# Rogers random predator equation: extensions and estimation by numerical integration

Ben Bolker

April 19, 2012

## 1 Introduction

Unlike the Holling type II functional response, which predicts the instantaneous attack rate, the Rogers random predator (RRP) equation (which allows for predator handling time and prey depletion over time) does not have a simple closed-form solution. Rogers (1972) gives a simple iterative solution, which is fast and effective but doesn't necessarily fit into standard maximum likelihood optimization approaches (Juliano, 1993; Vonesh and Bolker, 2005). However, as known in some mathematical circles (Corless et al., 1996) and as pointed out by McCoy and Bolker (2008), there is a "special function" called the Lambert $W$ function that can be used to compute the solution, making it a convenient plug-in for routines to fit parameters for the Rogers equation.

Various people have written me with questions about implementing variants of the Rogers equation:

- Amy Brooks wrote to ask if I had an analogous solution for the Rogers equation with *two* prey (non-interacting other than through shared predation), as discussed in (Colton, 1987).
- Ulrich Brose wrote to ask about the Beddington-DeAngelis functional response model, $\frac{dN}{dt} = -\frac{aNPT}{1+ahN+c(P-1)}$
- Owen Petchey wrote to ask about an alternative form of the attack rate, $a = bN^q$
- Adrian Stier has been working on a problem with a background mortality (density-independent depletion, i.e. $\frac{dN}{dt} = -N(\mu_0 + aP/(1+ahN))$.
- ?? has a variant with density-dependent attack rate

Unfortunately, it seems that the derivation of both Rogers' iterative scheme and the Lambert $W$ function form of the solution are fairly specific to the instantaneous Holling type II functional form, and don't (as far as I have been able to figure out) generalize to other situations easily. (The original Rogers paper is pretty dense: I have looked through it but have *not* sat down and worked through it in detail, which might be required in order to really understand whether/how it can be generalized.) Therefore, in general one has to resort to brute force (numerical integration) to solve these problems. (The exception is Ulrich Brose's problem, which it turns out can be rescaled to be equivalent to the RRP.)

Below, I develop various simple (??) bits of code to "solve" these variants of the RRP by brute force, and compare them with the results of an even more general/brute-force approach, i.e. numerically integrating the population dynamics.

## 2 Derivation of Lambert $W$ form for RRP equation for a single predator

(This doesn't really need to go here, but I wanted to write it down somewhere ...)

$$\begin{aligned}
N &= N_0\left(1-e^{-a(T-hN)}\right)\\
1-N/N_0 &= e^{-a(T-hN)}\\
&= e^{-aT}\cdot e^{ahN}\\
&= e^{-a(T-hN_0)}\cdot e^{ahN_0\left(\frac{N}{N_0}-1\right)}\\
ahN_0(1-N/N_0)\cdot e^{ahN_0\left(1-\frac{N}{N_0}\right)} &= ahN_0e^{-a(T-hN_0)}\\
ahN_0(1-N/N_0) &= W\left(ahN_0e^{-a(T-hN_0)}\right)\\
N &= N_0-\frac{W\left(ahN_0e^{-a(T-hN_0)}\right)}{ah}
\end{aligned} \tag{1}$$

Or starting from scratch, we have $dN/dt = -aN/(1+ahN) = -A(N)$ so we compute $\int -dN/A(N)$ and equate it to $T$. In Mathematica:

```
m := Integrate[-(1 + a1 h1 Np)/(a1 Np), Np]
```

The answer (indefinite integral), as we could have figured out ourselves, is $-\log(N)/a - hN$. To figure out the number eaten,

```
Solve[-h1 (Np - N0) - (Log[Np] - Log[N0])/a1 == T, Np]
```

This gives

```
Np -> ProductLog[a1 E^(a1 h1 N0 - a1 T) h1 N0]/(a1 h1)
```

which is equivalent to our answer above (since `ProductLog` is Mathematica's version of the Lambert $W$), and we are computing the number eaten here (not the number surviving).

