Assessment for All initiative(a4a)
Modelling Individual Growth and Using Stochastic Slicing to Convert Length-based Data Into Age-based Data

Ernesto Jardim[1], Colin Millar[1,2,3], Finlay Scott[1], Chato Osio[1], and Iago Mosqueira[1]

[1]European Commission, Joint Research Centre, Sustainable resources directorate, Water and Marine Resources unit, 21027 Ispra (VA), Italy
[2]Marine Scotland Freshwater Laboratory, Faskally, Pitlochry, Perthshire PH16 5LB, UK
[3]International Council for the Exploration of the Sea (ICES), H. C. Andersens Boulevard 44-46, 1553 Copenhagen V, Denmark
[*]Corresponding author ernesto.jardim@jrc.ec.europa.eu

May 25, 2018

Abstract

The document explains the approach being developed by a4a to integrate uncertainty in individual growth into stock assessment and advice. It presents a mixture of text and code, where the first explains the concepts behind the methods, while the last shows how these can be run with the software provided.

Contents

1 Backround 3

2 License, documentation and development status 3

3 Installing and loading libraries 3

4 How to read this document 3

5 a4aGr - The growth class 4

6 Adding uncertainty to growth parameters with a multivariate normal distribution 5

7 Adding uncertainty to growth parameters with a multivariate triangle distribution 9

8 Adding uncertainty to growth parameters with statistical copulas 12

9 Converting from length to age based data - the `l2a()` method 14

1 Backround

The a4a stock assessment framework is based on age dynamics. Therefore, to use length information it must be processed before it can be used in an assessment. The rationale is that the processing should give the analyst the flexibility to use a range of sources of information, *e.g.* literature or online databases, to grab information about the species growth model and the uncertainty about the model parameters.

Within the a4a framework this is handled using the *a4aGr* class. In this section we introduce the *a4aGr* class and look at the variety of ways that parameter uncertainty can be included.

2 License, documentation and development status

The software is released under the EUPL 1.1.

For more information on the a4a methodologies refer to Jardim, et.al, 2014, Millar, et.al, 2014 and Scott, et.al, 2016.

Documentation can be found at http://flr-project.org/FLa4a. You are welcome to:

- Submit suggestions and bug-reports at: https://github.com/flr/FLa4a/issues
- Send a pull request on: https://github.com/flr/FLa4a/
- Compose a friendly e-mail to the maintainer, see 'packageDescription('FLa4a')'

3 Installing and loading libraries

To run the FLa4a methods the reader will need to install the package and its dependencies and load them. Some datasets are distributed with the package and as such need to be loaded too.

```
# from CRAN
install.packages(c("copula", "triangle", "coda"))
# from FLR
install.packages(c("FLCore", "FLa4a"), repos = "http://flr-project.org/R")
```

```
# libraries
library(FLa4a)
library(XML)
library(reshape2)
library(latticeExtra)
# datasets
data(ple4)
data(ple4.indices)
data(ple4.index)
data(rfLen)
```

```
packageVersion("FLCore")
```

```
## [1] '2.6.5.9005'
```

```
packageVersion("FLa4a")
```

```
## [1] '2.0.0'
```

4 How to read this document

The target audience for this document are readers with some experience in R and some background on stock assessment.

The document explains the approach being developed by a4a for fish stock assessment and scientific advice. It presents a mixture of text and code, where the first explains the concepts behind the methods, while the last shows how these can be run with the software provided. Moreover, having the code allows the reader to copy/paste and replicate the analysis presented here.

The sections and subsections are as independent as possible, so it can be used as a reference document for the FLa4a.

Finally, this is a live document which will be updated and released often.

5 a4aGr - The growth class

The conversion of length data to age is performed through the use of a growth model. The implementation is done through the *a4aGr* class.

```
showClass("a4aGr")
```

```
## Class "a4aGr" [package "FLa4a"]
##
## Slots:
##
## Name:      grMod  grInvMod    params      vcov     distr      name
## Class:   formula   formula     FLPar     array character character
##
## Name:       desc     range
## Class: character   numeric
##
## Extends: "FLComp"
```

To construct an *a4aGr* object, the growth model and parameters must be provided. Check the help file for more information.

Here we show an example using the von Bertalanffy growth model. To create the *a4aGr* object it's necessary to pass the model equation ($length \sim time$), the inverse model equation ($time \sim length$) and the parameters. Any growth model can be used as long as it's possible to write the model (and the inverse) as an R formula.

```
vbObj <- a4aGr(grMod = ~linf * (1 - exp(-k * (t - t0))), grInvMod = ~t0 - 1/k *
    log(1 - len/linf), params = FLPar(linf = 58.5, k = 0.086, t0 = 0.001, units = c("cm",
    "year-1", "year")))

# Check the model and its inverse
lc = 20
predict(vbObj, len = lc)
```

```
##    iter
##          1
##   1 4.86575
```

```
predict(vbObj, t = predict(vbObj, len = lc)) == lc
```

```
##    iter
##       1
##   1 TRUE
```

The predict method allows the transformation between age and lengths using the growth model.

```
predict(vbObj, len = 5:10 + 0.5)
```

```
##    iter
##          1
##   1 1.149080
##   2 1.370570
##   3 1.596362
##   4 1.826625
##   5 2.061540
##   6 2.301299
```

```
predict(vbObj, t = 5:10 + 0.5)
```

```
##    iter
##          1
##   1 22.04376
##   2 25.04796
##   3 27.80460
##   4 30.33408
##   5 32.65511
##   6 34.78488
```

6 Adding uncertainty to growth parameters with a multivariate normal distribution

Uncertainty in the growth model is introduced through the inclusion of parameter uncertainty. This is done by making use of the parameter variance-covariance matrix (the `vcov` slot of the *a4aGr* class) and assuming a distribution. The numbers in the variance-covariance matrix could come from the parameter uncertainty from fitting the growth model parameters.

