Type I and Type II Errors - Lab

Introduction

In this lab, you'll run some of your own simulations to learn more about type I and type II errors. Remember that, the result of a statistical hypothesis test and the corresponding decision of whether to reject or accept the null hypothesis, is not infallible. A test provides evidence for or against the null hypothesis and then you decide whether to accept or reject it based on that evidence, but the evidence may lack the strength to arrive at the correct conclusion. Incorrect conclusions made from hypothesis tests fall in one of two categories, i.e. Type I and Type II errors. By running some of these simulations, you should have a better idea of why a 95% confidence level is often used for hypothesis testing.

Objectives

You will be able to:

- Differentiate how Type I and Type II errors relate to the p and z-value
- Describe the relationship between alpha and Type I errors
- Create simulations and visualizations to represent scenarios involving Type I and Type II errors

Alpha and Beta

Alpha (α): is the probability of a Type I error i.e. finding a difference when a difference does not exist.

Most medical literature uses an alpha cut-off of 5% (0.05), indicating a 5% chance that a significant difference is actually due to chance and is not a true difference.

Beta (β): is the probability of a Type II error i.e. not detecting a difference when one actually exists.

Beta is directly related to study power (Power = $1 - \beta$) which you will investigate further in the next lesson. Most medical literature uses a beta cut-off of 20% (0.2), indicating a 20% chance that a significant difference is missed.

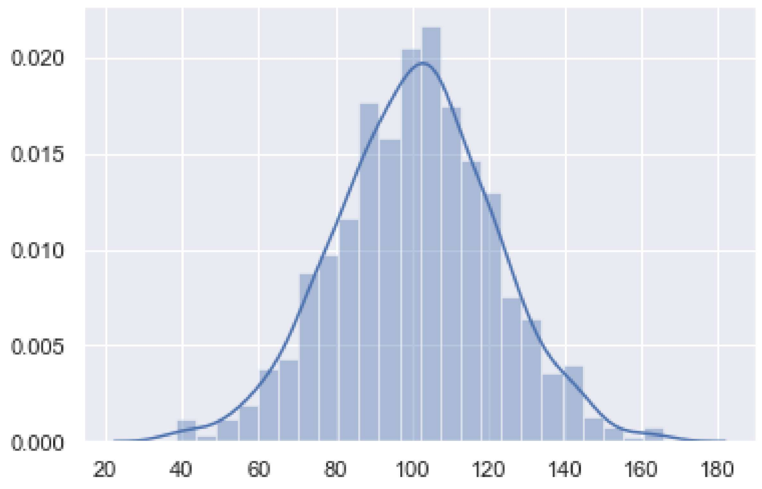
Now you will attempt to create a simulation to visualize this phenomenon using Python.

```
import numpy as np
import pandas as pd
import scipy.stats as stats
import matplotlib.pyplot as plt
import math
import random

import seaborn as sns
sns.set(color_codes=True)
```

First, create a population of 1000 elements with a mean of 100 and a standard deviation of 20.

```
# Create a population with mean=100 and sd=20 and size = 1000
pop = np.random.normal(100, 20, 1000)
pop.dtype
sns.distplot(pop)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a201e9550>
```



Now take two samples from this population and comment on the difference between their means and standard deviations. How would you ensure the independence between the elements of these samples?

```
k = 100
sample1 = np.random.choice(pop,100,replace=True)

print ("Sample 1 Summary")
stats.describe(sample1)
```

```
Sample 1 Summary
```

```
DescribeResult(nobs=100, minmax=(66.40851232603063, 161.62571006846179),
mean=104.35830962968306, variance=362.19033587986775,
skewness=0.3616388937115025, kurtosis=0.01471382255051834)
```

```
sample2 = np.random.choice(pop,100,replace=True)
print ("Sample 2 Summary")
stats.describe(sample2)
```

```
Sample 2 Summary
```

```
DescribeResult(nobs=100, minmax=(53.4503814938987, 142.2385929977146),
mean=98.15819084577764, variance=423.12632108466056,
skewness=-0.02741488284820187, kurtosis=-0.7119732006166188)
```

You can see that if you took two samples from this population, the difference between the mean of samples 1 and 2 is very small (this can be tried repeatedly). You must sample with replacement in order to ensure the independence assumption between elements of the sample.

