Notes for cue vs uptake vs half-saturation trade-off invasion model (JAM-BSrev)

K Todd-Brown (ktoddbrown@gmail.com)

2017-04-11

Contents

TODO: Convert color scheme to wesanderson colorpack

Set up

```
library(assertthat)
library(reshape2)
library(ggplot2)
library(GGally)
library(plyr)
library(rootSolve)
library(deSolve)
library(knitr)
library(pander)
library(cowplot)

#sourceFiles <- 'R/invasion.R'
#l_ply(sourceFiles, source)

set.seed(100) #reproducible random selection

sessionInfo()
```

```
## R version 3.3.2 (2016-10-31)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: macOS Sierra 10.12.4
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] cowplot_0.7.0  pander_0.6.0   knitr_1.15.1   deSolve_1.14
##  [5] rootSolve_1.7  plyr_1.8.4     GGally_1.3.0   ggplot2_2.2.1
##  [9] reshape2_1.4.2 assertthat_0.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.9        magrittr_1.5       munsell_0.4.3
##  [4] colorspace_1.3-2   stringr_1.2.0      tools_3.3.2
##  [7] grid_3.3.2         gtable_0.2.0       htmltools_0.3.5
## [10] yaml_2.1.14        lazyeval_0.2.0     rprojroot_1.2
## [13] digest_0.6.12      tibble_1.2         RColorBrewer_1.1-2
## [16] evaluate_0.10      rmarkdown_1.3      stringi_1.1.2
## [19] scales_0.4.1       backports_1.0.5    reshape_0.8.6
```

Basic model description

The goal of this model to simulation biologically mediated decomposition. There are two main pools, Substrate (C) and Biomass (B). Carbon is added to the substrate pool via inputs (I), leached from both pools at a rate proportional to the carbon in the pool (h fraction of the biomass pool B, and m fraction of the substate pool C), and transfer between the substrate C pool and biomass B pool via a reverse Monod uptake with a maximum rate v, half-saturation constant k, and carbon use effiency ϵ. Reverse Monod uptake was selected to ensure that both substrate pools were sensitive to changes in inputs.

$\frac{dB}{dt} = \frac{\epsilon v B C}{k+B} - hB$

$\frac{dC}{dt} = I - mC - \frac{vBC}{k+B}$

At steady state ($\frac{dB}{dt} = \frac{dC}{dt} = 0$) this implies that:

$B = \frac{Iv\epsilon - mhk}{hv+mh}$

$C = \frac{h}{\epsilon v}(k+B)$

If we further assume that there is a trade off between both 1) uptake v and carbon use effiency (ϵ) such that $v = \frac{e^{b\epsilon}-e^b}{1-e^b}$ and the half saturation constant (k) such that $k = av + k_{min} = \frac{a}{1-e^b}(e^{b\epsilon} - e^b)$. Defined as:

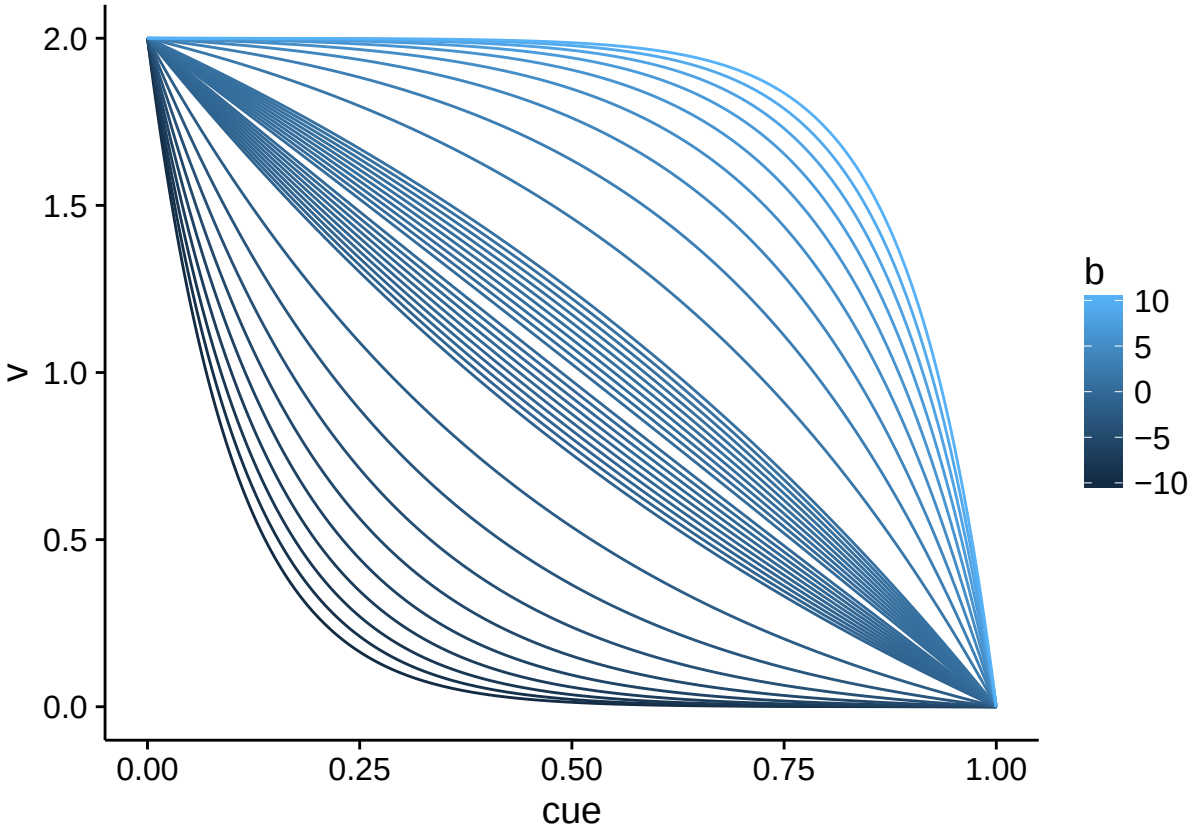
This leads to the following steady state calculations for one biomass pool:

```
cue_v_tradeoff <- function(b, vmax, cue){
  return(vmax*(exp(b*cue)-exp(b))/(exp(0)-exp(b)))
}

v_k_tradeoff <- function(kmin, a, v){
  return(kmin+a*v)
}
```

```
steadyState <- function(b, vmax, kmin, a, h, m, I, cue){
  v <- cue_v_tradeoff(b=b, vmax=vmax, cue)
  k <- v_k_tradeoff(kmin=kmin, a=a, v=v)

  B <- (I*v*cue-m*h*k)/(h*v+m*h)
  C <- h/(cue*v) * (k+B)

  return(list(B=B, C=C))
}
```

Visualize trade-off between cue vs uptake and k

```
tradeoff.df <- adply(.data=c(-0.1*1:9, -1*1:10, 0.1*1:9, 1:10), .margins=c(1), .id=c('id'), .fun=functi
  ans <- data.frame(b=b, cue=seq(0, 1, length=1000))
  ans$v <- cue_v_tradeoff(b=b, vmax=2, cue=ans$cue)
  return(ans)
})

ggplot(tradeoff.df) + geom_line(aes(x=cue, y=v, group=b, color=b))
```




```
tradeoff.df$k <- v_k_tradeoff(kmin=0, a=100/2, tradeoff.df$v) #slope is max SOC over vmax range
```

Steady state sensitivity to inputs

Recall that at steady state:

$B = \frac{I v \epsilon - mhk}{hv + mh}$

$C = \frac{h}{\epsilon v}(k + B)$

This imples that: $\frac{dB}{dI} = \frac{v\epsilon}{hv+mh}$ $\frac{dC}{dI} = \frac{1}{v+m}$

Optimum carbon use effiency

We want the optimum carbon use effiency where biomass is maximum as steady state but that is not possible to solve analytically (you end up with $y = xe^x$, solve for x). We instead bruite force a solution, iterating ϵ over $[0, 1]$ in incraments of 0.01

```
optimum_cue <- function(b, vmax, kmin, a, h, m, I){
  cue <- seq(0, 1, length=100)
  ans <- steadyState(b, vmax, kmin, a, h, m, I, cue)
  ans$B[ans$B <= 0 | ans$C <= 0] <- -Inf
  ans$C[ans$B <= 0 | ans$C <= 0] <- -Inf

  best <- which.max(ans$B)
  return(list(B_opt=ans$B[best], C_opt=ans$C[best], cue_opt=cue[best]))
}
```

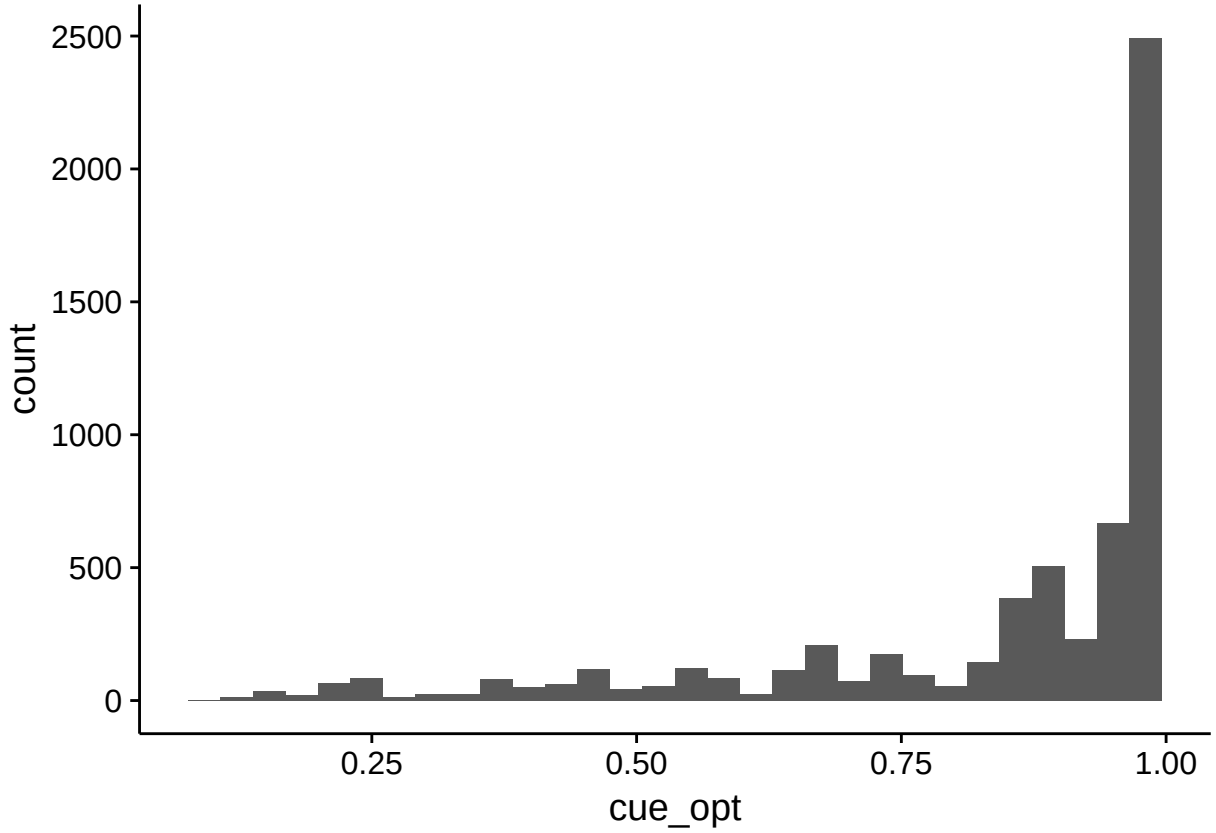
Generate reasonable parameter sets without compitition

Given that we have input targets between 0.1 and 10 mg-C per g-soil per day, total carbon pools of between 10 to 500 mg-C per g-soil, and a biomass to total carbon ratio of between 0.1 and 15 percent. Let's generate some reasonable parameters to draw on.

