Biomass Dynamic Management Procedures

Laurence Kell

August 13th, 2016

The `mpb` package implements methods for provding management advice based based on biomass dynamic stock assessment models and for this advice to simulation tested using Management Strategy Evaluation (MSE).

The provision of fisheries management advice requires the assessment of stock status relative to reference points, the prediction of the response of a stock to management, and checking that predictions are consistent with reality. Although biomass dynamic models have been criticised as being too simplistic to capture the population dynamics. If a simple model can provide robust advice on stock status and the response of a stock to management why use anything more complicated (Ludwig and Walters 1985)? For example the Pella-Tomlinson model is used by the IWC to set catch limits. Neither the form of the model nor its parameters are meant to provide an accurate representation of the dynamics of the population. Rather, it has been demonstrated by simulation that when a biomass dynamic model is used as an integral part of a management strategy with a HCR it allows the robust calculation and setting of catches limits Butterworth and Punt (1999).

The Class

The package has nethods for importing data, exporting results, fitting models, checking diagnostics, plotting, estimation of uncertainly, projection using harvest control rules (HCRs), and for the provision of advice which can be simulation tested using Management Strategy Evaluation (MSE). The `mpb`class includes slots for data (i.e. catch), parameters, estimates of historical stock status, reference points, diagnostics, and summary statistics.

The main class in `mpb` is `biodyn`. Full documentation, on slots and methods is provided e.g.

```
??biodyn
```

First an object has to be created, one way is to use the class creator.

```
bd=biodyn()
```

or to use another object, e.g. an `FLQuant` that respresents the catch

```
bd=biodyn(catch=FLQuant(100,dimnames=list(year=1990:2010)))
```

A convinient way to create an new object is from an existing one, i.e. by coercion from an FLStock

```
data(ple4)
bd =as(ple4,"biodyn")
```

Or from an existing assessment input file such as that of ASPIC e.g.

```
asp=aspic("aspic.inp")
bd =as(asp,"biodyn")
```

Simulated objects can also be created e.g.

```
bd=sim()
```

Stock assessment

The main processes influencing the dynamics of exploited populations are gains due to growth and recruitment and losses due to fishing and natural mortality. In a biomass dynamic stock assessment production function the dynamics of recruitment, growth and natural mortality are simplified into a single production function (P) e.g. that of Pella and Tomlinson (1969).

The dynamics are determined by the population growth rate in the abscence of density dependence (r) and the shape of the production function (p). if $p = 1$ then the maximum sustainable yield (MSY) is found halfway between o and virgin biomass (K); as p increases MSY shifts to the right.

There is insuffcient infomation in the catch data to estimate the few parameters and so additional data, e.g. time series of relative abundance from catch per unit effort (CPUE) or surveys are required for calibration.

$$B_{t+1} = B_t - C_t + P_t \tag{1}$$

Figure 1: Biomass next year equals the biomass this year less catches and plus production.

$$\frac{r}{p} \cdot \ B(1 - (\frac{B}{K})^p) \tag{2}$$

Figure 2: Production function.

Methods

Plotting can be used to examine an object, explore data, check outputs, diagnose problems, and summarise results. biodyn uses ggplot2 as this allows a variety of basic plots to be provided as part of the package and these to be modified and new plots developed as required.

```
bd=sim()
```

```
Warning in ifelse(!
is.na(suppressWarnings(as.numeric(quants[1]))),
as.numeric(quants[1]), : NAs introduced by
coercion
```

```
Warning in ifelse(!
is.na(suppressWarnings(as.numeric(quants[max(dims[,
1])]))), : NAs introduced by coercion
```

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

```
Warning in ifelse(!
is.na(suppressWarnings(as.numeric(quants[1]))),
as.numeric(quants[1]), : NAs introduced by
coercion
```

```
Warning in ifelse(!
is.na(suppressWarnings(as.numeric(quants[1]))),
as.numeric(quants[1]), : NAs introduced by
coercion
```

```
plot(bd)+theme_bw()
```

```
Warning in ifelse(!
is.na(suppressWarnings(as.numeric(quants[1]))),
as.numeric(quants[1]), : NAs introduced by
coercion
```

```
Warning in ifelse(!
is.na(suppressWarnings(as.numeric(quants[1]))),
as.numeric(quants[1]), : NAs introduced by
coercion
```

```
x=sim()
```

```
Warning in ifelse(!
is.na(suppressWarnings(as.numeric(quants[1]))),
as.numeric(quants[1]), : NAs introduced by
coercion
```

```
Warning in ifelse(!
is.na(suppressWarnings(as.numeric(quants[max(dims[,
1])]))), : NAs introduced by coercion
```

```
plotPrd(x)+
  geom_path( aes(stock,catch),
             model.frame(FLQuants(x,"stock","catch")))+
  geom_point(aes(stock,catch),
             model.frame(FLQuants(x,"stock","catch")))+
  theme_bw()+theme(legend.position="none")
```