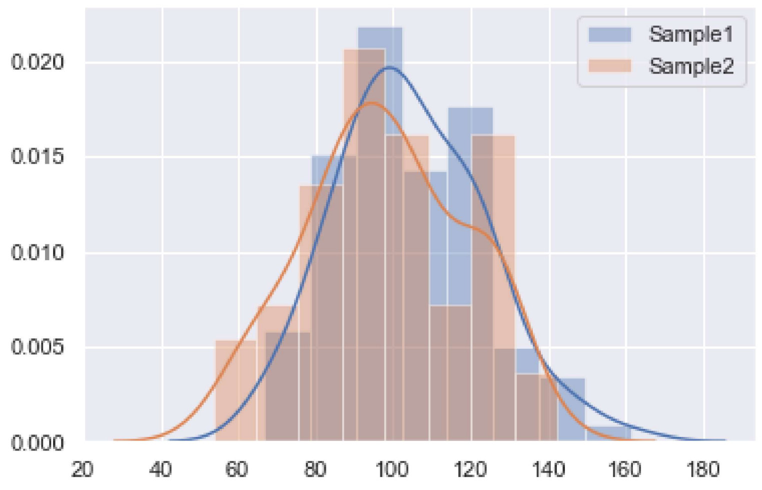
There is, however, still a probability of seeing a very large difference between values, even though they're estimates of the same population parameters. In a statistical setting, you'd interpret these unusually large differences as evidence that the two samples are statistically different. It depends on how you define statistical significance. In statistical tests, this is done by setting a significance threshold α (alpha). Alpha controls how often we'll get a type I error. A type I error occurs when the statistical test erroneously indicates a significant result.

You can run a two-sample t-test with the independence assumption on these samples and, as expected, the null hypothesis will fail to be rejected due to similarities between distributions. You can also visualize the distribution to confirm the similarity between means and SDs.

```
# test the sample means
stats.ttest_ind(sample1, sample2)
```

```
Ttest_indResult(statistic=2.2124710940967014, pvalue=0.0280770750981863)
```

```
plt.figure("Test Samples")
sns.distplot(sample1, label='Sample1')
sns.distplot(sample2, label='Sample2')
plt.legend()
plt.show()
```



Simulating Type I and II errors

Type I error

Remember that when a hypothesis test is being performed, scientists are trying to determine if two samples are from the same population or not. When a hypothesis is rejected, they are concluding that a sample must have come from a different population. Type I error describes a situation where you reject the null hypothesis when it is actually true. It assumes two samples come from a *different* population when, in reality, they are from the *same* population. This type of error is also known as a "false positive" or "false hit". The type I error rate is equal to the significance level α, so setting a higher confidence level (and therefore lower α) reduces the chances of getting a false positive.

How alpha affects the prevalence of Type I errors.

Next, we shall see how alpha affects the rate of type I errors.

Exercise: Write a function `type_1_error` in Python to encapsulate the code shown above in order to repeat hypothesis tests on two randomly drawn distributions. The t-test will mostly fail to reject the null hypothesis, except, when by random chance you get a set of **extremely** different samples thus reject the null hypothesis (type I error). The frequency of such bad results depends upon the value of alpha.

`type_1_error` should take in the parameters:

- `population` : (NumPy array) a random normal distribution
- `num_tests` : (int) specifies the number of hypothesis tests to compute
- `alphas` : (list) a list of the alpha levels at which you are testing

`type_1_error` should return:

- `sig_tests` : (DataFrame) a dataframe that has the columns 'type_1_error', 'p_value', 'alpha'

Within `type_1_error` , you should:

1. Repeatedly take two random samples from `population` and run independent t-tests.
2. Store the p-value, alpha, and a boolean variable to show whether the null hypothesis **was rejected** or not (i.e. if p-value is less than alpha), for each test

To test your function:

1. Create a population distribution with a mean of 100, a standard deviation of 20, and a size of 1000
2. Specify the number of hypothesis tests to be 1000
3. Create a list of alphas = [0.001, 0.01, 0.05, 0.1, 0.2, 0.5]

