biodyn

Laurence Kell

August 13th, 2014

```
library(biodyn)
library(ggplot2)
library(plyr)
library(diags)
library(kobe)
library(ggplotFL)
```

Introduction

BIOMASS DYNAMIC stock assessment models have been criticised as being too simplistic to capture the actual population dynamics. However, if a simple model can provide advice on stock status relative to reference points and predict the response of a stock to management why use anything more complicated?

The biodyn package is intended to help answer this question by being simulation tested using Managment Strategy Evaluation (MSE) using FLR.

The Class

The biodyn class includes methods for fitting, examining goodness of fit diagnostics, estimating uncertainly in stock status relative to reference points, deriving quantities used in management, running projections, simulating Harvest Control Rules (HCRs), conducting Management Strategy Evaluation (MSE) and for plotting.

The class includes data (i.e. slots) for catch, parameter estimates, fitted stock biomass and residuals from the fits of the CPUE used as proxies for stock trends.

Stock assessment

Russell [1] summarised the key processes influencing the dynamics of exploited populations in a single equation, where the biomass B_t this year is a function of the biomass last year (B_{t-1}) plus gains due to growth (G) and recruitment (R) and losses due to fishing (F) and natural mortality (M).

Recognising that there may be a mismatch between the stock assumptions and the population the equation can be expanded to include gains due to immigration (I) and losses due to emigration (E).

[1]

$$B_t = B_{t-1} + (G + R) - (F + M) \quad (1)$$

Figure 1: The Russell equation

$$f(B_2) = B_1 + (G + R + I) - (F + M + H) \quad (2)$$

Figure 2: Russell equation with migration

In a biomass dynamic stock assessment production function the dynamics of recruitment, growth and natural mortality are simplified into a single production function P which can be modelled by a variety of surplus production functions such as that of Pella-Tomlinson **?**.

The dynamics i.e. productivity and reference points are determined by r and the shape of the production function p. if $p = 1$ then MSY is found halfway between 0 and K; as p increases MSY shifts to the right.

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

$$B_{t+1} = B_t - C_t + P_t \qquad (3)$$

Figure 3: Biomass dynamic

$$\frac{r}{p} \cdot B(1 - (\frac{B}{K})^p) \qquad (4)$$

Figure 4: An equation

Creating an object

First a biodyn object has to be created. There are various ways of creating a new object, the first way is to use the class creator

```
bd =biodyn()
```

Supplying the catch helps to set the dimensions

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

Perhaps the easist way is to create an new object from an existing one, i.e. coercion from an FLStock

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

or aspic

```
library(aspic)
asp=aspic("http://http://rscloud.iccat.int/kobe/swon/2013/aspic/run2/")
bd =as(asp,"biodyn")
```

Simulated objects can also be created

```
bd=simBiodyn()
```

Plotting

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.

Time Series

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

Production Function

```
library(reshape)
x=simBiodyn()
plotPrd(x)+
  geom_path( aes(stock,catch),
           model.frame(FLQuants(x,"stock","catch")))+
  geom_point(aes(stock,catch),
           model.frame(FLQuants(x,"stock","catch")))
```

