biodyn

Laurence Kell

August 13th, 2014

```r
library(biodyn)
```

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?

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).

[1]

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

Figure 1: The Russell equation

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).

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

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 **?**].

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

Figure 3: Biomass dynamic

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.

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

Figure 4: An equation

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.

The Class

The package includes methods for fitting, examining goodness of fit diagnostics, estimating uncertainly in stock status relative to reference points, running projections and Harvest Control Rules (HCRs) and conducting Management Strategy Evaluation (MSE).

biodyn has slots for the catch, parameter estimates, fitted stock biomass and residuals from the fits of the CPUE used as proxies for stock biomass. There are a variety of methods for deriving quantities used in management such as reference points and for plotting. First an object of class biodyn has to be created

Creating an object

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 is 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")
```

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 then for these to be modified and new plots developed as required.

Time Series

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

plot(x)
```

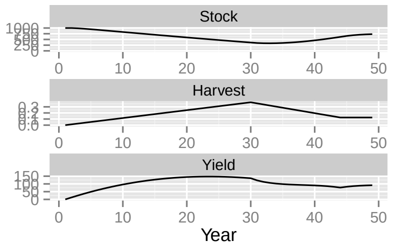


Figure 5: Simulated CPUE series

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")))
```

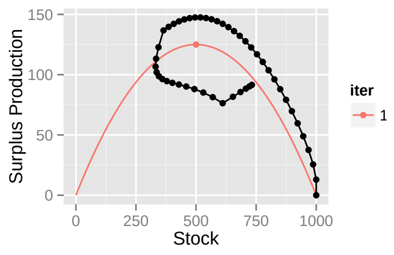


Figure 6: Simulated CPUE series

Diagnostics

See below ## Advice See below

Comparisons with other classes

plotMSE

Estimation

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

It is alway good to check estimated values with the true ones. Theredore we simulate a stock with know parameters and exploitation history

```
bd = simBiodyn()
```

We also need a CPUE series so we simulate that by taking the mid year biomass and adding an error term.

```
cpue = (stock(bd)[, -dims(bd)$year] + stock(bd)[,
    -1])/2
cpue = rlnorm(1, log(cpue), 0.2)
```

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

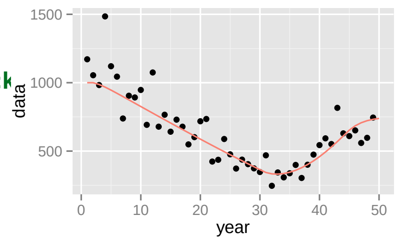


Figure 7: Simulated CPUE series

Starting values for parameters are also required. These can be set by informed guesses or last years estimates (if you can replicate them). 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. The shape by default is assummed to be symetric (i.e. p=1) and B_0 (the ratio of the initial biomass to carrying capacity) could 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 estimates the stock based on the initial parameters and catch and so catchability and the CV of the fit of the CPUE index can also be guessed.

```
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
   1.02462    0.16265
units:  NA
```

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

```
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 1.0246e-01 1.0246e+00
  sigma1 1.0000e+00 1.6265e-02 1.6265e-01
        option
params   max
  r      5.0000e+00
  k      1.0000e+04
  p      1.0000e+01
  b0     1.0000e+01
  q1     1.0246e+01
  sigma1 1.6265e+00
units:  NA
```

Estimation

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')
save(bdHat, cpue, file = "/home/laurie/Desktop/bdHat.RData")
```

Since we know the true parameters then we can check why the fit isnt perfect.

```
params(bdHat)
```

```
An object of class "FLPar"
params
        r         k         p        b0
  0.50624 987.62679   1.00000   1.00000
       q1    sigma1
  1.05231   0.16181
units:  NA
```

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

```
An object of class "FLPar"
params
      r       k       p      b0      q1
1.01248 0.98763 1.00000 1.00000 1.02703
 sigma1
0.99485
units:  NA
```

Residual Patterns

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.

Although biomass dynamic models only use catch and effort data and estimate a few parameters, the same diagnstics are required when comparing fits from statistical catch-at-size models with potentially 1000s of parameters.

Patterns in residuals of the fits to the CPUE and stock abundnace may indicate a violation of models assumptions. Residuals and covariates are in the diags slot.

The residuals are found in the diags slot.

```
rsdl = bdHat@diags

head(rsdl)
```

```
  year stock catch  index    hat stockHat
1    1 987.6  0.00 1171.2 1039.3    987.6
2    2 981.2 12.93 1054.5 1032.5    981.2
3    3 965.2 25.53  982.9 1015.6    965.2
4    4 944.7 37.55 1484.6  994.1    944.7
5    5 922.2 48.93 1120.4  970.4    922.2
6    6 898.8 59.65 1044.2  945.8    898.8
  residual residualLag     qqx      qqy
1  0.11950     0.02113  0.7916  0.11950
2  0.02113    -0.03280  0.2586  0.02113
3 -0.03280     0.40104 -0.3661 -0.03280
4  0.40104     0.14370  2.3188  0.40104
5  0.14370     0.09897  0.9405  0.14370
6  0.09897    -0.22075  0.6585  0.09897
     qqHat harvest
1  0.11996 0.00000
2  0.03591 0.01309
3 -0.06262 0.02619
4  0.36080 0.03929
5  0.14345 0.05240
6  0.09897 0.06550
```

Violation of the assumptions 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.).

```
ggplot(rsdl) + geom_point(aes(qqx, qqy)) + stat_smooth(aes(qqx,
    qqHat), method = "lm", se = T, fill = "blue",
    alpha = 0.1) + theme_ms(14, legend.position = "bottom")
```

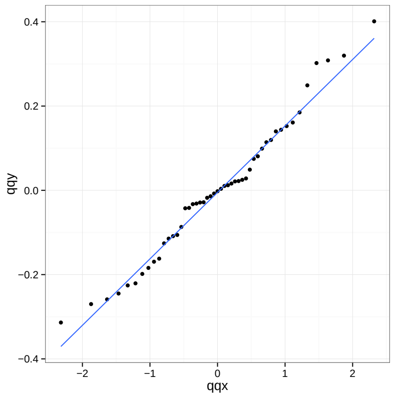


Figure 8: Quantile-quantile plot to compare residual distribution with the normal distribution.

