ANEXO I: CÓDIGO DESARROLLADO PARA LOS ANÁLISIS DE ESTE INFORME

Se detallan a continuación los procedimientos utilizados en los análisis del informe "REVISIÓN DE LOS PROCEDIMIENTOS UTILIZADOS PARA EVALUAR LA DINÁMICA DE LA BIOMASA DE LA VIEIRA PATAGÓNICA EN LA UNIDAD DE MANEJO B"

Estimación de la eficiencia por el modelo no-lineal

```
library(LaplacesDemon)
library(openxlsx)

lances=read.xlsx(xlsxFile="datos.xlsx")
A = 0.10128809

obs=lances$N
ln=length(obs)
bar1=which(lances$Barrido==1)
bar2=which(lances$Barrido==2)
CT=sum(obs)

mon.names <- "LP"

parm=c(log_Ni=log(CT*1.2), log_sd1=0, log_sd2=0)

lo=c(log_Ni=log(CT), log_sd1=-Inf, log_sd1=-Inf)
up=c(log_Ni=log(CT)*2, log_sd1=10, log_sd1=10)

MyData <- list(mon.names=mon.names, parm.names=names(parm), n=ln)

Model <- function(parm, Data)
{
  parm=pmin(parm, up)
  parm=pmax(parm, lo)

  Ni=exp(parm["log_Ni"])
  sd1=exp(parm["log_sd1"])
  sd2=exp(parm["log_sd2"])
  p=(1-CT/Ni)^(1/ln)
  espe <<- pmax(Ni*(1-p)*p^(0:(ln-1)), 1)

  slk1 = dnorm(log(obs[bar1]), log(espe[bar1]) - (sd1^2)/2, sd1, log = TRUE)
  slk2 = dnorm(log(obs[bar2]), log(espe[bar2]) - (sd2^2)/2, sd2, log = TRUE)
  LP = sum(slk1, slk2)

  Modelout <- list(LP=LP, Dev=-2*LP, Monitor=LP, yhat=espe, parm=parm)
  return(Modelout)
}

parm=c(log_Ni=log(CT)*1.2, log_sd1=-3, log_sd2=-3)
parm[]=rnorm(3, parm[], abs(parm[]*0.1))

FitLA <- LaplaceApproximation(Model=Model, parm=parm, Data=MyData, Iterations=30000, Samples=3000)

Posterior=FitLA$Posterior

N.sir=exp(Posterior[,'log_Ni'])
p.sir=(1-CT/N.sir)^(1/ln)
e.sir=(1-p.sir)*A/mean(lances$Area, na.rm=TRUE)
```

Estimación de la eficiencia por el modelo de superposición de lances

```
library(raster)
library(rgdal)
library(rgeos)
library(sf)

ll =  "+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"
utm = "+proj=utm +zone=21 +south +datum=WGS84 +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0"

j=1
po = SpatialPoints(data.frame(xo=lances$Lon0[j], yo=lances$Lat0[j]))
pf = SpatialPoints(data.frame(xf=lances$Lon1[j], yf=lances$Lat1[j]))
Lines = as(rbind(po,pf), "SpatialLines")
```

```
for(j in 2:nrow(lances)){
  po = SpatialPoints(data.frame(xo=lances$Lon0[j], yo=lances$Lat0[j]))
  pf = SpatialPoints(data.frame(xf=lances$Lon1[j], yf=lances$Lat1[j]))
  Lines = rbind(Lines, as(rbind(po,pf), "SpatialLines"))
}

proj4string(Lines) = ll
LinesUTM= spTransform(Lines, utm)

LinesUTM$ID=lances$Lance

PolyLances = buffer(LinesUTM,  width=2.5/2, dissolve=F)

PropBarridosRaster = function(Inter){
R=raster::raster(res=c(0.1, 0.1), ext=extent(Inter), crs=proj4string(Inter))
Q = fasterize::fasterize(sf=st_as_sf(Inter), raster=R, field="Value",  fun='sum')
return(table(Q[])/length(which(Q[]>0)))
}

TablaR = list()

for(l in 1:length(PolyLances)){
    Inter = raster::intersect(PolyLances[l,], PolyLances[1:l,])
    Inter$Value=rep(1, length(Inter))
    TablaR[[l]] = PropBarridosRaster(Inter)
  }

TablaP = matrix(0, length(TablaR), length(TablaR))

  for(k in 1:length(TablaR)){
    cols=as.numeric(names(TablaR[[k]]))
    TablaP[k,cols]=TablaR[[k]]
  }

Capturas = lances$N
area = area(PolyLances)

expo=0:(length(Capturas)-1)

parm = c(log_d=1, log_e=-1, log_s=1)
parm[]=parm+rnorm(3,0,0.2)

ib1=which(lances$Barrido <3)
ib2=which(lances$Barrido >=3)

Ajuste=function(parm){
    d=exp(parm["log_d"])
    e=exp(parm["log_e"])
    s1=exp(parm["log_s1"])
    s2=exp(parm["log_s2"])
    p.d=dunif(d, 0, 50, log=TRUE)
    p.e=dbeta(e, 1,1, log=TRUE)
    p.s=sum(dunif(c(s1,s2), 0, 50, log=TRUE))

    A = matrix(d * (1-e)^(expo), length(Capturas), 1)
    espe <-  e * area * apply(TablaP %*%  A, 1, sum)
    SLL=-sum(p.d,p.e, p.s)-sum(dnorm(x=log(Capturas[ib1]), log(espe[ib1])-(s1^2)/2, s1, log = TRUE))
         -
         sum(dnorm(x=log(Capturas[ib2]), log(espe[ib2])-(s2^2)/2, s2, log = TRUE))
    return(SLL)
}

mon.names <- paste0("PosteriorLike")

MyData <- list(mon.names=mon.names, parm.names=names(parm), n=nrow(lances), y=Capturas)

Model=function(parm, Data){
  LP=-Ajuste(parm)
  C.hat=espe
  Modelout <- list(LP=LP, Dev=-2*(LP), Monitor=LP, yhat=C.hat, parm=parm)
  return(Modelout)
}

FitLA <-  LaplaceApproximation(Model, parm=parm, Data=MyData, Iterations=10000, Samples=3000)
```

```
Posterior=FitLA$Posterior
```

