How to use the rSPAMM package

T. A. Øigård and M. Biuw

2020-11-03

Overview

In this pdf you will find instructions on how to use the `rSPAMM` package for assessment of various harp and hooded seal populations. It will guide you through the model fitting, obtaining estimated quantities, finding and exploring, various catch options, how to structure the data set, and how to visualize the modelled population dynamics. In the appendix we provide a script for a complete analysis of the demo data, and we present the population dynamics model used.

To load the `rSPAMM` package type

```
library(rSPAMM)
```

Data used by the population dynamics model and how to load them

The population dynamics model use historical catch records, fecundity rates, age specific proportions of mature females, and estimates of pup production to estimate the population size trajectory. Two types of reproductive data are used in the model: information on the proportion of females that are mature at a given age (i.e., maturity ogive) and the proportion of mature females that are pregnant at a given year (i.e. fecundity rate). In this section we will describe what type of data is used, which data files are needed, and the format the various data are stored.

Demo data

When cloning the repository and installing the *rSPAMM* R package a demo data set is installed. In addition a full data set is available in the *wk_WKSEALS-2020/data/Norway/* folder.

The demo data is reproductive data, catch data, pup production estimates and priors used for population dynamics modelling of the harp seal population in the East Ice (White Sea).

To load the demo data:

```
data("harpeastDemo")
```

The demo data is a list called `harpeast` containing two lists called `data` and `parameters`. The `data` list contains the data needed to fit the population dynamics model and the `parameters` list contains the parameters estimated by the model along with initial values of them.

```
names(harpeast$data)
#>  [1] "Amax"              "Cdata"             "Nc"
#>  [4] "pupProductionData" "Np"                "Ftmp"
#>  [7] "Pmat"              "Npred"             "priors"
#> [10] "Npriors"           "CQuota"            "Pper"
#> [13] "fecundity"
```

```
names(harpeast$parameters)
#> [1] "logK"    "Mtilde"  "M0tilde"
```

To use the demo data set in the following examples it would be easiest to split the list in two separate lists `data` and `parameters`, i.e.,

```
data = harpeast$data
parameters = harpeast$parameters
```

Loading the full data set

The full data set is stored in the repositorys data folder (in the Norway folder).

It does not matter what is set to working directory, but when loading these data to use with the *rSPAMM* **you might want to adjust the *dataPath* variable defined below**. In this example it is assumed that the working directory is set to the *rSPAMM* model folder, i.e., that the working directory is the root folder of the *rSPAMM* package.

In order to load the data you have to specify which population you want to load the data for. The various alternatives are `harpeast`, `harpwest`, and `hooded` (which is the hooded seal population in the West Ice - the ice along the east coast of Greenland). In the examples that follows we will use the `harpeast` population.

To load the data used you run the following function:

```
dataPath1 = "../../../../data/Norway/"
dataPath2 = "../../../../../data/Norway/"
population = "harpeast"
dataFiles = paste0(dataPath1,population)
data <- load.data(population = dataFiles)
```

If dataPath1 does not work, try changing to dataPath2. Most importantly, you just want to specify the path to reflect where your data is stored.

To explore the data object you can

```
names(data)
#>  [1] "Amax"              "Cdata"             "Nc"
#>  [4] "pupProductionData" "Np"                "Ftmp"
#>  [7] "Pmat"              "Npred"             "priors"
#> [10] "Npriors"           "CQuota"            "Pper"
#> [13] "fecundity"
```

The data object is a list and you can further explore the actual values used by e.g.,

```
data$Amax
#> [1] 20

data$pupProductionData
#>         V1     V2    V3
#>  [1,] 1998 286260 0.150
#>  [2,] 2000 322474 0.098
#>  [3,] 2000 339710 0.105
#>  [4,] 2002 330000 0.103
#>  [5,] 2003 328000 0.181
#>  [6,] 2004 231811 0.190
#>  [7,] 2004 234000 0.205
#>  [8,] 2005 122658 0.162
#>  [9,] 2008 123104 0.199
#> [10,] 2009 157000 0.108
#> [11,] 2010 163032 0.198
#> [12,] 2013 128786 0.237
```

Parameters to be estimed and how to load them

As briefly mentioned earlier the population dynamics model is described by three parameters. The initial population size K, the mortality of the 1+ population M, and the pup mortality M_0. The initial population size is the population size for the year the model is fit from, and this is determined by the availability of the catch data. The earliest catch data is from 1946, so the model is fit from 1946 and up to present. It also predicts the population dynamics into the future, and this is default set to 15 years.

To load the initial values of the parameters to be estimated run:

```
parameters <- load.initial.values(population = "harpeast")
```

The parameters object is also a list and the content looks like this for the `harpeast` population

```
parameters
#> $logK
#> [1] 15.42495
#>
#> $Mtilde
#> [1] -2.090741
#>
#> $M0tilde
#> [1] 0.2006707
```

Here `logK` is the log transformed initial population size, `M0tilde` is the logit transformed pup mortality, and `Mtilde` is the logit transformed 1+ mortality (seals of age 1 and greater). The reason that transformed parameters are estimated instead of the non transformed parameters is that the log transformation of the initial population ensures that the model provides a stricktly positive estimate of the initial population. The logit tranformation of the mortalities ensures that the estimates are bounded between 0 and 1.

Model fitting and obtaining estimates

The model is implemented in the Template Model Builder framework and makes use of the `TMB` R package. This means that the code for model is written in C++ and will be compiled when you install the `rSPAMM` package. When installing the `rSPAMM` package the `TMB` package and all dependencies should also be installed automatically.

Fitting the population dynamics model

The optimization routine used is `nlminb`. To fit the model run:

```
optobj <- run.model(data = data, par = parameters)
#>
#> --------------------------------------------------
#>
#>  Optimization converged:  relative convergence (4)
#>
#>
#>  Parameter estimates
#>  -------------------
#>  Initial population size: K =  1710731
#>  Pup mortality:           M0 =  0.27
#>  1+ mortality:             M =  0.13
#>
#> --------------------------------------------------
```

When the model is fitted some key information will be written to screen. This key information show whether the model converged, what type of convergence and estimates of the parameters that describes the modelled population dynamics.

The `run.model` function returns a list containing the model object `obj` and the final optimized opject `opt`.

Obtaining the estimated quantities

In addition to the estimated parameters needed to describe the modelled population dynamics of a population the model fit also provides the user with other relevant quantities. The Table below lists the various quantities obtained.

