Positive feedback measures

Marc

10 April 2016

Positive feedback model

The purpose of this document is to present and explore different measures of positive feedback. For this, we consider a model governed by the following three reactions:

$$\emptyset \underset{k_2}{\overset{k_1}{\rightleftharpoons}} A \underset{k_3}{\longrightarrow} A + A$$

where k_1 is the birth rate, k_2 is the decay rate and k_3 is the positive feedback strength. We want to study stochastic realisations of this model using the stochastic simulation algorithm which is implemented in the function "simple_positive_feedback.r", so first we must source the function.

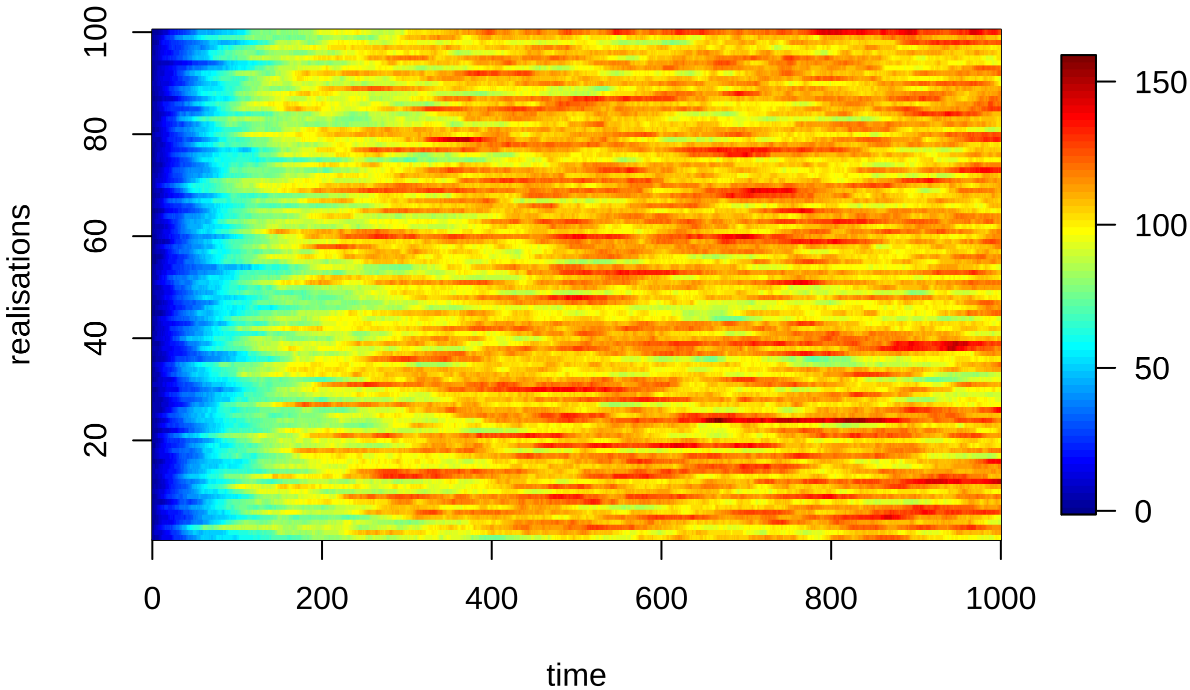
```
source('~/Positive_feedback_tests/simple_positive_feedback.r')
```

Let's define parameters, run 100 trajectories of the model and obtain the output:

```
k_1 = 1
k_2 = 0.01;
k_3 = 0.001;
maxtime = 1000;
timestep = 0.1;
numberofrealisations = 100;
output = simple_positive_feedback(k_1,k_2,k_3,maxtime,timestep,numberofrealisations);
```

We can also make a heatmap plot (you may have to install the "fields" package for this – install.packages("fields")):

```
time = seq(0,maxtime,by=timestep);
realisations = 1:numberofrealisations;
image.plot(time,realisations,output);
```