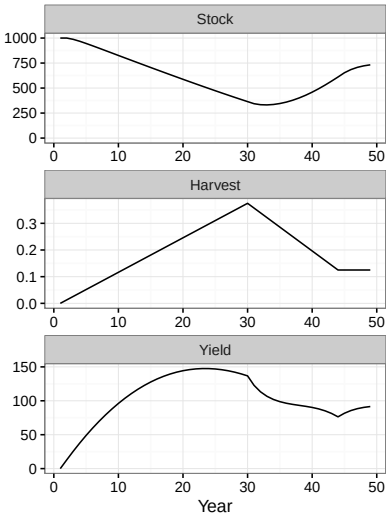


Figure 3: Production function with simulated time series

```
Warning: Removed 1 rows containing missing
values (geom_path).
```

```
Warning: Removed 1 rows containing missing values
(geom_point).
```

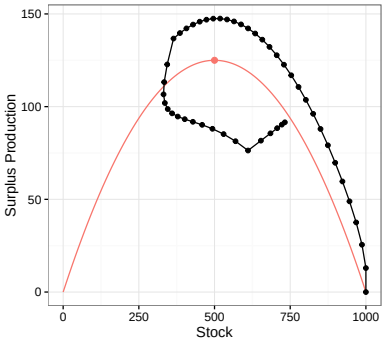


Figure 4: Simulated CPUE series

Estimation

FITTING TO DATA can be done using either maximum likelihood or by running Monte Carlo Markov Chain (MCMC) simulations.

When first using a stock assessment it helps to be able to check estimated values with the true ones. Therefore we simulate a stock with know parameters and exploitation history

```
bd=sim()
```

```
Warning in ifelse(!
is.na(suppressWarnings(as.numeric(quants[1]))),
as.numeric(quants[1]), : NAs introduced by
coercion
```

```
Warning in ifelse(!
is.na(suppressWarnings(as.numeric(quants[max(dims[,
1])]))), : NAs introduced by coercion
```

A CPUE series is also needed for fitting and can be simulated that by taking the mid year biomass and adding error.

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

```
Warning in ifelse(!
is.na(suppressWarnings(as.numeric(quants[1]))),
as.numeric(quants[1]), : NAs introduced by
coercion
```

```
Warning in ifelse(!
is.na(suppressWarnings(as.numeric(quants[max(dims[,
1])]))), : NAs introduced by coercion
```

```
cpue=rlnorm(1,log(cpue),.2)
```

```
ggplot(as.data.frame(cpue))+
  geom_point(aes(year,data))+
  geom_line(aes(year,data),data=as.data.frame(stock(bd)),col="salmon")+
  theme_bw()
```

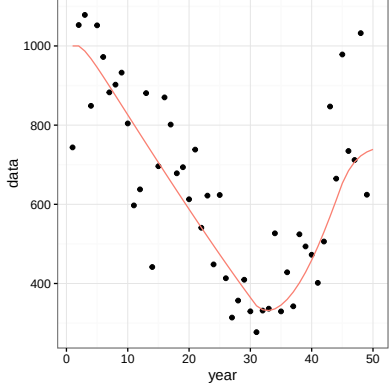


Figure 5: Simulated CPUE series

Starting values for parameters are also required. These can be set by informed guesses. If you know the catch then MSY should be somewhere close and if you can provide a guess for r (the default is 0.5) then carrying capacity (k) can be calculated; by default is assummed to be symetric (i.e. p=1) and B_0 (the ratio of the initial biomass to carrying capacity) can be set to 1 if data are available from the start of the fishery. The robustness of fixing any parameters should be checked.

```
bd=biodyn(catch=catch(bd))
```

The constructor also calculates the stock based on the initial parameters and catch and this allows catchability and the CV of the fit of the CPUE index to be calculated.

```
setParams( bd)=cpue
params(bd)
```

```
An object of class "FLPar"
params
         r          k          p         b0
   0.50000 1000.00000    1.00000    1.00000
        q1     sigma1
   1.03283    0.19096
units:  NA
```