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")
```

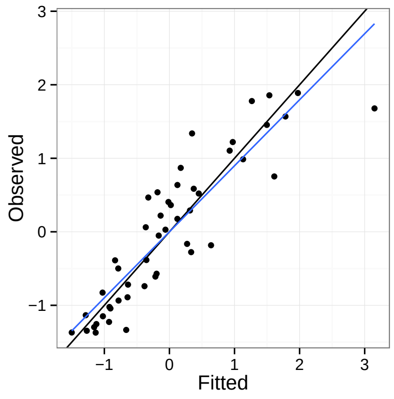


Figure 9: Observed CPUE verses fitted, blue line is a linear resgression fitted to points, black the y=x line.

Year Patterns

Next the residuals are plotted agianst 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(yintercep
    geom_point() + stat_smooth(method = "loess",
    se = F) + theme_ms(14, legend.position = "bottom")
```

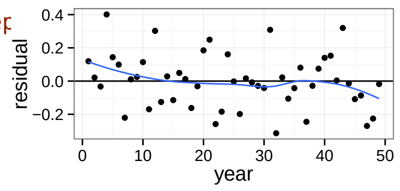


Figure 10: Residuals by year, with lowess smoother

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 =
    geom_point() + stat_smooth(method = "loess",
    se = F) + theme_ms(14, legend.position = "bottom")
```

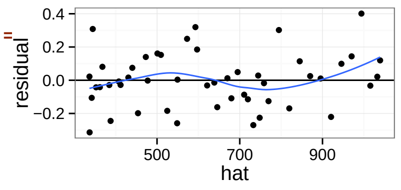


Figure 11: Plot of residuals against fitted value, to check variance relationship.

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.

```
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")
```

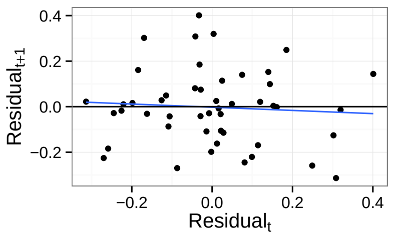


Figure 12: Plot of autocorrelation, i.e. $residual_{t+1}$ verses $residual_t$.

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(aes(year, index), data = rsdl) + geom_point() +
    stat_smooth(method = "loess", se = F) + geom_line(aes(year,
    hat), col = "black", data = rsdl) + scale_x_continuous(breaks
    3000, 10)) + theme_ms(14, legend.position = "bottom")
```

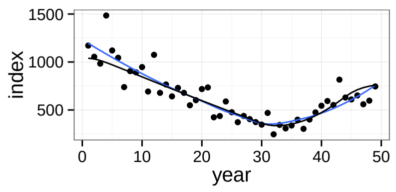


Figure 13: Plot predicted stock trend by index

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

```
res = profile(bdHat, which = "r", fixed = c("b0",
    "p"), cpue, range = c(1.2, 3))
ggplot(res) + geom_line(aes(r, ll))
```

1D

```
ggplot(res, aes(r, k, z = ll))
```

2D

```
ggplot(res, aes(r, k, z = ll)) + stat_contour(aes(colour = ..level..),
    size = 1)
```

likelihood components

```
ggplot(res, aes(r, k, z = ll)) + stat_contour(aes(colour = ..level..),
    size = 1)
```

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.

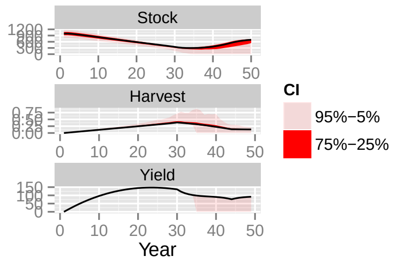
```
sims = biodyns(‘Best Fit‘ = bd)
```

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.

```
save(bdHat, cpue, file = "/home/laurie/Desktop/bdHat.RData")
sims[["Vcov"]] = mvn(bdHat, 500, nms = c("r",
    "k"), fwd = TRUE)

plot(sims[["Vcov"]])


save(sims, file = "/home/laurie/Desktop/sims.RData")
```



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(diags['residuals'])

cpueSim = rlnorm(100, log(cpueSim), 0.25)

cpueSim[cpueSim == 0] = NA

plot(sims[["CPUE"]])

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

Jack knife

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

```
sims[["Jack Knife"]] = fit(bdHat, jackknife(cpue))
```

The results from the fit can then be used to estimate uncertainty

```
plotJack(sims[["Jack Knife"]], bdHat)
```

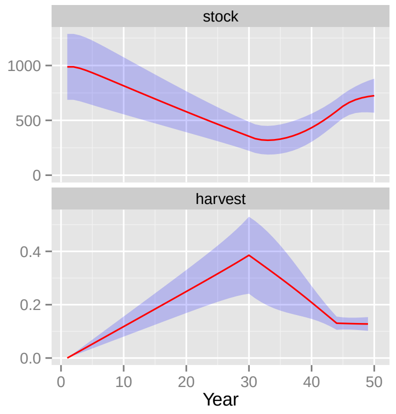


Figure 14: Plot predicted stock trend by index

MCMC

Monte Carlo Markov Chain

```
sims[["MCMC"]] = fit(bdHat, cpue, cmdOps = c("-mcmc 100000, -mcsav
```

```
[1] TRUE
```

```
plot(sims[["MCMC"]])
```

