1.001	1250
phia[8]	0.388	0.054	0.289	0.386	0.500	FALSE	1	1.001	1250
phia[9]	0.388	0.054	0.289	0.386	0.500	FALSE	1	1.001	1250
phia[10]	0.388	0.054	0.289	0.386	0.500	FALSE	1	1.001	1250
phia[11]	0.388	0.054	0.289	0.386	0.500	FALSE	1	1.001	1250
phia[12]	0.388	0.054	0.289	0.386	0.500	FALSE	1	1.001	1250
phia[13]	0.388	0.054	0.289	0.386	0.500	FALSE	1	1.001	1250
phia[14]	0.388	0.054	0.289	0.386	0.500	FALSE	1	1.001	1250
phia[15]	0.388	0.054	0.289	0.386	0.500	FALSE	1	1.001	1250
phia[16]	0.388	0.054	0.289	0.386	0.500	FALSE	1	1.001	1250
phia[17]	0.388	0.054	0.289	0.386	0.500	FALSE	1	1.001	1250
phia[18]	0.388	0.054	0.289	0.386	0.500	FALSE	1	1.001	1250
phia[19]	0.388	0.054	0.289	0.386	0.500	FALSE	1	1.001	1250
phia[20]	0.388	0.054	0.289	0.386	0.500	FALSE	1	1.001	1250
phia[21]	0.388	0.054	0.289	0.386	0.500	FALSE	1	1.001	1250
phia[22]	0.388	0.054	0.289	0.386	0.500	FALSE	1	1.001	1250
phia[23]	0.388	0.054	0.289	0.386	0.500	FALSE	1	1.001	1250
phia[24]	0.388	0.054	0.289	0.386	0.500	FALSE	1	1.001	1250
phia[25]	0.388	0.054	0.289	0.386	0.500	FALSE	1	1.001	1250
phia[26]	0.388	0.054	0.289	0.386	0.500	FALSE	1	1.001	1250
phia[27]	0.388	0.054	0.289	0.386	0.500	FALSE	1	1.001	1250
phia[28]	0.388	0.054	0.289	0.386	0.500	FALSE	1	1.001	1250
nu[1]	0.751	0.094	0.535	0.761	0.908	FALSE	1	1.000	5379
nu[2]	0.766	0.082	0.584	0.774	0.903	FALSE	1	1.000	6000
nu[3]	0.531	0.104	0.322	0.533	0.725	FALSE	1	1.001	1616
nu[4]	0.699	0.092	0.494	0.708	0.854	FALSE	1	1.000	6000
nu[5]	0.760	0.078	0.587	0.766	0.893	FALSE	1	1.000	3548
nu[6]	0.780	0.084	0.592	0.788	0.921	FALSE	1	1.000	6000
nu[7]	0.766	0.098	0.539	0.777	0.925	FALSE	1	1.000	4364
nu[8]	0.681	0.091	0.482	0.688	0.836	FALSE	1	1.000	6000
nu[9]	0.510	0.148	0.217	0.518	0.762	FALSE	1	1.000	3083
nu[10]	0.792	0.088	0.590	0.799	0.934	FALSE	1	1.000	6000
nu[11]	0.830	0.072	0.672	0.838	0.949	FALSE	1	1.000	6000
nu[12]	0.604	0.110	0.368	0.612	0.789	FALSE	1	1.000	4068
nu[13]	0.730	0.077	0.561	0.738	0.863	FALSE	1	1.000	6000
nu[14]	0.778	0.072	0.621	0.783	0.901	FALSE	1	1.000	6000
nu[15]	0.651	0.108	0.414	0.662	0.829	FALSE	1	1.000	6000
nu[16]	0.780	0.093	0.572	0.788	0.934	FALSE	1	1.000	5325
nu[17]	0.807	0.081	0.628	0.814	0.937	FALSE	1	1.000	5223
nu[18]	0.764	0.081	0.583	0.772	0.899	FALSE	1	1.000	6000
nu[19]	0.838	0.070	0.684	0.846	0.950	FALSE	1	1.000	4822
nu[20]	0.839	0.062	0.704	0.844	0.944	FALSE	1	1.000	4935
nu[21]	0.793	0.079	0.615	0.801	0.924	FALSE	1	1.001	1704
nu[22]	0.771	0.079	0.600	0.777	0.908	FALSE	1	1.000	6000
nu[23]	0.855	0.072	0.693	0.862	0.966	FALSE	1	1.001	2505
nu[24]	0.753	0.086	0.561	0.760	0.897	FALSE	1	1.000	6000
nu[25]	0.806	0.074	0.641	0.813	0.930	FALSE	1	1.000	6000
nu[26]	0.805	0.074	0.647	0.812	0.928	FALSE	1	1.000	6000
nu[27]	0.759	0.083	0.573	0.766	0.899	FALSE	1	1.000	6000
nu[28]	0.763	0.090	0.558	0.771	0.911	FALSE	1	1.000	6000
nu[29]	0.814	0.079	0.644	0.821	0.943	FALSE	1	1.001	3764