The params slot holds the fitted parameters. But before fitting the control slot has to be provided with initial guesses, upper and lower bounds (min and max) and any difficult to estimate parameters to be fixed, i.e. setting the phase of B_0 and p to be 1. Parameters can be estimated sequentially by setting phase >0.

```
setControl(bd)=params(bd)
```

```
bd@control
```

```
An object of class "FLPar"
        option
params   phase       min         val
  r       1.0000e+00  5.0000e-02  5.0000e-01
  k       1.0000e+00  1.0000e+02  1.0000e+03
  p      -1.0000e+00  1.0000e-01  1.0000e+00
  b0     -1.0000e+00  1.0000e-01  1.0000e+00
  q1      1.0000e+00  1.0328e-01  1.0328e+00
  sigma1  1.0000e+00  1.9096e-02  1.9096e-01
```

```
        option
params   max
  r       5.0000e+00
  k       1.0000e+04
  p       1.0000e+01
  b0      1.0000e+01
  q1      1.0328e+01
  sigma1  1.9096e+00
units:  NA
```

Help

Example datasets

Methods

Data

Fitting

Diagnostics

Harvest control rules

Simulation

Examples

Validation I

Validation II

MSE

Maximum Likelihood

Estimation can be performed using maximum likelihood

```
bd@control[3:4,"phase"]=-1
bdHat=fit(bd,cpue)

# plot(biodyns("True"=bd,"Hat"=bdHat))+
#   theme(legend.position="bottom")
#save(bdHat,file="/home/laurie/Desktop/flr/git/biodyn/data/bdHat.RData")
```

Since the true parameters are known then we can check the fits.

```
params(bdHat)
```

```
An object of class "FLPar"
```

```
params
        r         k         p        b0
  0.56086 905.99768   1.00000   1.00000
       q1    sigma1
  1.12868   0.96441
units:  NA
```

```
params(bdHat)/params(bd)
```

```
An object of class "FLPar"
params
     r      k      p     b0     q1 sigma1
1.1217 0.9060 1.0000 1.0000 1.0928 5.0504
units:  NA
```

Diagnostics

GOODNESS OF FIT diagnostics are important for transparency, replicability and ensuring that a global solution has actually been found, i.e. that when the assessment is repeated that you get the same solution.

Residual Patterns

Patterns in residuals of the fits to the CPUE and stock abundance may indicate a violation of models assumptions. Which in turn may result in biased estimates of parameters, reference points and stock trends. In addition variance estimates obtained from bootstrapping assume that residuals are Independently and Identically Distributed (i.i.d.).

Residuals are found in the diags slot.

```
head(bdHat@diags)
```

```
  name year       obs      hat     residual
1    1    1  743.7624 1128.680 -0.417082453
2    1    2 1052.8168 1121.382 -0.063093065
3    1    3 1078.3876 1103.280 -0.022820896
4    1    4  848.7870 1080.046 -0.240950786
5    1    5 1052.0443 1054.446 -0.002280081
6    1    6  972.1623 1027.772 -0.055625968
   residualLag        qqx          qqy
1 -0.063093065 -1.8718707 -0.417082453
2 -0.022820896  0.1540745 -0.063093065
3 -0.240950786  0.4213913 -0.022820896
4 -0.002280081 -0.8636957 -0.240950786
5 -0.055625968  0.5962318 -0.002280081
6 -0.125335179  0.2586331 -0.055625968
        qqHat
1 -0.40585929
2 -0.06057707
3 -0.01501822
4 -0.23403582
5  0.01477988
6 -0.04275713
```

Normally Distributed

Checking the distribution of residuals can be done by plotting the obsevered quantiles against the predicted quantiles from the assumed

distribution.

Q-Q plots compare a sample of data on the vertical axis to a statistical population on the horizontal axis, in this case a normal distribution. If the points follow a strongly nonlinear pattern this will suggest that the data are not distributed as a standard normal i.e. $X\ N(0,1)$. Any systematic departure from a straight line may indicate skewness or over or under dispersion.

```
rsdl=bdHat@diags
ggplot(rsdl)                                                    +
  geom_point( aes(qqx,qqy))                                     +
  stat_smooth(aes(qqx,qqHat),method="lm",se=T,fill="blue", alpha=0
  theme_bw()+theme(legend.position="bottom")
```

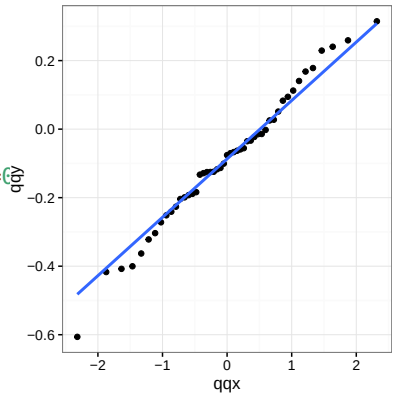


Figure 6: Quantile-quantile plot to compare residual distribution with the normal distribution.

