Frequentist Inference Case Study - Part A

1. Learning objectives

Welcome to part A of the Frequentist inference case study! The purpose of this case study is to help you apply the concepts associated with Frequentist inference in Python. Frequentist inference is the process of deriving conclusions about an underlying distribution via the observation of data. In particular, you'll practice writing Python code to apply the following statistical concepts:

- the *z*-statistic
- the *t*-statistic
- the difference and relationship between the two
- the Central Limit Theorem, including its assumptions and consequences
- how to estimate the population mean and standard deviation from a sample
- the concept of a sampling distribution of a test statistic, particularly for the mean
- how to combine these concepts to calculate a confidence interval

Prerequisites

To be able to complete this notebook, you are expected to have a basic understanding of:

- what a random variable is (p.400 of Professor Spiegelhalter's *The Art of Statistics, hereinafter AoS*)
- what a population, and a population distribution, are (p. 397 of *AoS*)
- a high-level sense of what the normal distribution is (p. 394 of *AoS*)
- what the t-statistic is (p. 275 of *AoS*)

Happily, these should all be concepts with which you are reasonably familiar after having read ten chapters of Professor Spiegelhalter's book, *The Art of Statistics*.

We'll try to relate the concepts in this case study back to page numbers in *The Art of Statistics* so that you can focus on the Python aspects of this case study. The second part (part B) of this case study will involve another, more real-world application of these tools.

For this notebook, we will use data sampled from a known normal distribution. This allows us to compare our results with theoretical expectations.

2. An introduction to sampling from the normal distribution

First, let's explore the ways we can generate the normal distribution. While there's a fair amount of interest in sklearn within the machine learning community, you're likely to have heard of scipy if you're coming from the sciences. For this assignment, you'll use scipy.stats to complete your work.

This assignment will require some digging around and getting your hands dirty (your learning is maximized that way)! You should have the research skills and the tenacity to do these tasks independently, but if you struggle, reach out to your immediate community and your mentor for help.

In [1]:
```
from scipy.stats import norm
from scipy.stats import t
import numpy as np
import pandas as pd
from numpy.random import seed
import matplotlib.pyplot as plt
```

Q1: Call up the documentation for the `norm` function imported above. (Hint: that documentation is here). What is the second listed method?

In [ ]:

A: pdf()

Q2: Use the method that generates random variates to draw five samples from the standard normal distribution.

A:

In [2]:
```
seed(47)
# draw five samples here
samples = norm.rvs(size=5)
```

Q3: What is the mean of this sample? Is it exactly equal to the value you expected? Hint: the sample was drawn from the standard normal distribution. If you want a reminder of the properties of this distribution, check out p. 85 of *AoS*.

A:

In [3]:
```
# Calculate and print the mean here, hint: use np.mean()
mean = np.mean(samples)
print(mean)
```

```
0.19355593334131074
```

Q4: What is the standard deviation of these numbers? Calculate this manually here as $\sqrt{\frac{\sum_i(x_i - \bar{x})^2}{n}}$ (This is just the definition of **standard deviation** given by Professor Spiegelhalter on p.403 of *AoS*). Hint: np.sqrt() and np.sum() will be useful here and remember that numPy supports broadcasting.

A:

In [4]:
```
np.sqrt(np.sum(np.square(samples - mean))/len(samples))
```

Out[4]:
```
0.9606195639478641
```

Here we have calculated the actual standard deviation of a small data set (of size 5). But in this case, this small data set is actually a sample from our larger (infinite) population. In this case, the population is infinite because we could keep drawing our normal random variates until our computers die!

In general, the sample mean we calculate will not be equal to the population mean (as we saw above). A consequence of this is that the sum of squares of the deviations from the *population* mean will be bigger than the sum of squares of the deviations from the *sample* mean. In other words, the sum of squares of the deviations from the *sample* mean is too small to give an unbiased estimate of the *population* variance. An example of this effect is given here. Scaling our estimate of the variance by the factor $n/(n-1)$ gives an unbiased estimator of the population variance. This factor is known as Bessel's correction. The consequence of this is that the n in the denominator is replaced by $n-1$.

You can see Bessel's correction reflected in Professor Spiegelhalter's definition of **variance** on p. 405 of *AoS*.

