AIFFD Chapter 5 - Age and Growth

Derek H. Ogle

Contents

This document contains R versions of the boxed examples from **Chapter 5** of the "Analysis and Interpretation of Freshwater Fisheries Data" book. Some sections build on descriptions from previous sections, so each section may not stand completely on its own. More thorough discussions of the following items are available in linked vignettes:

- the use of linear models in R in the preliminaries vignette,
- differences between and the use of type-I, II, and III sums-of-squares in the preliminaries vignette, and
- the use of "least-squares means" is found in the preliminaries vignette.

The following additional packages are required to complete all of the examples (with the required functions noted as a comment and also noted in the specific examples below).

```
> library(FSA)       # Summarize, Subset, headtail, fitPlot, lencat, alkIndivAge
> library(car)       # Anova, recode
> library(lsmeans)   # lsmeans
> library(nlstools)  # overview
```

In addition, external tab-delimited text files are used to hold the data required for each example. These data are loaded into R in each example with `read.table()`. Before using `read.table()` the working directory of R must be set to where these files are located on **your** computer. The working directory for all data files on **my** computer is set with

```
> setwd("C:/aaaWork/Web/fishR/BookVignettes/aiffd2007")
```

In addition, I prefer to not show significance stars for hypothesis test output, reduce the margins on plots, alter the axis label positions, and reduce the axis tick length. In addition, contrasts are set in such a manner as to force R output to match SAS output for linear model summaries. All of these options are set with

```
> options(width=90,continue=" ",show.signif.stars=FALSE, contrasts=c("contr.sum","contr.poly"))
> par(mar=c(3.5,3.5,1,1),mgp=c(2.1,0.4,0),tcl=-0.2)
```

5.1 Creating an Age-Length Key

Fisheries scientists often collect length data on large samples, but age data, because of the large amount of effort evolved, are generally collected on smaller samples (i.e., subsamples). In some cases, we wish to convert our length data to age data. We do this through the use of an age-length key. We start with a data set containing individual length and age data. By dividing length data into a series of discrete intervals, we can determine the frequency of ages within each interval. These frequencies are transformed into probabilities, which are later used to convert numbers at length to numbers at age. In this example, we have age (`age`) and length (`tl`) data for adult Spotted Sucker (*Minytrema melanops*). We create a series of length intervals and create a new variable (`LCat`) that is a discrete representation of the length data. In this case, we develop 2cm (20mm) length-groups and name each group by the low end of the interval. We then determine cell frequencies and calculate cell probabilities using the `lencat()` function.

5.1.1 Ogle Comment

In my opinion, the age-length key method shown in Box 5.1 of the text is cumbersome because of the tremendous amount of "if...then...else" statements and the fact that the user must re-enter data (i.e., the percentage at age for a given length interval when expanding the age-length key). In addition, the final result (table on p. 201) produces fractional fish in certain age-length categories. I recommend that the user consider the Isermann and Knight (2005) method for which Isermann and Knight (2005) provided a SAS program and I have provided an R program (age-length key chapter here). Nevertheless, I will attempt to recreate the process shown in Box 5.1.

5.1.2 Preparing Data

The aged and length data files are read and the structures are observed below.

```
> d1.age <- read.table("data/box5_1_Aged.txt",header=TRUE)
> str(d1.age)
```

```
'data.frame':   61 obs. of  3 variables:
 $ sex: Factor w/ 2 levels "F","M": 2 2 2 1 1 1 2 2 1 1 ...
 $ tl : int  100 111 114 99 104 120 250 255 250 252 ...
 $ age: int  1 1 1 1 1 1 2 2 2 2 ...
```

```
> d1.len <- read.table("data/box5_1_Length.txt",header=TRUE)
> str(d1.len)
```

```
'data.frame':   416 obs. of  1 variable:
 $ tl: int  336 336 336 395 395 395 395 386 386 386 ...
```

5.1.3 Constructing an Age-Length Key

The first step in constructing the age-length key is to create a variable that identifies the length interval category for each fish in the age sample. This variable is constructed, with default name `LCat`, and appended to the data frame containing the age-sample with `lencat()` from the `FSA` package. In this context, `lencat()` requires four arguments as described below.

- a formula of the form `~len` where `len` generically represents the length variable,
- `data`: the data frame containing the age-sample,

- `startcat`: a value identifying the starting length measurement category, and
- `w`: a value identifying the width of the length measurement categories.

The `lencat()` function returns a data frame that consists of the original data frame plus a variable containing the length interval categories for each fish. The default name of the new variable (`LCat`) can be changed with the `vname=` argument. The `lencat()` function result must be assigned to an object, preferably named differently from the original age sample.

It is important when using an age-length key to make sure that lengths in the age-sample span the same range as the lengths in the length- (i.e., unaged) sample. Unfortunately, this is not the case with the Spotted Sucker data set provided with Box 5.1. Nevertheless, one can find the minimum length in the age-sample using `Summarize()` from the `FSA` package.

```
> Summarize(d1.age$tl,digits=1)
```

```
      n     mean       sd      min       Q1   median       Q3      max percZero
   61.0    378.1    106.4     99.0    354.0    418.0    444.0    490.0      0.0
```

The length intervals can then start with an even-number 20-mm interval (the authors of Box 5.1 chose to use 20-mm wide intervals) just below the minimum length in the age-sample. In this example, one could start with either 80- or 90-mm as a start. I will choose to start with 80-mm to most closely match the work done in Box 5.1 (note that the authors of Box 5.1 used 90 but only had a 10-mm width on the first interval). The length intervals are then appended to the dataframe (which was renamed to `d1.age1`) with `lencat()`.

```
> d1.age1 <- lencat(~tl,data=d1.age,startcat=80,w=20)
> headtail(d1.age1)      # to verify the correct addition of the length categories
```

```
   sex  tl age LCat
1    M 100   1  100
2    M 111   1  100
3    M 114   1  100
59   F 486   9  480
60   F 485  10  480
61   F 490  10  480
```

Once the length category variable has been added to the age sample data frame, `xtabs()` is used to construct the summary contingency table of numbers of fish in each combined length and age category. The row variable (length category) is the first and the column variable (age) is the second part of the formula in the first argument to this function. The saved `table` object is then submitted as the first argument to `prop.table()` along withmargin=1‘ as a second argument (this is R’s way of saying “row”) to construct a row-proportions table. The resulting row-proportions table is the actual age-length key (as proportions and not percentages as shown in Box 5.1 in the text) determined from the age sample and is ready to be applied to the length sample.

```
> d.raw <- xtabs(~LCat+age,data=d1.age1)
> d.key <- prop.table(d.raw,margin=1)
> round(d.key,3)                          # rounded for display only
```

```
     age
LCat     1     2     3     4     5     6     7     8     9    10
  80 1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  100 1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
```

```
  120 1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  240 0.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  300 0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  320 0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  340 0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  360 0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  380 0.000 0.000 0.333 0.667 0.000 0.000 0.000 0.000 0.000 0.000
  400 0.000 0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000
  420 0.000 0.000 0.000 0.333 0.667 0.000 0.000 0.000 0.000 0.000
  440 0.000 0.000 0.000 0.000 0.286 0.357 0.000 0.214 0.143 0.000
  460 0.000 0.000 0.000 0.000 0.000 0.000 0.500 0.000 0.500 0.000
  480 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.333 0.667
```

