Introduction to systems biology

Assignment 1

Problem 1: [1 point]

In a few sentences, without referring to specific numbers, what are the major quantitative differences that impact biology between *E.coli, S. cerevisiae* and human cells?

Solution:

E. coli is the smallest of the three with much less cell volume, fewer proteins per cell, and with the smallest genome. In addition, the regulatory complexity is reduced. This is unsurprising as *E. coli* is a prokaryote and thus lacks both a nucleus and other specialized organelles. *S. cerevisiae* and human cells are eukaryotes and thus have a nucleus and a more complex cell structure. Human cells are more complex of the two with a larger cell volume, more proteins per cell, and the largest genome.

Of the three cell types, *E.coli* is the quickest to transcribe genes and translate proteins but suffers from a reduced mRNA and protein lifetime, at least when compared to *S. cerevisiae* and human cells. Lastly, the cell generation time is the greatest for human cells and the least for *E. coli*.

Problem 2: [1 point]

Complete part b of Exercise 1.1 (page 16) from book chapter 1. The one that refers to the response time.

Solution:

We assume that before the production rate changes that our system is at equilibrium. That is to say, the production is fixed at $y^s = \beta_1/\alpha$. At time $t = 0$, the production rate of the system changes from β_1 to β_2, thus altering the dynamics of the system. The concentration of y at time t is now described by the first-order, linear equation:

$$\frac{dy}{dt} = \beta_2 - \alpha y. \tag{1}$$

The closed-form solution of such a system is well known and indeed it is given to be:

$$y(t) = \frac{\beta_2}{\alpha} + \left(\frac{\beta_1}{\alpha} - \frac{\beta_2}{\alpha}\right)\exp(-\alpha t). \tag{2}$$

The new steady state y^{sn} can be found by either setting equation (1) equal to 0 and solving for y or by taking the limit of equation (2) as t tends to infinity. In both cases, we get that $y^{sn} = \beta_2/\alpha$.

We are asked to find the response time τ of this new system. The response time is defined as the time at which the system is halfway between its initial state and final state. As our system begins at β_1/α and ends at β_2/α, our point of interest is the mean:

$$y^m = \frac{1}{2}\left(\frac{\beta_1}{\alpha} + \frac{\beta_2}{\alpha}\right).$$

Consequently, determining τ amounts to solving:

$$y^m = \frac{1}{2}\left(\frac{\beta_1}{\alpha} + \frac{\beta_2}{\alpha}\right) = \frac{\beta_2}{\alpha} + \left(\frac{\beta_1}{\alpha} - \frac{\beta_2}{\alpha}\right)\exp(-\alpha\tau),$$

for τ. The expression above can be rewritten as:

$$\frac{1}{2}(\beta_1 - \beta_2) = (\beta_1 - \beta_2)\exp(-\alpha\tau),$$

which further reduces to:

$$\frac{1}{2} = \exp(-\alpha\tau).$$

Using logarithmic rules we finally arrive at $\tau = \ln 2/\alpha$, where ln is the natural log.

Problem 3: [1.5 points]

Read the paper *"Can a biologist fix a radio? - Or, what I learned while studying apoptosis"*

a. [1 point]: *"A related argument is that engineering approaches are not applicable to cells because these little wonders are fundamentally different from objects studied by engineers."* The author of the article disagrees with this argument. What do you think? Write a short essay (half page, ~250 words) where you argue both sides.

b. [0.5 points]: *"Another argument is that we know too little to analyze cells in the way engineers analyse their systems. But, the question is whether we would be able to understand what we need to learn if we do not use a formal description."* This issue is also raised by the author. Take a look at figure 3 in the article. Elaborate two reasons why today (unfortunately) we are not capable to represent a biological system with a figure like Fig 3b.

