# Bayesian Demographic Trajectory analysis tutorial: BaSTA and BaFTA

Fernando Colchero

2025-09-16

## Contents



## Introduction

This is a general introduction to the R packages BaSTA and BAFTA. The aim is to provide a simple tutorial that allows users to start running survival and fertility analyses with both packages. For a more in depth description of the models and methods, please refer to the relevant papers (Colchero and Clark 2012, Colchero et al. 2012, Colchero et al. 2021, Colchero 2025), and the two vignettes, available under "01docs/Vignettes/".

Together with this tutorial, I provide R scripts with all the commands provided here. They also include additional analyses to give you a better idea of the kinds of analyses you can carry out with both packages. You can find both under "02code/BaSTAtutorialRcode.R" and "02code/BaFTAtutorialRcode.R".

In Fig. 1 I show the file structure for the workshop materials.

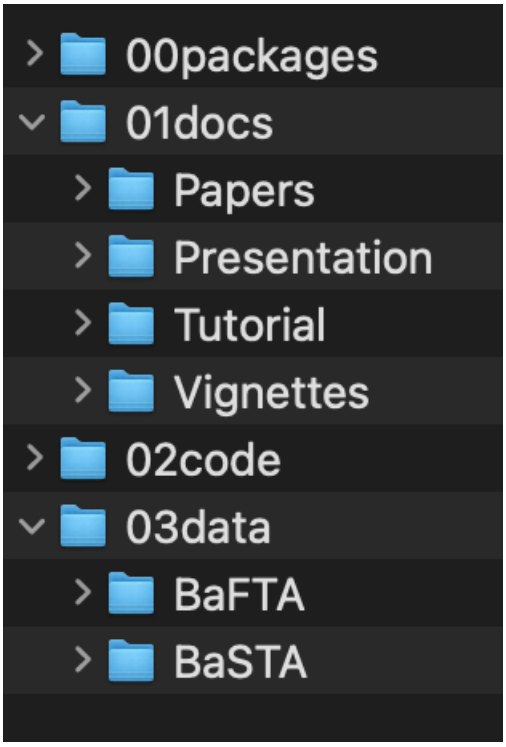



Figure 1: A caption

## Installing packages and loading additional code

First, make sure to point R to the workshop's working directory as:

```
# Set WD to workshop directory:
setwd("Path to Workshop directory/ColcheroWorkshopEvoDemoS2025/")
```

BaSTA version 2.02 is available in CRAN, although I recommend that you install the latest version from my GitHub repository. To install BaSTA and BaFTA from GitHub, use the following commands:

```
# Install and load 'devtools':
install.packages("devtools")
library(devtools)

# Install BaSTA:
install_git("https://github.com/fercol/basta2.0", subdir = "pkg/")

# Install BaFTA:
install_git("https://github.com/fercol/BaFTA", subdir = "pkg/")
```

In case installing from GitHub doesn't work, you can install them from the versions available in the workshop's directory as

```
# Install BaSTA from local directory:
install.packages("00packages/BaSTA_2.0.3.tar.gz", type = "source",
                 repos = NULL)

# Install BaFTA from local directory:
install.packages("00packages/BaFTA_1.0.0.tar.gz", type = "source",
```

```
                repos = NULL)
```

For both packages, you will also need to install snowfall for parallel computing:

```
# Install snowfall:
install.packages("snowfall")
```

In addition to the libraries, I provide two R scripts with additional code to complement both packages. Load the additional functions as:

```
# Load extra functions for BaSTA:
source("02code/BaSTAextraFunctions.R")

# Load additional functions for BaFTA:
source("02code/BaFTAextraFunctions.R")
```

# BaSTA: Bayesian Survival Trajectory Analysis

Here I will briefly show how to run analyses on **capture-mark-recapture (CMR)** and on **census** data. CMR it the type of data that was traditionally used with BaSTA, while the package has been recently extended to allow the use of census data. These type of data are relevant for studies where individuals are followed constantly and for which constructing a recapture matrix is unnecessary.

You can find the R code file associated to this section at "02code/BaSTAtutorialRcode.R' '.

## Capture-mark-recapture (CMR) data

### Loading and checking data

To illustrate the use of BaSTA on CMR data, I provide the capture-mark-recapture data on Eurasian sparrowhawk (*Accipiter nisus*), compiled by the British ornithologist Ian Newton. Until recently, this dataset was hosted in the Long-term Individual-basedTime Series (LITS) database (Jones et al. 2008). Load the data as:

```
# Sparrowhawk:
sphDat <- read.csv(file = "03data/BaSTA/tables/SparrowhawkData.csv")
```

Any CMR dataset to be analyzed with BaSTA needs to be structured as: a) the first three columns are the individual ID, and the times of birth and death, commonly in years; b) these are followed by the capture-recapture-matrix; and c) any covariates to be included in the analysis. Here is the structure for the sparrowhawk data:

Table 1: Example of CMR table for BaSTA based on the sparrowhawk dataset.

| ID | birth | death | X1971 | CMR.cols. | X1999 | sex | repr | wing | mass |
|---|---|---|---|---|---|---|---|---|---|
| DA53804 | 1986 | NA | 0 | - | 0 | f | 4 | 0.4677805 | 0.4689179 |
| DA53824 | NA | NA | 0 | - | 0 | f | 4 | 0.9210977 | 0.1701786 |
| DA53825 | 1984 | NA | 0 | - | 0 | f | 4 | 1.1477563 | 0.9917118 |
| DS85729 | 1974 | NA | 0 | - | 0 | f | 4 | 0.6944391 | 0.1390599 |
| EB01503 | 1974 | NA | 0 | - | 0 | f | 0 | 0.1844573 | 0.4191280 |

Note that only the first and last columns of the capture matrix in Table 1 are visible here. To verify that

there are not issues with the data, use function **DataCheck** as:

```
sphCheck <- DataCheck(object = sphDat, studyStart = 1971, studyEnd = 1999)
```

You can use function **summary** to print the results to the console as:

```
summary(sphCheck)
```

```
> DATA CHECK: 2025-09-16 06:09:49.664401
> ================================
>
> DATA SUMMARY:
> =============
> - Number of individuals:         572
> - Number with known birth year:  378
> - Number with known death year:    0
> - Number with known birth
>   AND death years:                 0
> - Number of recaptures:          948
> - Earliest detection year:      1971
> - Latest detection year:        1999
> - Earliest birth year:          1969
> - Latest birth year:            1996
> - Earliest death year:            NA
> - Latest death year:              NA
>
> NUMBER OF PROBLEMATIC RECORDS:
> ==============================
> - Death < Entry:  0
> - No obs.:        0
> - Birth > Death:  0
> - Death < Depart: 0
> - Birth > Entry:  0
> - Birth = 1 in Y: 1
>
> Note: you can use function FixCMRdata() to fix issues.
```