Q5: If all we had to go on was our five samples, what would be our best estimate of the population standard deviation? Use Bessel's correction ($n-1$ in the denominator), thus $\sqrt{\frac{\sum_i(x_i - \bar{x})^2}{n-1}}$.

A:

In [5]:
```
np.sqrt(np.sum(np.square(samples - mean))/(len(samples)-1))
```

Out[5]:
```
1.0740053227518152
```

Q6: Now use numpy's std function to calculate the standard deviation of our random samples. Which of the above standard deviations did it return?

A:

In [6]:
```
np.std(samples)
```

Out[6]:
```
0.9606195639478641
```

Q7: Consult the documentation for np.std() to see how to apply the correction for estimating the population parameter and verify this produces the expected result.

A:

In [7]:
```
np.std(samples,ddof=1)
```

Out[7]:
```
1.0740053227518152
```

In [ ]:

Summary of section

In this section, you've been introduced to the scipy.stats package and used it to draw a small sample from the standard normal distribution. You've calculated the average (the mean) of this sample and seen that this is not exactly equal to the expected population parameter (which we know because we're generating the random variates from a specific, known distribution). You've been introduced to two ways of calculating the standard deviation; one uses n in the denominator and the other uses $n-1$ (Bessel's correction). You've also seen which of these calculations np.std() performs by default and how to get it to generate the other.

You use n as the denominator if you want to calculate the standard deviation of a sequence of numbers. You use $n-1$ if you are using this sequence of numbers to estimate the population parameter. This brings us to some terminology that can be a little confusing.

The population parameter is traditionally written as σ and the sample statistic as s. Rather unhelpfully, s is also called the sample standard deviation (using $n-1$) whereas the standard deviation of the sample uses n. That's right, we have the sample standard deviation and the standard deviation of the sample and they're not the same thing!

The sample standard deviation

$$s = \sqrt{\frac{\sum_i(x_i - \bar{x})^2}{n-1}} \approx \sigma,$$

is our best (unbiased) estimate of the population parameter (σ).

If your dataset *is* your entire population, you simply want to calculate the population parameter, σ, via

$$\sigma = \sqrt{\frac{\sum_i(x_i - \bar{x})^2}{n}}$$

as you have complete, full knowledge of your population. In other words, your sample *is* your population. It's worth noting that we're dealing with what Professor Spiegehalter describes on p. 92 of *AoS* as a **metaphorical population**: we have all the data, and we act as if the data-point is taken from a population at random. We can think of this population as an imaginary space of possibilities.

If, however, you have sampled *from* your population, you only have partial knowledge of the state of your population. In this case, the standard deviation of your sample is not an unbiased estimate of the standard deviation of the population, in which case you seek to estimate that population parameter via the sample standard deviation, which uses the $n-1$ denominator.

Great work so far! Now let's dive deeper.

3. Sampling distributions

So far we've been dealing with the concept of taking a sample from a population to infer the population parameters. One statistic we calculated for a sample was the mean. As our samples will be expected to vary from one draw to another, so will our sample statistics. If we were to perform repeat draws of size n and calculate the mean of each, we would expect to obtain a distribution of values. This is the sampling distribution of the mean. **The Central Limit Theorem (CLT)** tells us that such a distribution will approach a normal distribution as n increases (the intuitions behind the CLT are covered in full on p. 236 of *AoS*). For the sampling distribution of the mean, the standard deviation of this distribution is given by

$$\sigma_{mean} = \frac{\sigma}{\sqrt{n}}$$

where σ_{mean} is the standard deviation of the sampling distribution of the mean and σ is the standard deviation of the population (the population parameter).

This is important because typically we are dealing with samples from populations and all we know about the population is what we see in the sample. From this sample, we want to make inferences about the population. We may do this, for example, by looking at the histogram of the values and by calculating the mean and standard deviation (as estimates of the population parameters), and so we are intrinsically interested in how these quantities vary across samples.

