FLSR

Laurence Kell

11. February 2011

Table of Contents

1 Introduction

Stock recruitment relationships are important when assessing and projecting stocks (e.g. for

completing the life cycle in population models) and for calculating target reference points such as maximum sustainable yield (MSY) and limits such as the level of fishing mortality that would drive the stock to extinction. Stock recruitment relationships are implemented by the FLSR class, an extension of FLModel. which can be used to fit stock recruitment relationship and provide estimates of uncertainty. It is also used by other FLR classes to specify a stock recruitment relationship, for example when performing stock projections using Flash or calculating biological reference points using FLBRP.

2 Stock Recruitment Relationships

Importance processes when modeling recruitment are density dependance, depensation. Density-dependent recruit production (also known as compensation) is when recruit production per spawner depends on population density and is lower at larger stock sizes. The maximum rate of recruit production per spawner occurs at lower stock sizes and is given by the slope at the origin of the stock recruitment relationship. In contrast depensation (or the Allee effect) is where recruit production per spawner is reduced at small population sizes (Stephens, Sutherland & Freckleton (1999).

Fish stocks can fluctuate extensively over a large range of spatial and temporal scales independently of human exploitation (Hjort 1914; Cushing 1995), such fluctuations are often ascribed to stochastic variations in key processes related to the environmental (e.g., Lehodey et al. 1997; Bailey 2000; Köster et al. 2005). In some cases an appropriate co-variate may explain recruitment variability (Schirripa, et al 2009). While poor or good recruitment may tend to occur for a few years in a row, i.e. be auto-correlated.

There are a variety of commonly used stock recruit relationships that are generally based on different assumptions about which process are the most important i.e.

Table 1. Stock recruitment relationships.

Functional Form	*Formula*	*Processes*
Beverton and Holt	$\alpha \cdot S/(\beta+S)$	Compensation
Cushing	$\alpha \cdot S^{\cdot\beta}$	
Deriso and Schnute	$\alpha \cdot (1-\beta \cdot \gamma \cdot ssb)^{(1/\gamma)}$	Both compensation and over compensation
Mean	α	implies recruitment is environmentally driven
Non-parametric	Lowess smoother	Non parametric, useful for description but not prediction
Pella-Tomlinson	$\alpha \cdot (1-(S/\beta)^{\gamma})$	
Ricker	$\alpha \cdot S \cdot \exp(-\beta \cdot S)$	Over compensation
Shepherd	$\alpha \cdot S/(1+(S/\beta)^{\gamma})$	Both compensation and over compensation
Segmented Regression	$if\ S \le \beta\ then \alpha \cdot S\ else \alpha \cdot \beta$	Emprirically based compensation

Depensation can be incorporated by the addition of an extra parameter i.e.

Table 2. Depensatory Stock recruitment relationship.

Functional Form	*Formula*
Beverton and Holt	$\alpha \cdot S^{\gamma}/(\beta + S^{\gamma})$
Ricker	$\alpha \cdot S^{\gamma} \cdot \exp(-\beta \cdot S)$
Shepherd	$\alpha \cdot S^{2}/(1+(S/\beta)^{\gamma})$

The various stock recruitment relationships fitted to North Sea herring data stock recruitment are shown in **figure 1**, recruitment per spawner is shown in **figure 2**.

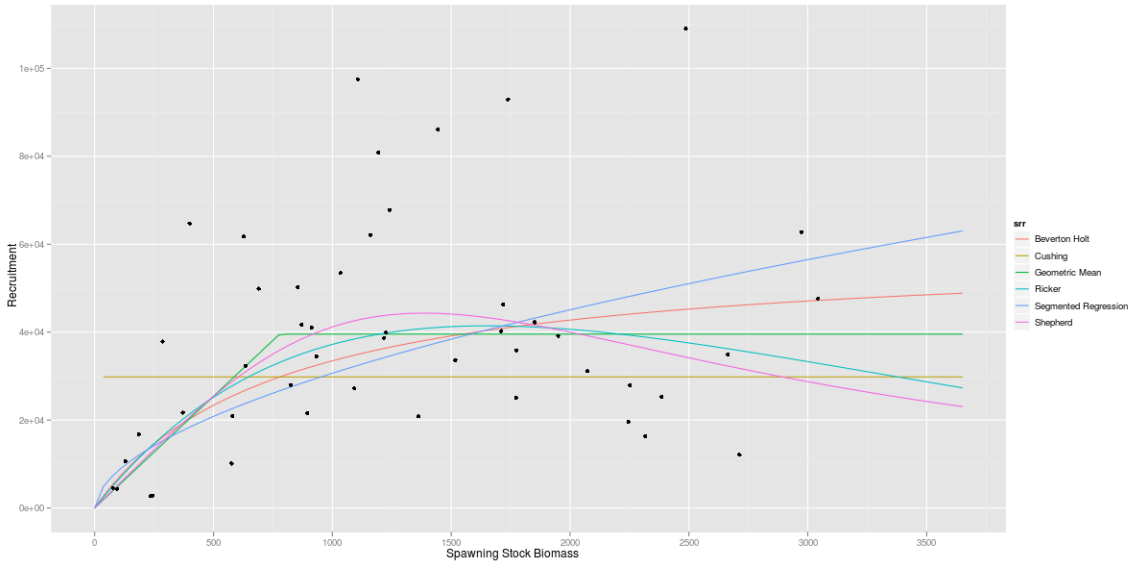

Figure 1. Stock recruitment relationships.

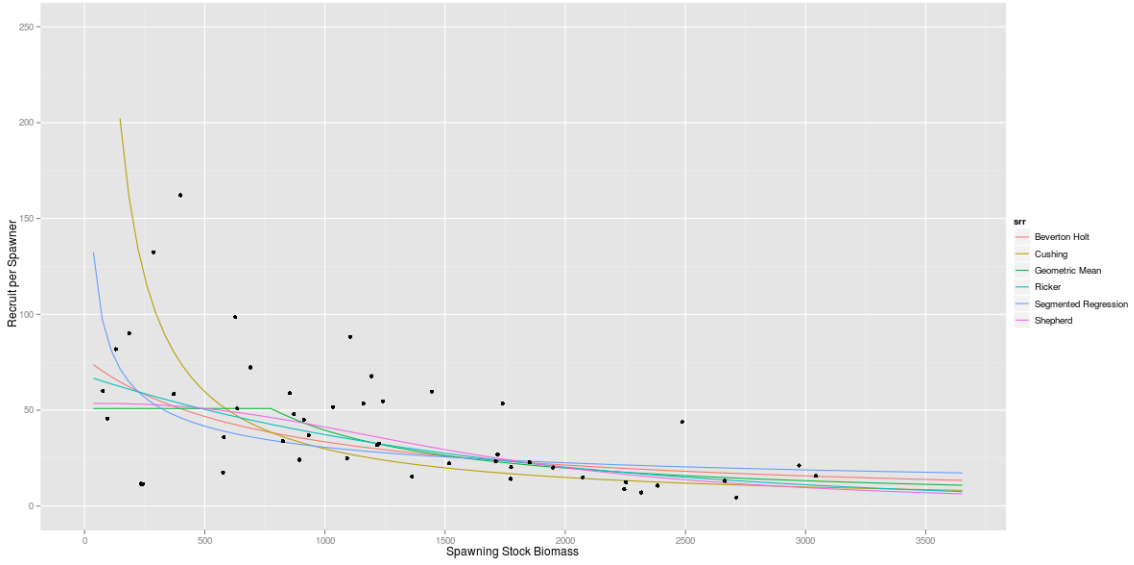

Figure 2. Recruit per spawner

Without density dependence the number of recruits would be proportional to spawning stock biomass, i.e. if you double SSB you double recruitment. However, many population processes are density dependent due to limited resources (e.g. food habitat) or opportunistic predators who switch to the most abundant prey. For example if food available for larvae is limited then as the number of larvae increase mortality due to starvation will also increase. If larvae are competing with each other so that some get all the food and others none then there will be a limit on the number of recruits,e.g. A Beverton and Holt relationship where after a certain level of egg production (i.e. SSB) recruitment is constant. However, if all larvae get the same proportion of the available food resource then after a certain larval population size all larvae will starve, this results in over compensation i.e. a Ricker stock recruitment relationship.

A depesatory Ricker relationship is contrasted with the conventional Ricker relationship in **figure 3** and recruitment per spawner is shown in **figure 4**.

```
iter    0 value -14.395239
final  value -14.395239
converged
```

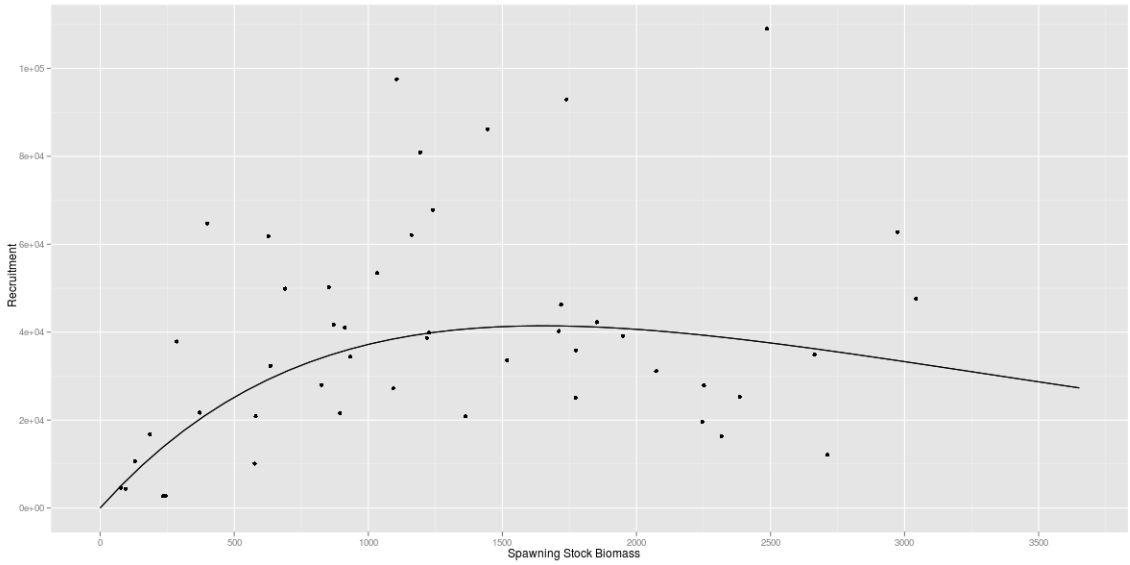

Figure 3. Ricker stock recruitment relationships with and without depensation.

