EcoState as surplus production model

James Thorson

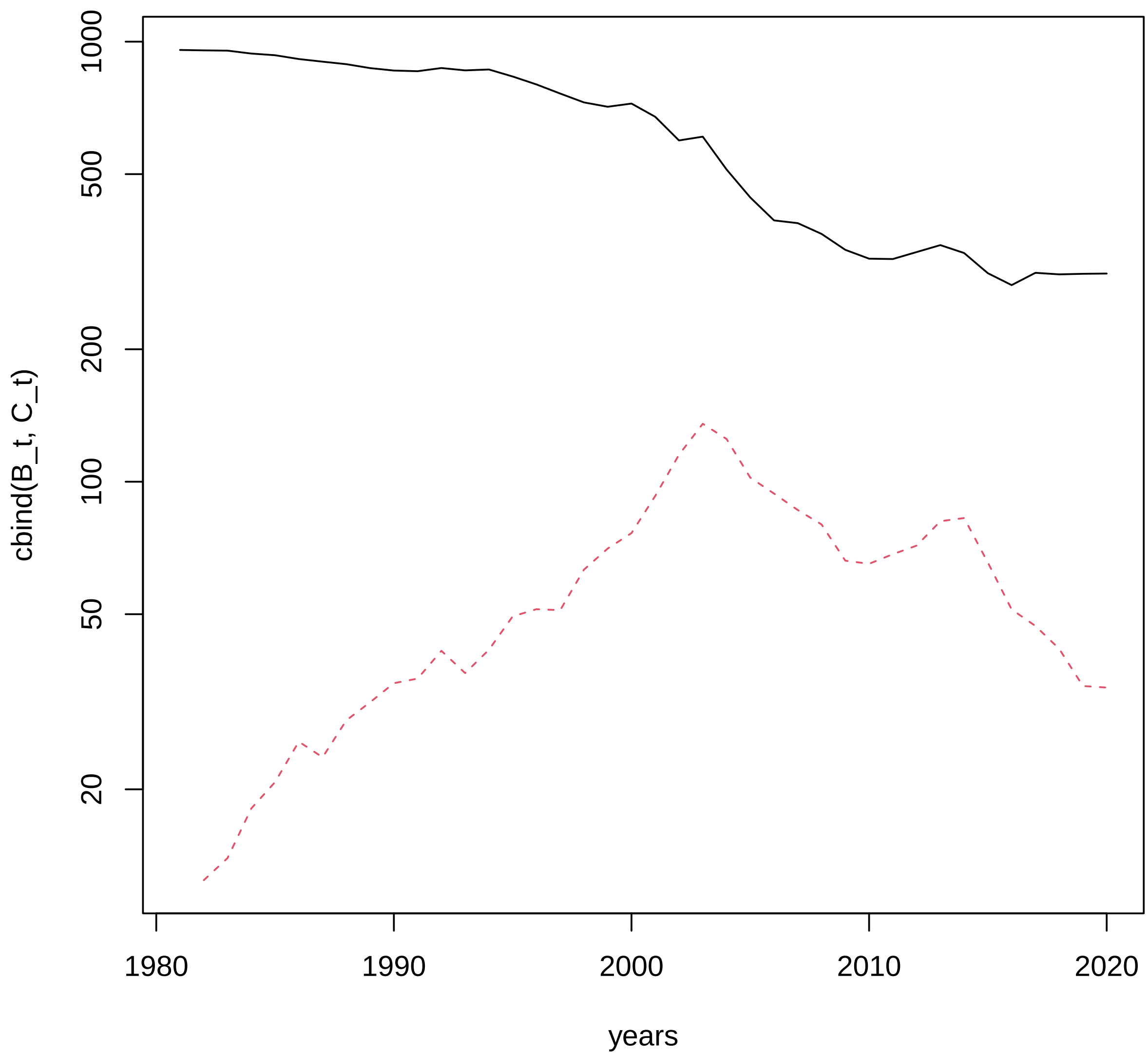
```r
library(EcoState)
```

`ecostate` is an R package for fitting the mass-balance dynamics specified by EcoSim as a state-space model. I can be used as a surplus production model by treating a single species as a "producer"