Don't think we can solve this for the two-prey case ... (maybe, but I haven't tried):

$$N_1 = N_{0,1}\left(1 - e^{-a_1(T-h_1N_1-h_2N_2)}\right) \quad (2)$$

$$N_2 = N_{0,2}\left(1 - e^{-a_2(T-h_1N_1-h_2N_2)}\right) \quad (3)$$

# 3 Two-prey case

## 3.1 Numerical integration

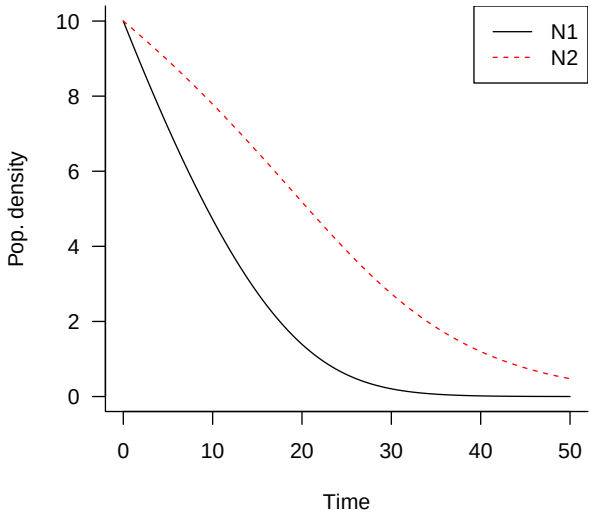
Solve by brute force numerical integration: at each moment assume the depletion rate follows the Holling type II expectation ($dN_i/dt = a_iN_i/(1 + \sum_j a_jh_jN_j)$).

Load the `deSolve` package and define the gradient function:

```
library(deSolve)
frgrad <- function(t,y,parms) {
  with(c(as.list(parms),as.list(y)),
       list(c(-a1*N1/(1+a1*h1*N1+a2*h2*N2),
              -a2*N2/(1+a1*h1*N1+a2*h2*N2)),
            NULL))
}
```

A simple example showing the time dynamics:

```
L1 <- lsoda(y=c(N1=10,N2=10),times=seq(0,50,by=0.1),
            parms=c(a1=0.3,a2=0.1,h1=1,h2=1),
            func=frgrad)
```




A function to compute the number eaten by time $T$:

```
rogers.lsoda <- function(N10,N20,a1,a2,h1,h2,T) {
  L1 <- lsoda(y=c(N1=N10,N2=N20),times=seq(0,T,length=2),
              parms=c(a1=a1,a2=a2,h1=h1,h2=h2),
              func=frgrad)
  c(N10,N20)-L1[2,-1]
}
```

## 3.2 Solution by optimization

It should be faster (although perhaps not enough to matter?) to solve the Rogers 2-prey equations (3) by numerical optimization rather than integrating all the way through.

It turns out that the solution is more robust if one (1) picks reasonably good starting points (I use the expectation from the Holling type II function without depletion); (2) uses bounded optimization to make sure the number eaten is between 0 and the starting number available; (3) defines a function to calculate the gradient.

The most straightforward to solve (3) would be to use Newton's method. It is marginally more convenient in R to use the built-in optimization tools to minimize $(\sum(\hat{N}_j - N_j)^2)$ — the squared deviation of the solution from the desired value.

Define sum-of-squares function, a gradient function, and an initial-estimate function:

```
rogers.dev <- function(p,N10,N20,a1,a2,h1,h2,T,
                       debug=FALSE) {
  N1 <- p[1]
  N2 <- p[2]
  x <- T-h1*N1-h2*N2
  expected <- c(N10,N20)*(1-exp(-c(a1,a2)*x))
  s <- sum((expected-p)^2)
  if(debug)cat(N1,N2,x,expected,p,s,"\n")
  s
}

## gradient
rogers.devgr <- function(p,N10,N20,a1,a2,h1,h2,T,
                         debug=FALSE) {
  N1 <- p[1]
  N2 <- p[2]
  x <- T-h1*N1-h2*N2
  expected <- c(N10,N20)*(1-exp(-c(a1,a2)*x))
  if (debug) cat(expected,"\n")
  ## deriv of (expected N_i wrt N_i)
  dNx = -c(N10,N20)*exp(-c(a1,a2)*x)*c(a1,a2)*c(h1,h2)
  ## deriv of (expected N_i wrt N_j)
  dNy = -c(N10,N20)*exp(-c(a1,a2)*x)*c(a1,a2)*c(h2,h1)
  dev = (expected-c(N1,N2))
  2*(dev*(-1+dNx)+rev(dev)*rev(dNy))
}

rogers.init <- function(N10,N20,a1,a2,h1,h2,T) {
  T*c(a1*N10,a2*N20)/(1+a1*h1*N10+a2*h2*N20)
}
```

A function to return the number eaten:

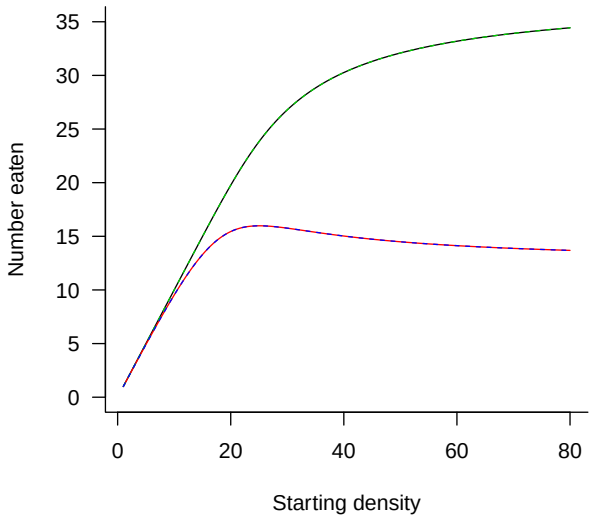
```
rogers.opt <- function(N10,N20,a1,a2,h1,h2,T,
                       start,debug=FALSE) {
  if (missing(start))
    start = rogers.init(N10,N20,a1,a2,h1,h2,T)
  if (debug)cat("start",start,"\n")
  O1 <- optim(fn=rogers.dev,
              gr=rogers.devgr,
              par=start,
              N10=N10,N20=N20,a1=a1,a2=a2,h1=h1,h2=h2,T=T,
              method="L-BFGS-B",upper=c(N10,N20),lower=c(0,0),
              debug=debug)
  if (O1$convergence==0) O1$par else stop("convergence failed")
```

```
}
```

Some basic tests:

```
rogers.lsoda(N10=10,N20=10,a1=0.3,a2=0.1,h1=1,h2=1,T=50)

##    N1    N2
## 9.999 9.525

rogers.opt(N10=10,N20=10,a1=0.3,a2=0.1,h1=1,h2=1,T=50)

## [1] 9.999 9.525
```

Try it for $a = \{0.3, 0.1\}$, $h = \{1, 1\}$, $T = 50$, and equal starting numbers from 1 to 80.




The optimization and `lsoda` results are overlaid — they're exactly the same. The "bump" in the less-attacked species is mildly interesting — I guess increasing the density of the more-attacked species is actually lowering the effective attack rate.

One could then plug these functions into a maximum-likelihood estimator (insert blatant plug for my `bbmle` package here. With one species one can do something like

```
mle2(eaten~dbinom(prob=rogers(init,a,h,T)/init,size=init))
```

For two species, one would either have to write out an explicit negative log-likelihood function, something like

```
rogers2nlik = function(N01,N02,a1,a2,h1,h2,T,N1,N2) {
  expected = rogers.opt(N01,N02,a1,a2,h1,h2,T)
  init = c(N01,N02)
  -sum(dbinom(c(N1,N2),prob=expected/init,size=init,log=TRUE))
}
```

(however, n.b. that this function is *not* properly vectorized — one would have to tweak it a bit so that it would deal with the whole data set at once, e.g. something like:

```
rogers2nlik = function(N01,N02,a1,a2,h1,h2,T,N1,N2) {
  expected = t(mapply(rogers.opt,
    N01,N02,
    MoreArgs=list(a1=a1,a2=a2,h1=h1,h2=h2,T=T)))
  -sum(dbinom(N1,prob=expected[,1]/N01,size=N01,log=TRUE))-
    sum(dbinom(N2,prob=expected[,2]/N02,size=N02,log=TRUE))
}
```

n.b. this isn't tested either! It's just a rough guide to what one would need to do.

Then:

```
mle2(minuslogl=rogers2nlik,start=...,...)
```

Slightly more cleverly, one could probably write a function that would allow use of the formula interface — there is an example like this buried, but not published, in Chapter 8 of my book (search `http://www.zoo.ufl.edu/bolker/emdbook/chap8.Rnw` for `dicweib`, or look in the `tests` directory of the source version of the `bbmle` package — yes, I know these are both pretty obscure).

# 4 Beddington-DeAngelis

I had a query from Ulrich Brose about using the Beddington-DeAngelis functional response model,

$$\frac{dN}{dt} = \frac{aNPT}{1 + ahN + c(P-1)},$$

which incorporates a predator interference term into the Holling type II; when $c = 0$ (or $P = 1$), it reduces to the type II. (Brose et al used slightly different notation in their message to me; here, changing things slightly from the previous case, I have also included $P$ explicitly — previously, it could essentially be included in $T$.)

In their message to me, they had derived the following integrated version:

$$N = N_0(1 - \exp((aNh - aPT)/(1 + c(P-1))))$$

(which I initially didn't believe but confirmed by brute-force testing: see below).