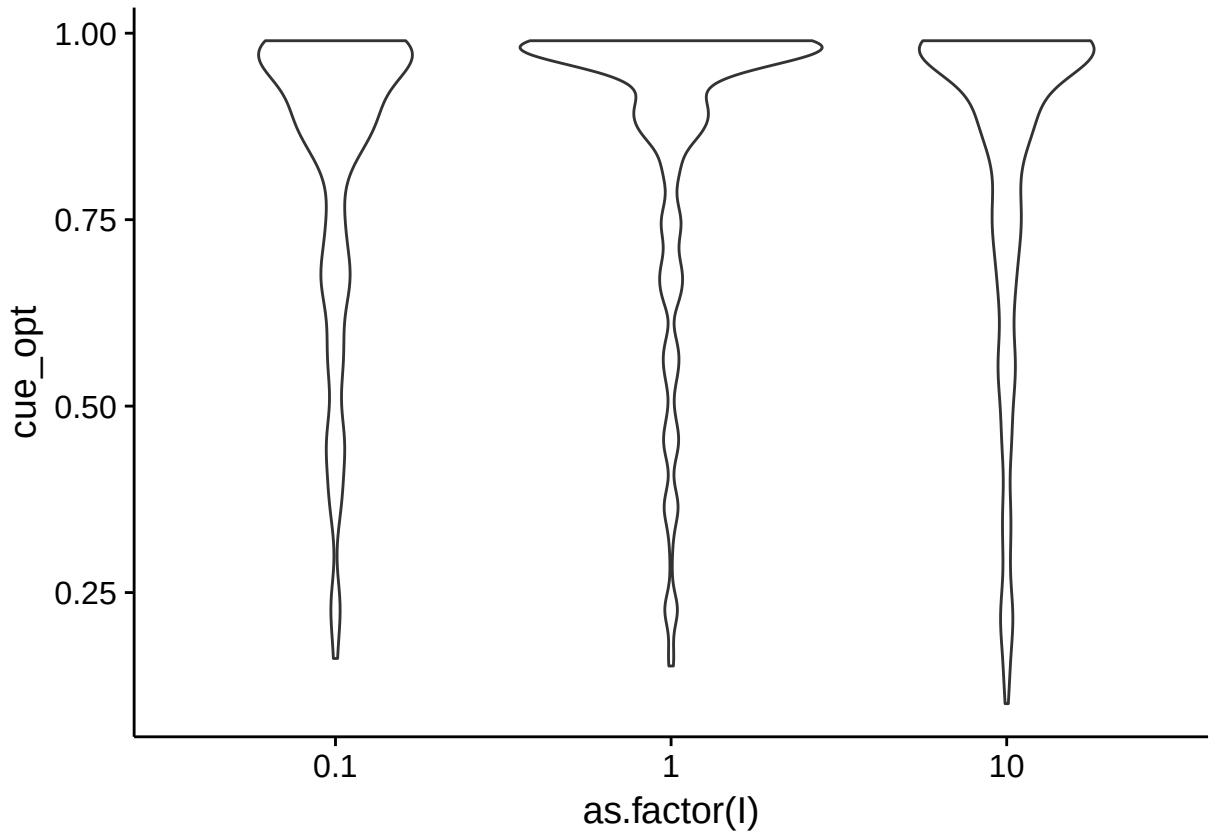
```
checkPools <- function(B, C){
  return( B > 0 & C > 0 &
    B+C > 10 & B+C < 500 & B/(B+C)> 0.001 & B/(B+C) < 0.15)
}

parm.ls <- list(b=c(-10, -1, -0.1, 0.1, 1, 10),
                vmax=c(0.1, 0.5, 1, 5, 10, 100),
                kmin=c(0, 1, 10, 100, 1000),
                a=c(1/10, 1, 10, 100, 1000),
                h=c(1e-4, 1e-3, 1e-2, 0.1, 1, 10),
                m=c(1e-4, 1e-3, 1e-2, 0.1, 1, 10),
                I=c(0.1, 1, 10))

parm.df <- expand.grid(parm.ls)

parm.df <- ddply(parm.df, names(parm.df), function(xx){
  ans <- as.data.frame(optimum_cue(b=xx$b, vmax=xx$vmax, kmin=xx$kmin, a=xx$a,
              h=xx$h, m=xx$m, I=xx$I))
  return(ans)
})
```

```
parm.df$validPools_opt <- checkPools(B=parm.df$B_opt, C=parm.df$C_opt)


ggplot(subset(parm.df, validPools_opt)) + geom_histogram(aes(x=cue_opt))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(subset(parm.df, validPools_opt)) + geom_violin(aes(x=as.factor(I), y=cue_opt))
```



Competition model co-existance

Consider the senerio where the native popuation is invated by a second competiting population that has a different carbon use effiency but is otherwise equivalent. Let the strategic carbon use effiency be where the native popuation is never invadable no matter what carbon use effiency value the invader has.

$\frac{dB_i}{dt} = \frac{\epsilon_i v_i B_i C}{k_i + B_i} - hB_i$ $\frac{dB_n}{dt} = \frac{\epsilon_n v_n B_n C}{k_n + B_n} - hB_n$ $\frac{dC}{dt} = I - mC - \frac{v_n B_n C}{k_n + B_n} - \frac{v_i B_i C}{k_i + B_i}$ where n and i stand for the native and invading populations respectively.

Assume steady state again $C = \frac{I}{m} - \frac{h}{m}(\frac{B_i}{\epsilon_i} + \frac{B_n}{\epsilon_n})$

$B_i = \frac{\epsilon_i v_i}{h} C - k_i$

Implies that $(\frac{B_i}{\epsilon_i} + \frac{B_n}{\epsilon_n}) = \frac{v_i}{h} C - \frac{k_i}{\epsilon_i} + \frac{v_n}{h} C - \frac{k_n}{\epsilon_n}$ $(\frac{B_i}{\epsilon_i} + \frac{B_n}{\epsilon_n}) = C\frac{v_n + v_i}{h} - (\frac{k_i}{\epsilon_i} + \frac{k_n}{\epsilon_n})$

Implying that $mC = I - h(C\frac{v_n + v_i}{h} - (\frac{k_i}{\epsilon_i} + \frac{k_n}{\epsilon_n}))$ $C(m + v_n + v_i) = I + h(\frac{k_i}{\epsilon_i} + \frac{k_n}{\epsilon_n})$ $C = [I + h(\frac{k_i}{\epsilon_i} + \frac{k_n}{\epsilon_n})](m + v_n + v_i)^{-1}$

This requires a new steady state equation

```
steadyState_compete <- function(b, vmax, kmin, a, h, m, I, cue_i, cue_n){
  v_i <- cue_v_tradeoff(b=b, vmax=vmax, cue_i)
  k_i <- v_k_tradeoff(kmin=kmin, a=a, v=v_i)

  v_n <- cue_v_tradeoff(b=b, vmax=vmax, cue_n)
  k_n <- v_k_tradeoff(kmin=kmin, a=a, v=v_n)

  C <- (I + h * (k_i/cue_i + k_n/cue_n))/(m + v_n + v_i)
  B_i <- C * cue_i * v_i / h - k_i
```

```
  B_n <- C * cue_n * v_n / h - k_n

  return(list(C=C, B_i=B_i, B_n=B_n))
}
```

Strategic CUE

If both CUE are the same then the populations converge to the same biomass at steady state (but are not the same as a single biomass pool because $\frac{1}{k+B_i} + \frac{1}{k+B_n} \neq \frac{1}{k+B_i+B_n}$). We are looking for the CUE value where, if the invasion cue is only slightly different from the native CUE we minimize the difference between the two biomass pools.

```
strategic_cue <- function(b, vmax, kmin, a, m, h, I){
  cue <- seq(0, 1, length=100)
  v <- cue_v_tradeoff(b=b, vmax=vmax, cue)
  k <- v_k_tradeoff(kmin=kmin, a=a, v=v)

  ans <- steadyState_compete(b, vmax, kmin, a, h, m, I, cue_n=cue, cue_i=cue-0.005)

  measure <- abs(ans$B_n-ans$B_i)
  measure[ans$B_n <= 0 | ans$C <= 0] <- Inf
  flag <- which.min(measure)

  return(list(cue_strat=cue[flag], C_strat=ans$C[flag], B_strat=ans$B_n[flag]))
}
```