### Single runs

To run a single analysis you should use function **basta**. For instance, to run an analysis with a Weibull hazard rate or mortality function, you can use:

```
# Weibull model:
outSph1 <- basta(object = sphDat, studyStart = 1971, studyEnd = 1999,
                 model = "WE", shape = "simple", formulaMort = ~ sex - 1,
                 parallel = TRUE, nsim = 4, ncpus = 4)
```

Here, arguments **model** and **shape** specify the main mortality function (here Weibull) and the general shape (options are *simple*, *Makeham*, and *bathtub*). The output can be visualized by printing it to the console or with function **summary** as:

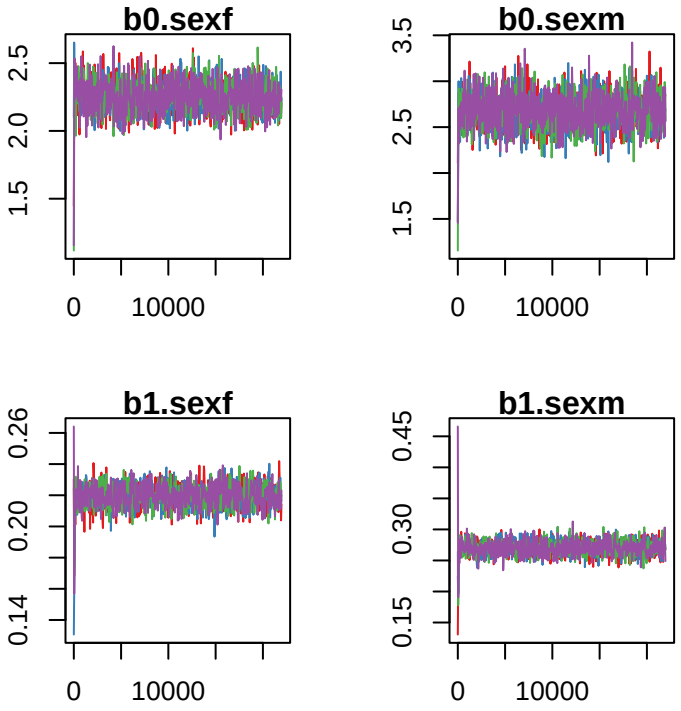
```
summary(outSph1, digits = 3)
```

```
>
> Call:
> Model                     : WE
```

```
> Shape                     : simple
> Minimum age               : 0
> Covars. structure         : fused
> Cat. covars.              : sexf, sexm
> Cont. covars.             :
>
> Model settings:
>    niter   burnin thinning     nsim
>    22000     2001       40        4
>
> Mean Kullback-Leibler
> discrepancy calibration (KLDC):
>                b0 b1
> sexm - sexf 0.985  1
>
> Coefficients:
>          Mean  StdErr Lower95%CI Upper95%CI SerAutocorr UpdateRate
> b0.sexf 2.250 0.11027      2.042      2.469      0.0715      0.253
> b0.sexm 2.672 0.18280      2.334      3.039      0.0143      0.246
> b1.sexf 0.219 0.00694      0.206      0.233      0.1581      0.263
> b1.sexm 0.270 0.01133      0.249      0.292      0.0820      0.252
> pi      0.503 0.01547      0.474      0.533      0.1362      1.000
>         PotScaleReduc
> b0.sexf             1
> b0.sexm             1
> b1.sexf             1
> b1.sexm             1
> pi                  1
>
> Convergence:
> Appropriate convergence reached for all parameters.
>
> DIC:
> 2382.077
```

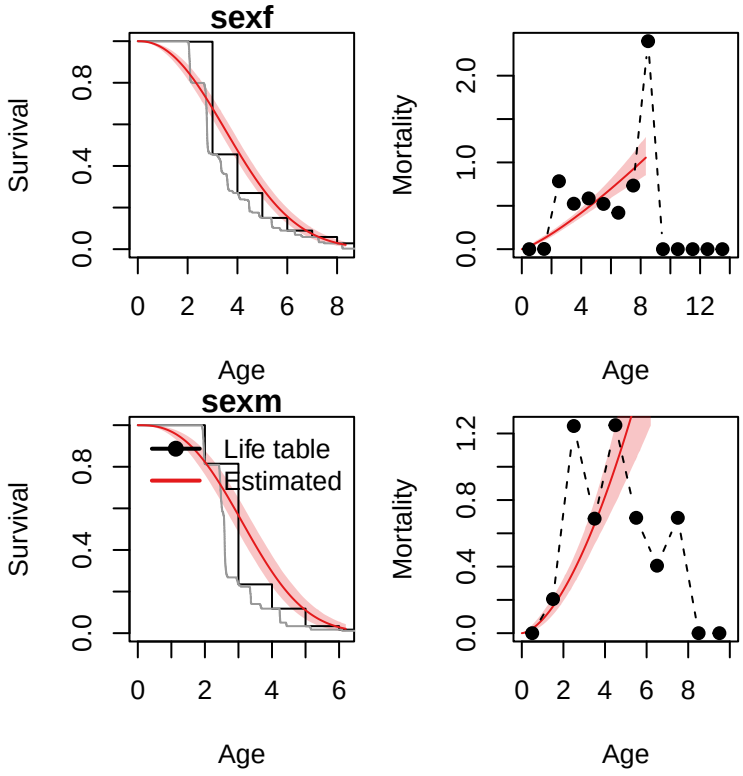
To plot the traces you can use the built-in function **plot** as:

```
plot(outSph1)
```