Variable name	Description
rep	Check it out
rep.matrix	Check
rep.names	check
indN0	Indexes of where the fitted pup population is found in the rep.matrix
indN1	Indexes for the modelled 1+ population
indNtot	Indexes for the total population
indD1	Index for the estimated of D
indD1new	Index for the estimated D1new
indN0Current	Index for the modelled pup abundance in the current year
indN1Current	Index for the modelled 1+ abundance in the current year
indNtotCurrent	Index for the modelled total population in the current year
years	The years the model is fitted for
Kest	Estimated initial population size
Kest.sd	Standard deviation of the initial population size
Mest	Estimated 1+ mortality
Mest.sd	Standard deviation of the 1+ mortality
M0est	Estimated pup mortality
M0est.sd	Standard deviation of the estimated pup mortality
D1	Estimated depletion coefficient
DNmax	Estimated of depletion coefficient using Nmax and Npred
N0Current	Estimated pup abundance for the current year
N1Current	Estimated 1+ abundance for the current year
NTotCurrent	Estimated total abundance for the current year
D1.sd	Standard deviation of estimated D
DNmax.sd	Standard deviation of estimated D using Nmax and Npred

Variable name	Description
N0Current	Standard deviation of current pup abundance
N1Current	Standard deviation of current 1+ abundanceI
NTotCurrent	Standard deviation of current total abundance

The model results can be obtained by running:

```r
res <- model.results(data = data,optobject = optobj)
```

The `res` object is a list containing all the estimates of the quantities listed in the Table above.

```r
names(res)
#>  [1] "rep"            "rep.matrix"     "rep.rnames"     "indN0"
#>  [5] "indN1"          "indNTot"        "indD1"          "indNmax"
#>  [9] "indN0Current"   "indN1Current"   "indNTotCurrent" "years"
#> [13] "Kest"           "Kest.sd"        "Mest"           "Mest.sd"
#> [17] "M0est"          "M0est.sd"       "D1"             "DNmax"
#> [21] "N0Current"      "N1Current"      "NTotCurrent"    "NTotmax"
#> [25] "D1.sd"          "DNmax.sd"       "N0Current.sd"   "N1Current.sd"
#> [29] "NTotCurrent.sd"
```

A table for the most interesting quantities needed can be made by running:

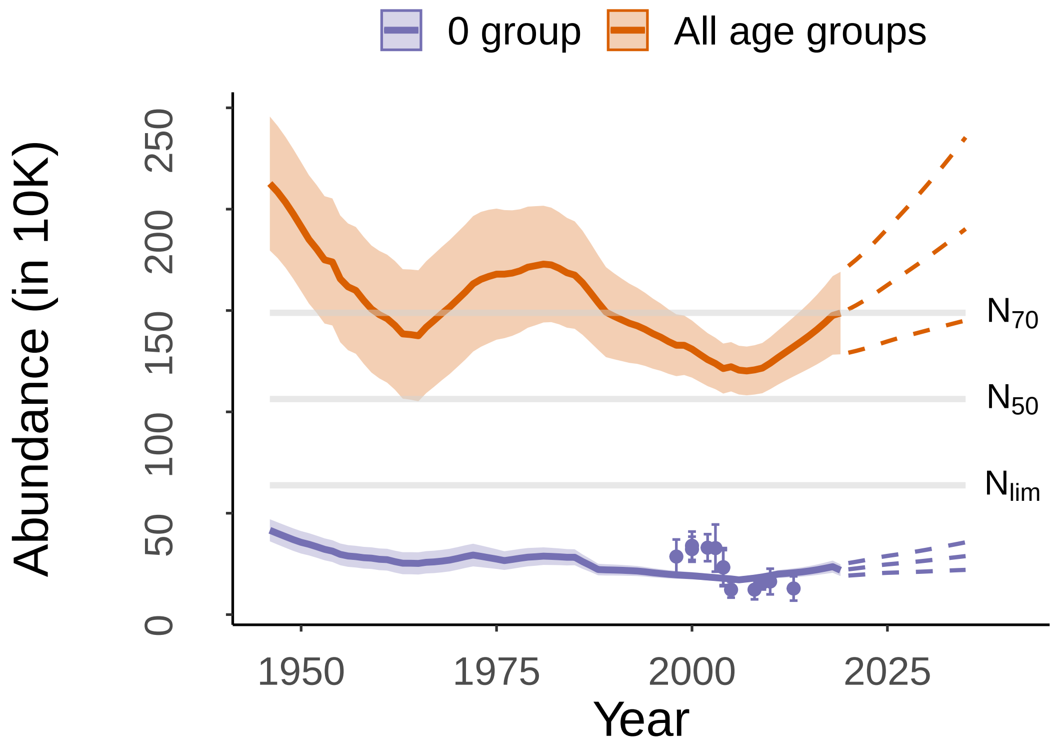
```r
partab <- par.table(results=res, dat=data)

partab
#>              par    parNames         Mean           SD
#> 1           Kest       N1946 1.710731e+06 1.414960e+05
#> 2          M0est          M0 2.691386e-01 2.532728e-01
#> 3          M1est         M1+ 1.286299e-01 5.159749e-02
#> 4      N0Current     N0,2019 2.188735e+05 1.478594e+04
#> 5      N1Current    N1+,2019 1.268621e+06 8.980014e+04
#> 6    NTotCurrent NTotal,2019 1.487495e+06 1.038380e+05
#> 7             D1         D1+ 1.258786e+00 7.413898e-02
#> 8 NTotprojected NTotal,2035 8.946455e-01 1.733573e-01
```

Visualize the modelled population dynamics

In this section we will show how to create graphics to present the various output of the modelling results. To display the modelled pup abundance and the total abundance you run the following function:

```r
plotRes(res,data)
#> Warning: package 'ggplot2' was built under R version 3.6.3
```

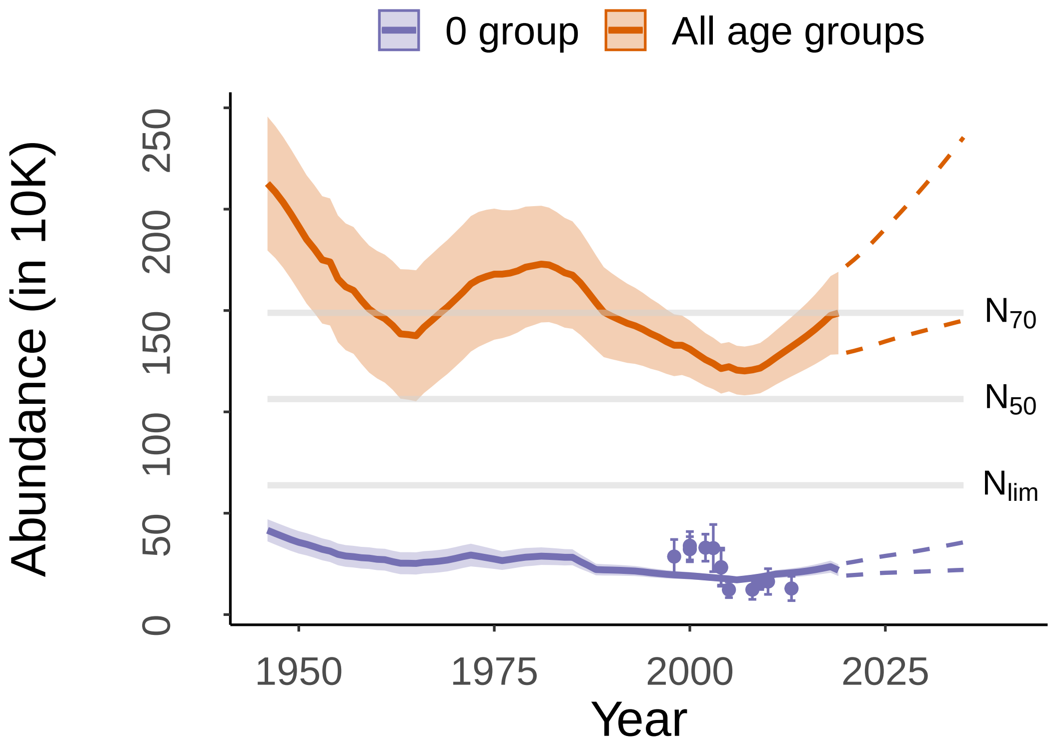


In order to open an OS independent graphical devise use the option `grDev = TRUE`, i.e,

```
plotRes(res,data,grDev = TRUE)
```