Tillman

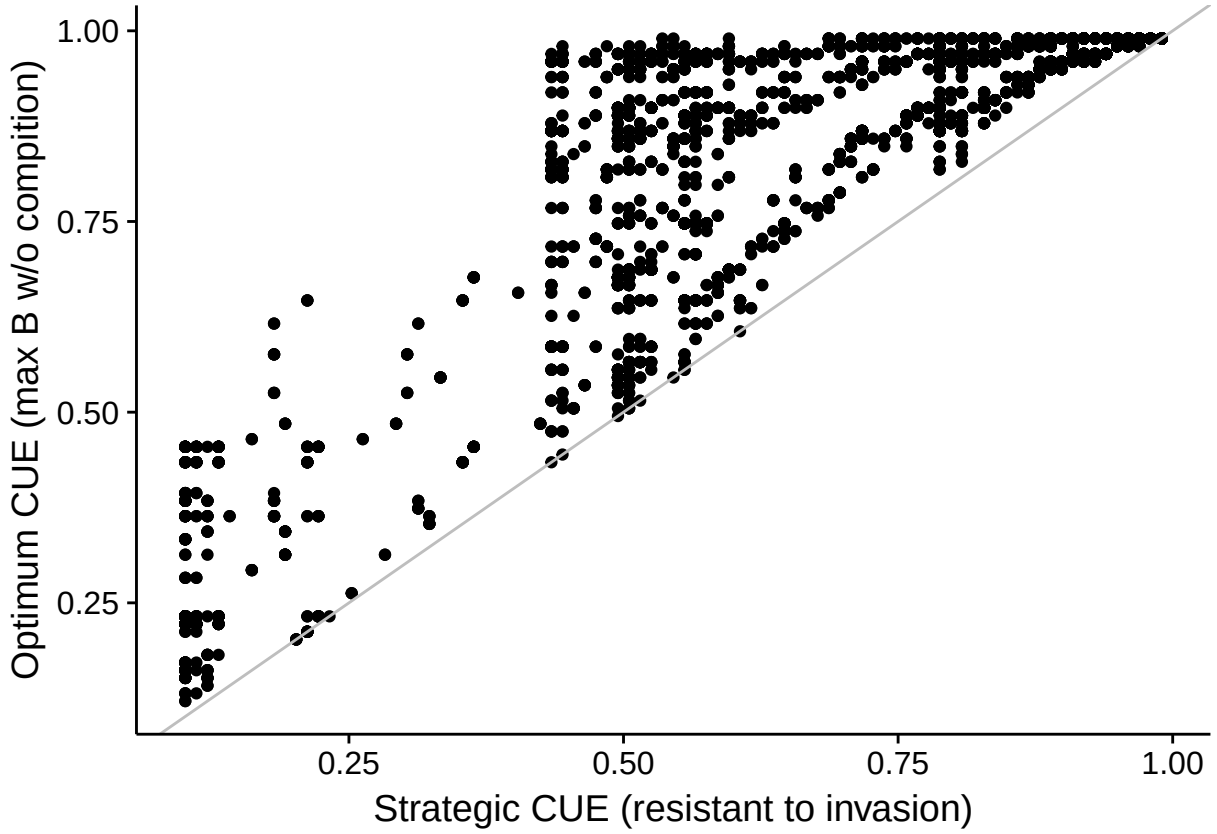
Tillman's R* maximaizes the growth rate. The logic goes something like this. In the strategic solution there exists some ϵ_n such that for all ϵ_i, $B_i = 0$ and $B_n > 0$. This is true when the only valid invasion parameterization is equal to the nature population (ie $\epsilon_n = \epsilon_i$) and is the only solution for $\frac{h}{C} = \frac{\epsilon_n v_n}{k_n + B_n}$, that is CUE is selected to maximize microbial uptake.

```
tillman_cue <- function(b, vmax, kmin, a, m, h, I){
  cue <- seq(0, 1, length=100)
  v <- cue_v_tradeoff(b=b, vmax=vmax, cue)
  k <- v_k_tradeoff(kmin=kmin, a=a, v=v)

  ans <- steadyState_compete(b, vmax, kmin, a, h, m, I, cue, cue)

  measure <- cue*v/(k+ans$B_n)
  measure[ans$B_n <= 0 | ans$C <= 0] <- -Inf
  flag <- which.max(measure)

  return(list(cue_til=cue[flag], C_til=ans$C[flag], B_til=ans$B_n[flag]))
}
```

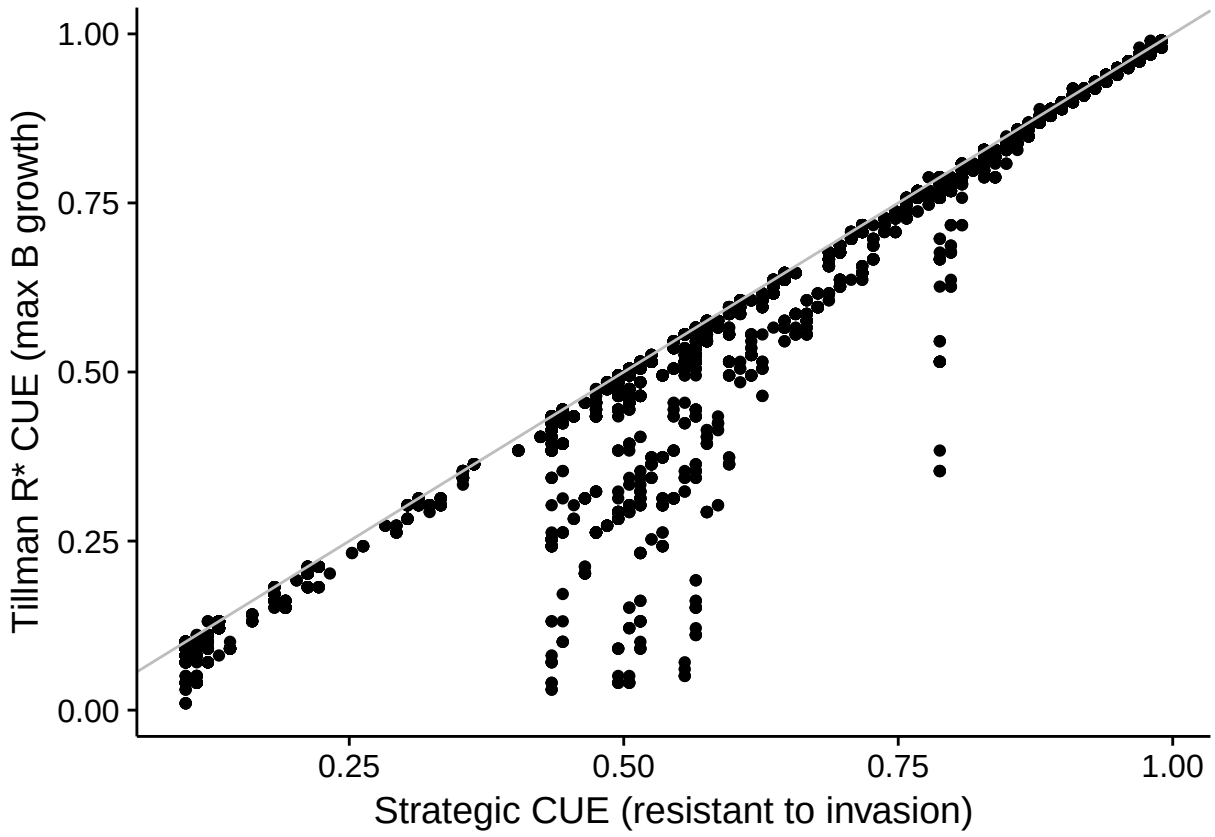
```
parm.df <- ddply(parm.df, names(parm.df), function(xx){
  return(as.data.frame(tillman_cue(b=xx$b, vmax=xx$vmax, kmin=xx$kmin,
                        a=xx$a, m=xx$m, h=xx$h, I=xx$I)))
})

parm.df$validPools_til <- with(parm.df, checkPools(B=parm.df$B_til, C=parm.df$C_til))
```

```
parm.df <- ddply(parm.df, names(parm.df), function(xx){
  return(as.data.frame(strategic_cue(b=xx$b, vmax=xx$vmax, kmin=xx$kmin,
                         a=xx$a, m=xx$m, h=xx$h, I=xx$I)))
})

parm.df$validPools_strat <- with(parm.df, checkPools(parm.df$B_strat, C=parm.df$C_strat))
```

```
ggplot(subset(parm.df, validPools_opt & validPools_strat)) +
  geom_point(aes(y=cue_opt, x=cue_strat)) +
  geom_abline(slope=1, intercept=c(0), color='grey') +
  labs(y='Optimum CUE (max B w/o compition)', x='Strategic CUE (resistant to invasion)')
```

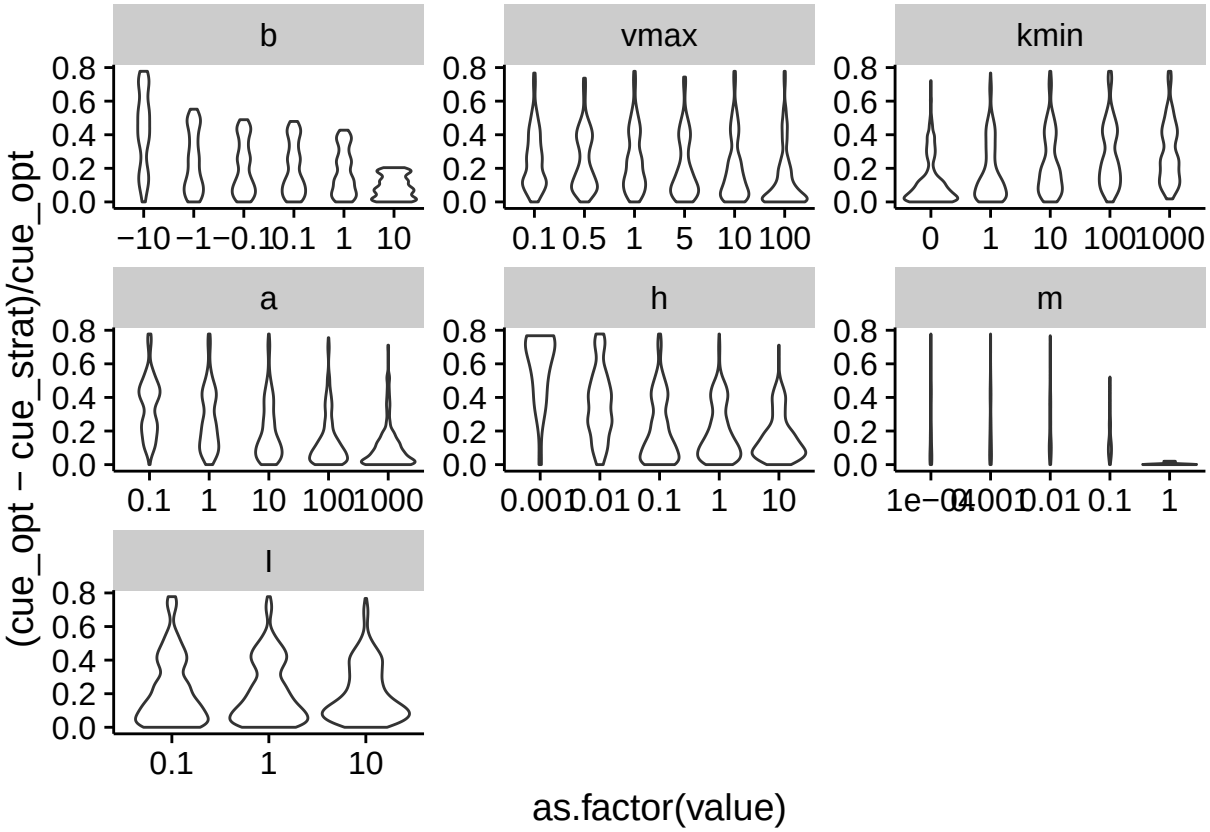



```
ggplot(subset(parm.df, validPools_til & validPools_strat)) +
  geom_point(aes(y=cue_til, x=cue_strat)) +
  geom_abline(slope=1, intercept=c(0), color='grey') +
  labs(y='Tillman R* CUE (max B growth)', x='Strategic CUE (resistant to invasion)')
```