Here we set the variance-covariance matrix by scaling a correlation matrix, using a cv of 0.2. Based on

$$\rho_{x,y} = \frac{\Sigma_{x,y}}{\sigma_x \sigma_y}$$

and

$$CV_x = \frac{\sigma_x}{\mu_x}$$

```
# Make an empty cor matrix
cm <- diag(c(1, 1, 1))
# k and linf are negatively correlated while t0 is independent
cm[1, 2] <- cm[2, 1] <- -0.5
# scale cor to var using CV=0.2
cv <- 0.2
p <- c(linf = 60, k = 0.09, t0 = -0.01)
vc <- matrix(1, ncol = 3, nrow = 3)
l <- vc
l[1, ] <- l[, 1] <- p[1] * cv
k <- vc
k[, 2] <- k[2, ] <- p[2] * cv
t <- vc
t[3, ] <- t[, 3] <- p[3] * cv
mm <- t * k * l
diag(mm) <- diag(mm)^2
mm <- mm * cm
# check that we have the intended correlation
all.equal(cm, cov2cor(mm))
```

```
## [1] TRUE
```

Create the a4aGr object as before but now we also include the vcov argument for the variance-covariance matrix.

```
vbObj <- a4aGr(grMod = ~linf * (1 - exp(-k * (t - t0))), grInvMod = ~t0 - 1/k *
    log(1 - len/linf), params = FLPar(linf = p["linf"], k = p["k"], t0 = p["t0"],
    units = c("cm", "year-1", "year")), vcov = mm)
```

First we show a simple example where we assume that the parameters are represented using a multivariate normal distribution.

```
# Note that the object we have just created has a single iteration of each
# parameter
vbObj@params
```

```
## An object of class "FLPar"
## params
##  linf     k    t0
## 60.00  0.09 -0.01
## units:  cm year-1 year
```

```
dim(vbObj@params)
```

```
## [1] 3 1
```

```
# We simulate 10000 iterations from the a4aGr object by calling mvrnorm()
# using the variance-covariance matrix we created earlier.
vbNorm <- mvrnorm(10000, vbObj)
# Now we have 10000 iterations of each parameter, randomly sampled from the
# multivariate normal distribution
vbNorm@params
```

```
## An object of class "FLPar"
## iters:  10000
##
## params
##                linf                   k                  t0
## 59.892591(11.80818)  0.090222( 0.01774) -0.010025( 0.00203)
## units:  cm year-1 year
```

```
dim(vbNorm@params)
```

```
## [1]     3 10000
```

We can now convert from length to ages data based on the 10000 parameter iterations. This gives us 10000 sets of age data. For example, here we convert a single length vector using each of the 10000 parameter iterations:

```
ages <- predict(vbNorm, len = 5:10 + 0.5)
dim(ages)
```

```
## [1]     6 10000
```

```
# We show the first ten iterations only as an illustration
ages[, 1:10]
```

```
##    iter
##           1        2        3        4        5        6        7
##   1 1.009617 1.039485 1.473891 1.496597 1.359310 1.314266 1.550928
##   2 1.205005 1.239009 1.763697 1.785596 1.624958 1.573438 1.856156
##   3 1.403487 1.441438 2.060388 2.079550 1.896329 1.839074 2.168817
##   4 1.605161 1.646860 2.364299 2.378631 2.173674 2.111504 2.489283
##   5 1.810133 1.855363 2.675790 2.683022 2.457261 2.391085 2.817954
##   6 2.018511 2.067041 2.995248 2.992915 2.747378 2.678204 3.155260
##    iter
##           8        9       10
##   1 0.7811929 1.070902 1.009409
##   2 0.9315880 1.279151 1.204656
##   3 1.0841260 1.491549 1.403219
##   4 1.2388688 1.708263 1.605211
##   5 1.3958811 1.929473 1.810755
##   6 1.5552305 2.155369 2.019976
```

The marginal distributions can be seen in Figure 1.