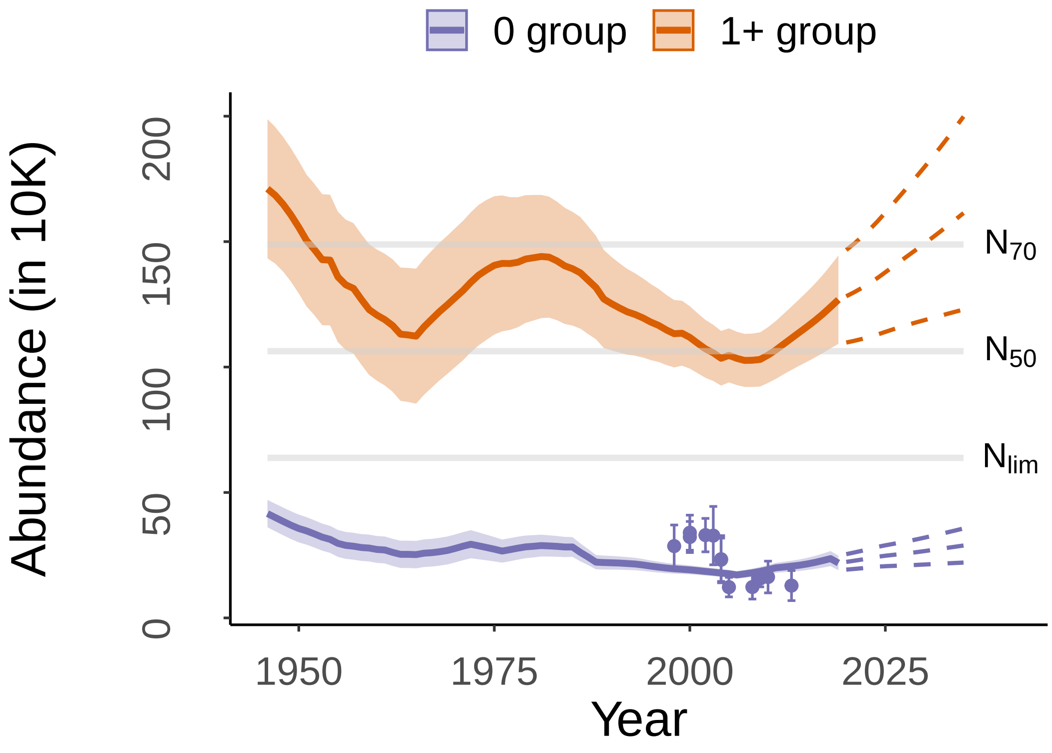
If you want to turn off the mean trajectory for the predictions and only plot the 95 percent konfidence intervals for the projections you add the option `plotProjMean = FALSE`, i.e.,

```
plotRes(res,data,plotProjMean = FALSE)
```



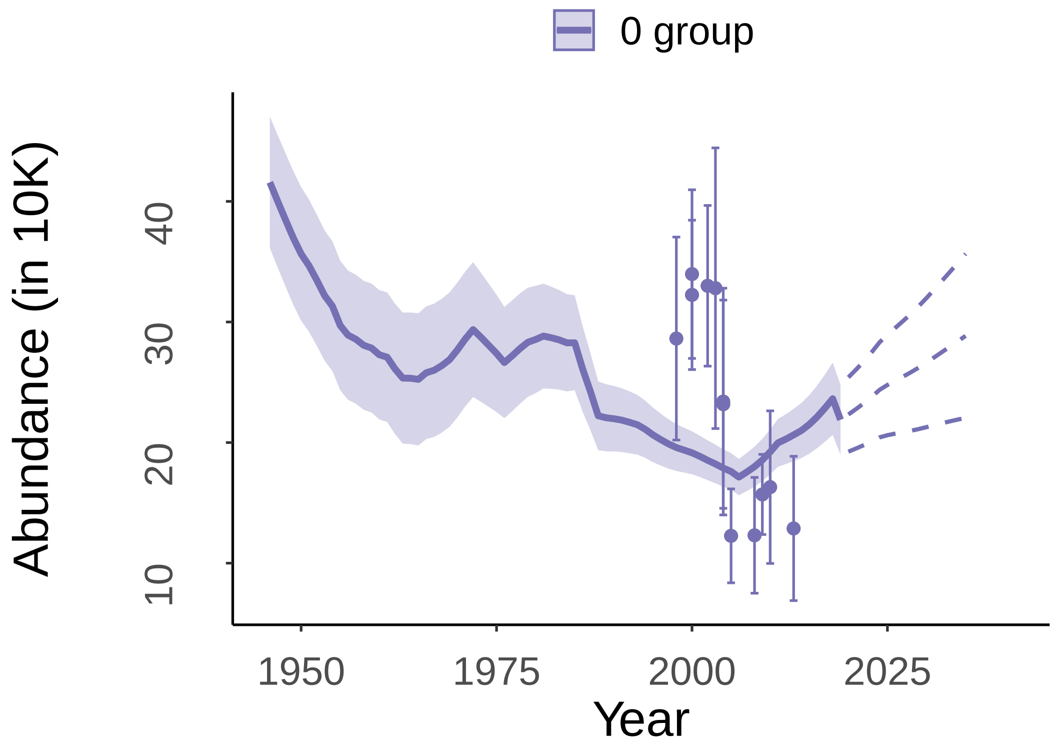
If you want to show the pup trajectory and the 1+ group you change the `compnent` option, i.e.,

```
plotRes(res,data,component = c("N0","N1"))
```

