Systems Biology: Individual Assignment

University of Cambridge

Henrik Åhl

May 26, 2017

Preface

This is an assignment report in connection to the *Systems Biology* module in the Computational Biology course at the University of Cambridge, Easter term 2017. All related code is as of May 26, 2017 available through a Github repository by contacting hpa22@cam.ac.uk.

Exercises

A Under stationarity we have $\langle R^+ \rangle = \langle R^- \rangle$ for all species. That gives us

$$
\begin{aligned}
\langle x_0 \rangle &= \frac{\lambda_0}{\beta_0} \\
\langle x_1 \rangle &= \frac{\lambda_1}{\beta_1} \\
\langle x_2 \rangle &= \frac{\lambda_2 \langle x_0 x_1 \rangle}{\beta_2} = \frac{\lambda_2 \langle x_0 \rangle \langle x_1 \rangle}{\beta_2} = \frac{\lambda_0 \lambda_1 \lambda_2}{\beta_0 \beta_1 \beta_2}.
\end{aligned}
$$

B We calculate $\mathbf{D}$ from the $D_{ii} = \dfrac{2}{\tau_1} \dfrac{\langle s_i \rangle}{\langle x \rangle_i}$. Since we have no cross-production or decay, we have

$$
\mathbf{D} = \begin{pmatrix} \frac{2}{\tau_0} \frac{1}{\langle x_0 \rangle} & 0 & 0 \\ 0 & \frac{2}{\tau_1} \frac{1}{\langle x_1 \rangle} & 0 \\ 0 & 0 & \frac{2}{\tau_2} \frac{1}{\langle x_2 \rangle} \end{pmatrix} = \begin{pmatrix} \frac{2\beta_0^2}{\lambda_0} & 0 & 0 \\ 0 & \frac{2\beta_0}{\lambda_1} & 0 \\ 0 & 0 & \frac{2\beta_2^2 \beta_0 \beta_1}{\lambda_0 \lambda_1 \lambda_2} \end{pmatrix}
$$

due to Little's law, which gives us $\bar{\tau} = \left(\dfrac{1}{\beta_0}, \dfrac{1}{\beta_1}, \dfrac{1}{\beta_2} \right)$.

We compute $\mathbf{M}$ using the relationships $M_{ij} = \dfrac{H_{ij}}{\tau_i}$ and $H_{ij} = \dfrac{\partial \ln\left(R_i^-/R_i^+\right)}{\partial \ln x_j}\Bigg|_{x=\langle x\rangle}$. This gives us

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & -1 & 1 \end{pmatrix} \Rightarrow \mathbf{M} = \begin{pmatrix} \frac{1}{\tau_0} & 0 & 0 \\ 0 & \frac{1}{\tau_1} & 0 \\ -\frac{1}{\tau_2} & -\frac{1}{\tau_2} & \frac{1}{\tau_2} \end{pmatrix}$$

C We solve by hand to find that $\mathbf{M}\eta + (\mathbf{M}\eta)^T = \mathbf{D}$ gives us the equation system

$$\begin{aligned}
\frac{2}{\tau_0}\eta_{00} &= \frac{2}{\tau_0}\frac{1}{\langle x_0\rangle} \\
\left(\frac{1}{\tau_0} + \frac{1}{\tau_1}\right)\eta_{01} &= 0 \\
\left(\frac{1}{\tau_0} + \frac{1}{\tau_2}\right)\eta_{02} - \frac{1}{\tau_2}\eta_{00} - \frac{1}{\tau_2}\eta_{01} &= 0 \\
\frac{2}{\tau_1}\eta_{11} &= \frac{2}{\tau_1}\frac{1}{\langle x_1\rangle} \\
\left(\frac{1}{\tau_1} + \frac{1}{\tau_2}\right)\eta_{12} - \frac{1}{\tau_2}\eta_{01} - \frac{1}{\tau_2}\eta_{11} &= 0 \\
\frac{2}{\tau_2}(\eta_{22} - \eta_{02} - \eta_{12}) &= \frac{2}{\tau_2}\frac{1}{\langle x_2\rangle}
\end{aligned}$$

