The Jack Knife

Laurence Kell

August 13th, 2014

Introduction

The jack knife can be used to estimate the variance and bias of parameter estimates and derived quantities.

Estimation

The jackknife estimate of a parameter is derived by omitting in turn the i^{th} observation from the sample data set then re-estimating the parameters and calculating the mean $\overline{\theta}_i$.

$$f(X) \simeq \overline{f} \equiv \frac{1}{N} \sum_{i=1}^{N} f_i^J$$

Variance estimation

The variance can then be calculated i.e.

$$\sigma_{f(\bar{x})} = \sqrt{N-1}\sigma_{f^J}$$

where

$$\sigma_{f^J}^2 \equiv \overline{(f^J)^2} - (\overline{f^J})^2$$

Th covariance for the α^{th} and β^{th} parameters is given by

$$COV_{\alpha,\beta} = \frac{N-1}{N} \sum_{i=1}^{N} [\theta_i^J - \overline{\theta^J}]_a [\theta_i^J - \overline{\theta^J}]_b$$

where

$$\overline{\theta_\alpha^J} = \frac{1}{N} \sum i = 1 N \theta_J^{i,\alpha}$$

The reason the sample covariance matrix has $N-1$ as the denominator rather than N is because the population mean $E(x)$ is not known and is replaced by the sample mean.

Bias estimation and correction

The bias of the estimator, calculated over the entire sample, can be estimated by

$$f(X) \simeq Nf(\bar{x}) - (N-1)\overline{f^J}$$

This reduces bias by an order of magnitude, from $O(N^{-1})$ to $O(N^{-2})$.

Simulation

Simulate an object with known properties using the biodyn package

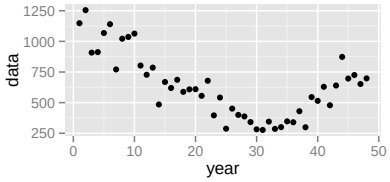
```
library(plyr)
library(biodyn)

bd=simBiodyn()
bd=window(bd,end=49)
```

A proxy for stock abundance is also required, therefore an unbiased catch per effort (CPUE) series is generated from mid year stock biomass.

```
cpue=(stock(bd)[,-dims(bd)$year]+stock(bd)[,-1])/2
cpue=rlnorm(1,log(cpue),.2)

ggplot(cpue)+geom_point(aes(year,data))
```



Perform an assessment

```
#set parameters
setParams(bd) =cpue
setControl(bd)=params(bd)
control(bd)[3:4,"phase"]=-1

#fit
bdHat=fit(bd,cpue)
```

```
[1] TRUE
```

```
plot(biodyns("Estimate"=bdHat,"Actual"=bd))+
  theme(legend.position="bottom")
```

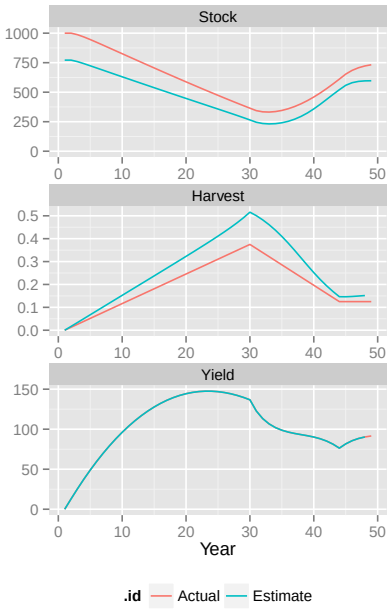


Figure 1: Fitted compared to actual time series

Jack knife procedure

```
bdJK=fit(bdHat,jackknife(cpue))

plotJack(bdHat,bdJK)


library(ggplotFL)
true=stock(bd   )[,49]
hat =stock(bdHat)[,49]
jack=stock(bdJK )[,49]
```

FLQuant

```
n   =dims(jack)$iter
mn  =apply(jack, 1:5, mean)
rsdl=sweep(jack,1:5,mn,"-")
ss  =apply(rsdl^2,1:5,sum)

bias         =(n-1)*(hat-mn)
biasCorrected=n*hat-(n-1)*mn

se  =sqrt(((n-1)/n)*ss)
```

FLPar

```
true=bmsy(bd   )
hat =bmsy(bdHat)
jack=bmsy(bdJK )

n   =dims(jack)$iter
mn  =apply(jack,  seq(length(dim(jack))-1), mean)
rsdl=sweep(jack,  seq(length(dim(jack))-1), mn,"-")
ss  =apply(rsdl^2,seq(length(dim(jack))-1),sum)

bias         =(n-1)*(hat-mn)
biasCorrected=n*hat-(n-1)*mn

se  =sqrt(((n-1)/n)*ss)
```

FLPars

```
true=refpts(bd   )
hat =refpts(bdHat)
jack=refpts(bdJK )

n   =dims(jack)$iter
```

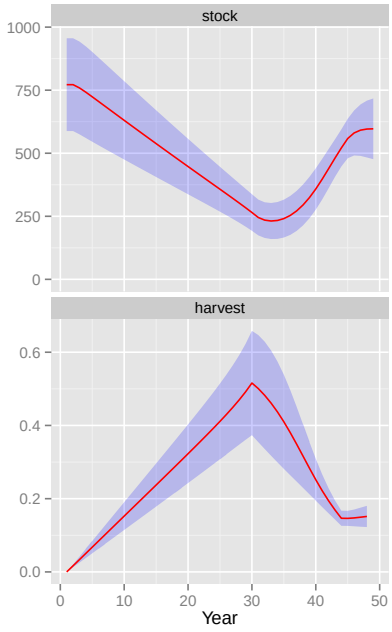


Figure 2: Jack knife error bars

```
mn  =apply(jack,  seq(length(dim(jack))-1), mean)
rsdl=sweep(jack,  seq(length(dim(jack))-1), mn,"-")
ss  =apply(rsdl^2,seq(length(dim(jack))-1),sum)

bias         =(n-1)*(hat-mn)
biasCorrected=n*hat-(n-1)*mn
se           =sqrt(((n-1)/n)*ss)
cov          =FLPar(cov(model.frame(rsdl)[,dimnames(rsdl)[[1]]])*(n-1)*(n-1)/n)
```

Perform an assessment with Monte Carlo simulation of CPUE series

Maybe perform a Monte Carlo simulation of the Jack knife to see how well it works.

Simulations

```
library(ggplotFL)
bd=simBiodyn()
bd=window(bd,end=49)

cpue=(stock(bd)[,-dims(bd)$year]+stock(bd)[,-1])/2
cpue=rlnorm(250,log(cpue),.2)

plot(cpue)


#set parameters
setParams(bd) =cpue
setControl(bd)=params(bd)
control(bd)[3:4,"phase"]=-1

#fit
catch(bd)=propagate(catch(bd),250)
bdHat=fit(bd,cpue)

plot(biodyns("Hat"=bdHat,"True"=bd))

median(stock(bd)[,49])
median(stock(bdHat)[,49])
```

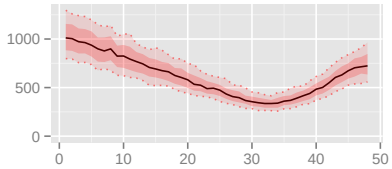


Figure 3: Simulated CPUE series