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

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

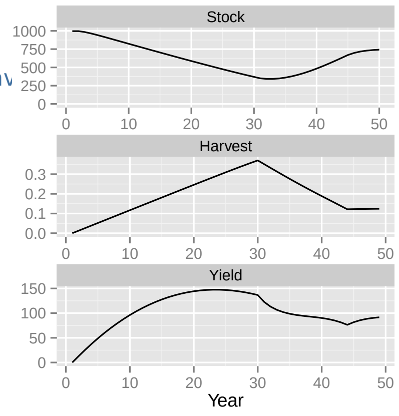
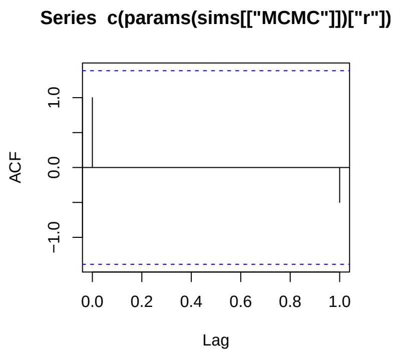


Figure 15: Plot predicted stock trend by index



Comparison

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 $F : F_{MSY}$. These can be found in the mng slot.

```
head(bdHat@mng)
```

```
An object of class "FLPar"
       var
param   hat        sd
  _r      0.506240   0.070767
  _k    987.630000 119.480000
  _q[1]   1.052300   0.135120
  _s[1]   0.161810   0.016345
  r       0.506240   0.070767
  k     987.630000 119.480000
units:  NA
```

The variance matrix is in the mngVcov slot

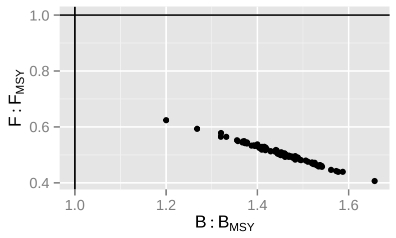
```
bdHat@mngVcov
```

These can be used to simulate joint distributions

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

mvrnorm(10, currentState, currentStateVar)
```

```
      bbmsy  ffmsy
 [1,] 1.578 0.4426
 [2,] 1.561 0.4524
 [3,] 1.494 0.4826
 [4,] 1.486 0.4879
 [5,] 1.383 0.5301
 [6,] 1.397 0.5305
 [7,] 1.453 0.5003
 [8,] 1.488 0.4853
 [9,] 1.436 0.5156
[10,] 1.538 0.4601
```

```
ggplot(data = as.data.frame(mvrnorm(100, currentState,
    currentStateVar)), aes(bbmsy, ffmsy)) + geom_point() +
    geom_hline(aes(yintercept = 1)) + geom_vline(aes(xintercept =
    xlab(expression(B:B[MSY])) + ylab(expression(F:F[MSY]))
```



Advice Plots

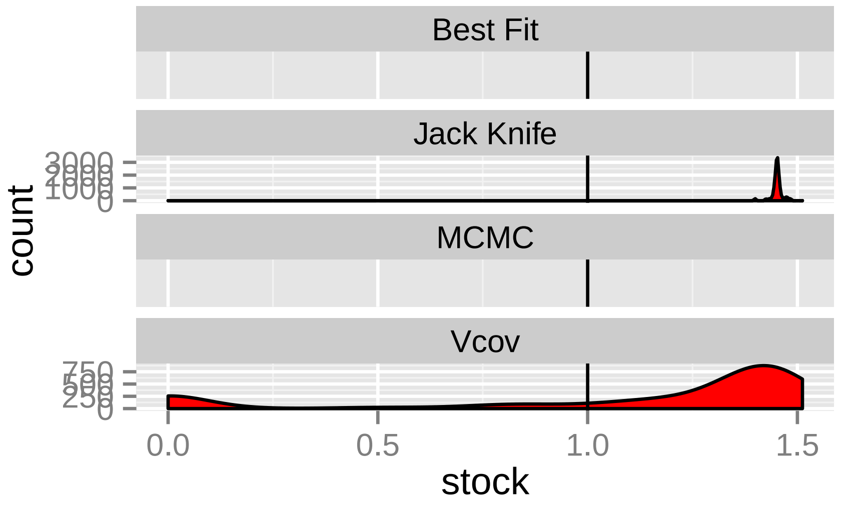
```
library(kobe)
setGeneric("kobe", function(object, method, ...) standardGeneric("kobe"))
```