which solves as

$$\begin{aligned}
\eta_{00} &= \frac{1}{\langle x_0\rangle} \\
\eta_{01} &= 0 \\
\eta_{02} &= \frac{\tau_0}{\tau_0+\tau_2}(\eta_{00} + \eta_{01}) = \frac{\tau_0}{\tau_0+\tau_2}\frac{1}{\langle x_0\rangle} \\
\eta_{11} &= \frac{1}{\langle x_1\rangle} \\
\eta_{12} &= \frac{\tau_1}{\tau_1+\tau_2}(\eta_{00} + \eta_{11}) = \frac{\tau_1}{\tau_1+\tau_2}\frac{1}{\langle x_1\rangle} \\
\eta_{22} &= \frac{1}{\langle x_2\rangle} + \eta_{02} + \eta_{12} = \frac{1}{\langle x_2\rangle} + \frac{\tau_0}{\tau_0+\tau_2}\frac{1}{\langle x_0\rangle} + \frac{\tau_1}{\tau_1+\tau_2}\frac{1}{\langle x_1\rangle}.
\end{aligned}$$

Verification by use of Wolfram Alpha assures us of correct calculations.

In contrast to the previous two-species system, we are in this case not exact in the linearisation due to the R_2^+ term, which is non-linear. The first-order approximation is therefore indeed precisely that – an approximation.

D We can expect x_0 fluctuations to have a negligible effect on x_2 when the average x_0 abundance is high, or when $\tau_0 \gg \tau 2$ as the η_{02} then trends towards 0.

The fluctuations on x_2 due to fluctuations in x_0 are comparable to the x_1 fluctuations depend on how we choose to read the question. Given that we seek $\eta_{02} \approx \eta_{11}$ we have

$$\langle x_0\rangle\frac{\tau_0+\tau_2}{\tau_0} \approx \langle x_1\rangle.$$

If we instead are looking for $\eta_{02} \approx \eta_{12}$ (which is presumably the intention sought-after scenario) we have that

$$\frac{1}{\langle x_0 \rangle} \frac{\tau_0}{\tau_0 + \tau_2} \approx \frac{1}{\langle x_1 \rangle} \frac{\tau_1}{\tau_1 + \tau_2}$$

and as a result $\tau_0 \approx \tau_1$ given that the averages are of the same order of magnitude. Another possibility is that the averages both are very high, which causes them to dominate the equation. However, given that we are modelling mRNA molecules, and these tend to be low in abundance, this is not a very realistic scenario.

E Using $i = 1$ and $j = 2$, we get

$$\frac{1}{\tau_2} \frac{\mathrm{Cov}(x_1, x_2\beta_2 - \lambda_2 x_0 x_1)}{\langle x_1 \rangle \langle R_2^{\pm} \rangle} + \frac{1}{\tau_1} \frac{\mathrm{Cov}(x_2, x_1\beta_1 - \lambda_1)}{\langle x_2 \rangle \langle x_1 \rangle \langle R_1^{\pm} \rangle} = 0$$

when inserting the corresponding values for our other constants. Using the two covariance rules of $\mathrm{Cov}(x, y + a) = \mathrm{Cov}(x, y)$ and $\mathrm{Cov}(x, ay) = a\mathrm{Cov}(x, y)$ for some constant a we can expand on the previous statements. We get

$$\frac{1}{\tau_2} \frac{\beta_2 \mathrm{Cov}(x_1, x_2)}{\langle x_1 \rangle \langle R_2^{\pm} \rangle} - \frac{1}{\tau_2} \frac{\lambda_2 \langle x_0 \rangle \mathrm{Cov}(x_1, x_1)}{\langle x_1 \rangle \langle R_2^{\pm} \rangle} + \frac{1}{\tau_1} \frac{\mathrm{Cov}(x_2, x_1\beta_1)}{\langle x_2 \rangle \langle x_1 \rangle \langle R_1^{\pm} \rangle} = 0$$

where we can now identify the rates to be either the influx or the outflux term. In order to eliminate constants, we choose outflux, influx and outflux respectively. We therefore get

$$\frac{1}{\tau_2} \frac{\mathrm{Cov}(x_1, x_2)}{\langle x_1 \rangle \langle x_2 \rangle} - \frac{1}{\tau_2} \frac{\mathrm{Var}(x_1)}{\langle x_1 \rangle^2} + \frac{1}{\tau_1} \frac{\mathrm{Cov}(x_1, x_2)}{\langle x_1 \rangle \langle x_2 \rangle} = 0$$