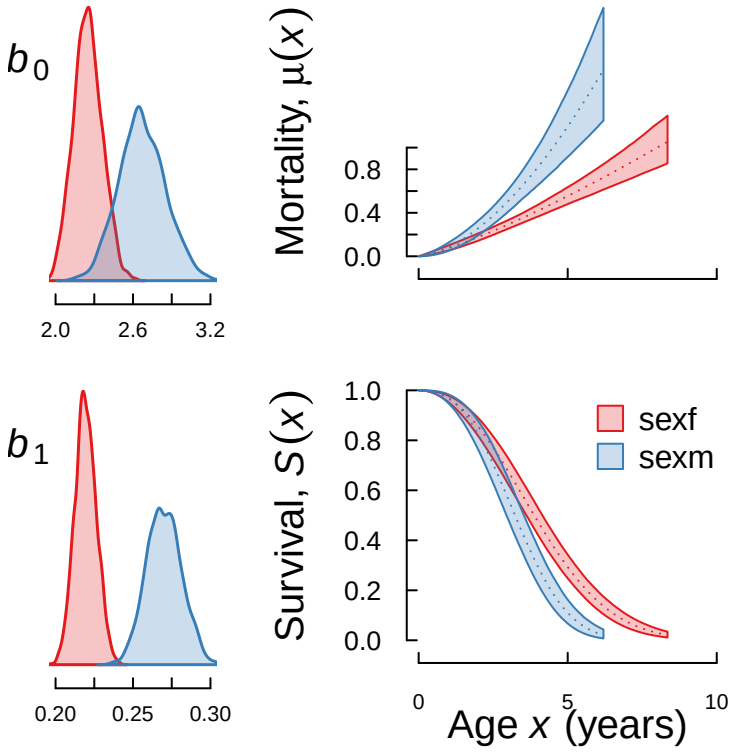

While you can visually inspect the goodness of fit as:

```
plot(outSph1, type = 'gof')
```




You can now also produce the "fancy" plots available in previous versions as:

```
plot(outSph1, type = 'fancy')
```




**Multiple runs**

You can use function **multibasta** to test multiple models and shapes. Here's an example comparing the Gompertz, Weibull and logistic models with all shapes:

```
# Run multiple models:
multiout <- multibasta(object = sphDat, studyStart = 1971, studyEnd = 1999,
                       models = c("GO", "WE", "LO"),
                       shapes = c("simple", "Makeham", "bathtub"),
                       formulaMort = ~ sex - 1,
                       parallel = TRUE, nsim = 4, ncpus = 4)
```

To inspect the results, use function **summary**, which provides a table with the sorted DICs, as:

```
# Summary of results:
summary(multiout)
```

```
>
> Call:
> Covars. structure         : 0
> Minimum age               : fused
> Cat. covars.              : sexf, sexm
> Cont. covars.             :
>
> Model settings:
>    niter   burnin thinning     nsim
>    22000     2001       40        4
>
> DICs:
>    model   shape D.ave D.mode  pD  k  DIC DICdiff Rank
```

```
> 3     WE  simple  2171   1749 211  4 2382    0.00    1
> 9     WE bathtub  2177   1761 208 10 2385    2.81    2
> 6     WE Makeham  2176   1721 228  6 2403   21.11    3
> 10    LO bathtub  2057   1219 419 12 2476   93.74    4
> 7     LO Makeham  2050   1180 435  8 2485  103.38    5
> 4     LO  simple  2046   1163 442  6 2488  105.74    6
> 8     GO bathtub  2359   2098 130 10 2489  107.34    7
> 2     GO  simple  2352   2078 137  4 2490  107.52    8
> 5     GO Makeham  2357   2059 149  6 2506  123.98    9
```

## Census data

One of the lateset additions is to include what can be described as **census** data. This is typical of studies where individuals are followed constantly (e.g., lab studies, captive populations, primate field studies).

### Loading and checking data

To illustrate the structure of the data, load the elephant data provided by Crawley et al. (2020):

```
# Elephant data (from Crawley et al (2020) Sci Reps):
eleDat <- read.csv(file = "03data/BaSTA/tables/elephantData.csv")
```

The structure of the data is as follows

Table 2: Example of census table for BaSTA based on the elephant dataset.

| ID | Birth.Date | Min.Birth.Date | Max.Birth.Date | Entry.Date | Depart.Date | Depart.Type | Sex |
|---|---|---|---|---|---|---|---|
| 1 | 1962-03-28 | 1962-03-28 | 1962-03-28 | 1962-03-28 | 2005-09-26 | D | Male |
| 2 | 1986-02-15 | 1986-02-15 | 1986-02-15 | 1986-02-15 | 2019-07-22 | C | Female |
| 3 | 1974-02-13 | 1974-02-13 | 1974-02-13 | 1974-02-13 | 2020-01-26 | D | Female |
| 4 | 1996-08-02 | 1996-08-02 | 1996-08-02 | 1996-08-02 | 2000-12-08 | C | Female |
| 5 | 1960-02-15 | 1960-02-15 | 1960-02-15 | 1960-02-15 | 2005-11-12 | D | Female |

As you can see in Table 2, the data uses several date columns in 'YYYY-mm-dd' format, three for the birth date, including a minimum and maximum birth date, and columns for the entry and departure dates. The minimum and maximum birth dates allow to account for ranges in the potential ages when the birth date is uncertain or unknown. The **Entry.Date** and **Depart.Date** columns, as their names suggest, show the first and last time the individual was recorded. When the entry date is later than the birth date, then the individual is treated as truncated. The column **Depart.Type** uses two categories: **C** for censored (i.e., still alive by the departure date), and **D** for death at the departure date.

To detect whether there are issues with the data, use function **DataCheck** as:

```
# Check data consistency:
eleCheck <- DataCheck(object = eleDat, dataType = 'census')
```

```
summary(eleCheck)
```

```
> DATA CHECK: 2025-09-16 06:09:51.231924
> ================================
>
> DATA SUMMARY:
> =============
```

```
> Total number of records  : 2962
> Number censored (C)      : 1031
> Number uncensored (D)    : 1931
> Number with unknown birth: 0
>
> NAs IN DATES COLUMNS:
> ---------------------
> No NAs found in dates columns.
>
> DATES RANGES:
> -------------
> Birth.Date    Min.Birth.Date  Max.Birth.Date  Entry.Date  Depart.Date
> 1950-01-01    1950-01-01  1950-01-01  1950-01-01  1950-05-06
> 2025-09-23    2025-09-23  2025-09-23  2025-09-23  2025-09-23
>
> INCONSISTENCIES BETWEEN DATES COLUMNS:
> --------------------------------------
> None.
>
> INCONSISTENCIES IN DEPARTURE TYPES:
> -----------------------------------
> None
```

