Computer exercise in parameter estimation and stochastic population dynamics

Mikael Pontarp, Biology, Lund

Modelling Biological Systems, BIOS13

The data

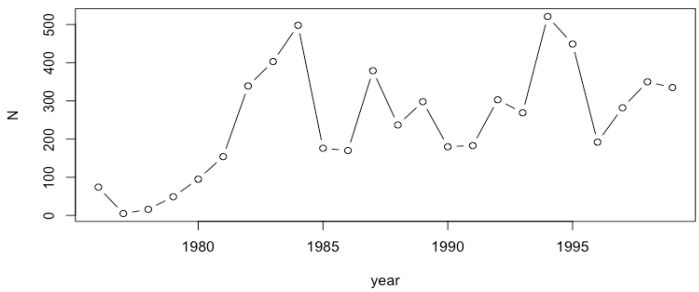
We shall analyse a time series of the ringlet butterfly (*Aphantopus hyperantus*, luktgräsfjäril, Brauner Waldvogel). Some facts: The ringlets have one generation per year. They spend the winter as larvae. The larvae feed on grass before they pupate in early summer.

The original data consists of weekly transect counts at a particular site somewhere in Great Britain. Here you are presented yearly totals of the sightings of ringlets, i.e. the total number of ringlets sighted during a whole season. The data can only be regarded as a population index, somehow reflecting the actual density of ringlet butterflies. We will nonetheless *ignore measurement errors* and take the data for granted, assuming it is an accurate measure of actual population densities.

Model fitting

The data, Ringlets.csv, can be downloaded from L@L. Next, read it into R, and plot it:

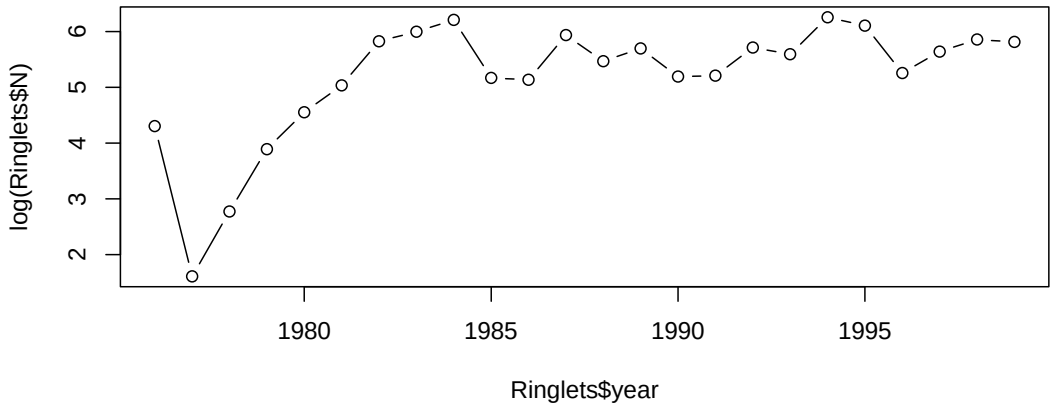
```
> Ringlets <- read.csv('Ringlets.csv')
> Ringlets
   year   N
1  1976  74
2  1977   5
3  1978  16
4  1979  49
...

> plot(Ringlets, type='b')
```



Note the dramatic decline right in the beginning, between 1976 and 1977. It looks even more dramatic on a log-scale. Check it!

How do we check that on a log-scale? Well, one simple way is this:

```
> plot( Ringlets$year,  log(Ringlets$N), type='b' )
```



In 1976 there was a *severe* drought in Britain. We will return to this rare event later. For the moment, it is sufficient to note that something extreme happened. After the sharp decline in 1976-1977, the population recovered and seems to have settled around a population size around 300.

Our task here is to fit a suitable model to this data, and later use that model to predict the future of the population. The standard procedure in model-fitting is to compare some observed entity with the model prediction, and then minimize the error (the residuals), i.e. the difference between observed and predicted, by adjusting the model parameters. In a standard linear regression we predict the 'y-values' as a linear function of the 'x-values' (whatever *x* and *y* are) . The question here is what to predict. There are several possibilities.