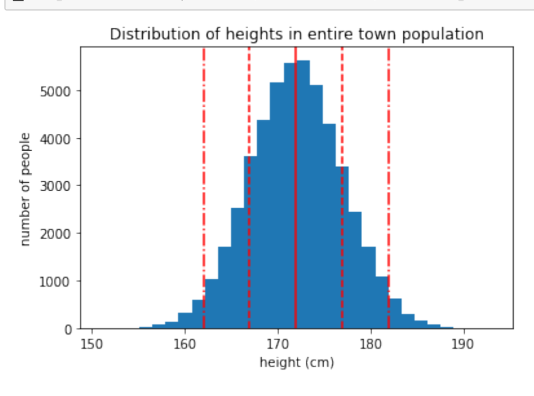
In other words, now that we've taken one sample of size n and made some claims about the general population, what if we were to take another sample of size n? Would we get the same result? Would we make the same claims about the general population? This brings us to a fundamental question: *when we make some inference about a population based on our sample, how confident can we be that we've got it 'right'?*

We need to think about **estimates and confidence intervals**: those concepts covered in Chapter 7, p. 189, of *AoS*.

Now, the standard normal distribution (with its variance equal to its standard deviation of one) would not be a great illustration of a key point. Instead, let's imagine we live in a town of 50,000 people and we know the height of everyone in this town. We will have 50,000 numbers that tell us everything about our population. We'll simulate these numbers now and put ourselves in one particular town, called 'town 47', where the population mean height is 172 cm and population standard deviation is 5 cm.

In [8]:
```
seed(47)
pop_heights = norm.rvs(172, 5, size=50000)
```

In [9]:
```
_ = plt.hist(pop_heights, bins=30)
_ = plt.xlabel('height (cm)')
_ = plt.ylabel('number of people')
_ = plt.title('Distribution of heights in entire town population')
_ = plt.axvline(172, color='r')
_ = plt.axvline(172+5, color='r', linestyle='--')
_ = plt.axvline(172-5, color='r', linestyle='--')
_ = plt.axvline(172+10, color='r', linestyle='-.')
_ = plt.axvline(172-10, color='r', linestyle='-.')
```



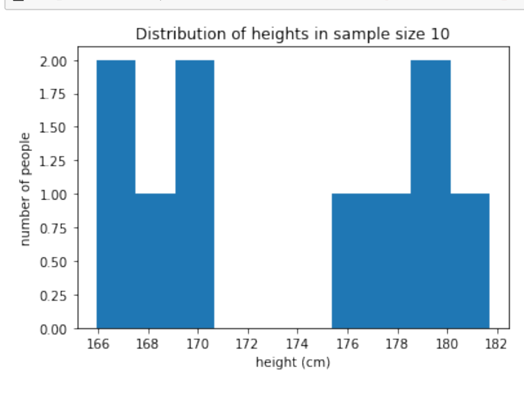
Now, 50,000 people is rather a lot to chase after with a tape measure. If all you want to know is the average height of the townsfolk, then can you just go out and measure a sample to get a pretty good estimate of the average height?

In [10]:
```
def townsfolk_sampler(n):
    return np.random.choice(pop_heights, n)
```

Let's say you go out one day and randomly sample 10 people to measure.

In [11]:
```
seed(47)
daily_sample1 = townsfolk_sampler(10)
```

In [12]:
```
_ = plt.hist(daily_sample1, bins=10)
_ = plt.xlabel('height (cm)')
_ = plt.ylabel('number of people')
_ = plt.title('Distribution of heights in sample size 10')
```



The sample distribution doesn't resemble what we take the population distribution to be. What do we get for the mean?

In [13]:
```
np.mean(daily_sample1)
```

Out[13]:
```
173.47911444163503
```

And if we went out and repeated this experiment?

In [14]:
```
daily_sample2 = townsfolk_sampler(10)
```

In [15]:
```
np.mean(daily_sample2)
```

Out[15]:
```
173.7317666636263
```

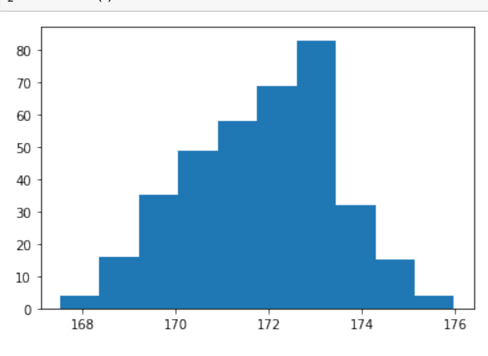
Q8: Simulate performing this random trial every day for a year, calculating the mean of each daily sample of 10, and plot the resultant sampling distribution of the mean.

