simulation & inference of transients

Carl Boettiger

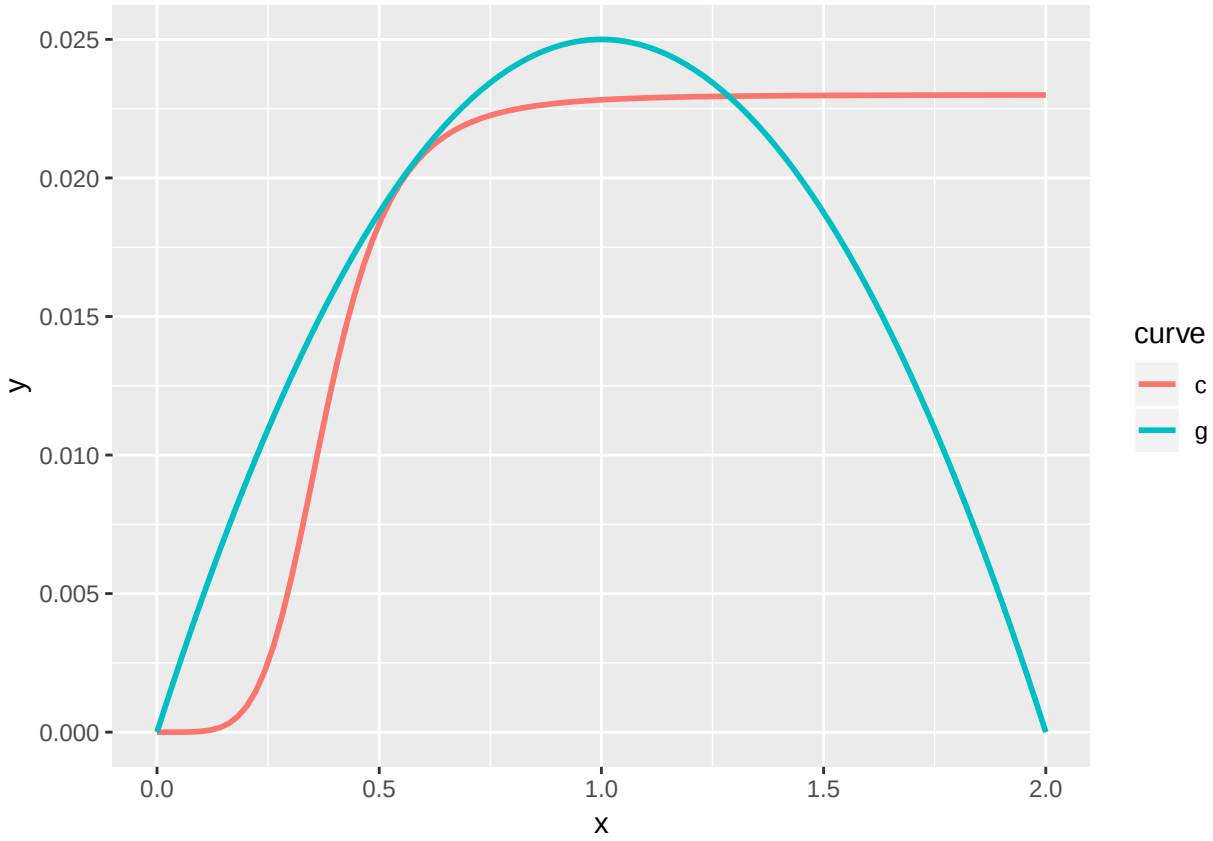
5/30/2019

```
library(tidyverse)
```

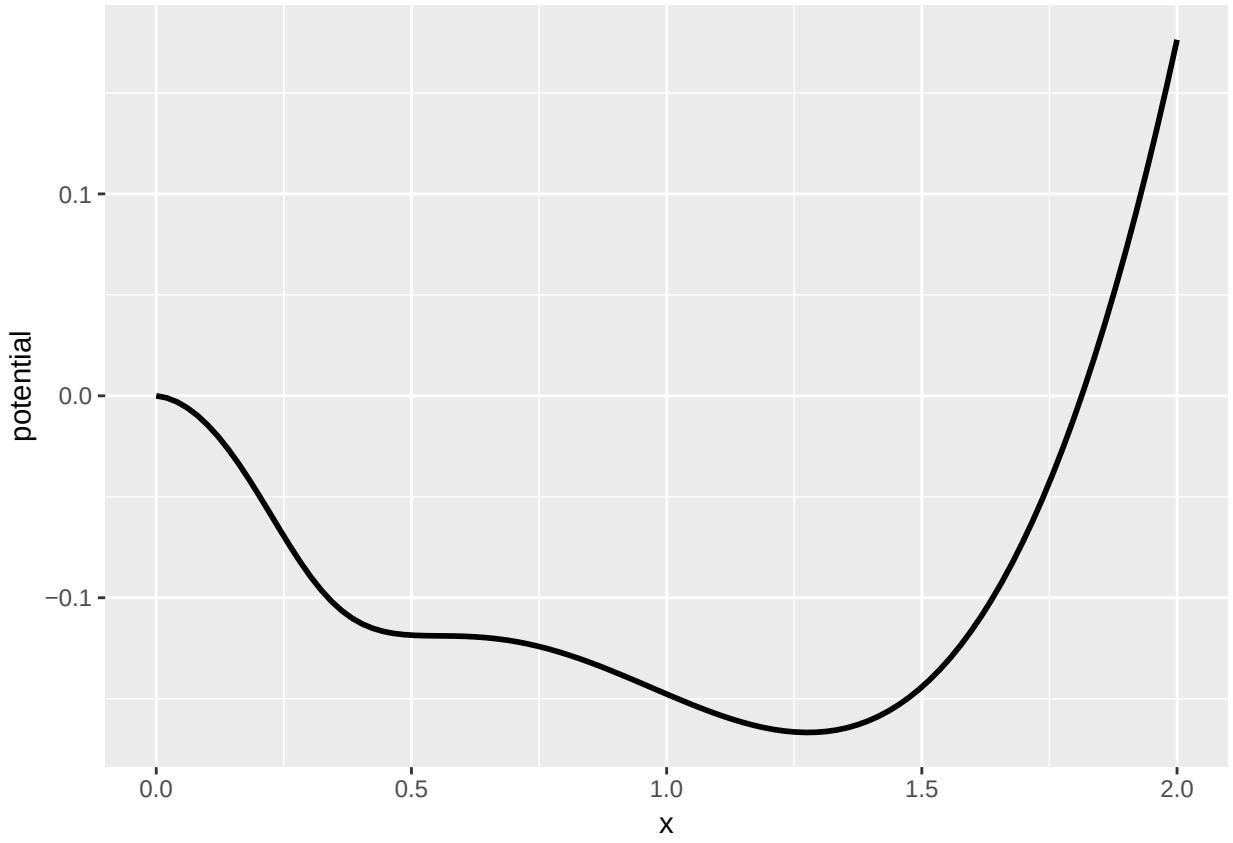
```
N <- 1e4
r <- 0.05; K <- 2
a <- 0.023; H <- 0.38; Q <- 5
x0 <- 0.2; sigma <- 0.02
growth <- function(x, r, K) x * r * (1 - x / K)
consumption <- function(x, a, H, Q) a * x^Q / (x^Q + H^Q)
```

```
theory <-
  tibble(x = seq(0,2, length.out = 100)) %>%
  mutate(g = growth(x, r, K),
         c = consumption(x, a, H, Q)) %>%
  mutate(potential = - cumsum(g - c)) %>%
  gather(curve, y, -x, -potential)
```

```
theory %>%
  ggplot(aes(x, y, col = curve)) +
  geom_line(lwd = 1)
```



```
theory %>%
  ggplot(aes(x, potential)) +
  geom_line(lwd = 1)
```




```
library(nimble)
```