If we want to fit the Ricker model, the model in itself predicts population size:

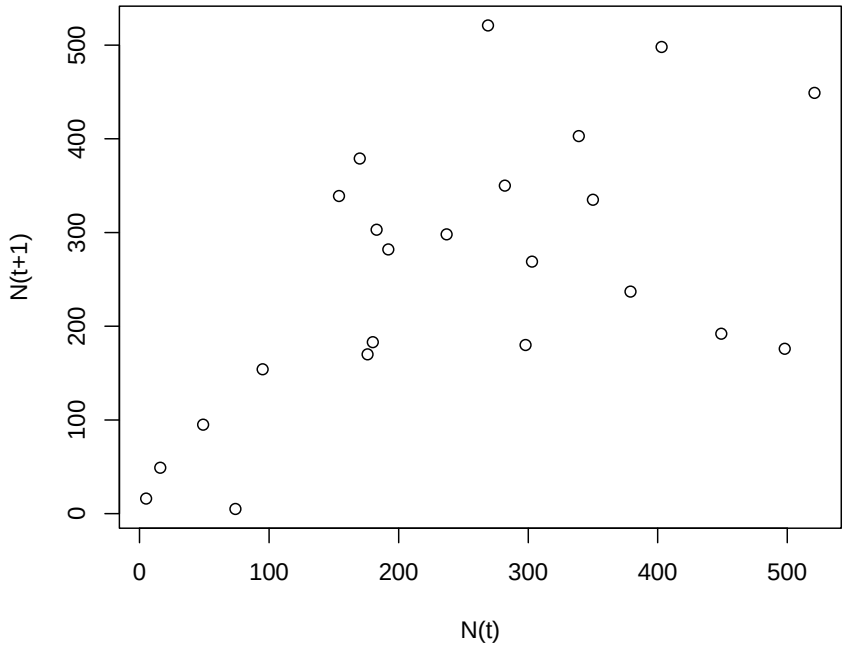
$$\widehat{N}_{t+1} = N_t e^{r_0\left(1-\frac{N_t}{K}\right)}$$

(The hat ^ indicates a predicted value) With this approach, we can not predict the first population size, N_1, but all the rest (N_2 - N_{24}). In other words, we use population sizes $N_1 - N_{23}$ to predict N_2 - N_{24}. To simplify the further treatment, let's create two vectors with the corresponding population sizes:

```
> Nt <- Ringlets$N[1:23]
> Ntplus1 <- Ringlets$N[2:24]
```

And perhaps plot them against each other:

```
> plot(Nt, Ntplus1, xlab='N(t)', ylab='N(t+1)')
```

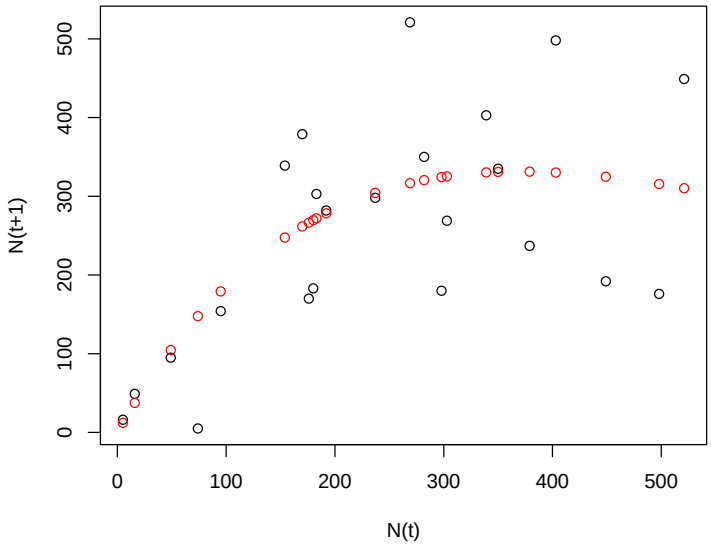


Does it look like a Ricker function? Hard to tell. In any case, there's a lot of noise, which we assume is 'environmental stochasticity'. We can now use the nls (Non-linear Least Squares) function in R to estimate the r_0 and K parameters:

```
> Rickerfit <- nls( Ntplus1 ~ Nt*exp(r0*(1-Nt/K)), data=list(Nt=Nt),
start=list(r0=1,K=100) )
> summary(Rickerfit)

Formula: Ntplus1 ~ Nt * exp(r0 * (1 - Nt/K))

Parameters:
   Estimate Std. Error t value Pr(>|t|)
r0   0.8913     0.2212   4.029 0.000606 ***
K  329.3594    29.7616  11.067 3.19e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 103.9 on 21 degrees of freedom
...
```

We can also check the predicted values with our data, by adding them to the previous plot:

```
> points(Nt, fitted(Rickerfit), col='red')
```

