QNT 402 Quantum Information
Homework 6

1. [35] *MLE for qubit lifetime.* Suppose that we have a new qubit that is great, but unfortunately it explodes after some time t. We have a bunch of them and we're going to measure the time from their creation until they explode. (Obviously, IBM wants to know this before they buy them from us!) We know based on some other work that the probability of the qubit surviving until time t is given as $P(t;\tau) = \frac{1}{\tau}e^{-\frac{t}{\tau}}$, where τ parameterizes the distribution. (Let's call it the qubit lifetime).
   a.) [5] Find the expectation value of t.
   b.) [5] Find the variance of t.
   c.) [10] Find the maximum likelihood estimator τ_e for the parameter τ.
   d.) [15] We can simulate a likelihood function as follows.
      1. Generate a list of random numbers that are distributed according to an exponential distribution with $\tau = 1\ s$ – perhaps 1000 samples long.
      2. Find the mean of that list.
      3. Repeat 1 and 2 over and over again – perhaps 1000 times.
      4. Make a histogram of the means that you found.

   Using this protocol, make a histogram and normalize it correctly to simulate the likelihood function. What shape is this distribution and how does it relate to the answers you found in parts (a-b)?

2. [55] *Pandas love data.* In this problem we'll use Pandas to process some data. Pandas is a library for python that provides a database structure for storing data and a set of tools for analyzing that data. See the official site at: https://pandas.pydata.org/docs/index.html. We'll analyze a famous data set that was used in a Washington Post article about climate change. The full data set is available at: https://github.com/washingtonpost/data-2C-beyond-the-limit-usa. From this repository, we will need two files. They are:
      1. https://github.com/washingtonpost/data-2C-beyond-the-limit-usa/blob/main/data/processed/climdiv_national_year.csv -- This file has the average temperature of different US states from 1895 to 2019.
      2. https://github.com/washingtonpost/data-2C-beyond-the-limit-usa/blob/main/data/raw/noaastate.txt -- This file list the identifying codes ('fips') for each state according to some NOAA identifier.

   A Pandas structure allows you to pull out the data you want from a database in a number of ways. In this quick primer we'll play with a few of them.
   a.) [0] Read in the data using Pandas csv_reader() function to a variable called something like rawData.
   b.) [5] You can view the Pandas data in a number of ways. First simply have python display the data. Jupyter tends to do this nicely. Next, use the label feature of Pandas to view the Years in the database by the command rawData['year']. Now, plot rawData['year'], why does it oscillate?
   c.) [5] Now, let's look at data for California. Find the fips identifier for California from the NOAA file. As we've learned from the above, the command rawData['fips']

returns a list of the fips. Thus, the command rawData['fips' == 4] returns a list of True and False, where the entries that are True represent data for California. The Pandas .loc[] property takes this list of True and False and returns only the part of the database that correspond to True. Thus, rawData.loc[(rawData['fips' == 4])] gives on the data for California. Use this to plot the temperature in California over all available years.

d.) [5] Add the plot of the temperatures in New York over all available years to the plot you just made.

e.) [5] What is the mean and standard deviation of all temperatures in the data base? (Use the Pandas functions here instead of numpy as etc.)

f.) [5] What is the mean and standard deviation of all temperatures for California in the data base? (Use the Pandas functions here instead of numpy as etc.)

g.) [10] Plot the average temperature over all states in the database from 1895 until 2019.

h.) [5] Find the mean and standard deviation of the first 30 years of the data set.

i.) [5] Find the mean and standard deviation of the last 30 years of the data set.

j.) [10] Using a Z-score, test if the average temperature of the last 30 years is higher than the average temperature in the first 30 years at the 95% and 90% confidence levels. A Z-score assumes things follow a normal distribution, so this might feel a bit worrisome since maybe there are dynamics going on in the temperature and just taking a mean and standard deviation doesn't capture that. If you feel that way, you're right. This is weird but in the real world there is sometimes no good answer. Therefore, it's important to state clearly what you did so others can interpret your result accordingly.

(Fun tip: I found a nice tutorial using this exact data set! It's available at: https://www.storybench.org/exploring-climate-data-using-the-python-libraries-matplotlib-and-pandas/ and could be useful for getting your feet wet.)

3. [70] *Fitting in the lab.* It is often the case in the lab that you pull some data from a device like an oscilloscope and you want to extract some unknown parameters as well as have some sort of estimate of your confidence in them. In this problem, we will simulate some 'fake data' and fit it using scipy's curve_fit() to get a feel for what's good and what's not good about this process. (We'll study this because if you work in a lab you will do this often, even if it's not formally correct.).

a.) [15] First, we need to make some fake data. Let's assume we have a voltage measured by an oscilloscope that is ideally (in Volts) $V(t) = 1\ \sin \omega t + 0.33$, where $\omega = 2\pi \times 1$ MHz. Suppose, we sample this voltage for 10 μs and record 300 evenly space data points. Now, assume at each point in time there is noise added to this signal that is Gaussian distributed with a mean of 0 V and a standard deviation of 0.5 V. Generate the data sample and plot it on the same graph as the ideal signal.

b.) [10] Suppose that in the lab after we recorded this data, we want to know the value of the offset (i.e. the 0.33 V number) and we choose to extract this by fitting the function $V(t) = \sin \omega t + B$, where B is our fit parameter. That is, we are assuming we have perfect knowledge of the amplitude of the sine wave, its frequency, and the timing and we are just fitting to extract the offset. Use sci_pi's curve_fit() to fit to the data and extract B. Compare the value of B you find to the true value.

Compare the extracted standard deviation of parameter (found from the covariance matrix) with what you expect.

c.) [5] You should have found in part (b) that curve_fit() returned a value for the standard deviation of B that agrees with your estimate for the standard error of the mean of the Gaussian you chose. How can this be? Remember, we never passed any information to curve_fit about the error bar on each point (that can be done using curve_fit's input "sigma"). Given that, how did it know how to correctly estimate the confidence region?

d.) [15]cNow, assume that we didn't know the amplitude of the sine wave or the frequency and fit the data with the function $V(t) = A \sin 2\pi f t + B$ to extract $A, f,$ and B. (Hint: Often when fitting oscillating functions it's important to give your fitting algorithm an initial guess for the parameters so it doesn't get stuck in a local minimum.) Compare the extracted parameters to the true values you used to generate the data. Are the extracted parameters reasonable given the expected error confidence interval?

e.) [5] Look at the off-diagonal elements in the covariance matrix you found in part (d). Explain the signs of these elements.

f.) [20] Now, let's assume that we have a different type of error. Instead of adding a random offset at each time step, let's assume there was a random step in the phase of the sine wave at each time. Assume that random step in phase is Gaussian-distributed about a mean of zero with a standard deviation of $\pi/10$. Generate the 'fake data' and repeat part (d) and (e).

Disclaimer: As we saw in lecture, using the χ^2 to set confidence intervals (which is how most algorithms work) is only straightforward for fitting one linearly dependent parameter. So, using curve_fit for parts (d-f) is dodgy. However, the point of this problem is to see that if you don't take the error bounds too seriously, one can use these convenient algorithms to get good parameter estimates. That said, if the results are important (e.g. a publication, etc.) then one needs to be more careful. In general, this means running a Monte Carlo simulation of your experiment and using e.g. the Feldman-Cousins algorithm.

4. [15] *Covariance with a sum*. Suppose we have a function $f(x,y) = x + y$. Given the elements of the covariance matrix V, find the variance in f.