What I realized, *after* writing much of the code below, is that this model can be written as a re-parameterization of the Holling type II where the attack rate depends on $P$. If we define $\phi = 1 + c(P-1)$, we have:

$$\begin{aligned}\frac{dN}{dt} &= \frac{aNPT}{(1+c(P-1))+ahN} \\ &= \frac{aNPT}{\phi + ahN} \\ &= \frac{(a/\phi)NPT}{1+(a/\phi)hN}\end{aligned}$$

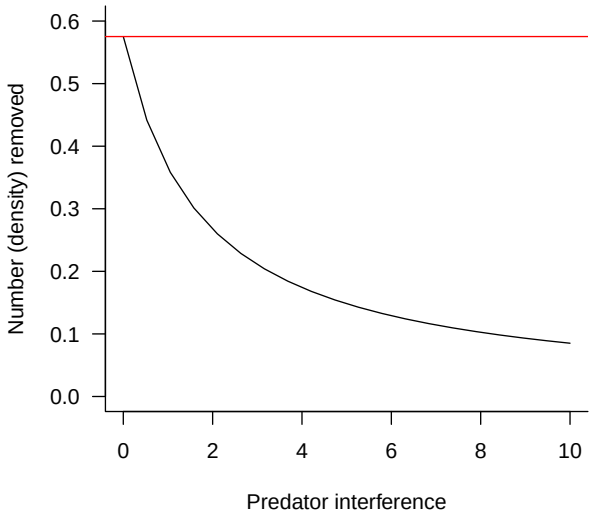
For an experiment with a single value of $P$, $a$ and $c$ can't be separated; for varying $P$, we can just reparameterize (see below).

Code to solve the ODE by brute force:

```
BDAfr <- function(N,P,a,h,c) {
  a*N*P/(1+a*h*N+c*(P-1))
}
gfun <- function(t,y,parms) {
  with(as.list(c(y,parms)),
       list(c(N=-BDAfr(N,P,a,h,c)),NULL))
}
ffun <- function(N0,P,a,h,T,c) {
  r <- lsoda(c(N=N0),times=c(0,T),parms=c(a=a,h=h,P=P,c=c),
        func=gfun)
  N0-r[2,2]
}
```

Check that we recover the Rogers answer when $c = 0$ (and/or $P = 1$):

```
library(emdbook)

## Loading required package: MASS
## Loading required package: lattice

rogers.pred <- function(N0,a,h,P,T) {
  N0 - lambertW(a*h*N0*exp(-a*(P*T-h*N0)))/(a*h)
}
## checks
rogers.pred(N0=1,a=0.5,h=0.5,P=2,T=2)-
  ffun(N0=1,a=0.5,h=0.5,P=2,T=2,c=0)

##          N
## 1.477e-08
```




Check the ODE integration against an iterative approach:

```
rr2 <- function(N0,a,h,P,T,c,maxit=200,tol=1e-4) {
  Np <- N <- N0/2
  it <- 1
  repeat {
    Np <- N
    N <- N0*(1-exp((a*N*h-a*P*T)/(1+c*(P-1))))
    it <- it +1
    if (it==maxit || abs(Np-N) < tol) break
  }
  if (it==maxit) stop("reached maxit without convergence")
  N
}
ffun(N0=1,P=2,a=0.5,h=0.5,T=2,c=4)-
  rr2(N0=1,P=2,a=0.5,h=0.5,T=2,c=4,tol=1e-8)

##           N
## -2.119e-06

## OK, these are the same
```

But in this case the Lambert W approach should work too:

```
bda.pred <- function(N0,a,h,T,P,c) {
  a = a/(1+c*(P-1))
  N0 - lambertW(a*h*N0*exp(-a*(P*T-h*N0)))/(a*h)
}
```

```
ffun(N0=1,P=2,a=0.5,h=0.5,T=2,c=4)-
  bda.pred(N0=1,P=2,a=0.5,h=0.5,T=2,c=4)

##           N
## -2.119e-06
```

Somewhat to my surprise, the iterative approach is actually *fastest* (for this particular test case). Don't know how much speed, robustness, etc. vary across parameter space??

```
system.time(replicate(1000,ffun(N0=1,P=2,a=0.5,h=0.5,T=2,c=4)))

##    user  system elapsed
##   1.920   0.000   1.836

system.time(replicate(1000,rr2(N0=1,P=2,a=0.5,h=0.5,T=2,c=4)))

##    user  system elapsed
##   0.048   0.000   0.043


system.time(replicate(1000,bda.pred(N0=1,P=2,a=0.5,h=0.5,T=2,c=4)))

##    user  system elapsed
##   0.212   0.000   0.197
```

# 5 Power-law attack

Suppose (as in Owen Petchey's problem) that the attack rate in the absence of handling time constraints is not linear in prey density, but a power law of prey density: $a(N) = bN^q$ (rather than $= aN$ in the usual case).

