Inferential Statistics Ib - Frequentism

Learning objectives

Welcome to the second Frequentist inference mini-project! Over the course of working on this mini-project and the previous frequentist mini-project, you'll learn the fundamental concepts associated with frequentist inference. The following list includes the topics you will become familiar with as you work through these two mini-projects:

- the *z*-statistic
- the *t*-statistic
- the difference and relationship between the two
- the Central Limit Theorem, 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 confidence intervals and p-values
- how those confidence intervals and p-values allow you to perform hypothesis (or A/B) tests

Prerequisites

- what a random variable is
- what a probability density function (pdf) is
- what the cumulative density function is
- a high-level sense of what the Normal distribution

If these concepts are new to you, please take a few moments to Google these topics in order to get a sense of what they are and how you might use them.

These two notebooks were designed to bridge the gap between having a basic understanding of probability and random variables and being able to apply these concepts in Python. This second frequentist inference mini-project focuses on a real-world application of this type of inference to give you further practice using these concepts.

In the previous notebook, we used only data from a known normal distribution. You'll now tackle real data, rather than simulated data, and answer some relevant real-world business problems using the data.

Hospital medical charges

Imagine that a hospital has hired you as their data analyst. An administrator is working on the hospital's business operations plan and needs you to help them answer some business questions. This mini-project, as well as the bootstrap and Bayesian inference mini-projects also found in this unit are designed to illustrate how each of the inferential statistics methods have their uses for different use cases. In this assignment notebook, you're going to use frequentist statistical inference on a data sample to answer the questions:

- has the hospital's revenue stream fallen below a key threshold?
- are patients with insurance really charged different amounts than those without? Answering that last question with a frequentist approach makes some assumptions, or requires some knowledge, about the two groups. In the next mini-project, you'll use bootstrapping to test that assumption. And in the final mini-project of the unit, you're going to create a model for simulating *individual* charges (not a sampling distribution) that the hospital can use to model a range of scenarios.

We are going to use some data on medical charges obtained from Kaggle. For the purposes of this exercise, assume the observations are the result of random sampling from our one hospital. Recall in the previous assignment, we introduced the Central Limit Theorem (CLT), and how it tells us that the distributions of sample statistics approach a normal distribution as n increases. The amazing thing about this is that it applies to the sampling distributions of statistics that have been calculated from even highly non-normal distributions of data. Remember, also, that hypothesis testing is very much based on making inferences about such sample statistics. You're going to rely heavily on the CLT to apply frequentist (parametric) tests to answer the questions in this notebook.

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import t
from numpy.random import seed
medical = pd.read_csv('data/insurance2.csv')
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-2-14bdbabb94b5> in <module>
      4 from scipy.stats import t
      5 from numpy.random import seed
----> 6 medical = pd.read_csv('data/insurance2.csv')

~\anaconda\lib\site-packages\pandas\io\parsers.py in parser_f(filepath_or_buffer, sep, delimite
r, header, names, index_col, usecols, squeeze, prefix, mangle_dupe_cols, dtype, engine, converte
rs, true_values, false_values, skipinitialspace, skiprows, skipfooter, nrows, na_values, keep_de
fault_na, na_filter, verbose, skip_blank_lines, parse_dates, infer_datetime_format, keep_date_co
l, date_parser, dayfirst, iterator, chunksize, compression, thousands, decimal, lineterminator,
 quotechar, quoting, doublequote, escapechar, comment, encoding, dialect, tupleize_cols, error_b
ad_lines, warn_bad_lines, delim_whitespace, low_memory, memory_map, float_precision)
    700                     skip_blank_lines=skip_blank_lines)
    701
--> 702         return _read(filepath_or_buffer, kwds)
    703
    704     parser_f.__name__ = name

~\anaconda\lib\site-packages\pandas\io\parsers.py in _read(filepath_or_buffer, kwds)
    427
    428     # Create the parser.
--> 429     parser = TextFileReader(filepath_or_buffer, **kwds)
    430
    431     if chunksize or iterator:

~\anaconda\lib\site-packages\pandas\io\parsers.py in __init__(self, f, engine, **kwds)
    893             self.options['has_index_names'] = kwds['has_index_names']
    894
--> 895         self._make_engine(self.engine)
    896
    897     def close(self):