Observed against Fitted

It is assumed that an index is proportional to the stock so when plotting the observed against the fitted values the points should fall around the $y = x$ line, if they do not then the index may not be a good proxy for the stock trend.

```
library(diags)

ggplot(with(rsdl, data.frame(obs=stdz(obs),hat=stdz(hat))))    +
    geom_abline(aes(slope=1,intercept=0))                      +
    geom_point( aes(obs,hat))                                  +
    stat_smooth(aes(obs,hat),method="lm", se=F)                +
    theme_bw()+theme(legend.position="bottom")                 +
    xlab("Fitted") + ylab("Observed")
```

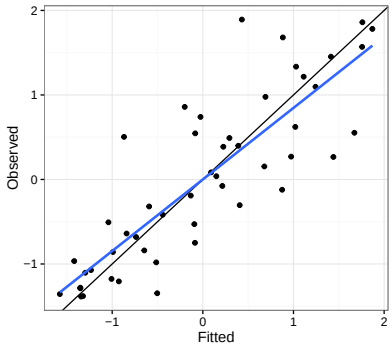


Figure 7: Observed CPUE verses fitted, blue line is a linear resgression fitted to points, black the y=x line.

Year Patterns

The residuals are plotted against year along with a lowess smoother to see if the proxy for the stock doesnt agree with the estimated stock trend,

```
dat=transform(subset(rsdl,!is.na(residual),
                     residual=stdz(residual,na.rm=T)))

ggplot(aes(year,residual),data=dat)  +
  geom_hline(aes(yintercept=0))      +
  geom_point()                       +
  stat_smooth(method="loess",se=F)   +
  theme_bw()+theme(legend.position="bottom")
```

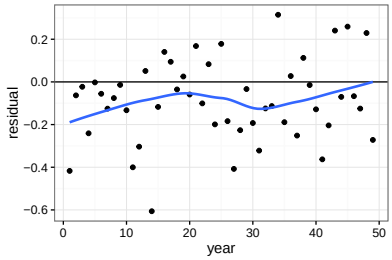


Figure 8: Residuals by year, with lowess smoother

Variance

It is also assumed that variance does not vary with the mean, this assumption can be checked by plotting the residuals against the fitted values.

```
ggplot(aes(hat, residual),
       data=subset(rsdl,!is.na(hat) & !is.na(residual)))   +
  geom_hline(aes(yintercept=0))         +
  geom_point()                          +
  stat_smooth(method="loess",se=F)      +
  theme_bw()+theme(legend.position="bottom")
```

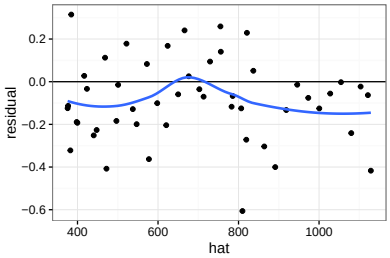


Figure 9: Plot of residuals against fitted value, to check variance relationship.

Autocorrelation

It is assumed that the residuals are not autocorrelated. Plots of the residuals against each other with a lag of 1 to identify autocorrelation. Significant autocorrelations could be due to an increase in catchability with time; which may result in a more optimistic estimate of current stock status as any decline in the stock is masked by an increase in catchability.

```
ggplot(rsdl)                                                +
  geom_point( aes(residual,residualLag))                    +
  stat_smooth(aes(residual,residualLag),method="lm",se=F) +
  geom_hline(aes(yintercept=0))     +
  xlab(expression(Residual[t]))     +
  ylab(expression(Residual[t+1]))   +
  theme_bw()+theme(legend.position="bottom")
```

```
Warning: Removed 1 rows containing non-finite
values (stat_smooth).
```

```
Warning: Removed 1 rows containing missing values
(geom_point).
```

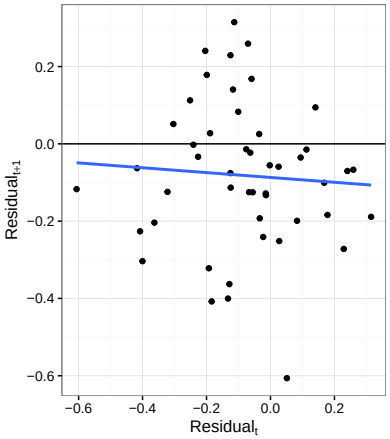


Figure 10: Plot of autocorrelation, i.e. $residual_{t+1}$ verses $residual_t$.