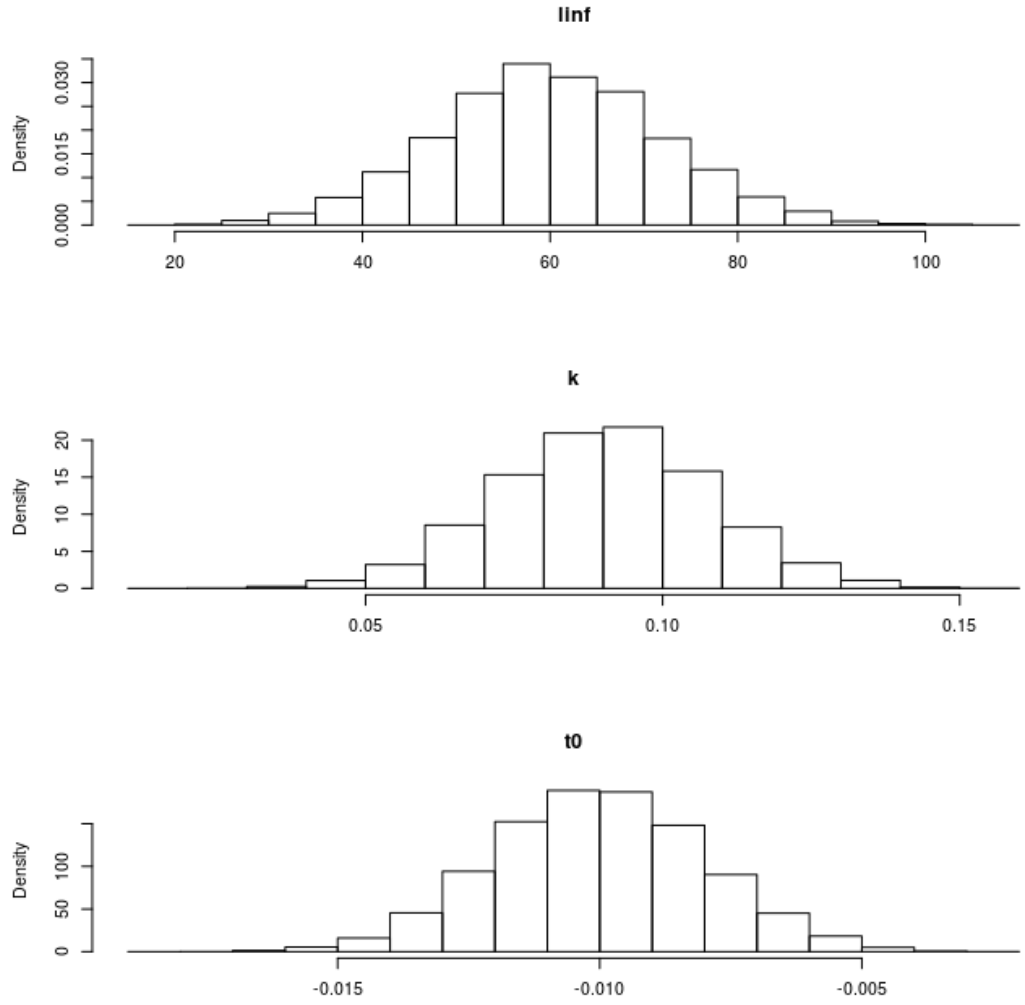

Figure 1: The marginal distributions of each of the parameters from using a multivariate normal distribution.

The shape of the correlation can be seen in Figure 2.

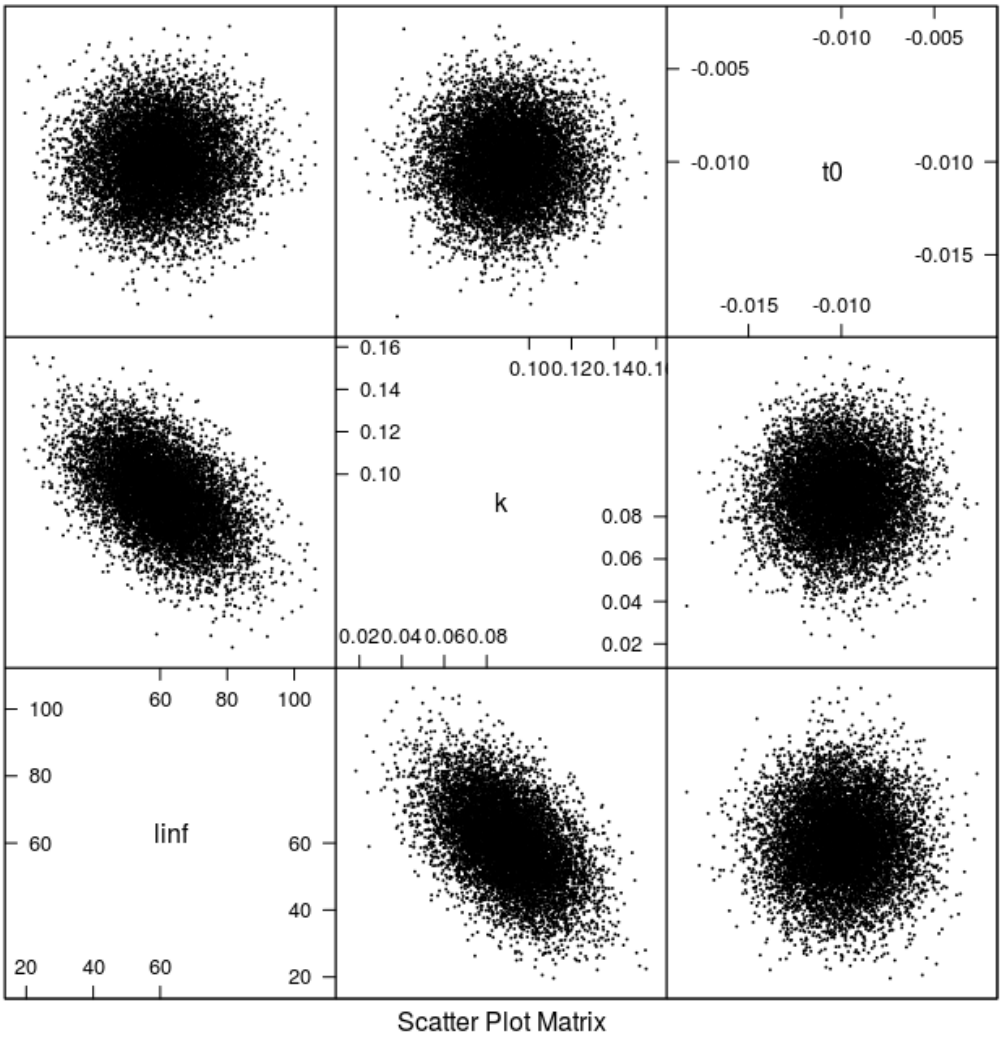

Figure 2: Scatter plot of the 10000 samples parameter from the multivariate normal distribution.

