Looking at predictability and chaos of the logistic map

Owen Petchey

30 Apr 2015

Contents

Calculating the Lyapunov exponent from time series

First simulate two time series

Set the value of r, the number of iterations, the initial abundance of population 1, and the difference between initial abundance of population 1 and population 2:

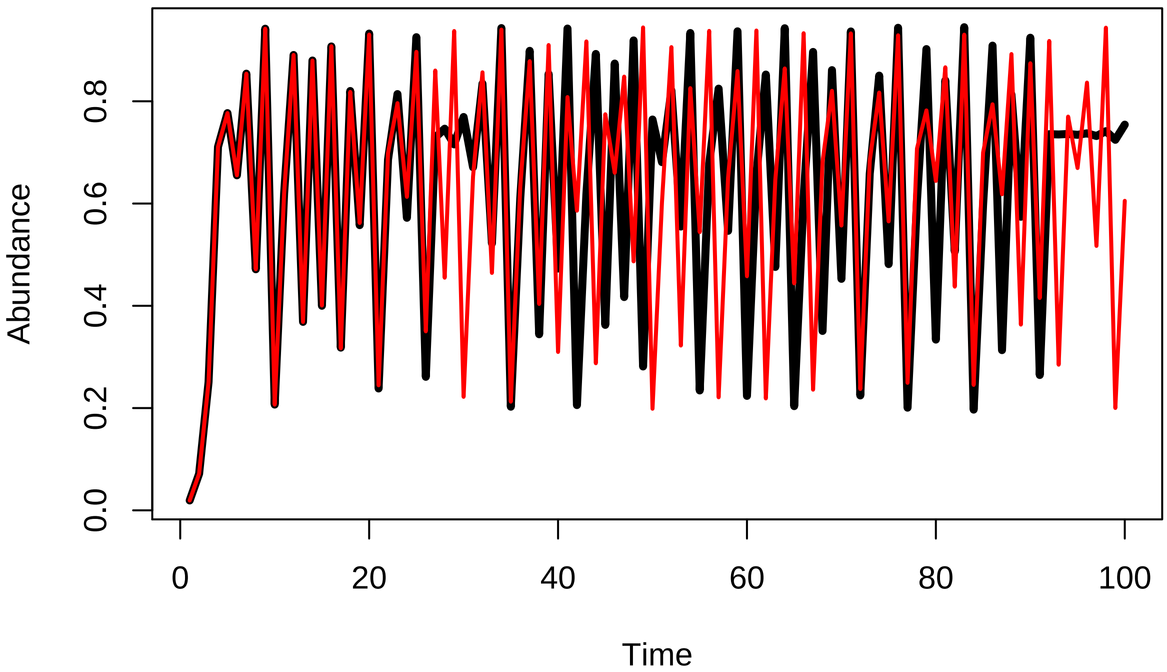
```
r <- 3.78
max_its <- 100
N0 <- runif(1, 0.001, 0.1)
N0.diff <- N0/100000
```

Simulate the logistic map:

```
N1 <- numeric(length=max_its)
N1[1] <- N0
N2 <- numeric(length=max_its)
N2[1] <- N0 + N0.diff
for(i in 2:max_its) {
  N1[i] <- r * N1[i-1] * (1 - N1[i-1])
  N2[i] <- r * N2[i-1] * (1 - N2[i-1])
}
```

The two series of population dynamics:

```
plot(1:max_its, N1, type="l",
     xlab="Time",
     ylab="Abundance",
     lwd=4)
lines(1:max_its, N2, type="l",
      lwd=2, col="red")
```



Estimate Lyapunov exponent from growth in difference

Get the absolute difference between the two time series:

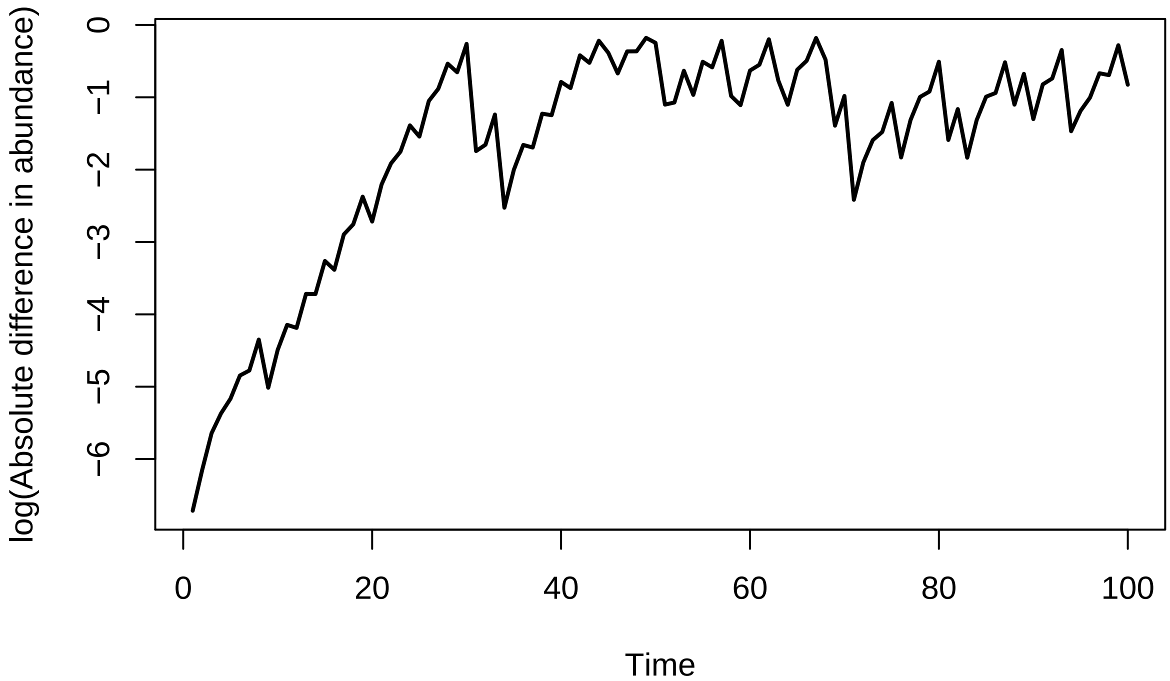
```
diffs <- abs(N1-N2)
```

And remove any zeros (including their positions from a vector of times), since these will cause problems when we need to log the difference:

```
times <- 1:length(diffs)
times <- times[diffs!=0]
diffs <- diffs[diffs!=0]
```

Plot the difference between the two time series through time:

```
  plot(1:max_its, log10(abs(N1-N2)), type="l",
       xlab="Time",
       ylab="log(Absolute difference in abundance)",
       lwd=2)
```