Profiles

Likelihood profiles are useful to check that you are actually at a global solution and not stuck on a small hill with your back to the mountain. They are also useful for evaluating the infomation content of the data and whether different data sets are telling you different things and you need to ask more questions to determine the truth.

The control slot can be used to produce a profile, i.e. fix a parameter or parameters for a range of values and then find the maximum likelihood by estimating the other parameters.

1D

```
bdHat=fit(bdHat,cpue)
setControl(bdHat)=params(bdHat)
res=profile(bdHat,which='r',fixed=c('b0','p'),
            cpue,range=seq(0.95,1.03,.002))
ggplot(subset(res,ll.ll<0))+
  geom_line(aes(r,ll.ll))  +
  theme_bw()


res=profile(bdHat,which=c('r','k'),fixed=c('b0','p'),
            cpue,range=seq(0.97,1.03,.02))
ggplot(res, aes(r, k, z=ll.ll))+
  stat_contour(aes(colour = ..level..), size = 1)+
  theme_bw()
```

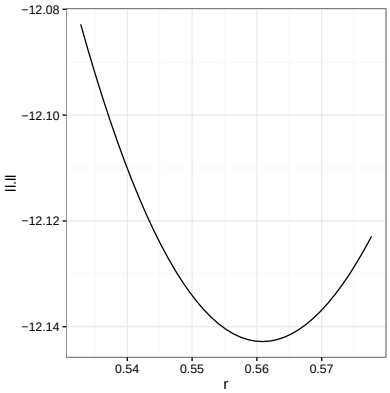


Figure 11: Likelihood profile for r

likelihood components

```
bd=sim()
```

```
Warning in ifelse(!
is.na(suppressWarnings(as.numeric(quants[1]))),
as.numeric(quants[1]), : NAs introduced by
coercion
```

```
Warning in ifelse(!
is.na(suppressWarnings(as.numeric(quants[max(dims[,
1])]))), : NAs introduced by coercion
```

```
Us  =FLQuants("Unbiased"     =
                rlnorm(1,log((stock(bd)[,-dims(bd)$year]+
                              stock(bd)[,-1])/2),0.2),
              "Increase in q"=
                rlnorm(1,log((stock(bd)[,-dims(bd)$year]+
                              stock(bd)[,-1])/2),0.2))
```

```
Warning in ifelse(!
is.na(suppressWarnings(as.numeric(quants[1]))),
as.numeric(quants[1]), : NAs introduced by
coercion

Warning in ifelse(!
is.na(suppressWarnings(as.numeric(quants[1]))),
as.numeric(quants[1]), : NAs introduced by
coercion

Warning in ifelse(!
is.na(suppressWarnings(as.numeric(quants[1]))),
as.numeric(quants[1]), : NAs introduced by
coercion

Warning in ifelse(!
is.na(suppressWarnings(as.numeric(quants[max(dims[,
1])]))), : NAs introduced by coercion
```

```
bds=bd

setParams( bds)=Us
setControl(bds)=params(bds)

bds@control[3:4,"phase"]=-1
bds=fit(bds,index=Us)
bds@control[,c("min")]=bds@params*0.1
bds@control[,c("val")]=bds@params
bds@control[,c("max")]=bds@params*10

prfl=profile(bds,which='r',index=Us,
             range=seq(0.70,1.05,.001))

ggplot(prfl)+
  geom_line(aes(r,ll,group=index,col=index))+
  facet_wrap(~index,scale="free",ncol=1)          +
  theme(legend.position="bottom")
```

Profile Slot

Stock status

A main objective of stock assessment is to estimate uncertainly in stock status. This requires estimates of distributions as well as point estimates.