Finally, it is important to replace all of the "blank" cells ("NA"s in R parlance) with zeroes. This is most easily accomplished with the code below where the code inside of the square brackets basically finds each position in the `d.key` matrix that has a value of "NA" and the entire code replaces these positions with zeroes.

```
> d.key[which(is.na(d.key))] <- 0
> round(d.key,3)                          # rounded for display only
```

```
     age
LCat     1     2     3     4     5     6     7     8     9    10
  80  1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  100 1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  120 1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  240 0.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  300 0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  320 0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  340 0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  360 0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  380 0.000 0.000 0.333 0.667 0.000 0.000 0.000 0.000 0.000 0.000
  400 0.000 0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000
  420 0.000 0.000 0.000 0.333 0.667 0.000 0.000 0.000 0.000 0.000
  440 0.000 0.000 0.000 0.000 0.286 0.357 0.000 0.214 0.143 0.000
  460 0.000 0.000 0.000 0.000 0.000 0.000 0.500 0.000 0.500 0.000
  480 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.333 0.667
```

The age-length key just computed does not match what is shown in Box 5.1 of the text for a variety of reasons. Most importantly, the age-length key shown in Box 5.1 of the text is a **column**- rather than **row**-proportions table. The row-proportions table shown here is correct. In addition, as noted above, the authors used percentages rather than proportions (which is inefficient because later, in their program, they divide each percentage by 100 to make it a proportion) and their first interval starts at 90-mm and is only 10-mm wide. In addition, I prefer to show the intermediate length intervals that do not contain any fish in the age-sample. This may be cumbersome with a wide range of lengths but it will help troubleshoot problems if the length-sample contains fish of these lengths (i.e., there will be no rule to say what age these fish should be).

5.1.4 Applying the Age-Length Key I

The first step in applying the age-length key is to construct the length frequency in the same 20-mm wide length intervals used to construct the age-length key. If the age- and length-samples span the same lengths

then one can apply `lencat()` as before, but to the length sample. As noted before, the length-sample for the Spotted Suckers contains lengths of fish that were not present in the age-sample and, thus, are not present in the age-length key. This can be seen by quickly summarizing the total lengths in the length-sample (look at the maximum value).

```
> Summarize(d1.len$tl,digits=2)
```

```
       n     mean       sd      min       Q1   median       Q3      max
  416.00   460.99    54.26   318.00   430.50   463.00   497.00   580.00
percZero
    0.00
```

Because of the different ranges of lengths in the two samples there is no relation in the age-length key that explains what ages fishes of the non-represented lengths should be. Thus, for the purposes of this example, the length sample will be reduced to only those fish with total lengths less than 500 mm to correspond with the lengths in the age-length key. The following use of `Subset()` from the `FSA` package. creates a new data frame (called `d1.len1`) containing only fish from the old data frame with a total length less than 500 mm.

```
> d1.len1 <- Subset(d1.len,tl<500)
```

The length intervals variable is then appended to this new data frame and the frequency of fish in each of the length intervals is found with `xtabs()`.

```
> d1.len2 <- lencat(~tl,data=d1.len1,startcat=80,w=20)
> ( len.freq <- xtabs(~LCat,data=d1.len2) )
```

```
LCat
300 320 340 360 380 400 420 440 460 480
  3   6  12  12  30  28  48  51  61  83
```

As explained in the text, the idea now is to apportion the number of fish in each length interval of the length-sample into age categories based on the relationships shown in the age-length key. So, for example, of the 3 300-mm fish in the length sample 100% should be assigned age-3 and of the 30 380-mm fish in the length sample 33.3% should be assigned age-3 and 66.7% should be assigned age-4.

As Box 5.1 in the text shows, these calculations can become tedious if you have to do all of these calculations individually. Fortunately, if you are very careful with the construction of the length intervals, these calculations can be greatly simplified with matrix multiplication. The multiplication of two matrices requires that the number of columns of the first matrix be the same as the number of rows of the second matrix. The resulting matrix will have as many rows as the first matrix and as many columns as the second matrix.

The length frequency "vector" can be thought of as a "matrix" with one row and 10 columns as computed with `length()` (and look at the `len.freq` vector above). The age-length key has 14 rows and 10 columns as computed with `dim()` (and look at the age-length key matrix above). The dimensions of these matrices imply that we can appropriately multiply the length-frequency vector by the age-length key matrix.

```
> length(len.freq)  # number of columns in length frequency vector
```

```
[1] 10
```

```
> dim(d.key)          # dimensions (rows, columns) of the age-length key
```

```
[1] 14 10
```

In matrix multiplication, the cell in the resulting matrix is the sum of the product of each element of the corresponding row in the first matrix and each element in the corresponding column of the second matrix. For example, the value in the cell of the first row and third column of the resulting matrix is the sum of the product of the elements in the first row of the first matrix and the third column of the second matrix.

In this situation, the resulting matrix will consist of one row with as many columns as ages in the age-length key where each value in the row is the sum of the length frequency values (i.e., the row of the first matrix) times the corresponding column of the age-length key. Thus, for example, the age-3 column of the resulting matrix would be found with,

```
> 0*0+0*0+0*0+0*0+0*0+0*0+0*0+0*0+0*0+0*0+0*0+0*0+0*3+1*6+1*12+1*12+1*30*0.333+28*0+48*0+
    51*0+61*0+83*0+0*0
```

```
[1] 42.99
```

If you study this closely you will see that it says "all three 300-mm fish are age-3, all six 320-mm fish are age-3, all twelve 340-mm fish are age-3, all twelve 360-mm fish are age-3, 33.3% of the 30 380-mm fish are age-3, none of the 40 400-mm fish are age-3, ..." which results in "an estimated 43 age-3 fish." This IS the calculation that we want in order to produce the final age-frequency for the individuals in the length-sample.

Matrix multiplication is accomplished in R with the matrix multiplication operator `%*%`. This operator must be preceded by the first matrix and succeeded by the second matrix to be multiplied. Note that you will get an error if the dimensions do not match as discussed above.

```
> ( age.freq <- len.freq %*% d.key )
```

```
Error in len.freq %*% d.key: non-conformable arguments
```

5.1.5 Applying the Age-Length Key II

As noted in the beginning, it is my opinion that the Isermann and Knight (2005) method is a better method for handling age-length keys. In this section, I will demonstrate how to apply this method to the Spotted Sucker data. This section assumes that the age-length key has already been constructed (as shown above) and the modified length sample (i.e., only those fish less than 500 mm) is being used.

The Isermann and Knight (2005) method is implemented with `alkIndivAge()` from the `FSA` package. This function requires the following arguments,

- `key`: A numeric matrix containing the age-length key (as constructed with `prop.table()`).
- `dl`: A data frame containing the length-sample of fish.
- `cl`: A number or character string indicating which column of `dl` contains the length measurements.
- `ca`: A number or character string indicating which column of `dl` should receive the age assignments. If the column does not exist in the current data frame then one will be appended with the name given in `ca`.

The `alkIndivAge()` function will determine the length categories to construct based on the age-length key sent in the `key=` argument. The results of `alkIndivAge()` should be assigned to an object, preferably with a name different from the original length sample.

```
> d1.len3 <- alkIndivAge(d.key,~tl,data=d1.len1)
> headtail(d1.len3)
```

```
     tl age
1   336   3
2   336   3
3   336   3
332 405   4
333 405   4
334 405   4
```

The original (not modified) age-sample data frame and the modified length-sample data frame (i.e., now containing the ages assigned via the age-length key) can then be row-bound together with `rbind()` to construct a data frame that consists of lengths and ages for all fish in the study.

```
> d1.comb <- rbind(d1.len3,d1.age[,c("tl","age")])
> headtail(d1.comb)
```

```
     tl age
1   336   3
2   336   3
3   336   3
393 486   9
394 485  10
395 490  10
```