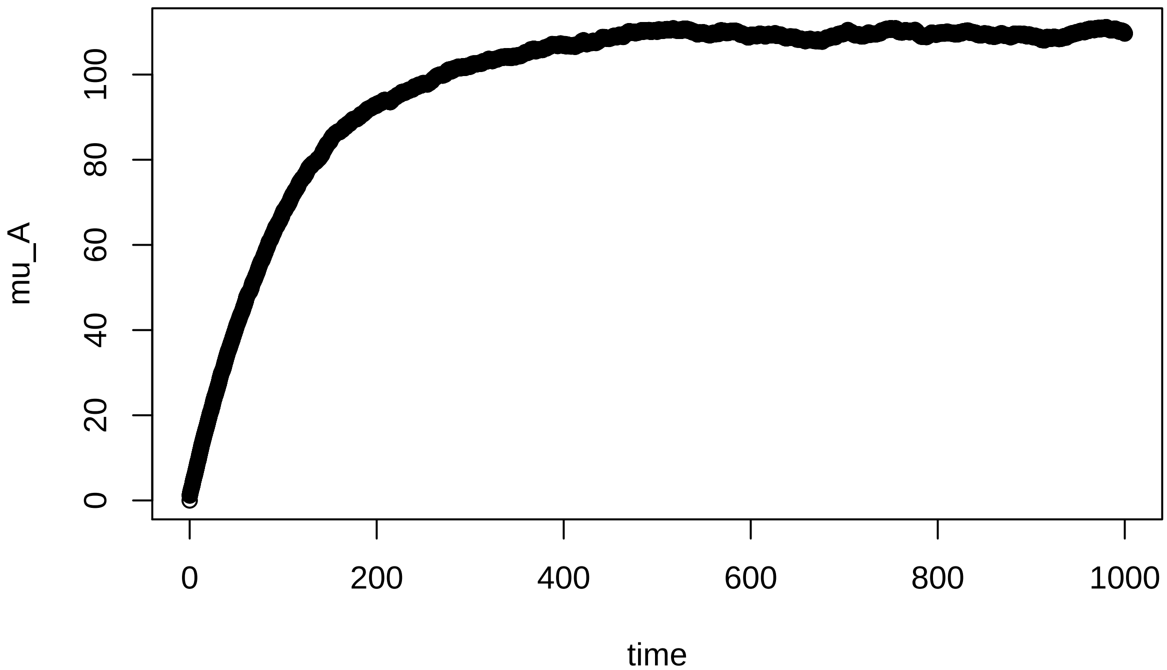


Now that we have a stochastic model with positive feedback strength as a parameter, we can try to find measures that are suitable for identifying positive feedbacks.

Mean

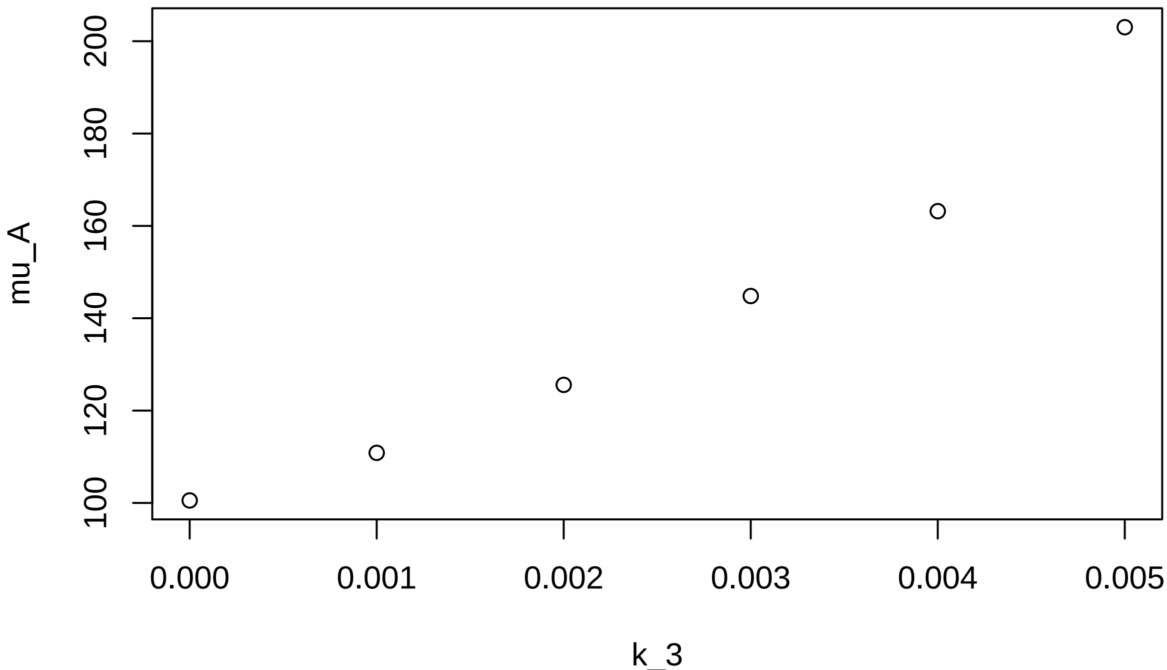
The first and most obvious measure is the mean. The mean value of A should increase with the positive feedback strength, k_3. To study this, first we can plot the mean time series for a single value of k_3:

```
time = seq(0,maxtime,by=timestep);
mu_A = rowMeans(output);
plot(time,mu_A);
```