kappa[1]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[2]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[3]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[4]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[5]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[6]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[7]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[8]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[9]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[10]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[11]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[12]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[13]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[14]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[15]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[16]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[17]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[18]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[19]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[20]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[21]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[22]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[23]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[24]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[25]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[26]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765

kappa[27]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[28]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
kappa[29]	4.287	0.086	4.117	4.288	4.451	FALSE	1	1.000	3765
omega[1]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[2]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[3]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[4]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[5]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[6]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[7]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[8]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[9]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[10]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[11]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[12]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[13]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[14]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[15]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[16]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[17]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[18]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[19]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[20]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[21]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[22]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[23]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[24]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[25]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[26]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[27]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[28]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
omega[29]	7.621	1.134	5.445	7.627	9.846	FALSE	1	1.001	2670
rho[1]	3.221	0.410	2.282	3.258	3.901	FALSE	1	1.000	4578
rho[2]	3.285	0.355	2.503	3.318	3.896	FALSE	1	1.000	6000
rho[3]	2.278	0.448	1.387	2.285	3.130	FALSE	1	1.001	1600
rho[4]	2.996	0.398	2.117	3.033	3.670	FALSE	1	1.000	6000
rho[5]	3.259	0.338	2.514	3.280	3.848	FALSE	1	1.001	2801
rho[6]	3.343	0.367	2.534	3.372	3.963	FALSE	1	1.000	6000
rho[7]	3.285	0.426	2.296	3.330	3.987	FALSE	1	1.000	6000
rho[8]	2.918	0.397	2.054	2.950	3.613	FALSE	1	1.000	6000
rho[9]	2.187	0.633	0.924	2.220	3.267	FALSE	1	1.000	3108
rho[10]	3.393	0.383	2.521	3.427	4.020	FALSE	1	1.000	4267
rho[11]	3.559	0.319	2.860	3.588	4.095	FALSE	1	1.000	6000
rho[12]	2.590	0.473	1.578	2.623	3.400	FALSE	1	1.000	4470
rho[13]	3.131	0.336	2.406	3.159	3.715	FALSE	1	1.000	6000
rho[14]	3.334	0.314	2.652	3.355	3.880	FALSE	1	1.000	6000
rho[15]	2.791	0.464	1.767	2.834	3.576	FALSE	1	1.000	5728
rho[16]	3.343	0.404	2.434	3.380	4.024	FALSE	1	1.000	6000
rho[17]	3.460	0.354	2.679	3.490	4.048	FALSE	1	1.000	5998
rho[18]	3.277	0.355	2.499	3.307	3.882	FALSE	1	1.000	6000
rho[19]	3.594	0.309	2.917	3.627	4.106	FALSE	1	1.000	3818
rho[20]	3.595	0.277	3.012	3.613	4.076	FALSE	1	1.000	6000
rho[21]	3.398	0.347	2.630	3.430	3.990	FALSE	1	1.001	1857
rho[22]	3.305	0.346	2.556	3.332	3.910	FALSE	1	1.001	5649
rho[23]	3.664	0.319	2.962	3.695	4.177	FALSE	1	1.001	2293
rho[24]	3.227	0.372	2.408	3.254	3.860	FALSE	1	1.000	6000
rho[25]	3.457	0.325	2.746	3.484	4.004	FALSE	1	1.000	6000
rho[26]	3.451	0.324	2.764	3.474	3.993	FALSE	1	1.000	6000
rho[27]	3.254	0.360	2.451	3.289	3.872	FALSE	1	1.000	6000
rho[28]	3.270	0.391	2.390	3.304	3.942	FALSE	1	1.000	6000
rho[29]	3.488	0.346	2.747	3.523	4.071	FALSE	1	1.000	5551
omega.rate[1]	0.612	0.270	0.211	0.571	1.273	FALSE	1	1.000	6000
omega.rate[2]	0.722	0.268	0.286	0.692	1.333	FALSE	1	1.000	4703
omega.rate[3]	0.398	0.155	0.143	0.381	0.750	FALSE	1	1.001	5830
omega.rate[4]	0.525	0.210	0.188	0.500	1.000	FALSE	1	1.001	2679
omega.rate[5]	0.472	0.190	0.167	0.444	0.917	FALSE	1	1.000	6000
omega.rate[6]	0.368	0.170	0.100	0.353	0.750	FALSE	1	1.000	4768
omega.rate[7]	0.824	0.373	0.286	0.769	1.714	FALSE	1	1.000	6000
omega.rate[8]	0.403	0.184	0.111	0.385	0.818	FALSE	1	1.000	6000
omega.rate[9]	0.655	0.306	0.200	0.600	1.375	FALSE	1	1.003	1253
omega.rate[10]	0.927	0.387	0.333	0.875	1.833	FALSE	1	1.000	6000
omega.rate[11]	0.809	0.294	0.333	0.769	1.500	FALSE	1	1.000	6000
omega.rate[12]	0.478	0.171	0.200	0.467	0.857	FALSE	1	1.000	6000
omega.rate[13]	0.551	0.195	0.227	0.526	1.000	FALSE	1	1.000	6000
omega.rate[14]	0.385	0.146	0.138	0.375	0.722	FALSE	1	1.000	4543
omega.rate[15]	0.288	0.126	0.083	0.273	0.562	FALSE	1	1.000	6000
omega.rate[16]	0.582	0.256	0.188	0.545	1.182	FALSE	1	1.000	6000
omega.rate[17]	0.635	0.257	0.222	0.600	1.222	FALSE	1	1.001	6000

omega.rate[18]	0.633	0.246	0.235	0.600	1.200	FALSE	1	1.000	6000
omega.rate[19]	0.537	0.206	0.210	0.500	1.000	FALSE	1	1.001	2005
omega.rate[20]	0.380	0.162	0.111	0.364	0.750	FALSE	1	1.001	2762
omega.rate[21]	0.540	0.228	0.188	0.500	1.083	FALSE	1	1.001	6000
omega.rate[22]	0.496	0.210	0.167	0.467	1.000	FALSE	1	1.002	2840
omega.rate[23]	0.613	0.260	0.200	0.583	1.222	FALSE	1	1.000	6000
