Set 5.

1. $p(X=x) = \frac{e^{-\lambda}\lambda^{x}}{x!}$

$$E(X) = \sum_{x=0}^{\infty} x \frac{e^{-\lambda}\lambda^{x}}{x!} = \sum_{x=1}^{\infty} x \frac{e^{-\lambda}\lambda^{x}}{x!}$$

$$= \sum_{x=1}^{\infty} \frac{e^{-\lambda}\lambda^{x}}{(x-1)!}$$

$$= \lambda e^{-\lambda} \sum_{x=1}^{\infty} \frac{\lambda^{x-1}}{(x-1)!}$$

$$= \lambda e^{-\lambda} \sum_{x=1}^{\infty} \frac{\lambda^{x}}{x!}$$

$$= \lambda e^{-\lambda} e^{\lambda}$$

$$= \lambda \quad ✓$$

2. $p(x=x|\lambda) = \frac{e^{-\lambda}\lambda^{x}}{x!}$

A. $L(\lambda|x) = p(x=x_1|\lambda)\,p(x=x_2|\lambda) \dots p(x=x_n|\lambda)$

$$= \frac{e^{-\lambda}\lambda^{x_1}}{x_1!} \cdot \dots \cdot \frac{e^{-\lambda}\lambda^{x_n}}{x_n!} = \frac{e^{-n\lambda}\lambda^{x_1+x_2+\dots+x_n}}{x_1!\,x_2!\,x_3!}$$

$$= \frac{e^{-n\lambda}\lambda^{\Sigma x_i}}{\prod_{i=1}^{n} x_i!}$$

$$\ln L(\lambda|x) = -n\lambda + \sum_{i=1}^{n} x_i \ln\lambda - \ln\left(\prod_{i=1}^{n} x_i!\right)$$

B. $\frac{d \ln L(\lambda|x)}{d\lambda} = -n + \frac{\Sigma x_i}{\lambda} = 0$

$$\boxed{\hat{\lambda} = \frac{\sum_{i=1}^{n} x_i}{n}}$$

3. $X, Y \sim \text{Geom}(p)$

$p(X=k) = (1-p)^{k-1} p$, $k = 1, 2, 3, \ldots$

a. Show $P(X+Y=K) = (k-1)(1-p)^{k-2} p^2$ for $k = 2, 3, \ldots$

$$p(X+Y=k) = \sum_{i=1}^{k-1} p(X=i, Y=k-i)$$
$$= \sum_{i=1}^{k-1} p(X=i)\,p(Y=k-i)$$
$$= \sum_{i=1}^{k-1} p(1-p)^{i-1} p(1-p)^{k-i-1}$$
$$= p^2 (1-p)^{k-2} p^2 \sum_{i=1}^{k-1} 1$$
$$= p^2 (1-p)^{k-2} (k-1) \checkmark$$

B. Z is negative hinomial

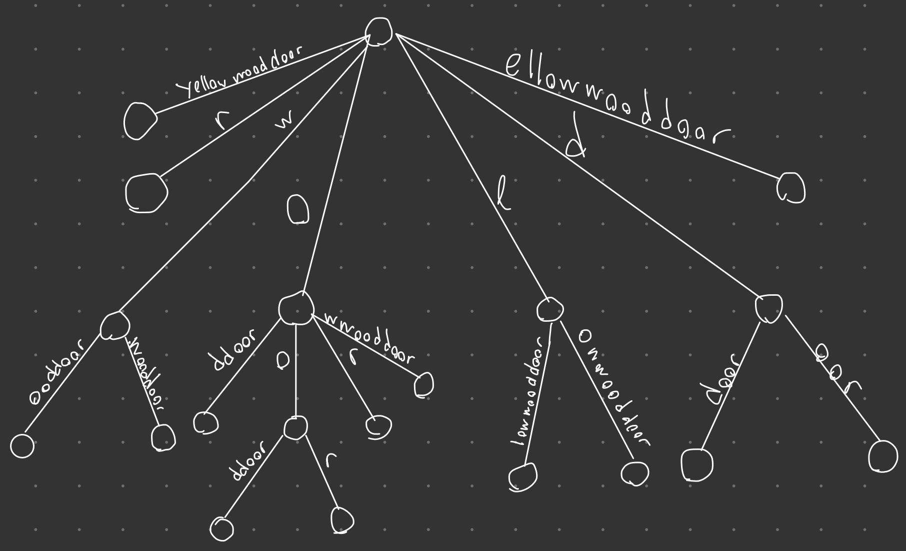
$$p(Z=z) = \binom{z-1}{r-1} p^r (1-p)^{z-r}, \quad z = r, r+1 \ldots$$

Show $X+Y \sim \text{negbinom}(2, p)$

$$\overset{r=2}{p(Z=z)} = \binom{z-1}{1} p^2 (1-p)^{z-2}$$
$$= (z-1) p^2 (1-p)^{z-2} \quad = (k-1)(1-p)^{k-2} p^2 \quad \checkmark$$

for $z = 2, 3, \ldots$ for $k = 2, 3, \ldots$

4.


Yellowwooddoor
r
w
o
ellowwooddoor
d
l
ooddoor
wooddoor
ddoor
o
r
wwooddoor
ddoor
r
lowwooddoor
owwooddoor
door
oor

problem3

February 13, 2023

Bi/Be/Cs 183 2022-2023: Intro to Computational Biology TAs: Meichen Fang, Tara Chari, Zitong (Jerry) Wang

Submit your notebooks by sharing a clickable link with Viewer access. Link must be accessible from submitted assignment document.

Make sure Runtime → Restart and run all works without error

HW 5 (Midterm) Problem 3c. (8 points)

The sum of n i.i.d. geometric random variables with parameter p has a negative binomial distribution with parameters $r = n$ and p.

Draw 10,000 samples from a negative binomial distribution with $r = 10$, $p = 0.5$ and plot your sampled values in the form of a histogram.

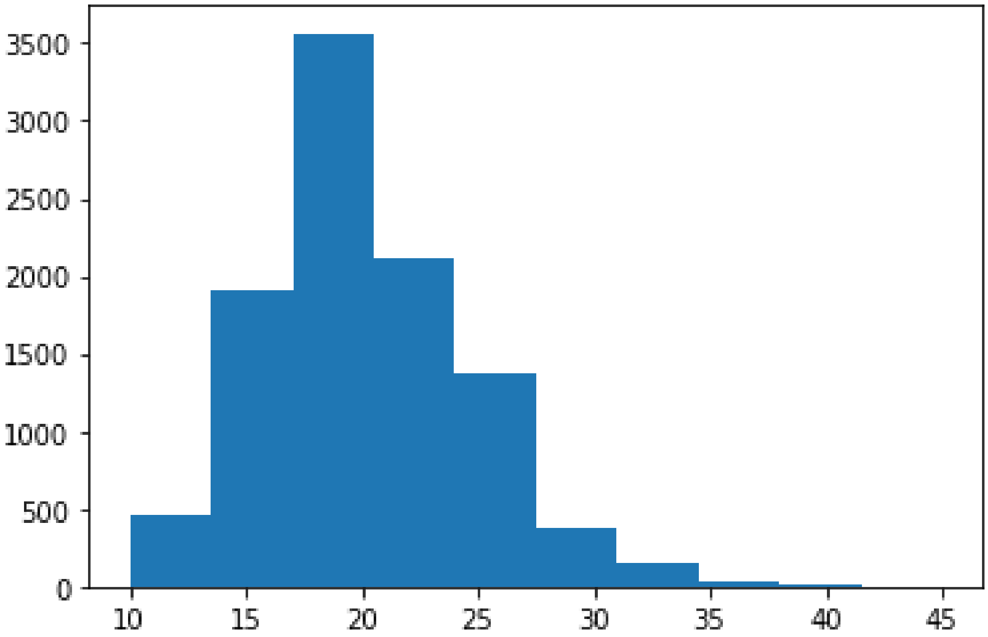
You may **not** use functions that directly generate samples from a negative binomial distribution such as `numpy.random.negative_binomial`. (Hint: a useful function is `numpy.random.geometric`)

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
```

```
In [2]: n = 10000
        r = 10
        p = 0.5
```

```
In [3]: samples = np.sum(np.random.geometric(p, (r, n)), axis=0)
        plt.hist(samples)
```

```
Out[3]: (array([ 468., 1901., 3564., 2116., 1379.,  373.,  157.,   28.,   10.,
                   4.]),
         array([10. , 13.5, 17. , 20.5, 24. , 27.5, 31. , 34.5, 38. , 41.5, 45. ]),
         <BarContainer object of 10 artists>)
```


3500
3000
2500
2000
1500
1000
500
0
10
15
20
25
30
35
40
45

problem5

February 13, 2023

Bi/Be/Cs 183 2022-2023: Intro to Computational Biology TAs: Meichen Fang, Tara Chari, Zitong (Jerry) Wang

Submit your notebooks by sharing a clickable link with Viewer access. Link must be accessible from submitted assignment document.

Make sure Runtime → Restart and run all works without error

HW 5 (Midterm) Problem 5

For this problem you will be exploring various models which can be used to describe count data i.e. the gene-count matrices we use in single-cell.

Single-cell gene counts, which describe stochastically sampled, discrete measurements of UMI counts, are often modeled as being generated from a negative binomial (or Gamma-Poisson) distribution. However, there is a common assumption that droplet-based methods for single-cell RNA seq incur an overabundance of zeros (more zero counts) than would be predicted by random sampling. Thus it is also common to see single-cell data modeled with zero-inflated negative binomials (the ZINB distribution, with an extra parameter for the probability of zero counts).

You will explore how these assumptions and models fit to real datasets.

```
In [1]: #To run a code cell, select the cell and hit Command/Ctrl+Enter or click the run/play
        #Click Insert --> Code Cell or the '+ Code' option to insert a new code cell
```

```
In [2]: #Click Insert --> Text Cell or the '+ Text' option to insert a cell for text as below
```

```
In [3]: # This is  used to time the running of the notebook
        import time
        start_time = time.time()
```

Text here for descriptions, explanations, etc

0.1 Import data and install packages

```
In [4]: !pip --quiet install anndata
```

```
In [5]: import numpy as np
        import scipy.io as sio
        import pandas as pd
        import matplotlib.pyplot as plt #Can use other plotting packages like seaborn

        import anndata
        from scipy import optimize
        from scipy.special import gammaln
        from scipy.special import psi
        from scipy.special import factorial
        from scipy.optimize import fmin_l_bfgs_b as optim

In [6]: # ! allows you to run commands in the command line, as you would in your normal termin

In [7]: # Download control sample from indrops platform
        # File format is h5ad

        import requests
        from tqdm import tnrange, tqdm_notebook
        def download_file(doi,ext):
            url = 'https://api.datacite.org/dois/'+doi+'/media'
            r = requests.get(url).json()
            netcdf_url = r['data'][0]['attributes']['url']
            r = requests.get(netcdf_url,stream=True)
            #Set file name
            fname = doi.split('/')[-1]+ext
            #Download file with progress bar
            if r.status_code == 403:
                print("File Unavailable")
            if 'content-length' not in r.headers:
                print("Did not get file")
            else:
                with open(fname, 'wb') as f:
                    total_length = int(r.headers.get('content-length'))
                    pbar = tnrange(int(total_length/1024), unit="B")
                    for chunk in r.iter_content(chunk_size=1024):
                        if chunk:
                            pbar.update()
                            f.write(chunk)
                return fname

        download_file('10.22002/xsret-sb590','.gz')
```

```
/tmp/ipykernel_11951/3748164775.py:21: TqdmDeprecationWarning: Please use `tqdm.notebook.trange
  pbar = tnrange(int(total_length/1024), unit="B")


  0%|          | 0/19383 [00:00<?, ?B/s]
```

```
Out[7]: 'xsret-sb590.gz'

In [8]: !gunzip *.gz
        !mv xsret-sb590 Klein.h5ad

In [9]: indrops = anndata.read('Klein.h5ad')
In [10]: indrops
```

```
Out[10]: AnnData object with n_obs Œ n_vars = 953 Œ 25435
             obs: 'total_counts'
             var: 'empirical_mean', 'empirical_variance', 'empirical_zero_fraction', 'ml_mean'
             uns: 'global_dispersion', 'name'
```

Use the function below for b).

```
In [82]: # X = numpy array of the data (e.g. 1D array with all the counts for one gene)
         # initial params is a numpy array representing the initial values of
         # size and prob parameters
         # Returns: Dict with 'r' and 'p' fits
         def fit_nbinom(X, initial_params=None):
             ''' This code is adapted from https://github.com/gokceneraslan/fit_nbinom
             '''
             infinitesimal = np.finfo(float).eps

             def log_likelihood(params, *args):
                 r, p = params
                 X = args[0]
                 N = X.size

                 # MLE estimate based on the formula on Wikipedia:
                 # http://en.wikipedia.org/wiki/Negative_binomial_distribution#Maximum_likeliho
                 result = np.sum(gammaln(X + r)) \
                     - np.sum(np.log(factorial(X))) \
                     - N * (gammaln(r)) \
                     + N * r * np.log(p) \
                     + np.sum(X * np.log(1 - (p if p < 1 else 1 - infinitesimal)))

                 return -result

             if initial_params is None:
                 # reasonable initial values (from fitdistr function in R)
                 m = np.mean(X)
                 v = np.var(X)
                 size = (m ** 2) / (v-m) if v > m else 10

                 # convert mu/size parameterization to prob/size
                 p0 = size / ((size + m) if size + m != 0 else 1)
                 r0 = size
                 initial_params = np.array([r0, p0])

             bounds = [(infinitesimal, None), (infinitesimal, 1)]
             optimres = optim(log_likelihood,
                              x0=initial_params,
                              args=(X,),
```