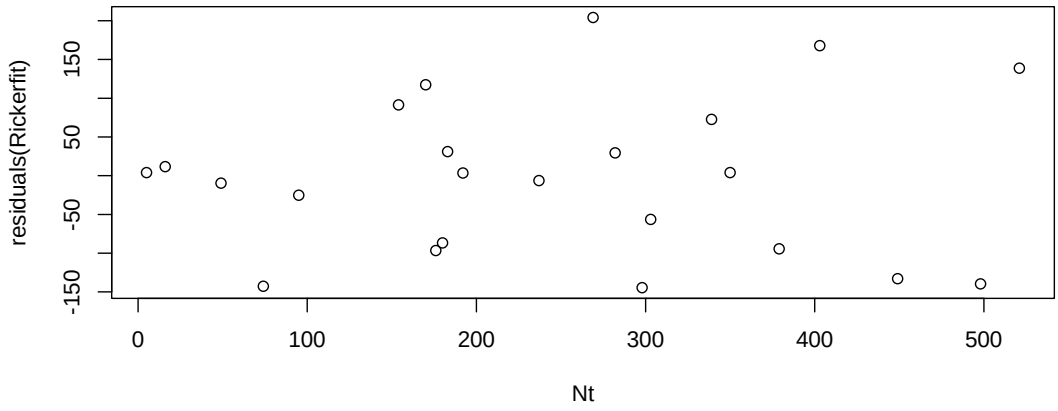


Does it look ok?

Well…

It may look ok, but maybe not entirely so. For instance a plot of the residuals against population size shows an alarming dependence:

```
> plot(Nt, residuals(Rickerfit))
```

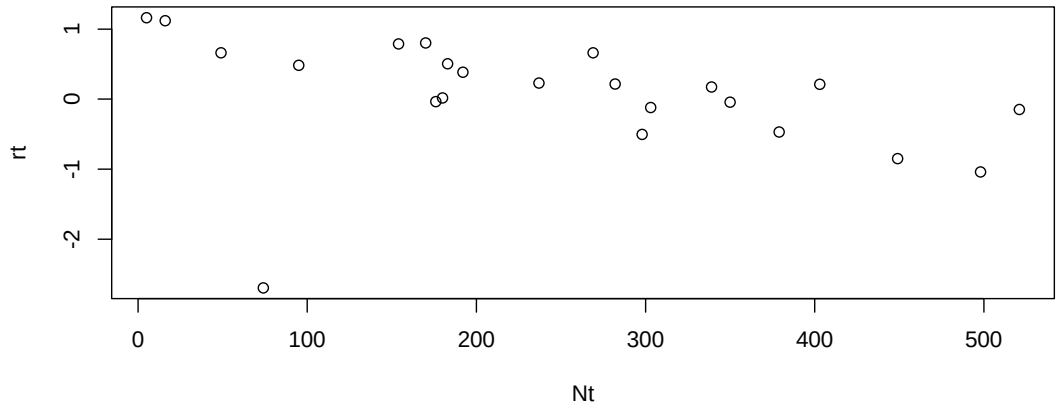


It seems as if the magnitude of the residuals increases with population size (which is quite natural, if you think about it). More importantly, *we have a specific idea on the origin of all that noise*. It is assumed to be environmental stochasticity, which by definition affects all individuals equally. It should thus somehow affect *per capita* growth rates, and not the growth of the entire population directly. The parameter fit above was carried out by minimizing the residuals in terms of predicted population

size. What we should do is to let the model predict per capita growth rates instead, putting the residuals (corresponding to environmental effects) in the right place. We will moreover work on a log scale, which from experience (and some theory) works best in this case.

Calculate the log population growth rate, $r_t = \ln(\frac{N_{t+1}}{N_t})$ as a new variable and plot it against the predictor Nt:

```
> rt <- log(Ntplus1/Nt)
> plot(Nt,rt)
```



According to the Ricker equation, this should be a linear relationship:

$$\ln(\frac{N_{t+1}}{N_t}) = r_0\left(1 - \frac{N_t}{K}\right)$$

Does the r_t plot look like a straight line? It does and it doesn't. There is one outlier that stand out more now than ever before. it is the first year growth:

```
> rt[1]
[1] -2.694627
```

This corresponds to the severe drought in 1976, mentioned above. Just looking at the r_t-plot it is clear that any model will have a hard time to predict that event. We will here (partly for simplicity) choose to eliminate that point from the data-set, and perhaps return to it later on. Thus: remove the year 1976 data-point and recalculate `Nt`, `Ntplus1` and `rt`! Also remake the plots if you wish.