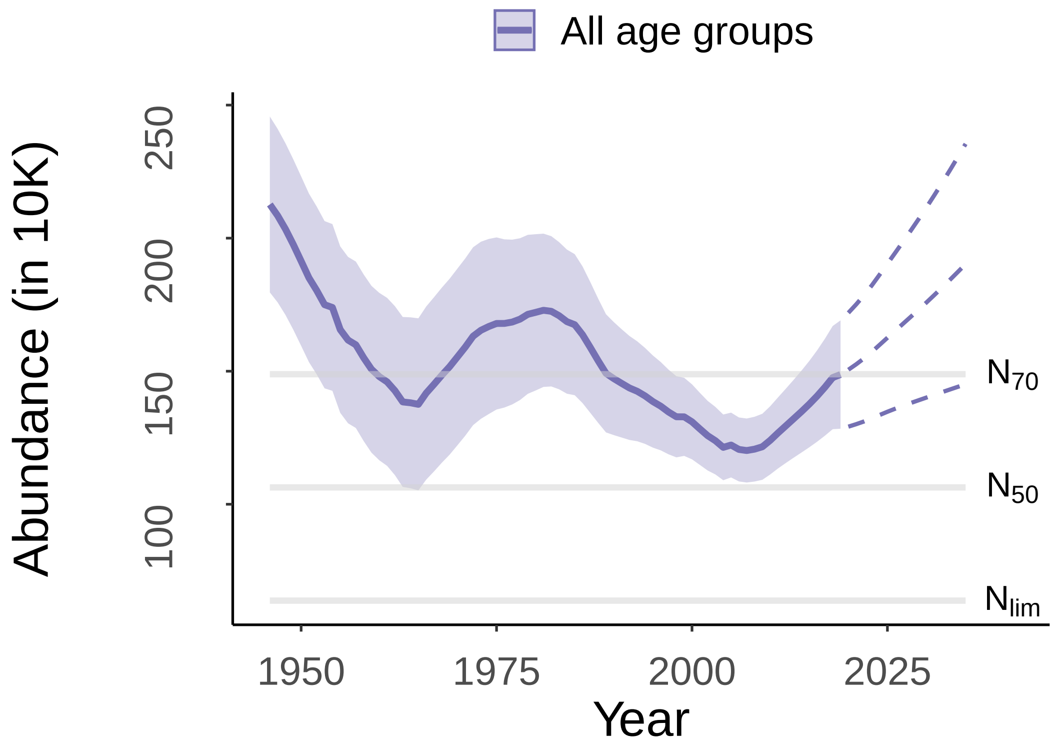


You can also plot the fit to the pup abundance, the 1+ group, or the total abundance separately by specifying that in the `component` option:

```
plotRes(res,data,component = c("N0"))
```

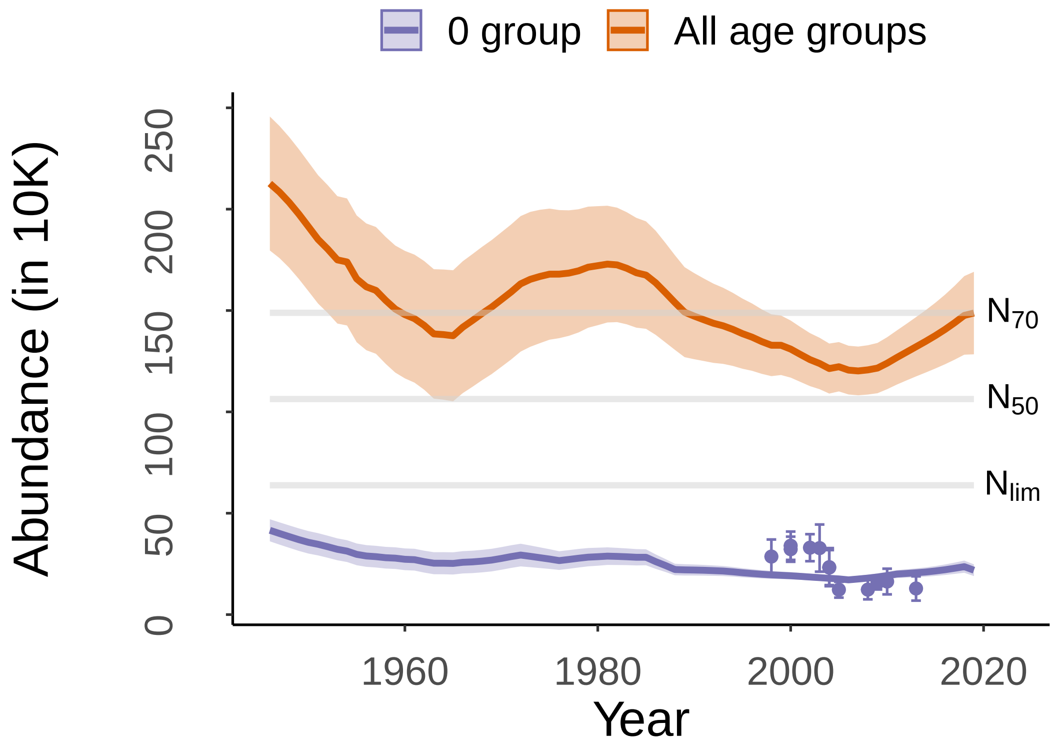


```r
plotRes(res,data,component = c("Ntot"))
```



You can also choose to not plot the future projections by setting `plotProjections = FALSE` .

```
plotRes(res,data,plotProjections = FALSE)
```



The horizontal lines for N_{lim}, N_{50} and N_{70} can be turned off by setting `plotNlims = FALSE`. For other options you can run `?plotRes`.

If you want to visualize certain types of data you can plot the historical catch level used, the fecundity vector and the birth ogive curves for each time period. To plot the catch data run the `plotCatch()` function:

```
plotCatch(catch = data$Cdata)
```

Default the bars are next to each other, but the bars can be plotted on top of each other by setting `position = "stack"`:

```
plotCatch(catch = data$Cdata,position = "stack")
```

You can plot the modelled population dynamics (pup abundance and total abundance, not the 1+ population) and the reported catch in the same figure using `plotResCatch()`

```
plotResCatch(res,data)
```

If you want to explore the birth ogive data used for various periods you can run the `plotOgive()` function:

```
plotOgive(dat = data)
```