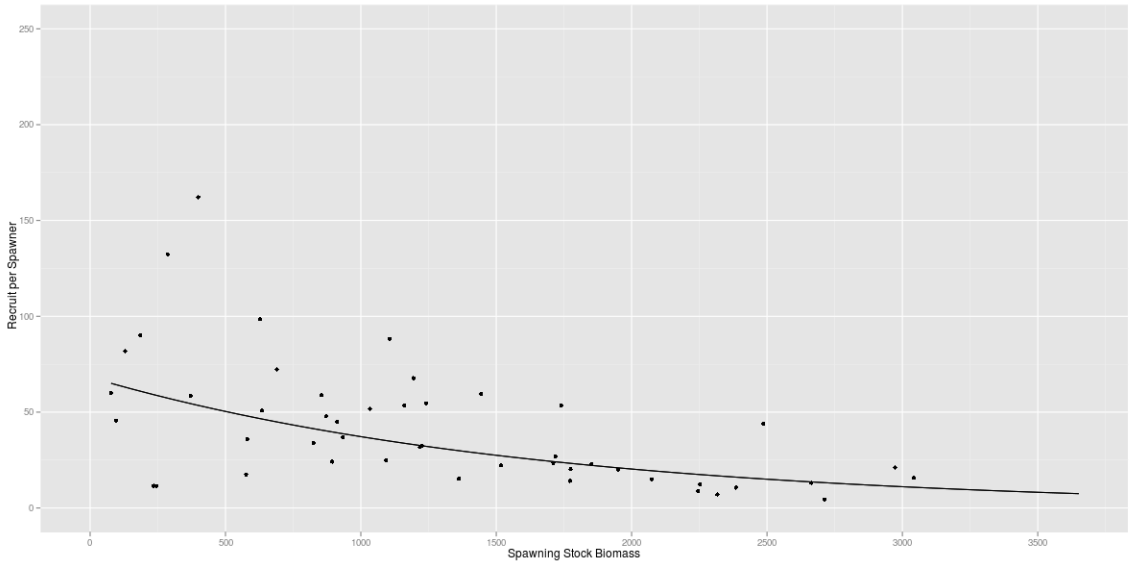

Figure 4. Recruit per spawner

Although evidence for depensation is scarce it could occur if predators are obligatory predators on a species of larvae, this means that a constant number of larvae will be consumed regardless of larval abundance. Therefore at low SSB when fewer larvae are produced mortality will increase and the slope at the origin (R/SSB) will decrease.

3 Steepness and virgin biomass

The same stock recruitment relationship can be parametrised in alternative ways, for example to provide better guesses for the parameter values, to fix certain parameters or provide priors in a way that is related to knowledge of biological proceses, or to reduce correlation between parameters so that both can be estimated independently. An important example in this regard are virgin biomass (the biomass when there is no fishing i.e. F=0.0) and steepness (the ratio of expected recruitment at 40% of virgin biomass to that at virgin biomass) which tells you when recruitment over fishing is likely to occur.

To estimate steepness and virgin biomass it is necessary to specify or estimate $SPR_{F=0}$, (the spawner per recruit at F=0.0). Generally $SPR_{F=0}$, is included as a fixed parameter as it is often difficult to estimate both steepness and $SPR_{F=0}$ together as it is highly unlikely to have information in the data on both steepness and $SPR_{F=0}$ together.

There are methods in FLR that can be used to change parameters between "alpha/beta" and "steepness/virgin biomass" parameterisations.

Table 3. Steepness and virgin biomass as functions of alpha and beta

Functional Form	*Steepness*	*Virgin Biomass*
Beverton and Holt	$\alpha \cdot S/(\beta+S)$	
Cushing	$\alpha \cdot S^{\cdot \beta}$	
Deriso and Schnute	$\alpha \cdot (1-\beta \cdot \gamma \cdot ssb)^{(1/\gamma)}$	

Mean	α
Non-parametric	Lowess smoother
Pella-Tomlinson	$\alpha \cdot (1-(S/\beta)^{\gamma})$
Ricker	$\alpha \cdot S \cdot \exp(-\beta \cdot S)$
Shepherd	$\alpha \cdot S/(1+(S/\beta)^{\gamma})$
Segmented Regression	$if\ S \leq \beta\ then\ \alpha \cdot S\ else\ \alpha \cdot \beta$

Table 4. Alpha and Beta as functions of steepness and virgin biomass

Functional Form	*Alpha*	*Beta*
Beverton and Holt	$4\gamma\tau / [\Psi_{F=0}(5\tau\text{-}1)]$	$[\alpha\ \Psi_{F=0}(\tau^{1}\text{-}1)] / 4$
Cushing		
Deriso and Schnute		
Mean		
Non-parametric		
Pella-Tomlinson		
Ricker		
Shepherd		
Segmented Regression		

3 FLSR Class

The FLSR class is an S4 object which combines the data (i.e. slots) and the actions (i.e. methods) performed on the data. This type of design is known as object oriented (OO) programming (see Object Oriented Programming II (2001) by Robert Gentleman). A major reason for using OO programming is that it encourages code reuse. So that the same code (functions, procedures, data structures) can be used in many different places. This is a good thing both for the user and developer because it means that you do not have to relearn how to use each new application nor do you always have to reimplement when developing new sofware applications (How S4 Methods Work (2006) by John Chambers).

Methods and classes in R language are essentially programming concepts to enable good organisation of functions and of general objects, respectively. The line between user and developer is generally blurred as R user usually end up writing writing functions, at least once we get past the strict cut-and-paste stage.

Functions are the actions of the language; calls to them express what the user wants to happen. The arguments to the functions and the values returned by function calls are the objects. These objects represent everything we deal with. Actions create new objects (such as summaries and models) or present the information in the objects (by plots, printed summaries, or interfaces to other software). R is a functional, object-based system where users program to extend the capacity of the system in terms of new functionality and new kinds of objects. Once programming in R reaches a moderately

ambitious level, the total complexity of the added functionality can start to obscure what the software is doing. Hopefully by designing classes like FLSR for a well defined use with appropriate data and methods a clear structure is provided for the user.

3.1 Slots

The slots in an FLSR object contain a variety of types, from general summary information (common to all FLR classes), data required for fitting the stock recruitment relationship, specification of the relationship, the results and goodness of fit statistics. A summary of all the slots can be obtained using the generic method `summary`
after loading the example FLSR object for North Sea herring in FLR.

```
data(nsher)

summary(nsher)
```

Chunk 1:

3.1.1 General Information

name	character: i.e. the stock that the data came from
desc	character: a description that helps in remembering where the data came from
range	numeric: the range of years and ages in the data, can be used to truncate results of methods without changing the data

3.1.2 Data

rec	FLQuant: observed recruits
ssb	FLQuant: observed stock size
covar	FLQuants: covariate, e.g. for environmental data

3.1.3 Model

model	formula: functional form of stock recruitment relationship
params	FLPar: model parameters
logl	function: specification of log likelihood
initial	function: initial best guess for parameters
gr	function: gradient

3.1.4 Results

fitted	FLArray: fiited values from model
residuals	FLArray: residuals from fit
vcov	array: the variance covariance matrix by iteration
hessian	Array: the hessian matrix of 2nd derivatives of the log-likelihood wrt parameters by iteration
logLik	logLik: a generic function from stat
details	list: contains additional information returned by the optimiser

3.2 Methods

Methods allow the model to be fitted, plots and summaries to be made and quantities like the hessian to be computed. Some quantities are not saved as slots (i.e. as data) since they can be derived from the slots directly for example the r squared can be calculated from the residuals. This saves in memory and also ensures that values are consistent within a class, however to the user the interface is the same for example if r squared was contained in a slot or was calculated by a method the code would be identical i.e. `rSq(x)`.

The methods can be listed using

```
showMethods(class="FLSR")
```

Chunk 2:

fmle	Fits model to data using method of maximum likelihood
predict	Generates predicted values for either the original data or new data points

plot	plot of data, fit and diagnostics
summary	summary by slot
print	prints all slots in full

Statistics

AIC	Akaike information criteria

sigma	sigma or the standard deviation of the residuals
rSq	R squared statistic

profile:	likelihood profile, either a surface for 2 parameters, or a profile for 1,
jacknife	can be used on FLQuant data slots to jacknife the data and then the object can be re-fitted

gr	Gradient, i.e. 1st order derivatives of the likelihood with respect to the parameters
computeHessian:	Hessian, i.e. 2st order partial derivatives of the likelihood with respect to the parameters

lowess:	lowess fit, a non parametric fit to the data, useful for suggesting a functional form
spr0:	spawner per recruit at virgin biomass.

ab	returns parameters as alpha and beta, from an FLSR where parameters are specified as steepness and virgin biomass
sv	returns parameters as steepness and virgin biomass, from an FLSR where parameters are specified as steepness and virgin biomass

Model

Before using FLSR you need to specify a stock recruitment relationship to fit, this is done by using the model method to select one of the inbuilt functional forms i.e. Beverton and Holt, Ricker, Cushing, Shepherd, Segmented Regression or mean recruitment (`bevholt(), ricker(), cushing(), shepherd(), mean(), segreg()`, either for the traditional parametrisation of alpha and beta, or in terms of steepness and virgin biomass (`bevholtSV(), rickerSV(), cushingSV(), shepherdSV(), meanSV(), segregSV()`. The user can create new functional forms as well.

The model contains 5 elements, the model itself, the loglikelihood, the starting values and the upper

and lower bounds.

```
#### Model and likelihood are functions of the FLSR class
model(nsher)
logl(nsher)

#### initial values for the optimiser
initial(nsher)

#### lower and upper limits for the parameters
lower(nsher)
upper(nsher)

### where did these come from
ricker()
```

Chunk 3:

4 Basic operations with FLSR

To use the FLSR class you first have to create an FLSR object by initialising the various slots; i.e. the data in the ssb & rec slots and the model to fit. Fitting a particular stock recruitment relationship may also require various decision to be made such as fixing some parameters. Once a model has been fitted then goodness of fit diagnostics will have to be inspected.