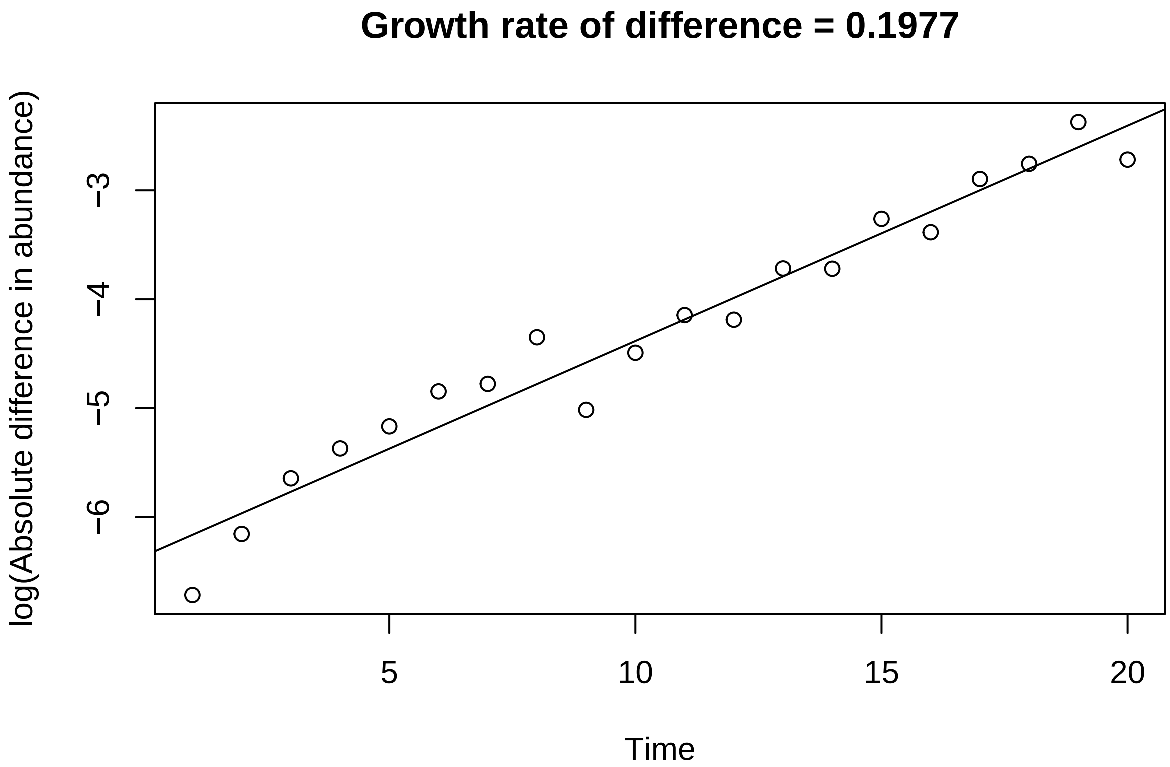


From this graph, select a range of times over which to calculate the growth rate of difference (be careful to choose a time range that contains data):

```
min.time <- 1 ## not less than 1 please
max.time <- 20
```

And plot this part of the data with the estimated Lyapunov exponent:

```
x <- min.time:max.time
y <- log10(abs(N1-N2))[min.time:max.time]
plot(x, y, type="p",
     xlab="Time",
     ylab="log(Absolute difference in abundance)",
     main=paste("Growth rate of difference =",  round(coef(lm(y~x))[2],4)))
abline(lm(y~x))
```




```
le0 <- coef(lm(y~x))[2]
```

Calculating the Lyapunov exponent by time-delayed embedding

This is the method uesd by Beninca et al. (Nature, 2008).

pdf of M. T. Rosenstein, J. J. Collins, C. J. De Luca, A practical method for calculating largest Lyapunov exponents from small data sets, Physica D 65, 117 (1993)

Do this for both of the time series made above. Owen guessed the parameter values below:

```
output1 <- lyap_k(N1, m=4, d=4, ref=4, t=4, s=40, eps=4)
```

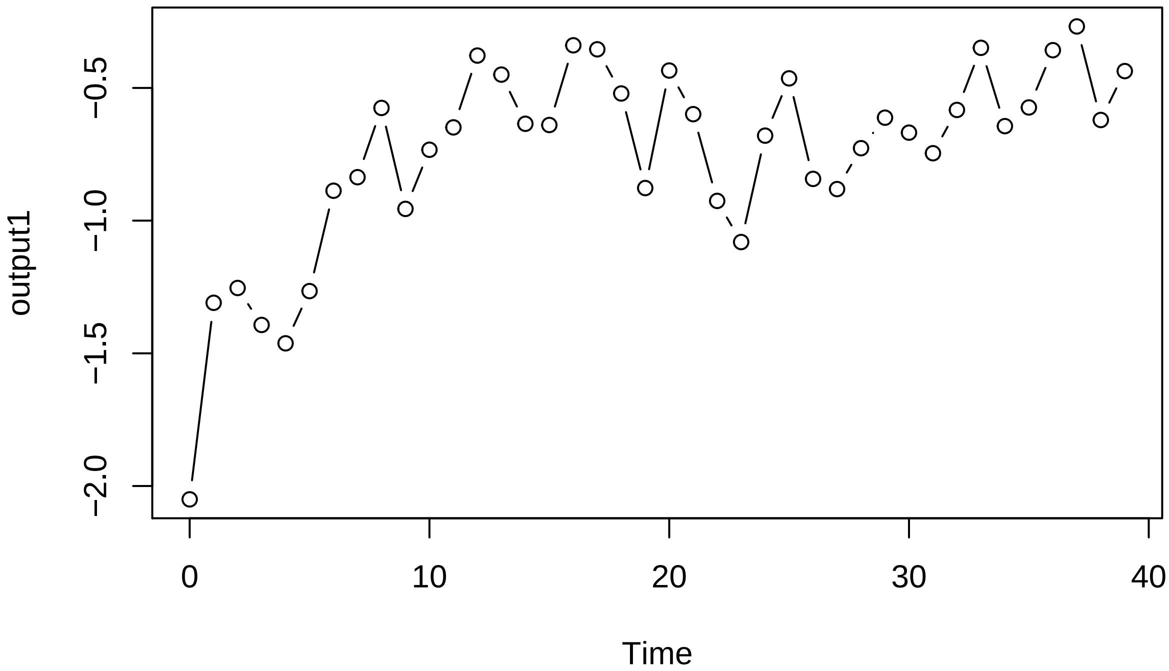
```
## Finding nearests
## Keeping  4  reference points
## Following points
```

```
output2 <- lyap_k(N2, m=4, d=4, ref=4, t=4, s=40, eps=4)
```