omega.rate[24]	0.585	0.244	0.200	0.545	1.143	FALSE	1	1.000	6000
omega.rate[25]	0.539	0.220	0.188	0.500	1.000	FALSE	1	1.000	6000
omega.rate[26]	0.513	0.216	0.167	0.500	1.000	FALSE	1	1.001	3832
omega.rate[27]	0.473	0.209	0.143	0.455	0.917	FALSE	1	1.000	4602
omega.rate[28]	0.624	0.280	0.200	0.583	1.286	FALSE	1	1.000	6000
R[1]	3.945	2.771	1.000	3.000	11.000	FALSE	1	1.001	2491
R[2]	1.314	1.276	0.000	1.000	4.000	TRUE	1	1.001	3658
R[3]	1.895	1.548	0.000	2.000	6.000	TRUE	1	1.000	6000
R[4]	1.218	1.192	0.000	1.000	4.000	TRUE	1	1.001	6000
R[5]	1.433	1.320	0.000	1.000	5.000	TRUE	1	1.000	6000
R[6]	1.382	1.270	0.000	1.000	4.000	TRUE	1	1.001	4368
R[7]	0.936	1.019	0.000	1.000	3.000	TRUE	1	1.000	6000
R[8]	1.052	1.080	0.000	1.000	4.000	TRUE	1	1.000	3234
R[9]	0.778	0.920	0.000	1.000	3.000	TRUE	1	1.000	6000
R[10]	0.508	0.754	0.000	0.000	2.000	TRUE	1	1.000	6000
R[11]	1.123	1.134	0.000	1.000	4.000	TRUE	1	1.000	6000
R[12]	2.086	1.627	0.000	2.000	6.000	TRUE	1	1.001	3424
R[13]	1.800	1.523	0.000	2.000	5.000	TRUE	1	1.001	2966
R[14]	2.418	1.788	0.000	2.000	7.000	TRUE	1	1.000	6000
R[15]	2.300	1.706	0.000	2.000	6.000	TRUE	1	1.000	6000
R[16]	1.129	1.144	0.000	1.000	4.000	TRUE	1	1.001	1724
R[17]	1.155	1.186	0.000	1.000	4.000	TRUE	1	1.000	6000
R[18]	1.352	1.269	0.000	1.000	4.000	TRUE	1	1.000	4500
R[19]	1.596	1.395	0.000	1.000	5.000	TRUE	1	1.001	2130
R[20]	1.933	1.516	0.000	2.000	5.000	TRUE	1	1.000	6000
R[21]	1.525	1.309	0.000	1.000	5.000	TRUE	1	1.000	6000
R[22]	1.299	1.262	0.000	1.000	4.000	TRUE	1	1.002	2045
R[23]	1.118	1.147	0.000	1.000	4.000	TRUE	1	1.000	2874
R[24]	1.286	1.243	0.000	1.000	4.000	TRUE	1	1.001	6000
R[25]	1.213	1.180	0.000	1.000	4.000	TRUE	1	1.000	6000
R[26]	1.338	1.234	0.000	1.000	4.000	TRUE	1	1.001	3600
R[27]	1.236	1.219	0.000	1.000	4.000	TRUE	1	1.000	6000
R[28]	0.948	1.038	0.000	1.000	3.000	TRUE	1	1.001	4668
R[29]	0.919	1.011	0.000	1.000	3.000	TRUE	1	1.000	6000
S[1]	3.976	2.817	1.000	3.000	11.000	FALSE	1	1.000	6000
S[2]	5.596	2.173	2.000	5.000	10.000	FALSE	1	1.000	6000
S[3]	6.895	2.259	3.000	7.000	11.000	FALSE	1	1.000	6000
S[4]	7.357	2.298	3.000	7.000	12.000	FALSE	1	1.000	6000
S[5]	6.428	2.203	2.000	6.000	11.000	FALSE	1	1.001	3359
S[6]	5.989	2.106	2.000	6.000	10.000	FALSE	1	1.000	6000
S[7]	4.437	1.896	1.000	4.000	8.000	FALSE	1	1.001	2829
S[8]	4.280	1.877	1.000	4.000	8.000	FALSE	1	1.000	6000
S[9]	4.088	1.772	1.000	4.000	8.000	FALSE	1	1.001	2444
S[10]	3.445	1.660	1.000	3.000	7.000	FALSE	1	1.001	2318
S[11]	4.280	1.847	1.000	4.000	8.000	FALSE	1	1.000	6000
S[12]	6.790	2.228	3.000	7.000	11.000	FALSE	1	1.000	6000
S[13]	8.636	2.549	4.000	9.000	14.000	FALSE	1	1.000	6000
S[14]	9.241	2.566	4.000	9.000	14.000	FALSE	1	1.000	6000
S[15]	9.099	2.525	4.000	9.000	14.000	FALSE	1	1.000	5877
S[16]	6.270	2.222	2.000	6.000	11.000	FALSE	1	1.001	2779
S[17]	4.900	1.986	1.000	5.000	9.000	FALSE	1	1.000	6000
S[18]	5.274	2.007	2.000	5.000	9.000	FALSE	1	1.000	6000
S[19]	6.243	2.145	2.000	6.000	11.000	FALSE	1	1.000	3749
S[20]	7.017	2.232	3.000	7.000	11.000	FALSE	1	1.001	1841
S[21]	6.168	2.170	2.000	6.000	11.000	FALSE	1	1.001	2460
S[22]	5.372	2.033	2.000	5.000	10.000	FALSE	1	1.000	6000
