Computational Biology I: Quantitative Data Analysis
Frühjahrsemester 2019

Exercise 3

Pascal Grobecker and Athos Fiori

Reminder: Arrays and functions

Last exercise we introduced the array data type of the Numpy package (`import numpy as np`). You can call the functions in the package by adding "`np.`" in front of the functions (eg. `np.exp()` will call the `exp()` function). Arrays are in a way specialized lists that only contain numbers and that can have more than one dimension. They are used to make mathematical calculations practical and efficient. Lets say, a function involves operations that are defined element wise on arrays, like `*`, `/`, `log` and `exp` etc. Then, in order to apply the function over a sequence of number you don't need to use a for-loop.

```
In [1]: x = arange(51) # array integers from 0 to 50

In [2]: pmf = exp(log_hypergeometricPMF(100,50,50,x))
```

Here the function will work on all the elements in x. For more info on arrays, check the last exercise sheet and probably have a look at the extensive manual at
https://docs.scipy.org/doc/numpy-dev/user/quickstart.html
In the template you will find several functions which are already implemented by us and we encourage you to make use of them to save a bit of writing!

How to submit the exercise

Submit your completed `Exercise3.py` file by email to biocomp1-bioz@unibas.ch. The submission deadline is Thursday, midnight.

1. Check that your functions are working properly before submitting them!
2. Don't forget to attach your exercise file!
3. Don't change the name of the file or the name of any function within the file!
4. Please send your solutions from the same email address that you used to registering at the course.

Homework

Confidence intervals vs. Bayesian posterior probability intervals In the cell culture problem of exercise 2 you were asked to calculate the posterior probability that M cells in a population of N total cells express a certain protein given that you have sampled n cells from the population of which m express it, i.e., $P(M|N,n,m)$. From this posterior distribution you were able to infer a 95% posterior probability interval which contains the true value of M with 95% probability. In this exercise we will contrast this posterior probability interval with a confidence interval which is used in orthodox statistics to characterize the uncertainty in your estimate. Remember that confidence intervals are calculated from the likelihood function $P(m|N,M,n)$ only. Their interpretation is as following: a 95% confidence interval is the interval which in 95% of the cases lies over

the true value of M, when it is repeatedly estimated from data of the same kind. In order to find the boundaries of that interval, lets define a right-hand p-value as

$$p_r(k \geq m|M) = \sum_{k=m}^{N} P(k|N, M, n).$$

Note that this p-value is a function of M. You find the lower interval boundary M_{min} by picking the smallest M such that $p_r > 0.025$. Conversely, the left-hand p-value is

$$p_l(k \leq m|M) = \sum_{k=0}^{m} P(k|N, M, n).$$

The upper interval boundary M_{max} is the largest value of M for which $p_l > 0.025$.

1. Complete the function `cellCulture5` which calculates the 95% confidence interval. Tip: compare your results for the data set $n = 10, m = 3, N = 1000$ with the 95% posterior probability interval. What do you notice?

   Hint: *You can rewrite the sum in the following way to make the calculation easier:*

$$\sum_{k=m}^{N} x_k = \sum_{k=0}^{N} x_k - \sum_{k=0}^{m-1} x_k$$

Estimating promoter diversity In our lab we are interested in transcription regulation in the bacterium *E. coli* and to study what makes a piece of DNA a functional promoter we decided to use an "in lab evolution" strategy. We started out by creating totally random sequences of roughly 150 base pairs each, cloned these sequences into a plasmid, directly upstream of a gene encoding GFP, and then transformed E. coli cells with these plasmids. Thus we created an initial population with millions of cells that each carried a unique plasmid containing GFP with a 150 base pair promoter sequence that is entirely random. Of course, the very large majority of these sequences are not going to function as a promoter, i.e. they will not drive expression of GFP. But through a process of selection and mutation we were able to evolve a pool of bacteria that each carry a plasmid with a promoter that drives expression of GFP at a given level (that we selected for).
The question now is: How many different unique functional promoter sequences have we evolved? In solving this problem we assume:

1. Our pool has been purged of non-functional sequences: all remaining sequences are functional.
2. Our pool of sequences has many copies of each unique promoter sequence.
3. All unique promoter sequences are roughly equally abundant.