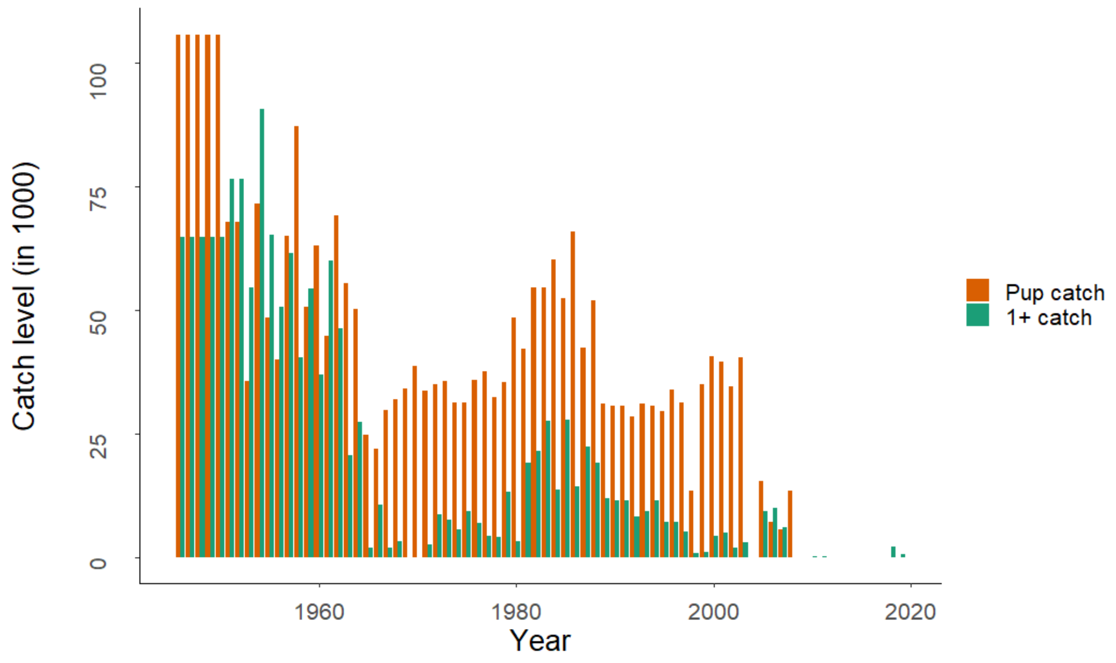


Figure 1: Reported catch levels of pups and 1+ animals with bars next to each other.

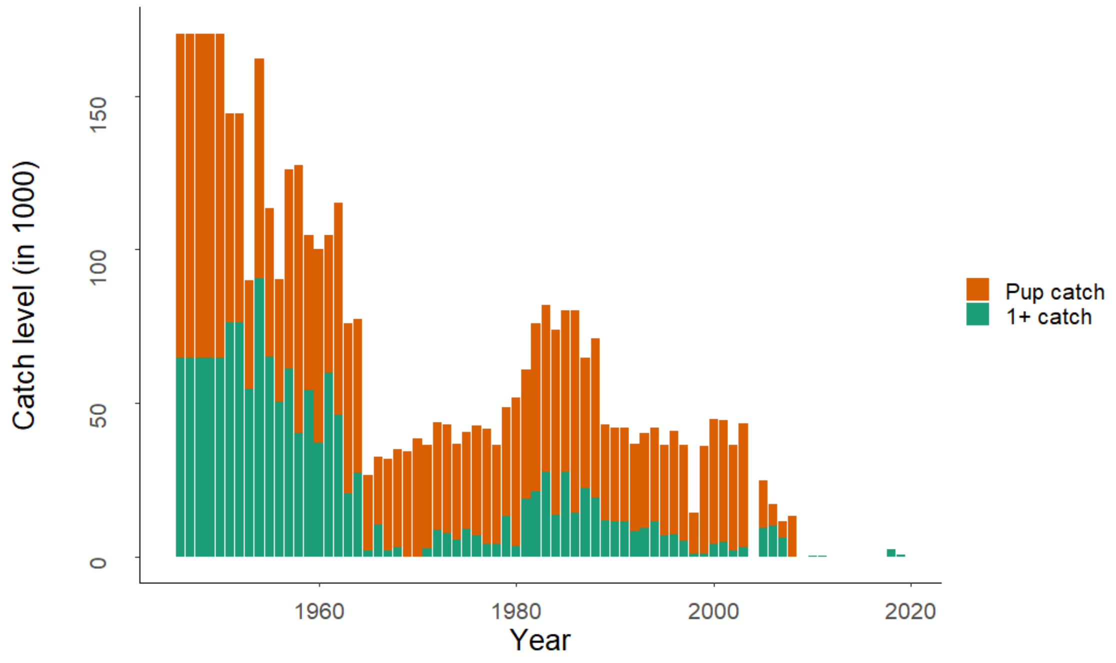


Figure 2: Reported catch levels of pups and 1+ animals with bars on top of each other.