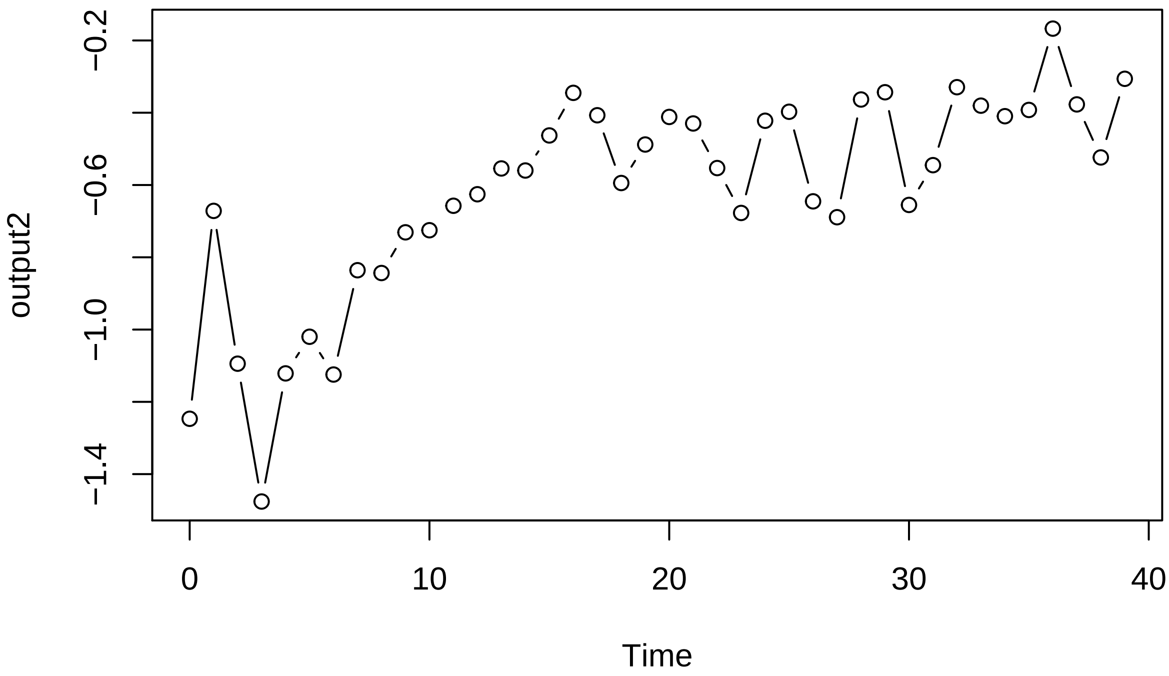
```
## Finding nearests
## Keeping  4  reference points
## Following points
```

Again need to carefully select the time range over which to calculate the Lyapunov exponent. Find it from these graphs

```r
plot(output1, type="b")
```




```r
plot(output2, type="b")
```



Select time range for each Lyapunov exponent (this is really fudgy at present!):

```r
start <- 4
end <- 8
le1 <- lyap(output1, start, end)[2]
```

Get the two Lyapunov exponents:

```
start <- 4
end <- 8
le2 <- lyap(output2, start, end)[2]
```