A:

In [ ]:

In [16]:
```
seed(47)
# take your samples here
daily_mean = [np.mean(townsfolk_sampler(10)) for i in range(365)]
```

In [17]:
```
plt.hist(daily_mean, bins=10)
plt.show()
```



The above is the distribution of the means of samples of size 10 taken from our population. The Central Limit Theorem tells us the expected mean of this distribution will be equal to the population mean, and standard deviation will be $\sigma/\sqrt{n}$, which, in this case, should be approximately 1.58.

Q9: Verify the above results from the CLT.

A:

In [18]:
```
yearly_mean = np.mean(daily_mean)
print(yearly_mean)
```

```
171.8660049358649
```

In [19]:
```
mean_std = np.std(daily_mean)
print(mean_std)
```

```
1.5756704135286475
```

Remember, in this instance, we knew our population parameters, that the average height really is 172 cm and the standard deviation is 5 cm, and we see some of our daily estimates of the population mean were as low as around 168 and some as high as 176.

Q10: Repeat the above year's worth of samples but for a sample size of 50 (perhaps you had a bigger budget for conducting surveys that year)! Would you expect your distribution of sample means to be wider (more variable) or narrower (more consistent)? Compare your resultant summary statistics to those predicted by the CLT.

A: The distribution of sample means will be narrower by taking 50 samples as opposed to 10.

In [20]:
```
seed(47)
# calculate daily means from the larger sample size here
daily_mean = [np.mean(townsfolk_sampler(50)) for i in range(365)]
```

In [21]:
```
mean_std = np.std(daily_mean)
clt_mean_std = 5 / np.sqrt(50)
print(mean_std, clt_mean_std)
```

```
0.6736107539771146 0.7071067811865475
```

What we've seen so far, then, is that we can estimate population parameters from a sample from the population, and that samples have their own distributions. Furthermore, the larger the sample size, the narrower are those sampling distributions.

Normally testing time!

All of the above is well and good. We've been sampling from a population we know is normally distributed, we've come to understand when to use n and when to use $n-1$ in the denominator to calculate the spread of a distribution, and we've seen the Central Limit Theorem in action for a sampling distribution. All seems very well behaved in Frequentist land. But, well, why should we really care?

Remember, we rarely (if ever) actually know our population parameters but we still have to estimate them somehow. If we want to make inferences to conclusions like "this observation is unusual" or "my population mean has changed" then we need to have some idea of what the underlying distribution is so we can calculate relevant probabilities. In frequentist inference, we use the formulae above to deduce these population parameters. Take a moment in the next part of this assignment to refresh your understanding of how these probabilities work.

Recall some basic properties of the standard normal distribution, such as that about 68% of observations are within plus or minus 1 standard deviation of the mean. Check out the precise definition of a normal distribution on p. 394 of *AoS*.

Q11: Using this fact, calculate the probability of observing the value 1 or less in a single observation from the standard normal distribution. Hint: you may find it helpful to sketch the standard normal distribution (the familiar bell shape) and mark the number of standard deviations from the mean on the x-axis and shade the regions of the curve that contain certain percentages of the population.

A: Observing a value of 1 or less can be broken down into two parts: the probability of observing a value less than the mean of zero (50%) plus one half the probability of observing a value within plus/minus 1 standard deviation of the mean (68% / 2 = 34%). The probability of observing a value of 1 or less is 84%.

Calculating this probability involved calculating the area under the curve from the value of 1 and below. To put it in mathematical terms, we need to *integrate* the probability density function. We could just add together the known areas of chunks (from -Inf to 0 and then 0 to $+\sigma$ in the example above). One way to do this is to look up tables (literally). Fortunately, scipy has this functionality built in with the cdf() function.

Q12: Use the cdf() function to answer the question above again and verify you get the same answer.

A:

In [22]:
```
print(norm().cdf(1))
```

```
0.8413447460685429
```

Q13: Using our knowledge of the population parameters for our townsfolks' heights, what is the probability of selecting one person at random and their height being 177 cm or less? Calculate this using both of the approaches given above.

A:

In [23]:
```
seed(47)
print(norm(172, 5).cdf(177))
```

```
0.8413447460685429
```

Q14: Turning this question around — suppose we randomly pick one person and measure their height and find they are 2.00 m tall. How surprised should we be at this result, given what we know about the population distribution? In other words, how likely would it be to obtain a value at least as extreme as this? Express this as a probability.