Since the value of k_3 is small, the mean value (μ) of A takes a value just above the steady state with no positive feedback ($\frac{k_1}{k_2}$). We can now look at how the final value of this time series varies with k_3.

```
count = 0;
mu_A <- rep(0, 6);
for (k_3 in seq(0,0.005,by=0.001)){
  count = count + 1;
  output = simple_positive_feedback(k_1,k_2,k_3,maxtime,timestep,numberofrealisations);
  temp_mean_A = rowMeans(output);
  mu_A[count] = temp_mean_A[length(temp_mean_A)];
}
k_3 = seq(0,0.005,by=0.001);
plot(k_3,mu_A);
```



We can see that the mean value increases with positive feedback strength. The next measure we will inspect is the noise of A, specifically the Fano Factor.

Fano Factor

The Fano Factor (F) of A is defined as

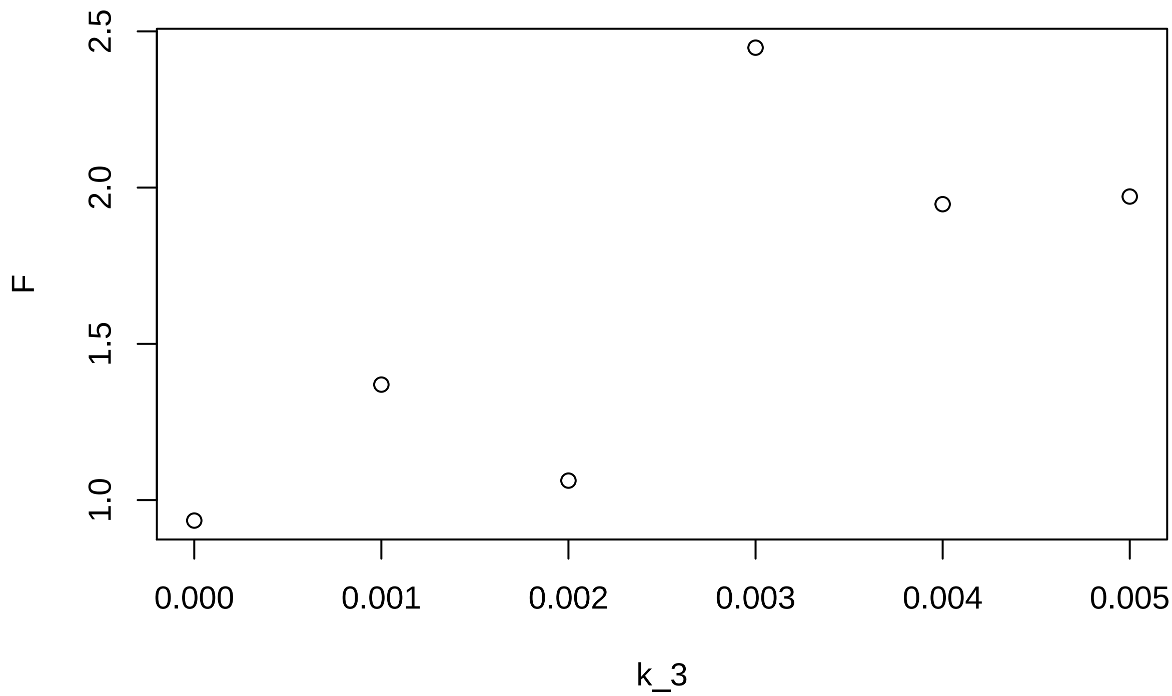
$$F = \frac{\sigma^2}{\mu}$$

where σ^2 is the variance of A. To compute this, we need to define the variance, which can be done using the following function:

```
RowVar <- function(x) {
  rowSums((x - rowMeans(x))^2)/(dim(x)[2] - 1)
}
```