Diagnostics

See below

Advice

See below

Comparisons with other classes

```
plotMSE()
```

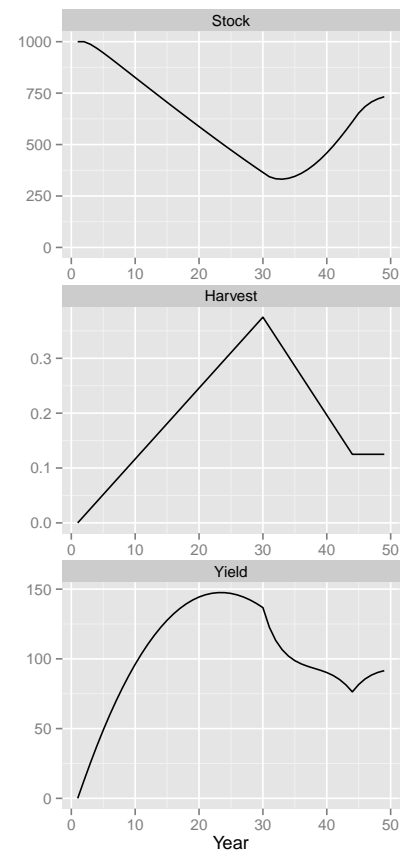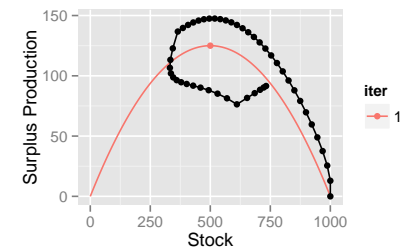

Figure 5: Simulated time series

Figure 6: Simulated CPUE series

Monte Carlo Simulation

```
harvest=rlnorm(200,log(harvest(bd)[,-1]),.2)

bd=fwd(bd,harvest=harvest)

plot(bd,worm=3)+
  theme(legend.position="bottom")
```
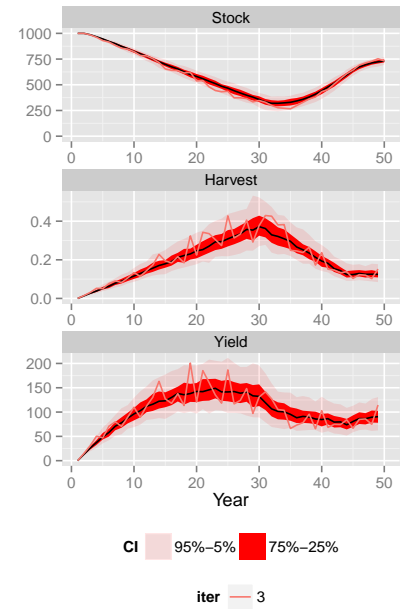

Figure 7: Monte Carlo simuation time series with confidence intervals and a worm (single simulation).

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=simBiodyn()
```

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
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="salm
```
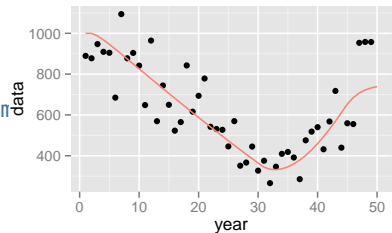

Figure 8: Simulated CPUE series

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 assumed 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),msy=mean(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
  1.03137    0.19175
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  1.0314e-01  1.0314e+00
  sigma1   1.0000e+00  1.9175e-02  1.9175e-01
        option
params  max
  r        5.0000e+00
  k        1.0000e+04
  p        1.0000e+01
  b0       1.0000e+01
  q1       1.0314e+01
  sigma1   1.9175e+00
units:  NA
```

Maximum Likelihood

Estimation can be performed using maximum likelihood

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

[1] TRUE

# plot(biodyns("True"=bd,"Hat"=bdHat))+
#   theme(legend.position="bottom")
```

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

```
params(bdHat)

An object of class "FLPar"
params
```

```
        r          k          p          b0
  0.57526 883.04136    1.00000    1.00000
       q1     sigma1
   1.20040    0.18835
units:  NA
```

params(bdHat)/**params**(bd)

```
An object of class "FLPar"
params
       r         k         p        b0        q1
1.15052 0.88304 1.00000 1.00000 1.16389
 sigma1
0.98226
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)
```

```
  year stock catch  index     hat stockHat
1    1 883.0  0.00  934.9 1060.0    883.0
2    2 876.6 12.93 1142.9 1052.2    876.6
3    3 861.0 25.53 1068.1 1033.6    861.0
4    4 841.8 37.55  881.5 1010.5    841.8
5    5 821.1 48.93  809.4  985.6    821.1
6    6 799.9 59.65 1090.5  960.2    799.9
  residual residualLag      qqx      qqy
1 -0.12562     0.08268 -0.36611 -0.12562
2  0.08268     0.03289  0.42139  0.08268
3  0.03289    -0.13653  0.05118  0.03289
4 -0.13653    -0.19701 -0.42139 -0.13653
5 -0.19701     0.12729 -1.11394 -0.19701
6  0.12729     0.32755  0.65852  0.12729
      qqHat harvest
1 -0.0858692 0.00000
2  0.0779623 0.01464
3  0.0009429 0.02934
4 -0.0973707 0.04408
5 -0.2414483 0.05883
6  0.1272947 0.07359
```

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=
  theme_ms(14,legend.position="bottom")
```


Figure 9: 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.