```
## nimble version 0.7.1 is loaded.
## For more information on NIMBLE and a User Manual,
## please visit http://R-nimble.org.
```

```
##
## Attaching package: 'nimble'
```

```
## The following object is masked from 'package:stats':
##
##     simulate
```

```
# Define stochastic model in BUGS notation
may  <- nimble::nimbleCode({
  log(r) ~ dnorm(mu_r, sd_r)
  log(K) ~ dnorm(mu_K, sd_K)
  log(a) ~ dnorm(mu_a, sd_a)
  log(H) ~ dnorm(mu_H, sd_H)
  log(Q) ~ dnorm(mu_Q, sd_Q)
  log(x0) ~ dnorm(mu_x0, sd_x0)
  log(sigma) ~ dnorm(mu_sigma, sd_sigma)
  x[1] <- x0
  for(t in 1:(N - 1)){
    # Determinstic mean looks like standard R
```

```
    mu[t] <- x[t] + x[t] * r * (1 - x[t] / K) - a * x[t] ^ Q / (x[t] ^ Q + H ^ Q)
    # Note the use of ~ in BUGS to show 'distributed as normal'
    dB[t] ~ dnorm(0, sd = sigma)
    x[t + 1] <- max(0, mu[t] + dB[t])
  }
})
#
constants <- list(
  mu_r = 0, sd_r = 1,
  mu_K = 0, sd_k = 1,
  mu_a = 0, sd_a = 1,
  mu_H = 0, sd_H = 1,
  mu_Q = 0, sd_Q = 1,
  mu_x0 = 0, sd_x0 = 1,
  mu_sigma = 0, sd_sigma = 1
)

model <- nimbleModel(code = may, constants = constants)
```

```
## defining model...
```

```
## building model...
```