Owen wanted to know if one could get away with substituting $a(N)$ into the Lambert $W$ equation. Unfortunately not.

Define an attack-rate function:

```
attack <- function(b,q,N) {
  b*N^q
}
```

And a modified RRP:

```
fun1 <- function(b,q,h,P,T,N0) {
  A <- attack(b,q,N0)
  N0-lambertW(A*h*N0*exp(-A*(P*T-h*N0)))/(A*h)
}
```

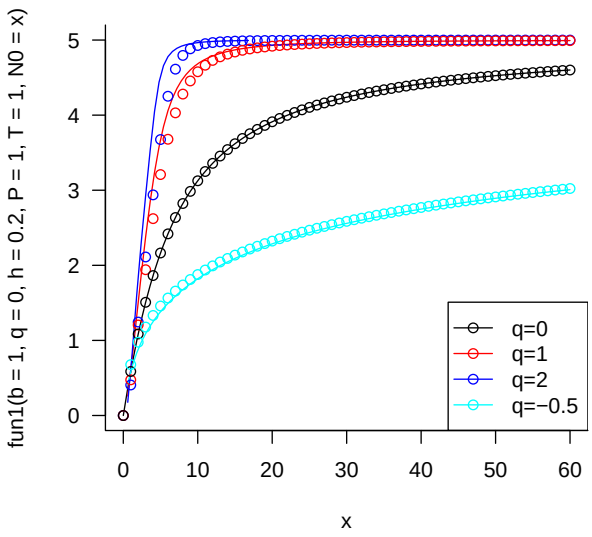
Try this function with a series of different $q$ values, and with an explicit ODE solution to double-check:

```
gradfun <- function(t,y,parms) {
  with(as.list(c(y,parms)),
       { A  <- attack(b,q,N)
         grad <- -A*N/(1+A*h*N)
         list(grad,NULL)
       })
}

fun2 <- function(b,q,h,P,T,N0) {
  L <- lsoda(c(N=N0),
             times=c(0,T),
             func=gradfun,
             parms=c(b=b,q=q,h=h,P=P))
  N.final <- L[2,2]
  N0-N.final
}
```

Quick test shows the answers are different:

```
fun1(b=1,q=1,h=0.2,P=1,T=1,N0=30)

## [1] 4.97

fun2(b=1,q=1,h=0.2,P=1,T=1,N0=30)

##     N
## 4.967
```

Curve shows (wrong) Lambert $W$ solution, points show numerical integration solution:




Iterative solution? I tried to check this, but for some reason it's not working. Fix it later ...

```
fun3 <- function(b,q,h,P,T,N0,maxit=40,Nstart=A*N0/(1+h*N0)) {
  A <- attack(b,q,N0)
  N <- Nstart
  diff <- 1000
  it <- 1
  while (diff>0.0001 && it < maxit) {
    e <- (N*h-P*T)
    N <- N0*(1-exp(A*e))
    it <- it + 1
    cat(it,N,"\n")
  }
  N
}
f6 = fun2(b=1,q=0,h=0.2,P=1,T=1,N0=30)
```

# 6 Density-independent depletion

This example (due to Adrian Stier) uses

$$\frac{dN}{dt} = -N(\mu_0 + aP/(1+ahN)).$$

I don't have the example ready to go here. It was slow/finicky to do the fitting, especially if one wants to compute likelihood profiles etc.. Some thoughts on this are listed below.

# 7 Density-dependent attack rate

Suppose we model attack rate as $(d+bN_0)/(1+cN_0)$ (need to figure out where this is from, it's not entirely consistent that the attack rate depends only on *initial* density: shouldn't that be subject to depletion too, so that what we really have is the solution to $dN/dt = -(a(N(t))/(1+a(N(t))hN(t))\,dt$? Nevertheless, let's forge ahead:

```
predfun <- function(b,c,d,h,T,N0,debug=FALSE) {
  a <- (d+b*N0)/(1+c*N0)
  r <- N0 - (1/(a*h))*lambertW(a*h*N0*exp(a*(h*N0-T)))
  if (debug) cat(mean(a),b,c,d,h,mean(r),"\n")
  r
}
dlW <- function(x) plogis(-lambertW(x))
```

# 8 Multiple predators with different attack rates

Rate of consumption at time $t$ = instantaneous consumption from both predators $= -dN/dt = C(t) = a_1N/(1+a_1h_1N) + a_2N/(1+a_2h_2N)$ (wrote $N(t)$ as $N$ for convenience here).