4.1.1 Creating an object

New `FLSR` objects can be created by coercion from objects of other classes e.g. `FLStock`, `FLBiol` or directly from time series of recruits and SSB. The benefit of coercion is that the recruitment and SSB objects in the `FLSR` object are correctly aligned by cohort and that SSB is calculated based upon the data and properties of the original object. For example spawning time might vary. Example based on the North Sea plaice `FLStock` data set.

```
#### Creation from an FLStock
data(ple4)

ple4SR<-as.FLSR(ple4)

### Note the shift in years, since 1st age and recruitment is at age 1 in ple4
ssb(ple4SR)
rec(ple4SR)
```

Chunk 4:

```
> ple4SR <- as.FLSR(ple4)
> ssb(ple4SR)
An object of class "FLQuant"
, , unit = unique, season = all, area = unique

     year
age   1957   1958   1959   1960   1961   1962   1963   1964   1965   1966
  all 274205 288540 296825 308164 321354 372863 370373 363077 344013 361549
     year
age   1967   1968   1969   1970   1971   1972   1973   1974   1975   1976
  all 416563 402521 377432 333933 316343 319062 268714 278648 293136 310954
     year
age   1977   1978   1979   1980   1981   1982   1983   1984   1985   1986
  all 316929 303433 297122 272416 262061 263998 314021 326341 348675 375392
     year
age   1987   1988   1989   1990   1991   1992   1993   1994   1995   1996
  all 445855 391254 408489 368969 335747 269528 228668 193093 174408 173903
     year
age   1997   1998   1999   2000   2001   2002   2003   2004   2005   2006
  all 185308 211327 184733 208393 234078 162725 179158 151508 167531 173783
     year
age   2007
  all 166061

units:  kg
> rec(ple4SR)
An object of class "FLQuant"
, , unit = unique, season = all, area = unique

   year
age 1958    1959    1960    1961    1962    1963    1964    1965    1966
  1  698110  863386  757299  860577  589154  688367 2231504  694575  586779
   year
age 1967    1968    1969    1970    1971    1972    1973    1974    1975
  1  401298  434281  648877  650584  410281  366633 1312097 1132831  864875
   year
age 1976    1977    1978    1979    1980    1981    1982    1983    1984
  1  692849  988889  913474  891160 1128822  869640 2029493 1306601 1261067
   year
age 1985    1986    1987    1988    1989    1990    1991    1992    1993
  1 1849179 4732214 1918256 1770637 1184055 1033216  910370  773003  522410
   year
age 1994    1995    1996    1997    1998    1999    2000    2001    2002
  1  434986 1153325 1283485 2105676  765785  836929  927442  516739 1612473
   year
age 2003    2004    2005    2006    2007    2008
  1  505292 1159019  714344  820006  949341  844041

units:  thousands
```

Before fitting the functional form of the stock recruitment relationship and the error model have to be specified this can be done in two ways, after creation of an object or when the object is created, i.e.

```
## after creating the new object
model(ple4SR)<-bevholt()

## when creating the new object
ple4SR<-as.FLSR(ple4,model="bevholt")
```

Chunk 5:

5 Fitting

Once the form of the stock recruitment relationship has been selected using the supplied functions, the likelihood will be specified, commonly this is log normal, although other error models could be used, e.g. normal log or normal. New models can be added by modifying the examples provided in FLR.

A generic method srModel will also be available in Flash in FLR ver 3, to provide a common interface allowing any combinations of different choices (e.g. model, likelihood) to be made. This will also provide gradients ensuring fast robust fits using automatic differentiation, see below.

When fitting a non-linear model it is necessary to supply initial values for the fitting algorithm, the better these are the more likely the algorithm will converge to the correct solution. This is done automatically for you in FLSR, but the defaults can always be changed if required. Comparing the likelihoods of the two fits will show that the initial one with badly chosen starting values has a much smaller likelihood than the one the one where starting values were set automatically indicating that the fit failed

```
#### fitting
## copy objec
her4RK       <-nsher

## choose SRR
model(her4RK)<-ricker()

badFit        <-fmle(her4RK, start=c(a=1,b=2))

her4RK       <-fmle(her4RK)

#### plot fit and diagnostics
plot(her4RK)
```

Chunk 6:

It is good practice to plot the fitted result, to see if the functional form chosen is appropriate and if there are any patterns in the residuals that suggest the assumptions made are violated.

It may sometimes be useful to set bounds on parameters, e.g. to stop them becoming negative which may in some cases predict negative recruitment or to fix parameters if you want for example to create some scenarios about productivity. However initially it is best not to specify bounds but to see what the data tell you.

```
# parameter values can also be fixed
her4RK  <-fmle(her4RK, fixed=list(a=63580373))

# Comparison of fits using AIC
AIC(her4RK)
AIC(her4RKFixed)
```

Chunk 7:

The Akaike's information criterion (AIC) for the simpler model (her4Fixed) is smaller than that of the full model (her4SR) suggesting that the model with a fixed value for alpha fits better than the model where it is estimated. This is an initial sign of a lack of information in the data with respect to the parameters of the model.

6 Likelihood

Fitting the model by the method of maximum likelihood assumes are that recruitment is some function of spawning stock biomass (S) plus a process error term i.e.

$$R_{t+n} = f(S_t) + e_t$$

The deviates might also not be independent (i.e. low recruitments tending to occur together or high recruitments being followed by low recruitments). The residuals can therefore be fitted by assumming an autoregressive a first order auto-regressive process assuming normal errors (Seber & Wilde, 1988)

$$l(\theta \mid \phi, \sigma_a^2) = -\frac{1}{2}\ln\left(\frac{1}{2\pi}\right) - \frac{n}{2}\ln\left(\sigma_a^2\right) + \frac{1}{2}\ln\left(1-\phi^2\right) - \frac{1}{2\sigma_a^2} S_1(\theta,\phi)$$

Where

$$S_1(\theta,\phi) = \left(1-\phi^2\right)\left\{y_1 - f(x_1 \mid \theta)\right\}^2 + S_2(\theta,\phi)$$

$$S_2(\theta,\phi) = \sum_{i=2}^{n}\left\{y - \phi y_{i-1} - f(x_i \mid \theta) + \phi f\left(x_{i-1} \mid \theta\right)\right\}^2$$

$$\sigma_a^2 = (1-\phi^2)\sigma^2$$

In a non-linear model there are often nuisance parameters, i.e. ones which although are included as parameters in a likelihood function can be expressed in terms of the other parameters. This can be useful because standard likelihood methods can become unreliable or fail entirely when there are many nuisance parameters or when the nuisance parameters are highly-dimensional. So that instead of estimating the variance parameter (σ) for the Normal and Lognormal distribution as a parameter it is calculate from the data conditional on the parameters using the sigma method.

6.1 Diagnostics

An important first diagnostic is the plot method.

```
plot(her4RK)
```

Chunk 8:

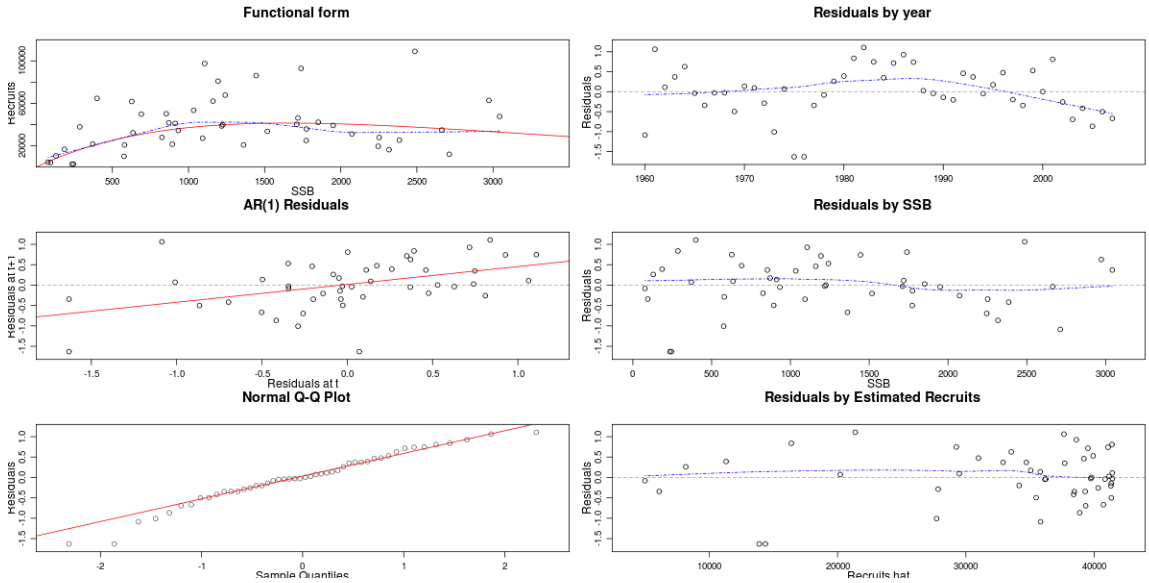

Figure 5. Diagnostics

This produces a plot with six panels, the 1st panel row 1 column 1 shows the stock-recruit pairs with the fitted stock recruitment relationship and a lowess smother to help suggest an appropriate functional form. Then going clockwise the next plot shows the residuals plotted against year to help identify any systematic departure from the model assumptions, i.e. does the model appropriate for all years as a pattern in the residuals might indicate that average recruitment was either less or greater than expected indicating either the wrong choice of model or a regime shift. The next plot is of the residuals against SSB again to evaluate whether the model is a reasonable fit to the data. The next plot is of the residuals against the fitted values as a check of the variance. Next the observed residuals are plotted against their expected quantiles if they come from a normal distribution, any systematic departure from the straight line indicates a violation of the assumptions, i.e. skewness or over or under dispersion. The final lot of the residuals with a lag of time 1, is to identify autocorrelation.

Often fitting will fail because of strong correlation between explanatory variables or because the likelihood is virtually flat at the maximum. A check for this is to profile the likelihood, i.e. calculate it for a range of fixed parameters values while estimating the other parameters. Plotting this will show whether you are actually at a maximum (i.e. the solution) . The likelihood profile of a parameter can be estimated by repeatedly fixing a parameter ove a range of values and then maximising the likelihood over all other parameters. Likelihood profiles are useful to measure the uncertainty associated with a parameter estimate and to evaluate how well a parameter can be estimated from the data. Profiles can also be obtained for pairs of parameters in order to investigate correlation between parameters.