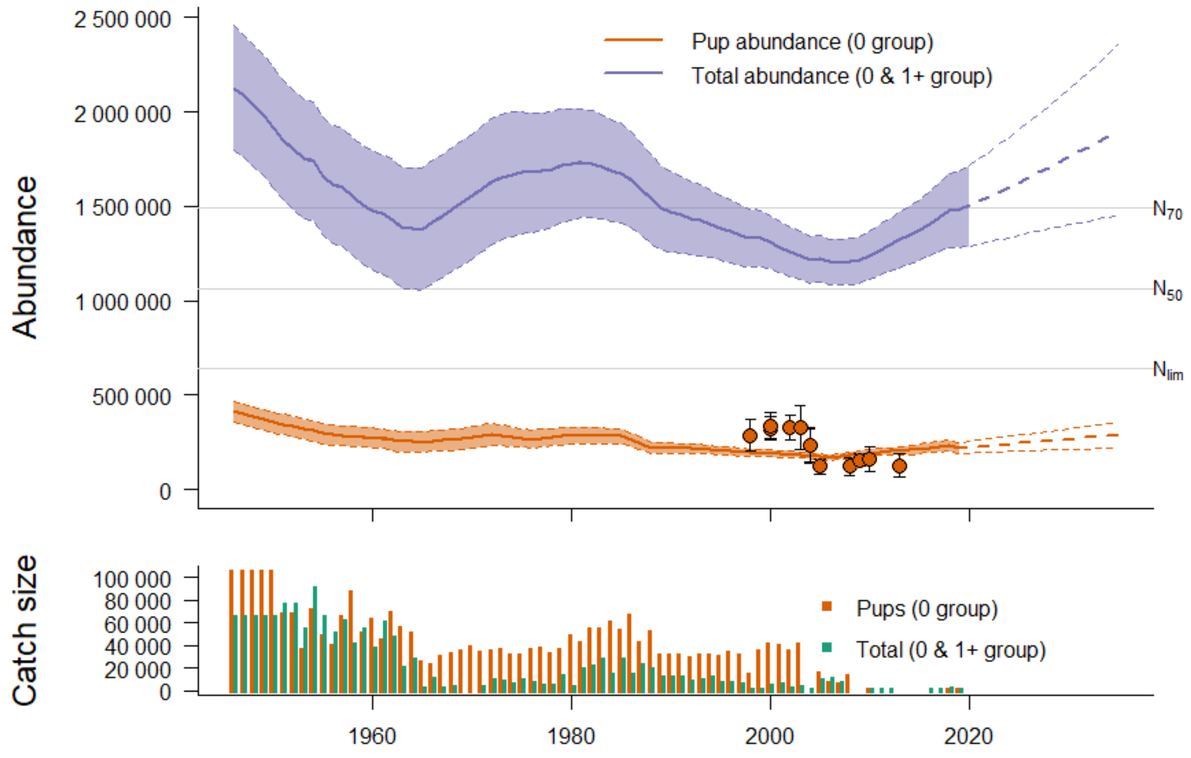
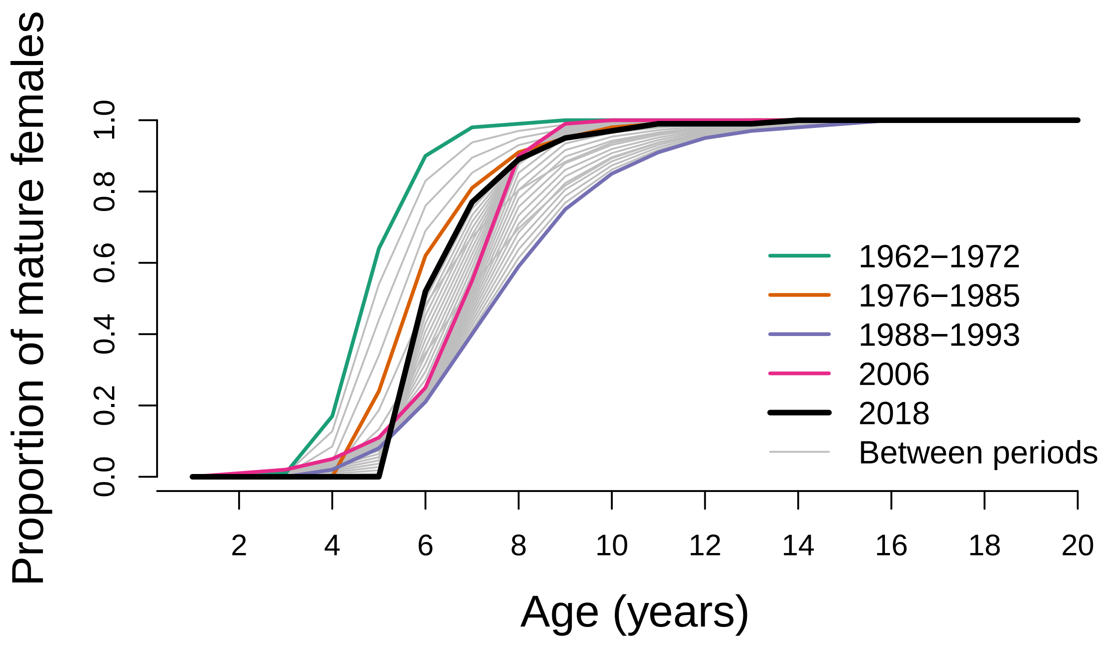
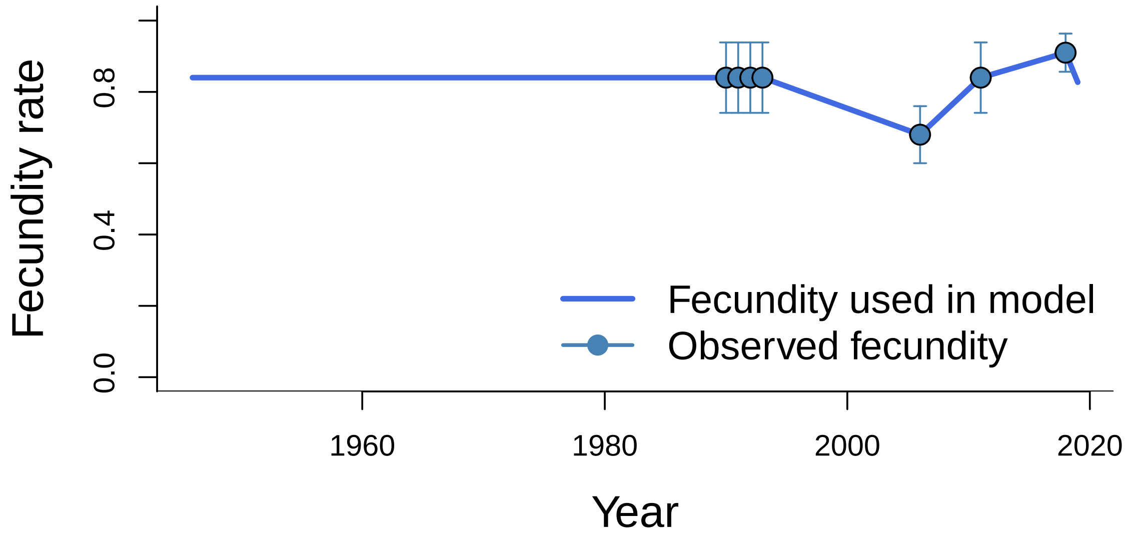


Figure 3: Reported catch levels of pups and 1+ animals with bars next to each other.



Since fecundity rates is not available for all years they are interpolated between missing years. You can plot the fecundity rates used in the modelling by running `plotFecundity()`. Default it plots both the linear interpolated fecundity rates and the observed fecundity ratesfor a given population.

```
plotFecundity(dat = data)
```



Exploring various catch options

In this section we will describe how to explore various catch options such as finding the equilibrium catch level, i.e., the fixed annual catch level that keeps the future projected population abundance constant, finding the catch level that would reduce the population to N_{70} (70% of the current population size) with probability 0.8 over a 15-years period, and Potential Biological Removals (PBR) catch level.

Equilibrium catch level

To find the equilibrium catch level we run the function `find.eq.quota()`.

Default is that the function assumes that zero pups are in the catch and that all catch are 1 year old animals or more. To change he proportion of pups and 1+ animals in the catch you use the `quota` variable, i.e., the following finds the equilibrium catch level assuming 15% pups and 85% 1+ animals in the catch:

```
EquilibriumCatch <- find.eq.quota(data = data,
                                  parameters = parameters,
                                  quota = c(0.15,0.85))
#> -----------------------------------------------------
#>
#> Estimated equilibrium:
#> Pups:   3440
#> Adults: 19492
#> Total : 22932
#>
```

