Multiple Hypothesis Testing

Create a data frame consisting of 1000 points.

The probability of making one or more accidental false discovery increases the more tests we perform. Multiple hypothesis testing occurs frequently in fields where we can generate a lot of features, and we need to test whether each of the features represents a possible novel discovery. For example, in genomics applied to disease studies we want to test whether any of the thousands of genes, has activity significantly different than in non-diseased individuals.

In this blog post, I will cover few of the popuar methods used to adjust p-values in these cases and explain when different approaches are sensible:

1. False Positive Rate (FPR)
2. Family-Wise Error Rate (FWER)
3. False Discovery Rate (FDR)

To run this code, we need Python with `pandas`, `numpy`, `scipy` and `statsmodels` libraries installed.

Create test data

We will create a simulated example to better understand how various manipulation of p-values can lead to different conclusions.

For the purpose of this example, we start by creating a dataframe of 1000 features. 990 of which (99%) will have their values generated from a Normal distribution with mean = 0, called a Null model. 1% of the features will be generated from a Normal distribution mean = 3, called a Non-Null model.

```
import pandas as pd
import numpy as np
from scipy.stats import norm
from statsmodels.stats.multitest import multipletests
```

```
np.random.seed(42)

df = pd.DataFrame({
    'hypothesis': np.concatenate((
        ['null'] * 990,
        ['non-null'] * 10,
    )),
    'x': np.concatenate((
        norm.rvs(0, 1, size=990),
        norm.rvs(2, 1, size=10),
    ))
})
```

For each of the 1000 feature, p-value is a probability of observing its value, if we assume it follows a Null hypothesis (sampled from a Normal distribution with 0 mean).

Cummulative Normal distribution `norm.cdf()` represents the probability of obtaining a value equal to or less than the one observed, so `1 - norm.cdf()` is a p-value:

```
df['p_value'] = 1 - norm.cdf(df['x'], loc = 0, scale = 1)

df
```

False Positive Rate

The first concept is called a False Positive Rate, and is defined as a fraction of null hypothesis that we flag as "significant" (also called Type I errors). The "raw" p-values calculated above represent false positive rate by their very definition: small p-values are simply the probabilities of obtain value at least as large when we sampled null distribution.

If we apply a common (albeit somewhat arbitrary) threshold of 0.05:

```
df['is_raw_p_value_significant'] = df['p_value'] <= 0.05
df.groupby(['hypothesis', 'is_raw_p_value_significant']).size()
```

notice that 53 out of 990 null hypotheses are flagged as "significant". The False Positive Rate: FPR = 53 / (53 + 937) = 0.053.

However, the main problem with FPR is that in a real scenario we do not a priori know which hypotheses are null and which are not. Then, the raw p-value on its own (False Positive Rate) can be of little use. In our case when the fraction of non-null features is very small, most of the features flagged as significant will be null, since there are many more of them.

Out of 59 features flagged true ("positive"), 53 are from null distributio, so false positives. That means that about 90% of reported significant features (53 / 59) are false, and is not particularly useful.

There are two common methods of addressing this issue: Family-Wise Error Rate and False Discovery Rate.

Family-Wise Error Rate

The earliest and perhaps the most popular method for correcting for multiple hypothesis testing is a Bonferroni procedure.