Using the profile method

```
#### Profile both parameters
profile(her4RK,which=c("a","b"))

#### Profile alpha
profile(her4RK,which=c("a"))

#### Profile beta
profile(her4RK,which=c("b"))
```

Chunk 9:

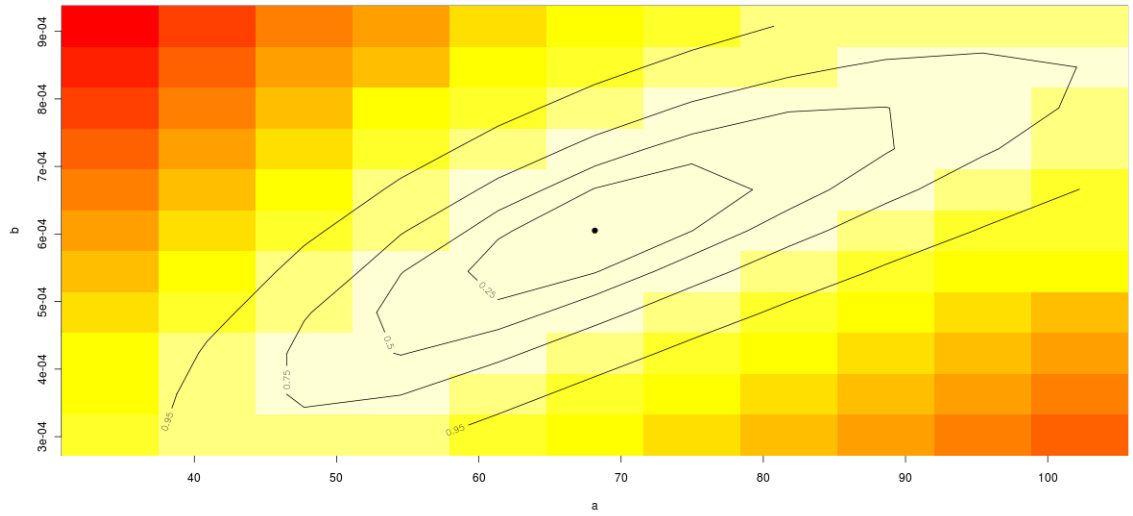

Figure 6. Likelihood profile of alpha and beta

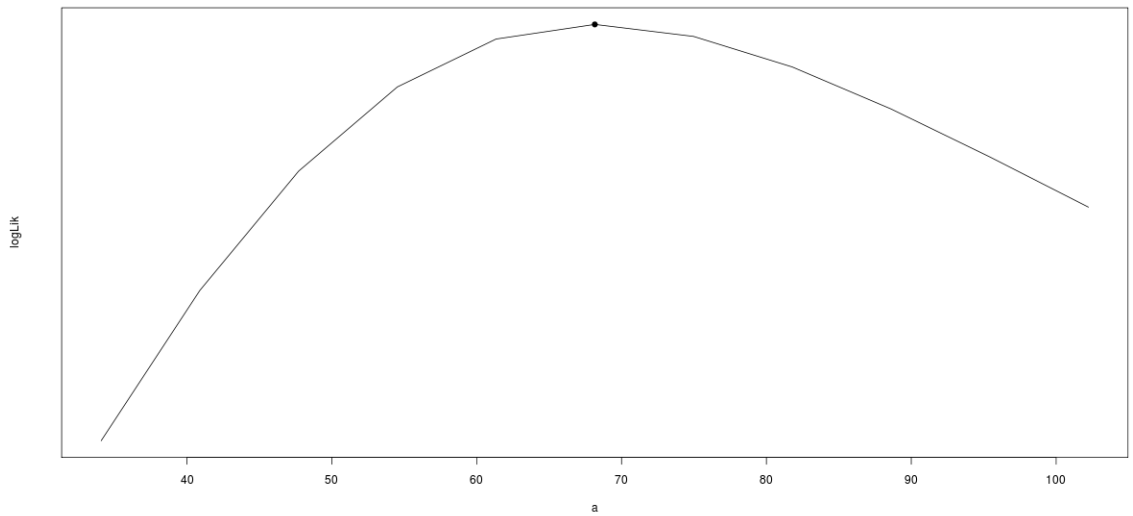

Figure 7. Likelihood profile of alpha

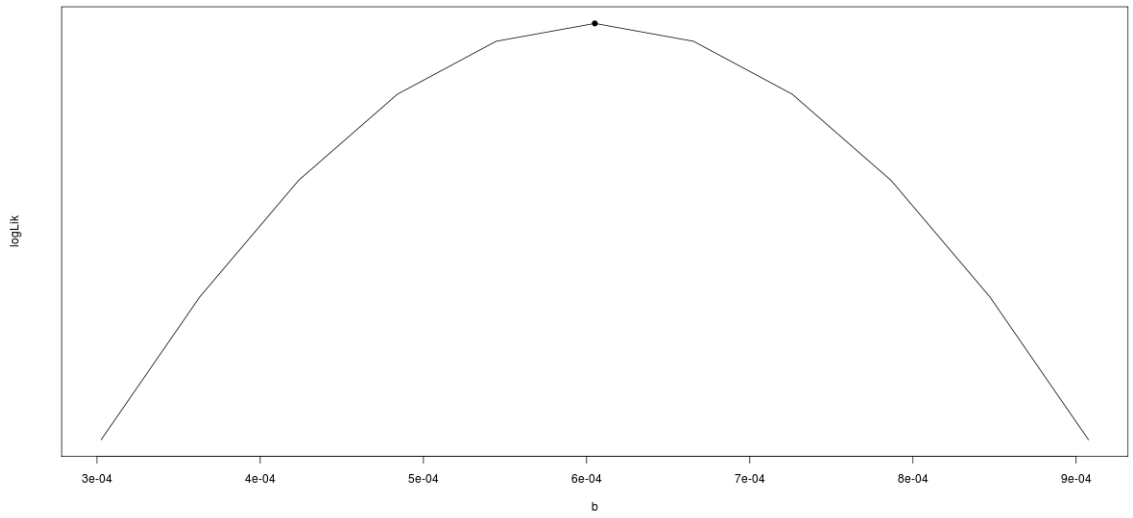

Figure 8. Likelihood profile of beta

The surface shows that alpha & beta are correlated, i.e. if you increase alpha the estimate of beta also increases, this can also be seen from the covariance matrix. The hessian and covariance matrices are important diagnostics in non-linear problems, since the hessian being the second partial derivatives of the likelihood shows you the shape of the likelihood at the solution. For example if there is correlation between alpha and beta, converting the covariance matrix to the correlation matrix shows that this is indeed the case.

```
#### Covariance matrix
vcov(her4RK)

#### Correlation matrix
cov2cor(vcov(her4RK)[,,1])
```

Chunk 10:

```
> vcov(her4RK)
, , iter = 1


              a            b
  a 41.846910942 3.640540e-04
  b  0.000364054 2.906528e-08

> cov2cor(vcov(her4RK)[, , 1])

          a        b
  a 1.000000 0.330101
  b 0.330101 1.000000
```

If despite trying various scalings any diagonal elements are still small in magnitude then there may not be not enough information in the data to estimate them. Re-parameterisation may help, e.g. as steepness and virgin biomass or the collection of more data or the use of priors. This may also make it easier to choose appropriate values which is often important when there is little information in the data.

6.2 Steepness and virgin biomass

Steepness and virgin biomass can be obtained from an FLSR object originally parameterised in terms of alpha and bet and a FLSR object can also be parameterised in terms of steepness and virgin biomass when fitting. Usually you will want to fix $SPR_{F=0}$ as there will not be enough information in the data to estimate it.

```
## fit Beverton and Holt
nsherBH       <-nsher
model(nsherBH)<-bevholt()
nsherBH       <-fmle(nsherBH)

## Get Steepness and virgin biomass
svPar<-params(sv(nsherBH,spr0=0.045))

## convert back
ab(svPar,"bevholt",spr0=0.045)

## Steepness & Virgin biomass fitting
nsherBHSV     <-nsherBH
model(nsherBHSV)<-bevholtSV()
nsherBHSV       <-fmle(nsherBHSV,fixed=list(spr0=0.045))

## note fixing spr0
profile(nsherBHSV,fixed=list(spr0=0.045))

hessian(nsherBH)
hessian(nsherBHSV)

vcov(nsherBH)
vcov(nsherBHSV)

cov2cor(vcov(nsherBHSV)[,,1])
cov2cor(vcov(nsherBH)[,,1])
```

Chunk 11:

```
iter    0 value -7.760533
iter   10 value -8.138179
final  value -8.142571
converged
An object of class "FLPar"
params
        a         b      spr0
59092.588   765.053     0.045
units:  NA NA NA
iter    0 value -4.008721
iter   10 value -6.792729
iter   20 value -7.398588
final  value -8.142571
converged

> hessian(nsherBH)
, , iter = 1


                a             b
  a -3.241851e-08  1.128209e-06
  b  1.128209e-06 -4.757217e-05

> hessian(nsherBHSV)
, , iter = 1


                s             v
  s -253.79575513 -1.432206e-02
  v   -0.01432206 -1.142553e-05

> vcov(nsherBH)
, , iter = 1


           a         b
  a 176609099 4188413.5
  b   4188413  120352.0

> vcov(nsherBHSV)
, , iter = 1


              s           v
  s  0.004240112    -5.31504
  v -5.315039851 94185.79130

> cov2cor(vcov(nsherBHSV)[, , 1])

             s          v
  s  1.0000000 -0.2659654
  v -0.2659654  1.0000000
> cov2cor(vcov(nsherBH)[, , 1])

           a         b
  a 1.0000000 0.9084822
  b 0.9084822 1.0000000
```

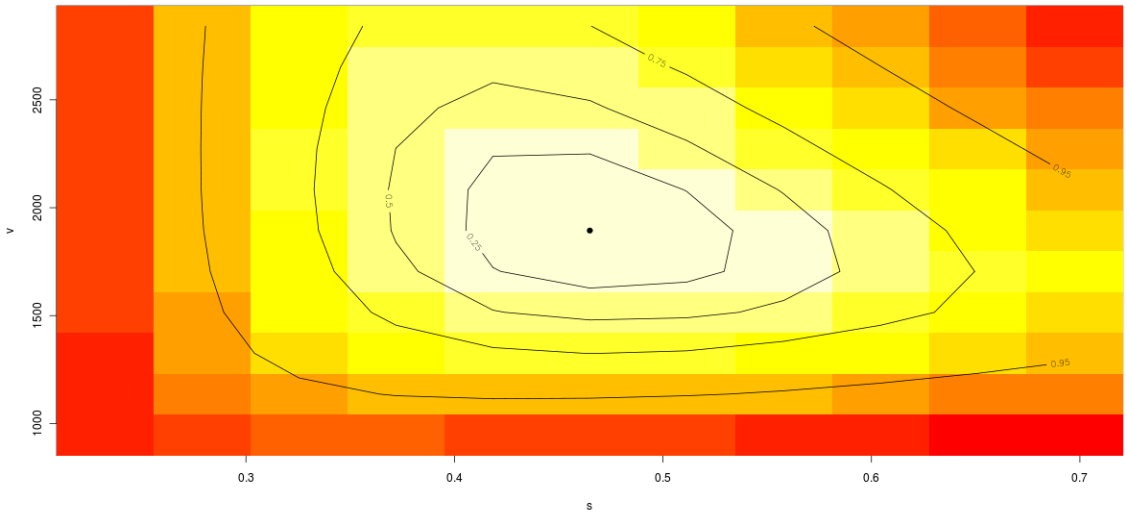

Figure 9. Fit diagnostics

Often these is also insufficient information to estimate steepness since there is no data on how recruitment declines at low biomass and so this may be fixed. For example if you want to evaluate the effect of different values of steepness on biological reference points etc.

```
nsherBHSV<-fmle(nsherBHSV,fixed=list(s=0.9, spr0=0.045))
```

Chunk 12:

As always in non-linear fitting care should be taken to ensure that the actual solution has been found. This is discussed in the following sections.