Solution:

a. One of the uses of mathematics is that it allows us to describe objects in an abstract manner. The benefits of such abstractions are many but one is particularly relevant to this topic viz. that two objects, which at first appear to be completely independent of each other, are revealed to be equivalent (from a certain perspective) once they have been abstracted. When such a relationship exists we say (mathematically) that the two objects are *isomorphic* to each other and it is an immensely powerful connection; indeed, it allows us to study the properties of one object from the perspective of the other and vice versa. This is all well and good when we are examining mathematical structures but unfortunately biological systems are not mathematical objects. There is no reason to expect a one-to-one correspondance between a mathematical approximation of a system and the system itself. Such a limitation may appear to render the engineering approach inappropriate; at least at first glance. All is not lost however as there exists a weaker form of isomorphisms; namely *homomorphisms*. If two objects are homomorphic we can say that they are similar in some sense but not identical as information is lost in translation. Nevertheless, we may still be able to divine some properties of one object through the lense of the other, justifying the endeavor despite its "limitations". The preceeding discussion may seem irrelevant but it is my hope that it illuminates *why* I agree with the author. Even if mathematical representations of biological systems are crude approximations they may yet prove to be *useful*. That is not to say that I am dismissive of the classical approach. Ideally, it is the synthesis of *both* approaches that will ultimately yield the greatest benefit as it allows two schools of thought, two points of view, to completement and support each other in the pursuit, and generation, of new knowledge.

b. ewrwer2

est

Problem 4: [1.5 points]

Complete Exercise 2.1 (page 31) from the book chapter 2. Along with your answer, please submit your code too (plain text, GitHub and Colab are ideal ways to share your code). Choose the programming language of your preference, any programming language is accepted for this exercise.

Solution:

We are asked to generate 100 Erdos-Renyi graphs, each with 400 nodes and 500 arrows, and find the mean and standard deviation of the total number of self-arrows.

Let n denote the number of nodes and k the number of edges (arrows) in the graph G. Denote the adjaceny matrix of graph G with A. Then A is of size $n \times n$. Furthermore, denote the ij-th element of A with a_{ij}. The element a_{ij} can only take one of two possible values; 1 if there exists an edge between nodes i and j, and 0 otherwise. As G can only have k edges we have that $T = \sum_{i,j}^{k} a_{ij} = k$. Note that the diagonal element a_{ii} takes on a value of 1 if there exists an edge from node i to itself (self arrow).

To naively simulate such a system it suffices to randomly select element a_{ij} and change its value to 1. The process is then repeated until $T = k$. Then, to find the total number of self-arrows for a given iteration, we can find the trace A. Such a method was implemented in `erdosRenyiNaive.R` which can be seen at the end of this document and on GitHub.

The results of the simulations can be seen in the following table:

Type	Value
Theoretical mean	1.250000
Theoretical SD	1.118034
Simulation mean	1.250000
Simulation SD [unbiased]	1.200799
Simulation SD [biased]	1.199297
Computational time	20.358714

Note that the computational time is in seconds.

If its the diagonal of the adjaceny matrix that is of interest it is possible to improve on `erdosRenyiNaive.R`. Let X_i be a Bernoulli random variable with success probability p such that:

$$X_i = \begin{cases} 1 & \text{edge from node } i \text{ to itself exists} \\ 0 & \text{otherwise.} \end{cases}$$

Then $Y = \sum X_i$ is the total number of nodes with self-arrows. Now, Y is a binomial random variable with parameters k (as the total number of arrows represent the total number of "trials") and $p = 1/n$. Furthermore, the expected value of Y is $E[Y] = kp$ and the variance is $Var(Y) = kp(1-p)$. Note that when p is a small quantity (as is the case in the book) then $1-p \approx 1$ and we have $Var(Y) \approx kp$. Using this approach we can simulate the diagonal more efficiently, thus allowing for more iterations which will in turn yield better estimates.

Below is a table where the mean and standard deviations have been computed for increasing numbers of simulations. When the computational times are compared to the naive method we see that there is significant decrease. Furthermore, the mean of the simulations approach the theoretical value. The implementation can be seen in `erdosRenyiDiag.R` or on GitHub.

Iteration	Simulations	Mean	SD	Time
1	100	1.30000	1.000000	0.0011549
2	1000	1.25800	1.142252	0.0017545
3	10000	1.25150	1.116142	0.0222073
4	100000	1.24745	1.117296	0.2178085

Problem 5: [1 point]

Complete Exercise 2.9 (page 34) from the book chapter 2.