Analytical calculation of Lyapunov exponents

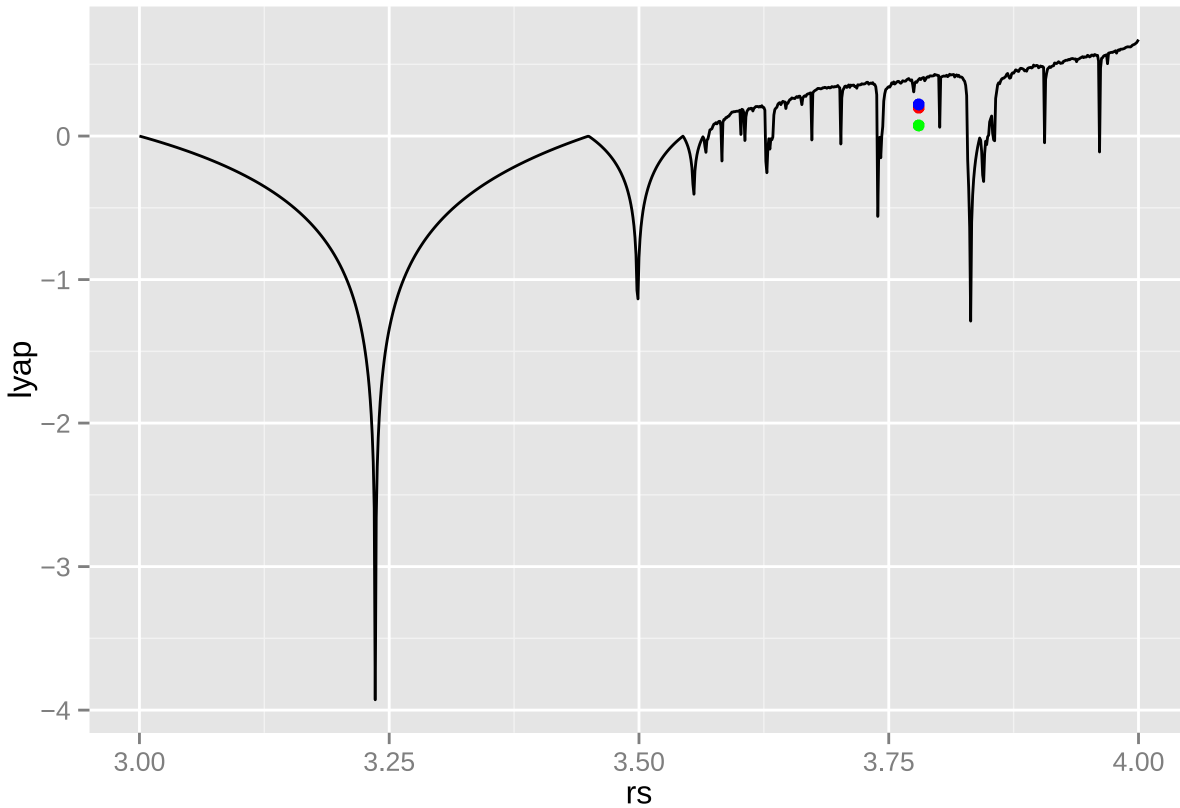
The following calculates the Lyapunov exponent from the value of r, assuming exponential growth (decline) in small differences between two time series, via some differentiation by the chain rule. The code is not evaluated to make this document; previous made data is loaded, for speed.

```
rs <- seq(3, 4, by=0.001)
lyap <- rep(NA, length(rs))
nits <- 10000
for(j in 1:length(rs)) {
  xn1 <- runif(1)
  lyp <- 0
  for(i in 1:nits) {
    xn <- xn1
    xn1 <- rs[j]*xn*(1-xn);
    if(i>300)
      ## The next line is the from some maths that Owen doesn't fully understand!
      lyp <- lyp + log(abs(rs[j]-2*rs[j]*xn1))
  }
  lyp <- lyp/nits
  lyap[j] <- lyp
}
write.csv(cbind(rs, lyap), "~/Desktop/rs_lyaps.csv")
```

Load the previously created data from github (code might need adapting for windows):