A:

In [24]:
```
print(1 - norm(172, 5).cdf(200))
```

```
1.0717590259723409e-08
```

What we've just done is calculate the ***p-value*** of the observation of someone 2.00m tall (review *p*-values if you need to on p. 399 of *AoS*). We could calculate this probability by virtue of knowing the population parameters. We were then able to use the known properties of the relevant normal distribution to calculate the probability of observing a value at least as extreme as our test value.

We're about to come to a pinch, though. We've said a couple of times that we rarely, if ever, know the true population parameters; we have to estimate them from our sample and we cannot even begin to estimate the standard deviation from a single observation.

This is very true and usually we have sample sizes larger than one. This means we can calculate the mean of the sample as our best estimate of the population mean and the standard deviation as our best estimate of the population standard deviation.

In other words, we are now coming to deal with the sampling distributions we mentioned above as we are generally concerned with the properties of the sample means we obtain.

Above, we highlighted one result from the CLT, whereby the sampling distribution (of the mean) becomes narrower and narrower with the square root of the sample size. We remind ourselves that another result from the CLT is that *even if the underlying population distribution is not normal, the sampling distribution will tend to become normal with sufficiently large sample size*. (**Check out p. 199 of AoS if you need to revise this**). This is the key driver for us 'requiring' a certain sample size, for example you may frequently see a minimum sample size of 30 stated in many places. In reality this is simply a rule of thumb; if the underlying distribution is approximately normal then your sampling distribution will already be pretty normal, but if the underlying distribution is heavily skewed then you'd want to increase your sample size.

Q15: Let's now start from the position of knowing nothing about the heights of people in our town.

- Use the random seed of 47, to randomly sample the heights of 50 townsfolk
- Estimate the population mean using np.mean
- Estimate the population standard deviation using np.std (remember which denominator to use!)
- Calculate the (95%) margin of error (use the exact critial z value to 2 decimal places - look this up or use norm.ppf()) Recall that the ***margin of error*** is mentioned on p. 189 of the *AoS* and discussed in depth in that chapter).
- Calculate the 95% Confidence Interval of the mean (***confidence intervals*** are defined on p. 385 of *AoS*)
- Does this interval include the true population mean?

A:

In [25]:
```
seed(47)
# take your sample now
n_samples = 50
samples = norm.rvs(172, 5, size=n_samples)
```

In [26]:
```
population_mean_estimate = np.mean(samples)
print(population_mean_estimate)
```

```
171.09434218281885
```

In [27]:
```
population_std_estimate = np.std(samples,ddof=1)
print(population_std_estimate)
```

```
4.868476091077329
```

In [28]:
```
critical_value = norm.ppf(0.975)
standard_error = population_std_estimate / np.sqrt(n_samples)
margin_of_error = critical_value * standard_error
print(round(margin_of_error,2))
```

```
1.35
```

In [29]:
```
confidence_interval = population_mean_estimate + np.array([-margin_of_error, margin_of_error])
print(np.round(confidence_interval,2))
```

```
[169.74 172.44]
```

Q16: Above, we calculated the confidence interval using the critical z value. What is the problem with this? What requirement, or requirements, are we (strictly) failing?

A: The critical value assumes a normal distribution. This may not be the case.

Q17: Calculate the 95% confidence interval for the mean using the *t* distribution. Is this wider or narrower than that based on the normal distribution above? If you're unsure, you may find this resource useful. For calculating the critical value, remember how you could calculate this for the normal distribution using norm.ppf().

A:

In [30]:
```
critical_value = t.ppf(0.975, df=50)
standard_error = population_std_estimate / np.sqrt(n_samples)
margin_of_error = critical_value * standard_error
print(round(margin_of_error,2))
```

```
1.38
```

In [31]:
```
confidence_interval = population_mean_estimate + np.array([-margin_of_error, margin_of_error])
print(np.round(confidence_interval,2))
```

```
[169.71 172.48]
```

In [ ]:

This is slightly wider than the previous confidence interval. This reflects the greater uncertainty given that we are estimating population parameters from a sample.

4. Learning outcomes

Having completed this project notebook, you now have hands-on experience:

- sampling and calculating probabilities from a normal distribution
- identifying the correct way to estimate the standard deviation of a population (the population parameter) from a sample
- with sampling distribution and now know how the Central Limit Theorem applies
- with how to calculate critical values and confidence intervals

In [ ]: