Please provide complete and well-written solutions to the following exercises.

Due September 5, 4PM PST, to be uploaded as a single PDF document to brightspace. It is acceptable to instead upload a Jupyter Notebook, assuming you write in complete sentences where appropriate, and format your responses to be easily readable (i.e. if you only submit one big block of code with nothing written about what you did, then many points will be deducted from your score).

Homework 1

Exercise 1. Download and install this Python on your personal computer. Specifically, download Anaconda (a popular Python distribution platform) from here: https://www.anaconda.com/download. Instructions for downloading and installing this software can be found: here.

Exercise 2. As needed, refresh your knowledge of proofs and logic by reading the following document by Michael Hutchings: http://math.berkeley.edu/∼hutching/teach/proofs.pdf

Exercise 3. As needed, take the following quizzes on logic and set theory:

http://scherk.pbworks.com/w/page/14864234/Quiz%3A%20Logic
http://scherk.pbworks.com/w/page/14864241/Quiz%3A%20Sets

(These quizzes are just for your own benefit; you don't need to record your answers anywhere.)

Exercise 4. In Python, do the following:

- Perform the following operation, and report the result:

$$\begin{pmatrix} 2 & 3 \\ 4 & 5 \end{pmatrix} \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + 4 \begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix}.$$

- Plot the function $f(x) = x^3 + e^x$ for x values in the interval $[0, 3]$.
- Describe the output of the following program.

```
x = 1
while x != 0:
    x = x / 2
    print(x)
```

Exercise 5. In Python, the logical value `True` represents true, and the logical value `False` represents a false statement. For example, `3<5` evaluates to a `True`, and `5<3` evaluates to `False`.

Python's logical operations include: `and` for logical and, `or` for logical or, `not` for logical negation. Python's relational operations include: `<` for less than, `<=` for less than or equal to, `==` for equality, `!=` for not equality. (The command `&` is logical and that also works for arrays. The command `|` is logical or that also works for arrays.)

- Compute the following expression by hand, and in Python:

  `( (2<3) and (4<2) ) or not(4<8).`

- Describe the output of the following program.

```
x=1
while (x<5) and not(x< -5):
    x = x + np.random.random()
    print(x)
```

- Logical operations can also apply to vectors, using Numpy functions. Compute the output of the following program by hand, and in Python:

```
x = np.array([False, True, False])
y = np.array([True, True, False])
z = np.array([False, False, True])
a = (x & y) | z
print(a)
```

(Note: Numpy's logical arrays can also be summed, where True acts as 1 and False acts as 0, so the sum of [True, True, False] would be 2.)

Exercise 6. Using Python, verify that its random number generator agrees with the law of large numbers and central limit theorem. For example, the command `np.random.random(10**7)` generates 10^7 samples of a number that is equally likely to have any value in the interval $(0, 1)$ [rounded to the nearest floating point value]. You should then average these values, using e.g. the `np.mean` command, and check how close the average is to $1/2$. Then, make a histogram of samples using the `hist` function from `matplotlib.pyplot`, and check how close the histogram is to a Gaussian function of the form

$$t \mapsto \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{t^2}{2\sigma^2}}, \qquad \forall\, t \in \mathbf{R}.$$

(More specifically, examine the histogram of the mean of the columns of the random matrix `((np.random.rand(10**3, 10**4)) - 1/2 )/np.sqrt(10**-3)` with say 100 bins, using e.g. the command `np.mean(..., axis=0)`. (Which value of $\sigma > 0$ gives you the closest fit between the histogram and the Gaussian function?)

(It is okay if you just try a few σ values and then pick your favorite one. The last part of this question does not need an extremely precise answer. This question is just meant to explore the concept of a bell curve, rather than to choose the absolute best fit for your histogram.)

(Above we intentionally chose our Gaussian density to have mean zero.)