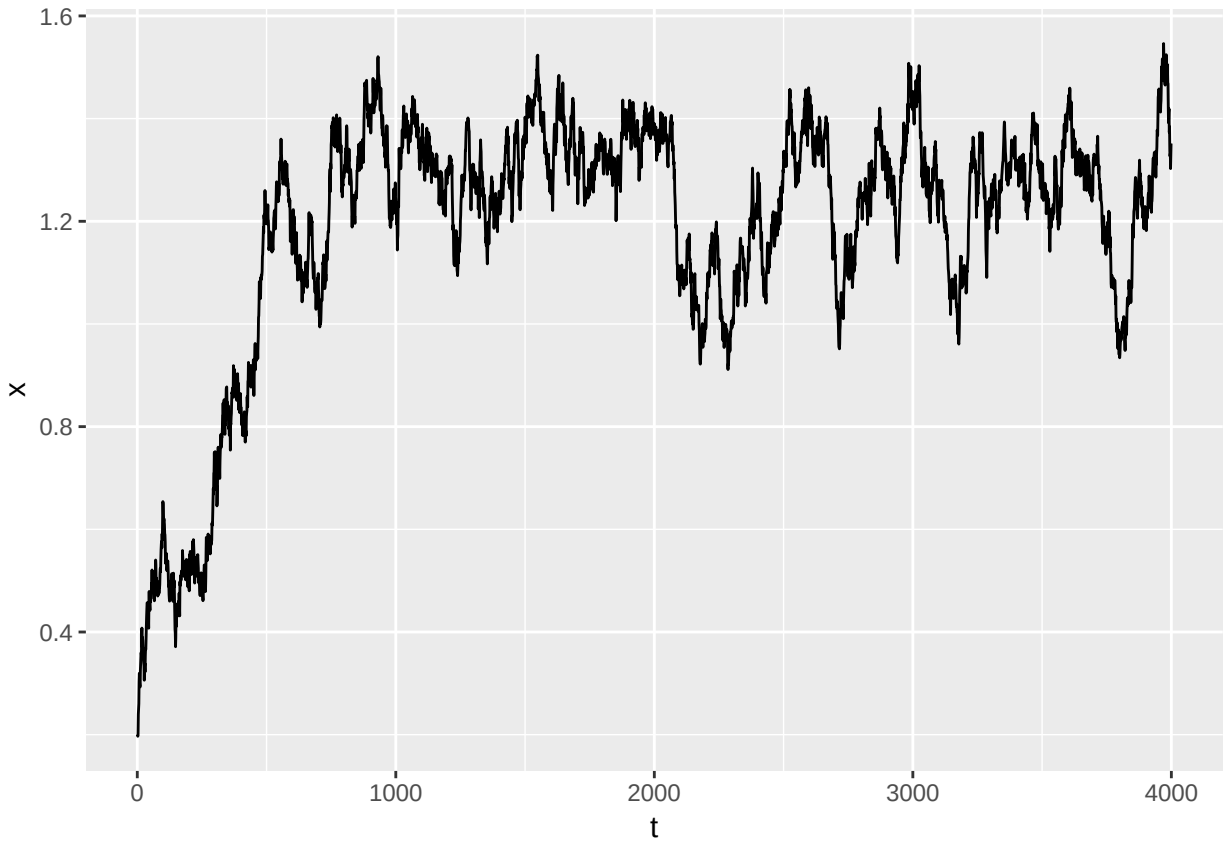
```
## running calculate on model (any error reports that follow may simply reflect missing values in model
## checking model sizes and dimensions... This model is not fully initialized. This is not an error. To
## model building finished.
```

```
cmodel <- compileNimble(model)
```

```
## compiling... this may take a minute. Use 'showCompilerOutput = TRUE' to see C++ compilation details.
## compilation finished.
```

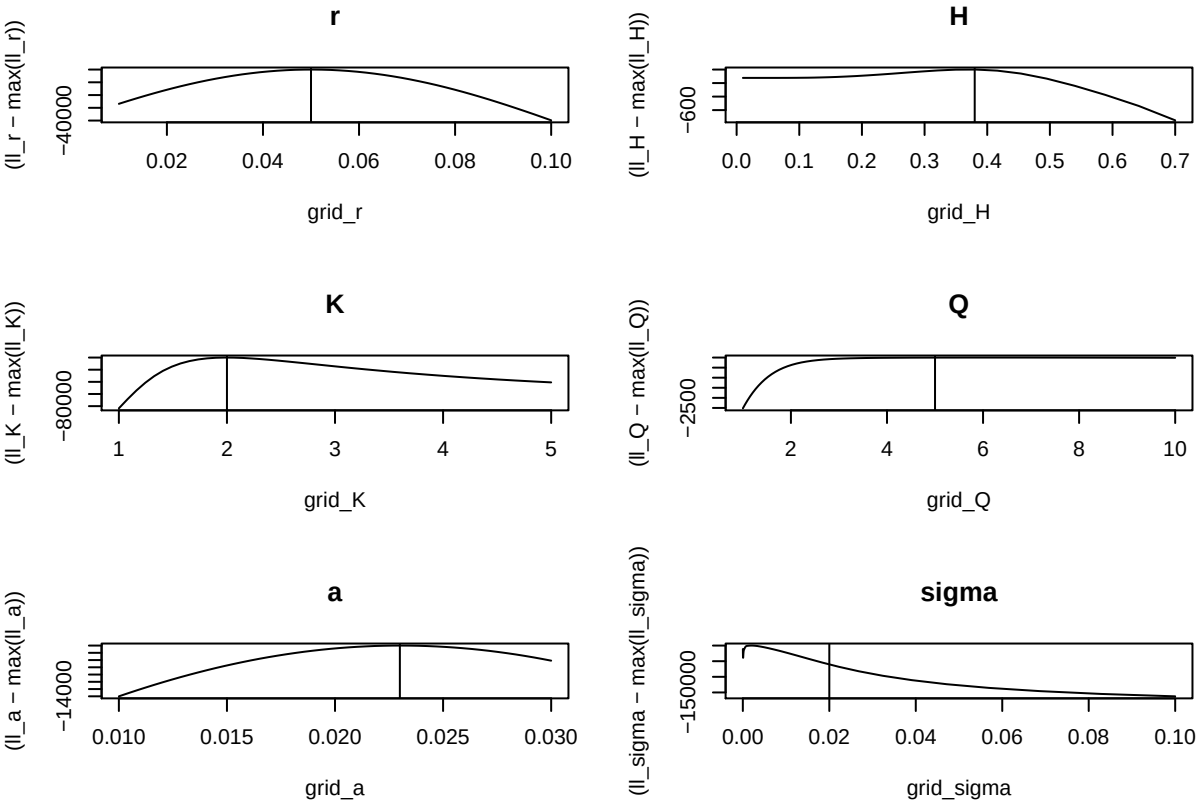
```
set.seed(123)
cmodel$r <- r; cmodel$K <- K
cmodel$a <- a; cmodel$H <- H; cmodel$Q <- Q
cmodel$x0 <- x0; cmodel$sigma <- sigma
simulate(cmodel, nodes = c('x', 'mu', 'dB'))
df <- tibble(t = seq_along(cmodel$x), x = cmodel$x)
```

```
df %>% filter(t < 4000) %>% ggplot(aes(t, x)) + geom_line()
```




```
get_ll <- function(x, r, K, a, H, Q, sigma){
  TIME <- length(x)
  mu <- x + x * r * (1 - x / K) - a * x^Q / (x^Q + H^Q)
  sum(dnorm(x = x[-1], mean = mu[-TIME], sd = sigma), log = T)
}
##
grid_r <- seq(1e-2, 1e-1, l=5e1)
ll_r <- sapply(grid_r, function(r) get_ll(a = cmodel$a, x = cmodel$x, r = r, K = cmodel$K,
                                          H = cmodel$H, Q = cmodel$Q, sigma = cmodel$sigma))
##
grid_K <- seq(1, 5, l=5e1)
ll_K <- sapply(grid_K, function(K) get_ll(a = cmodel$a, x = cmodel$x, r = cmodel$r, K = K,
                                          H = cmodel$H, Q = cmodel$Q, sigma = cmodel$sigma))
##
grid_a <- seq(1e-2, 3e-2, l=5e1)
ll_a <- sapply(grid_a, function(a) get_ll(a = a, x = cmodel$x, r = cmodel$r, K = cmodel$K,
                                          H = cmodel$H, Q = cmodel$Q, sigma = cmodel$sigma))
##
grid_H <- exp(seq(log(1e-2), log(0.7), l=5e1))
ll_H <- sapply(grid_H, function(H) get_ll(a = cmodel$a, x = cmodel$x, r = cmodel$r, K = cmodel$K,
                                          H = H, Q = cmodel$Q, sigma = cmodel$sigma))
##
grid_Q <- exp(seq(log(1), log(10), l=5e1))
ll_Q <- sapply(grid_Q, function(Q) get_ll(a = cmodel$a, x = cmodel$x, r = cmodel$r, K = cmodel$K,
                                          H = cmodel$H, Q = Q, sigma = cmodel$sigma))
##
grid_sigma <- exp(seq(log(1e-5), log(1e-1), l=5e1))
```

```r
ll_sigma <- sapply(grid_sigma, function(sigma)
  get_ll(a = cmodel$a, x = cmodel$x, r = cmodel$r, K = cmodel$K,
         H = cmodel$H, Q = cmodel$Q, sigma = sigma))
##
layout(matrix(1:6, 3, 2))
plot(grid_r, (ll_r - max(ll_r)), type = "l", main = "r")
abline(v = cmodel$r)
plot(grid_K, (ll_K - max(ll_K)), type = "l", main = "K")
abline(v = cmodel$K)
plot(grid_a, (ll_a - max(ll_a)), type = "l", main = "a")
abline(v = cmodel$a)
plot(grid_H, (ll_H - max(ll_H)), type = "l", main = "H")
abline(v = cmodel$H)
plot(grid_Q, (ll_Q - max(ll_Q)), type = "l", main = "Q")
abline(v = cmodel$Q)
plot(grid_sigma, (ll_sigma - max(ll_sigma)), type = "l", main = "sigma")
abline(v = cmodel$sigma)
```




```r
ll_QH <- apply(expand.grid(grid_Q, grid_H), 1, function(pair)
  if(pair[2] > (0.6 / 3) * pair[1] + 0.18 - (0.6 / 3)) NA else
  get_ll(a = cmodel$a, x = cmodel$x, r = cmodel$r, K = cmodel$K,
         H = pair[2], Q = pair[1], sigma = cmodel$sigma))
ll_Ka <- apply(expand.grid(grid_K, grid_a), 1, function(pair)
  get_ll(a = pair[2], x = cmodel$x, r = cmodel$r, K = pair[1],
         H = cmodel$H, Q = cmodel$Q, sigma = cmodel$sigma))
ll_rK <- apply(expand.grid(grid_r, grid_K), 1, function(pair)
  get_ll(x = cmodel$x, r = pair[1], K = pair[2], a = cmodel$a,
         H = cmodel$H, Q = cmodel$Q, sigma = cmodel$sigma))
```

```
ll_asigma <- apply(expand.grid(grid_a, grid_sigma), 1, function(pair)
  get_ll(x = cmodel$x, r = cmodel$r, K = cmodel$K, a = pair[1],
         H = cmodel$H, Q = cmodel$Q, sigma = pair[2]))
```

```
layout(matrix(1:4, 2, 2))
image(grid_Q, grid_H, matrix((ll_QH - max(ll_QH, na.rm = T)), length(grid_Q), length(grid_H)))
points(cmodel$Q, cmodel$H)
##
image(grid_K, grid_a, matrix((ll_Ka - max(ll_Ka)), length(grid_K), length(grid_a)))
points(cmodel$K, cmodel$a)
##
image(grid_r, grid_K, matrix((ll_rK - max(ll_rK)), length(grid_r), length(grid_K)))
points(cmodel$r, cmodel$K)
##
image(grid_a, grid_sigma, matrix((ll_asigma - max(ll_asigma)), length(grid_a), length(grid_sigma)))
points(cmodel$a, cmodel$sigma)
```