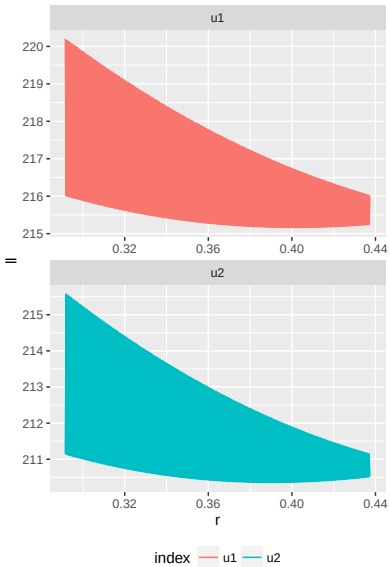


Figure 12: Likelihood profile by data conmponent, i.e. CPUE series

```
bd   =window(sim(),end=39)
cpue=(stock(bd)[,-dims(bd)$year]+
      stock(bd)[,-1])/2
cpue=rlnorm(1,log(cpue),.2)
bdHat=bd

setParams( bdHat)=cpue
setControl(bdHat)=params(bdHat)
bdHat@control[3:4,"phase"]=-1
bdHat=fit(bdHat,cpue)

sims=biodyns("True"=bd,"Best Fit"=bdHat)
```

There are various ways to estimate undercertainty in parameter estimates and quantities derived from them, i.e. use the covariance matrix provided by a maximum likelihood fit, bootstrapping, the jack knife or Bayesian methods such as Monte Carlo Markov Chain,

Variance/Covariance Matrix

Fitting using maximum likelihood provides the covariance matrix for the parameters. We can use this to conduct a Monte Carlo simulation of the parameter estimates to derive uncertainty in derived quantities.

```
vcov(bdHat)
```

```
An object of class "FLPar"
        params
params   r           k           p
  r       6.9029e-03 -9.8269e+00          NA
  k      -9.8269e+00  1.3995e+04          NA
  p               NA          NA          NA
  b0              NA          NA          NA
  q1      1.2882e-02 -1.8407e+01          NA
  sigma1  0.0000e+00  0.0000e+00          NA
        params
params   b0          q1          sigma1
  r              NA  1.2882e-02  0.0000e+00
  k              NA -1.8407e+01  0.0000e+00
  p              NA          NA          NA
  b0             NA          NA          NA
  q1             NA  2.5775e-02  0.0000e+00
  sigma1         NA  0.0000e+00  8.4456e+05
units:  NA NA NA NA NA NA
```

The Bootstrap

The Bootstrap can be used to simulate CPUE series replicates and the model refitted.

```
cpueSim=bdHat@diags[,c("year","hat")]
names(cpueSim)[2]="data"
cpueSim=as.FLQuant(cpueSim)

cv=sd(sims[["Best Fit"]]@diags[,"residual"])

cpueSim=rlnorm(100,log(cpueSim),cv)

cpueSim[cpueSim==0]=NA

plot(cpueSim,na.rm=TRUE)

sims[["CPUE"]]=fit(propagate(bdHat,100),cpueSim)
```

Jack knife

The Jack knife is a relatively quick procedure and so suitable for simulation testing

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

sims[["Jack Knife"]]=bdJK

#plotJack(bdJK)
```

MCMC

Monte Carlo Markov Chain

```
sims[["MCMC"]]=fit(bdHat,cpue,cmdOps=c("-mcmc 1000000, -mcsave 5000"))
```

Diagnostics need to be run to make sure that the results have actually estimated a stationary distribution.

```
acf(c(params(sims[["MCMC"]])["r"]))
```

Stock Status Reference Points

The Precautionary Approach requires stock status to be estimated relative to reference points. The covariance matrix can be used to estimate uncertainty in derived quantities, i.e. those used for management such as $F : F_{MSY}$.

```
bdHat@mng
```

```
bdHat@mngVcov
```

```
currentState   =bdHat@mng[c("bbmsy","ffmsy"),"hat",drop=T]
currentStateVar=bdHat@mngVcov[c("bbmsy","ffmsy"),
                              c("bbmsy","ffmsy"),drop=T]

refs=mvrnorm(100,currentState,currentStateVar)

ggplot(data=as.data.frame(refs))+
   geom_histogram(aes(x=bbmsy))
```

Marginal Density for Stock