Number eaten by time $T$ is $\int_0^T (-dN/dt)\,dt$, but this is more easily done by just figuring out $N(0) - N(T)$, (i.e. getting number surviving), so we need to integrate both sides of

$$\int \frac{dN}{a_1N/(1+a_1h_1N) + a_2N/(1+a_2h_2N)} = T + C$$

If we were just to do this for RRP we would have $\int 1/(a_1N) + h_1\,dN = \log N/a_1 + h_1N = T + C$ (so we can see immediately where the Lambert $W$ stuff has to come from).

If we try to do this in Mathematica, we start by defining the attack rate, as above:

```
Aa[Np] := a1 Np /(1 + a1 h1 Np) + a2 Np/(1 + a2 h2 Np)
```

Integrate $-dN/N$:

```
m := Integrate[-1/Aa[Np], Np]
```

```
-(a1 a2 (a1 + a2) h1 h2 (h1 + h2) Np +
 a1 a2 (h1 + h2)^2 Log[Np] +
  (a1 h1 - a2 h2)^2 Log[a2 + a1 (1 + a2 (h1 + h2) Np)])/
       (a1 a2 (a1 + a2) (h1 + h2)^2)
```

(This could also have been done by a good student in first semester calculus, although the algebra is a little more tedious ...)

Unfortunately Mathematica can't solve for $N$ in this case.

We can solve this by brute force, though:

```
f1 <- function(N,a1,a2,h1,h2) {
  -(a1*a2*(a1+a2)* h1*h2* (h1+h2)*N +
    a1*a2*(h1+h2)^2*log(N)+
    (a1*h1 - a2*h2)^2* log(a2+a1*(1+a2*(h1+h2)*N)))/
        (a1*a2*(a1+a2)*(h1+h2)^2)
}
f2 <- function(N,N0,a1,a2,h1,h2,T) {
  f1(N,a1,a2,h1,h2)-f1(N0,a1,a2,h1,h2)-T
}
f3 <- function(N0,a1,a2,h1,h2,T) {

N0-uniroot(f2,interval=c(0,N0),N0=N0,a1=a1,a2=a2,h1=h1,h2=h2,T=T)$root
}
```

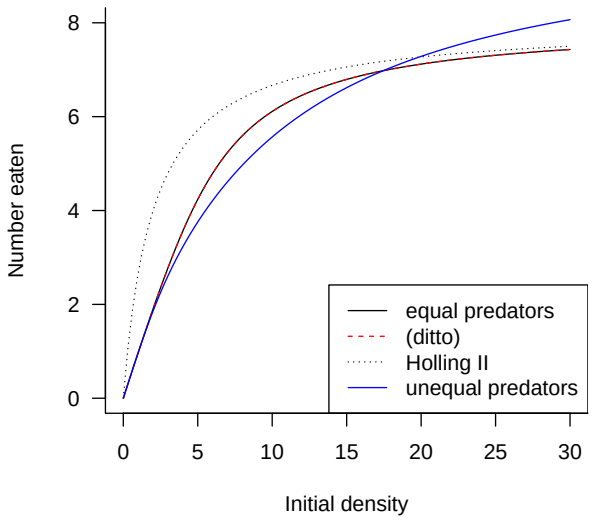
The results should be identical if we compare RRP with two predators (of the same species) with $a$ and $h$ vs this new formulation with $a_1 = a_2$ and $h_1 = h_2$.

**This approach should be applicable much more generally than the Lambert $W$ solution and generally much faster and more robust (though requiring slightly more hand/user calculation) than numerical integration ...**

Some haphazard examples:

```
rogers.pred(N0=1,a=0.5,h=0.5,P=2,T=2)

## [1] 0.8333

f3(N0=1,a1=0.5,a2=0.5,h1=0.5,h2=0.5,T=2)

## [1] 0.8333

rogers.pred(N0=1,a=1,h=0.5,P=2,T=2)

## [1] 0.9702

f3(N0=1,a1=1,a2=1,h1=0.5,h2=0.5,T=2)

## [1] 0.9702
```

Draw the curve for RRP and the equivalent, then try two predators with $a = \{1, 2\}$ and $h = \{1, 0.25\}$:




# 9 More thoughts on optimization

A problem from the r-help mailing list (definition of `dat` hidden):

Naive Nelder-Mead fit starting from (arbitrary??) starting values:

```
library(bbmle)
tt <- try(r1 <- mle2(FR~dbinom(size=N0,
                      prob=rogers.pred(N0,a,h,T=24,P=1)/N0),
           start=list(a=1.5,h=0.04),
           method="Nelder-Mead",data=dat))
```

```
## Error in dbinom(x, size, prob, log) :
##   Non-numeric argument to mathematical function
```