```
ggplot(with(rsdl, data.frame(obs=stdz(index),hat=stdz(hat)))) +
    geom_abline(aes(0,1))                                       +
    geom_point( aes(obs,hat))                                   +
    stat_smooth(aes(obs,hat),method="lm", se=F)                 +
    theme_ms(14,legend.position="bottom")                       +
    xlab("Fitted") + ylab("Observed")
```


Figure 10: 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_ms(14,legend.position="bottom")
```
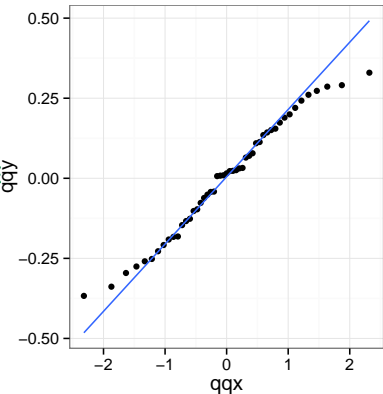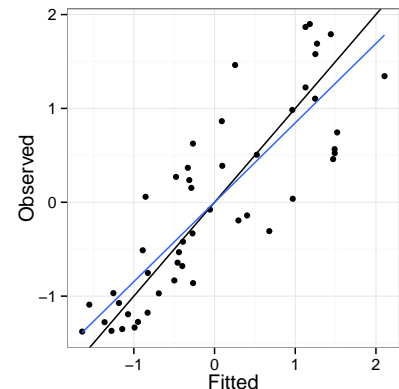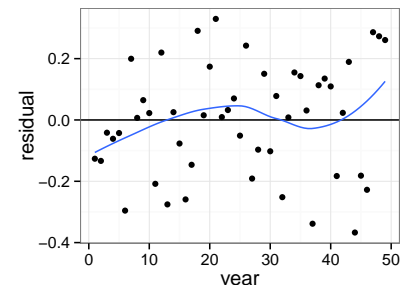

Figure 11: Residuals by year, with lowess smoother

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_ms(14,legend.position="bottom")
```
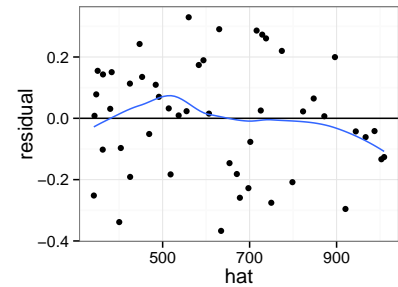
Autocorrelation

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_ms(14,legend.position="bottom")
```
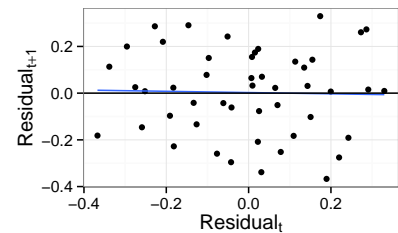

Figure 12: Plot of residuals against fitted value, to check variance relationship.

Figure 13: 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)
```

```
[1] TRUE
```

```
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.u1<0))+geom_line(aes(r,ll.u1))
```

2D

```
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.u1))+
  stat_contour(aes(colour = ..level..), size = 1)
```

likelihood components

```
bd=simBiodyn()
```

```
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))
```

```
bds=bd
```

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

```
bds@control[3:4,"phase"]=-1
bds=fit(bds,index=Us)
```
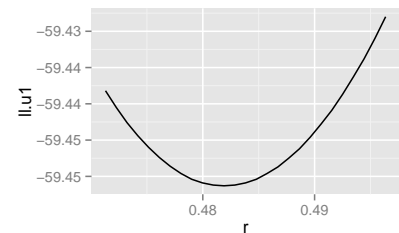
```
[1] TRUE
```


Figure 14: Likelihood profile for r

```
bds@control[,c("min")]=bds@params*0.1
bds@control[,c("val")]=bds@params
bds@control[,c("max")]=bds@params*10

fn=function(x) cbind(model.frame(params(x)["r"]),
                     ll=model.frame(x@ll)[,-3],
                     ll=apply(x@ll,2,sum))

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

ggplot(subset(melt(prfl[,c("r","ll.u1","ll.u2","1")],id="r"),
              value<10))+
  geom_line(aes(r,value,group=variable,col=variable))+
  facet_wrap(~variable,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.