Comment. Your answer to the above exercise should include the histogram you generated. To save a figure file in Python, you can use the command `plt.savefig('filename.pdf')`

to save the current Python figure as a PDF file called `filename.pdf` (assuming you used the command `import matplotlib.pyplot as plt`).

Exercise 7. Let n be a positive integer. For any $x = (x_1, \ldots, x_n), y = (y_1, \ldots, y_n) \in \mathbf{R}^n$, define

$$\langle x, y \rangle := \sum_{i=1}^{n} x_i, y_i, \qquad ||x|| := \sqrt{\langle x, x \rangle}.$$

Denote the unit sphere in $\mathbf{R}^n$ as

$$S^{n-1} := \{x \in \mathbf{R}^n : \ ||x|| = 1\}.$$

Let $v^{(1)}, \ldots, v^{(m)} \in S^{n-1}$. Show that there exists $1 \leq i < j \leq m$ such that

$$\langle v^{(i)}, v^{(j)} \rangle \geq 1 - \frac{64}{m^{2/n}}.$$

That is, if m is large enough, two of the unit vectors will be highly correlated with each other. (The inner product of unit vectors can be interpreted as a correlation.)

(Hint: use the pigeonhole principle and argue by contradiction, i.e. derive a contradiction by assuming that $\langle v^{(i)}, v^{(j)} \rangle < 1 - z$ for all i, j. Note that $\sqrt{z} \leq \cos^{-1}(1-z)$ for any $0 < z < 1$, so each pair of vectors has angle larger than $\sqrt{z}$. Suppose each pair of vectors has angle larger than 2ε. Then the union $\cup_{i=1}^{m} \{x \in S^{n-1} : \ \cos^{-1}\langle x, v^{(i)} \rangle \leq \varepsilon\}$ is disjoint, so the volume of S^{n-1} is at least the sum of the volumes of the sets on the right. Using known volume estimates on the sphere, this means that $m \int_0^{\varepsilon} \sin^{n-1} \theta d\theta \leq \int_0^{\pi} \sin^{n-1} \theta d\theta$. Using another known estimate, the latter integral is at most $\sqrt{2\pi}/\sqrt{n}$. Also, $\int_0^{\varepsilon} \sin^{n-1} \theta d\theta \geq \int_0^{\varepsilon} (\theta/2)^{n-1} d\theta = 2^{-n+1} \varepsilon^n / n$, so $m \leq \sqrt{2\pi}\sqrt{n} 2^{n-1} \varepsilon^{-n}$. Setting $2\varepsilon = \sqrt{z}$, we get $m \leq \sqrt{2\pi}\sqrt{n} 4^n 2^{-1} z^{-n/2} \leq 2\sqrt{n}(4/\sqrt{z})^n$, so $z = 64 m^{-2/n}$ gives a contradiction.)

Exercise 8. Recall that two real-valued random variables $X, Y : \Omega \to \mathbf{R}$ with nonzero variance have correlation defined by

$$\mathrm{Corr}(X, Y) := \mathbf{E}\Big(\frac{X - \mathbf{E}X}{\sqrt{\mathrm{Var}(X)}} \frac{Y - \mathbf{E}Y}{\sqrt{\mathrm{Var}(Y)}} \Big),$$

where $\mathbf{E}$ denotes expected value and $\mathrm{Var}(X) := \mathbf{E}(X - \mathbf{E}X)^2$ denotes variance. Put another way, if we define the inner product $\langle W, Z \rangle := \mathbf{E}WZ$ and the norm $||W|| := \langle W, W \rangle^{1/2}$ for any real-valued random variables W, Z, then $\left\| \frac{X - \mathbf{E}X}{\sqrt{\mathrm{Var}(X)}} \right\| = 1$, so we can interpret the correlation of X and Y as an inner product of vectors of unit length, as in Exercise 7:

$$\mathrm{Corr}(X, Y) = \Big\langle \frac{X - \mathbf{E}X}{\sqrt{\mathrm{Var}(X)}}, \frac{Y - \mathbf{E}Y}{\sqrt{\mathrm{Var}(Y)}} \Big\rangle.$$

In this exercise we will consider a sample space $\Omega = \{1, \ldots, n\}$ with $n = 24$ points and with the uniform probability law on Ω, so that $\mathbf{E}X = \frac{1}{n} \sum_{i=1}^{n} X(i)$.