```
[1] "kobe"
```

```
source("~/Desktop/flr/pkgs/biodyn/R/biodyn-kobe.R")
```

Marginal Density for Stock/BMSY

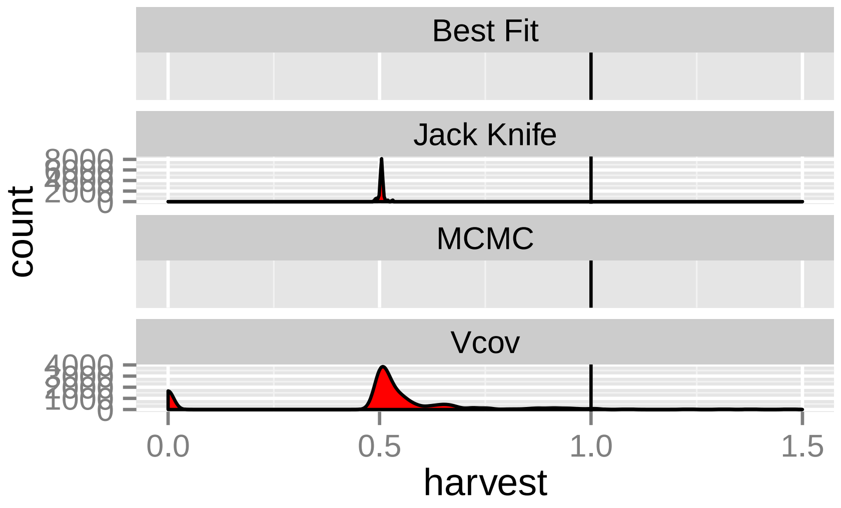
```
df = kobe(sims)
ggplot(subset(df, year == 49)) + geom_density(aes(x = stock,
    y = ..count..), position = "stack", fill = "red") +
    geom_vline(aes(xintercept = 1)) + facet_wrap(~.id,
    scale = "free_y", ncol = 1)
```



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

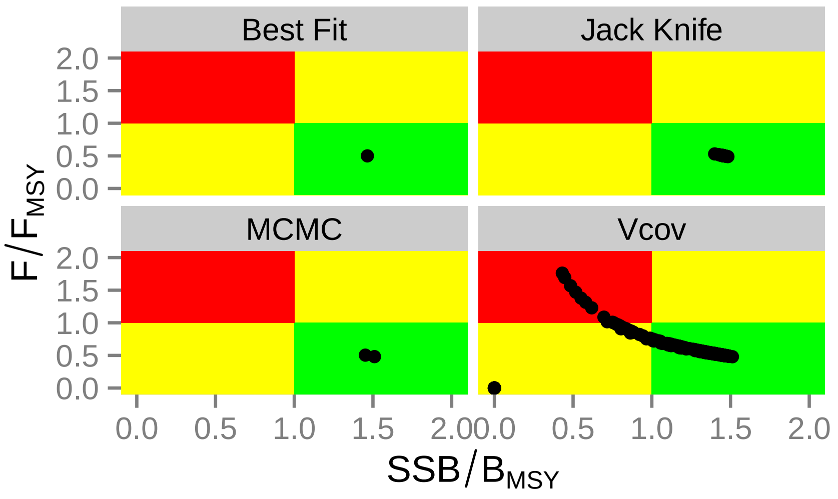
Marginal Density for Harvest/FMSY

```
ggplot(subset(df, year == 49)) + geom_density(aes(x = harvest,
    y = ..count..), position = "stack", fill = "red") +
    geom_vline(aes(xintercept = 1)) + facet_wrap(~.id,
    scale = "free_y", ncol = 1) + scale_x_continuous(limits = c(0,
    1.5))
```



Kobe Phase Plot

```
library(kobe)
kobePhase() + geom_point(aes(stock, harvest),
    data = subset(df, year == 49)) + facet_wrap(~.id,
    ncol = 2)
```



Projections

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

```
harvest = rlnorm(1, log(harvest(bdHat))[, -dims(bdHat)$year],
    0.1)
```

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

plot(bdHat)
```

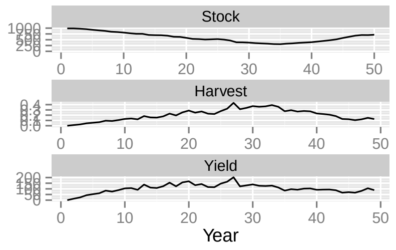


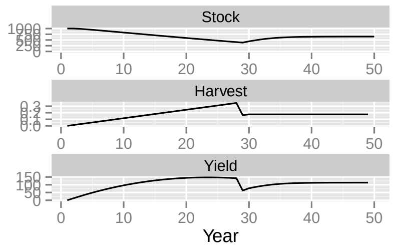
Figure 16:

Harvest Control Rules

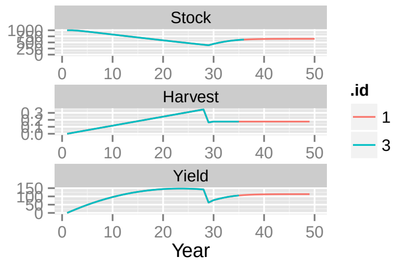
IN HIS LATER BOOKS[2] use the \newthought
Use simulated data

2 http://www.edwardtufte.com/tufte/books_be

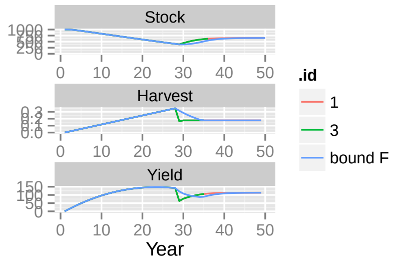
```
bd = simBiodyn()

## simulate HCRs, annual, tri-annula, F bound,
## TAC bound
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)

plot(bd)
```



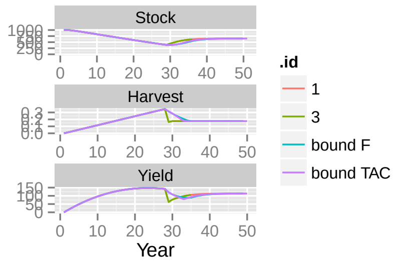
```
bd = window(bd, end = 29)
for (i in seq(29, 49, 3)) bd = fwd(bd, harvest = hcr(bd,
    refYrs = i, yrs = i + 1:3)$hvt)
simHCR[["3"]] = bd
save(simHCR, file = "/home/laurie/Desktop/sims.RData")
plot(simHCR)
```