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

[1] TRUE

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,
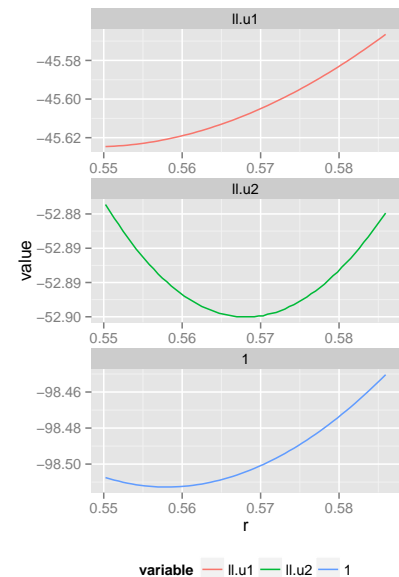

Figure 15: Likelihood profile by data conmponent, i.e. CPUE series

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.

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

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)

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


Figure 16: Bootstrapped CPUE series

Jack knife

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

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

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

```
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)
```
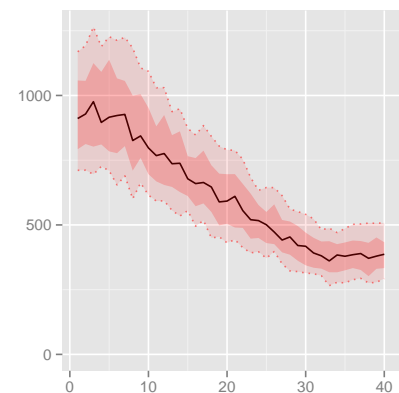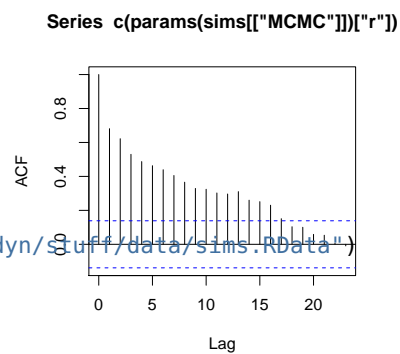
MCMC

Monte Carlo Markov Chain

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

```
[1] TRUE
```

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

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

Series c(params(sims[["MCMC"]])["r"])


```
save(sims,cpue,bdJK,bdHat,file="/home/laurie/Desktop/flr/git/biodyn/stuff/data/sims.RData")
```

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_density(aes(x=bbmsy,y = ..count..))
```

Marginal Density for Stock/BMSY

```
source('~/Desktop/flr/git/biodyn/R/biodyn-kobe.R')
df=kobe(sims[-c(1:2)])
ggplot(subset(df,year==39))+
  geom_density(aes(x=stock, y=..count..), position = "stack",fill="red")+
  facet_wrap(~.id,scale="free_y",ncol=1)+
  geom_vline(aes(xintercept=c(stock(sims[["Best Fit"]])[,"38"]%/%bmsy(sims[["Best Fit"]]))))

  #scale_x_continuous(limits=c(0.5,1.5))
```

Marginal Density for Harvest/FMSY

```
ggplot(subset(df,year==39))+
  geom_density(aes(x=harvest, y=..count..), position = "stack",fill="red")+
  facet_wrap(~.id,scale="free_y",ncol=1)+
  scale_x_continuous(limits=c(0,2.0))+
  geom_vline(aes(xintercept=c(harvest(sims[["Best Fit"]])[,"38"]%/%fmsy(sims[["Best Fit"]]))))
```

Kobe Phase Plot