This occurs because we end up trying a value in `lambertW` that gives complex results. In particular, running with `options(error=recover)` shows that we get to $a = -0.262$, $h = 1.75$:

```
rogers.pred(5,a=-0.262,h=1.75,P=1,T=24)

## [1] 12.39+5.46i
```

Works with `method="L-BFGS-B"`, as long as we set `lower` strictly *greater than* 0:

```
r2 <- mle2(FR~dbinom(size=N0,
                     prob=rogers.pred(N0,a=a,h=h,T=24,P=1)/N0),
           start=list(a=1.5,h=0.04),
           method="L-BFGS-B",lower=1e-5,data=dat)
```

Could we have done better starting from Holling parameters? We can fit a Holling model without specifying starting parameters by using a GLM with an inverse link (as also detailed here). If $p = a/(1 + ahN)$ then $1/p = 1/a + hN$, so:

```
r3 <-
glm(cbind(FR,N0-FR)~N0,family=binomial(link="inverse"),data=dat)
```

We can translate back to $\{a, h\}$ as follows:

```
startparams <- glmparams <-
with(as.list(coef(r3)),list(a=1/`(Intercept)`,h=N0))
startparams$a <- startparams$a/24  ## need to scale parameters
from /hour to /day
startparams$h <- startparams$h*24  ## need to scale parameters
from hours to days
```

```
r2B <- mle2(FR~dbinom(size=N0,
                      prob=rogers.pred(N0,a=a,h=h,T=24,P=1)/N0),
            start=startparams,method="Nelder-Mead",data=dat)
```

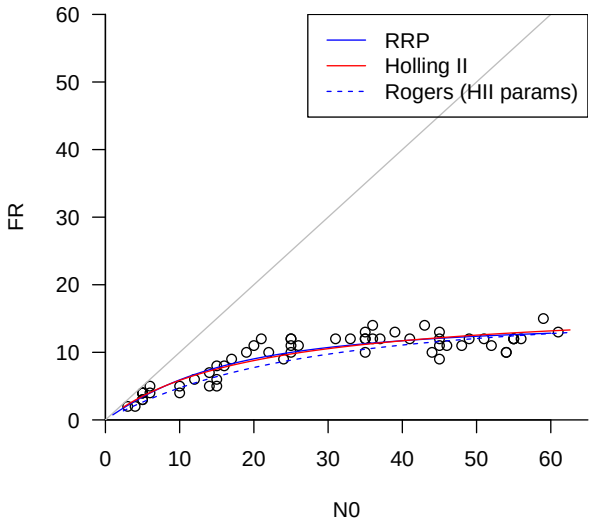
No, this fails again — even though the parameters are apparently on about the correct order of magnitude ...

```
##   holling.start     RRP
## a       0.03656 0.06074
## h       1.36593 1.56138
```

The overall RRP fit to data is slightly better, but not much better. In terms of AIC:

```
AICtab(r2,r3,weights=TRUE)

##    dAIC df weight
## r2 0.0  2  0.634
## r3 1.1  2  0.366
```

Or a picture:




The dashed line shows the predictions of the RRP model on the basis of the Holling parameters (i.e., the same instantaneous attack rate and handling time, applied for 24 hours, but with depletion). The RRP conclusion is that most of the observed limitation is due to depletion (and hence increased searching time), rather than to predator saturation ... Predator saturation (fraction of time spent handling rather than searching) at density $N$ is $1/(aN)/(1/(aN)+h) = 1/(1+ahN)$; for $N = 60$ this is 0.1495 for RRP vs. 0.2502 for Holling type II (yes, I could also compute confidence intervals for these quantities — but I claim that 15% vs. 25% is potentially ecologically important).

There are pretty big differences between the Holling (non-depletion) $a$ and $h$ estimates ...