Código utilizado para estimar parámetros del modelo 3enor

```
library(Rcpp)

cppFunction('NumericVector ProjectNo(double No, NumericVector theta,
                                     NumericVector Ct){
  int n = Ct.length();
  NumericVector out(n);
  out[0]=No;
  for (int k = 1; k < n; k++) {
  out[k] =  theta[k]*(out[k-1]- Ct[k]);
 }
 return pmax(1, out);
  }')

Years = 2012:2024
C2023 = 25000*(1000*1000)/25
Ct = c(NA, 139200000,0,0,769320000,1034800000,396760000,92000000,41920000,0,1113760000, C2023,NA)
Nobs = c(NA, 10572897069, 10601188868, 8931773092, 8236083348, 7833727943, 5232205495, 4448336556,
    3529895557, 8343037322, 6215955922, NA, NA)

indi=which(Years %in% 2013:2022 == TRUE)

mon.names <- c("B2021", "B2023", "B2024")

MyData <- list(mon.names=mon.names, parm.names=names(lo), n=n, y=Nobs[indi])

Model <- function(parm, Data)
{
  prior = sum(dnorm(parm, 0, 50, log=TRUE))
  No=exp(parm[1])
  sigma=exp(parm[5])
  theta=c(rep(exp(parm[2]),5), rep(exp(parm[3]),3), rep(exp(parm[4]),5))

  Nexp=ProjectNo(No=No, Ct=Ct, theta=theta)

  LL=sum(dlnorm(x=Nobs[indi], log(Nexp[indi]), sigma, log=TRUE))
  LP=LL+prior
  B2021=theta[9]*(Nexp[8]-Ct[8])*25/(1000*1000)
  B2023=Nexp[n-1]*25/(1000*1000)
  B2024=theta[n]*(Nexp[(n-1)]-C2023)*25/(1000*1000)
  Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(B2021, B2023, B2024), yhat=Nexp[indi], parm=parm)
  return(Modelout)
}

start[]= c(rnorm(1, log(max(Nobs, na.rm=T)*0.75), 2), rnorm(3, 0, 1), rnorm(1, 1, 1))

FitLA = LaplaceApproximation(Model, parm=start, Data=MyData, Iterations=300000, Samples=10000)

par(mfrow=c(5,2), mar=c(3,3,1,1), mgp=c(1.5,0.5,0))

Posterior=FitLA$Posterior
```