We can now examine the examine the relationship between the positive feedback strength and Fano Factor. Probably need more realisations to get a decent estimate of the Fano Factor.

```
numberofrealisations = 20;
count = 0;
F <- rep(0, 6);
for (k_3 in seq(0,0.005,by=0.001)){
  count = count + 1;
  output = simple_positive_feedback(k_1,k_2,k_3,maxtime,timestep,numberofrealisations);
  temp_mean_A = rowMeans(output);
  temp_var_A = RowVar(output);
  F[count] = temp_var_A[length(temp_var_A)]/temp_mean_A[length(temp_mean_A)];
}
k_3 = seq(0,0.005,by=0.001);
plot(k_3,F);
```



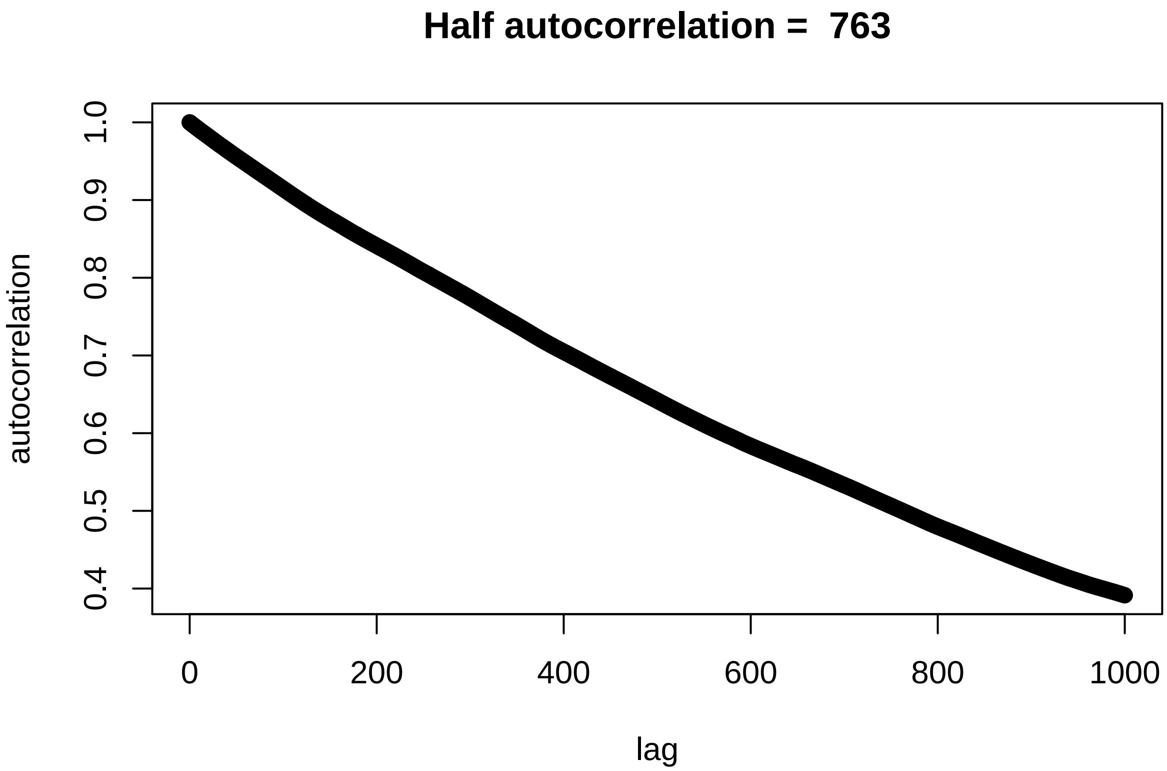
Autocorrelation

For a discrete process with known mean and variance for which we observe n observations $\{X_1, X_2, \ldots, X_n\}$, an estimate of the autocorrelation may be obtained as

$$\hat{R}(k) = \frac{1}{(n-k)\sigma^2} \sum_{t=1}^{n-k} (X_t - \mu)(X_{t+k} - \mu)$$