Explore drivers of strategic and optimum differences

```
diff.df <- subset(parm.df, validPools_opt & validPools_strat)

diff.df <- melt(diff.df, measure.vars=names(parm.ls))

ggplot(diff.df) + geom_violin(aes(x=as.factor(value), y=(cue_opt-cue_strat)/cue_opt)) +
    facet_wrap(~variable, scale='free')
```



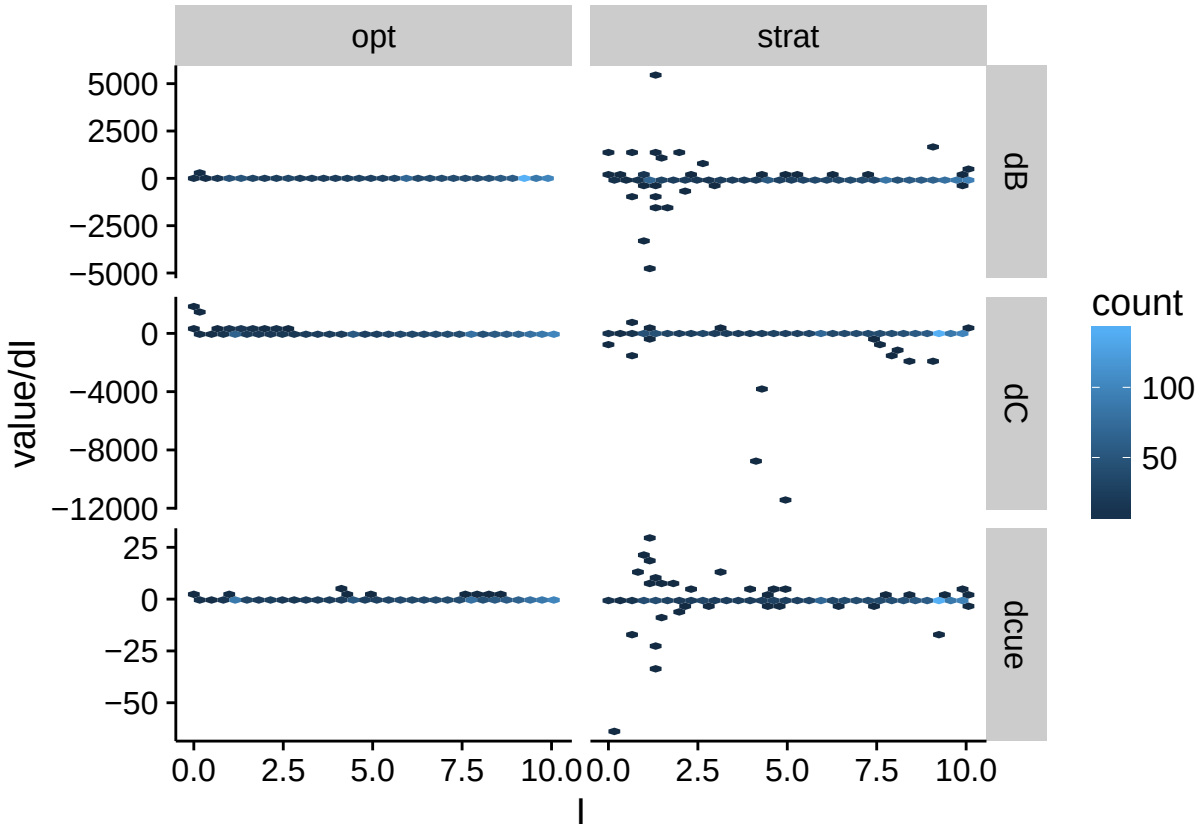
We are particullary interested in differences in inputs

```
validParms <- unique(subset(parm.df, validPools_strat & validPools_opt,
                     select=c("b", "vmax", "kmin", "a", "h", "m")))
I.arr <- seq(0.1, 10, by=0.3)
Igrad <- ddply(validParms, c("b", "vmax", "kmin", "a", "h", "m"), function(xx){
      ans1 <- ddply(data.frame(xx, I=I.arr), c('I'), function(yy){
        yy1 <- with(data=yy, data.frame(optimum_cue(b, vmax, kmin, a, h, m, I)))
        yy2 <- with(data=yy, data.frame(strategic_cue(b, vmax, kmin, a, h, m, I)))
        if(nrow(yy1) == nrow(yy2)){
          return(cbind(yy1, yy2))
        }else
          return(data.frame())
      })
      ans2 <- ddply(data.frame(xx, I=I.arr*rnorm(length(I.arr), mean=1, sd=0.05)),
                    c('I'), function(yy){
        yy1 <- with(data=yy, data.frame(optimum_cue(b, vmax, kmin, a, h, m, I)))
        yy2 <- with(data=yy, data.frame(strategic_cue(b, vmax, kmin, a, h, m, I)))
        if(nrow(yy1) == nrow(yy2)){
          return(cbind(yy1, yy2))
        }else
          return(data.frame())
      })

      if(nrow(ans1) != nrow(ans2)){
        #print(xx[c("b", "vmax", "kmin", "a", "h", "m"),])
        return(NULL)
      }
```

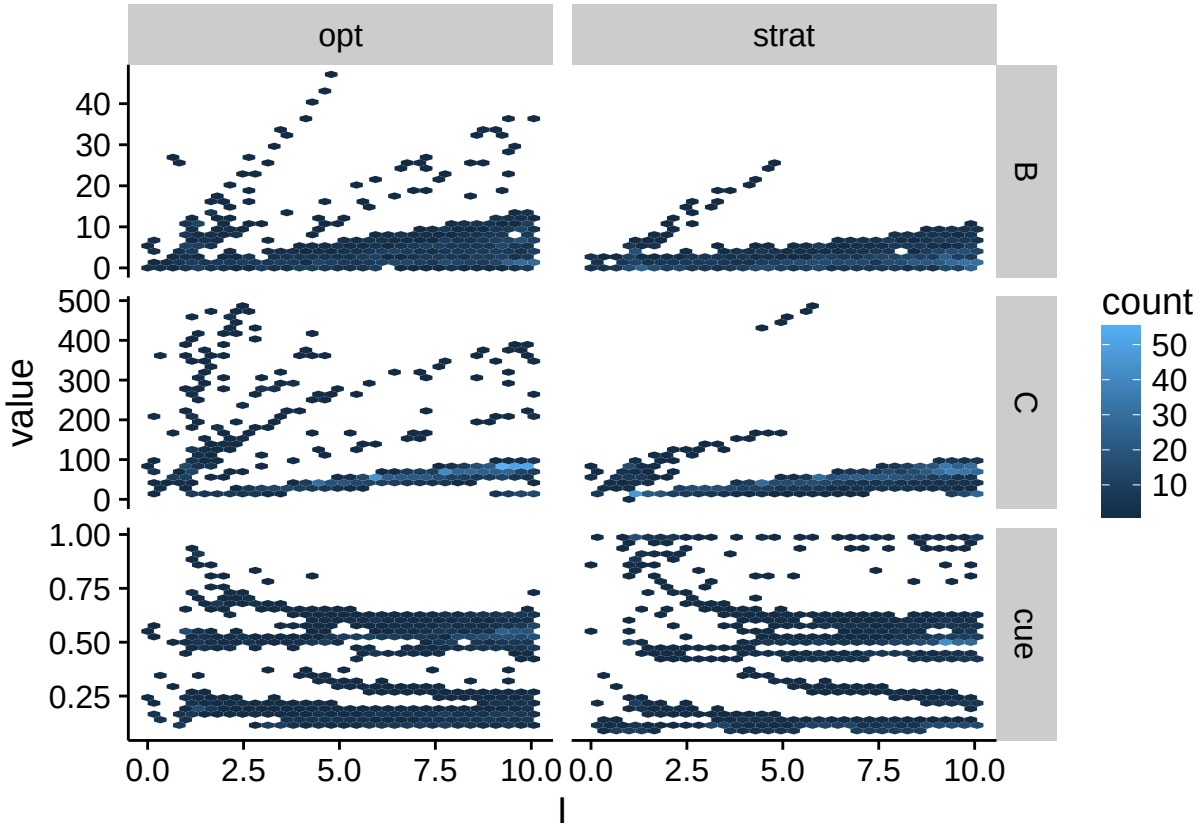
```
      dvar <- ans1-ans2
      names(dvar) <- sprintf('d%s', names(dvar))

      return(cbind(ans1, dvar))
})
Igrad$validParam <- with(Igrad, checkPools(B=B_opt, C=C_opt)&
                           checkPools(B=B_strat, C=C_strat))
```

```
plot.df <- melt(subset(Igrad, validParam),
                measure.vars=names(Igrad)[grepl('_(opt)|(strat)', names(Igrad))])
temp <- as.data.frame(matrix(unlist(strsplit(as.character(plot.df$variable), '_|\\.')), ncol=2, byrow=T
names(temp) <- c('variable', 'type')
plot.df$variable <- NULL
plot.df <- cbind(plot.df, temp)
plot.df$isGrad <- grepl('d', as.character(plot.df$variable))

ggplot(subset(plot.df, isGrad)) + geom_hex(aes(x=I, y=value/dI)) +
  facet_grid(variable~type, scales='free')
```

```
## Warning: Removed 20 rows containing non-finite values (stat_binhex).
```




```
ggplot(subset(plot.df, !isGrad)) + geom_hex(aes(x=I, y=value)) +
  facet_grid(variable~type, scales='free')
```