The assigned ages in the `rb.comb` data frame can then be used to, for example, compute an overall age-frequency with `xtabs()` or calculate summary statistics of size-at age for ALL individuals in the study with `Summarize()`.

```
> xtabs(~age,data=d1.comb)
```

```
age
 1  2  3  4  5  6  7  8  9 10
 6  4 54 77 56 25 32 13 70 58
```

```
> Summarize(tl~age,data=d1.comb,digits=2)
```

```
   age  n   mean    sd min  Q1 median    Q3 max percZero
1    1  6 108.00  8.37  99 101  107.5 113.2 120        0
2    2  4 251.75  2.36 250 250  251.0 252.8 255        0
3    3 54 361.39 24.06 318 344  359.5 379.0 399        0
4    4 77 410.56 16.77 382 395  413.0 418.0 438        0
5    5 56 437.62  8.99 420 431  437.0 444.0 459        0
6    6 25 449.20  5.64 441 444  450.0 451.0 459        0
7    7 32 465.78  3.95 462 463  463.0 467.0 474        0
8    8 13 449.62  5.53 441 444  450.0 452.0 459        0
9    9 70 474.93 15.20 446 467  470.0 490.0 499        0
10  10 58 492.83  5.94 480 490  496.0 497.0 499        0
```

5.2 Determining Mean Back-Calculated Length at Age

In addition to providing estimates of age, hard parts are often used to back-calculate length at younger ages. To demonstrate how this is accomplished, we will be using a data set determined from scales and describing the age and growth of Spotted Sucker from the Savannah River. For each fish, our data set contains an identification number (`ID`), sex (`sex`), total length-at-capture (`Lc`), year of capture (`date`), age (`age`), radius of ageing structure (scale) at capture (`Sc`), annulus i (`inc`), and scale radius at each annulus i (`Si`) for each individual annulus.

5.2.1 Ogle Comment

The authors of Box 5.2 refer the reader to the review by Francis (1990) for back-calculation methods that are alternatives to the Dahl-Lea and Fraser-Lee methods. It is my opinion that Francis (1990) provides a definitive review of all back-calculation methods and I strongly suggest any fisheries biologist who plans to perform back-calculations read his review. The rest of the document below follows the analyses that were performed in Box 5.2.

5.2.2 Preparing Data

The Box5_2.txt is read and the structure and first six lines of the data frame are observed.

```
> d2 <- read.table("data/box5_2.txt",header=TRUE)
> str(d2)
```

```
'data.frame':   416 obs. of  8 variables:
 $ ID  : Factor w/ 63 levels "07447","200404111",..: 1 1 1 3 3 3 3 4 4 4 ...
 $ sex : Factor w/ 2 levels "F","M": 2 2 2 1 1 1 1 1 1 1 ...
 $ Lc  : int  336 336 336 395 395 395 395 386 386 386 ...
 $ date: int  2004 2004 2004 2004 2004 2004 2004 2004 2004 2004 ...
 $ age : int  3 3 3 4 4 4 4 4 4 4 ...
 $ Sc  : num  16.3 16.3 16.3 18.4 18.4 18.4 18.4 18.6 18.6 18.6 ...
 $ inc : int  1 2 3 1 2 3 4 1 2 3 ...
 $ Si  : num  5 12.9 16.3 4.8 9.9 16.5 18.4 4.9 8.5 13.6 ...
```

```
> head(d2)
```

```
     ID sex  Lc date age   Sc inc   Si
1 07447   M 336 2004   3 16.3   1  5.0
2 07447   M 336 2004   3 16.3   2 12.9
3 07447   M 336 2004   3 16.3   3 16.3
4 35334   F 395 2004   4 18.4   1  4.8
5 35334   F 395 2004   4 18.4   2  9.9
6 35334   F 395 2004   4 18.4   3 16.5
```

It is very important to note that this data file is already in what is called one-radii-per-line format. This is noted by the fact that the first three lines of the data file all pertain to fish number 07447. Note that all information in those three rows is the same except in the columns labeled `inc` and `Si` which are the previous age i (i.e., annulus number) and the scale radius at age i. It is common for the data to be collected in what is called one-fish-per-line format where all information for one fish is found in one row of the data file. The data must be in one-radii-per-line format to efficiently back-calculate length-at-age. Methods of converting from one format to the other format are described in Chapter 2 here.

The data must be in one-radii-per-line format for back-calculation methods.

For some back-calculation methods (e.g., Dahl-Lea) it is not important that the measurements made on the fish and those made on the scales are in the same units. However, a common set of units is required for other methods (e.g., Fraser-Lee). The scale and fish measurements should be converted to common units to guard against forgetting to do this later. The authors note that the fish measurements in this example are mm whereas the scale measurements are in cm at a magnification of 24X. Thus, the scale measurements are converted to actual mm (and stored in a new variable) as shown below.

```
> d2$Si2 <- d2$Si*10/24   # convert radial measurements
> d2$Sc2 <- d2$Sc*10/24   # convert total scale radius
> head(d2)
```

```
     ID sex  Lc date age   Sc inc   Si      Si2      Sc2
1 07447   M 336 2004   3 16.3   1  5.0 2.083333 6.791667
2 07447   M 336 2004   3 16.3   2 12.9 5.375000 6.791667
3 07447   M 336 2004   3 16.3   3 16.3 6.791667 6.791667
4 35334   F 395 2004   4 18.4   1  4.8 2.000000 7.666667
5 35334   F 395 2004   4 18.4   2  9.9 4.125000 7.666667
6 35334   F 395 2004   4 18.4   3 16.5 6.875000 7.666667
```

In general, the scale and length measurements should be in the same units (i.e., the same scale).

5.2.3 Dahl-Lea Method

We start with the simple case in which the growth of the structure used for ageing is directly proportional to the growth of the fish. This method is generally referred to as the Dahl-Lea method (Dahl 1907, Lea1910) and allows one to back-calculate length at age for individual fish. The formula is

$$L_i = L_c \frac{S_i}{S_c}$$

where L_i is back-calculated length at annulus i, L_c is length at capture, S_i is ageing-structure radius at annulus i, and S_c is ageing-structure radius at capture. Using the R code below, we can generate back-calculated total lengths (L_i) and calculate mean length at age for the Spotted Sucker population.

The Dahl-Lea method of back-calculating fish length at previous age (and stored in a variable called `Li1`) can be accomplished very simply with an R expression (note the use of the modified scale measurement variables). Summary statistics (similar to the table shown on page 206 of Box 5.2 in the text) can be computed with `Summarize()`.

```
> d2$LiDL <- d2$Lc*d2$Si2/d2$Sc2
> head(d2)
```

```
     ID sex  Lc date age   Sc inc   Si      Si2      Sc2     LiDL
1 07447   M 336 2004   3 16.3   1  5.0 2.083333 6.791667 103.0675
2 07447   M 336 2004   3 16.3   2 12.9 5.375000 6.791667 265.9141
3 07447   M 336 2004   3 16.3   3 16.3 6.791667 6.791667 336.0000
4 35334   F 395 2004   4 18.4   1  4.8 2.000000 7.666667 103.0435
5 35334   F 395 2004   4 18.4   2  9.9 4.125000 7.666667 212.5272
6 35334   F 395 2004   4 18.4   3 16.5 6.875000 7.666667 354.2120
```

```
> Summarize(LiDL~inc,data=d2)
```

```
   inc  n      mean       sd    min     Q1 median    Q3   max percZero
1    1 65  93.21687 19.27293  63.61  77.86  87.16 106.5 152.9        0
2    2 65 217.27094 40.52307 141.10 188.90 215.20 243.0 339.8        0
3    3 65 328.93495 45.24603 230.80 296.30 335.00 360.4 412.9        0
4    4 52 373.76110 44.18784 258.70 339.20 387.00 410.1 437.0        0
5    5 35 388.88135 43.20384 295.70 357.90 391.70 430.0 446.0        0
6    6 27 404.21979 38.13072 323.40 379.50 406.00 440.9 459.0        0
7    7 21 413.52603 30.17742 350.80 400.10 421.80 431.3 463.0        0
8    8 18 430.09080 27.60281 372.80 414.00 440.30 448.9 473.0        0
9    9 18 452.11693 28.21794 397.10 437.10 457.20 467.0 495.9        0
10  10 14 464.07799 28.00553 408.10 455.80 468.40 473.9 508.7        0
11  11 11 478.12451 30.49704 423.60 461.90 483.60 486.3 526.0        0
12  12  9 488.78812 26.28516 441.20 471.50 496.00 500.2 529.0        0
13  13  6 498.42473 24.85923 467.70 481.10 496.40 517.2 530.0        0
14  14  4 501.28996 20.35016 489.70 489.80 491.90 503.4 531.7        0
15  15  3 516.92132 29.52618 499.00 499.90 500.80 525.9 551.0        0
16  16  2 540.95833 38.12484 514.00 527.50 541.00 554.4 567.9        0
17  17  1 580.00000       NA 580.00 580.00 580.00 580.0 580.0        0
```