```
library(kobe)
kobePhase()+
  geom_point(aes(stock,harvest),data=subset(df,year==39))+
  facet_wrap(~.id,ncol=2)
```
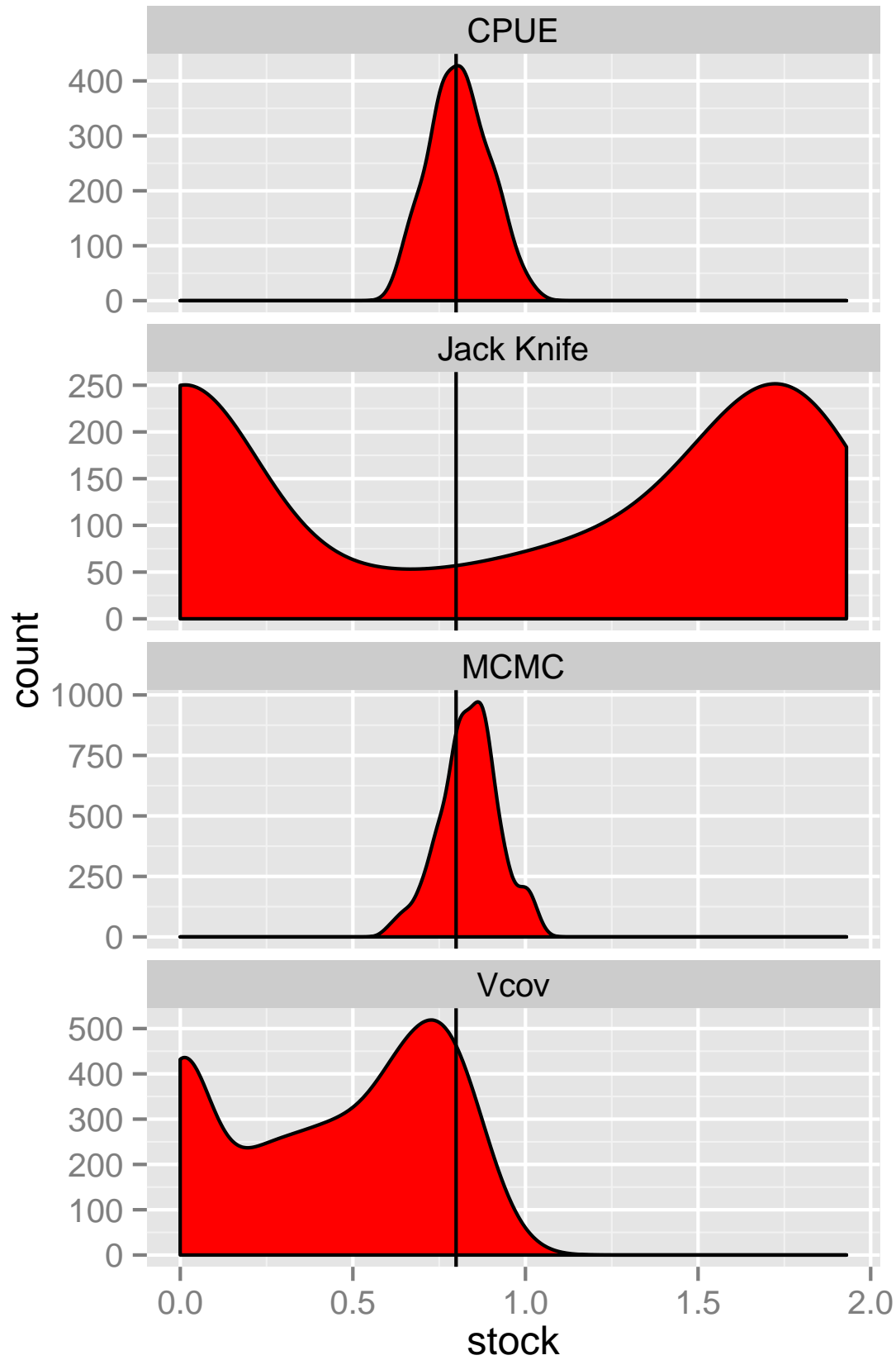
```
[1] "yellow" "green"  "red"
```

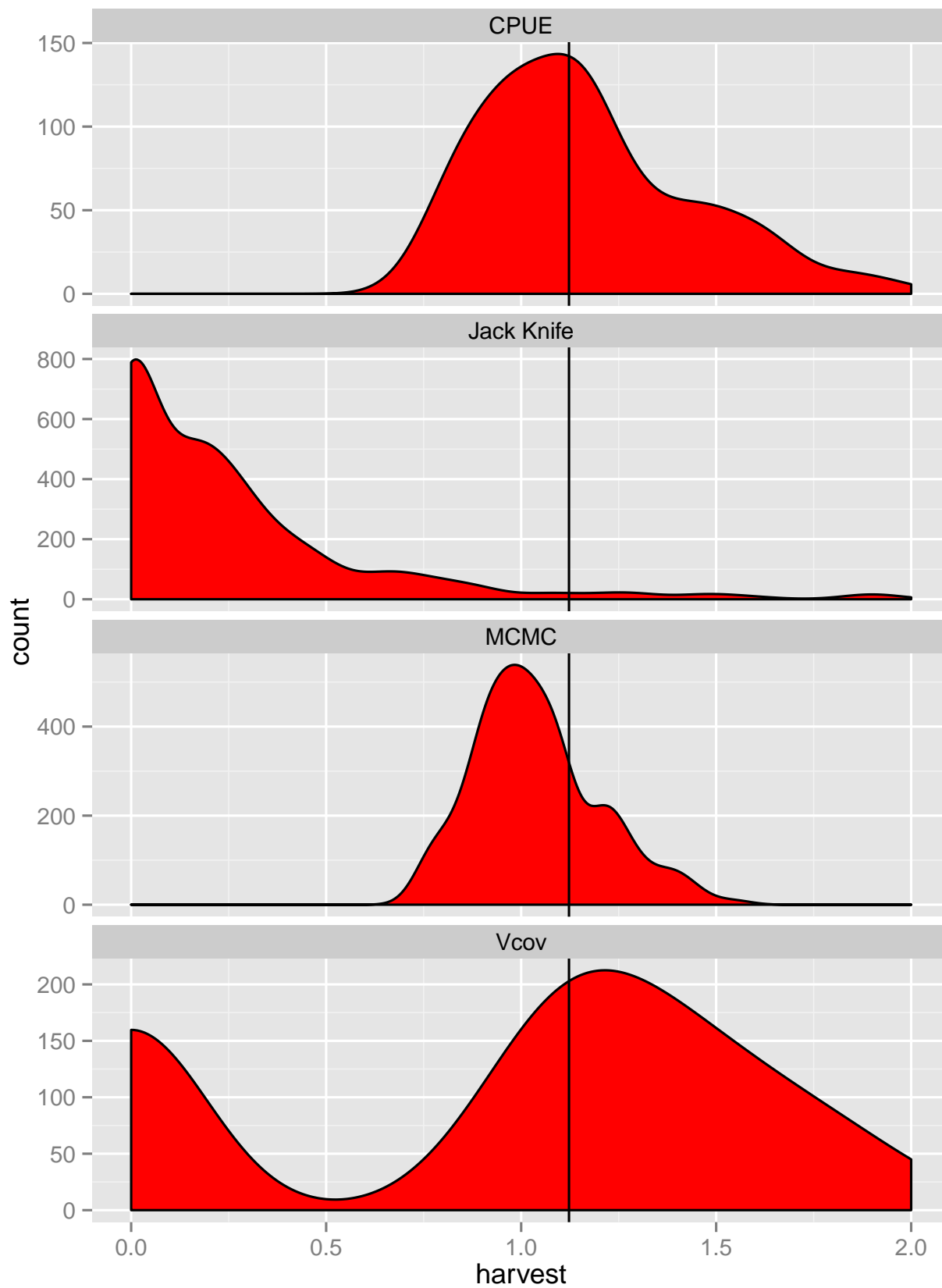Figure 17: Densities of Stock/BMSY from different methods for estimating uncertainty.

Figure 18: Densities of Harvest/FMSY from different methods for estimating uncertainty.

Projections

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

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


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

Harvest Control Rules

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

Annual