S[23]	4.984	1.965	1.000	5.000	9.000	FALSE	1	1.001	3043
S[24]	4.899	1.975	1.000	5.000	9.000	FALSE	1	1.000	6000
S[25]	5.277	1.996	2.000	5.000	9.000	FALSE	1	1.000	6000
S[26]	5.380	2.010	2.000	5.000	10.000	FALSE	1	1.002	1207
S[27]	5.085	1.979	2.000	5.000	9.000	FALSE	1	1.001	1825
S[28]	4.416	1.835	1.000	4.000	8.000	FALSE	1	1.002	942
S[29]	4.122	1.785	1.000	4.000	8.000	FALSE	1	1.000	6000
I[1]	6.047	2.337	2.000	6.000	11.000	FALSE	1	1.000	6000
I[2]	7.994	2.573	3.000	8.000	13.000	FALSE	1	1.000	3845
I[3]	10.292	2.895	5.000	10.000	16.000	FALSE	1	1.000	6000
I[4]	7.403	2.559	3.000	7.000	13.000	FALSE	1	1.000	6000
I[5]	8.072	2.638	3.000	8.000	13.000	FALSE	1	1.001	1699
I[6]	7.255	2.476	3.000	7.000	13.000	FALSE	1	1.000	6000
I[7]	5.189	2.127	2.000	5.000	10.000	FALSE	1	1.000	5809
I[8]	8.071	2.504	4.000	8.000	13.000	FALSE	1	1.000	6000
I[9]	5.206	2.092	2.000	5.000	10.000	FALSE	1	1.000	6000

```
I[10]            6.161  2.171    2.000    6.000   11.000    FALSE 1 1.001  1923
I[11]            8.758  2.543    4.000    9.000   14.000    FALSE 1 1.000  6000
I[12]           10.973  2.957    5.000   11.000   17.000    FALSE 1 1.000  6000
I[13]            9.233  2.808    4.000    9.000   15.000    FALSE 1 1.000  6000
I[14]           10.510  3.038    5.000   10.000   17.000    FALSE 1 1.000  6000
I[15]            8.329  2.773    3.000    8.000   14.000    FALSE 1 1.001  2833
I[16]            5.551  2.228    2.000    5.000   10.000    FALSE 1 1.000  6000
I[17]            7.124  2.429    3.000    7.000   12.000    FALSE 1 1.000  6000
I[18]            7.978  2.530    3.000    8.000   13.000    FALSE 1 1.000  6000
I[19]            8.868  2.723    4.000    9.000   14.000    FALSE 1 1.000  6000
I[20]            8.670  2.729    4.000    9.000   14.000    FALSE 1 1.001  1784
I[21]            6.485  2.398    2.000    6.000   11.025    FALSE 1 1.001  1880
I[22]            7.320  2.475    3.000    7.000   13.000    FALSE 1 1.000  6000
I[23]            6.650  2.348    3.000    7.000   12.000    FALSE 1 1.002  1722
I[24]            7.442  2.527    3.000    7.000   13.000    FALSE 1 1.000  4122
I[25]            7.604  2.494    3.000    7.000   13.000    FALSE 1 1.000  6000
I[26]            7.288  2.455    3.000    7.000   13.000    FALSE 1 1.000  4180
I[27]            6.882  2.389    3.000    7.000   12.000    FALSE 1 1.000  6000
I[28]            5.981  2.220    2.000    6.000   11.000    FALSE 1 1.000  4062
I[29]            6.697  2.373    3.000    7.000   12.000    FALSE 1 1.000  6000
N[1]            13.967  3.253    8.000   14.000   21.000    FALSE 1 1.000  6000
N[2]            14.904  2.733   10.000   15.000   21.000    FALSE 1 1.000  6000
N[3]            19.082  2.769   14.000   19.000   25.000    FALSE 1 1.000  4293
N[4]            15.978  2.753   11.000   16.000   22.000    FALSE 1 1.000  6000
N[5]            15.932  2.688   11.000   16.000   22.000    FALSE 1 1.000  6000
N[6]            14.627  2.564   10.000   15.000   20.000    FALSE 1 1.000  6000
N[7]            10.562  2.479    6.000   10.000   16.000    FALSE 1 1.000  6000