5.2.4 Fraser-Lee Model

In some cases, structures such as scales may take some time to form after hatch or metamorphosis. Consequently, early length estimates are biased. The Fraser-Lee model (Fraser 1916, Lee1920) accounts for this bias by including a biological intercept in the model. The model is,

$$L_i = a + (L_c - a)\frac{S_i}{S_c}$$

The parameter a is the intercept determined from the ageing-structure radius and fish length relationship and the other variables are previously defined.

Because we are using scales to back-calculate length at age, we will likely require a correction factor. Because we did not collect empirical data or find information in the literature regarding the length of scale formation in Spotted Sucker, then we must estimate this parameter by modeling the known relationship between ageing-structure radius and fish-length at capture from our Spotted Sucker data set. Even had we found this information in the literature, performing the below calculations is another good way to check one's data.

The regression of length at age i on scale radius at age i as described in Box 5.2 of the text is conducted with `lm()` with a formula of the form `response~explanatory` as the first argument and the data frame containing the variables in `data=`. The ANOVA table and estimated coefficients, among other statistics, are extracted from the saved `lm` object with `anova()` and `summary()`, respectively.

```
> lm1 <- lm(LiDL~Si2,data=d2)
> anova(lm1)
```

```
           Df  Sum Sq Mean Sq F value    Pr(>F)
Si2         1 6486641 6486641  3672.4 < 2.2e-16
Residuals 414  731254    1766
Total     415 7217895
```

```
> summary(lm1)
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  35.4914     5.1755   6.858 2.56e-11
Si2          50.8834     0.8397  60.600  < 2e-16

Residual standard error: 42.03 on 414 degrees of freedom
Multiple R-squared: 0.8987, Adjusted R-squared: 0.8984
F-statistic:  3672 on 1 and 414 DF,  p-value: < 2.2e-16
```

The intercept from this model (35.49) is an estimate of the correction factor required for the Fraser-Lee method. The back-calculated length-at-age using the Fraser-Lee method is then constructed by extracting this value from the saved `lm` object and saving it to it's own object and then writing an R expression for the Fraser-Lee equation. Summary statistics (similar to the Table shown on page 209 of Box 5.2 in the text) are then computed with `Summarize()`.

```
> a <- coef(lm1)[1]               # get the correction factor (first coefficient in lm1)
> d2$LiFL <- a + (d2$Lc-a)*(d2$Si2/d2$Sc2)
> head(d2)
```

```
     ID sex  Lc date age   Sc inc   Si      Si2      Sc2     LiDL     LiFL
1 07447   M 336 2004   3 16.3   1  5.0 2.083333 6.791667 103.0675 127.6719
2 07447   M 336 2004   3 16.3   2 12.9 5.375000 6.791667 265.9141 273.3172
3 07447   M 336 2004   3 16.3   3 16.3 6.791667 6.791667 336.0000 336.0000
4 35334   F 395 2004   4 18.4   1  4.8 2.000000 7.666667 103.0435 129.2762
5 35334   F 395 2004   4 18.4   2  9.9 4.125000 7.666667 212.5272 228.9226
6 35334   F 395 2004   4 18.4   3 16.5 6.875000 7.666667 354.2120 357.8768
```

```
> Summarize(LiFL~inc,data=d2)
```

```
   inc  n     mean       sd   min    Q1 median    Q3   max percZero
1    1 65 120.8734 17.29790  94.4 107.0  115.8 132.0 175.4        0
2    2 65 234.5123 36.27838 165.7 209.5  234.0 259.3 346.5        0
3    3 65 336.7997 39.87565 250.1 309.9  340.5 363.8 414.6        0
4    4 52 379.4626 38.91109 275.8 350.6  391.0 410.2 437.0        0
5    5 35 394.8710 38.44757 310.2 366.4  395.7 430.0 446.0        0
6    6 27 409.7779 34.16028 335.9 388.3  413.9 442.6 459.0        0
7    7 21 419.0092 27.23395 362.0 406.9  425.2 433.7 463.0        0
8    8 18 434.8316 25.29983 382.6 422.4  444.6 450.6 476.8        0
9    9 18 455.2892 25.95792 405.2 442.8  459.2 467.0 498.2        0
10  10 14 466.9554 26.23349 415.4 459.4  469.4 476.5 510.0        0
11  11 11 480.5609 29.01173 429.8 464.2  485.5 489.7 526.0        0
12  12  9 490.6470 25.16001 446.2 473.2  496.0 505.1 529.0        0
13  13  6 499.9276 24.45414 470.9 482.3  496.8 520.0 530.0        0
14  14  4 502.6126 21.39454 490.4 491.2  492.7 504.2 534.6        0
15  15  3 517.8175 30.30332 499.0 500.3  501.7 527.2 552.8        0
16  16  2 541.3280 38.64768 514.0 527.7  541.3 555.0 568.7        0
17  17  1 580.0000       NA 580.0 580.0  580.0 580.0 580.0        0
```

5.3 Assessing Differences in Length at Age Between Groups

Now that we have corrected back-calculated length at age in Box 5.2, we can test for differences between groups. For example, we commonly want to test for a sex effect on the length at age. We can use our previous example to evaluate differences between sexes by means of an analysis of covariance (ANCOVA) approach. We start with our Spotted Sucker data set containing fish identification number (`ID`), sex (`sex`), total length at capture (`Lc`), year of capture (`date`), age (`age`), radius of aging structure (scale) at capture (`Sc`), annulus increment number i (`inc`), and radius of aging structure at i (`Si`). We calculate the length at each increment using a direct proportion method and incorporate the Fraser-Lee correction factor calculated in Box 5.2. Given that growth has a curvilinear component, we create a dummy variable (`incsq`) (The authors use the word "dummy variable" here but this is not a correct use of the word. A dummy variable indicates to which group an individual below; e.g., 0=female and 1=male. In this case, the authors simply create a squared term of a quantitative explanatory variable so as to allow for a quadratic or second-degree polynomial regression.) to be incorporated into the model.

The same data from from Box 5.2 is used here, taking note that `LiFL` contains the back-calculated lengths at age using the Fraser-Lee method.