Growth curves for the 1000 iterations can be seen in Figure 3.

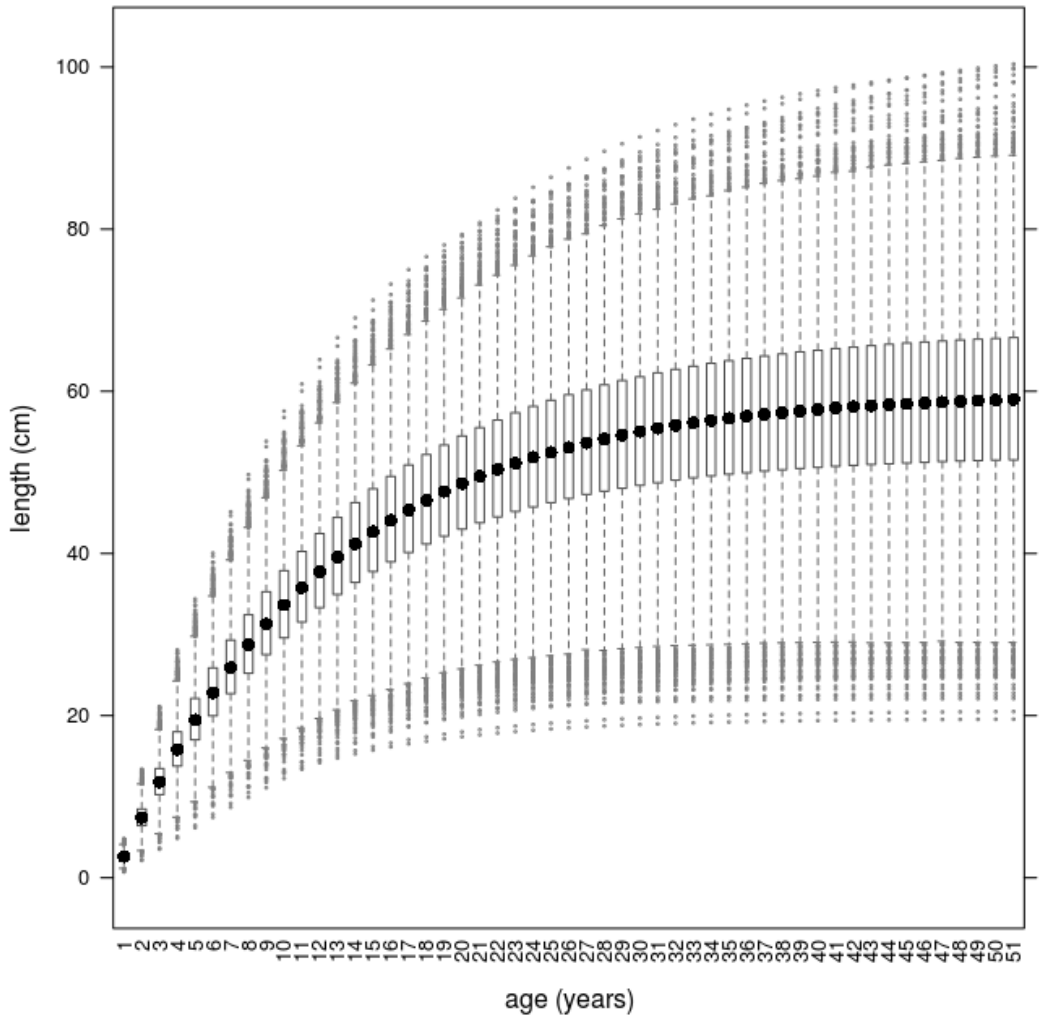

Figure 3: Growth curves using parameters simulated from a multivariate normal distribution.

7 Adding uncertainty to growth parameters with a multivariate triangle distribution

One alternative to using a normal distribution is to use a triangle distribution. We use the package `triangle`, where this distribution is parametrized using the minimum, maximum and median values. This can be very attractive if the analyst needs to scrape information from the web or literature and perform some kind of meta-analysis.

Here we show an example of setting a triangle distribution with values taken from Fishbase.

```
# The web address for the growth parameters for redfish (Sebastes
# norvegicus)
addr <- "http://www.fishbase.org/PopDyn/PopGrowthList.php?ID=501"
# Scrape the data
tab <- try(readHTMLTable(addr))
# Interrogate the data table and get vectors of the values
linf <- as.numeric(as.character(tab$dataTable[, 2]))
k <- as.numeric(as.character(tab$dataTable[, 4]))
t0 <- as.numeric(as.character(tab$dataTable[, 5]))
# Set the min (a), max (b) and median (c) values for the parameter as a list
# of lists Note that t0 has no 'c' (median) value. This makes the
# distribution symmetrical
```

```
triPars <- list(list(a = min(linf), b = max(linf), c = median(linf)), list(a = min(k),
    b = max(k), c = median(k)), list(a = median(t0, na.rm = T) - IQR(t0, na.rm = T)/2,
    b = median(t0, na.rm = T) + IQR(t0, na.rm = T)/2))
# Simulate 10000 times using mvrtriangle
vbTri <- mvrtriangle(10000, vbObj, paramMargins = triPars)
```

The marginals will reflect the uncertainty on the parameter values that were scraped from Fishbase but, as we don't really believe the parameters are multivariate normal, here we adopted a distribution based on a t copula with triangle marginals. The marginal distributions can be seen in Figure 4 and the shape of the correlation can be seen in Figure 5.