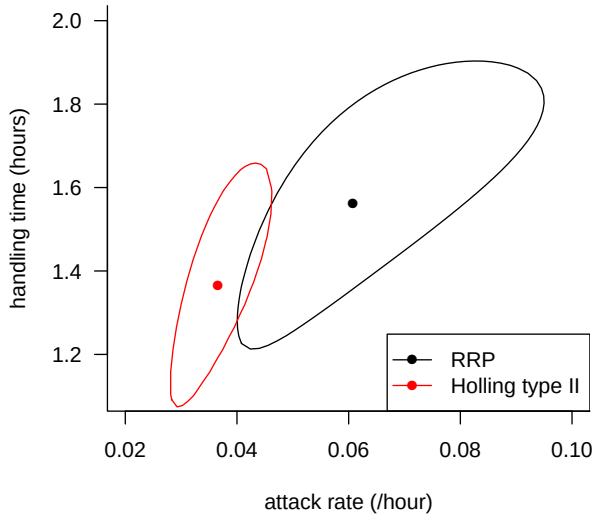
```
##     2.5 %  97.5 %
## a 0.02957 0.04433
## h 1.12988 1.60099
```

and the RRP ones ...

```
##     2.5 %  97.5 %
## a 0.04452 0.08419
## h 1.30630 1.81256
```

I think this is correct (although I'm still not 100% sure that I've scaled everything correctly in making the comparison, especially in the $N_0$ vs $FR$ plot).

TO DO: illustrate using the derivative of the Lambert $W$ to construct an analytical gradient for the RRP model — should make things even more robust/faster when used with `nlminb` or `L-BFGS-B`?




(Bivariate confidence intervals are somewhat wider than univariate ones, so the confidence regions do actually overlap in this case — but they may still be significantly different, as they don't overlap by much ...)

# 10 Future thoughts

- is there a back-of-the envelope calculation to decide whether depletion could be important? For example, fit Holling type II and then compare implied consumption rates at the beginning and end of the experiment? For the example above, for $N_0 = 60$ we would have an initial probability of 0.0091 per hour and a final probability of 0.0229 per hour (assuming $N \approx 12$ at the end of the experiment); this ratio is 2.5, enough (perhaps) to be concerned about ...
- one could certainly clean a lot of this up and hide the details to allow end-users to fit 2-predator data, or other more general RRP extensions, in a few lines of code (including loading a package or `source`ing a file full of code)
- `rogers.lsoda` is potentially more useful than `rogers.opt`, even though it is marginally less efficient (0.578 seconds vs. 0.16 to do the calculations above), because it can more easily be generalized to other situations (interference, trait-mediated interactions, etc.)

- Juliano and Williams (1987) simulated data with which to test (single-prey) Rogers equation fitting techniques in a more sophisticated way, by allowing a log-normal distribution of handling times and an exponential distribution of attack times. It would be interesting (although perhaps not worthwhile in terms of making a huge difference to the estimates) to work out a statistical method that actually took this process error into account ...
- in general, could improve calculation of second derivatives by (1) using parscale/ changing to log scale, or (??) trying `numDeriv`
- is there an analogue/extension of Rogers RP equation for multiple predators with varying attack rates and handling times? If so, we could cheat and pretend that the baseline mortality rate is due to an additional predator with a handling time of zero ... (i.e. density-ind. mortality) ... but I don't think there is ...
- we can construct confidence intervals for the curves etc. by bootstrapping, but (1) this will be really slow with our current code and (2) it's not optimal because there aren't very many replicates (3?) within treatment/density combination in the current Stier experimental design (3?)
- power tools: try AD Model Builder? Can't see an easy way to do this with WinBUGS ... might be able to use MCMCpack to get a posterior distribution ...
- Does the RRP have an inverse? It's not impossible — in which case we could use it as a custom link in a GLM ...

# References


Colton, T. F. 1987. Extending functional response models to include a second prey type: An experimental test. Ecology **68**:900–912. URL `http://links.jstor.org/sici?sici=0012-9658%28198708%2968%3A4%3C900%3AEFRMTI%3E2.0.CO%3B2-I`.

Corless, R. M., G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth. 1996. On the Lambert $W$ function. Advances in Computational Mathematics **5**:329–359.

Juliano, S. A. 1993. Nonlinear curve fitting: predation and functional response curves. Pages 159–182. *in* S. M. Scheiner and J. Gurevitch, editors. Design and analysis of ecological experiments. Chapman & Hall, New York.

Juliano, S. A. and F. M. Williams. 1987. A comparison of methods for estimating the functional response parameters of the random predator equation. Journal of Animal Ecology **56**:641–653.

McCoy, M. W. and B. M. Bolker. 2008. Trait-mediated interactions: influence of prey size, density and experience. Journal of Animal Ecology **77**:478–486. URL `http://www.ingentaconnect.com/content/bsc/janim/2008/00000077/00000003/art00007`.

Rogers, D. J. 1972. Random search and insect population models. Journal of Animal Ecology **41**:369–383.

Vonesh, J. R. and B. M. Bolker. 2005. Compensatory larval responses shift trade-offs associated with predator-induced hatching plasticity. Ecology **86**:1580–1591.