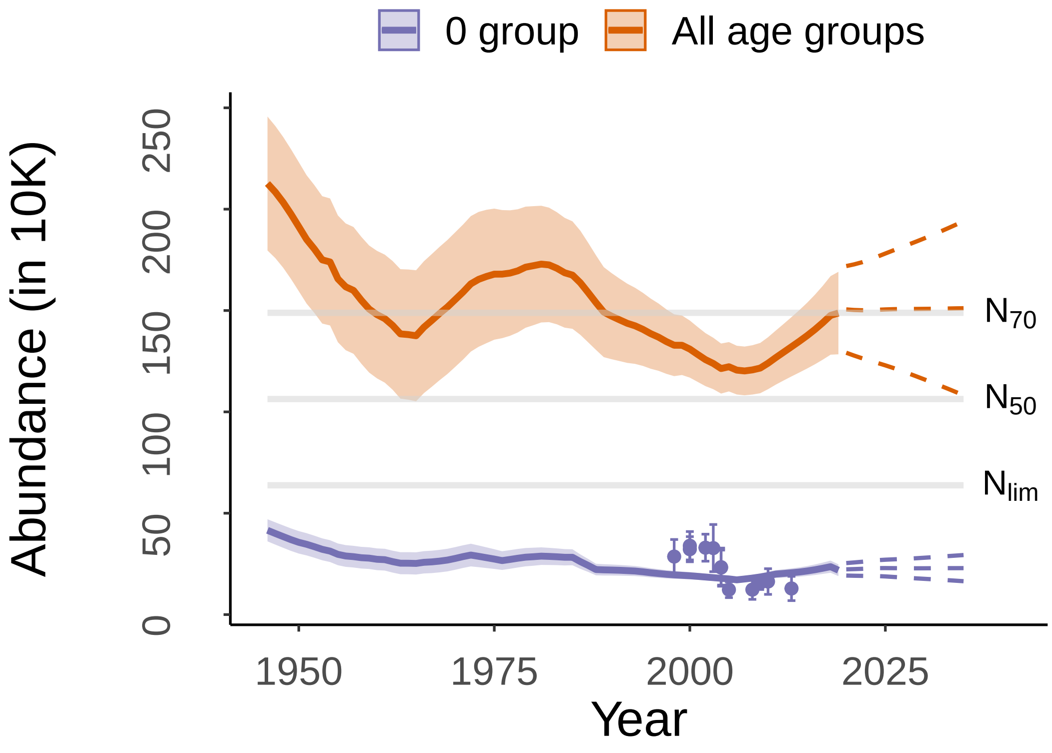
```
#> ----------------------------------------------------

EquilibriumCatch
#> [1]  3439.723 19491.766
```

You have now found the fixed annual catch level that stabilizes the future 1+ population under the estimated model for the `harpeast` population and can re-run the model using this equilibrium catch level for the future projection. Default in the when running the model fit assumes zero catch for the future predictions. To re-run the model and plot the estimated future trajectory using the equilibrium catch level you run the following code:

```
#Rerun the model
data$CQuota = EquilibriumCatch
optEq = run.model(data = data, par = parameters)
#>
#> -------------------------------------------------
#>
#>  Optimization converged:  relative convergence (4)
#>
#>
#>  Parameter estimates
#>  -------------------
#>  Initial population size: K =  1710731
#>  Pup mortality:          M0 =  0.27
#>  1+ mortality:            M =  0.13
#>
#> -------------------------------------------------
resEq = model.results(data = data,optobject = optEq)

#Plot model
plotRes(resEq,data)
```



N_{70} catch level

To find the the catch level that would reduce the population to 70% of the historical maximum observed (modelled) with probability 0.8 over a 15-year period you run the function `find.N70.quota()`:

```
catchN70 = find.N70.quota(data = data,
                          parameters = parameters,
                          quota = c(0,1))
#>
#>  ---------------------------------------
#>
#>  Current population size is already within
#>  the 80 percent confidence interval of
#>  the  15  year prediction.
#>  Hence, no catch level will be estimated.
#>
#>  ---------------------------------------
```

For this population it turns out that the current population size is below N_{70}. Because of this there is no point estimating the N_{70} catch level.

This example assume that zero pups are in the catch and that all catch are 1 year old animals or more. To change he proportion of pups and 1+ animals in the catch you use the `quota` variable, i.e., use `quota = c(0.15,0.85)` to find the $N70$ catch level assuming 15% pups and 85% 1+ animals in the catch:

If the N_{70} population size was below the current population size you could re-run the model using the estimated N_{70} catch level

```
data$CQuota = catchN70
optN70 = run.model(data = data, par = parameters)
resN70 = model.results(data = data,optobject = optN70)

#Plot the estimated future trajectory using the N70 catch level
plotRes(resN70,data)
```

The Potential Biological Removals (PBR) catch level

Potential Biological Removals has been defined as:

$$PBR = \frac{1}{2} R_{max} F_r N_{min},$$

where R_{max} is the maximum rate of increase for the population (default to 0.12 for pinnipeds), F_r is the recovery factor with values between 0.1 and 1 (default to 0.5), and N_{min} is the estimated population size using 20% percentile of the log-normal distribution. The PBR catch level assumes that the age structure of the removals is proportional to the age composition of the population (i.e. 14%). To find the PBR catch level using the default parameters you run:

```
pbrCatch = PBR(n0=partab[4,3],
               n1=partab[5,3],
               se0=partab[4,4],
               se1=partab[5,4])

pbrCatch
#> $Nmin
#> [1] 1402092
#>
#> $CV
#> [1] 0.07031021
#>
#> $PBR
#> [1] 42063
#>
#> $n0catch
#> [1] 5889
#>
#> $n1catch
#> [1] 36174
```

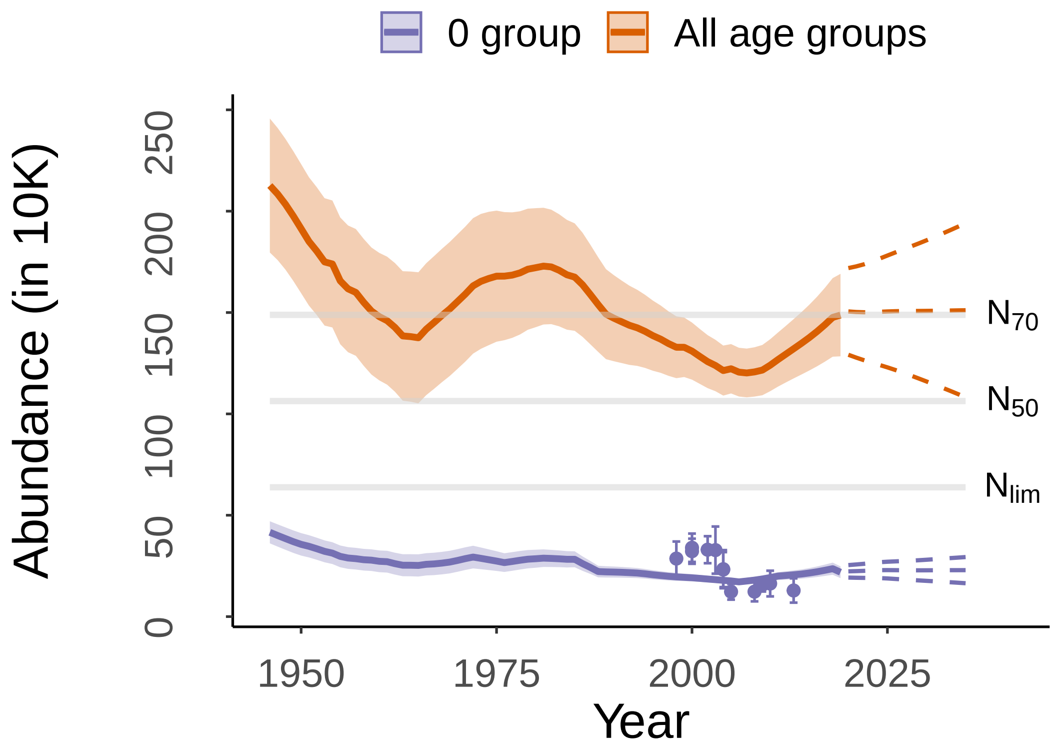
where n0 is the current population size, n1 is the current 1+ population size, se0 is the standard deviation of the estimated current pup abundance, and se1 is the standard deviation of the estimated current 1+ abundance. Recall from the Table above that these quantities could be found using the `par.tab()` function.