5.3.1 Assessing Differences in Length-At-Age between Groups

In Box 5.3 in the text the author's use a second-degree polynomial (i.e., quadratic) model to determine if "growth" (i.e., back-calculated length-at-age) differed between males and females. To perform this analysis, a squared version of the explanatory variable (i.e., `incSqrd`) is created and stored in the data frame. The linear model is then fit with a right-hand-side that contains the `sex` group factor variable, both `inc` and `incSqrd` quantitative explanatory variables (to perform the quadratic regression), and the interaction between `sex` and both quantitative explanatory variables (to completely assess whether there was a difference between the sexes). The saved `lm` object is submitted to `anova()` to extract the Type-I SS and to `Anova()` from the `car` package with `type="III"` to extract the type-III SS.

```
> d2$incSqrd <- d2$inc^2
> lm2 <- lm(LiFL~sex+inc+incSqrd+inc*sex+incSqrd*sex,data=d2)
> anova(lm2)              # type-I SS
```

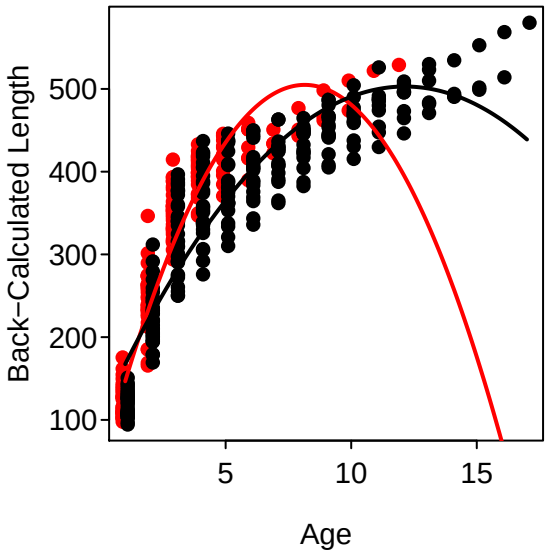
```
             Df  Sum Sq Mean Sq   F value    Pr(>F)
sex           1  116872  116872   62.7762 2.205e-14
inc           1 4287009 4287009 2302.7123 < 2.2e-16
incSqrd       1  937827  937827  503.7418 < 2.2e-16
sex:inc       1    9743    9743    5.2334   0.02267
sex:incSqrd   1  134986  134986   72.5061 3.189e-16
Residuals   410  763306    1862
Total       415 6249742
```

```
> Anova(lm2,type="III")  # type-III SS
```

```
             Sum Sq      Df  F value    Pr(>F)
(Intercept)  244395       1  131.274 < 2.2e-16
sex           49899       1   26.803 3.537e-07
inc         1974817       1 1060.748 < 2.2e-16
incSqrd      688101       1  369.605 < 2.2e-16
sex:inc      143893       1   77.290 < 2.2e-16
sex:incSqrd  134986       1   72.506 3.189e-16
Residuals    763306     410
Total           416 3999397
```

It is interesting to look at a plot of the results – even though it is a bit cumbersome to construct this plot. From this it is immediately apparent that the choice of a quadratic model was a poor choice. I would suggest using a von Bertalanffy model instead (see Box 5.4) or, for comparing two groups, in Chapter 9 here.

```
> # base schematic for adding points to
> plot(LiFL~inc,data=d2,col="white",xlab="Age",ylab="Back-Calculated Length")
> # add points for males and females, with slight offsets so they can be seen
> points(LiFL~I(inc-0.1),col="red",pch=16,data=Subset(d2,sex=="M"))
> points(LiFL~I(inc+0.1),col="black",pch=16,data=Subset(d2,sex=="F"))
>
> # create a sequence of increment numbers
> incs <- seq(1,17,0.1)
> # predict ages for males and females & then plot the lines
> predM <- predict(lm2,data.frame(inc=incs,incSqrd=incs^2,sex=rep("M",length(incs))))
> predF <- predict(lm2,data.frame(inc=incs,incSqrd=incs^2,sex=rep("F",length(incs))))
> lines(predM~incs,col="red",lwd=2)
> lines(predF~incs,col="black",lwd=2)
```



5.4 Fitting a von Bertalanffy Growth Curve

The length-at-age data on Spotted Sucker illustrated in Figure 5.4 (text) and Box 5.2 will be used here. For each individual in the data set, we have entered total length (`tl`) and age (`age`). Therefore, each fish represents a single degree of freedom in the analysis. To minimize bias, similar numbers of fish from each year-class should be included in the model. If older or younger age-classes are not well represented in the analysis, confidence limits at the extremes of the curve may expand dramatically or the model will fail to converge. Parameters for the growth curve can now be estimated iteratively using a nonlinear regression approach with the following R code (I have provided a more thorough analysis of how to fit von Bertalanffy models is in Chapter 9 here. You may want to read that document before continuing with the description below for Box 5.4).

5.4.1 Preparing Data

The `box5_4.txt` is read and the structure of the data frame is observed.

```
> d4 <- read.table("data/box5_4.txt",header=TRUE)
> str(d4)
```

```
'data.frame':   95 obs. of  2 variables:
 $ tl : int  388 418 438 428 539 432 444 421 438 419 ...
 $ age: int  4 4 4 5 10 4 7 4 4 4 ...
```

5.4.2 Getting Initial Values for the Model Parameters

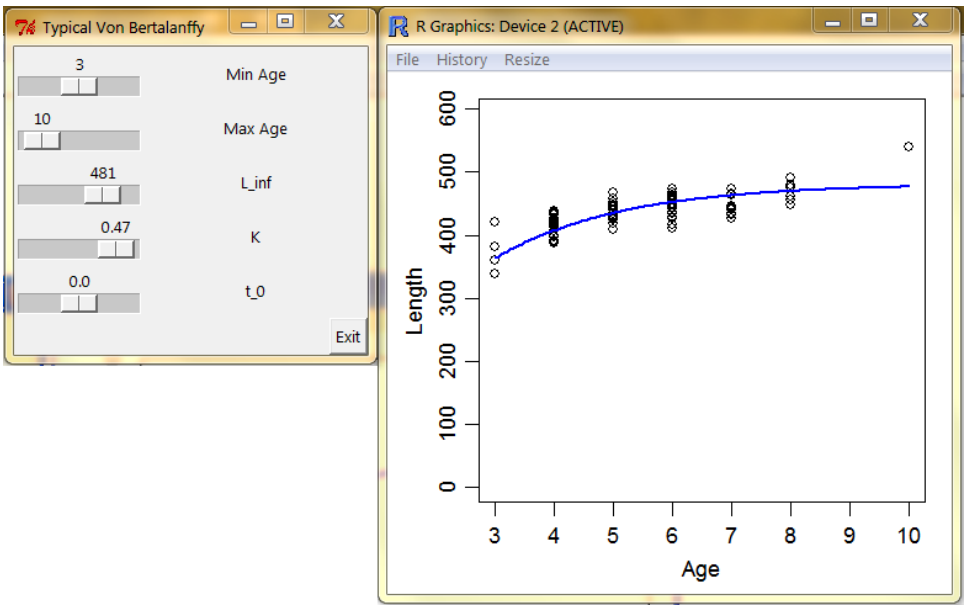
As noted in Box 5.4 in the text, the computer algorithms for fitting non-linear models are iterative and require starting values for the parameters. An alternative to what was described in Box 5.4 in the text for identifying starting values for the traditional von Bertalanffy equation is to plot the data and use interactive graphics to visually "fit" the von Bertalanffy model. The parameters from this process can then serve as the starting values for the more objective non-linear modeling algorithm. The `vbStarts()` function in the `FSA` package provides a mechanism for visually fitting the von Bertalanffy model. For this purpose, `vbStarts()` has the three arguments below.

- a formula of the form `len~age`,
- `data=`: a data frame that contains the `len` and `age` variables,
- `type=`: a string that identifies which parameterization to use (`model="typical"` is used for the traditional von Bertalanffy model used in Box 5.4),
- `dynamicPlot=TRUE`: which will create the interactive graphic.

This function will produce a graphics and a dialog box with slider bars corresponding to the parameters of the von Bertalanffy model (see below). The slider bars can be manipulated until an approximate fit is obtained. The values of the parameters at this approximate fit are then entered into a named R list with `list()` for later use.

```
> vbStarts(tl~age,data=d4,type="typical",dynamicPlot=TRUE)
```

```
> sv <- list(Linf=481,K=0.47,t0=0)
```



5.4.3 Fitting the von Bertalanffy Model

The von Bertalanffy model as an R formula is best placed into an object as shown below. When doing this, make sure that `tl` and `age` are the exact names of the length and age variables in your data frame and that `Linf`, `K`, and `t0` are the exact names of the parameters that you stored in your starting values list.

```
> vbmdl <- tl~Linf*(1-exp(-K*(age-t0)))
```

The non-linear model fitting procedure in R is implemented with `nls()`, which requires the model formula as the first argument, the list of starting values in `start=`, and the data frame in `data=`. The coefficient estimates and the correlations among coefficient estimates are extracted from the saved `nls` object with `overview()` from the `nlstools` package. One should pay very close attention to the correlations among parameter estimates shown in the results below. These values illustrate VERY strong correlations among the parameters which hinders interpretation as many other parameter triplets would provide nearly the same model fit (this is illustrated with the lack of change in SS for the last 15 or so iterations of the model fitting as shown in Box 5.4 of the text).