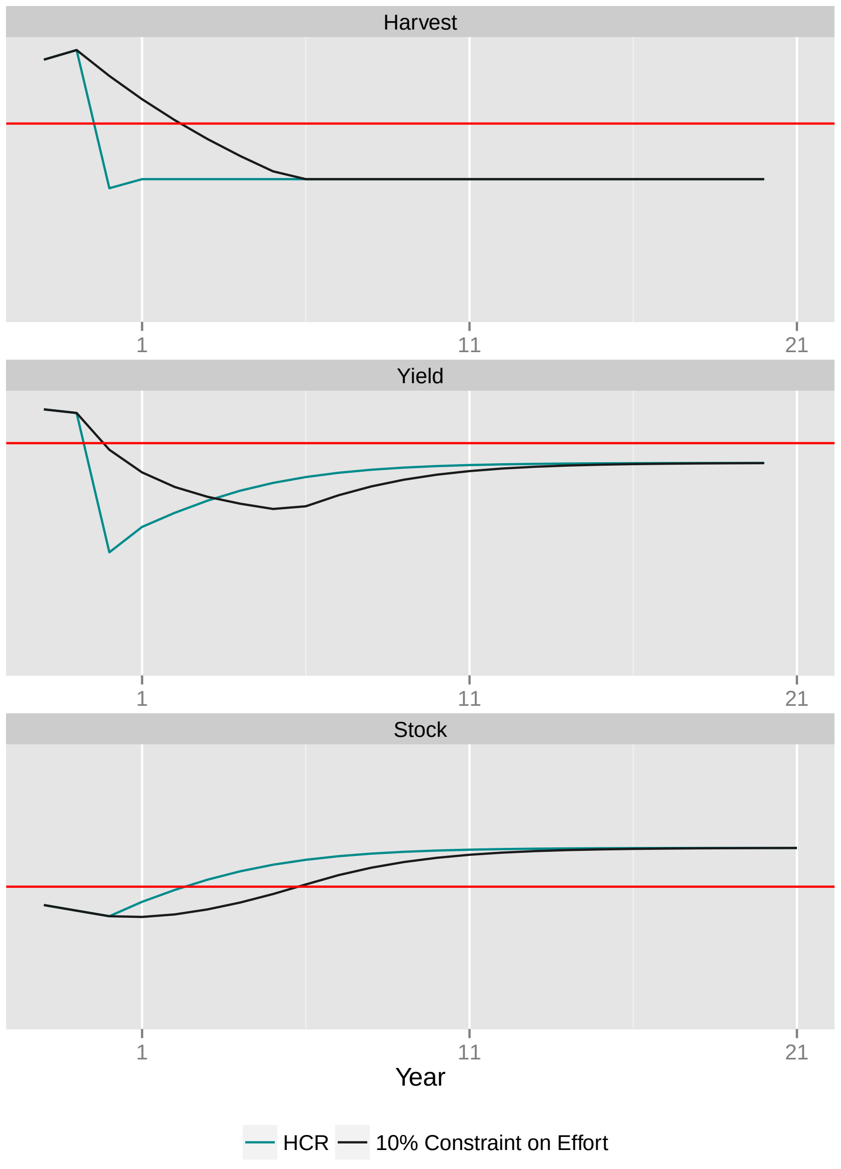
```
bd = window(bd, end = 29)
for (i in seq(29, 49, 1)) bd = fwd(bd, harvest = hcr(bd,
    refYrs = i, yrs = i + 1, bndF = c(0.9, 1.1))$hvt)
simHCR[["bound F"]] = bd

plot(simHCR)
```



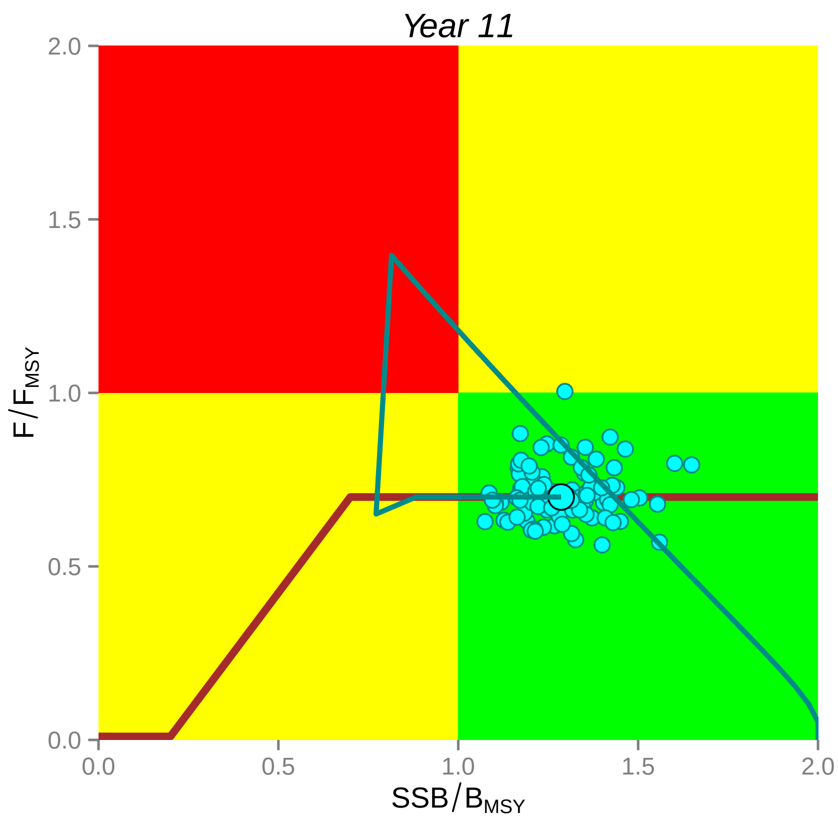
```
bd = window(bd, end = 30)
for (i in seq(29, 49, 1)) bd = fwd(bd, catch = hcr(bd,
    refYrs = i, yrs = i + 1, tac = T, bndTac = c(0.9,
        1.1))$tac)
simHCR[["bound TAC"]] = bd

plot(simHCR)
```