when we note that $\mathrm{Cov}(x_1, x_1) = \mathrm{Var}(x_1)$. Restructuring we find that

$$\mathrm{Cov}(x_1, x_2) = \mathrm{Var}(x_1) \frac{\langle x_2 \rangle}{\langle x_1 \rangle} \frac{\tau_1}{\tau_1 + \tau_2}$$

which is equivalent to

$$\eta_{12} = \eta_{11} \frac{\tau_1}{\tau_1 + \tau_2}$$

i.e. the correlations are related via a time-averaging constant. We note that the expression is completely independent of x_0 and its related parameters, as we would expect in the correlation between x_1 and x_1 due to x_1 not having any direct nor indirect interaction with x_0. The only particle which is affected by both is, as we can see from the correlations, x_2.

Using the definition $\rho_{12} = \dfrac{n_{12}}{CV_1 CV_2}$ we see that we can rewrite this as

$$\rho_{12} = \frac{CV_1}{CV_2} \frac{\tau_1}{\tau_1 + \tau_2} = \frac{CV_1}{CV_2} \frac{1}{1 + \dfrac{\tau_2}{\tau_1}}.$$

F In order to get a suitable range we set $\langle x_0 \rangle := 10$ and τ_0 uniformly covering the interval $[0.001\tau_2, \tau_1]$ as inferred from our previous discussion of the domains for the effects. (We assume the lecturers are indeed asking for the case when $\eta_{02} \approx \eta_{12}$ in the ambiguously phrased question.)

As fig. 1 shows, the relationship between the ratio of CV_1 and CV_2 behave similarly as before. Note however that, when comparing to our previous figure, we now have the quota CV_1/CV_2 rather than the inverse. Compared to the case where we have two species, the correlation between particle one and particle two have now gone down as we have introduced yet another molecule which affects the abundance of x_2. Because we do not have any direct interaction between x_0 and x_1 we see a decrease in correlation when we introduce our extra particle. The ratio CV_2/CV_1 increases as CV_2 goes up due to the added particle, whereas CV_1 stays constant.