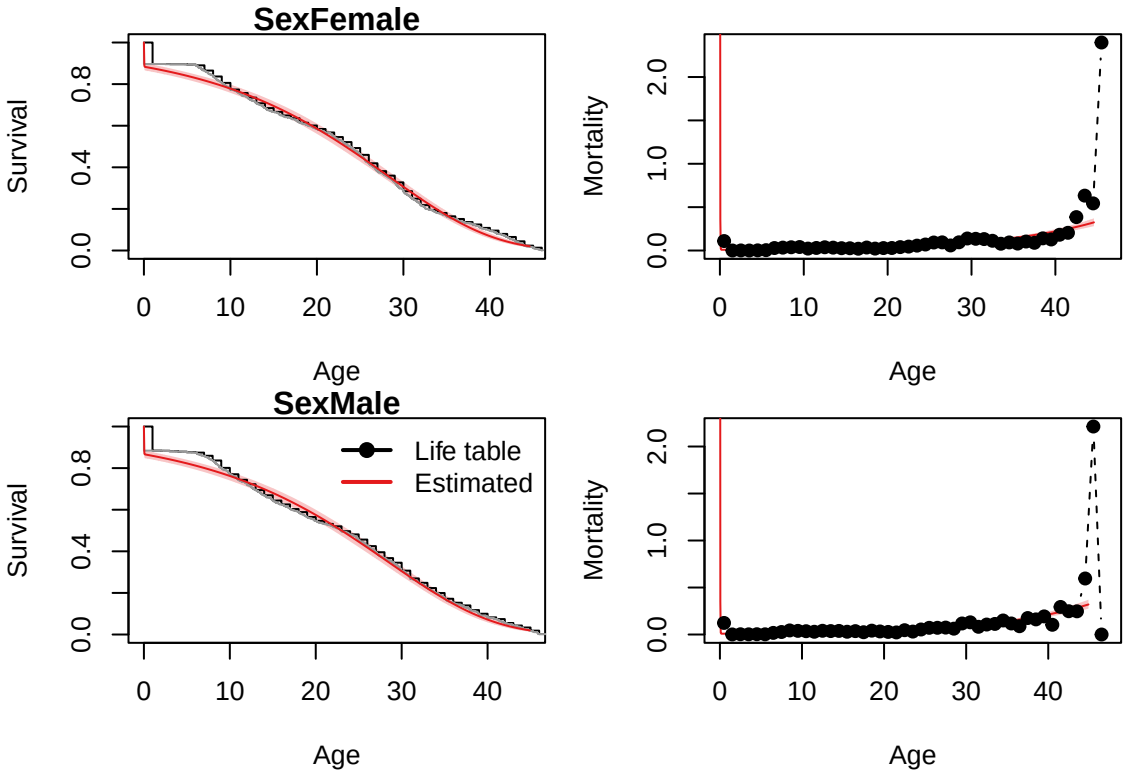
### Analysis from birth and older ages

To run a basic analysis using a Siler (1979) model (i.e., **model = "GO"** and **shape = "bathtub"**) with sex as a covariate, run function basta as:

```
# Siler model from birth:
outEle1 <- basta(object = eleDat, dataType = "census", model = "GO",
                 shape = "bathtub", formulaMort = ~ Sex - 1,
                 parallel = TRUE, ncpus = 4, nsim = 4)
```

While inspecting the goodness of fit graphically, you can see that the fit is poor for the first 10 years:

```
plot(outEle1, type = 'gof')
```

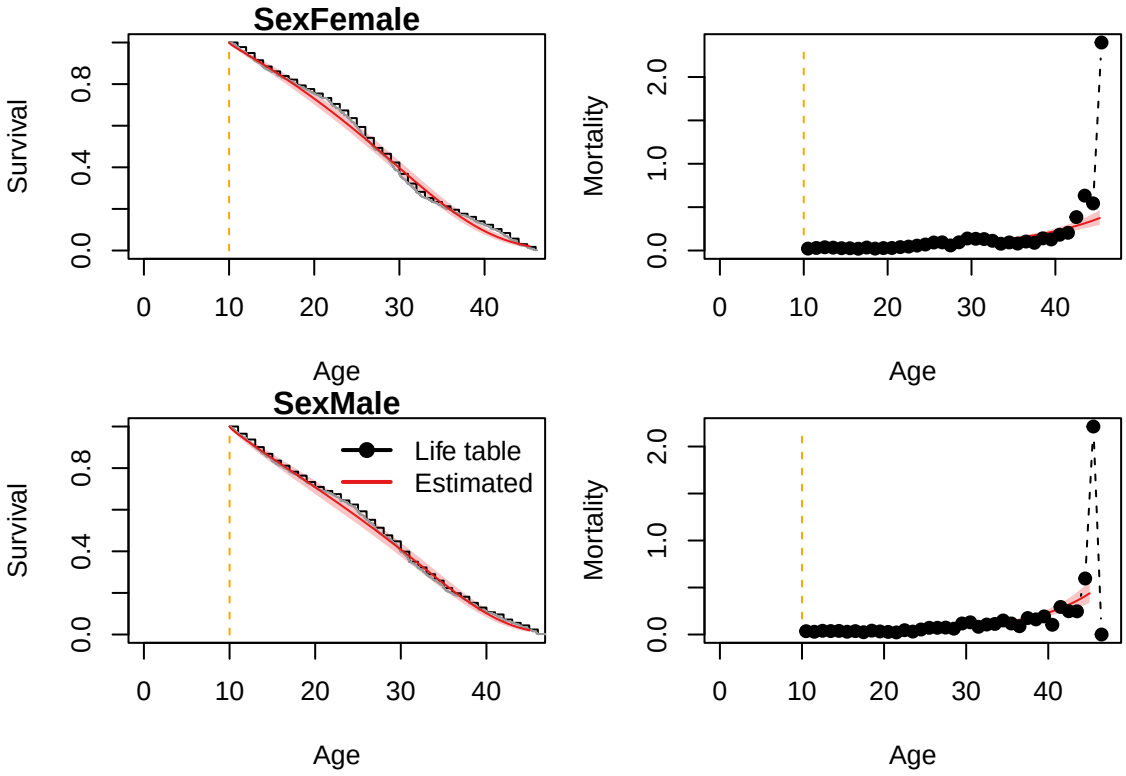



This is a common issue with many datasets for which infant and early mortality records may be unreliable. A way to improve the fit, particularly when interested on senescence, is to fit the mortality model to data at older ages using argument **minAge** as:

```
# Run with minAge = 10:
outEle2 <- basta(object = eleDat, dataType = "census", model = "GO",
                 shape = "bathtub", formulaMort = ~ Sex - 1, minAge = 10,
                 parallel = TRUE, ncpus = 4, nsim = 4)
```

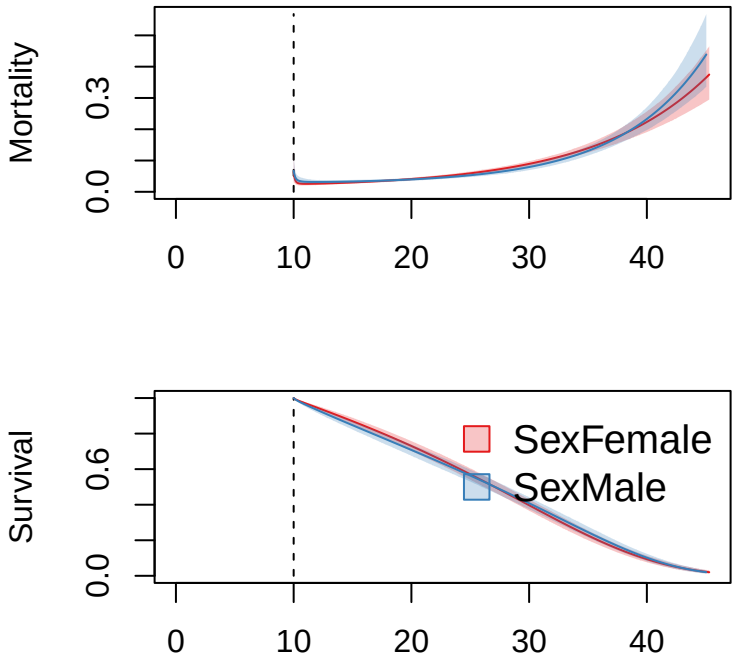
Which produces a considerably better goodness of fit plot:

```
plot(outEle2, type = 'gof')
```




To visualize the survival and mortality curves by categorical covariates use argument **type = "demorates"** as

```
plot(outEle2, type = 'demorates')
```



# BaFTA: Bayesian Fertility Trajectory Analysis

## Aggregated data

Here I provide data from 12 studies on the same number of mammalian species. You can find the metadata at "03data/BaFTA/tables/speciesFertMeta.csv' '. Load the data and extract the list of species as

```
# Load aggregated data:
datAggr <- read.csv(file = "03data/BaFTA/tables/speciesFert.csv")

# Extract species:
species <- unique(datAggr$species)
```

As an example, here is the data for northern fur seals (*Callorhinus ursinus*) from Lander (1998):

```
# Select one of the species:
isp <- 1

# Subset data:
spDat <- datAggr[which(datAggr$species == species[isp]), ]
```

The aggregated data needed to run BaFTA are:

Table 3: Example of aggregated data table for BaFTA based on the northern fur seal data. Only the first five rows are shown.

| Age | Fertility | nParents | nOffspring |
|---|---|---|---|
| 4 | 0.04 | 1450 | 58 |
| 5 | 0.37 | 1303 | 483 |
| 6 | 0.70 | 1217 | 852 |
| 7 | 0.80 | 1235 | 988 |
| 8 | 0.85 | 1270 | 1080 |

As shown in Table 3, the columns required are the **Age**, commonly in years, **nParents** for the number of adult individuals available at the corresponding age, **nOffspring** for the number of offspring produced by parents of that age, and **Fertility** commonly calculated as the ratio between the offspring and parents. Note that inference is carried out on the **nParents** and **nOffspring** columns; the **Fertility** column is only use as references.

### Single run

To run an analysis using the gamma function, simply use function **bafta** as:

```
# Quadratic:
outAggr1 <- bafta(object = spDat, model = "gamma", nsim = 6, ncpus = 6)
```

Note that it is not necessary to specify **dataType = "aggregated" since it is the default. You can print the output to the screen:

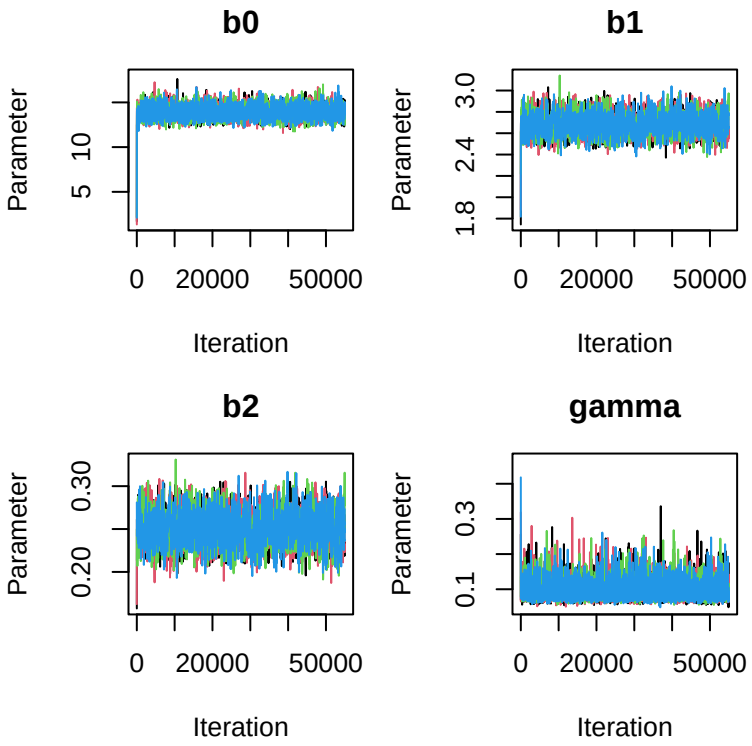
```
outAggr1
```

```
>
> Call:
> Model             : gamma
> Data type         : aggregated
```

```
> Num. iterations   : 55000
> Burnin            : 5001
> Thinning          : 20
> Number of sims.   : 4
> Computing time    : 11.45 secs
>
> Coefficients:
>           Mean      SD    Lower   Upper Neff  Rhat
> b0    13.92966 0.67957 12.68534 15.3622 4443 1.000
> b1     2.68495 0.09275  2.50902  2.8741 2759 1.001
> b2     0.25034 0.01853  0.21504  0.2883 2632 1.001
> gamma  0.09386 0.02657  0.06397  0.1634 7981 1.000
>
> Convergence:
> All parameter chains converged.
>
> Model fit:
>
> DIC = 218.31
>
> Predictive loss:
>  Good. Fit Penalty Deviance
>     122922  117549   240471
```