```
bd    =simBiodyn()


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")
```


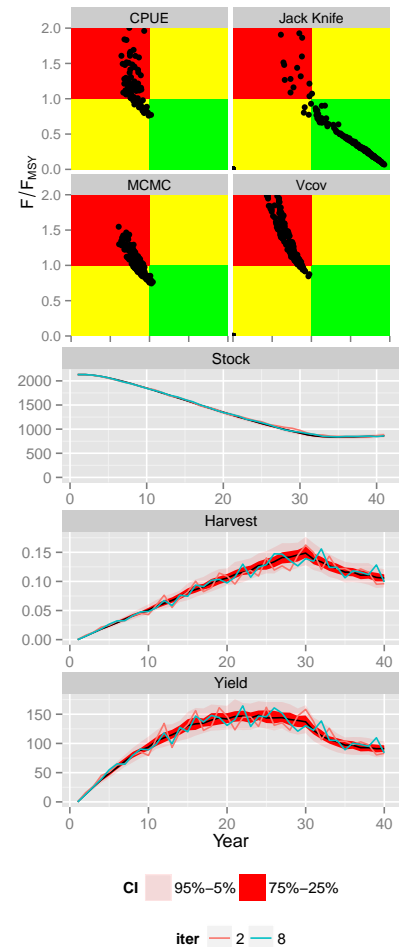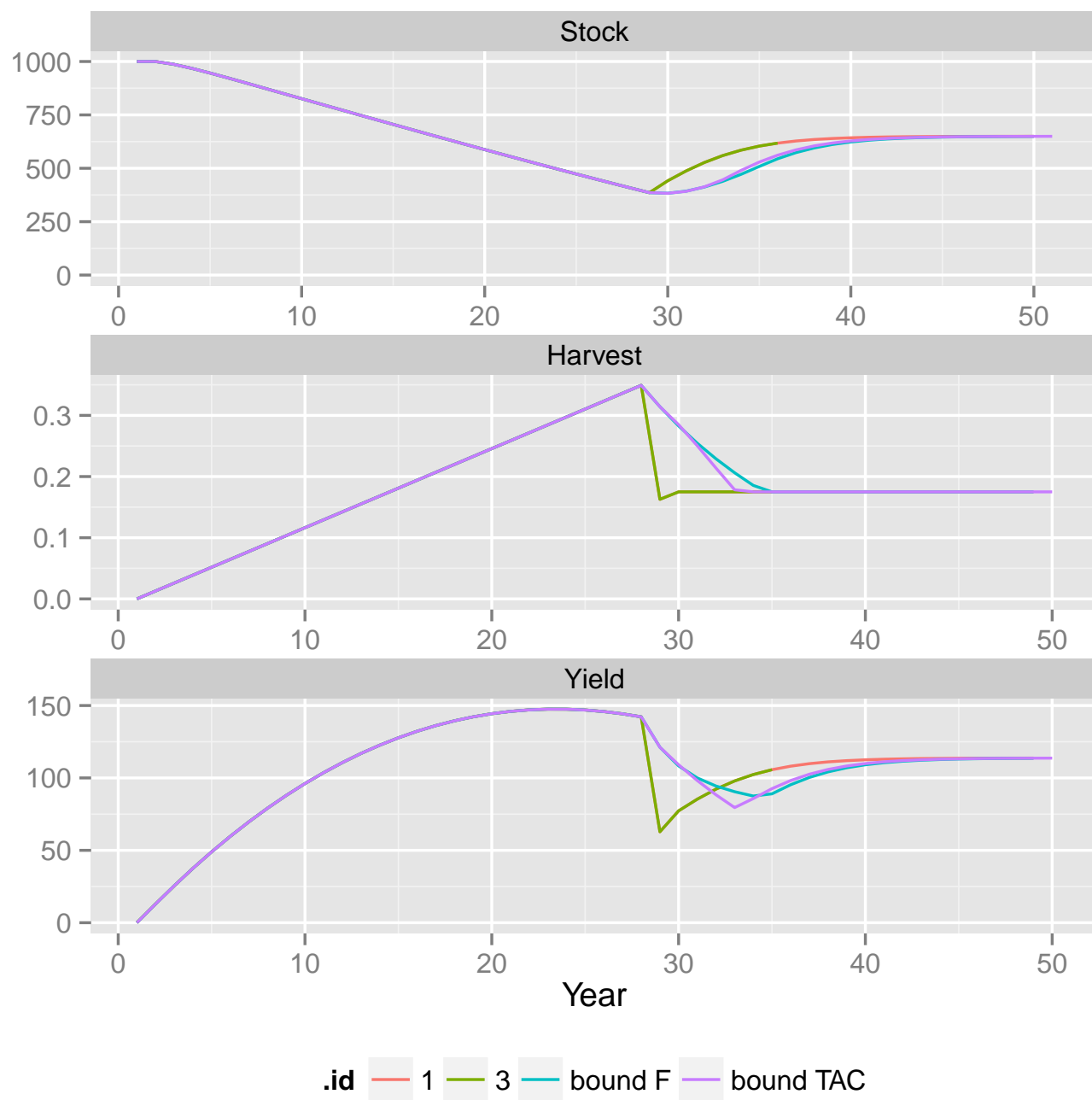Figure 19: Projection

Figure 20: Plots of projections

Stochasticity

Process Error and Harvest Control Rule

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

bd=window(simBiodyn(),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))$ta
         pe   =pe)

plot(bd)

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)

[1] "yellow" "green"  "red"
```
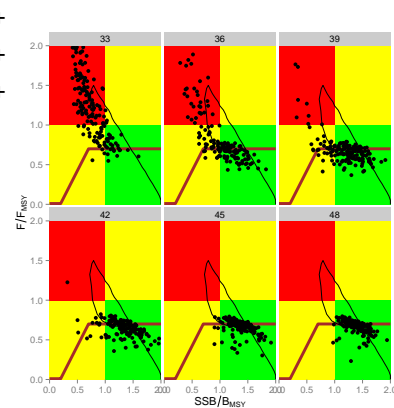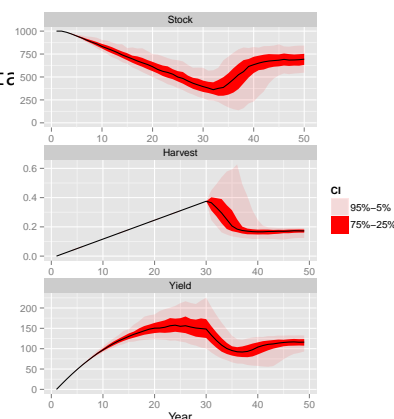
MSE

