Lab 3

Ondrea Robinson

October 27, 2019

In this lab, we practice asking statistical questions. This process consists of two steps:

1. Writing down in words *precisely* the question you are trying to ask.
2. Converting our precise English word question to a mathematical expression.

So in this lab our process will be to write each question, express the question mathematically and then solve the expression numerically.

1 Problem 1

In this problem we're looking at temperature data associated with an experiment. For the experiment to work the temperature should be around 12K with fluctuations up to 0.4K (standard deviation). Sometimes the thermal control systems malfunction however and we get false or anomalously high or low readings. We definitely want to identify and throw out all the data when the thermal control system was not working (and the temperature was truly off from nominal). While it is possible to have an error in the thermometry such that the true temperature was fine, and we just had a wonky reading, in an abundance of caution we want to throw those values out too

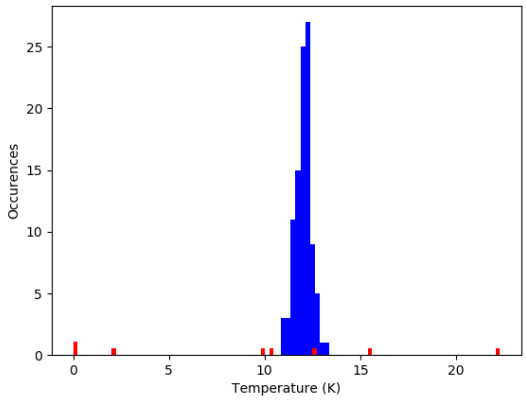
First we'll simulate a bit of data. We'll create an array of Gaussian distributed data points within 0.4K of 12K then throw in several bad values $[10.0, 10.3, 2.1, 0.0, 0.0, 15.6, 22.3, 12.7]$, all either too low or too high. See below for implementation in python:

```
goodPoints = stats.norm.rvs(12.0, 0.4, 100)
x = np.arange(6, 18, 1000)
badPoints = np.array([10.0, 10.3, 2.1, 0, 0, 15.6, 22.3, 12.7], dtype='float')

figure(1)
plt.hist(goodPoints, color='b')
plt.hist(badPoints,bins=100, density=True, color='r')
plt.xlabel("Temperature (K)")
```

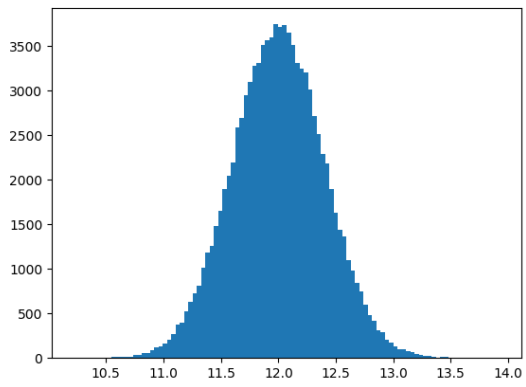
```
plt.ylabel("Occurences")
x = np.arange(6, 18, 1000)
```

. For this example, the outlier points on the histogram are colored red and we're only displaying 100 data points in order to make the plot more readable.



This is another plot of the data with 100k values using the code:

```
goodPoints = stats.norm.rvs(12.0, 0.4, 100000)
figure(2)
plt.hist(goodPoints, bins=100)
plt.xlabel("Temperature(K)")
plt.ylabel("Occurences")
plt.show()
```



1.1 A

Our goal is to identify the bad data points and eliminate them from our data set. Since we know the range of our bad data, we can design a threshold that excludes as much bad data as possible, while keeping the maximum amount of good data. To phrase it as a statistical question:

If I take data from within [10.4, 12.6] how much good data did I keep (true-positive), how much bad data did I discard (false-negative), and how much is incorrectly omitted (true-negative) or included (false-positive)?

We've chosen this threshold because it includes the most good data while also being above and below the closest data points near 12 (10.3 and 12.7). Now let's express this in code. We want to keep track of the data we include or discard from the bad set and the data we include or discard from the good set. Since the data sets are already separated, we can run them through the same test and record the values for each. This is a test of the 'good' data:

```
lower = 10.4
upper = 12.6
trueNeg = 0
truePos = 0
for i  in range(len(goodPoints)):
data = goodPoints[i]
if (data < lower) or (upper < data):
    trueNeg = trueNeg + 1
else:
    truePos = truePos + 1
print(truePos)
print(trueNeg)
```

And this is a test of our bad data:

```
lower = 10.4
upper = 12.6
falseNeg = 0
falsePos = 0
for i  in range(len(badPoints)):
data = badPoints[i]
if (data < lower) or (upper < data):
    falseNeg = falseNeg + 1
else:
    falsePos = falsePos + 1
print(falsePos)
print(falseNeg)
```

With the values we get out we can make a truth table:

	True	False
Positive	93212	0
Negative	6788	8

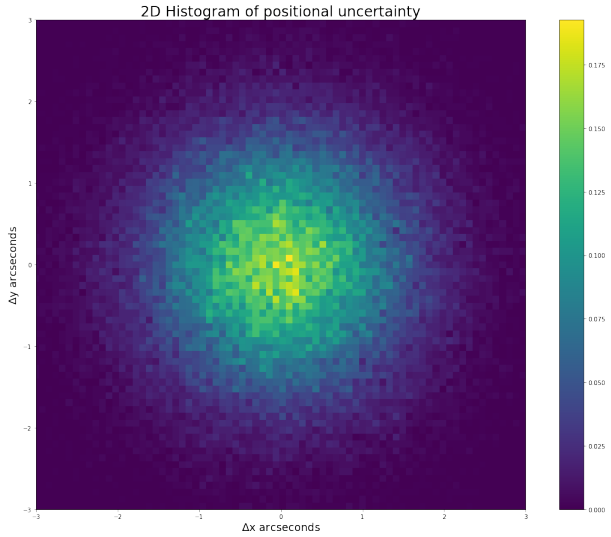
In this case we knew the values of our bad data and were able to design the perfect threshold in order to eliminate all false positives. But what if we asked the same statistical question without knowing our data set? Say we chose a threshold of [11.2, 12.8] (2σ below and above 12) we could rerun our tests and our truth table would now look like:

	True	False
Positive	95473	1
Negative	4527	7

We've minimized more of the true-negatives but increased the false positives. How the threshold is chosen depends on the end goal. If we didn't care about bad data getting in as long as we kept all of our good data, we wouldn't need a very strict threshold. We could just find the minimum and maximum of our good data set and set those as our lower and upper limits. If we mostly cared about excluding any bad data we could just set the range like we did in the first example. Most of the time however we're not sure what our 'bad' data will look like and choose our threshold based on the demands of the experiment.

2 Problem 2

In this example we're looking for asteroids. If we look at the alignment of stars on subsequent images, they don't perfectly align due to atmospheric and instrumental effects (even ignoring proper motion). The resulting distribution is two dimensional, and for this lab let's assume it is a 2D Gaussian with 1 arcsecond RMS. Or said another way, if I histogram how far all the (stationary) stars appear to have moved I get something like:



See code in Lab3 Html. This 2D Gaussian corresponds to a one dimensional Rayleigh distribution. To put our statistical question into words:

If I have a Rayleigh distributed background with rate parameter 1arcsecond, and my 'signal-like' definition corresponds to larger distances, what's the minimum distance measurement that would yield a significance of 5σ?"

In this case, we recognize the Rayleigh distribution as the one dimensional projection of the 2D Gaussian. First let's put our statistical question into a math question: *What's the minimum value that has a probability of $\frac{1}{3.5*10^6} = 5\sigma$ on a Rayleigh distributed pdf?* So to find the distance we need we can use the ppf function in python with the following code:

```
prob = 1 - (1/(3.5*(10**6)))
meas = stats.rayleigh.ppf(prob)
print(meas)
```

After running our code the variable meas = 5.48 arcseconds. We're looking at the minimum value that would give us $1-5\sigma$, the probability under most of the curve.

3 Problem 3

A key background for detecting gamma rays is cosmic ray bombardment. Cosmic rays are charged particles that come from all directions from our perspective on earth. The sun and the moon block cosmic rays in patches of the sky. Assume in a moon sized patch on the sky we normally have a cosmic ray rate of 1 cosmic ray per minute (arrivals are random in time). If we can observe where the moon is for 8 hours per night (not too close to the horizon) and we

observe for 15 days and see 6800 cosmic rays, what is the significance of our moon shadow detection?

3.1 Part I

First let's phrase our problem as a statistical question. We want to know what the significance of our detection is. Our detection is 6800 cosmic rays in $15 * 8 * 60 = 7200$ minutes. This is a rate of 0.944 cosmic rays detected per minute as compared to the average rate of 1 per minute. We're looking over a period of 7200 minutes and have to compare our measurement value to the sum of all the 1 minute distributions in our interval. But this isn't a continuous interval since we have to do multiple trials for each night. Since we're looking at the number of events per time interval we assume the background (given) distribution is Poisson. So our question becomes *What is the significance of a measurement of 6800 cosmic rays over the course of 15 trials of 480 minutes if the average rate for one minute is a detection of 1 ray per minute on a Poisson distribution?*

3.2 Part II

Now let's actually think through the mathematics of this problem. We need to compare a measurement over 480 minutes to a given, typical distribution for 1 minute then multiply this distribution by our trials factor, 15. We need to find the average Poisson distribution for 480 minutes to compare our measurement against. We do this as we've done before by finding the convolution of our distributions. Then once we have the correct distribution we can integrate to find the probability of getting our measurement value or lower (cdf(value)). We're looking for lower values because this more significant of a measurement (cosmic rays*blocked*). Once we have our probability we then plug this into our ppf function to find the significance against a normal distribution. Recall this gives us a gauge of how likely our measurement value originated from the background rather than the source we want to measure.

3.3 Part III

Now that we've discussed it, let's implement. It's difficult to create a Python distribution outside of the SciPy package options, so rather than setting up a new distribution we can use the fact that our resulting distribution after convolving will be a Poisson distribution with average 480*(original average=1) for one night. Our x values will be rescaled since we're going to average (to find the rate) so our input value will also be scaled ($0.944 * 480 = 453$). Then we'll multiply the probability by 15. The resulting python computation looks like:

```
oneNightProb = stats.poisson.cdf(453, 480)/480
sigma = stats.norm.ppf(15*oneNightProb)
print(sigma)
```

We get out $\sigma = 2.7$.