Numerical validation code

Let's pick some points where the gap between the Optimum CUE and Strategic CUE is small (< 20 percent) and large (> 20 percent).

```
bigGap.df <- subset(parm.df, validPools_opt & validPools_strat & abs(cue_strat-cue_opt) > 0.2)
smallGap.df <- subset(parm.df, validPools_opt & validPools_strat & abs(cue_strat-cue_opt) < 0.2)

numParm.df <- rbind.fill(bigGap.df[sample.int(nrow(bigGap.df), size=4),],
                         smallGap.df[sample.int(nrow(smallGap.df), size=4),])
numParm.df$index <- 1:nrow(numParm.df)
pander(numParm.df[,c(names(parm.ls), c('cue_opt','C_opt', 'B_opt', 'cue_strat','C_strat', 'B_strat'))])
```

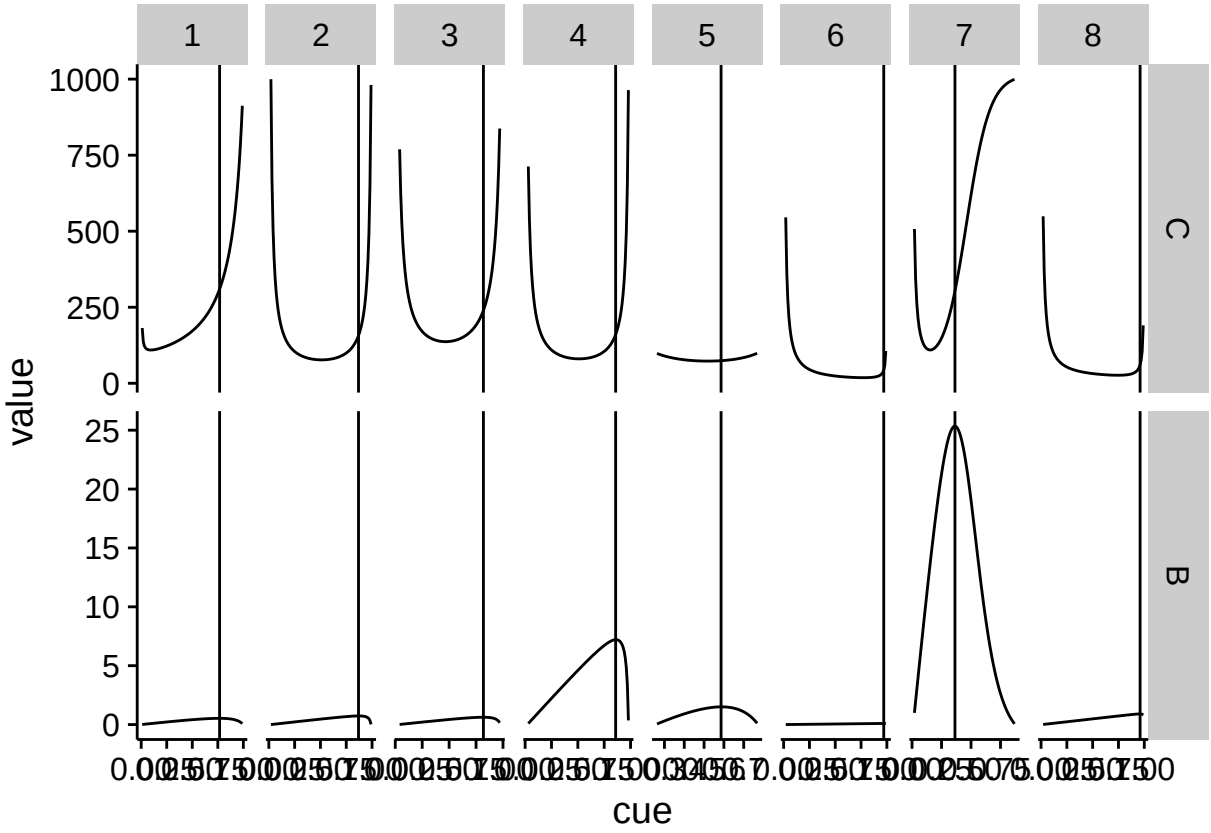
Table 1: Table continues below

b	vmax	kmin	a	h	m	I	cue_opt	C_opt	B_opt	cue_strat
-0.1	0.1	0	0.1	10	0.01	10	0.7677	310	0.5297	0.5051
0.1	1	1	1	10	0.01	10	0.8687	157	0.7323	0.5758
-0.1	0.5	1	1	10	0.01	10	0.8182	239.5	0.6222	0.5354
0.1	5	1000	10	0.1	0.001	1	0.8586	160.2	7.211	0.5152
-1	1	1000	1000	0.01	0.001	0.1	0.5859	74.32	1.505	0.5253
0.1	100	10	1	10	0.001	1	0.9697	43.05	0.0928	0.7778
-10	1	1	100	0.1	0.01	10	0.3636	303	25.35	0.2121
10	0.1	1	1	1	0.001	1	0.9596	60.65	0.9014	0.7879

C_strat	B_strat
95.27	0.2273
67.8	0.2674
120.7	0.2344
80.19	2.35
72.91	0.6035
18.44	0.03528
88.68	9.559
21.23	0.3845

Validate optimium CUE

```
opt.validate <- ddply(numParm.df, c('index', 'C_opt', 'B_opt'), function(xx){
  cue <- seq(0, 1, length=100)
  ans <- steadyState(xx$b, xx$vmax, xx$kmin, xx$a, xx$h, xx$m, xx$I, cue)
  return(data.frame(cue=cue, C=ans$C, B=ans$B))
})

plot.df <- melt(subset(opt.validate, B>0 & C>0), measure.vars=c('C', 'B'))
ggplot(plot.df) + geom_line(aes(x=cue, y=value, group=index)) +
  geom_vline(data=numParm.df, aes(xintercept=cue_opt)) +
  facet_grid(variable~index, scales='free')
```