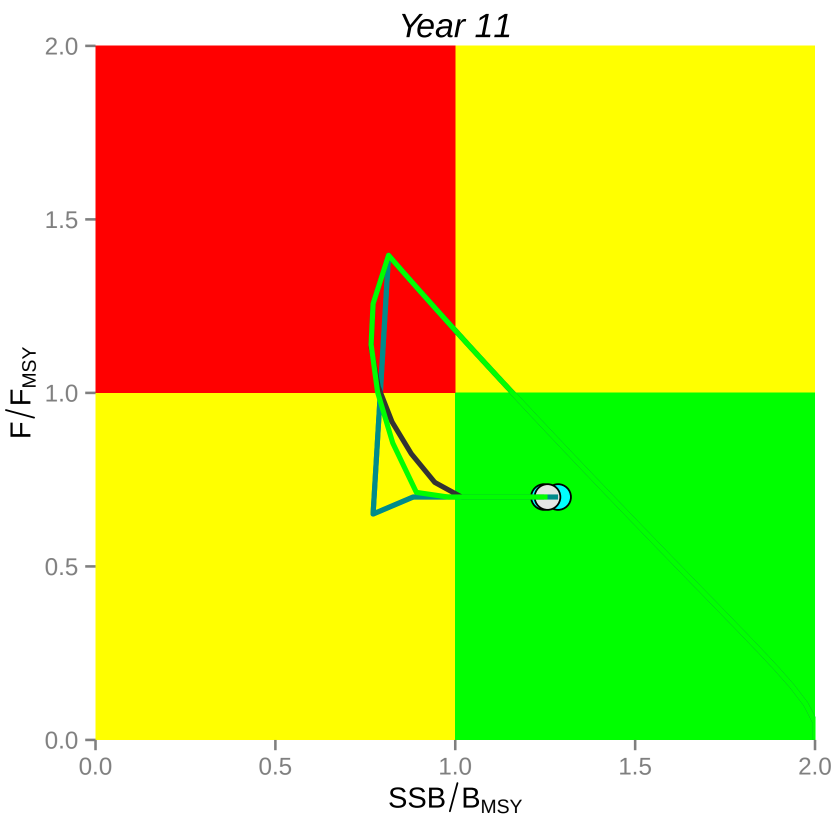
```
p. = plot(simHCR[c(1, 3)]) + theme(legend.position = "bottom") +
    scale_colour_manual(values = c("cyan4", "grey10"),
        labels = c("HCR", "10% Constraint on Effort")) +
    guides(col = guide_legend(title = NULL)) +
    scale_x_continuous(limits = c(27, 50), breaks = c(30,
        40, 50), labels = c(1, 11, 21)) + scale_y_continuous(breaks = NULL)

p.$data = transform(p.$data, qname = factor(qname,
    levels = c("Harvest", "Yield", "Stock")))
p. + geom_hline(aes(yintercept = X1), data = cbind(qname = c("Yield",
    "Harvest", "Stock"), data.frame(refpts(bd))),
    col = "red")
```



```
kb = ldply(simHCR[c(1)], function(x, sd = 0.1) model.frame(mcf(FLQuants(stock = rlnorm(100,
    log(stock(x)%/%bmsy(x)), sd), harvest = rlnorm(100,
    log(harvest(x)%/%fmsy(x)), sd)))))
kb = subset(kb, year %in% 29:50)

pt = ldply(simHCR[c(1)], function(x) model.frame(mcf(FLQuants(stock = stock(x)%/%bmsy(x),
    harvest = harvest(x)%/%fmsy(x)))))
pt = subset(pt, year %in% 1:50)
pt. = ddply(pt, .(year, .id), with, data.frame(stock = median(stock),
    harvest = median(harvest)))

i = 40
print(kobePhase(subset(kb, year %in% i)) + geom_line(aes(stock,
    harvest), data = biodyn:::hcrPlot(bd), col = "brown",
    size = 1.5) + ggtitle(paste("Year", i - 29)) +
    theme(legend.position = "none", plot.background = element_rect(fill = "transparent",
        colour = NA), plot.title = element_text(lineheight = 0.8,
        face = "italic")) + xlim(0, 2) + ylim(0,
    2) + geom_point(aes(stock, harvest, col = .id,
    fill = .id), shape = 21, size = 3) + geom_point(aes(stock,
    harvest, group = .id, fill = .id), col = "black",
    data = subset(pt., year == i), shape = 21,
    size = 5) + geom_path(aes(stock, harvest,
    col = .id, group = .id), data = subset(pt,
    year <= i), size = 1) + scale_fill_manual(values = c("cyan1",
    "green", "red", "yellow")) + scale_colour_manual(values = c("cyan4",
    "green", "red", "yellow")) + coord_cartesian(xlim = c(0,
    2), ylim = c(0, 2)))


library(FLCore)
library(plyr)
library(kobe)
library(ggplot2)

kb = ldply(simHCR[1:2], function(x, sd = 0.1) model.frame(mcf(FLQuants(stock = stock(x)%/%bmsy(x),
    harvest = harvest(x)%/%fmsy(x)))))
kb = subset(kb, year %in% 29:50)

pt = ldply(simHCR, function(x) model.frame(mcf(FLQuants(stock = stock(x)%/%bmsy(x),
    harvest = harvest(x)%/%fmsy(x)))))
pt = subset(pt, year %in% 1:50)
pt. = ddply(pt, .(year, .id), with, data.frame(stock = median(stock),
    harvest = median(harvest)))

i = 40
kobePhase(subset(kb, year %in% i)) + # geom_line(aes(stock,harvest),data=hcrPlot(bd),col='brown',size=1.5)
```