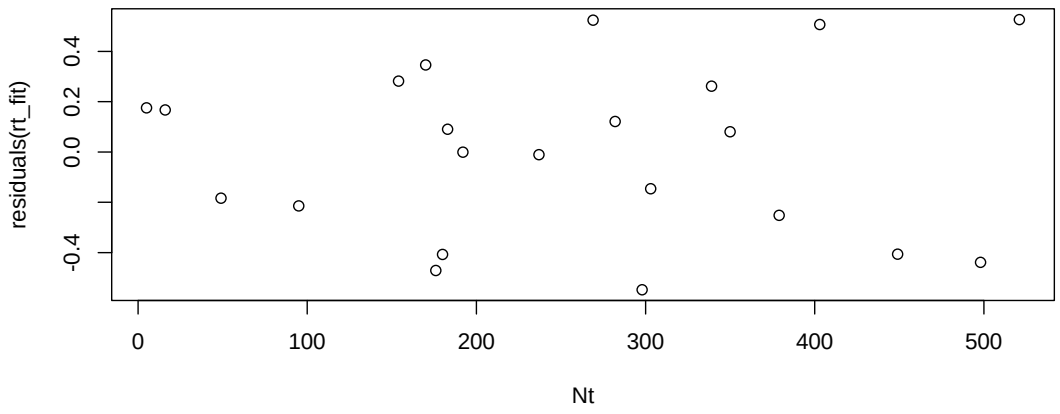
```
> Nt <- Ringlets$N[2:23]
> Ntplus1 <- Ringlets$N[3:24]
> rt <- log(Ntplus1/Nt)
```

We can now use `nls` again to fit the Ricker, but this time predicting r_t:

```
> rt_fit <- nls( rt ~ r0*(1-Nt/K), data=list(Nt=Nt), start=list(r0=1,K=100) )
> summary(rt_fit)

Formula: rt ~ r0 * (1 - Nt/K)

Parameters:
   Estimate Std. Error t value Pr(>|t|)
r0   1.0038     0.1486   6.753 1.44e-06 ***
K  311.4885    24.6677  12.627 5.50e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.3445 on 20 degrees of freedom
…
```

Do the predicted values look ok? And the residuals? Make the appropriate plots!

```
> plot(Nt,residuals(rt_fit))
```



At least it looks better now. There is no obvious increase in amplitude with N_t.

Note: We could have used a standard linear regression (`rt_fit <- lm( rt ~ Nt, data=list(Nt=Nt) )`), but then we would have to recalculate the values of r_0 and K from the standard regression parameters.

If we are happy with this model fit (well, I am) we can start *using* the parameters to run simulations. The model, including the environmental fluctuations, now looks like this:

$$N_{t+1} = N_t e^{r_0\left(1-\frac{N_t}{K}\right)+\epsilon_t},$$

where $\epsilon_t \in N(0,\sigma)$. The only parameter still missing is the σ. *However*, since the ϵ in the model corresponds exactly to the residual in the last model fit, we can use the residual standard error as a direct estimate of σ (make sure you find this number in the output):

$$\hat{\sigma} = 0.3445$$

You may want to check that this estimate is equal to $\sqrt{\frac{RSS}{n-k}}$, where RSS is the residual sum of squares, *n* is the number of residuals and *k* is the number of parameters:

```
> sqrt(sum(residuals(rt_fit)^2)/(length(rt)-2))
```

$\hat{r} = 1.004$
$\hat{K} = 311$
$\hat{s} = 0.3445$

An alternative model, model selection

It may be worthwhile to try a different model before we start simulating. The Hassel model looks like this:

$$N_{t+1} = \frac{\lambda N_t}{(1 + aN_t)^b}$$

It has three parameters: λ, a and b. Use the same method as above to fit these three parameters to the Ringlet data. Starting values can be chosen as `start=list(lambda=1.1,a=0.001,b=1).`

First we need to transform the prediction to a prediction of r_t instead of N_t:

$$r_t = \ln\left(\frac{N_{t+1}}{N_t}\right) = \ln\left(\frac{\frac{\lambda N_t}{(1 + aN_t)^b}}{N_t}\right) = \ln\left(\frac{\lambda}{(1 + aN_t)^b}\right)$$

Next, make the corresponding fit:

```
> Hassel_fit <- nls( rt ~ log(lambda/(1+a*Nt)^b), data=list(Nt=Nt),
start=list(lambda=1.1,a=0.001,b=1) )
> summary(Hassel_fit)

Formula: rt ~ log(lambda/(1 + a * Nt)^b)

Parameters:
       Estimate Std. Error t value Pr(>|t|)
lambda  2.896326   0.664457   4.359 0.000338 ***
a       0.000924   0.003043   0.304 0.764694
b       4.283423  11.646055   0.368 0.717088
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.3524 on 19 degrees of freedom
```

Which model is the best???