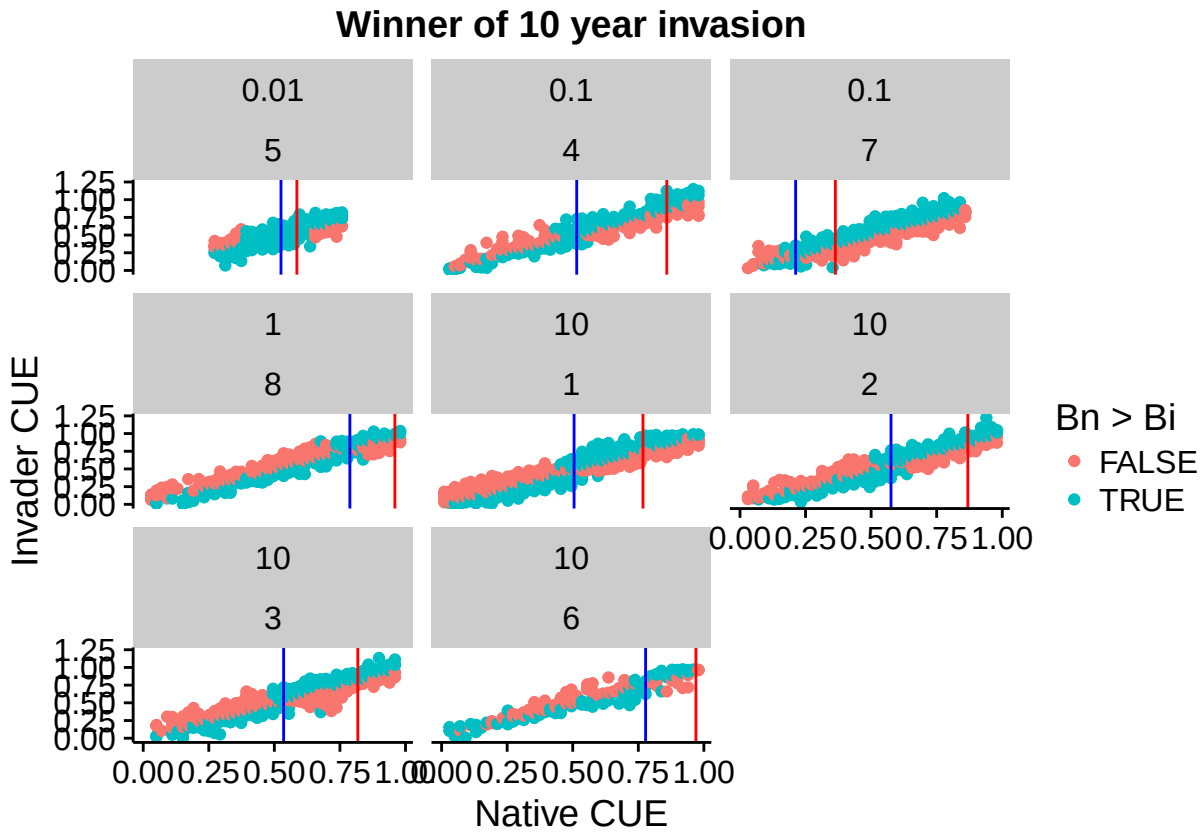


Validate strategic CUE

```
#numParm.df <- subset(parm.df, validPools_opt)
#numParm.df$index <- 1:nrow(numParm.df)
#numParm.df <- subset(numParm.df, index %in% sample(numParm.df$index, size=10))

compitition.model <- function(t, y, parms){
  C <- y[1]; Bn <- y[2]; Bi <- y[3]

  ans <- with(parms,
              c(dC = I - m*C - C*(vn*Bn/(kn+Bn) + vi*Bi/(ki+Bi)),
                   dBn = cuen*vn*Bn*C/(kn+Bn) - hn*Bn,
                   dBi = cuei*vi*Bi*C/(ki+Bi) - hi*Bi)
       )
  return(list(ans))
}

runInvasions <- ddply(numParm.df, c('index'), function(xx){

  parm <- as.list(xx[, c('b', 'vmax', 'kmin', 'a', 'm', 'I')])

  parm$hn <- xx$h
  parm$hi <- xx$h

  cueCombo <- expand.grid(cuen=seq(1, 100, length=50)/100,
                          rel_cuei=rnorm(10, sd=0.1))
  cueCombo$cuei <- cueCombo$cuen+rnorm(nrow(cueCombo), sd=0.1)
  cueCombo <- cueCombo[cueCombo$cuei > 0,]

  cueCombo$vn <- with(parm, cue_v_tradeoff(b=b, vmax=vmax, cueCombo$cuen))
  cueCombo$kn <- with(parm, v_k_tradeoff(kmin=kmin, a=a, v=cueCombo$vn))

  cueCombo$vi <- with(parm, cue_v_tradeoff(b=b, vmax=vmax, cueCombo$cuei))
  cueCombo$ki <- with(parm, v_k_tradeoff(kmin=kmin, a=a, v=cueCombo$vi))

  invade.df <- ddply(cueCombo, c('cuen', 'cuei'), function(cuePairs){
    comboParm <- c(parm, cuePairs)
    preInvade <- with(comboParm, steadyState(b, vmax, kmin, a, h=hn, m, I, cue=cuen))
    if(all(preInvade > 0)){
      y0 <- list(C=preInvade$C, Bn=preInvade$B, Bi=preInvade$B*0.1)
      invasion <- tryCatch({lsoda(y=unlist(y0), times=c(1, 7, 30, 365, 365*10),
                                  func=compitition.model, parms=comboParm)},
                           warning = function(w){return(data.frame())},
                           error = function(e){return(data.frame())}
      )
      names(y0) <- paste(names(y0), '0', sep='')
      if(nrow(invasion) == 5){
        finalPools <- as.data.frame(c(comboParm, y0, as.list(invasion[nrow(invasion),])))
      }else{
        finalPools <- as.data.frame(comboParm)
      }
    }else{
      finalPools <- as.data.frame(comboParm)
```

```
    }
  })

  return(invade.df)
})

runInvasions <- ddply(runInvasions, c('index'), function(xx){
  ans <- steadyState_compete(b=xx$b, vmax=xx$vmax, kmin=xx$kmin, a=xx$a, m=xx$m, I=xx$I,
                             h=xx$hn, cue_i=xx$cuei, cue_n=xx$cuen)
  names(ans) <- c('C_ss', 'B_i_ss', 'B_n_ss')
  return(data.frame(xx, as.data.frame(ans)))
})
```

```
temp <- merge(runInvasions, numParm.df, by='index')
ggplot(subset(temp, Bn > 0 & Bi > 0)) +
  geom_point(aes(x=cuen, y=cuei, color=Bn > Bi)) +
  geom_vline(data=numParm.df, aes(xintercept=cue_strat), color='blue') +
  geom_vline(data=numParm.df, aes(xintercept=cue_opt), color='red') +
  labs(x='Native CUE', y='Invader CUE', title='Winner of 10 year invasion') +
  facet_wrap(~h+index)
```

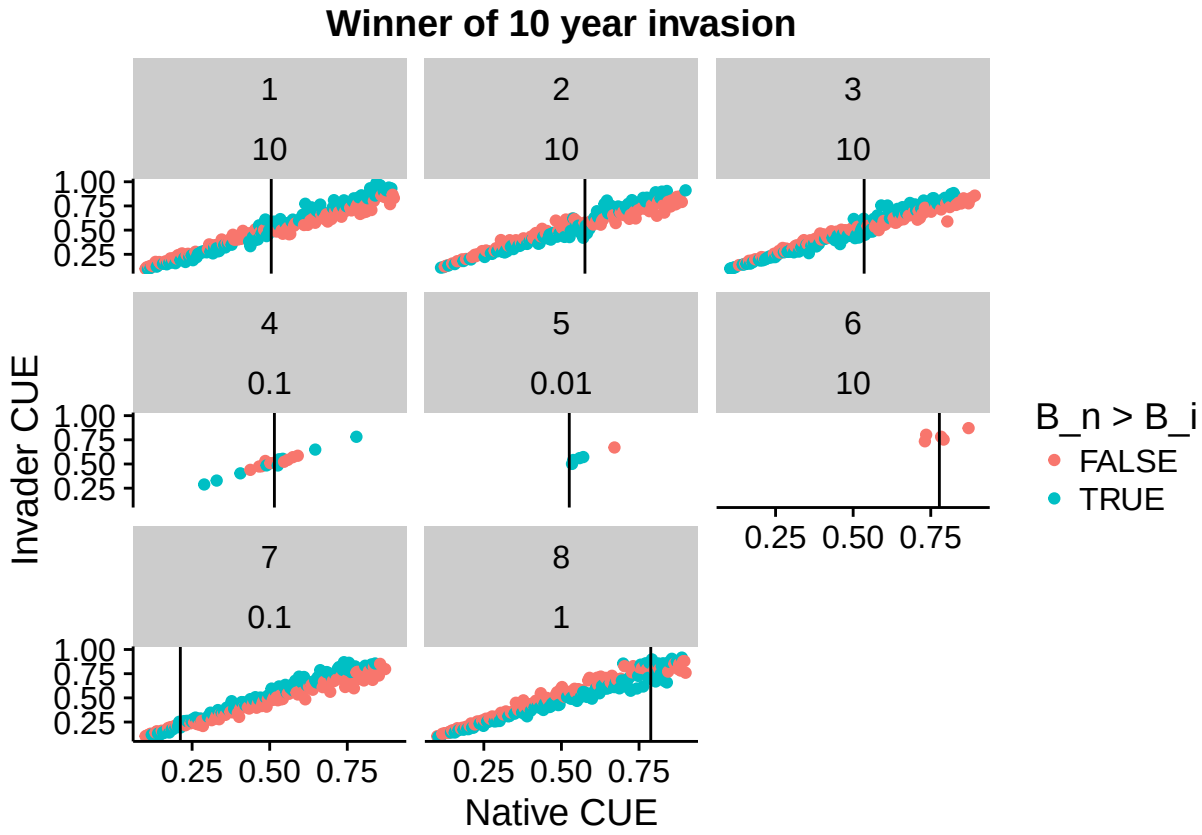


Numerical sketch book below

At steady state: $C = \frac{I+\sum_i \frac{k_i h}{\epsilon_i}}{m+\sum_i v_i}$ If there are n identically parameterized pools then: $C_n = \frac{I+n\frac{kh}{\epsilon}}{m+nv}$ Which is not the same as a single biomass pool $C_1 = \frac{I+\frac{kh}{\epsilon}}{m+v}$.

See below for a numerical example.

```
runInvasions_ss <- ddply(numParm.df, c('index'), function(xx){

  parm <- as.list(xx[, c('b', 'vmax', 'kmin', 'a', 'm', 'I')])

  ans <- ddply(data.frame(cue=seq(0.1, 0.9, length=200)), c('cue'), function(yy){
    ans <- data.frame(cue_n=yy$cue, cue_i=yy$cue*rnorm(1, mean=1, sd=0.1))
    return(data.frame(ans,
                      as.data.frame(steadyState_compete(b=xx$b,
                                                        vmax=xx$vmax, kmin=xx$kmin,
                          a=xx$a, m=xx$m, h=xx$h, I=xx$I,
                          cue_n=ans$cue_n, cue_i=ans$cue_i)
                      )))
  })
})

#merge(runInvasions_ss, numParm.df)
ggplot(subset(merge(runInvasions_ss, numParm.df), B_n > 0 & B_i > 0 & C > 0)) +
  geom_point(aes(x=cue_n, y=cue_i, color=B_n > B_i)) +
  geom_vline(data=numParm.df, aes(xintercept=cue_strat)) +
  labs(x='Native CUE', y='Invader CUE', title='Winner of 10 year invasion') +
  facet_wrap(~index+h)
```

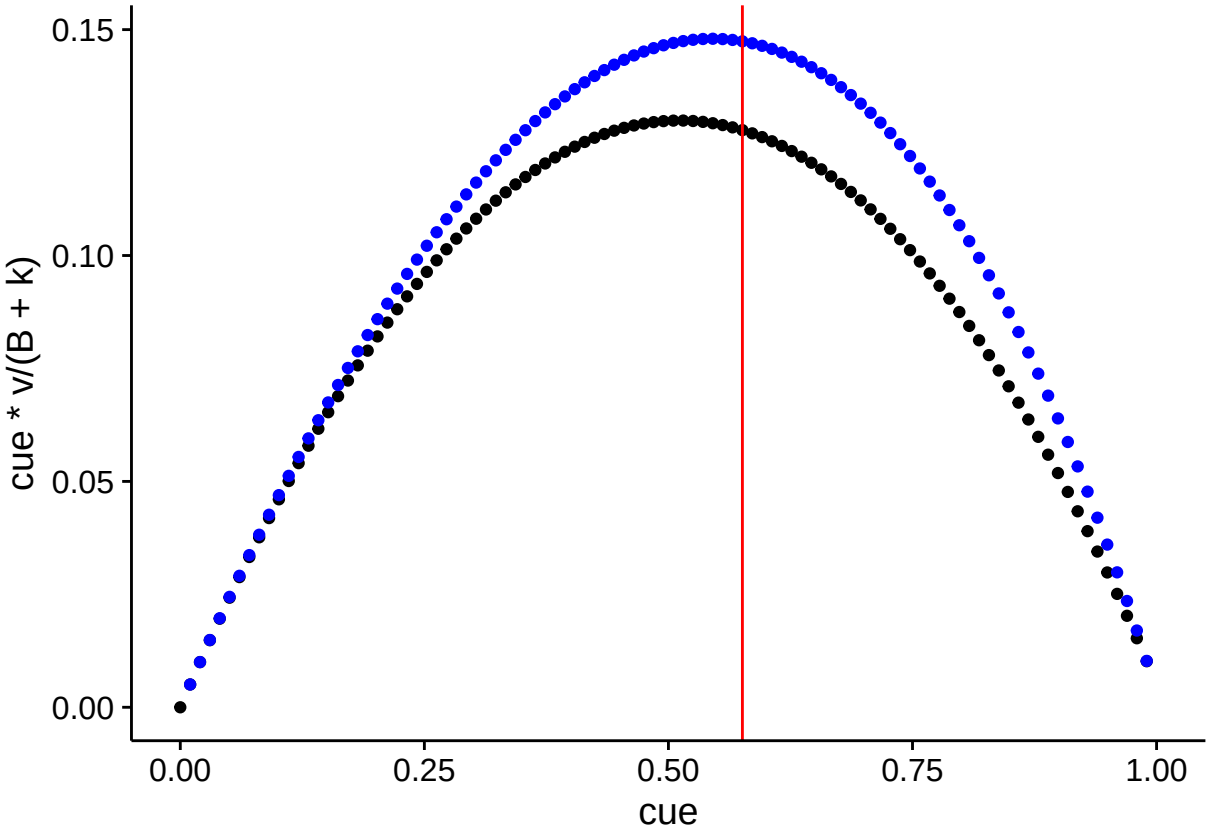



```
test <- subset(numParm.df, index==2)
tradeoffs.df <- data.frame(cue=seq(0, 1, length=100),
                  v=cue_v_tradeoff(b=test$b, vmax=test$vmax, cue=seq(0, 1, length=100)),
                  k=v_k_tradeoff(kmin=test$kmin, a=test$a,
```

```
                                  v=cue_v_tradeoff(b=test$b, vmax=test$vmax,
                                                   cue=seq(0, 1, length=100))))

tradeoffs.df <- ddply(tradeoffs.df, c('cue', 'v', 'k'), function(xx){
  ans1 <- as.data.frame(steadyState(test$b, test$vmax, test$kmin, test$a, test$h, test$m, test$I, xx$cu
  ans2 <- as.data.frame(steadyState_compete(test$b, test$vmax, test$kmin, test$a, test$h, test$m, test$
  names(ans2) <- sprintf('%s_compete', names(ans2))
  return(cbind(ans1, ans2))
})
ggplot(tradeoffs.df) + geom_point(aes(x=cue,y=cue*v/(B+k))) +
 geom_point(aes(x=cue,y=cue*v/(B_n_compete+k)), color='blue') +
  geom_vline(xintercept=test$cue_strat, color='red')
```