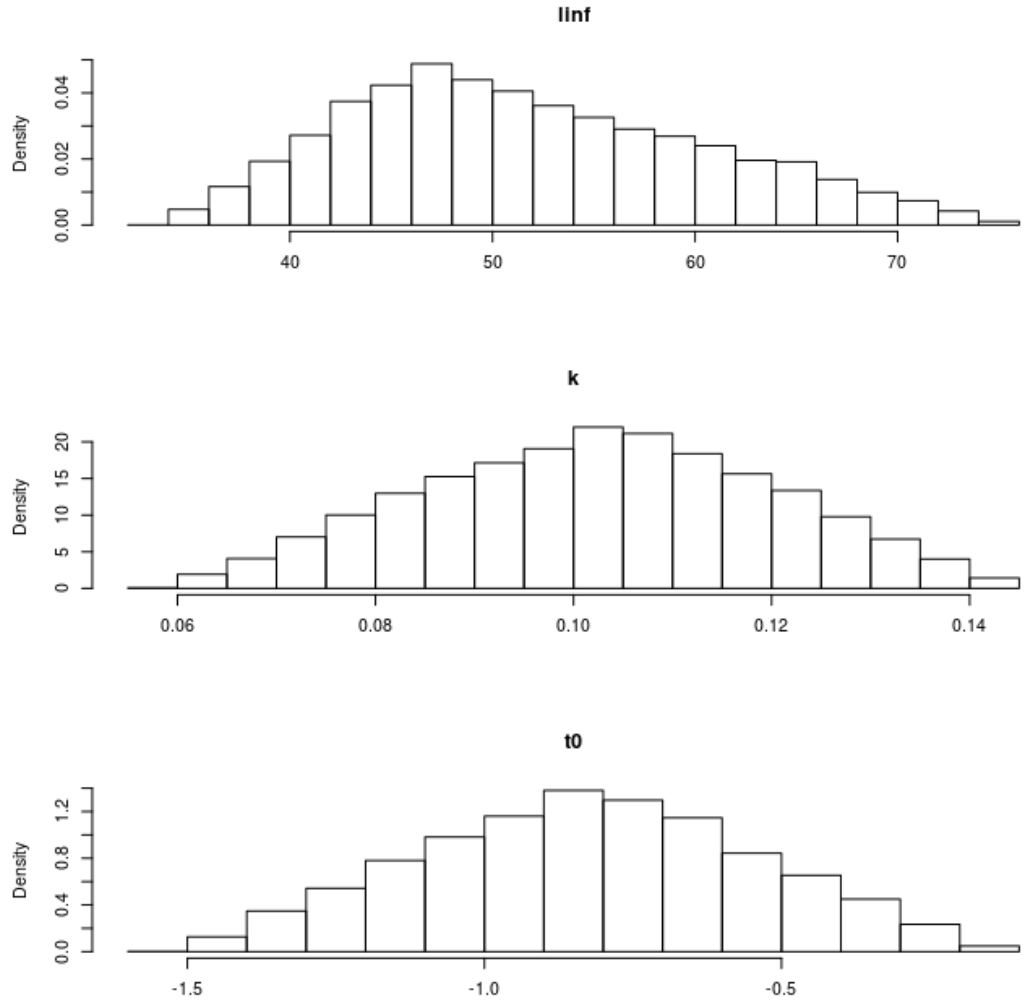

Figure 4: The marginal distributions of each of the parameters from using a multivariate triangle distribution.