Código Stan utilizado para estimar los parámetros del modelo discreto con retardo y crecimiento

```
data {
  int N;  // numero de años
  int Nina; // numero de años con CPUE
  int Ext; // numero de años proyección
  vector[N] I;    // Indice VC prospecciones
  vector[N] IR;    // Indice VR prospecciones
  vector[N] C; // Capturas flota
  real Cfut; // Capturas futuras
  vector[Nina] CPUE; // CPUE flota
  real g; // potencial crecimiento VC
  real gR; // potencial crecimiento VR
  real m; // mortalidad natural
  int INA[Nina]; // datos validos en CPUE
}

parameters {
```

```
  real<lower=0,upper=1> qI; // conversion VC prospección
  real<lower=0,upper=1> qR; // conversion VR prospección
  real<lower=0,upper=1> qF; // conversion VC flota
  real<lower=0> sigma_tau; // error proceso biomasa VC
  real<lower=0> sigma_phi; // error proceso biomasa VR
  real<lower=0> sigma_epsilon; // error obs VC prospección
  real<lower=0> sigma_upsilon; // error obs VR prospección
  real<lower=0> sigma_C; // error obs VC flota
  real<lower=0,upper=500000> B[N+Ext]; // Biomasa VC
  real<lower=0,upper=400000> R[N+Ext]; // Biomasa VR
}

transformed parameters {
  vector[N+Ext] Bio;
  vector[N] Imed;
  vector[N] IRmed;
  vector[Nina] CPUEmed;
  Bio[1] = B[1];

  for(t in 1:N){
    Imed[t] =  fmax(qI * B[t], 0.000001);
    IRmed[t] = fmax(qR * R[t], 0.000001);
  }

  for(t in 1:Nina){
  CPUEmed[t] = fmax(qF * B[INA[t]], 0.000001);
  }

  for(t in 2:N){
  Bio[t] = fmax(exp(-m) * g * (B[t - 1] - C[t - 1]) +
                        exp(-m) * gR * R[t - 1], 0.000001);
  }

  for(tf in (N+1):(N+Ext)){
  Bio[tf] = fmax(exp(-m) * g * (B[tf - 1] - Cfut) +
                        exp(-m) * gR * R[tf - 1], 0.000001);
  }

}

model {
  // priors
  qI ~ beta(1,1);
  qR ~ beta(1,1);
  qF  ~ beta(1,1);
  sigma_tau ~ exponential(1);
  sigma_phi ~ exponential(1);
  sigma_epsilon ~ exponential(1);
  sigma_upsilon ~ exponential(1);
  sigma_C ~ exponential(1);
  R ~ lognormal(8 , sigma_phi);
  B[1] ~ lognormal(12 , sigma_tau);
////////////////////////////////////////////////////////////

  for (t in 2:N) {
  B[t] ~  lognormal(log(Bio[t]), sigma_tau);
  }

  for(tf in (N+1):(N+Ext)){
  B[tf] ~  lognormal(log(Bio[tf]), sigma_tau);
  }

  for (t in 1:N) {
    I[t] ~ lognormal(log(Imed[t]), sigma_epsilon);
    IR[t] ~ lognormal(log(IRmed[t]), sigma_upsilon);
  }

  for(t in 1:Nina){
  CPUE[t] ~ lognormal(log(CPUEmed[t]), sigma_C);
  }
}
"
```

Preparación de datos y estimación en la interfase R para correr el código Stan

```
library(rstan)
library(readxl)

datos=read_xlsx("datosUMB.xlsx", sheet="datosModelo")
INA=which(!is.na(datos$CPUE))
CPUE=as.vector(na.omit(datos$CPUE))

dat_list <-list(N=nrow(datos), Nina=length(INA), I=datos$I, IR=datos$IR, C=datos$C, Cfut=25000, CPUE
    =CPUE, g=1.46, gR=1.1, m=0.3, INA=INA, Ext=3)

nthin=100

nsamples=3000
nburn=2*nsamples*nthin
niters=nburn+nsamples*nthin

m5 = stan(model_code=VieiraModel, model_name="VieraModel", data=dat_list, warmup = nburn, thin=nthin
    , iter=niters, chains=4, cores=4)

Posterior=as.array(m5)
```

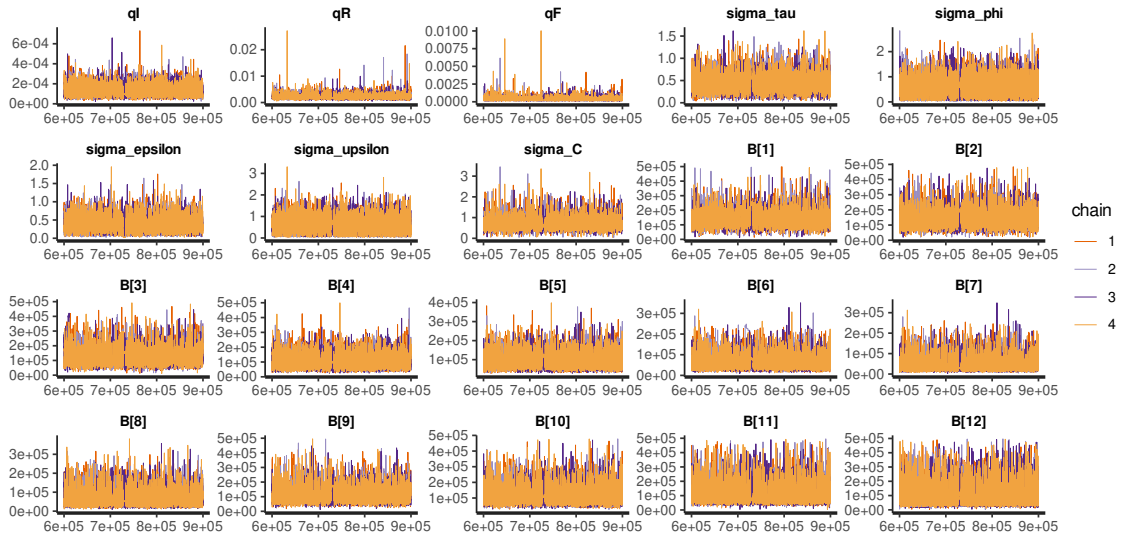

Fig. S 1. Cadenas de Markov de la estimación de los parámetros del modelo discreto con retardo en el crecimiento. Cada una de las 4 cadenas utilizadas en la estimación posee un color diferente y se puede visualizar la convergencia (índice de Gelman-Rubin Rmax=1.01).