To estimate how many unique promoter sequences we have evolved, we take a sample of n of them, and sequence them in a sequencing machine. Our data D thus simply consists of a set of DNA sequences. From this we now want to estimate the total number of unique promoter sequences N in our pool.
Thus, we get a file from the sequencing center with DNA sequences, and the first sequence in the file we are going to call S_1. We do not care what sequence it is. The second sequence in the file is either the same as the first sequence, or it is a new one. If it is the same as the first sequence we also call it S_1, otherwise we call it S_2. And so on. After sequencing n samples our data D consists of a sequence that will look something like this:

$$D = (S_1, S_2, S_3, S_4, S_2, S_5, S_6, S_3, S_1, S_7, S_6). \tag{1}$$

In this case we assumed $n = 11$ samples. Note that we observed the sequences S_1, S_2, S_3, and S_6 two times each, and the other sequences S_4, S_5 and S_7 once each. Lets call k the number of unique sequences within our sample. The probability $P(D|N, n, k)$ that, given N, we obtain precisely a particular series like this is

$$P(D|N, n, k) = \frac{N!}{(N-k)!} \frac{1}{N^n}. \tag{2}$$

1. **Theoretical Question** Explain why we pick formula 2 to estimate the probability!
   Hint: think of how many different permutations you can get by drawing k out of N promoters and the fact that we don't know all mutual sequences in advance!

2. Implement the function `likelihood` that, given the relevant statistics of you data-set D, calculates the likelihood $P(D|N,n,k)$. Implement your function such that it allows arrays as input arguments!

3. Using the likelihood function, implement the `posterior` function that calculates the posterior probability distribution $P(N|D,n,k)$ assuming that the prior $P(N)$ is uniform, i.e. a constant. It might seem we have a problem because N can be arbitrarily large, and then how do we normalize our prior distribution?
   Hint: We can make use of our prior information about the maximal number of promoter N. In fact we know, thanks to previous experiences, that N can never be bigger than $N_{max} = 10000$ (we can also leave apart the number). So then we can choose a prior 3 and see how much the posterior depends on N_{max}. If we have enough data the posterior should depend much on the prior and so on N_{max} since it is dominated by the data.
   $$P(N|I) = \frac{1}{N_{\max}}\Theta(N_{\max} - N), \tag{3}$$
   At the end of our calculations we can check how much the outcome depends on our maximum $N_{\max}$. Typically, we will find that it does not depend much on this upper bound at all, as long as the upper bound is far from values that are indicated by the data.
   Hint:have a look at the posterior distribution for different values of n and k using the `plot` *command. The plot function is already implemented for you!*

4. Using the posterior distribution as an input, implement the `cdf` function that provides the cumulative distribution as an output.
   Hint: have a look at the cumulative distribution for different values of n and k using the `plot` *command.*

5. When asked to give a single best guess N_{guess} for a value N_{rand} randomly sampled from our probability distribution, there is generally no "correct" answer. In decision theory, our answer will depend on a loss function $L(N_{\text{guess}}, N_{\text{rand}})$ that characterizes the cost of guessing wrongly as a function of how different our guess is from the random sample. Mode, median and mean are most commonly used as a best guess and correspond to simple loss functions. Depending on the distribution, they may all be different from each other.

   - The *mode* of a distribution is the best guess when the loss function is independent of how wrong our guess is from the sampled value, i.e. $L = 0$ if $N_{\text{guess}} = N_{\text{rand}}$ and $L = 1$ otherwise. It corresponds to the most likely sampled value of N_{rand}. *Hint:The mode of a dataset D is where its PMF takes a maximum value, so you have to find the N which maximises $P(N|D,n,k)$.*
   - The *median* is the best guess if the loss function is proportional to how wrong our guess is, i.e. for $L(N_{\text{guess}}, N_{\text{rand}}) = |N_{\text{guess}} - N_{\text{rand}}|$. It corresponds to the value of N for which we are equally likely to sample a larger or smaller N_{rand}. *Hint:The median is the smallest N for which* $\text{CDF}(N|D,n,k) \geq 0.5$*; use the* `cdf` *function to find it.*
   - The *mean* of a distribution is the best guess for a quadratic loss function $L(N_{\text{guess}}, N_{\text{rand}}) = (N_{\text{guess}} - N_{\text{rand}})^2$ and can be interpreted as the value of N averaged over many random samples. It is given by
     $$\langle N \rangle = \sum_{N=0}^{N_{\max}} N P(N|D,n,k). \tag{4}$$

   Implement the function `mode_median_mean` that returns all three values for our posterior distribution.

6. Implement the `ppi` function that uses the cumulative distribution to calculate the symmetric 98% probability interval, i.e. $[N_{\min}, N_{\max}]$ such that $P(N < N_{\min}|D,n,k) \approx 0.01$ and $P(N > N_{\max}|D,n,k) \approx 0.01$.

Theoretical Questions

1. Determine the mean, median and modal values for the set: [3, 8, 10, 7, 5, 14, 2, 9, 8]
2. Derive the mean and variance of the Poisson distribution:

$$P(n|\lambda, t) = \frac{(\lambda t)^n}{n!} \exp(-\lambda t)$$

3. Derive formula 2 using Bayes theorem!