We will compute the correlation of X and Y, where X is the chocolate consumption by country in kg per year for 24 different countries, and Y is the number of nobel laureates awarded to each country divided by its population. (If the nobel prize website lists a country

for a prize winner, then that country is counted once on the data list on wikipedia.) I copied the data below, together with sources (with country names removed).

Chocolate consumption by country in kg per year

```
8.8, 8.1, 7.9, 7.9, 7.6, 6.6, 6.5, 5.8, 5.7, 5.6, 5.4, 5.2, 5.1,
5, 4.9, 4.9, 4.9, 4.8, 4.4, 4.3, 1.2, 1.2, .9, .1
```

Nobel Laureates by country

```
25, 25, 115, 11, 137, 34, 0, 14, 18, 11, 5, 0, 22,
3, 14, 14, 6, 30, 411, 75, 1, 29, 11, 8
```

Populations by country (in millions)

```
9, 9.2, 84.7, 5.3, 67.6, 10.5, 1.4, 5.6, 37.6, 9.2, 5.6, 5.4, 18,
5.3, 6, 27.3, 10.9, 146.2, 335.9, 68.4, 203.1, 123.9, 62, 1409.7
```

What correlation do you find for X and Y? A paper here claimed to find a correlation of .791 but with different data, I found a smaller correlation.

In any case, with Exercise 7 in mind, recall that if you make a list of enough quantities of fixed length (in our case length $n = 24$), two of those quantities will be highly correlated with each other. In other words, finding two such quantities that happen to be correlated with each other does not mean anything.

For some other amusing examples of this, see this website.

Exercise 9 (Optional)**.** Read some articles about errors in numerical computing with some serious consequences such as

- the Pentium FDIV bug (wikipedia)
- assorted disasters
- failure to convert from metric to imperial

Exercise 10. Let $\mathcal{F}$ be the set of all positive double precision floating point numbers (except for `NaN`s and `Inf`s), that have the exponent `1023` (in their hexadecimal representation in Python). (For example, `np.finfo('d').max` is in $\mathcal{F}$, since)

- How many elements are in $\mathcal{F}$? That is, what is the cardinality $|\mathcal{F}|$ of $\mathcal{F}$.
- What fraction of elements of $\mathcal{F}$ are in the interval $[2^{1023}, 2^{1024})$?
- What fraction of elements of $\mathcal{F}$ are in the interval $[2^{1023}, \frac{3}{2}2^{1023})$?
- Using e.g. Numpy's `random` function, write a program that estimates the fraction of $x \in \mathcal{F}$ that satisfy the expression `x * (1/x)==1`. (It would take a pretty long time to check how many elements of $\mathcal{F}$ satisfy this equation, so you should not do that.)

Warning: Numpy's `random` function tries to find a uniformly random chosen number in the interval $(0, 1)$ and then round it to the nearest floating point number. This operation is

different than choosing a floating point number uniformly over all (positive) floating point numbers with a fixed exponent. (This is the point of the second and third items of this exercise.) For this reason, your answer to the last part of the question might be different from the output of the program:

```
x = np.random.rand(1000)
sum(x* (1/x) == 1) / 1000
```

Exercise 11. Do the following plot in Python

```
import matplotlib.pyplot as plt
x = np.arange(.988, 1.012, .0001)
y = x**7 - 7*x**6 + 21*x**5 - 35*x**4 + 35*x**3 - 21*x**2 + 7*x - 1
plt.plot(x, y)
plt.show()
```

This is the function $y(x) = (x-1)^7$ for $x \in [.988, 1.012]$. Does the plot look like a polynomial? Explain why or why not.