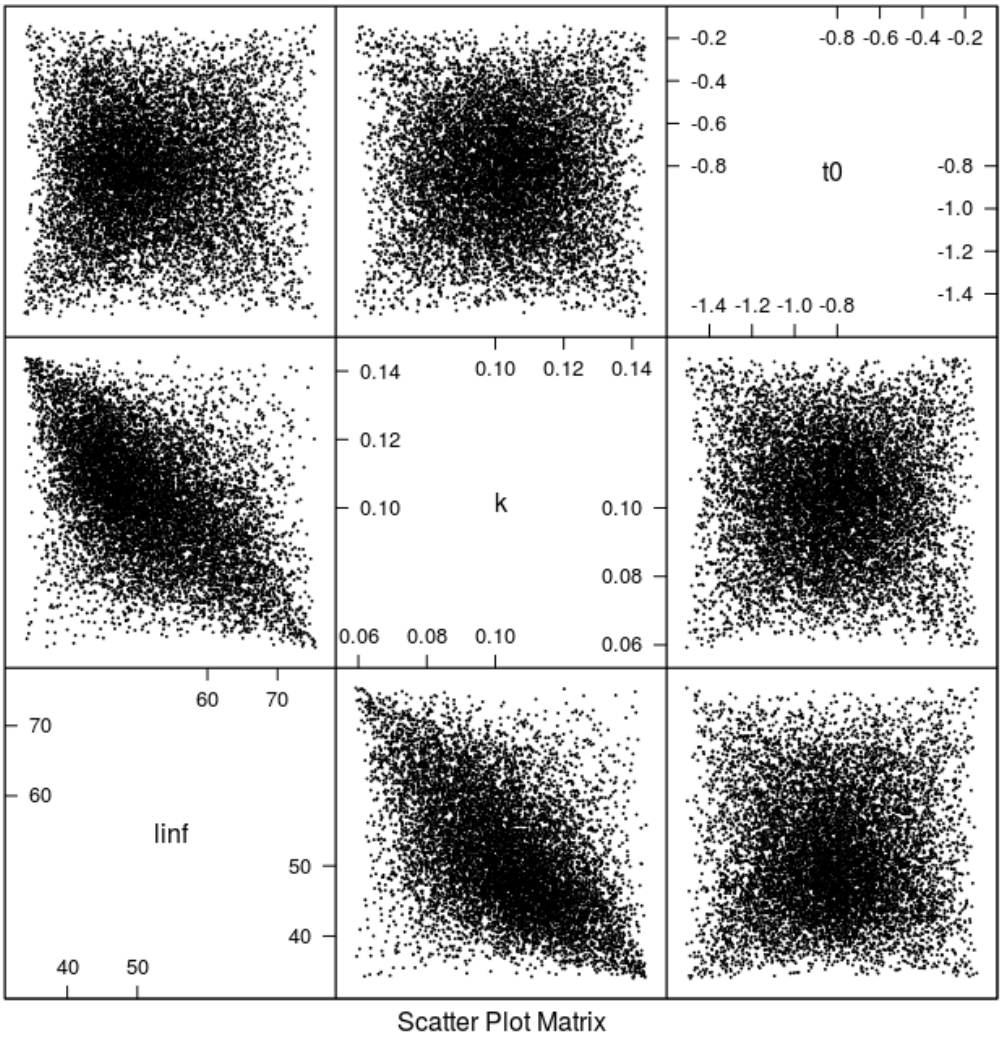

Figure 5: Scatter plot of the 10000 samples parameter from the multivariate triangle distribution.

We can still use `predict()` to see the growth model uncertainty (Figure 6).

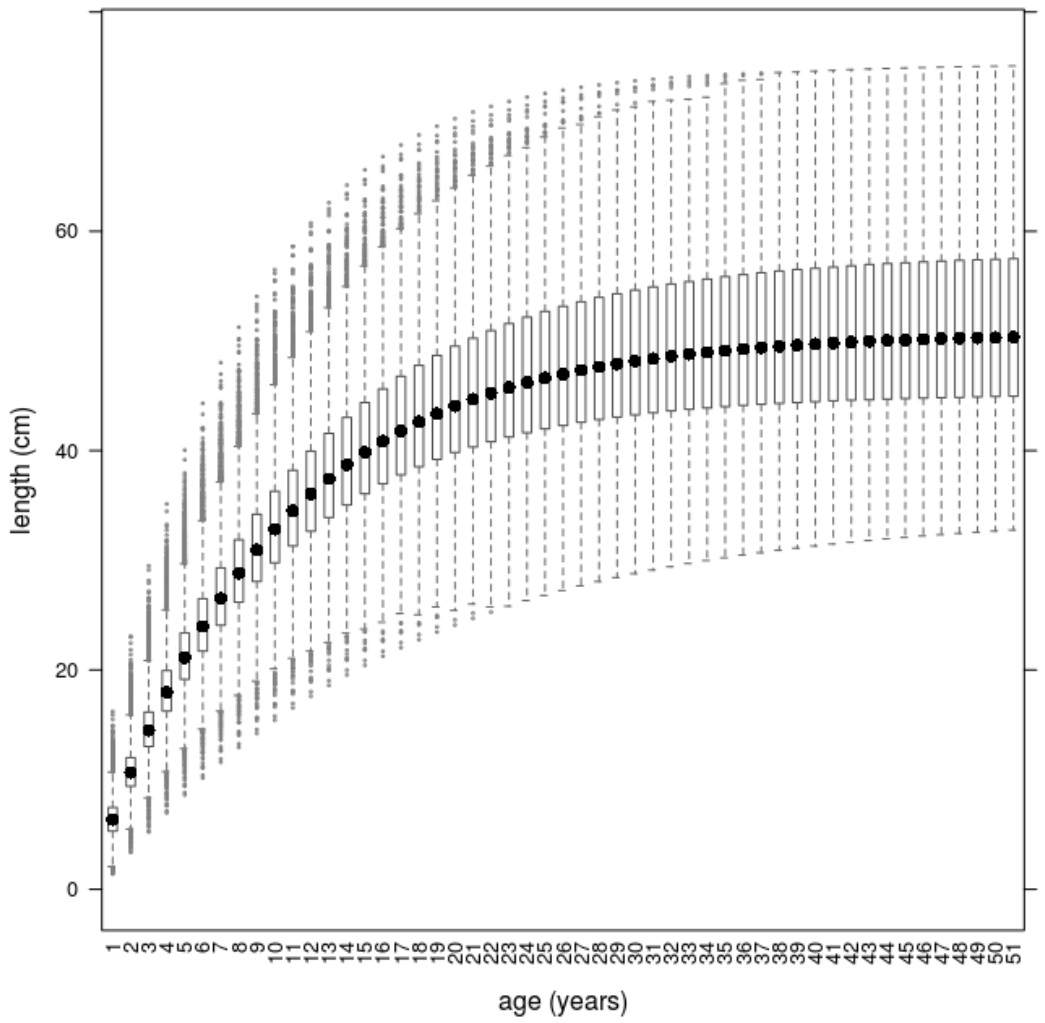

Figure 6: Growth curves using parameters simulated from a multivariate triangle distribution.

Remember that the above examples use a variance-covariance matrix that we essentially made up. An alternative would be to scrape the entire growth parameters dataset from Fishbase and compute the shape of the variance-covariance matrix yourself.

8 Adding uncertainty to growth parameters with statistical copulas

A more general approach to adding parameter uncertainty is to make use of statistical copulas and marginal distributions of choice. This is possible with the `mvrcop()` function borrowed from the package `copula`. The example below keeps the same parameters and changes only the copula type and family but a lot more can be done. Check the package `copula` for more information.

```
vbCop <- mvrcop(10000, vbObj, copula = "archmCopula", family = "clayton", param = 2,
    margins = "triangle", paramMargins = triPars)
```

The shape of the correlation changes (Figure 7) as well as the resulting growth curves (Figure 8).