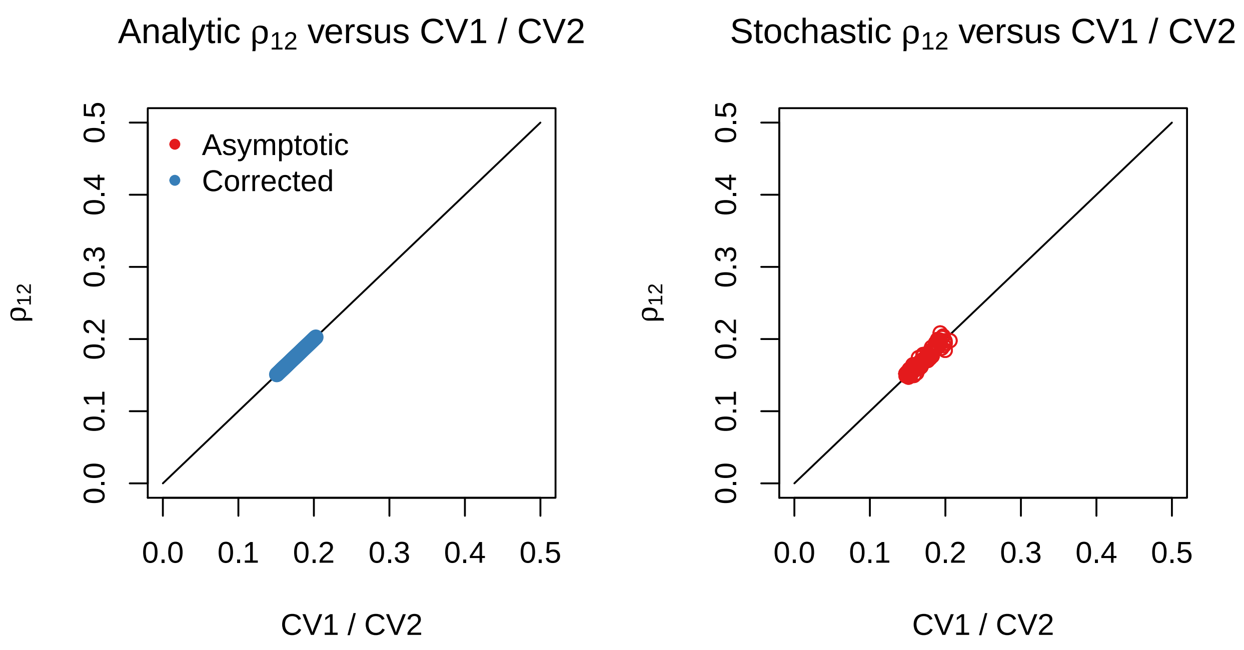

Figure 1

2 Acknowledgements

As always, many thanks to Julian Melgar for no particular reason. Also thanks to the people at Lund University who unwittingly lended me their computers for running the simulations.

A Code

```sh
#!/bin/sh

## This script runs the simulations in R in parallel

run () {
  START=$1
  END=$2
  for ii in $(seq $START $END); do
    (
      RETURN_VAL=$(nice -n 5 Rscript --vanilla /local/data/public/hpa22/assignments/syba1/
          gillespie.R $ii)
    ) &
  done
}

run 1 20
wait
run 21 40
wait
run 41 60
wait
run 61 80
wait
run 81 100

exit
```