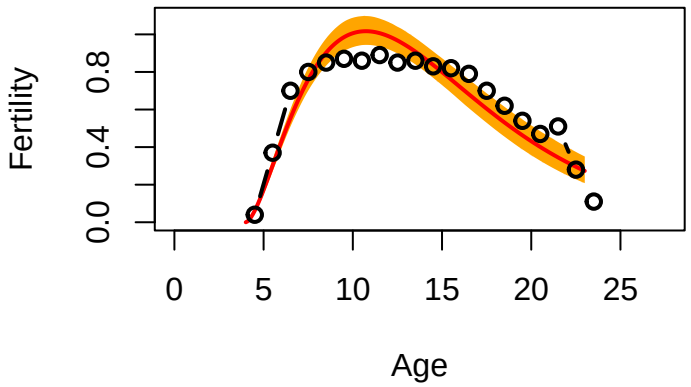
To inspect the traces, use the built-in function **plot** as:

```
plot(outAggr1)
```



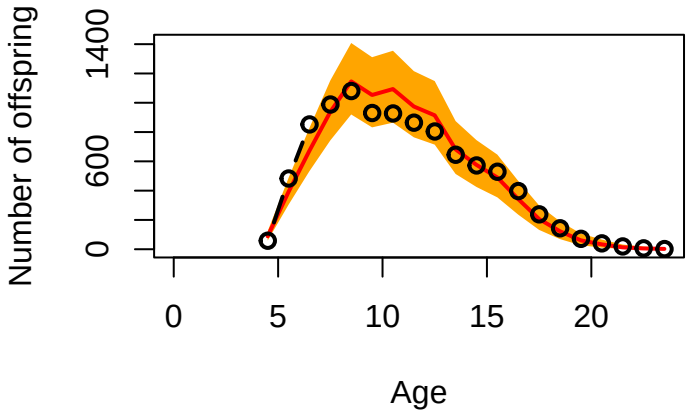

To inspect the fit to the Fertility data, modify the function with argument **type** as:

```
plot(outAggr1, type = "fertility")
```




while you can check the fit on the number of offspring as:

```
plot(outAggr1, type = "predictive")
```




### Multiple runs

As with BaSTA, in the extra code I provide a **multibafta** function. For illustration, let's select the models we wish to run on the aggregated data:

```
# Models:
models <- c("quadratic", "PeristeraKostaki", "gamma", "beta", "gammaMixture",
            "skewNormal")
```

We can then run the function as:

```
# Run multi-BaFTA:
multiout <- multibafta(object = spDat, models = models)
```

To visualize the output, simply print it to the console as:

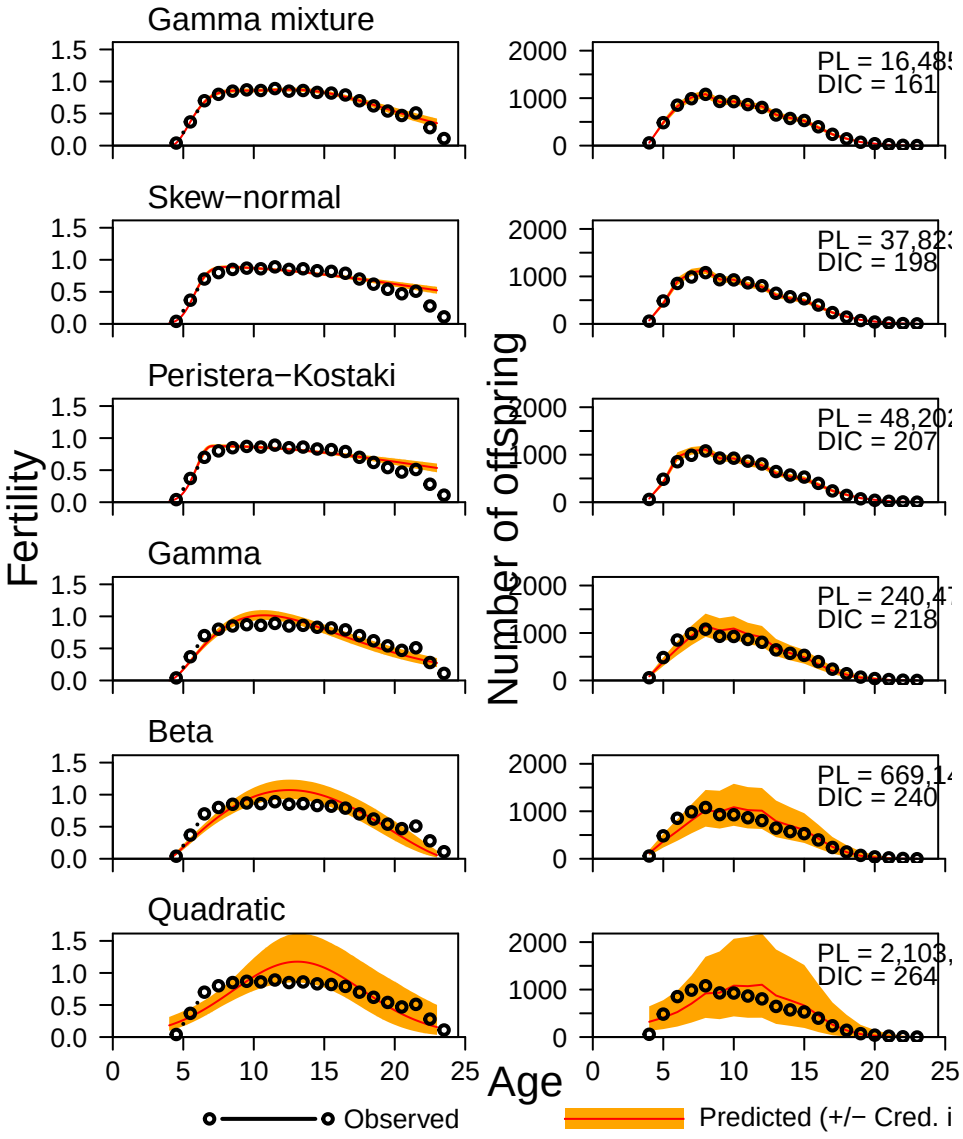
```
# Print output:
multiout
```

```
> Sorted by DIC:
>              model DIC PredLoss
> 5     gammaMixture 161    16485
> 6       skewNormal 198    37823
> 2 PeristeraKostaki 207    48202
> 3            gamma 218   240471
> 4             beta 240   669149
```

```
> 1        quadratic 264  2103748
>
> Sorted by PredLoss:
>              model DIC PredLoss
> 5     gammaMixture 161    16485
> 6       skewNormal 198    37823
> 2 PeristeraKostaki 207    48202
> 3            gamma 218   240471
> 4             beta 240   669149
> 1        quadratic 264  2103748
```

You can also use the built-in plot function to compare the fits of each model:

```
# Plot output:
plot(multiout, sortBy = "PredLoss")
```



## Individual level data with seasonal reproduction

Here I provide simulated data to illustrate how to prepare the data and run BaFTA on individual level data where reproduction is seasonal, and thus records are organized by age as integers. First load the data as:

```
# Load individual data:
datIndivSimp <- read.csv(file = "03data/BaFTA/tables/indivSimpFert.csv")
```

The data should include the following columns:

Table 4: Example of individual data with seasonal reproduction ('indivSimp') for BaFTA. Only the first five rows are shown.