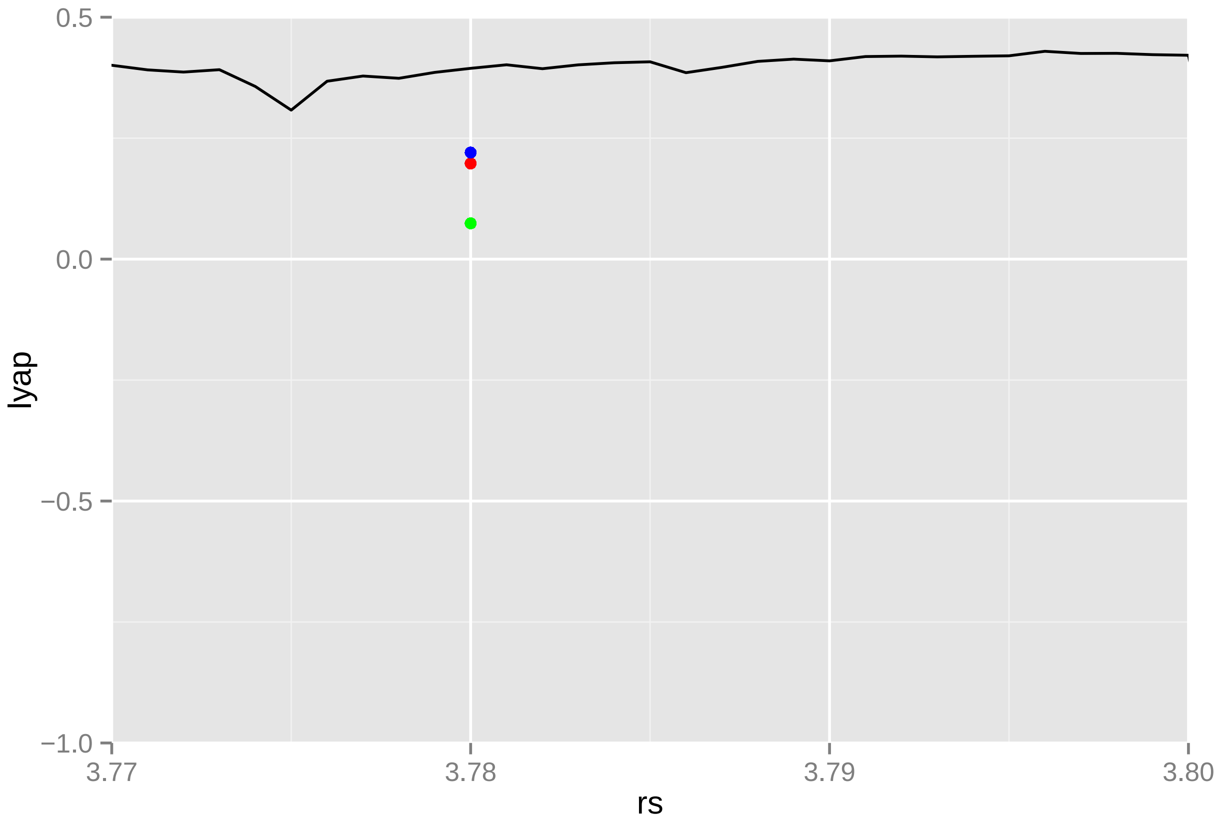
Comparison of the three methods

```
qplot(rs, lyap, data=dd, geom="line") +
  geom_point(aes(x=r, y=le0), col="red") +
  geom_point(aes(x=r, y=le1), col="blue") +
  geom_point(aes(x=r, y=le2), col="green")
```



And zoomed in:

```
qplot(rs, lyap, data=dd, geom="line") +
  geom_point(aes(x=r, y=le0), col="red") +
  geom_point(aes(x=r, y=le1), col="blue") +
  geom_point(aes(x=r, y=le2), col="green") +
  coord_cartesian(xlim=c(3.77, 3.8), ylim=c(-1, 0.5))
```




```
le0
```

```
##         x
## 0.1976677
```

```
le1
```

```
##    lambda
## 0.2203121
```

```
le2
```

```
##     lambda
## 0.07396539
```

Not brilliant, but if really bad, make sure for appropriate time windows over which the divergence rate is calculated.