../code/gillespie.sh

```r
#!/usr/bin/env Rscript
setwd("/local/data/public/hpa22/assignments/syba1")
rm(list = ls())

# Import arguments from BASH script (choice of tau1)
args = commandArgs(trailingOnly=TRUE)

if (length(args) != 1) {
  stop("This script needs 1 argument: tau0 index.")
}

tau0.idx <- args[1]

# Simulation parameters
no.iterations = 20e6
no.reactions  = 6
no.species    = 3
epsilon       = 10e-3
reaction.sign = c(1, -1, 1, -1, 1, -1)
reactions = rep(NA, no.reactions)
species.index = c(1, 1, 2, 2, 3, 3)

# Iteration independent parameters
gamma    = 29 # Birthday
avg.x0   = 10
avg.x1   = 10
avg.x2   = 1000
start.x0 = round(avg.x0)
start.x1 = round(avg.x1)
start.x2 = round(avg.x2)
start.t  = 0
tau1     = round((2 + 8 * gamma / 31) * 60)
tau2     = 180 * 60
tau0     = seq(tau2 / 1000, tau1, length.out = 100)
beta1    = 1 / tau1
beta2    = 1 / tau2
```

```r
lambda1  = avg.x1 * beta1
lambda2  = avg.x2 * beta2 / (avg.x0 * avg.x1)

# Calculate a weighted (co)variance
weighted.cov = function(var1, var2 = var1, weights, w.mean1, w.mean2 = w.mean1){
  1 / sum(weights) * sum(weights * (var1 - w.mean1) * (var2 - w.mean2))
}

step.size = 100000
start.pt = 100000
extras = matrix(NA, ncol = 15, nrow = (no.iterations-start.pt) / step.size)
count = 1


# Main function
# tau0 = tau2 / 1000 # Test with this
gillespie = function(tau0 = tau0, no.iterations = no.iterations){
  # System parameters
  # This time we change only for tau0
  beta0 = 1 / tau0
  lambda0 = avg.x0 * beta0

  # Preallocation and definition of output matrix
  probabilities = rep(NA, no.reactions)
  out.values = matrix(NA, ncol = 1 + no.species, nrow = no.iterations)
  out.values[1, ] = c(start.t, start.x0, start.x1, start.x2)
  ii = 2
  colnames(out.values) = c("t", "x0","x1","x2")

  while(ii < no.iterations + 1) {
    # Define reaction probabilities
    reactions = c(
      lambda0,
      beta0 * out.values[ii - 1, 2],
      lambda1,
      beta1   * out.values[ii - 1, 3],
      lambda2 * out.values[ii - 1, 2] * out.values[ii - 1, 3],
      beta2   * out.values[ii - 1, 4])
    sum.reactions = sum(reactions) # This is our "a_0" constant.

    # Increment / decrement reacting species, choosing one of the 6 possible reactions
    reaction.index = sample(1:no.reactions, 1, prob = reactions / sum.reactions)
    reaction.species = 1:no.species == ceiling(reaction.index / 2)

    # Update the time and chosen spec ies
    out.values[ii, 1:4] = out.values[ii - 1, 1:4] +
      c(rexp(1, sum.reactions), # Increment time (rexp is faster than runif trickery)
      reaction.sign[reaction.index] * reaction.species) # Update the correct species

    # Compute running average of each species, every 100000 iterations
    if ((ii - 1) %% step.size == 0 && ii > start.pt) {
      cat("iteration ", ii, " of ", no.iterations, "\n")
      wts = diff(out.values[start.pt:ii,  "t"])
      x0s = out.values[start.pt:(ii - 1), "x0"]
      x1s = out.values[start.pt:(ii - 1), "x1"]
      x2s = out.values[start.pt:(ii - 1), "x2"]

       # Weighted mean depending on different reaction times
       x0.mean = weighted.mean(x0s, wts)
       x1.mean = weighted.mean(x1s, wts)
       x2.mean = weighted.mean(x2s, wts)

       # Flux balance
       birth.x0 = lambda0
       death.x0 = beta0   * x0.mean
       birth.x1 = lambda1
       death.x1 = beta1   * x1.mean
      birth.x2 = lambda2 * x0.mean * x1.mean
      death.x2 = beta2   * x2.mean

      # Return the relative errors in balance
      extras[count, 1:3] = c((birth.x0 - death.x0) / birth.x0,
                             (birth.x1 - death.x1) / birth.x1,
                             (birth.x2 - death.x2) / birth.x2)

      # Fluctuation balance (from simulation and expected)
      n.00 = weighted.cov(var1 = x0s, var2 = x0s, weights = wts, w.mean1 = x0.mean, w.mean2 = x0.mean) / (x0.mean ** 2)
      n.01 = weighted.cov(var1 = x0s, var2 = x1s, weights = wts, w.mean1 = x0.mean, w.mean2 = x1.mean) / (x0.mean * x1.mean)
      n.02 = weighted.cov(var1 = x0s, var2 = x2s, weights = wts, w.mean1 = x0.mean, w.mean2 = x2.mean) / (x0.mean * x2.mean)
      n.11 = weighted.cov(var1 = x1s, var2 = x1s, weights = wts, w.mean1 = x1.mean, w.mean2 = x1.mean) / (x1.mean ** 2)
      n.12 = weighted.cov(var1 = x1s, var2 = x2s, weights = wts, w.mean1 = x1.mean, w.mean2 = x2.mean) / (x1.mean * x2.mean)
      n.22 = weighted.cov(var1 = x2s, var2 = x2s, weights = wts, w.mean1 = x2.mean, w.mean2 = x2.mean) / (x2.mean ** 2)

      # Theoretical answers
      n.00.exp = 1 / x0.mean
      n.01.exp = 0
      n.02.exp = tau0 / (tau0 + tau2) * n.00.exp
      n.11.exp = 1 / x1.mean
      n.12.exp = tau1 / (tau1 + tau2) * n.11.exp
      n.22.exp = 1 / x2.mean + n.02.exp + n.12.exp

      # Return the relative percent difference in balance
      extras[count, 4:9] = c((n.00 - n.00.exp) / n.00.exp,
                             (n.01 - n.01.exp),
                             (n.02 - n.02.exp) / n.02.exp,
                             (n.11 - n.11.exp) / n.11.exp,
                             (n.12 - n.12.exp) / n.12.exp,
                             (n.22 - n.22.exp) / n.22.exp)

      # Return normalised covariances
      extras[count, 10:15] = c(n.00, n.01, n.02, n.11, n.12, n.22)

      # Check if the errors are smaller than a threshold, and stop the simulation if true
      if(all(abs(extras[count, 1:9]) < epsilon)) {
        return(list(out.values, extras))
      }
      count = count + 1
    }
    ii = ii + 1
  }
  return(list(out.values, extras))
}

# Run the simulation
# a <- gillespie(tau1[as.numeric(tau1.idx)], no.iterations=no.iterations)
results = gillespie(tau0[as.numeric(tau0.idx)], no.iterations=no.iterations)
# results = gillespie(tau0, no.iterations=no.iterations)

# Store the resulting matrix in an .RData object
save(results, file = paste0(tau0.idx, "_result.RData"))

# All RData files save a matrix named "results", so loading any file
# will import an "a" matrix (watch out for overwriting!)
```

../code/gillespie.R