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

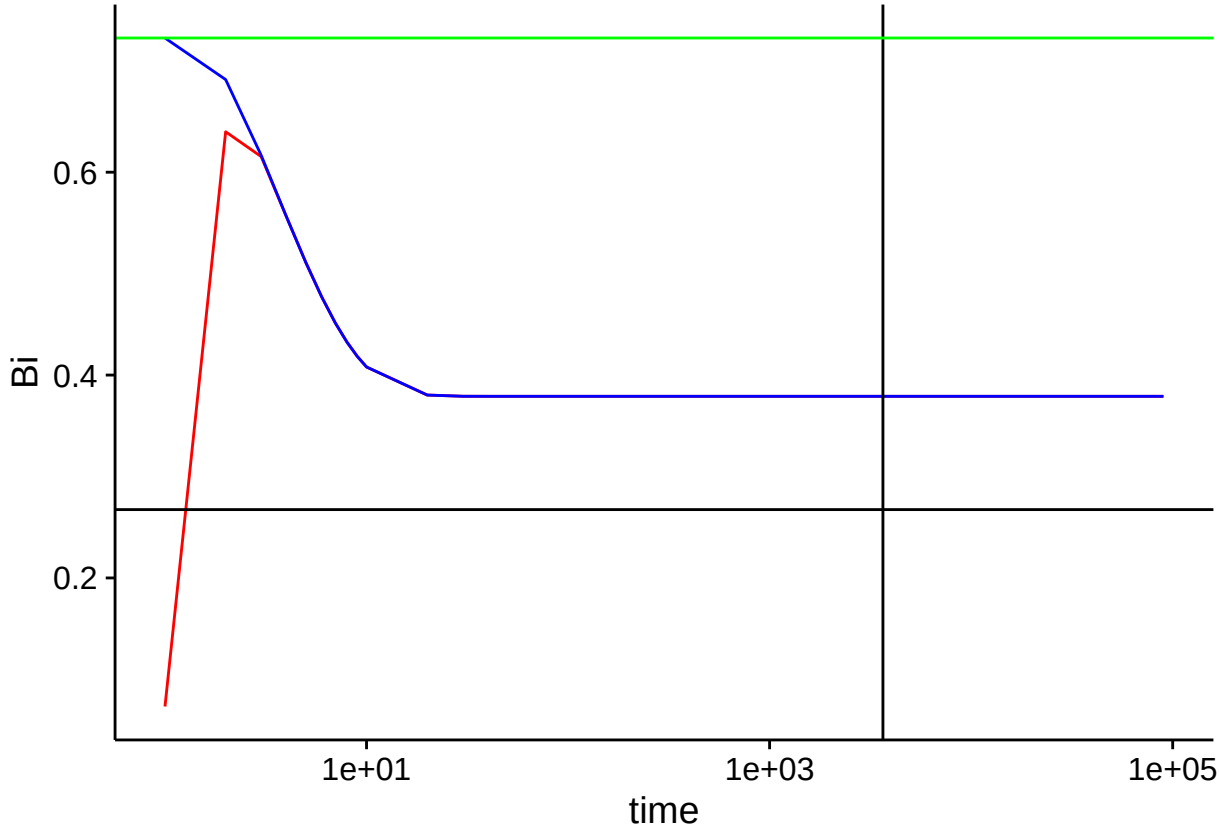
```
## Warning: Removed 2 rows containing missing values (geom_point).
```




```
steadyState_compete(b=test$b, vmax=test$vmax, kmin=test$kmin, a=test$a, h=test$h, m=test$m,
                    I=test$I, cue_n=test$cue_strat, cue_i=test$cue_strat+0.02)
```

```
## $C
## [1] 68.06053
##
## $B_i
## [1] 0.2717566
##
## $B_n
## [1] 0.2739377
```

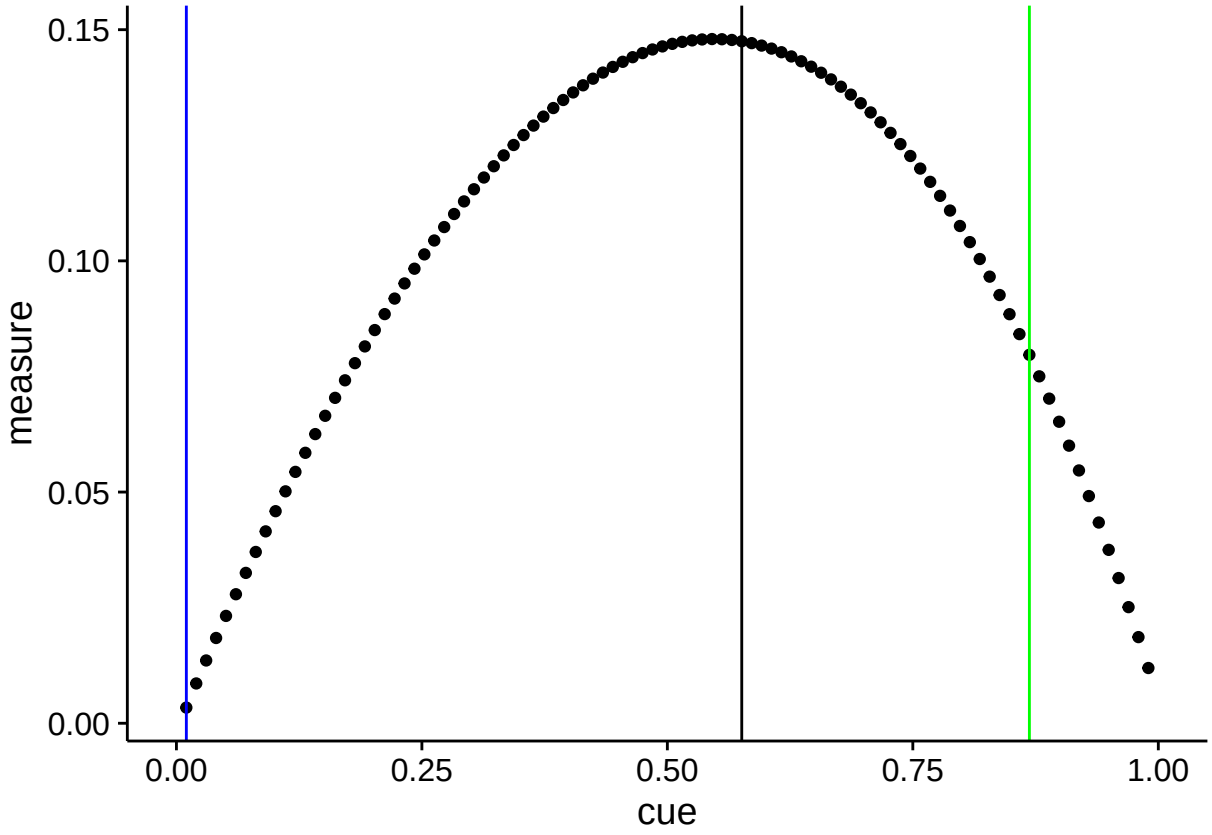
```
test <- subset(numParm.df, index ==2)
y0 <- list(C=test$C_opt, Bn=test$B_opt, Bi=test$B_opt*0.1)
comboParm <- c(test[1, c('b', 'vmax', 'kmin', 'a', 'm', 'I')],
               cuen=test$cue_opt,
               cuei=test$cue_opt,
               hn=test$h,
               hi=test$h)
comboParm <- c(comboParm,
               vn=cue_v_tradeoff(test$b, test$vmax, cue=comboParm$cuen),
               vi=cue_v_tradeoff(test$b, test$vmax, cue=comboParm$cuei))
comboParm <- c(comboParm,
               kn=v_k_tradeoff(test$kmin, test$a, v=comboParm$vn),
               ki=v_k_tradeoff(test$kmin, test$a, v=comboParm$vi))
carbonPools <- lsoda(y=unlist(y0), times=c(1:9, (1:9)*10, (1:9)*100, (1:9)*1e4),
                                   func=compitition.model, parms=comboParm)

ggplot(as.data.frame(carbonPools)) +
  geom_line(aes(x=time, y=Bi), color='red') +
  geom_line(aes(x=time, y=Bn), color='blue') +
  geom_vline(xintercept=10*365) + geom_hline(yintercept=test$B_strat) + geom_hline(yintercept=test$B_op
  scale_x_log10()
```