`biodyn:::mseBiodyn`

Sidenotes

One of the most prominent and distinctive features of this style is the extensive use of sidenotes. There is a wide margin to provide ample room for sidenotes and small figures. Any use of a footnote will automatically be converted to a sidenote. ²

[2] This is a sidenote that was entered using a footnote.

If you'd like to place ancillary information in the margin without the sidenote mark (the superscript number), you can use the `\marginnote` command.

This is a margin note. Notice that there isn't a number preceding the note.

Note also that the two footnote references (`tufte_latex` and `books_be`, both defined below) were also included in the margin on the first page of this document.

Tables

You can use the **xtable** package to format LaTeX tables that integrate well with the rest of the Tufte handout style. Note that it's important to set the `xtable.comment` and `xtable.booktabs` options as shown below to ensure the table is formatted correctly for inclusion in the document.

```
library(xtable)
options(xtable.comment = FALSE)
options(xtable.booktabs = TRUE)
xtable(head(mtcars[,1:6]), caption = "First rows of mtcars")
```

	mpg	cyl	disp	hp	drat	wt
Mazda RX4	21.00	6.00	160.00	110.00	3.90	2.62
Mazda RX4 Wag	21.00	6.00	160.00	110.00	3.90	2.88
Datsun 710	22.80	4.00	108.00	93.00	3.85	2.32
Hornet 4 Drive	21.40	6.00	258.00	110.00	3.08	3.21
Hornet Sportabout	18.70	8.00	360.00	175.00	3.15	3.44
Valiant	18.10	6.00	225.00	105.00	2.76	3.46

Table 1: First rows of mtcars