```
#load("/home/laurie/Desktop/flr/git/biodyn/stuff/data/sims.RData")
sims[["Jack Knife"]]=bdJK

c("Best Fit","CPUE","Jack Knife","MCMC")

boot=stock(sims[["CPUE"]])[,39]
# ggplot(as.data.frame(boot))+
#   geom_density(aes(x=data, y=..count..), position = "stack",fill="red")+
#   scale_x_continuous(limits=c(0,700))

mcmc=stock(sims[["MCMC"]])[,39]
# ggplot(as.data.frame(mcmc))+
#   geom_density(aes(x=data, y=..count..), position = "stack",fill="red")+
#   scale_x_continuous(limits=c(0,700))

vcov=rnorm(500,sims[["Best Fit"]]@mng["bnow","hat"],
               sims[["Best Fit"]]@mng["bnow", "sd"])
# ggplot(as.data.frame(vcov))+
#   geom_density(aes(x=data, y=..count..), position = "stack",fill="red")+
#   scale_x_continuous(limits=c(0,1000))

jack=randJack(500,stock(sims[[  "Best Fit"]])[,39],
                  stock(sims[["Jack Knife"]])[,39])
# ggplot(as.data.frame(jack))+
```

```
#   geom_density(aes(x=data, y=..count..), position = "stack",fill="red")+
#   scale_x_continuous(limits=c(0,700))

bnow=rbind(data.frame(Method="boot",stock=c(boot)),
           data.frame(Method="mcmc",stock=c(mcmc)),
           data.frame(Method="vcov",stock=c(vcov)),
           data.frame(Method="jack",stock=c(jack)))

ggplot(bnow)+
  geom_density(aes(x=stock, y=..count..), position = "stack",fill="red")+
  facet_wrap(~Method,scale="free_y",ncol=1)+
  geom_vline(aes(xintercept=c(stock(sims[["Best Fit"]])[,"39"])))
```

Marginal Density for Harvest/FMSY

```
boot=boot%/%bmsy(sims[["CPUE"]])
mcmc=mcmc%/%bmsy(sims[["MCMC"]])
vcov=rnorm(500,sims[["Best Fit"]]@mng["bbmsy","hat"],
               sims[["Best Fit"]]@mng["bbmsy", "sd"])
jack=randJack(500,stock(sims[  "Best Fit"]])[,39]%/%bmsy(sims[  "Best Fit"]]),
                  stock(sims[["Jack Knife"]])[,39]%/%bmsy(sims[["Jack Knife"]]))

bbmsy=rbind(data.frame(Method="boot",stock=c(boot)),
            data.frame(Method="mcmc",stock=c(mcmc)),
            data.frame(Method="vcov",stock=c(vcov)),
            data.frame(Method="jack",stock=c(jack)))

ggplot(bnow)+
  geom_density(aes(x=stock, y=..count..), position = "stack",fill="red")+
  facet_wrap(~Method,scale="free_y",ncol=1)+
  geom_vline(aes(xintercept=c(stock(sims[["Best Fit"]])[,"39"])))+
  scale_x_continuous(limits=c(0,2.0))
```

Kobe Phase Plot

```
library(kobe)

kb=rbind(data.frame(Method="Boot",kobe(sims[["CPUE"]],      what="pts")),
         data.frame(Method="MCMC",kobe(sims[["MCMC"]],      what="pts")),
         data.frame(Method="Vcov",kobe(sims[["Best Fit"]],  what="pts")),
         data.frame(Method="Jack",kobe(sims[["Jack Knife"]],what="pts")))

ggplot(kb)+
  geom_point(aes(stock,harvest),data=subset(df,year==39))+
  facet_wrap(~Method,scale="free_y",ncol=1)
```

Simulation

```
source('~/Desktop/flr/git/biodyn/R/biodyn-jackRand.R')
source('~/Desktop/flr/git/biodyn/R/biodyn-jackSummary.R')

sims[["Jack Knife"]]=randJack(500,bdHat,bdJK)

sims[["Vcov"]]=bdHat
sims[["Vcov"]]=mvn(bdHat,500,nms=c("r","k"),fwd=TRUE)
```

Projections

Once stock parameters and status has been estimated then projections need to be conducted to inform management.

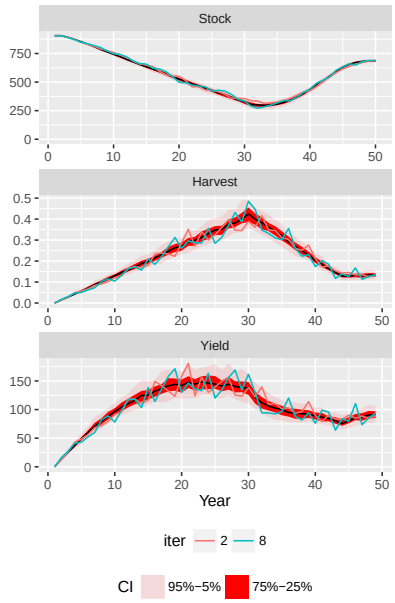
```
harvest=rlnorm(100,log(harvest(bdHat))[,-dims(bdHat)$year],.1)
```