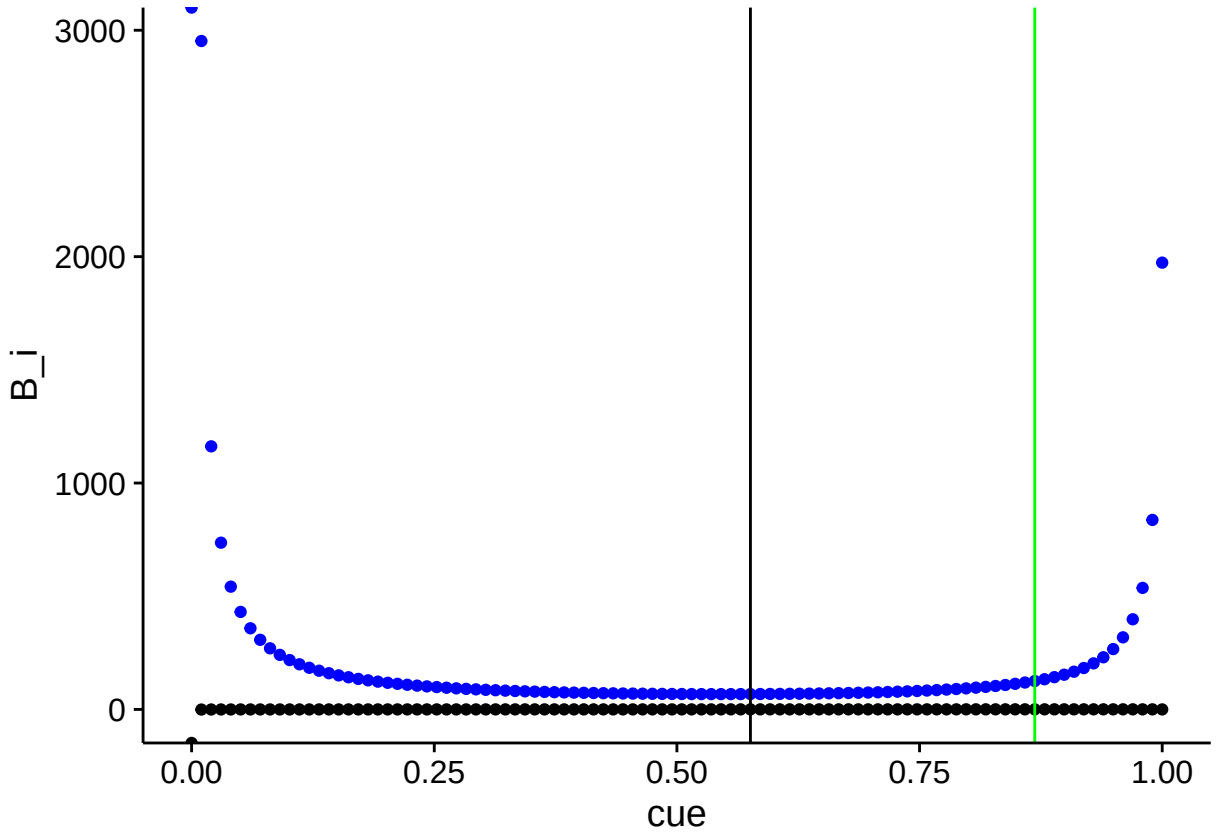

```
temp <- with(test, {
  cue <- seq(0, 1, length=100)
  v <- cue_v_tradeoff(b=b, vmax=vmax, cue)
  k <- v_k_tradeoff(kmin=kmin, a=a, v=v)
```

```
  ans <- steadyState_compete(b, vmax, kmin, a, h, m, I, cue_i=cue-0.005, cue_n=cue)

  measure <- cue*v/(k+ans$B_n)
  ans <- data.frame(cue, v, k, ans, measure)
  return(ans)
})

ggplot(temp) + geom_point(aes(x=cue, y=measure)) +
  geom_vline(xintercept=test$cue_strat) +
  geom_vline(xintercept=test$cue_opt, color='green') +
  geom_vline(aes(xintercept=cue[which.max(B_n)]), color='blue')
```

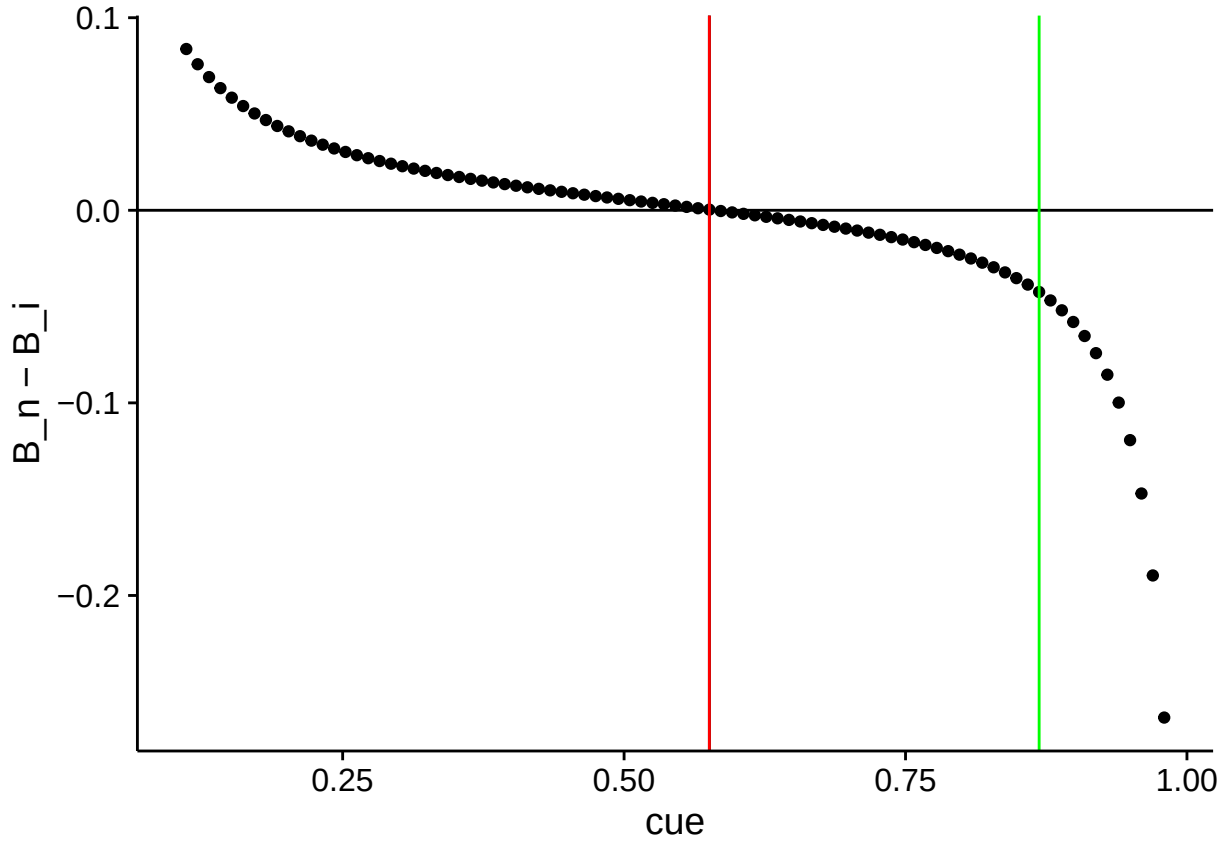
```
## Warning: Removed 2 rows containing missing values (geom_point).
```



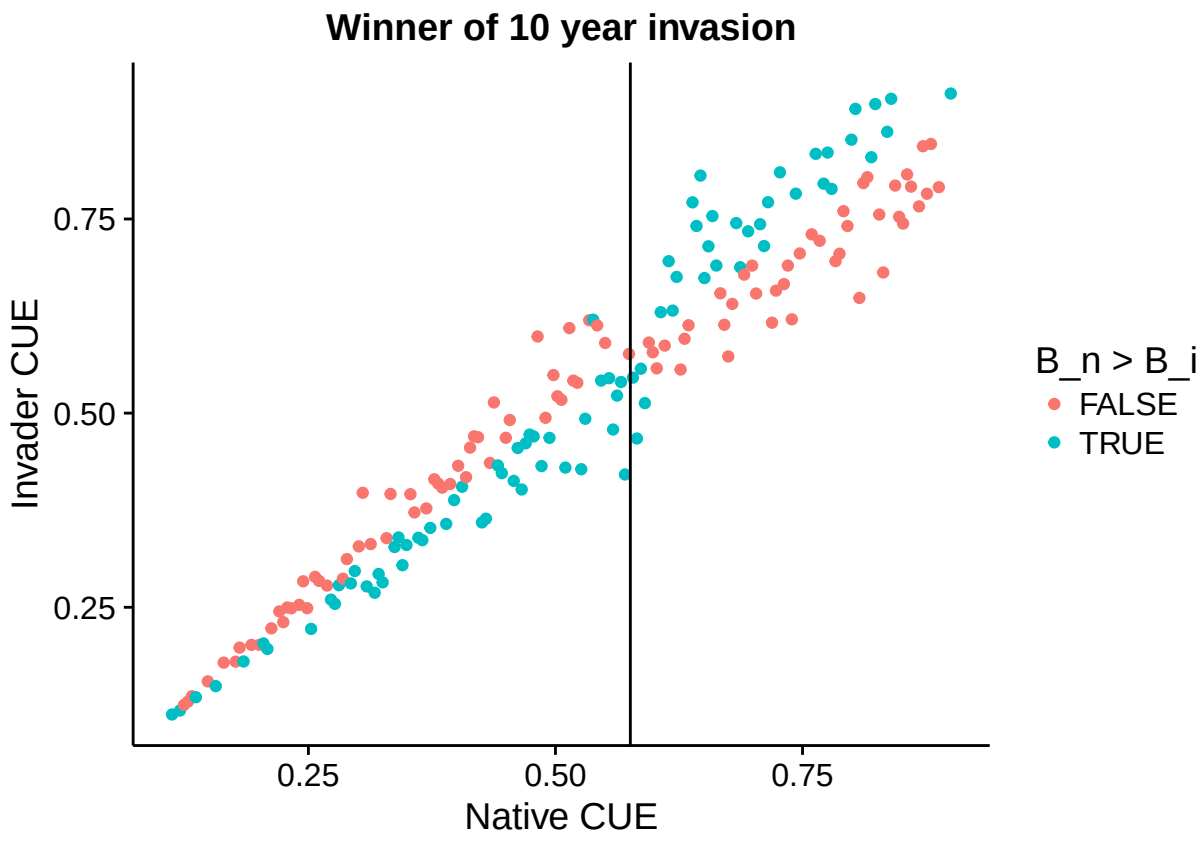

```
ggplot(temp) + geom_point(aes(x=cue, y=B_i)) +
  geom_point(aes(x=cue, y=C), color='blue') +
  geom_vline(xintercept=test$cue_strat) +
  geom_vline(xintercept=test$cue_opt, color='green')
```




```
ggplot(subset(temp, B_n > 0 & B_i > 0)) + geom_point(aes(x=cue, y=B_n-B_i)) +
  geom_hline(yintercept=0) + geom_vline(aes(xintercept=cue[which.min(abs(B_n-B_i))])) +
  geom_vline(xintercept=test$cue_strat, color='red') +
  geom_vline(xintercept=test$cue_opt, color='green')
```




```
ggplot(subset(merge(runInvasions_ss, numParm.df),
              index==test$index & B_n > 0 & B_i > 0 & C > 0)) +
  geom_point(aes(x=cue_n, y=cue_i, color=B_n > B_i)) +
  geom_vline(data=subset(temp, B_n > 0 & B_i > 0),
             aes(xintercept=cue[which.min(abs(B_n-B_i))])) +
  labs(x='Native CUE', y='Invader CUE', title='Winner of 10 year invasion')
```


Winner of 10 year invasion
Invader CUE
0.75
0.50
0.25
0.25
0.50
0.75
Native CUE
B_n > B_i
FALSE
TRUE