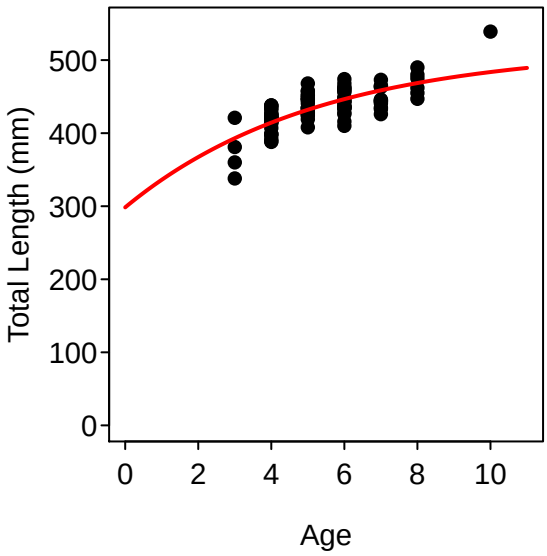
```
> ssvb <- nls(vbmdl,start=sv,data=d4)
> overview(ssvb)
```

```
------
Formula: tl ~ Linf * (1 - exp(-K * (age - t0)))

Parameters:
     Estimate Std. Error t value Pr(>|t|)
Linf 516.2350    48.8007  10.578   <2e-16
K      0.1900     0.1194   1.591    0.115
t0    -4.5423     3.4084  -1.333    0.186

Residual standard error: 18.11 on 92 degrees of freedom

Number of iterations to convergence: 32
Achieved convergence tolerance: 8.638e-06

------
Residual sum of squares: 30200

------
t-based confidence interval:
             2.5%       97.5%
Linf 419.31261275 613.1573362
K     -0.04719475   0.4271875
t0   -11.31164560   2.2269778

------
Correlation matrix:
           Linf          K         t0
Linf  1.0000000 -0.9872196 -0.9616493
K    -0.9872196  1.0000000  0.9927308
t0   -0.9616493  0.9927308  1.0000000
```

Finally, one can plot the model fit by plotting the raw data, creating a sequence of ages that cover the range of observed ages, using the model results to predicted lengths at each age in the sequence, and then plotting

the age sequence and predicted length pairs as a line with `lines()`. It is apparent from this plot that this model suffers from the lack of young small fish in the sample (i.e., it is unrealistic that the mean length of age-0 fish is nearly 500 mm).

```
> plot(tl~age,data=d4,pch=16,xlab="Age",ylab="Total Length (mm)",xlim=c(0,11),ylim=c(0,550))
> ages <- seq(0,11,0.1)
> preds <- predict(ssvb,data.frame(age=ages))
> lines(preds~ages,lwd=2,col="red")
```



5.4.4 Comparison Among Groups

The authors of Chapter 5 hint at comparisons of von Bertalanffy model parameters between groups but do not provide an example of such an analysis. An example of this analysis in R can be found in in Chapter 9 here.

5.5 Identifying the Environmental Effects on Growth

Often, fisheries scientists are interested in evaluating the effects of some management strategy on growth. Length limits, fertilization, and water level manipulations, for example, may all produce time-specific effects. We cannot simply compare pre-treatment length with post-treatment length. Weisberg and Frie (1987) demonstrated a method of isolating annular growth effects by calculating growth increment and assigning this not only to a specific age but to a specific year. We will use data collected from a population of Spotted Sucker to test the effects of an extended drought on growth. The drought occurred from 2000 through 2003. Rather than test for the effect of a specific individual year, we group years together by rainfall. Although it would have improved the statistical performance of the model, note that it is not necessary to sample pre-treatment fish length as long as the post-treatment sample contains a representative sample of fish that were alive during the pre-treatment period. In this case, year-classes from normal and drought (dry) years were present in the sample.

5.5.1 Preparing Data

The box5_5.txt is read and the structure of the data frame is observed. The `bcage` is converted to a group factor variable with `factor()`.

```
> d5 <- read.table("data/box5_5.txt",header=TRUE)
> str(d5)
```

```
'data.frame':   244 obs. of  10 variables:
 $ id    : Factor w/ 49 levels "04111","07447",..: 1 1 1 1 1 1 1 1 3 ...
 $ sex   : Factor w/ 2 levels "F","M": 1 1 1 1 1 1 1 1 1 ...
 $ tl    : int  486 486 486 486 486 486 486 486 395 ...
 $ w     : int  710 710 710 710 710 710 710 710 640 ...
 $ year  : int  2004 2004 2004 2004 2004 2004 2004 2004 2004 ...
 $ age   : int  9 9 9 9 9 9 9 9 4 ...
 $ bcyear: int  1995 1996 1997 1998 1999 2000 2001 2002 2003 2000 ...
 $ bcage : int  1 2 3 4 5 6 7 8 9 1 ...
 $ bctl  : int  76 181 275 326 366 412 433 463 486 103 ...
 $ growth: int  76 104 94 51 41 46 20 31 23 103 ...
```

```
> d5$bcage <- factor(d5$bcage)
```

The authors added a `group` variable that corresponded to pre-2000 and 2000-and-after which were labeled as "normal" and "dry", respectively. This variable is created in R with `recode()` from the `car` package which requires a variable as the first argument and recoding directives as the second argument. In this case, we want to recode `bcyear` such that years before 2000 are labeled as "normal" and years after 1999 are labeled as "dry." Thus, the recoding directive, which must be contained in double quotes, says that all years between 1995 (the first year present) and 1999 (the `:` operator creates a sequence of unit-spaced integers between the two numbers) will be recoded as "normal" and all other years (as denoted by `else=`) as "dry." Note that `as.factor.result=TRUE` forces R to return the result as a group factor variable.

```
> d5$group <- recode(d5$bcyear,"1995:1999='normal';else='dry'",as.factor.result=TRUE)
> str(d5)
```

```
'data.frame':   244 obs. of  11 variables:
 $ id    : Factor w/ 49 levels "04111","07447",..: 1 1 1 1 1 1 1 1 3 ...
 $ sex   : Factor w/ 2 levels "F","M": 1 1 1 1 1 1 1 1 1 ...
 $ tl    : int  486 486 486 486 486 486 486 486 395 ...
 $ w     : int  710 710 710 710 710 710 710 710 640 ...
 $ year  : int  2004 2004 2004 2004 2004 2004 2004 2004 2004 ...
 $ age   : int  9 9 9 9 9 9 9 9 4 ...
 $ bcyear: int  1995 1996 1997 1998 1999 2000 2001 2002 2003 2000 ...
 $ bcage : Factor w/ 9 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 9 1 ...
 $ bctl  : int  76 181 275 326 366 412 433 463 486 103 ...
 $ growth: int  76 104 94 51 41 46 20 31 23 103 ...
 $ group : Factor w/ 2 levels "dry","normal": 2 2 2 2 2 1 1 1 1 1 ...
```

5.5.2 Some Preliminaries

These data provide some limitations in the analysis as described in Box 5.5 of the text. To understand some of these limitations one must first look at the "structure" of the data and some summaries. The first thing to note about the `d5` data frame is that it is in one-increment-per-row format (discussed in Box 5.4). In this format a great deal of information specific to the fish (i.e., **NOT** the increment) is repeated in several rows in the data frame. This can be seen by looking at the first ten rows of the data frame using `head()`.

```
> head(d5,n=10)
```

```
      id sex  tl   w year age bcyear bcage bctl growth  group
1  04111   F 486 710 2004   9   1995     1   76     76 normal
2  04111   F 486 710 2004   9   1996     2  181    104 normal
3  04111   F 486 710 2004   9   1997     3  275     94 normal
4  04111   F 486 710 2004   9   1998     4  326     51 normal
5  04111   F 486 710 2004   9   1999     5  366     41 normal
6  04111   F 486 710 2004   9   2000     6  412     46    dry
7  04111   F 486 710 2004   9   2001     7  433     20    dry
8  04111   F 486 710 2004   9   2002     8  463     31    dry
9  04111   F 486 710 2004   9   2003     9  486     23    dry
10 35334   F 395 640 2004   4   2000     1  103    103    dry
```

It is not needed for the analysis but it is useful for summary information if we create a data frame that isolates (only once) the information for each fish. This reduction is accomplished by first realizing that the fish specific information is in the first six columns of `d5` and then using `unique()`, which will return only the unique rows of a data frame.

```
> d5fish <- unique(d5[,1:6])
> str(d5fish)
```

```
'data.frame':   51 obs. of  6 variables:
 $ id  : Factor w/ 49 levels "04111","07447",..: 1 3 4 11 12 15 16 19 20 24 ...
 $ sex : Factor w/ 2 levels "F","M": 1 1 1 1 1 1 1 1 1 1 ...
 $ tl  : int  486 395 386 463 467 446 463 360 450 437 ...
 $ w   : int  710 640 620 1000 850 1000 950 360 1100 750 ...
 $ year: int  2004 2004 2004 2004 2004 2004 2004 2004 2004 2004 ...
 $ age : int  9 4 4 7 9 5 7 3 6 5 ...
```

With this new data frame the number of fish of each age captured in each year is constructed with `xtabs()`. From this it is seen that all fish were captured in the same year, no fish less than age-3 were collected, and no age-8 (but age-7 and age-9) fish were captured. These results show that all information for "normal" rainfall years will come from the "early" growth increments on the "older" fish.

```
> xtabs(~year+age,data=d5fish)
```

```
      age
year    3  4  5  6  7  9
  2004 13 17  8  6  3  4
```

The linear model to be fit uses the `bcage` and `group` variables. A frequency table of these two variables (below) shows that it will be impossible to determine the effect of back-calculated ages six, seven, eight, and nine during the "normal" rainfall years as there is no data available for these age-group combinations. This unbalanced "design" will impact the ability to interpret an interaction in the model.

```
> tbl2 <- xtabs(~group+bcage,data=d5)
> addmargins(tbl2)
```

```
        bcage
group     1  2  3  4  5  6  7  8  9 Sum
  dry    31 39 45 35 17 13  7  4  4 195
  normal 21 13  7  4  4  0  0  0  0  49
  Sum    52 52 52 39 21 13  7  4  4 244
```

5.5.3 Fitting the Model I

The ANOVA model is fit with `lm()` and the ANOVA table with type-I SS is obtained by submitting the `lm` object to `anova()`. As described in Box 5.5 of the text, the interaction term is insignificant ($p = 0.5173$) and will be removed from the model in the next section.

```
> lm1 <- lm(growth~bcage*group,data=d5)
> anova(lm1)
```

```
             Df Sum Sq Mean Sq F value    Pr(>F)
bcage         8 333756   41720 95.5221 < 2.2e-16
group         1   6501    6501 14.8838 0.0001485
bcage:group   4   1422     356  0.8141 0.5172580
Residuals   230 100453     437
Total       243 442132
```

Before moving on, it should be noted that the type-III SS ANOVA table cannot be computed with `Anova()` because of the "unbalanced" design. In other words, the command below results in the error shown.

```
> Anova(lm1,type="III")
```

```
Error in Anova.III.lm(mod, error, singular.ok = singular.ok, ...): there are aliased coefficients in th
```

5.5.4 Fitting the Model II

The model without the interaction term is fit below and the corresponding ANOVA table with Type-I SS is again obtained with `anova()`. Because this model without the interaction does not have the difficulties associated with the missing values for the older ages in the "normal" rainfall years, a type-III SS ANOVA table is constructed by submitting the saved `lm` object to `Anova()` with `type="III"`. In either regard, both the `bcage` and `group` variables are "significant" indicating that growth differs by age (not surprisingly) and (more interestingly) by group.

```
> lm2 <- lm(growth~bcage+group,data=d5)
> anova(lm2)
```

```
           Df Sum Sq Mean Sq F value    Pr(>F)
bcage       8 333756   41720  95.827 < 2.2e-16
group       1   6501    6501  14.931 0.0001444
Residuals 234 101875     435
Total     243 442132
```

```
> Anova(lm2,type="III")
```

```
            Sum Sq     Df F value    Pr(>F)
(Intercept) 199470      1 458.168 < 2.2e-16
bcage       340056      8  97.635 < 2.2e-16
group         6501      1  14.931 0.0001444
Residuals   101875    234
Total          244 647902
```

The least-squares means are computed by submitting the `lm` object to `lsmeans()` from the `lsmeans` package. As there are two factors in this model, R must be told to compute the least-squares means separately by including the factor variable names separately in the `factors=` argument as such,

```
> lsmeans(lm2,~group)
```

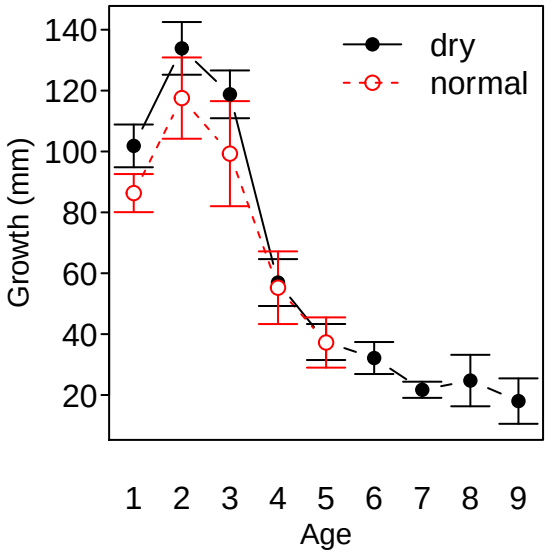
```
 group    lsmean       SE  df lower.CL upper.CL
 dry    60.77712 2.179773 234 56.48263 65.07161
 normal 47.17887 3.762883 234 39.76542 54.59233

Results are averaged over the levels of: bcage
Confidence level used: 0.95
```

```
> lsmeans(lm2,~bcage)
```

```
 bcage    lsmean        SE  df   lower.CL  upper.CL
 1      94.26940  2.913228 234  88.529893 100.00891
 2     126.38890  3.024304 234 120.430558 132.34724
 3     111.18526  3.166349 234 104.947063 117.42345
 4      51.36480  3.622065 234  44.228777  58.50082
 5      33.17197  4.681679 234  23.948344  42.39560
 6      25.35472  6.048607 234  13.438039  37.27141
 7      14.91516  8.080285 234  -1.004240  30.83457
 8      17.95088 10.580039 234  -2.893425  38.79518
 9      11.20088 10.580039 234  -9.643425  32.04518

Results are averaged over the levels of: group
Confidence level used: 0.95
```

The `fitPlot()` from the `FSA` package can be used to visualize the simultaneous effects of the two factors on `growth`. The so-called interaction plot is constructed by submitting the saved `lm` object to `fitPlot()`, along with the typical other arguments for modifying the plot. The observation that trajectories of growth by age for the "normal" and "dry" periods are generally parallel supports the conclusion of no interaction between age and group.

```
> fitPlot(lm2,xlab="Age",ylab="Growth (mm)",legend="topright",main="")
```



5.6 Estimating Growth from Mark and Recapture Data

In this example, data on the carapace length of Loggerhead Turtles (*Caretta caretta*) at mark and at recapture will be used to fit a von Bertalanffy growth curve by means of the Fabens (1965) method. For each individual in the data set, time at large (`days`) has been calculated from the mark and recapture dates. Carapace length at mark (`clmark`) and at recapture (`clrecap`) and time at large (`timeoutd`) has been entered for each individual. To calculate the von Bertalanffy growth parameters in a standard form, time at large has been converted from days to years (`timeouty`). Each individual, therefore, represents a single degree of freedom in the analysis. If older and younger age-classes are not well represented in the analysis, or if time at large is long with respect to the expected age of the animal, convergence criteria for parameter estimation may not be met. Parameters for the growth curve can now be estimated iteratively using a nonlinear regression approach with the following R code.