A model with more parameters easily generates a lower sum of squares. The Akaike Information Criterion (AIC) can be used to choose between models. R has a ready-made function for it:

```
> AIC(rt_fit)
```

Which model has the lowest AIC, i.e. which is the best according to the AIC?

```
> AIC(rt_fit)
[1] 19.45281
> AIC(Hassel_fit)
[1] 21.31359
```

The Ricker model has the lowest AIC, so it is the best!

Monte-Carlo simulations

Finally, let's run some simulations. Write an R-script that uses the parameter values of your chosen model and simulates Ringlet dynamics 100 years into the future, using the last known population size as a starting value. Plot the result. You may find use of 'Ricker_sim.R' at Live@Lund as a starting point.

Alter you script to instead run many (1000?) simulations, using the same parameter values and the same starting point for all. Plot the first 20 simulations in a single plot.

Find a way to estimate the risk of extinction, i.e. what proportion of your simulations went extinct? Assume a population goes extinct as soon as *N* drops below 2.

```
# This is the expanded and adjusted Ricker_sim.R
iterations <- 1000 # number of simulations
t_max <- 100 # length of simulation
r <- 1.004
K <- 311
sigma <- 0.3445
# empty plot:
plot(NA,type='n',xlim=c(0,t_max),ylim=c(0,1000))
extinct_count = 0
for( i in 1:iterations) {
 n <- rep(0,t_max)
 n[1] <- 10 # initial population size
 for( t in 1:(t_max-1)) {
        eps <- rnorm(1,0,sigma)
        n[t+1] <- n[t]*exp(r*(1-n[t]/K) + eps)
        # extinction?
        if ( n[t+1] < 2) {
                extinct_count <- extinct_count + 1
                break
        }
 }
 if (i<=20) {
        lines(n)
 }
}
cat('Extinction probability : ', extinct_count/iterations)
```

You may find these butterflies very hard to kill.

```
> source('Ricker_sim.R')
Extinction probability :  0
```

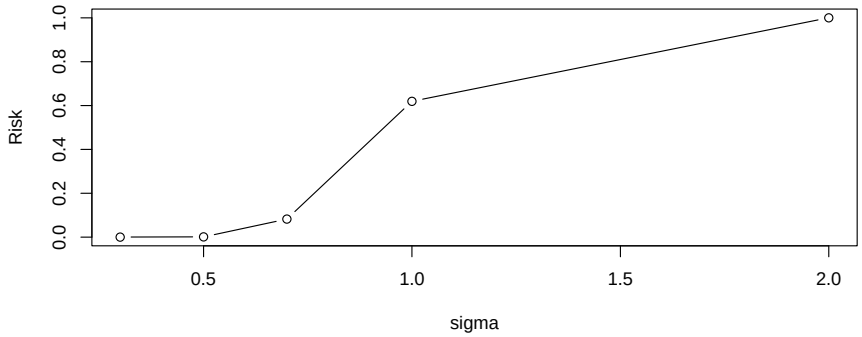
Hard to kill, yes.

Try to change the parameter values, to see if the extinction risk becomes more substantial (if you change the equilibrium population size it may be sensible to also change the starting population size).

Can you plot extinction risk as function of one or more of the parameters?

I ran the model with a few different values of σ (change the script, source it, note the risk of extinction, change the script agaian, a.s.o.) and then plotted the result like this:

```
> sigma <- c(0.3, 0.5, 0.7, 1, 2)
> Risk <- c(0, 0.001, 0.082, 0.619)
```



Of course, you can make a more elaborate loop of the whole thing if you want:

```
sigma <- seq(0,2,by=0.1)
Risk <- rep(0,length(sigma))
iterations <- 1000 # number of simulations
t_max <- 100 # length of simulation
r <- 1.004
K <- 311
for (s_i in 1:length(sigma)) {
 # empty plot:
 plot(NA,type='n',xlim=c(0,t_max),ylim=c(0,1000))
 extinct_count = 0
 for( i in 1:iterations) {
        n <- rep(0,t_max)
        n[1] <- 10 # initial population size
        for( t in 1:(t_max-1)) {
                eps <- rnorm(1,0,sigma[s_i])
                n[t+1] <- n[t]*exp(r*(1-n[t]/K) + eps)
                # extinction?
                if ( n[t+1] < 2) {
                        extinct_count <- extinct_count + 1
                        break
                }
        }
        if (i<=20) {
                lines(n)
        }
 }
 Risk[s_i] <- extinct_count/iterations
}

plot(sigma,Risk,type='b')
```