6.3 Problems with fitting

An example of a problem with fitting is shown using the ple FLStock data set and segmented regression

```
#### Problems with fitting
pleSegR<-as.FLSR(ple4)

## fit segmented regression
model(pleSegR)<-segreg()

## inspect default starting values
initial(pleSegR)(rec(pleSegR),ssb(pleSegR))

## fit
pleSegR       <-fmle(pleSegR)

## Inspect results
plot(pleSegR)

## Check fit
profile(pleSegR)
```

Chunk 13:

A cursory inspection of the plot suggests a reasonable fit, however the profile shows that the fit isn´t at the solution

```
An object of class "FLPar"
params
         a          b
3.1052e+00 2.9712e+05
units:  NA
iter    0 value -15.715675
final  value -17.272109
converged
```

On first glance the fit looks OK, but looking at the likelihood profile shows that in fact a solution has not been found and that the breakpoint should fall at a lower level of biomass.

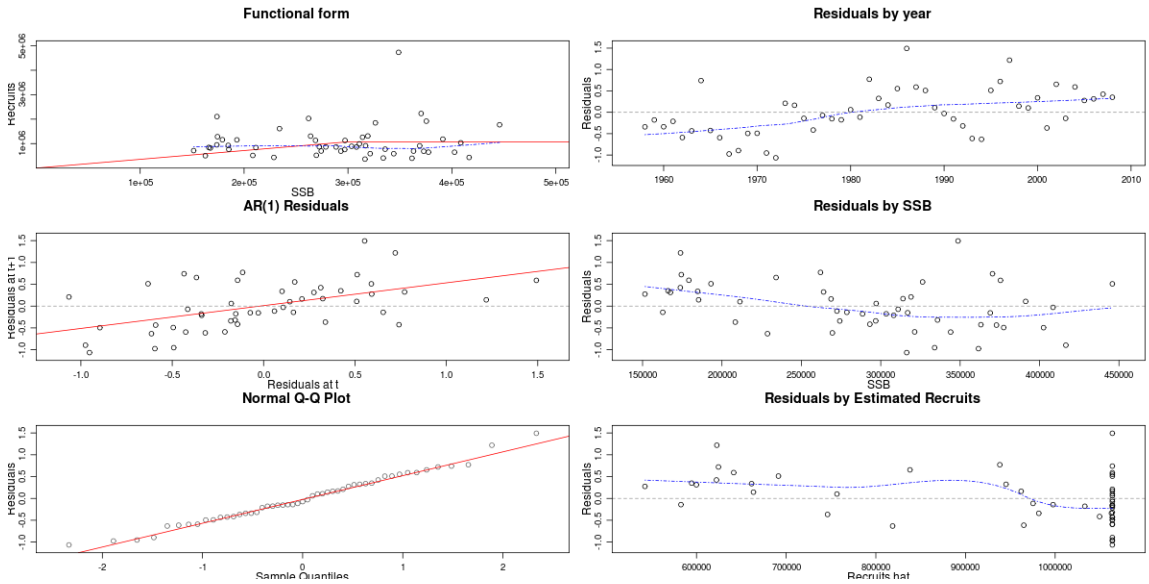

Figure 10. Fit diagnostics

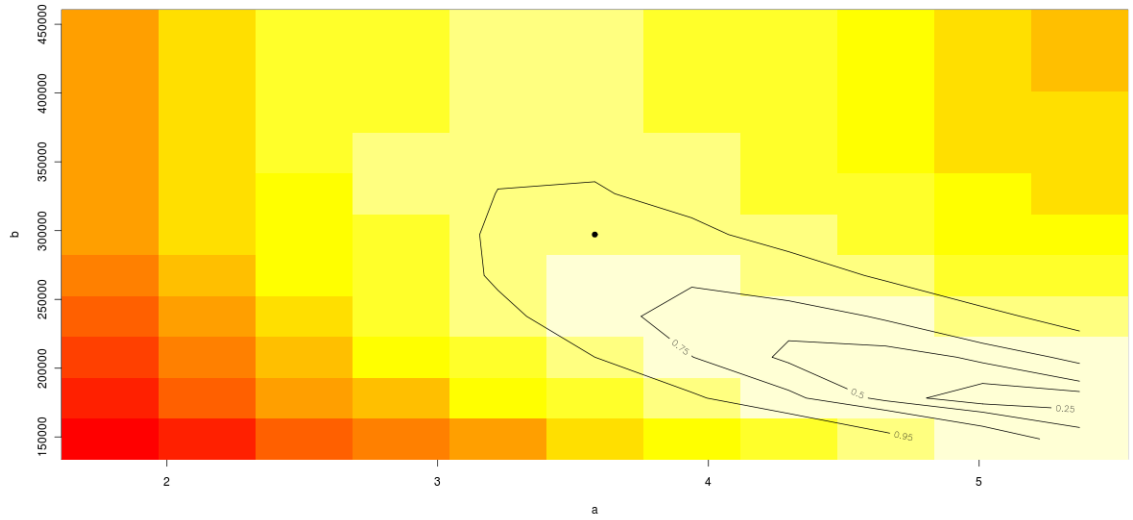

Figure 11. Likelihood profile of breakpoint

Since the default settings did not provide a solution, an option is therefore to scale the data.

```
#### Problems with fitting II

## scale data
pleSegR2<-transform(pleSegR,ssb=ssb/1000,rec=rec/1000)

## re-fit
pleSegR2<-fmle(pleSegR2)

## Inspect results
plot(pleSegR2)

## Check fit
profile(pleSegR2)
```

Chunk 14:

```
iter    0 value -15.715675
iter   10 value -20.155888
iter   20 value -21.900114
final  value -21.900123
converged
```

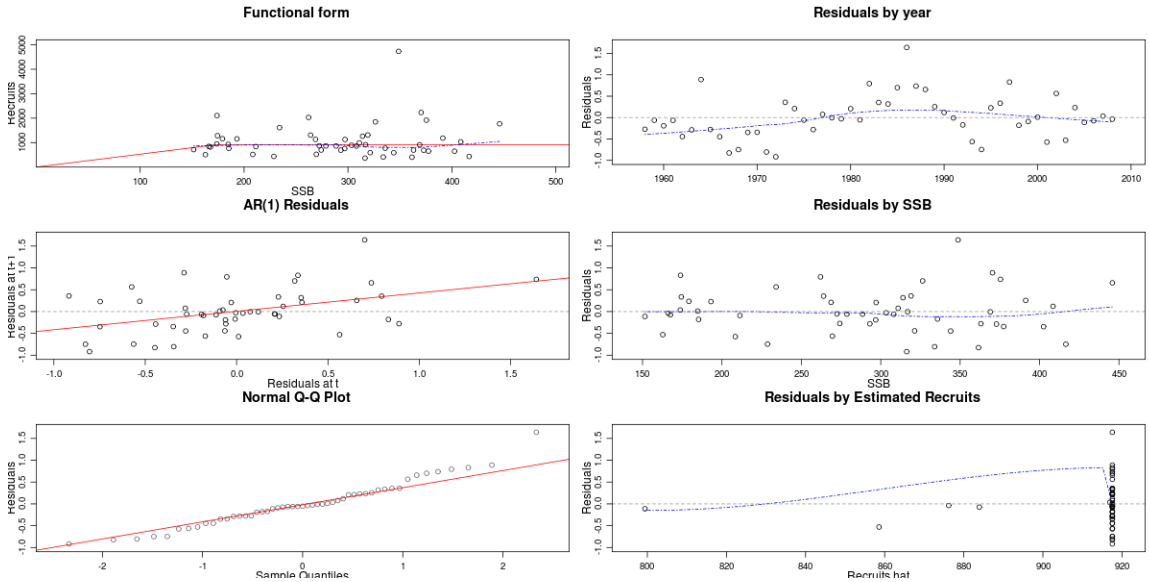

Figure 12. Likelihood profile of breakpoint

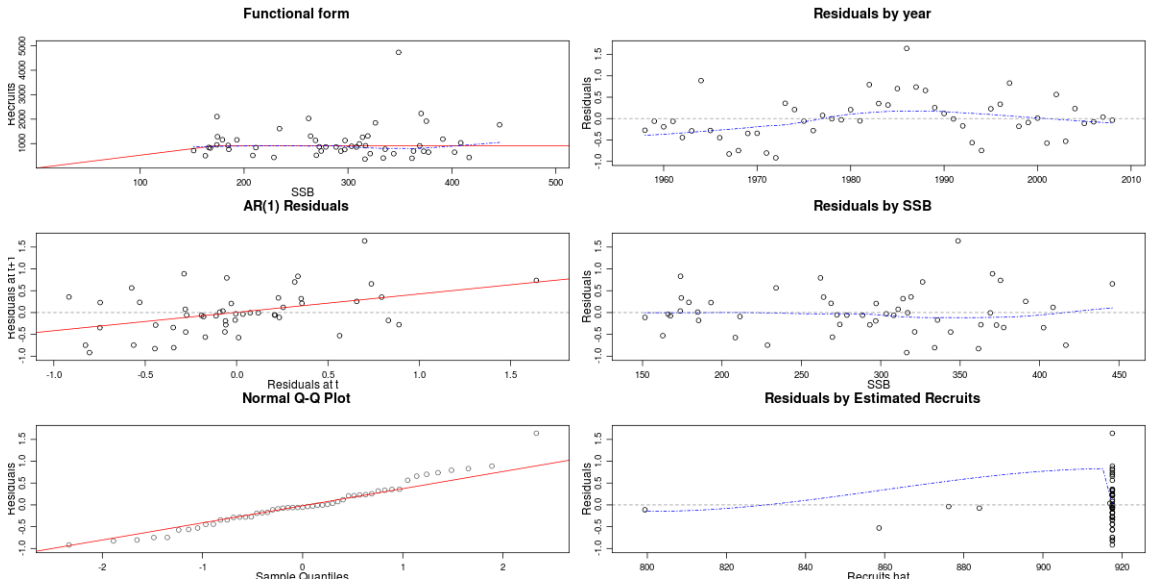

Figure 13. Fit diagnostics

What caused this problem? The hessians are quite different.

```
> hessian(pleSegR)
, , iter = 1


                a             b
  a -1.338628e+01 -8.068124e-05
  b -8.068124e-05 -3.329941e-04

> hessian(pleSegR2)
, , iter = 1


            a          b
  a -7.3909523 -0.1984453
  b -0.1984453 -6.1038295
```

Lets repeat the North Sea herring example above but first creating an FLSR object from an FLStock object.

```
#### Create FLSR object
her4SR<-as.FLSR(her4,model="ricker")

#### Fit
her4SR<-fmle(her4SR)

#### Inspect
params(her4SR)

vcov(her4SR)
```

Chunk 15:

```
iter    0 value -9.970266
final  value -9.970266
converged
```

Although the model seems to have been fitted the covariance matrix is full of NAs because the hessian has 0s in the off diagonal elements.

```
> params(her4SR)
An object of class "FLPar"
params
         a          b
6.8155e+01 6.0505e-07
units:  NA
> vcov(her4SR)
, , iter = 1


    a  b
  a NA NA
  b NA NA

> hessian(her4SR)
, , iter = 1


             a       b
  a -0.02681901   0e+00
  b  0.00000000 -5e+105
```

The reason for this is that the default minimisation method used is quasi-Newton which only approximates the hessian. and may only provide a very crude estimate which is not even certain to converge to the true hessian at the solution and so the covariance matrix should only be used with caution. The hessian can be also be calculated using the `computeHessian` method of FLR (rather than by fmle) which provides a better approximation, this can then be used to estimate the covariance matrix.

```
#### Hessian
hessian(her4SR)
computeHessian(her4SR)

#### compute covariance matrix from inverse of Hessian,
#### remember it is the negative log likelihood
-solve(computeHessian(her4SR))

#### correlation matrix
cov2cor(-solve(computeHessian(her4SR)))
```

Chunk 16:

`hessian()` extracts the hessian slot from FLSR, while `computeHessian()` calculates it for you. A subtle but important difference, if your fit fails or you haven´t fitted the object yet you might not have an estimate of the hessian. `computeHessian()` will use the parameter values (or the initial values if not fit has been made) to calculate it for you.

```
> computeHessian(her4SR)
              a             b
a   -0.02681902  3.328881e+03
b 3328.88054951 -2.743106e+12
> -solve(computeHessian(her4SR))
             a            b
a 3.729258e+01 4.525621e-08
b 4.525621e-08 3.646052e-13
> cov2cor(-solve(computeHessian(her4SR)))
           a          b
a 1.00000000 0.01227313
b 0.01227313 1.00000000
```

When fitting the data should be of an ‘appropriate’ scale, i.e with similar means and variances. Ideally the parameter should also be similar in scale to each other since fitting may terminate when a small absolute change in a parameter no longer causes the likelihood to change much. For example in segmented regression the parameters are the break point and the slope at the origin, therefore one parameter has the same units as SSB and the other recruits per SSB. Scaling the data can be done using the transform method, and scaling both recruits and SSB by the mean will make both parameters similar in scale

```
#### Transform the data
her4SR<-transform(her4SR,rec=rec/mean(rec),ssb=ssb/mean(ssb))
her4SR<-fmle(her4SR)

vcov(her4SR)
```

Chunk 17:

Problem now is parameters are no longer on the same scale as the data so it will be difficult to use the fitted object, either to calculate reference points or make projections.

```
#### Transform the data
her4SR<-as.FLSR(her4,model="segreg")
mnRec<-mean(rec(her4SR))
mnSSB<-mean(ssb(her4SR))

her4SR<-transform(her4SR,rec=rec/mnRec,ssb=ssb/mnSSB)
her4SR<-fmle(her4SR)

her4SR<-transform(her4SR,rec=rec*mnRec,ssb=ssb*mnSSB)
params(her4SR)["a"]<-params(her4SR)["a"]*mnSSB/mnRec
params(her4SR)["b"]<-params(her4SR)["b"]*mnSSB
her4SR<-fmle(her4SR,start=params(her4SR))
```

Chunk 18:

A rule of thumb is that the diagonal elements of the hessian should be a similar order of magnitude, i.e. the shape of the likelihood should be symmetric inspection of hessians above showed that when fitting failed the this wasn´t the case but was when fitting was successful. Put mathematically the aim of scaling is to optimise the condition of the hessian since convergence rate is related to the eigenvalues of the Hessian; in particular to the ratio of the smallest eigenvalue to the largest one. A poor condition number for the Hessian implied by the initial parameter will indicate whether the updates from an initial starting value are likely to be poor. An alternative to scaling the data is therefore to scale the parameters. This can be done using the parscale control option so that a unit change in the parameters produces a unit change in the likelihood, if you have the gradient, i.e. the derivatives of the likelihood with respect to each parameter then you know how much the likelihood changes for a change in each parameter. Therefore the appropraite scaling is to divide the parameters by the inverse of their gradient. A problem is that you need to do this before you´ve actually found the likelihood but hopefully the gradient should give you the right orders of magnitudes and anyway if you were at the maximum likelihood the derivatives would be zero. Doing this should result in the diagonal elements of the Hessian being about all the same magnitude, however since this is done internally within fmle it can be difficult to obtain the actual hessian used by fmle.

```
library(numDeriv)

computeGrad=function(object,method="Richardson",
      method.args=list(eps=1e-4, d=0.0001, zero.tol=sqrt(.Machine$double.eps/7e-
7), r=4, v=2, show.details=FALSE), ...){

     ## wrapper function from grad to call
     fn=function(x,sr){
       x.         =as.list(x)
       names(x.)  =dimnames(x)$params

        x.[["ssb"]]=sr@ssb
        x.[["rec"]]=sr@rec

       logl.      =sr@logl
       res        =do.call("logl.",x.)

       return(res)}

     grad(fn,x=c(object@initial(object@rec,object@ssb)),
               method=method,method.args=method.args,
               sr=object)}

her4SR<-fmle(her4SR,control=list(parscale=1/abs(computeGrad(her4SR))))
```

Chunk 19:

```
her4SV<-as.FLSR(her4,model="bevholtSV")
her4SV<-fmle(her4SV,fixed=list(spr0=0.045))
profile(her4SV,which=c("s","v"),fixed=list(spr0=0.045))

her4SV<-transform(her4SV,ssb=ssb/1000,rec=rec/1000)
her4SV<-fmle(her4SV,fixed=list(spr0=0.045))
profile(her4SV,which=c("s","v"),fixed=list(spr0=0.045))
```

Chunk 20:

```
> her4SV <- as.FLSR(her4, model = "bevholtSV")
> her4SV <- fmle(her4SV, fixed = list(spr0 = 0.045))
iter    0 value -4.008704
final  value -6.792683
converged
> profile(her4SV, which = c("s", "v"), fixed = list(spr0 = 0.045))
```

Figure 14. Fit diagnostics

```
> her4SV <- transform(her4SV, ssb = ssb/1000, rec = rec/1000)
> her4SV <- fmle(her4SV, fixed = list(spr0 = 0.045))
iter    0 value -4.008721
iter   10 value -6.792729
iter   20 value -7.398588
final  value -8.142571
converged
```

```
> profile(her4SV, which = c("s", "v"), fixed = list(spr0 = 0.045))
```

Figure 15. Fit diagnostics

If after fitting the hessian is not positive definite or there are negative variances in the covariance matrix then the approximation used to calculate the hessian may have broken down (e.g. if the likelihood is not quadratic). This is potentially a problem with FLModel/FLSR as it uses quasi-Newton methods which provide very crude approximations to the Hessian, and may not even be certain to converge to the true Hessian at the solution. Therefore typically quasi-Newton methods are not used to solve nonlinear least squares problems. However, if a quasi-Newton method is used to solve a least squares problem, then you shouldn´t use the hessian to estimate the covariance of the fitted parameters. More commonly, Gauss-Newton and Levenberg-Marquardt methods are used to solve least squares problems. Here, the Hessian of the least squares problem is approximated as 2*J'*J, where J is the Jacobian. This ignores second derivative terms, but when this approximation is used within Newton's method it tends to work well in practice on problems with small residuals. Another way of getting the Jacobian is to use Automatic Differentiation and this can be used to provide the Jacobian for optim and turn it into a Guass-Newton method. FLash implements the SRR in Adol-C and loading FLash will result in the correct hessian being returned.

7 Advanced operations with FLSR

7.1 Defining new stock recruitment relationships

To define a new model requires specifying it´s i) functional form, ii) likelihood iii) bounds and iv) starting values. For example the Deriso-Schnute model that can be used to model a variety of functional forms like the shepherd.

```
#### Deriso Schnute
dersch<-function(){
  logl <- function(a,b,c,rec,ssb) {
          res<-loglAR1(log(rec), log(a*ssb*(1-b*c*ssb)^(1/c)))
          return(res)
          }

  ## initial parameter values
  initial <- structure(function(rec, ssb){
     slopeAt0 <- max(quantile(c(rec)/c(ssb), 0.9, na.rm = TRUE))
     maxRec   <- max(quantile(c(rec), 0.75, na.rm = TRUE))

     ## Bevholt by default c=-1
     return(FLPar(a=slopeAt0, b=1/maxRec, c=-1))},

  lower=rep(-Inf, 3),
      upper=rep( Inf, 3))

  model  <- rec~a*ssb*(1-b*c*ssb)^(1/c)

  return(list(logl = logl, model = model, initial = initial))}


model(nsher)<-dersch()
nsher<-fmle(nsher,fixed=list(c=-1))
```

Chunk 21:

For large values of the "shape" parameter c the model reduces to the Ricker model as we then get Rec=a*ssb*exp(-b*ssb). While if c = -1 we get the Beverton and Holt model.

```
iter    0 value 5.146190
final  value -7.741517
stopped after 4 iterations
```

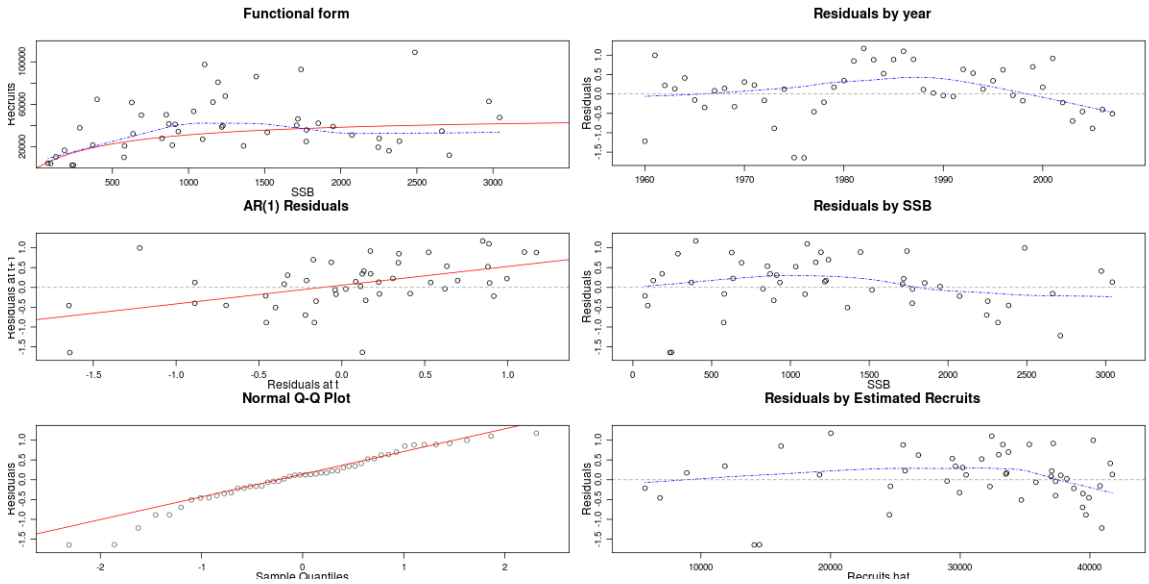

Figure 16. Fit diagnostics

8 Covariates

Covariates, can be used to model the effect of temperature on the stock recruitment relationship, se (Kell et al, 2005) where two alternative stock-recruitment relationships (SRR) were hypothesized based upon the Ricker form (Ricker, 1954), where a and b are the standard Ricker parameters and d is the temperature effect. SRR I: a affected by temperature (T) is given by RZa!SSB!eb!SSB!ed!T Survival of recruits is related to temperature, and the expected recruitment at any biomass level is scaled by the same amount for a given temperature. SRR II: b affected by temperature (T) is given by RZa!SSB!ebð1d!TÞSSB Density-dependent processes are acting so that recruitment is limited by available habitat, which in turn is related to temperature. For example, if temperature increases, the available habitat and hence virgin biomass declines.

Alpha

```
#### Original Ricker
ricker <- function(){
  logl <- function(a, b, rec, ssb)
      loglAR1(log(rec), log(a*ssb*exp(-b*ssb)))

  initial <- structure(function(rec, ssb) {
            # The function to provide initial values
    res  <-coefficients(lm(c(log(rec/ssb))~c(ssb)))
    return(FLPar(a=max(exp(res[1])), b=-max(res[2])))},

  # lower and upper limits for optim()
      lower=rep(-Inf, 2),
      upper=rep( Inf, 2))

      model  <- rec~a*ssb*exp(-b*ssb)

      return(list(logl=logl, model=model, initial=initial))}

#### Modified so temperature affects larval survival
rickerCovA <- function(){
  logl <- function(a, b, c, rec, ssb, covar){
              loglAR1(log(rec), log(a*(1+c*covar[[1]])*ssb*exp(-b*ssb)))}

  initial <- structure(function(rec, ssb, covar) {
            # The function to provide initial values
    res  <-coefficients(lm(c(log(rec/ssb))~c(ssb)))
    return(FLPar(a=max(exp(res[1])), b=-max(res[2]), c=0.0))},

  # lower and upper limits for optim()
      lower=rep(-Inf, 3),
      upper=rep( Inf, 3))

      model  <- rec~a*(1+c*covar[[1]])*ssb*exp(-b*ssb)

      return(list(logl=logl, model=model, initial=initial))}
```

Chunk 22:

```
#### Modified so temperature affects larval K
rickerCovB <- function(){
  logl <- function(a, b, c, rec, ssb, covar){
              loglAR1(log(rec), log(a*(1+c*covar[[1]])*ssb*exp(-b*ssb)))}

  initial <- structure(function(rec, ssb, covar) {
            # The function to provide initial values
    res  <-coefficients(lm(c(log(rec/ssb))~c(ssb)))
    return(FLPar(a=max(exp(res[1])), b=-max(res[2]), c=0.0))},

  # lower and upper limits for optim()
      lower=rep(-Inf, 3),
      upper=rep( Inf, 3))

      model  <- rec~a*ssb*exp(-b*(1+c*covar[[1]])*ssb)

      return(list(logl=logl, model=model, initial=initial))}
```

Chunk 23:

```
nao     <-read.table("http://www.cdc.noaa.gov/data/correlation/nao.data", skip=1,
nrow=62, na.strings="-99.90")
dnms    <-list(quant="nao", year=1948:2009, unit="unique", season=1:12,
area="unique")
nao     <-FLQuant(unlist(nao[,-1]), dimnames=dnms, units="nao")

#### include NAO as covar (note that it must be a FLQuants with a single component
#### called “covar” that matches the year span of the data) and adjust the model.

herSR         <-as.FLSR(her4)
model(herSR)  <-ricker()
herSR         <-fmle(herSR)

herCovA       <-as.FLSR(her4)
herCovA       <-transform(herCovA,ssb=ssb/1000,rec=rec/1000)
model(herCovA)<-rickerCovA()
covar(herCovA)<-FLQuants(covar=seasonMeans(trim(nao, year=dimnames(ssb(herCovA))
$year)))
herCovA       <-fmle(herCovA,fixed=list(c=0))

summary(herCovA)

herCovB       <-as.FLSR(her4)
herCovB       <-transform(herCovB,ssb=ssb/1000,rec=rec/1000)
model(herCovB)<-rickerCovB()
covar(herCovB)<-FLQuants(seasonMeans(trim(nao, year=dimnames(ssb(herCovB))$year)))
herCovB       <-fmle(herCovB,fixed=list(c=0))

summary(herCovB)
```

Chunk 24:

9 Autocorrelation

```
#### Modified so AR(1) residuals
rickerAR1 <- function()
  {
  ## log likelihood, assuming normal log.
  logl <- function(a, b, rho, rec, ssb)
      loglAR1(log(rec), log(a*ssb*exp(-b*ssb)), rho=rho)

  ## initial parameter values
  initial <- structure(function(rec, ssb) {
            # The function to provide initial values
    res  <-coefficients(lm(c(log(rec/ssb))~c(ssb)))
    return(FLPar(a=max(exp(res[1])), b=-max(res[2]), rho=0))
      },
  # lower and upper limits for optim()
      lower=rep(-Inf, 3),
      upper=rep( Inf, 3)
      )

  ## model to be fitted
      model  <- rec~a*ssb*exp(-b*ssb)

      return(list(logl=logl, model=model, initial=initial))}

#### Fit
model(her4SR)<-rickerAR1()
her4AR1      <-fmle(her4SR)
```

Chunk 25:

```
iter    0 value -17.817733
final  value -17.817733
converged
```

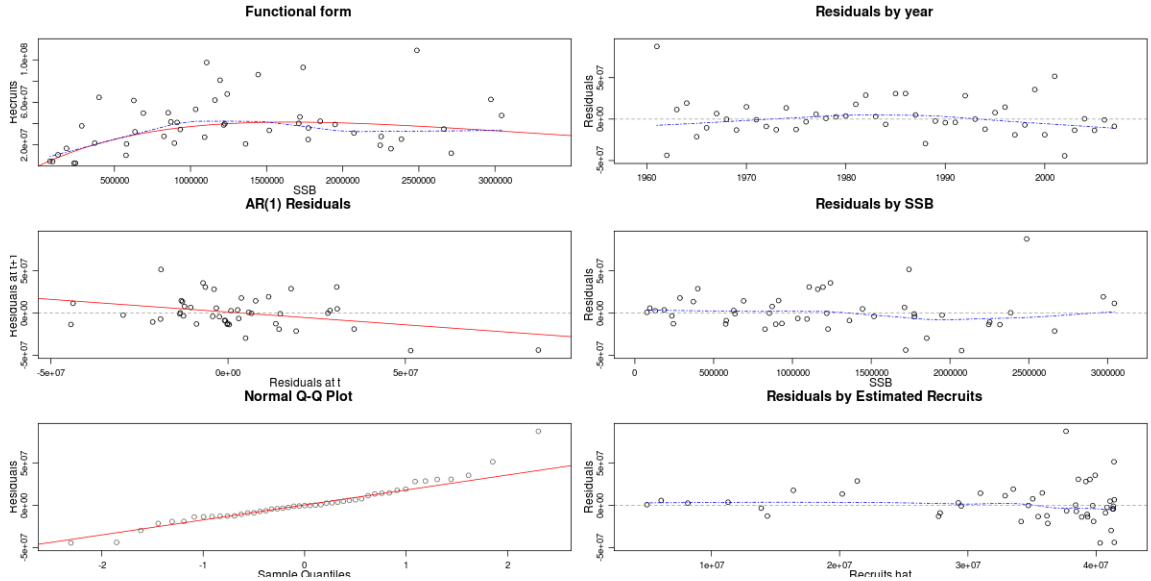

Figure 17. Fit diagnostics

10 Uncertainty

10.1 Parametric

For estimated parameters variances and covariances are given in covariance matrix, e.g. for alpha in the Beverton and Holt which represents the maximum recruitment. However, for derived values such as recruitment at a given SSB a bit more work is required. For example the delta method can be used, where the standard error of a derived quantities can be obtained using the covariance matrix and the derivatives of relationship between the quantity of interest and the estimated parameters using the Taylor expansion, i.e.

$$\begin{aligned}\operatorname{Var}(h_r) = &\sum_i \left(\frac{\partial h_r}{\partial B_i}\right)^2 \operatorname{Var}(B_i) + \\ &\sum_i \sum_{j \neq i} \left(\frac{\partial h_r}{\partial B_i}\right)\left(\frac{\partial h_r}{\partial B_j}\right) \operatorname{Cov}(B_i, B_j) \\ \operatorname{Cov}(h_r, h_s) = &\sum_i \left(\frac{\partial h_r}{\partial B_i}\right)\left(\frac{\partial h_s}{\partial B_i}\right) \operatorname{Var}(B_i) + \\ &\sum_i \sum_{j \neq i} \left(\frac{\partial h_r}{\partial B_i}\right)\left(\frac{\partial h_s}{\partial B_j}\right) \operatorname{Cov}(B_i, B_j)\end{aligned}$$

The variance covariance matrix can also be used to simulate the uncertainty of the parameters. Assuming the distributions from a multivariate normal (or multinormal) distribution, the variance covariance matrix can be decompsed into it's lower-triangular matrix (using the Cholesky decomposition) so that Or you could just simulate parameters using the covariance matrix then plug these into the equation i.e.

$S = AAt$

then a random vector X of the parameters can be generated via

$X = \mu + AZ,$

where Z is a d-dimensional vector of independent standard normal random variates.

```
# get the covariance matrix for steepness and virgin biomass
herCovar<-as.matrix(vcov(nsher)[,,1])

# calculate the lower trinagular decompostion
cholesky<-t(chol(herCovar))

cholesky

# generate a pair of random variates
c(params(nsher)[1:2,1])+cholesky%*%as.vector(rnorm(2))

# set up 1000 random variates
params(nsher)<-propagate(params(nsher),1000)

for (i in 1:1000)
   params(nsher)[c("s","v"),i]<-params(nsher)[c("s","v"),i]+cholesky%*
%as.vector(rnorm(2))

#check that these come from the original distribution
var(mc.params)
herCovar

#plot
plot(   params(nsher)[c("s","v"),])
plot(mc.params[,"a"]~mc.params[,"b"])
```

Chunk 26:

10.2 Jacknife

```
## jacknife
srJK      <-her4SR[["bevholt"]]
rec(srJK) <-jacknife(rec(srJK))

## removes a data point from each iter
iter(rec(srJK),1)
iter(rec(srJK),2)
iter(rec(srJK),3)

## set initial starting values to be equal to the first fit
params(her4SR[["bevholt"]])

model(srJK)<-bevholt()
## fits across all iters independently
srJK       <-fmle(srJK)

points(params(srJK)["s",1]~params(srJK)["v",1],col="red",pch=19)

mfJK<-model.frame(FLQuants(hat=fitted(srJK),ssb=ssb(srJK)))

p<-ggplot(mfJK, aes(x=ssb,y=hat,colour=iter)
p<-p + geom_line()
p<-p + geom_points()

## Standard Errors
jkSE<-function(a,b){
   (sum((a-b)^2)*(length(a)-1)/length(a))^.5
   }

jkSE(params(srJK)["s",],params(sr.)["s",1,drop=T])
```

Chunk 27:

11 Bootstrap

11.1 Pairs

```
dmns      <-dimnames(ssb(nsher))
dmns$iter <-1:100
mc.yrs    <-
as.integer(sample(dmns$year,length(dmns$iter)*length(dmns$year),replace=T))

sr.bt       <-nsher
rec(sr.bt)  <-FLQuant(c(rec(sr.bt)[,ac(mc.yrs)]),dimnames=dmns)
ssb(sr.bt)  <-FLQuant(c(ssb(sr.bt)[,ac(mc.yrs)]),dimnames=dmns)
model(sr.bt)<-bevholtSV()

## fits across all iters independently
sr.bt       <-fmle(sr.bt,fixed=(list(spr0=0.045)))

points(params(sr.bt)["s",]~params(sr.bt)["v",],col="yellow",pch=19)
```

Chunk 28:

11.2 Residuals

```
dmns      <-dimnames(ssb(nsher))
dmns$iter <-1:100
mc.yrs    <-
as.integer(sample(dmns$year,length(dmns$iter)*length(dmns$year),replace=T))

sr.bt       <-nsher
rec(sr.bt)  <-sweep(FLQuant(residuals(sr.bt)
[,ac(mc.yrs)],dimnames=dmns),2,fitted(nsher),"*")

model(sr.bt)<-bevholtSV()

## fits across all iters independently
sr.bt       <-fmle(sr.bt,fixed=(list(spr0=nsher.spr0)))

points(params(sr.bt)["s",]~params(sr.bt)["v",],col="yellow",pch=19)
```

Chunk 29:

11.3 Confidence Intervals

Likelihood Profiling can also be used to estimate confidence intervals using Maximum Likelihood theory and the fact that the ratio of likelihoods for models with different parameter values follows a chi-squared distribution. For example If LL(pmax) is the log-likelihood obtained by estimating all parameters and LL(p*) is the log-likelihood obtained by estimating a subset {*} of n parameters then 2[LL(pmax) – LL(p*)] ⇐ Chi-Squared(n, 1-a) can be used to calculate a (1-a)% confidence region for

the parameter set {*}.

For example, if one wanted a 95% confidence interval for a single parameter the confidence interval for p encompasses all values of p for which twice the difference between the log likelihood and the log likelihood of the best estimate of p is less than 3.84. This can be found by searching for all the values of the parameter p that result in a difference in the log-likelihoods obtained by estimating all parameters and p with a difference of 3.84.

An example of calculating confidence intervals for a single parameter i.e. alpha

```
# CI
ll<-logLik(nsher)

# Chi-squared for 1 parameter
qchisq(.95,1)

## create a function to minimise
fn<- function(x, y) {(fmle(y,fixed=c(a=x))@logLik - (ll-qchisq(.95,1)/2))^2}

## lower bound
optimise(f=fn,
   interval=c(params(nsher)["a",1,drop=T]*.5,params(nsher)["a",1,drop=T]),y=nsher)$minimum

## upper bound
optimise(f=fn,interval=c(params(nsher)["a",1,drop=T],params(nsher)["a",1,drop=T]*2.0),y=nsher)$minimum
```

Chunk 30:

Similarly,confidence regions can be constructed for more than 1 parameter. For example having already calculated a likelihood surface for the stock recruitment relationship we can create a 2 dimensional confidence interval using a chi-squared distribution with 2 degrees of freedom

```
param.grid<-profile(nsher,range=0.25)
param.grid<-data.frame(a=param.grid$x,b=param.grid$y,ll=c(param.grid$z))

image(  interp(param.grid[,"a"], param.grid[,"b"]*1000, param.grid[,"ll"]),xlab='a',ylab='b*1e+3')
contour(interp(param.grid[,"a"],          param.grid[,"b"]*1000,          param.grid[,"ll"]),add=T,
levels=(logLik(nsher)-qchisq(.95,2)/2), col="navy")
```

Chunk 31:

12 Likelihood Ratio Test

Likelihood ratios are a way of testing the goodness of fit between two models. In these tests, the two models need to be "nested", i.e., one has to be a simplification of the other. The likelihood ratio, Λ, is defined as the ratio of the maximum likelihoods of the more restricted vs the more relaxed models:

$$\Lambda = \frac{L(\theta_{simple})}{L(\theta_{complex})}$$

Twice log(Λ) is distributed as a Chi-square distribution with degrees of freedom equal to the difference in the number of parameters between the complex and simpler models:

$$-2\log(\Lambda) = -2(\log(\theta_{simple}) - \log(\theta_{complex})) \approx \chi^2_{(DF=complex-simple)}$$

How would you determine whether autocorrelation should be included in the model

Chunk 32:

13 Secondary Packages

13.1 FLash

In the Flash package FLSR is used and improved, used when making stock projections and improved since Flash uses automatic differentiation to calculate the true hessian matrix.

An example of a projection using the fwd method in Flash, see HCRs for more details

```
#### projections
```

Chunk 33:

To use Flash with FLSR you use the srModel() function, this is the same as bevholt() etc. in that it provides functions for FLSR. However, logl and gr are now versions that use automatic differentiation written in C++. So FLSR is much faster and fitting much more robust. The srModel interface is also more flexible since the "bevholt" interface requires a implemented function for every possible combination, i.e. if you have 10 functional forms, 5 likelihood types, 2 autocorelation options, 2 different ways of assuming depensation and 2 ways of modeling environmental covariates you need 10 times 5 times 2 times 2 times 2 or 400 functions. SrModel allows you this flexibility but with only

1 function.

```
#### speed I
```

Chunk 34:

```
#### speed II
```

Chunk 35:

```
#### speed III
```

Chunk 36:

```
#### robustness
```

Chunk 37:

```
#### father Jacks pants
```

Chunk 38:

13.2 FLBRP Biological reference points

Stock recruitment reltionships are needed if you want to estimate MSY based reference points, once you have an FLSR object you can pass it to FLBRP and estimate these.

```
#### Ref pts
her4Brp <-FLBRP(her4,sr=her4RK)
her4Brp <-brp(her4Brp)
plot(her4Brp)

refpts(her4Boot)[c("fmax","msy"),c("harvest","yield","ssb")]


her4Boot <-FLBRP(her4,sr=bootPair)
refpts(her4Boot)[c("fmax","msy"),c("harvest","yield","ssb")]

plot(refpts(her4Boot)[c("fmax","msy"),c("harvest","yield","ssb")])
```

Chunk 39:

14 Future?

ADMB

15 References

Nash, R.D.M., Dickey-Collas, M., and Kell, L.T., 2009. Stock and recruitment in North Sea herring (Clupea harengus); compensation and depensation in the population dynamics. Fisheries Research, 95: 88-97.

Schirripa, M. J., Goodyear, C. P., and Methot, R. M. 2009. Testing different methods of incorporating climate data into the assessment of US West Coast sablefish. – ICES Journal of Marine Science, 66: 1605–1613.