5.6.1 Preparing Data

The Box5_6.txt is read and the structure of the data frame is observed. As noted in Box 5.6 of the text, the `timeoutd` variable is the time-at-large in days and is converted to time-at-large in years `timeouty` below.

```
> d6 <- read.table("data/box5_6.txt",header=TRUE)
> str(d6)
```

```
'data.frame':   17 obs. of  5 variables:
 $ markd   : Factor w/ 16 levels "3/30/98","5/10/92",..: 6 12 15 15 13 9 11 1 ..
 $ clmark  : num  70.3 60.5 65.6 61.2 76.9 64.4 97.4 60.9 62.5 67 ...
 $ recapd  : Factor w/ 16 levels "11/15/95","12/24/94",..: 9 13 14 10 9 11 4 7..
 $ clrecap : num  76 64.7 67.9 65.2 79.1 67.5 97.9 67.7 69.1 69.5 ...
 $ timeoutd: int  1091 1090 714 1056 715 697 335 1544 1087 427 ...
```

```
> d6$timeouty <- d6$timeoutd/365
```

5.6.2 Getting Initial Values for the Model Parameters

As noted in Box 5.4, the von Bertalanffy model, even with the Fabens method, is a non-linear model that will require initial values for the model fitting. Unlike with the traditional von Bertalanffy model illustrated in Box 5.4, graphic methods for finding the starting values have not been developed for Fabens' method. Fortunately, finding starting values is fairly straight-forward. A simple starting value for the asymptotic mean length is the maximum length in the data frame. A reasonable starting value for the Brody growth coefficient is the average (from all fish in the sample) instantaneous growth rate between the time at marking and the time a recapture. These initial values are then entered into a list as shown in Box 5.4.

```
> svLinf <- max(d6$clrecap)
> svK <- with(d6, mean((log(clrecap)-log(clmark))/timeouty))
> ( Fsv <- list(Linf=svLinf,K=svK) )
```

```
$Linf
[1] 97.9

$K
[1] 0.05174481
```

5.6.3 Fitting the von Bertalanffy Model

The Fabens method von Bertalanffy model as an R formula is placed into an object, the non-linear model fitting procedure in R is implemented with `nls()`, and the coefficient estimates and the correlations among coefficient estimates are extracted with `overview()` as described in Box 5.4.

```
> Fvbmdl <- clrecap ~ clmark+(Linf-clmark)*(1-exp(-K*timeouty))
> tvb <- nls(Fvbmdl,start=Fsv,data=d6)
> overview(tvb)
```

```
------
Formula: clrecap ~ clmark + (Linf - clmark) * (1 - exp(-K * timeouty))

Parameters:
      Estimate Std. Error t value Pr(>|t|)
Linf 88.048090   3.519198   25.02 1.20e-13
K     0.084548   0.008202   10.31 3.35e-08

Residual standard error: 1.388 on 15 degrees of freedom

Number of iterations to convergence: 4
Achieved convergence tolerance: 3.312e-07

------
Residual sum of squares: 28.9

------
t-based confidence interval:
            2.5%      97.5%
Linf 80.54709649 95.5490841
K     0.06706564  0.1020298
```

```
------
Correlation matrix:
           Linf          K
Linf  1.0000000 -0.9079803
K    -0.9079803  1.0000000
```

```
Reproducibility Information
  Compiled Date: Wed May 13 2015
  Compiled Time: 7:49:04 PM
  Code Execution Time: 0.99 s

  R Version: R version 3.2.0 (2015-04-16)
  System: Windows, i386-w64-mingw32/i386 (32-bit)
  Base Packages: base, datasets, graphics, grDevices, grid, methods, stats,
    utils
  Required Packages: FSA, car, lsmeans, nlstools and their dependencies
    (coda, dplyr, estimability, gdata, gplots, graphics, Hmisc, knitr,
    lmtest, MASS, methods, mgcv, multcomp, mvtnorm, nnet, pbkrtest,
    plotrix, plyr, quantreg, relax, sciplot, stats)
  Other Packages: car_2.0-25, estimability_1.1, Formula_1.2-1, FSA_0.6.13,
    FSAdata_0.1.9, ggplot2_1.0.1, Hmisc_3.16-0, Kendall_2.2, knitr_1.10.5,
    lattice_0.20-31, lsmeans_2.17, multcomp_1.4-0, mvtnorm_1.0-2,
    NCStats_0.4.3, nlstools_1.0-1, plotrix_3.5-11, rmarkdown_0.6.1,
    survival_2.38-1, TH.data_1.0-6
  Loaded-Only Packages: acepack_1.3-3.3, assertthat_0.1, bitops_1.0-6,
    boot_1.3-16, caTools_1.17.1, cluster_2.0.1, coda_0.17-1,
    codetools_0.2-11, colorspace_1.2-6, DBI_0.3.1, digest_0.6.8,
    dplyr_0.4.1, evaluate_0.7, foreign_0.8-63, formatR_1.2, gdata_2.16.1,
    gplots_2.17.0, gridExtra_0.9.1, gtable_0.1.2, gtools_3.4.2, highr_0.5,
    htmltools_0.2.6, KernSmooth_2.23-14, latticeExtra_0.6-26, lme4_1.1-7,
    lmtest_0.9-33, magrittr_1.5, MASS_7.3-40, Matrix_1.2-0, mgcv_1.8-6,
    minqa_1.2.4, munsell_0.4.2, nlme_3.1-120, nloptr_1.0.4, nnet_7.3-9,
    parallel_3.2.0, pbkrtest_0.4-2, plyr_1.8.2, proto_0.3-10,
    quantreg_5.11, RColorBrewer_1.1-2, Rcpp_0.11.6, relax_1.3.15,
    reshape2_1.4.1, rpart_4.1-9, sandwich_2.3-3, scales_0.2.4,
    sciplot_1.1-0, SparseM_1.6, splines_3.2.0, stringi_0.4-1,
    stringr_1.0.0, tools_3.2.0, yaml_2.1.13, zoo_1.7-12
```

References

Dahl, K. 1907. The scales of herring as a means of determining age, growth and migration. Report of the Norwegian Fisheries and Marine Investigations 2(6):1–39.

Fabens, A. 1965. Properties and fitting of the von Bertalanffy growth curve. Growth 29:265–289.

Francis, R. 1990. Back-calculation of fish length: A critical review. Journal of Fish Biology 36:883–902.

Fraser, C. M. 1916. Growth of the spring salmon. Transactions of the Pacific Fisheries Society 1915:29–39.

Isermann, D. A., and C. T. Knight. 2005. A computer program for age-length keys incorporating age assignment to individual fish. North American Journal of Fisheries Management 25:1153–1160.

Weisberg, S., and R. V. Frie. 1987. Age and growth of fish. Pages 127–143 *in* R. C. Summerfelt and G. E. Hall, editors. Iowa State University Press.