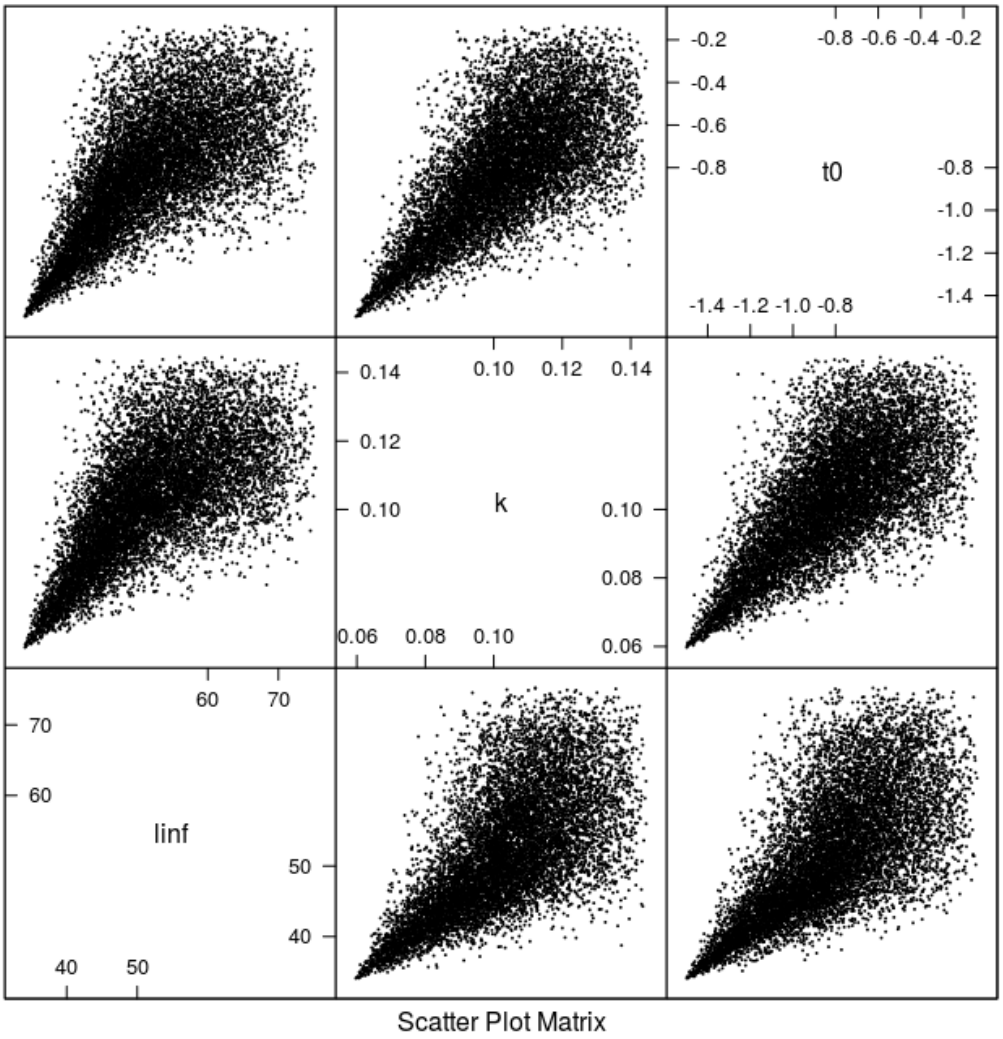

Figure 7: Scatter plot of the 10000 samples parameter from the using an archmCopula copula with triangle margins.

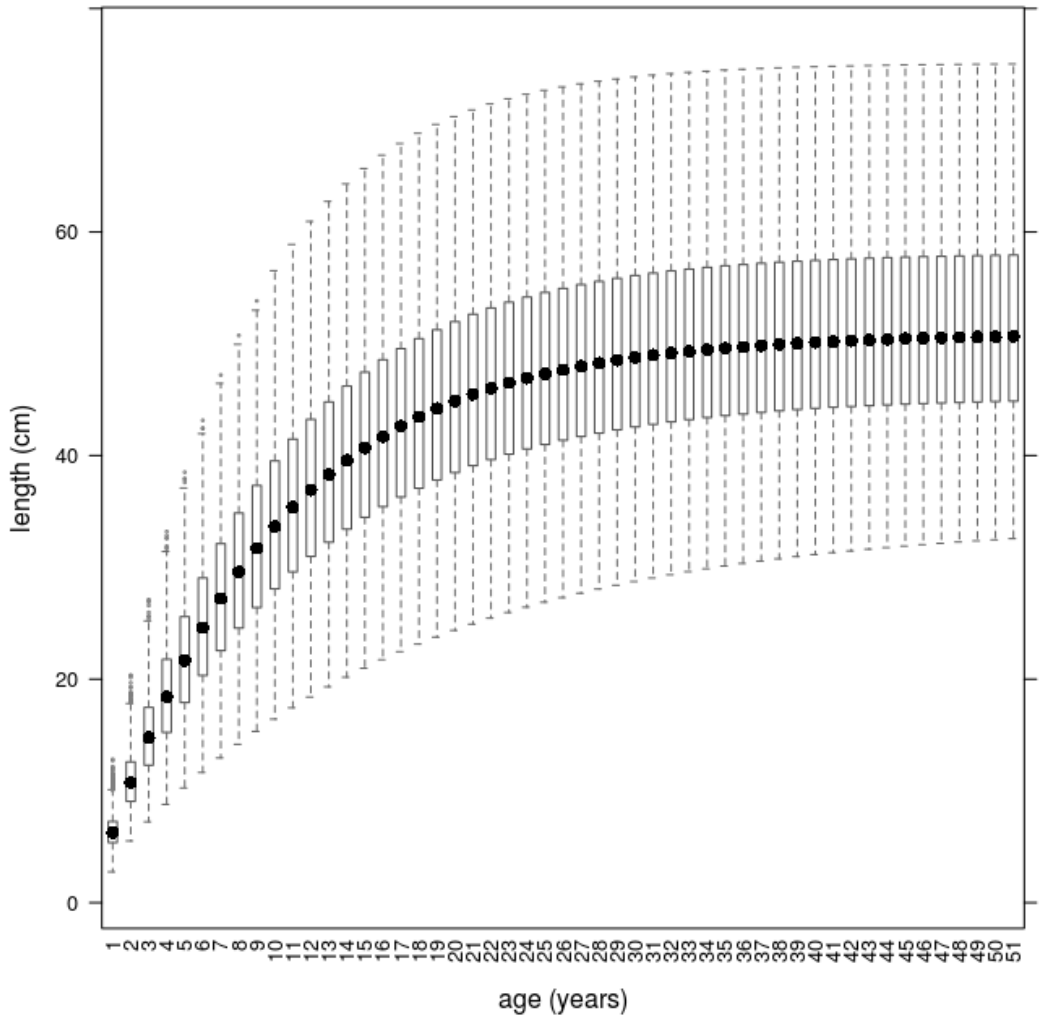

Figure 8: Growth curves from the using an archmCopula copula with triangle margins.

9 Converting from length to age based data - the `l2a()` method

After introducing uncertainty in the growth model through the parameters it's time to transform the length-based dataset into an age-based dataset. The method that deals with this process is `l2a()`. The implementation of this method for the *FLQuant* class is the main workhorse. There are two other implementations, for the *FLStock* and *FLIndex* classes, which are mainly wrappers that call the *FLQuant* method several times.

When converting from length-based data to age-based data you need to be aware of how the aggregation of length classes is performed. For example, individuals in length classes 1-2, 2-3, and 3-4 cm may all be considered as being of age 1 (obviously depending on the growth model). How should the values in those length classes be combined?

If the values are abundances then the values should be summed. Summing other types of values, such as mean weight, does not make sense. Instead these values are averaged over the length classes (possibly weighted by the abundance). This is controlled using the `stat` argument which can be either `mean` or `sum` (the default). Fishing mortality is not computed to avoid making wrong assumptions about the meaning of F at length.

We demonstrate the method by converting a catch-at-length *FLQuant* to a catch-at-age *FLQuant*. First we make an *a4aGr* object with a multivariate triangle distribution (using the parameters we set above). We use 10 iterations as an example. And call `l2a()` by passing in the length-based *FLQuant* and the *a4aGr* object.

```
vbTriSmall <- mvrtriangle(10, vbObj, paramMargins = triPars)
cth.n <- l2a(catch.n(rfLen.stk), vbTriSmall)
```

```
dim(cth.n)
```

```
## [1] 58 26  1  4  1 10
```

In the previous example, the *FLQuant* object that was sliced (`catch.n(rfLen.stk)`) had only one iteration. This iteration was sliced by each of the iterations in the growth model. It is possible for the *FLQuant* object to have the same number of iterations as the growth model, in which case each iteration of the *FLQuant* and the growth model are used together. It is also possible for the growth model to have only one iteration while the *FLQuant* object has many iterations. The same growth model is then used for each of the *FLQuant* iterations. As with all `FLR` objects, the general rule is *one or n* iterations.

As well as converting one *FLQuant* at a time, we can convert entire *FLStock* and *FLIndex* objects. In these cases the individual *FLQuant* slots of those classes are converted from length-based to age-based. As mentioned above, the aggregation method depends on the type of values the slots contain. The abundance slots (`*.n`, such as `stock.n`) are summed. The `*.wt`, `m`, `mat`, `harvest.spwn` and `m.spwn` slots of an *FLStock* object are averaged. The `catch.wt` and `sel.pattern` slots of an *FLIndex* object are averaged, while the `index`, `index.var` and `catch.n` slots are summed.

The method for *FLStock* classes takes an additional argument for the plusgroup.

```
aStk <- l2a(rfLen.stk, vbTriSmall, plusgroup = 14)
```

```
## Warning in .local(object, model, ...):  Individual weights, M and maturity will be (weighted)
averaged accross lengths,
##  harvest is not computed and everything else will be summed.
##  If this is not what you want, you'll have to deal with these slots by hand.
## Warning in .local(object, model, ...):  Some ages are less than 0, indicating a mismatch
between input data lengths
##  and growth parameters (possibly t0)
## Warning in .local(object, model, ...):  Trimming age range to a minimum of 0
```

```
## [1] "maxfbar has been changed to accomodate new plusgroup"
```

```
aIdx <- l2a(rfTrawl.idx, vbTriSmall)
```

```
## Warning in l2a(rfTrawl.idx, vbTriSmall):  Some ages are less than 0, indicating a mismatch
between input data lengths
##  and growth parameters (possibly t0)
## Warning in l2a(rfTrawl.idx, vbTriSmall):  Trimming age range to a minimum of 0
```

When converting with `l2a()` all lengths above Linf are converted to the maximum age, as there is no information in the growth model about how to deal with individuals larger than Linf.