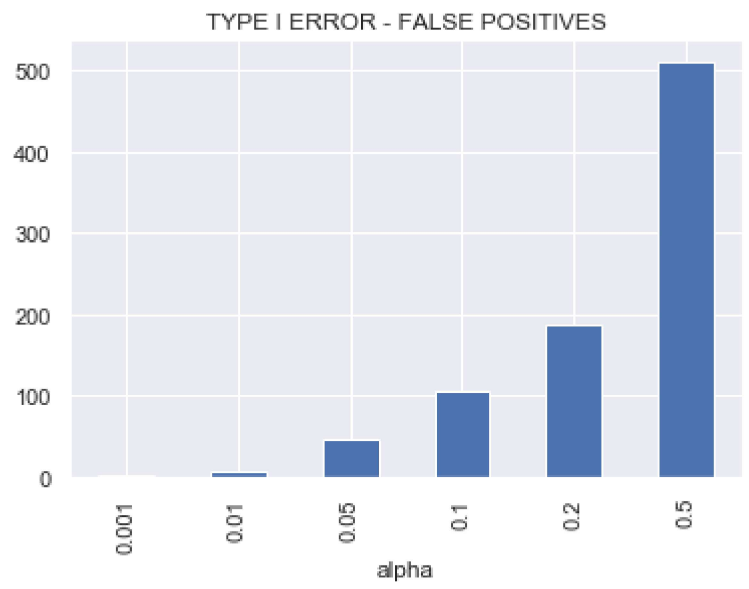
```python
def type_1_error(population, num_tests, alpha_set):
    """
    Parameters
    ----------
    population: ndarray
        A random normal distribution
    num_tests: int
        The number of hypothesis tests to be computed
    alpha_set: list
        List of alpha levels

    Returns
    ----------
    sig_tests : DataFrame
        A dataframe containing the columns 'type_1_error', 'p_value', and 'alpha'
    """
    columns = ['type_1_error','p_value','alpha']
    sig_tests = pd.DataFrame(columns=columns)
    counter = 0

    for i in range(1,num_tests+1):

        for alpha in alpha_set:

            # take two samples from the same population
            samp1 = np.random.choice(population,100,replace=True)
            samp2 = np.random.choice(population,100,replace=True)
            # test sample means
            result = stats.ttest_ind(samp1, samp2)

            # evaluate whether null hypothesis is rejected or not
            if result[1] < alpha:
                 sig_tests.loc[counter] = [1, result[1], alpha]
            else:
                 sig_tests.loc[counter] = [0, result[1], alpha]

            counter += 1

    return sig_tests
```

Now we have to summarize the results, this is done using the pandas `groupby()` method which sums the `type_1_error` column for each level of alpha. The `groupby()` method iterates over each value of alpha, selecting the type I error column for all rows with a specific level of alpha, and then applies the sum function to the selection.

What's the relationship between alpha and type I errors?

```
# group type I error by values of alpha
pop = np.random.normal(100, 20, 1000)
num_tests = 1000
alpha_set = [0.001, 0.01, 0.05, 0.1, 0.2, 0.5]
sig_tests_1 = type_1_error(pop, num_tests, alpha_set)
group_error = sig_tests_1.groupby('alpha')['type_1_error'].sum()
group_error.plot.bar(title = "TYPE I ERROR - FALSE POSITIVES")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2112c240>
```



The grouped data clearly shows that as value of alpha is increases from .001 to .5, the probability of type I errors also increases.

Type II error

This error describes a situation where you fail to reject the null hypothesis when it is actually false. Type II error is also known as a "false negative" or "miss". The higher your confidence level, the more likely you are to make a type II error.

How alpha affects the prevalence of Type II errors.

Exercise: Write a function called `type_2_error` similar to the above except samples should be taken from two different populations. The hypothesis test should, in most cases, reject the null hypothesis as the samples belong to different populations, except, in extreme cases where there is no significant difference between samples i.e. a type II error (False Negatives). Your function should demonstrate how the rate of false negatives is affected by alpha.

`type_2_error` should take in the parameters:

- `population` : (NumPy array) a random normal distribution
- `population_2` : (NumPy array) a random normal distribution with a different mean than the population
- `num_tests` : (int) specifies the number of hypothesis tests to compute
- `alphas` : (list) a list of the alpha levels at which you are testing

`type_2_error` should return:

- `sig_tests` : (DataFrame) a dataframe that has the columns 'type_2_error', 'p_value', 'alpha'

Within `type_2_error` , you should:

1. Repeatedly take two random samples from population and run independent t-tests.
2. Store p_value, alpha, and a boolean variable to show whether the null hypothesis **failed to be rejected** or not (i.e. if p-value is less than alpha), for each test

To test your function:

1. Create a population distribution with a mean of 100, a standard deviation of 20, and a size of 1000
2. Create a second population distribution with a mean of 110, a standard deviation of 20, and a size of 1000

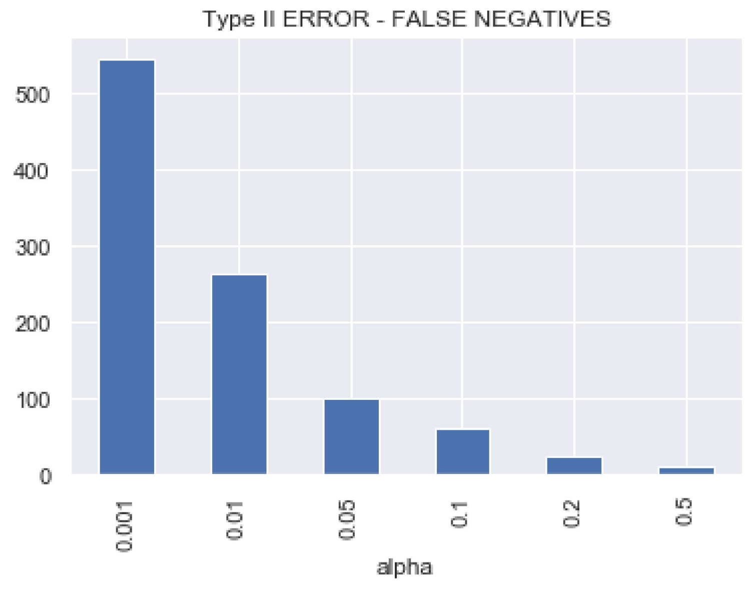
3. Specify the number of hypothesis tests to be 1000
4. Create a list of alphas = [0.001, 0.01, 0.05, 0.1, 0.2, 0.5]

```
def type_2_error(population, population_2, num_tests, alpha_set):

    """
    Parameters
    ----------
    population: ndarray
        A random normal distribution
    population_2: ndarray
        A different random normal distribution
    num_tests: int
        The number of hypothesis tests to be computed
    alpha_set: list
        List of alpha levels

    Returns
    ----------
    sig_tests : DataFrame
        A dataframe containing the columns 'type_2_error', 'p_value', and 'alpha'
    """

    columns = ['type_2_error','p_val','alpha']
    sig_tests = pd.DataFrame(columns=columns)
    counter = 0

    for i in range(1,num_tests+1):

        for alpha in alpha_set:

            # take two samples from the same population
            samp1 = np.random.choice(population,100,replace=True)
            samp2 = np.random.choice(population_2,100,replace=True)

            # test sample means
            result = stats.ttest_ind(samp1, samp2)

            # evaluate whether null hypothesis is rejected or not
            if result[1] > alpha:
                 sig_tests.loc[counter] = [1, result[1], alpha]
            else:
                 sig_tests.loc[counter] = [0, result[1], alpha]

            counter += 1

    return sig_tests
```

Now, create a visualization that will represent each one of these decisions. What's the relationship between alpha and type II errors?

```
pop = np.random.normal(100, 20, 1000)
pop2 = np.random.normal(110, 20, 1000)
num_tests = 1000
alpha_set = [0.001, 0.01, 0.05, 0.1, 0.2, 0.5]
sig_tests_2 = type_2_error(pop,pop2,num_tests,alpha_set)

group_error2 = sig_tests_2.groupby('alpha')['type_2_error'].sum()
group_error2.plot.bar(title = "Type II ERROR - FALSE NEGATIVES")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x10cfd7128>
```



The grouped data clearly shows that as value of alpha is increased from .001 to .5, the probability of type II errors decreases.

Why is an α level of 0.05 chosen as a cut-off for statistical significance?

The α level of 0.05 is considered s good balance to avoid excessive type I or type II errors.

If you decide to use a large value for alpha :

- Increases the chance of rejecting the null hypothesis
- The risk of a type II error (false negative) is REDUCED
- Risk of a type I error (false positive) is INCREASED

Similarly, if you decide to use a very small value of alpha, it'll change the outcome as:

- Increases the chance of accepting the null hypothesis
- The risk of a Type I error (false positive) is REDUCED
- Risk of a Type II error (false negative) is INCREASED

From above, you can see that in statistical hypothesis testing, the more you try and avoid a type I error (false positive), the more likely a type II error (false negative) will occur.

Summary

The key statistical point here is that there is always a trade off between false positives and false negatives. By increasing alpha, the number of false positives increases, but the number of false negatives decreases as shown in the bar graphs. The value of α = 0.05 is considered a reasonable compromise between these two types of errors. Within the concept of "significance," there is embedded a trade-off between these two types of errors.

Think of "significance" as a compromise between false positives and negatives, not as absolute determination.