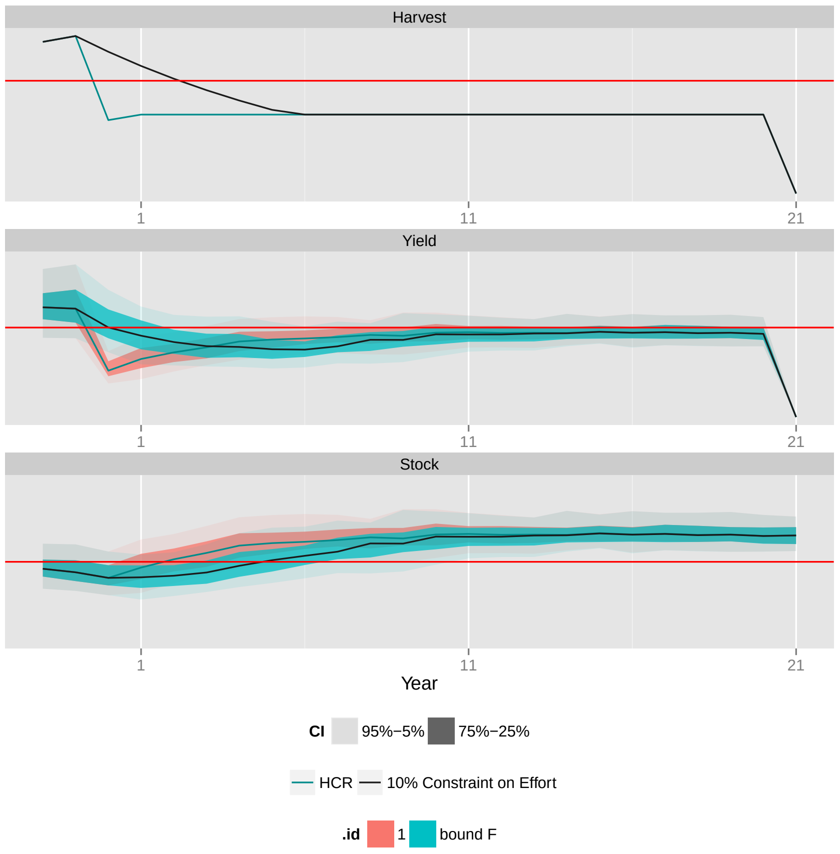
```
ggtitle(paste("Year", i - 29)) + theme(legend.position = "none",
    plot.background = element_rect(fill = "transparent",
        colour = NA), plot.title = element_text(lineheight = 0.8,
        face = "italic")) + xlim(0, 2) + ylim(0,
    2) + geom_point(aes(stock, harvest, col = .id,
    fill = .id, size = .id), shape = 21) + geom_point(aes(stock,
    harvest, fill = .id, group = .id), data = subset(pt.,
    year == i), shape = 21, col = "black", size = 5) +
    geom_path(aes(stock, harvest, col = .id, group = .id),
        data = subset(pt, year <= i), size = 1) +
    scale_fill_manual(values = c("cyan1", "cyan1",
        "grey90", "green", "red", "yellow")) +
    scale_size_manual(values = c(1, 3)) + scale_colour_manual(values = c("cyan4",
    "cyan4", "grey20", "green", "red", "yellow")) +
    coord_cartesian(xlim = c(0, 2), ylim = c(0,
        2))

pe = rlnorm(100, FLQuant(0, dimnames = list(year = 1:50)),
    0.5)
simHCR[[1]] = fwd(simHCR[[1]], harvest = harvest(simHCR[[1]])[,
    ac(1:50)], pe = pe)
```



```
simHCR[[3]] = fwd(simHCR[[3]], harvest = harvest(simHCR[[3]])[,
    ac(1:50)], pe = pe)
p. = plot(simHCR[c(1, 3)]) + theme(legend.position = "bottom") +
    scale_colour_manual(values = c("cyan4", "grey10"),
        labels = c("HCR", "10% Constraint on Effort")) +
    guides(col = guide_legend(title = NULL)) +
    scale_x_continuous(limits = c(27, 50), breaks = c(30,
        40, 50), labels = c(1, 11, 21)) + scale_y_continuous(breaks = NULL)

p.$data = transform(p.$data, qname = factor(qname,
    levels = c("Harvest", "Yield", "Stock")))
p. + geom_hline(aes(yintercept = X1), data = cbind(qname = c("Yield",
    "Harvest", "Stock"), data.frame(refpts(bd))),
    col = "red")
```

MSE

```
biodyn:::mseBiodyn
```


Harvest
Yield
Stock
1
11
21
Year
CI
95%−5%
75%−25%
HCR
10% Constraint on Effort
.id
1
bound F

Full Width Figures

You can arrange for figures to span across the entire page by using the `fig.fullwidth` chunk option.

```
qplot(wt, mpg, data = mtcars, colour = factor(cyl))
```

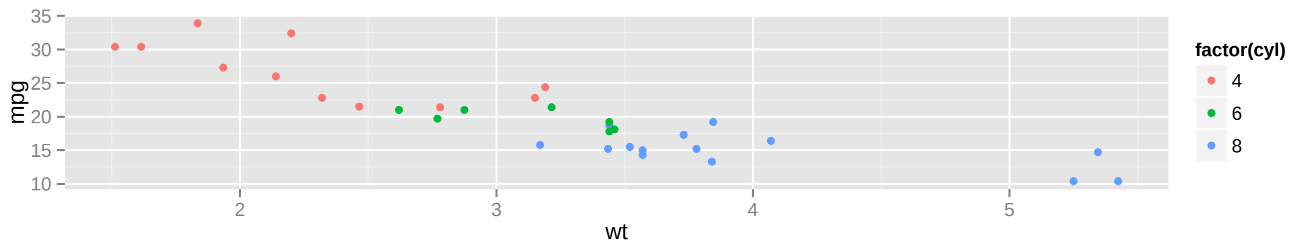


Figure 17: Full width figure

Note the use of the `fig.width` and `fig.height` chunk options to establish the proportions of the figure. Full width figures look much better if their height is minimized.