Simulation demonstration

We first simulate new data. To do so, we simulate a Schaefer production model with Gompertz effort dynamics:

```r
# Time-interval
years = 1981:2020
n_years = length(years)

# Biology
r = 0.2
K = 1000
sigmaB = 0.5
B0 = K * exp(sigmaB*rnorm(1))

# Effort dynamics
Bequil = 0.4 * K
Brate = 0.2
sigmaE = 0.1
E0 = 0.01

#
C_t = E_t = B_t = rep(NA, n_years)
B_t[1] = B0
E_t[1] = E0

#
for( t in 2:n_years ){
  B_t[t] = B_t[t-1] + r * B_t[t-1] * (1 - B_t[t-1]/K) * exp(sigmaB*rnorm(1))
  E_t[t] = E_t[t-1] * (B_t[t-1]/Bequil)^Brate * exp(sigmaE*rnorm(1))
  C_t[t] = B_t[t] * (1 - exp(-E_t[t]))
  B_t[t] = B_t[t] - C_t[t]
}

#
matplot( x=years, y=cbind(B_t,C_t), type="l", log="y")
```



We then set up inputs to EcoState

```
# Name taxa (optional, for illustration)
taxa = "target"
n_taxa = length(taxa)

# Ecopath-with-EcoSim parameters
# Diet matrix
DC_ij = array( 0, dim=c(1,1) )
PB_i = 0.1
QB_i = NA
EE_i = 1
B_i = 1
U_i = 0.2
```

```r
type_i = "auto"
which_primary = which(type_i=="auto")
which_detritus = which(type_i=="detritus")
V_ij = array( 2, dim=c(1,1) )

# reformat to longform data-frame
Catch = na.omit(data.frame( "Mass" = C_t, "Year" = years, "Taxon" = taxa ))
Biomass = data.frame( "Mass" = B_t, "Year" = years, "Taxon" = taxa )
```