Solution:

There are two steady states for the double-positive loop there are two steady states: either both X and Y are *on* or both are *off*. The double-negative loop does not enjoy the same steady states as the double-positive loop; viz. either X is *on* and Y is *off*, or X is *off* and Y is *off*.[1]

If X and Y belong to the same tissue the double-positive loop is more beneficial. If X and Y belong to different tissues then the double-negative loop is better.[1]

Code

LOAD.R

```
library(tidyverse)
library(data.table)
library(knitr)
library(kableExtra)
library(here)
```

erdosRenyiNaive.R

```
# Function to initialize an empty adjacency matrix
genMatrix <- function(n) {
        emptyMatrix <- matrix(0, nrow = n, ncol = n)
}

# Naive function to generate an Erdos-Renyi graph
# by populating an adjacency matrix for some
# graph with n nodes until k edges have been
# formed.
populateMatrix <- function(n, k) {
        adjMatrix <- genMatrix(n)
        cnt <- 0
        while(cnt < k) {
                row <- sample(1:n, size = 1)
                col <- sample(1:n, size = 1)
                if (adjMatrix[row, col] == 0) {
                        adjMatrix[row, col] <- 1
                }
                cnt <- sum(adjMatrix)
        }
        return(adjMatrix)
}
```

```r
# Function to count number of self arrows by
# computing the trace of the adjacency matrix
selfArrowsSum <- function(adjMatrix) {
        return(sum(diag(adjMatrix)))
}

# Function to compute the theoretical mean,
# theoretical standard deviation, simulated
# sample mean, simulated unbiased and biased
# sample standard deviation.
summaryStatistics <- function(data, n, k) {
        theoreticalMean <- k/n
        theoreticalSD <- sqrt(k/n)
        simulatedMean <- mean(data)
        simulatedUnbiasedSD <- sd(data)
        simulatedBiasedSD <- sd(data) * sqrt((n - 1) / n)
        return(list(theoreticalMean = theoreticalMean,
                    theoreticalSD = theoreticalSD,
                    simulatedMean = simulatedMean,
                    simulatedUnbiasedSD = simulatedUnbiasedSD,
                    simulatedBiasedSD = simulatedBiasedSD))
}

##################
### Test case ###
##################

# Seed for reproducibility
set.seed(1)

# Define nodes (n), arrows (k) and number of iterations (L)
n <- 400
k <- 500
L <- 100
```

```
# Initialize output data frame and iterate
# Store computational time for comparison
info <- data.frame(it = 1:L, sum = 0)

start <- Sys.time()
for (i in 1:L) {
        erdosRenyi <- populateMatrix(n, k)
        info[i, 2] <- selfArrowsSum(erdosRenyi)
}
stop <- Sys.time() - start
stop <- stop[[1]]

# Get results and export as table
res <- summaryStatistics(info$sum, n, k)
export <-
    data.frame(Type = c('Theoretical mean',
                        'Theoretical SD',
                        'Simulation mean',
                        'Simulation SD [unbiased]',
                        'Simulation SD [biased]',
                        'Computational time'),
               Value = c(unlist(res), stop))


path <- here('tables', 'erdNavTab.txt')
write.table(x = export, file = path, quote = F,
            sep = ';', row.names = F)
```

erdosRenyiDiag.R

```
set.seed(1)
# Initialize variables for simulation
# L (iterations), n (nodes), k (edges)
```

```r
L <- c(100, 1000, 10000, 100000)
n <- 400
k <- 500
l <- length(L)

# Initialize a results data frame
res <- data.frame(Iteration = 1:l,
                  Simulations = L,
                  Mean = 0,
                  SD = 0,
                  Time = 0)

# Iterate for each element of L and store results
for (i in 1:l) {
    start <- Sys.time()
    sim <- replicate(L[i], rbinom(1, k, p = 1/n))

    res[i, 3] <- mean(sim)
    res[i, 4] <- sd(sim)

    stops <- Sys.time() - start
    res[i, 5] <- stops[[1]]
}

path <- here('tables', 'erdDiaTab.txt')

# Write results to table and export
write.table(x = res, file = path, sep = ';',
            quote = F, row.names = F)
```

References

1. Alon, U. Network motifs: Theory and experimental approaches. *Nature Reviews Genetics* **8**, 450–461 (2007).