Main Column Figures

Besides margin and full width figures, you can of course also include figures constrained to the main column.

```
qplot(factor(cyl), mpg, data = mtcars, geom = "boxplot")
```

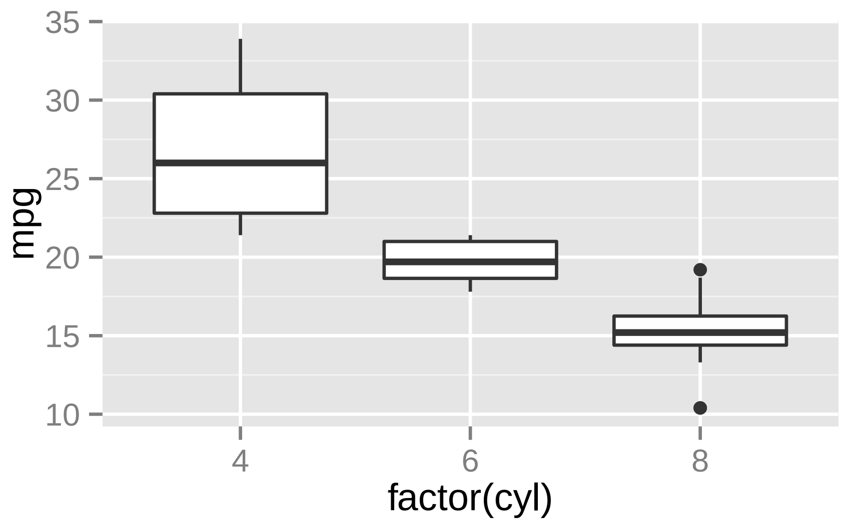


Figure 18: Another figure

Robustness

MSE

```
biodyn::mseBiodyn
```

Full Width Figures

You can arrange for figures to span across the entire page by using the `fig.fullwidth` chunk option.

```
qplot(wt, mpg, data = mtcars, colour = factor(cyl))
```

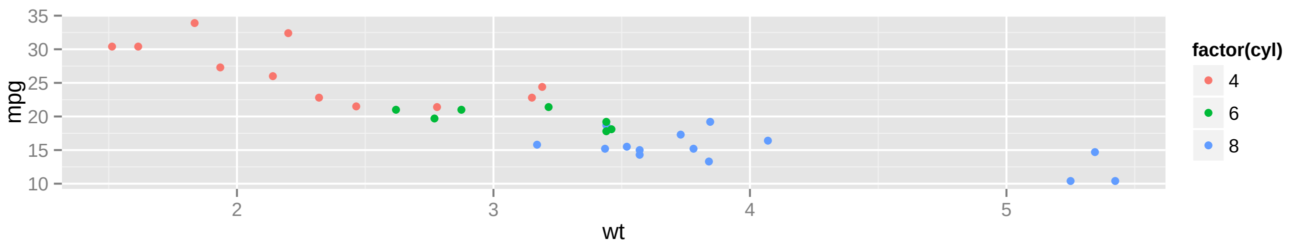


Figure 19: Full width figure

Note the use of the `fig.width` and `fig.height` chunk options to establish the proportions of the figure. Full width figures look much better if their height is minimized.

Main Column Figures

Besides margin and full width figures, you can of course also include figures constrained to the main column.

```
qplot(factor(cyl), mpg, data = mtcars, geom = "boxplot")
```

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. [3]

[3] 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.

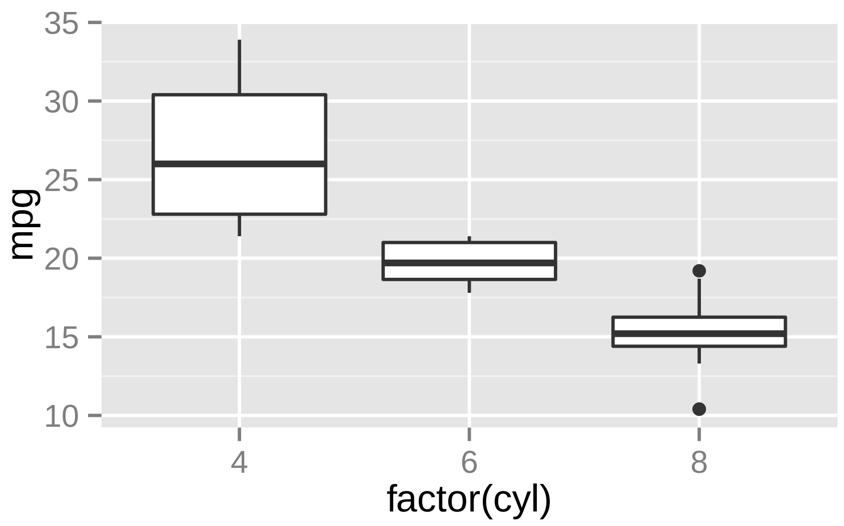


Figure 20: Another figure

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