Next, we fit them with `ecostate`

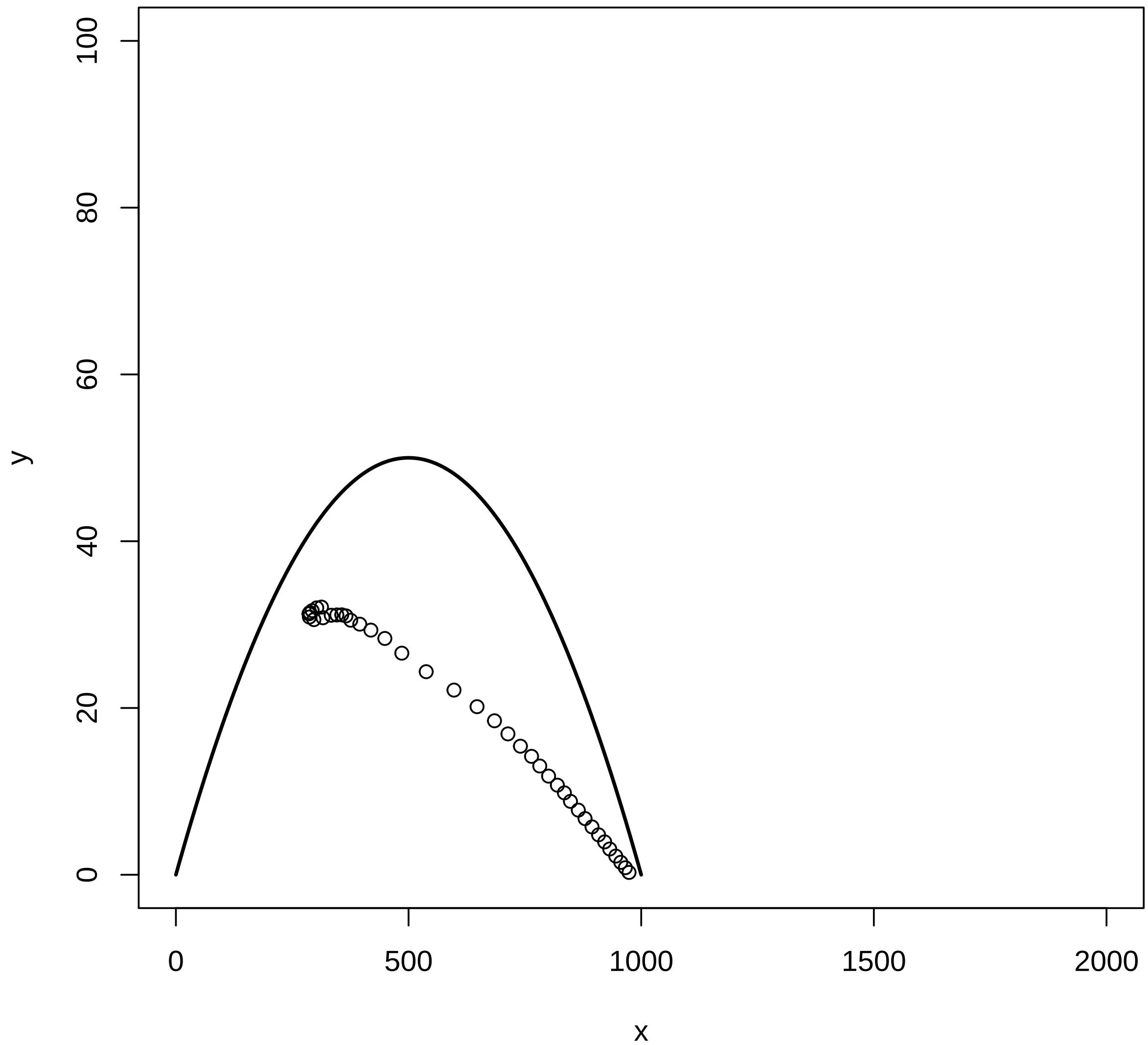
```r
# Settings: specify what parameters to estimate
fit_delta = taxa    # process errors
fit_Q = taxa        # catchability coefficient
fit_B0 = c()       # non-equilibrium initial condition
fit_B = taxa        # equilibrium biomass

type = factor("auto", levels=c("auto","hetero","detritus"))

# Label EwE inputs for each taxon as expected (so users can easily change taxa)
names(PB_i) = names(QB_i) = names(B_i) = names(EE_i) = names(type) = names(U_i) = taxa
  dimnames(DC_ij) = dimnames(V_ij) = list("Prey"=taxa, "Predator"=taxa)

# Run model
out0 = ecostate( taxa = taxa,
                years = years,
                catch = Catch,
                biomass = Biomass,
                PB = PB_i,
                QB = QB_i,
                DC = DC_ij,
                B = B_i,
                EE = EE_i,
                V = V_ij,
                type = type,
                U = U_i,
                fit_B = fit_B,
                fit_Q = fit_Q,
                fit_eps = fit_delta,
                fit_B0 = fit_B0,
                control = ecostate_control( inverse_method = "Standard",
                                            trace = 1,
                                            nlminb_loops = 0,
                                            getsd = FALSE,
                                            scale_solver = "simple",
                                            process_error = "epsilon" ) )

# Estimate logPB
pars = out0$tmb_inputs$p
  #pars$Vprime_ij[] = log(91 - 1)
map = out0$tmb_inputs$map
#map$logPB_i = factor(1)
map$Vprime_ij = factor(1)
```

```r
# Run model
out = ecostate( taxa = taxa,
                years = years,
                catch = Catch,
                biomass = Biomass,
                PB = PB_i,
                QB = QB_i,
                DC = DC_ij,
                B = B_i,
                EE = EE_i,
                V = V_ij,
                type = type,
                U = U_i,
                fit_B = fit_B,
                fit_Q = fit_Q,
                fit_eps = fit_delta,
                fit_B0 = fit_B0,
                control = ecostate_control( inverse_method = "Standard",
                                            trace = 1,
                                            nlminb_loops = 1,
                                            getsd = TRUE,
                                            scale_solver = "simple",
                                            process_error = "epsilon",
                                            map = map,
                                            tmb_par = pars ) )
#> Using `control$tmb_par`, so be cautious in constructing it
#> Using `control$map`, so be cautious in constructing it
#> Warning: This is an experimental compression method
#> Disable: 'config(tmbad.sparse_hessian_compress=0)'
#> done
#>   0:     194004.56:  0.00000  0.00000 -6.90776  0.00000
#>   1:     29895.515:  3.52723 -0.185825 -6.16375  1.72354