~\anaconda\lib\site-packages\pandas\io\parsers.py in _make_engine(self, engine)
   1120     def _make_engine(self, engine='c'):
   1121         if engine == 'c':
-> 1122             self._engine = CParserWrapper(self.f, **self.options)
   1123         else:
   1124             if engine == 'python':

~\anaconda\lib\site-packages\pandas\io\parsers.py in __init__(self, src, **kwds)
   1851         kwds['usecols'] = self.usecols
   1852
-> 1853         self._reader = parsers.TextReader(src, **kwds)
   1854         self.unnamed_cols = self._reader.unnamed_cols
   1855

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader.__cinit__()

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._setup_parser_source()

FileNotFoundError: [Errno 2] File b'data/insurance2.csv' does not exist: b'data/insurance2.csv'
```

In [ ]:
```
medical.shape
```

In [ ]:
```
medical.head()
```

Q: Plot the histogram of charges and calculate the mean and standard deviation. Comment on the appropriateness of these statistics for the data.

A:

In [ ]:
```
plt.hist(medical['charges'])
plt.xlabel('Medical Charges')
plt.ylabel('Frequency')
```

In [ ]:
```
np.mean(medical['charges']), np.std(medical['charges'])
```

Q: The administrator is concerned that the actual average charge has fallen below 12000, threatening the hospital's operational model. On the assumption that these data represent a random sample of charges, how would you justify that these data allow you to answer that question? And what would be the most appropriate frequentist test, of the ones discussed so far, to apply?

A: T test is the most appropriate frequentist test that can be used for Hypothesis testing.

Q: Given the nature of the administrator's concern, what is the appropriate confidence interval in this case? A one-sided or two-sided interval? Calculate the critical value and the relevant 95% confidence interval for the mean and comment on whether the administrator should be concerned?

A:

In [ ]:
```
from scipy.stats import ttest_1samp

ttest_1samp(medical.charges, 12000)
```

In [ ]:
```
from scipy import stats

stats.norm.interval(0.95, loc=medical.charges.mean(), scale=medical.charges.std())
```

The administrator then wants to know whether people with insurance really are charged a different amount to those without.

Q: State the null and alternative hypothesis here. Use the *t*-test for the difference between means where the pooled standard deviation of the two groups is given by

$$s_p = \sqrt{\frac{(n_0 - 1)s_0^2 + (n_1 - 1)s_1^2}{n_0 + n_1 - 2}}$$

and the *t* test statistic is then given by

$$t = \frac{\bar{x}_0 - \bar{x}_1}{s_p \sqrt{1/n_0 + 1/n_1}}$$

What assumption about the variances of the two groups are we making here?

A:

Q: Perform this hypothesis test both manually, using the above formulae, and then using the appropriate function from scipy.stats (hint, you're looking for a function to perform a *t*-test on two independent samples). For the manual approach, calculate the value of the test statistic and then its probability (the p-value). Verify you get the same results from both.

A:

In [ ]:
```
n0 = medical[medical.insuranceclaim==0]['charges'].values
n1 = medical[medical.insuranceclaim==1]['charges'].values

print(n0.std(), n1.std())
print(n0.shape, n1.shape)
print(n0.mean(), n1.mean())
```

In [ ]:
```
sp = np.sqrt((len(n0)-1)*(n0.std())**2 + (len(n1)-1)*(n1.std())**2)/(len(n0)+len(n1)-2))
sp
```

In [ ]:
```
t = (n0.mean()-n1.mean()) / sp*np.sqrt((1/len(n0))+(1/len(n1)))
t
```

In [ ]:
```
from scipy.stats import ttest_ind
ttest_ind(n0, n1)
```

Congratulations! Hopefully you got the exact same numerical results. This shows that you correctly calculated the numbers by hand. Secondly, you used the correct function and saw that it's much easier to use. All you need to do pass your data to it.

Q: In the above calculations, we assumed the sample variances were equal. We may well suspect they are not (we'll explore this in another assignment). The calculation becomes a little more complicated to do by hand in this case, but we now *know* of a helpful function. Check the documentation for the function to tell it not to assume equal variances and perform the test again.

A:

In [ ]:
```
```

Q: Conceptual question: look through the documentation for statistical test functions in scipy.stats. You'll see the above *t*-test for a sample, but can you see an equivalent one for performing a *z*-test from a sample? Comment on your answer.

A:

Learning outcomes

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