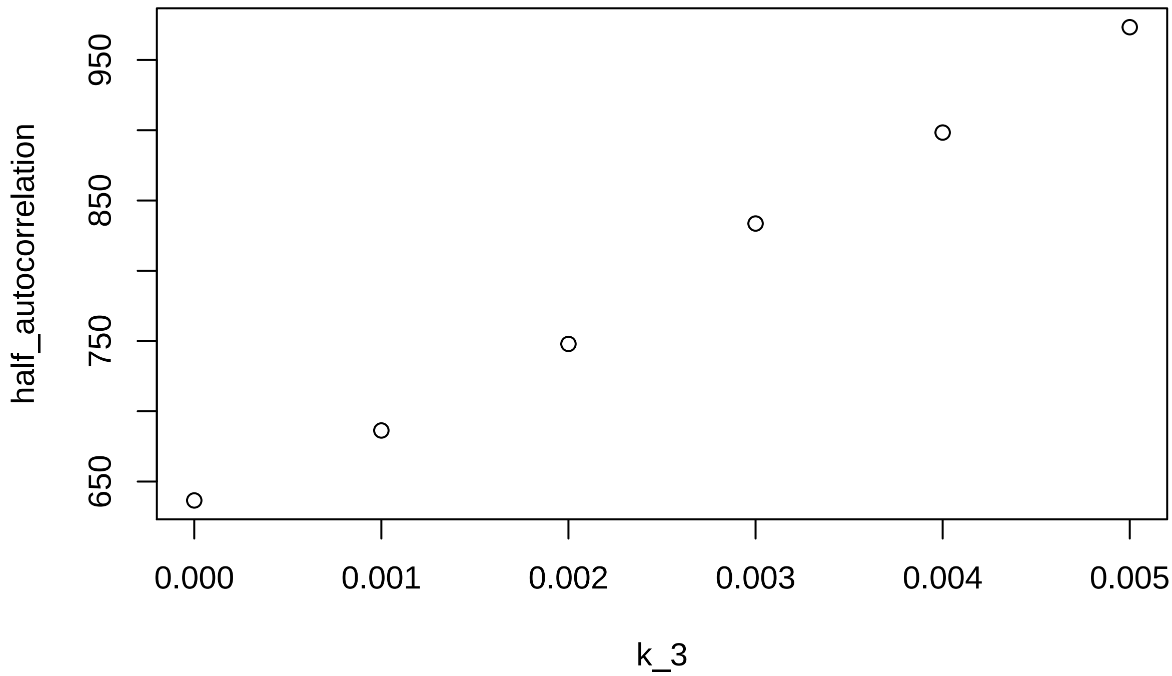
We can plot an example autocorrelation for a single trajectory of the model and plot it.

```
output = simple_positive_feedback(1,0.01,0.001,1000,0.1,1);
  autocorr = acf(output,lag.max = 1000,plot = FALSE)
test <- min(abs(autocorr$acf-0.5))
half_autocorr = which(abs(autocorr$acf-0.5) == test)
lag = autocorr$lag;
autocorrelation = autocorr$acf;
plot(lag,autocorrelation,main=paste("Half autocorrelation = ", half_autocorr))
```



Now we can examine how the half autocorrelation (which is displayed in the title of the previous plot) varies with k_3.

```
numberofrealisations = 100;
count1 = 0;
half_autocorr <- matrix(0,nrow=6,ncol=numberofrealisations);
for (k_3 in seq(0,0.005,by=0.001)){
  count1 = count1 + 1;
  for (count2 in seq(1,numberofrealisations,by=1)){
  output = simple_positive_feedback(1,0.01,k_3,1000,0.1,1);
  autocorr = acf(output,lag.max = 1000,plot = FALSE);
  test <- min(abs(autocorr$acf-0.5))
half_autocorr[count1,count2] = which(abs(autocorr$acf-0.5) == test);
  }
}
k_3 = seq(0,0.005,by=0.001);
half_autocorrelation = rowMeans(half_autocorr)
plot(k_3,half_autocorrelation);
```



Entropy

Named after Boltzmann's H-theorem, Shannon defined the entropy H (Greek letter H) of a discrete random variable X with possible values $x_1, \ldots, x_n$ and probability mass function $P(X)$ as:

$$H(X) = \mathrm{E}[\mathrm{I}(X)] = \mathrm{E}[-\ln(\mathrm{P}(X))]$$

.

Here E is the expected value operator, and I is the information content of X. $I(X)$ is itself a random variable.

The entropy can explicitly be written as

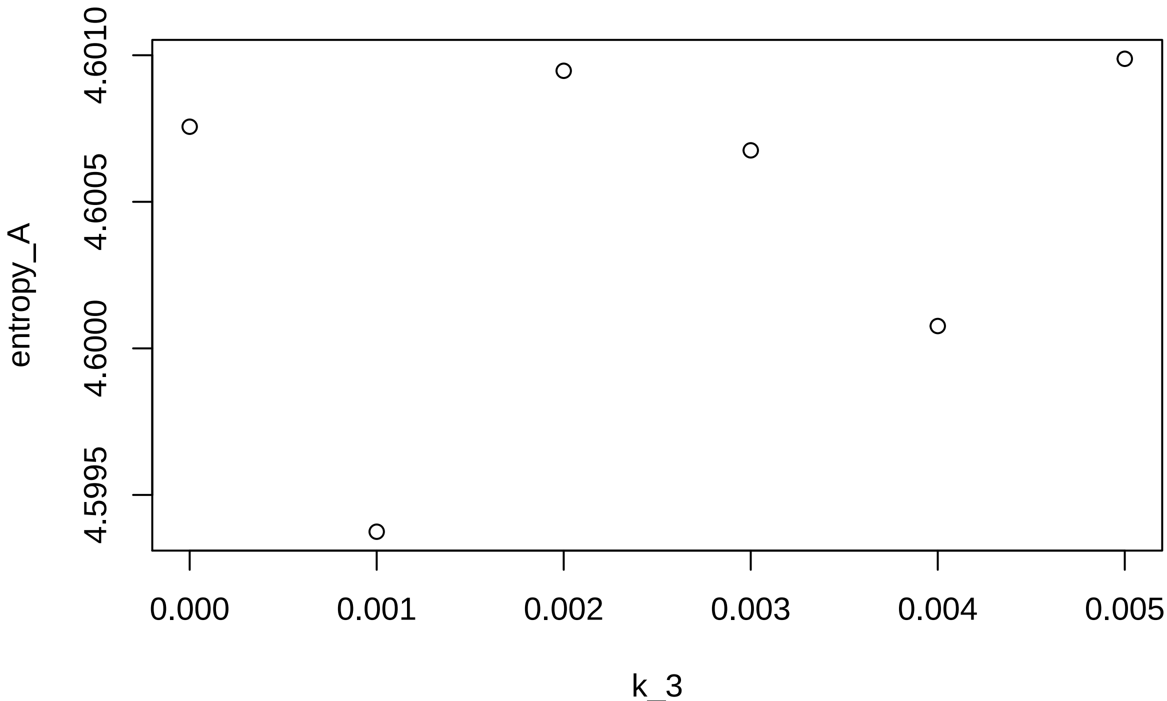
$$H(X) = \sum_{i=1}^{n} \mathrm{P}(x_i)\,\mathrm{I}(x_i) = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log_b \mathrm{P}(x_i),$$

where b is the base of the logarithm used. Common values of b are 2, Euler's number e, and 10, and the unit of entropy is shannon for $b = 2$, nat for $b = e$, and hartley for $b = 10$. When $b = 2$, the units of entropy are also commonly referred to as bits.

We can see how the entropy varies with k_3 by doing the following:

```
numberofrealisations = 100;
entropy_A <- rep(0,6);
count1 = 0;
for (k_3 in seq(0,0.005,by=0.001)){
  count1 = count1 + 1;
  A_end <- rep(0, numberofrealisations);
  for (count2 in seq(1,numberofrealisations,by=1)){
  output = simple_positive_feedback(1,0.01,k_3,1000,0.1,1);
  temp_A = output[length(output)]; #use final time series value
  A_end[count2] = temp_A;
```

```
  }
  entropy_A[count1] = entropy(A_end);
}
k_3 = seq(0,0.005,by=0.001);
plot(k_3,entropy_A);
```



Shannon entropy appears to remain constant, making it useless for positive feedback detection.