#>   2:     192.37308:  6.48126 -0.598082 -3.98327 0.190805
#>   3:     135.39392:  6.69977 -0.741684 -2.92306 -1.10807
#>   4:    -87.577084:  7.27840 -0.731335 -2.60918 -0.572867
#>   5:    -91.439541:  7.42611 -0.746277 -2.70930 -0.480954
#>   6:    -94.257742:  7.42793 -0.784154 -2.88742 -0.566673
#>   7:    -95.819626:  7.47597 -0.794758 -3.08258 -0.568087
#>   8:    -97.301331:  7.56082 -0.417927 -3.31814 -0.674572
#>   9:    -97.865051:  7.56126 -0.484479 -3.41384 -0.709539
#>  10:    -99.819128:  7.53294 -0.487480 -3.81494 -0.679519
#>  11:    -103.21645:  7.31262 -1.10493 -4.67358 -0.426014
#>  12:    -103.40542:  7.32248 -1.10664 -4.67391 -0.423803
#>  13:    -103.48160:  7.32411 -1.10947 -4.67528 -0.433422
#>  14:    -103.53174:  7.32752 -1.11998 -4.69244 -0.431517
#>  15:    -103.57385:  7.32478 -1.12748 -4.71131 -0.432507
#>  16:    -103.96678:  7.31119 -1.21455 -5.12663 -0.416269
#>  17:    -104.34393:  7.35739 -1.04373 -5.50882 -0.472252
#>  18:    -104.49342:  7.35991 -1.01254 -5.90830 -0.470768
#>  19:    -104.55531:  7.35731 -1.02526 -6.23921 -0.468100
#>  20:    -104.58710:  7.35550 -1.03424 -6.57025 -0.466595
#>  21:    -104.60284:  7.35548 -1.03615 -6.90142 -0.466345
#>  22:    -104.61084:  7.35497 -1.03543 -7.23259 -0.466334
```

```
#>  23:     -104.61543:  7.35569 -1.03484 -7.56375 -0.466599
#>  24:     -104.61771:  7.35558 -1.03479 -7.89493 -0.466575
#>  25:     -104.61886:  7.35550 -1.03491 -8.22610 -0.466540
#>  26:     -104.61945:  7.35549 -1.03495 -8.55727 -0.466531
#>  27:     -104.61975:  7.35550 -1.03495 -8.88844 -0.466533
#>  28:     -104.61991:  7.35550 -1.03494 -9.21961 -0.466534
#>  29:     -104.61999:  7.35550 -1.03494 -9.55078 -0.466534
#>  30:     -104.62003:  7.35550 -1.03494 -9.88195 -0.466534
#>  31:     -104.62005:  7.35550 -1.03494 -10.2131 -0.466534
#>  32:     -104.62006:  7.35550 -1.03494 -10.5443 -0.466534
#>  33:     -104.62007:  7.35550 -1.03494 -10.8755 -0.466534
#>  34:     -104.62007:  7.35550 -1.03494 -11.2066 -0.466534
#>  35:     -104.62007:  7.35550 -1.03494 -11.5378 -0.466534
#>  36:     -104.62007:  7.35550 -1.03494 -11.8690 -0.466534
#>  37:     -104.62007:  7.35550 -1.03494 -12.2001 -0.466534
#>  38:     -104.62007:  7.35550 -1.03494 -12.5313 -0.466534
#>  39:     -104.62007:  7.35550 -1.03494 -12.8625 -0.466534
#>  40:     -104.62007:  7.35550 -1.03494 -13.1937 -0.466534
#>  41:     -104.62007:  7.35550 -1.03494 -13.5248 -0.466534
#>  42:     -104.62007:  7.35550 -1.03494 -13.8560 -0.466534
#>  43:     -104.62007:  7.35550 -1.03494 -14.1872 -0.466534
```

Finally we can compare the estimated and true production function

```
P_t = (out$rep$g_ti[,1] - out$rep$m_ti[,1]) * out$rep$B_ti[,1]
B_t = out$rep$B_ti[,1]

x = seq(0, K, length=1000)
y = r * x * (1 - x/K)
plot( x=x, y=y, type="l", lwd=2, xlim=c(0,2*K), ylim=c(0,2*max(y)) )
points( x=B_t*exp(out$internal$parhat$logq_i), y=P_t*exp(out$internal$parhat$logq_i) )
```



100
80
60
40
20
0
y
0
500
1000
1500
2000
x