Re-run the model using this catch level

```
data$Quota = c(pbrCatch$n0catch,pbrCatch$n1catch)
optPBR = run.model(data = data, par = parameters)
#>
#> --------------------------------------------------
#>
#>  Optimization converged:  relative convergence (4)
#>
```

```
#>
#>  Parameter estimates
#>  -------------------
#>  Initial population size: K =  1710731
#>  Pup mortality:          M0 =  0.27
#>  1+ mortality:            M =  0.13
#>
#> --------------------------------------------------
resPBR = model.results(data = data,optobject = optPBR)

#Plot the estimated future trajectory using PBR catch level
plotRes(resPBR,data)
```



Appendix

Complete code for assessment

This section contains a script for a complete analysis of the harp seal population in the East Ice (in the White Sea) using the `rSPAMM` package.

```
#Load the rSPAMM package
library(rSPAMM)
```

```
###################
# Data
###################
# You can choose to use either the demo data which is
# included in the rSPAMM package or loading the full data

#Use this if analysing the demo data.
#Loading demo data and parameters
data("harpeastDemo")
data = harpeast$data
parameters = harpeast$parameters


#Use this if analysing latest version of the
#complete data set
#Download full data. Default the function will
#ask to create a new folder. If you already have
#a folder you can add the option
#"chooseFolder = FALSE".
#Note the Working Directory has to be set to the
#root folder of the downloaded data, i.e., not
#the Data or the Scripts folder.
downloadData()
#Loading the data
data = load.data(population = "harpeast")
#Loading the parameters
parameters = load.initial.values(population = "harpeast")


##################
# Model fitting
##################
#Run the model
optobj = run.model(data = data, par = parameters)

#Obtain the results
res = model.results(data = data,optobject = optobj)

#Create a nice table with the results
partab = par.table(results=res, dat=data)

#Plot the results (both the pup model fit and the N1+ population)
plotRes(res,data,grDev = TRUE)


##################
# Catch options
##################

#Find the equilibrium catch level
#In this example we are assuming 15% pups
#and 85% 1+ animals in the catch
EquilibriumCatch = find.eq.quota(data = data,
```

```r
                                  parameters =parameters,
                                  quota = c(0.15,0.85))

#Rerun the model using the estimated equilibrium catch level
data$CQuota = EquilibriumCatch
optEq = run.model(data = data, par = parameters)
resEq = model.results(data = data,optobject = optEq)

#Plot the estimated future trajectory using equilibrium catch level
plotRes(resEq,data,grDev = TRUE)

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

#Find the N70 catch level
catchN70 = find.N70.quota(data = data,
                           parameters = parameters,
                           quota = c(0,1))

#For this population it turns out that the current population
#size is below N70. Because of this there is no point estimating
#the N70 catch level.

#If the current population size was above N70 you could
#rerun the model using the estimated N70 catch level
data$CQuota = catchN70
optN70 = run.model(data = data, par = parameters)
resN70 = model.results(data = data,optobject = optN70)

#Plot the estimated future trajectory using the N70 catch level
plotRes(resN70,data,grDev = TRUE)

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

#Find the PBR catch level
#In this example we are assuming 14% pups
#and 86% 1+ animals in the catch
pbrCatch = PBR(n0=partab[4,3],
               n1=partab[5,3],
               se0=partab[4,4],
               se1=partab[5,4])

#Re-run the model using the PBR catch level
data$CQuota = c(pbrCatch$n0catch,pbrCatch$n1catch)
optPBR = run.model(data = data, par = parameters)
resPBR = model.results(data = data,optobject = optPBR)

#Plot the estimated future trajectory using PBR catch level
plotRes(resPBR,data,grDev = TRUE)
```

The population dynamics model

The population model is an age-structured population dynamics model. For initiation of the model it is assumed that the population had a stable age structure in year $y_0 = 1945$, i.e.,

$$N_{i,y_0} = N_{y_0} s_{1+}^{i-1}(1 - s_{1+}), \quad i = 1, \ldots, A-1,$$

$$N_{A,y_0} = N_{y_0} s_{1+}^{A-1}.$$

Here A is the maximum age group containing seals aged A and higher, and set to 20 years, and N_{y_0} is the estimated initial population size in year y_0. The model is parameterized by the natural mortalities M_0 and M_{1+} for the pups and seals of 1 year and older, respectively. These mortalities determine the survival probabilities $s_0 = \exp(-M_0)$ and $s_{1+} = \exp(-M_{1+})$.

The model has the following set of recursion equations:

$$N_{a,y} = (N_{0,y-1} - C_{0,y-1})\, s_0,$$

$$N_{a,y} = (N_{a-1,y-1} - C_{a-1,y-1})\, s_{1+}, \quad a = 2, \ldots, A-1,$$

$$N_{A,y} = [(N_{A-1,y-1} - C_{a-1,y-1}) + (N_{A,y-1} - C_{A,y-1})]\, s_{1+}.$$

Since available data do not allow for more detailed age-dependence in survival to be estimated it is assumed that the mortality rates are age-independent within the 1+ group. The $C_{a,y}$ are the age-specific catch numbers. Catch records are aggregated over age, and only provide information about the annual number of pups and number of 1+ seals caught. To obtain $C_{a,y}$ we assume that the age-distribution in the catch follows the modelled age distribution and employ *pro rata* rules in the model:

$$C_{a,y} = C_{1+,y} \frac{N_{a,y}}{N_{1+,y}}, \quad a = 1, \ldots, A.$$

where $N_{1+,y} = \sum_{y=1}^{A} N_{a,y}$, with $N_{a,y}$ being the number of individuals ate age a in year y.

The modelled pup abundance is given by

$$N_{0,y} = \frac{F_y}{2} \sum_{a=1}^{A} p_{a,y} N_{a.y},$$

where $N_{a,y}/2$ is the number of females ate age a in year y, F_y is the time-varying fecundity rates and $p_{a,y}$ are the time-varying age specific proportions of mature females.

The model is fitted to the survey pup production estimates and the fecundity rates by maximum likelihood. Assuming normality for the pup production estimates, their contribution to the log-likelihood function is

$$\sum_y -\log(\sigma_{0,y}) - \frac{1}{2} \frac{(N_{0,y} - n_{0,y})^2}{(\sigma_{0,y})^2},$$

where $n_{0,y}$ and $\sigma_{0,y}$ denote the survey pup production count and corresponding standard error for year y.

The model has a Bayesian flavour as priors are imposed on some of the parameters. A vague normal prior is assumed for the initial population size N_{y_0}. A truncated normal prior was used for both the pup mortalities M_0 and M_{1+}.

All parameter estimates are found by maximizing the likelihood function using the R package `TMB`.