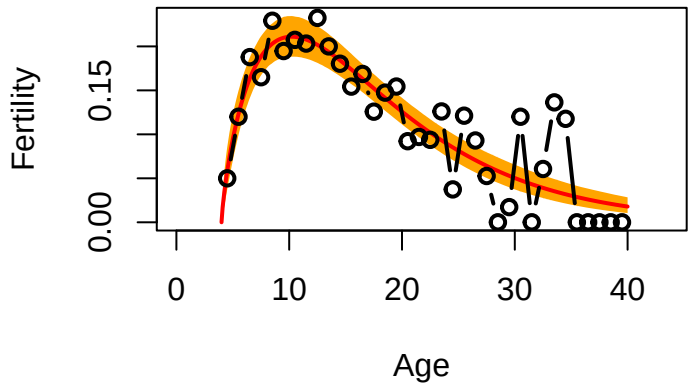
| indID | Age | nOffspring |
|---|---|---|
| I0002 | 4 | 0 |
| I0002 | 5 | 0 |
| I0002 | 6 | 0 |
| I0002 | 7 | 0 |
| I0002 | 8 | 0 |

As with the aggretated data, use function **bafta** to test for instance the gamma model as shown here:

```
# Run analysis:
outIndSimp1 <- bafta(object = datIndivSimp, dataType = "indivSimple",
                     model = "gamma", niter = 20000, burnin = 1001, nsim = 4,
                     ncpus = 4)
```

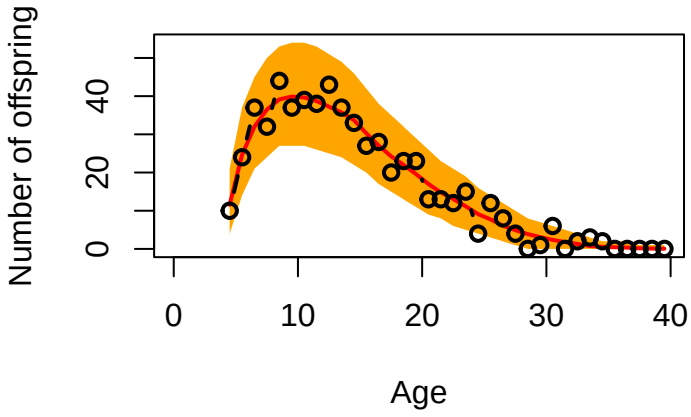
Note that we had to specify **dataType = 'indivSimple'**. To visually inspect the fit to the fertility data, use **plot** as:

```
plot(outIndSimp1, type = "fertility")
```




To check the goodness of fit, modify the plot as:

```
plot(outIndSimp1, type = "predictive")
```




### Individual level data with variable IBI

In many species, particularly mammals, reproduction is not seasonal, while inter-birth intervals (IBIs) can be variable. Here I provide an example on simulated data. Load the data as:

```
# Load individual data:
datIndivExt <- read.csv(file = "03data/BaFTA/tables/indivExtFert.csv")
```

The data includes the following columns:

Table 5: Example of individual data with continuous reproduction ('indivExt') and variable IBIs for BaFTA. Only the first five rows are shown.

| indID | Age | nOffspring | IBI | Time | First |
|---|---|---|---|---|---|
| I0020 | 6.251 | 0 | 1.251 | 1.251 | 0 |
| I0020 | 8.232 | 0 | 1.982 | 3.232 | 0 |
| I0020 | 9.530 | 0 | 1.298 | 4.530 | 0 |
| I0020 | 10.571 | 1 | 1.041 | 5.571 | 0 |
| I0020 | 11.863 | 0 | 1.292 | 6.863 | 0 |

As you can see from Table 5, there are three additional columns to those in Table 4. These are the time between reproductive events (**IBI**), the time since the minimum age at birth in the population (**Time**), and whether this was the first reproductive event (**First**). These additional columns are needed, since the model not only makes inferences on age-specific fertility, but also on the times to first offspring from the age at minimum birth, and on IBIs. To run, for example the Peristera-Kostaki (2007) model, modify the **dataType** and **model** arguments as:

```
# Run analysis:
outIndExt2 <- bafta(object = datIndivExt, dataType = "indivExtended",
              model = "PeristeraKostaki",
              niter = 20000, burnin = 1001, nsim = 4, ncpus = 4)
```

As with the other types of data, check the output with functions **print** or **summary** as:

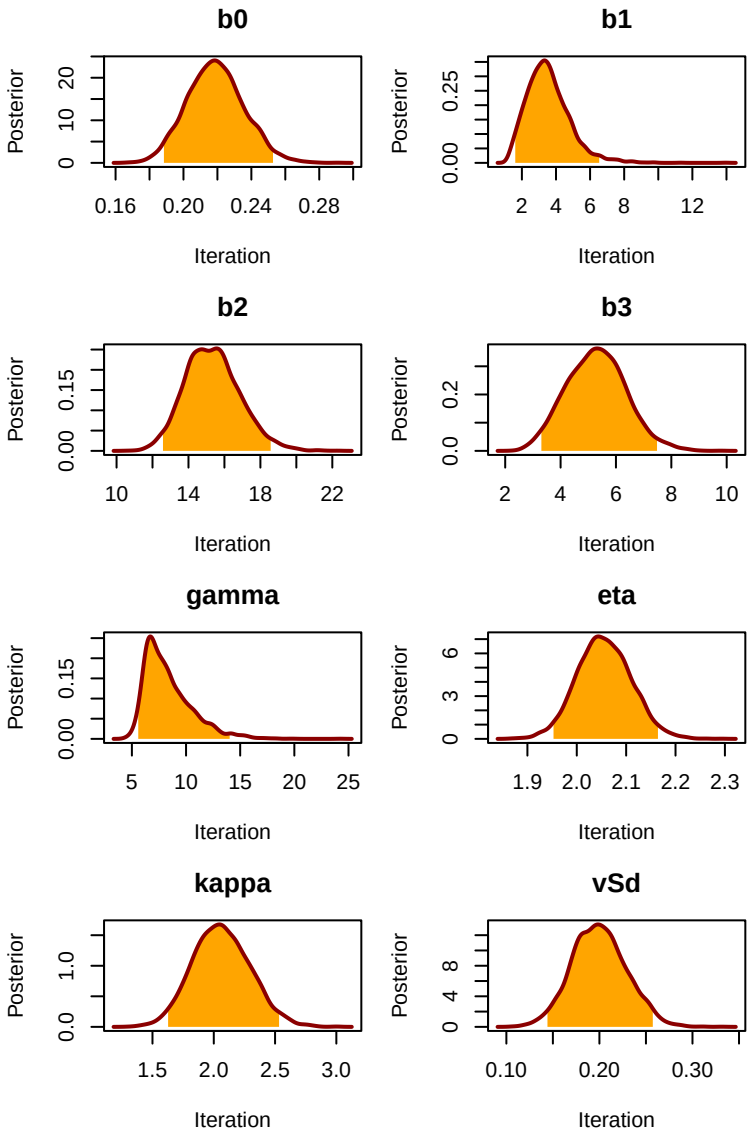
```
summary(outIndExt2, digist = 3)
```

```
>
> Call:
> Model             : PeristeraKostaki
> Data type         : indivExtended
```

```
> Num. iterations   : 20000
> Burnin            : 1001
> Thinning          : 20
> Number of sims.   : 4
> Computing time    : 3.36 mins
>
> Coefficients:
>          Mean      SD   Lower   Upper Neff  Rhat
> b0     0.2190 0.01663  0.1885  0.2530  171 1.000
> b1     3.5508 1.25793  1.5976  6.5438 1178 1.002
> b2    15.3379 1.51734 12.5797 18.5858 1729 1.000
> b3     5.2924 1.06398  3.3050  7.4975 1125 1.001
> gamma  8.3451 2.19242  5.5639 14.0497 3436 1.000
> eta    2.0556 0.05461  1.9525  2.1647 2743 1.001
> kappa  2.0634 0.23336  1.6253  2.5361 1266 1.000
> vSd    0.1998 0.02920  0.1441  0.2577 1026 1.001
>
> Convergence:
> All parameter chains converged.
>
> Model fit:
>
> DIC = 3169.02
>
> Predictive loss:
>  Good. Fit Penalty Deviance
>        316     314      630
```

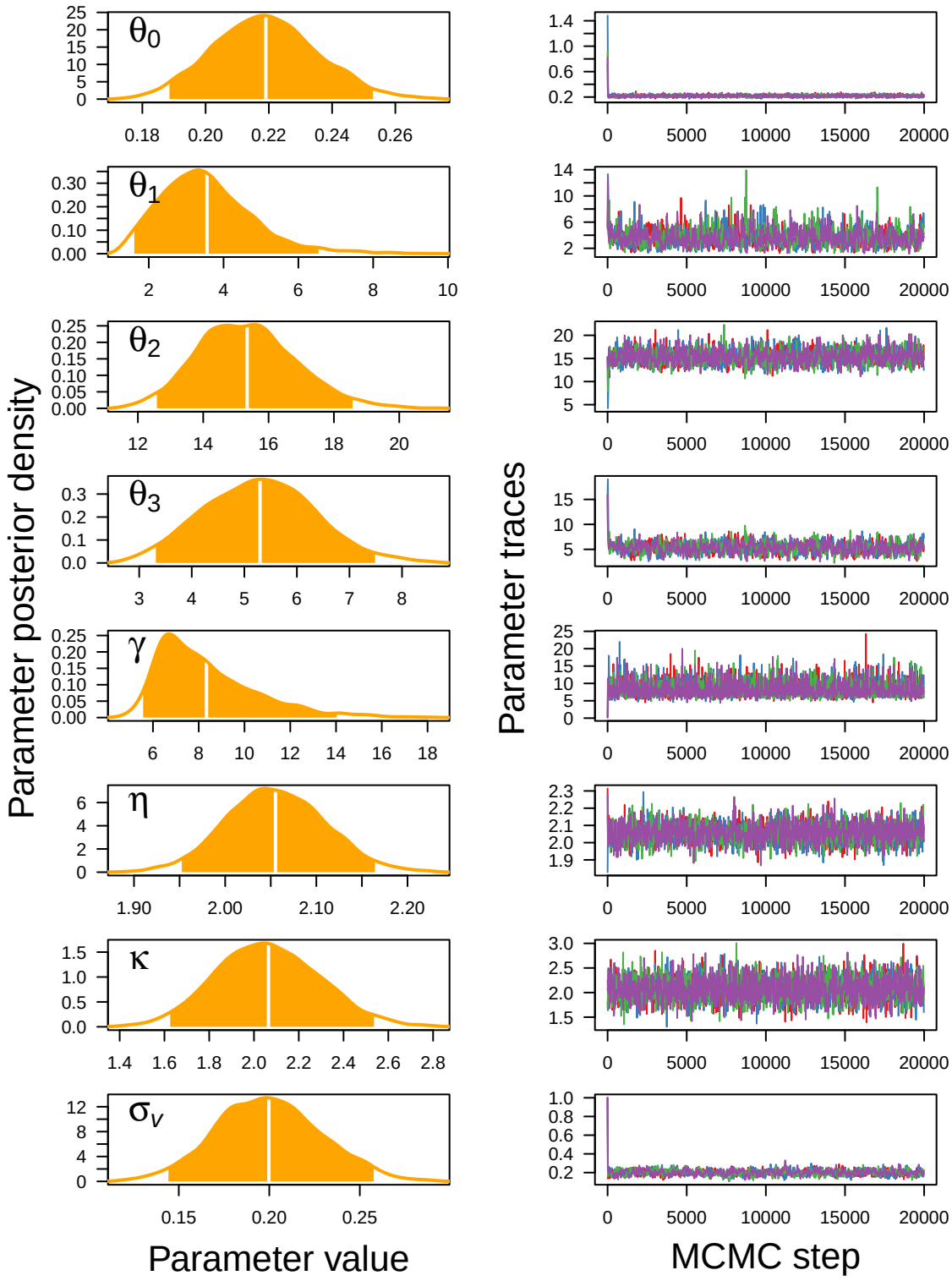
You can check the posterior density plots for the parameters as:

```
# Plot parameter posterior densities:
plot(outIndExt2, type = "density")
```




In the additional code, I provide a function that shows both the posterior densities and traces:

```
# Combined plot of posterior densities and traces:
PlotDensTrace(out = outIndExt2)
```


Parameter posterior density
Parameter traces
Parameter value
MCMC step
θ0
θ1
θ2
θ3
γ
η
κ
σv