N[8]            13.404  2.415    9.000   13.000   18.000    FALSE 1 1.000  5665
N[9]            10.072  2.357    6.000   10.000   15.000    FALSE 1 1.000  6000
N[10]           10.114  2.354    6.000   10.000   15.000    FALSE 1 1.000  3315
N[11]           14.162  2.577    9.000   14.000   20.000    FALSE 1 1.000  6000
N[12]           19.849  2.856   15.000   20.000   26.000    FALSE 1 1.000  6000
N[13]           19.669  2.906   14.000   20.000   26.000    FALSE 1 1.000  5361
N[14]           22.169  3.039   17.000   22.000   28.000    FALSE 1 1.000  6000
N[15]           19.728  2.905   14.000   20.000   26.000    FALSE 1 1.000  6000
N[16]           12.950  2.660    8.000   13.000   18.000    FALSE 1 1.000  6000
N[17]           13.178  2.507    9.000   13.000   18.000    FALSE 1 1.000  6000
N[18]           14.603  2.594   10.000   15.000   20.000    FALSE 1 1.000  6000
N[19]           16.708  2.705   12.000   17.000   22.000    FALSE 1 1.000  6000
N[20]           17.621  2.759   13.000   18.000   23.000    FALSE 1 1.000  6000
N[21]           14.178  2.622    9.000   14.000   19.000    FALSE 1 1.000  6000
N[22]           13.992  2.550    9.000   14.000   19.000    FALSE 1 1.000  6000
N[23]           12.753  2.541    8.000   13.000   18.000    FALSE 1 1.001  2775
N[24]           13.627  2.613    9.000   13.000   19.000    FALSE 1 1.001  2875
N[25]           14.094  2.548   10.000   14.000   19.000    FALSE 1 1.000  6000
N[26]           14.006  2.555    9.000   14.000   19.000    FALSE 1 1.001  4239
N[27]           13.203  2.498    9.000   13.000   18.000    FALSE 1 1.000  6000
N[28]           11.346  2.425    7.000   11.000   17.000    FALSE 1 1.000  6000
N[29]           11.738  2.513    7.000   12.000   17.000    FALSE 1 1.000  6000
deviance      1604.876 10.912 1585.675 1604.144 1628.509    FALSE 1 1.002  1274
```

 # Produce figure 4

```
co <- alpha(viridis_pal(option='E')(20)[c(1,11,18)], c(0.8, 0.8, 1))

png("Fig.4.png", width = 1*1500, height = 2*1000, res=300, type='cairo')

lw <- 1.5
cex.tif <- 0.9

layout(matrix(1:2,2,1,byrow = TRUE), widths = 1.5, heights = c(1, 1), TRUE)
par(las  = 1, cex = cex.tif, mar=c(3,4,2,2))

time <- c(1965, 1970, 1975, 1980, 1985, 1990)
plot(out1$mean$N, type="l", ylim=c(0, 30), ylab="Number of females",
xlab=NA, axes=FALSE)
polygon(x=c(1:29, 29:1), y=c(out1$q2.5$N, rev(out1$q97.5$N)), border=NA,
col=alpha('grey', 95))
lines(out1$mean$N, type="l", lwd=2)
```

```
points(woodchat11$count, pch=16, col='red', type='p', cex=0.8)
axis(1, at=1:29, tcl=-0.25, label=NA)
axis(1, at=c(2, 7, 12, 17, 22, 27), label=time)
axis(2, las=1)
legend("topright", pch = c(NA,16), lty=c(1,NA), lwd=c(2, NA), legend =
c('Estimate', 'Count'), col = c('black', 'red'), bty = 'n')

comp <- rbind(out1$mean$S, out1$mean$R, out1$mean$I)
a <- barplot(comp, col=co, axes=F, ylab='Number of females', ylim=c(0, 30))
legend("topright", pch = rep(15,3), legend = c('Immigrant', 'Local
recruit', 'Surviving adult'), col = rev(co), bty = 'n')
axis(1, at=a, tcl=-0.25, label=NA)
axis(1, at=a[c(2, 7, 12, 17, 22, 27)], label=time)
axis(2, las=1)

dev.off()
```