```
Warning in ifelse(!
is.na(suppressWarnings(as.numeric(quants[1]))),
as.numeric(quants[1]), : NAs introduced by
coercion
```

```
Warning in ifelse(!
is.na(suppressWarnings(as.numeric(quants[max(dims[,
1])]))), : NAs introduced by coercion
```

```
bdHat =fwd(bdHat,harvest=harvest)

plot(bdHat,worm=c(2,8))+
  theme(legend.position="bottom")
```

```
Warning in ifelse(!
is.na(suppressWarnings(as.numeric(quants[1]))),
as.numeric(quants[1]), : NAs introduced by
coercion

Warning in ifelse(!
is.na(suppressWarnings(as.numeric(quants[1]))),
as.numeric(quants[1]), : NAs introduced by
coercion

Warning in ifelse(!
is.na(suppressWarnings(as.numeric(quants[1]))),
as.numeric(quants[1]), : NAs introduced by
coercion

Warning in ifelse(!
is.na(suppressWarnings(as.numeric(quants[max(dims[,
1])]))), : NAs introduced by coercion
```



Harvest Control Rules

Use simulated data to run annual, tri-annual, F bound and TAC bounded HCRs

Annual

```
bd   =sim()

bd=window(bd,end=29)
for (i in seq(29,49,1))
  bd=fwd(bd,harvest=hcr(bd,refYrs=i,yrs=i+1)$hvt)
simHCR=biodyns("1"=bd)
```

Tri-annual

```
bd=window(bd,end=29)
for (i in seq(29,49,3))
  bd=fwd(bd,harvest=hcr(bd,refYrs=i,yrs=i+1:3)$hvt)
simHCR[["3"]]=bd
```

Bound on F

```
bd=window(bd,end=29)
for (i in seq(29,49,1))
  bd=fwd(bd,harvest=hcr(bd,refYrs=i,yrs=i+1,bndF=c(0.9,1.1))$hvt)
simHCR[["bound F"]]=bd
```

Bound on catch

```
bd=window(bd,end=30)
for (i in seq(29,49,1))
  bd=fwd(bd,catch=hcr(bd,refYrs=i,yrs=i+1,tac=T,bndTac=c(0.9,1.1))$tac)
simHCR[["bound TAC"]]=bd

plot(simHCR)+
  theme(legend.position="bottom")
```

Stochasticity

Process Error and Harvest Control Rule

```
pe=rlnorm(500,FLQuant(0,dimnames=list(year=1:50)),0.5)

bd=window(sim(),end=30)
bd.=bd
bd@stock =propagate(bd@stock, 500)
bd=fwd(bd,harvest=harvest(bd)[,2:30],pe=pe)
```

```
for (i in seq(30,48,1))
  bd=fwd(bd,
         catch=hcr(bd,refYrs=i,yrs=i+1,tac=T,bndTac=c(0.9,1.1))$tac,
         pe   =pe)

plot(bd)
```

```
library(kobe)
trks=biodyn::kobe(bd,what="trks")
trks=mdply(data.frame(Year=seq(33,49,3)),
           function(Year) subset(trks,year<=Year))

pts =transform(biodyn::kobe(bd,what="pts",year=seq(33,49,3)),
                Year=year)[,c("stock","harvest","Year")]

kobePhase()+
    geom_line(aes(stock,harvest),data=biodyn:::hcrPlot(bd.),
              col="brown",size=1.5)                              +
    geom_path( aes(stock,harvest),data=subset(trks,pctl=="50%"))+
    geom_point(aes(stock,harvest),data=subset(pts,Year>=33))    +
    facet_wrap(~Year)
```

MSE

```
biodyn:::mseBiodyn
```

References

Butterworth, DS, and AE Punt. 1999. "Experiences in the Evaluation and Implementation of Management Procedures." *ICES J. Mar. Sci.* 56 (6). Oxford University Press: 985–98.

Ludwig, Donald, and Carl J Walters. 1985. "Are Age-Structured Models Appropriate for Catch-Effort Data?" *Can. J. Fish. Aquat. Sci.* 42 (6). NRC Research Press: 1066–72.

Pella, J.J., and P.K. Tomlinson. 1969. *A Generalized Stock Production Model*. Inter-American Tropical Tuna Commission.