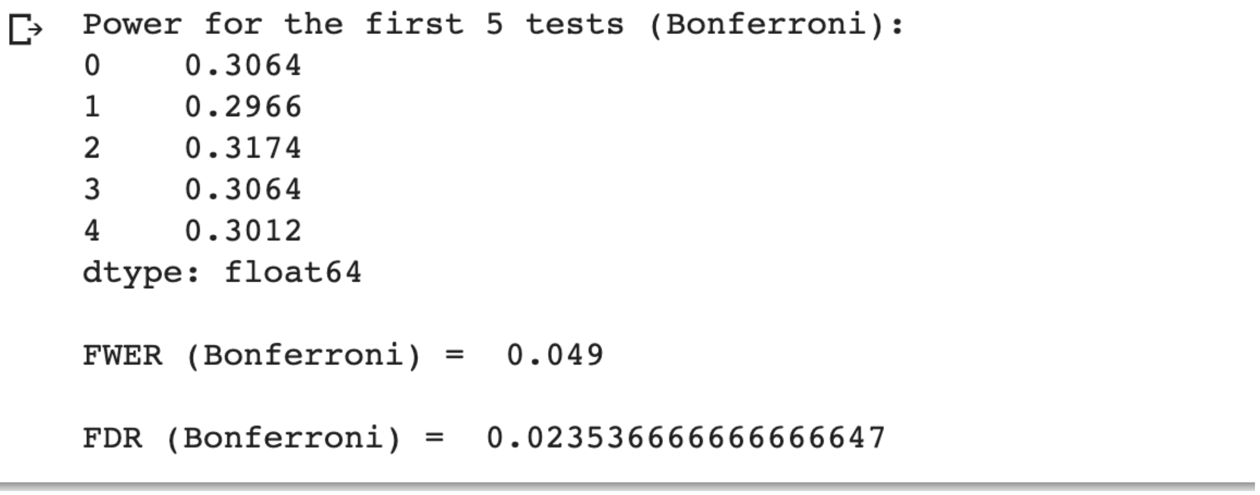
ECON626_Lab5

Qingwen Wang

1 Weaknesses of the Bonferroni correction

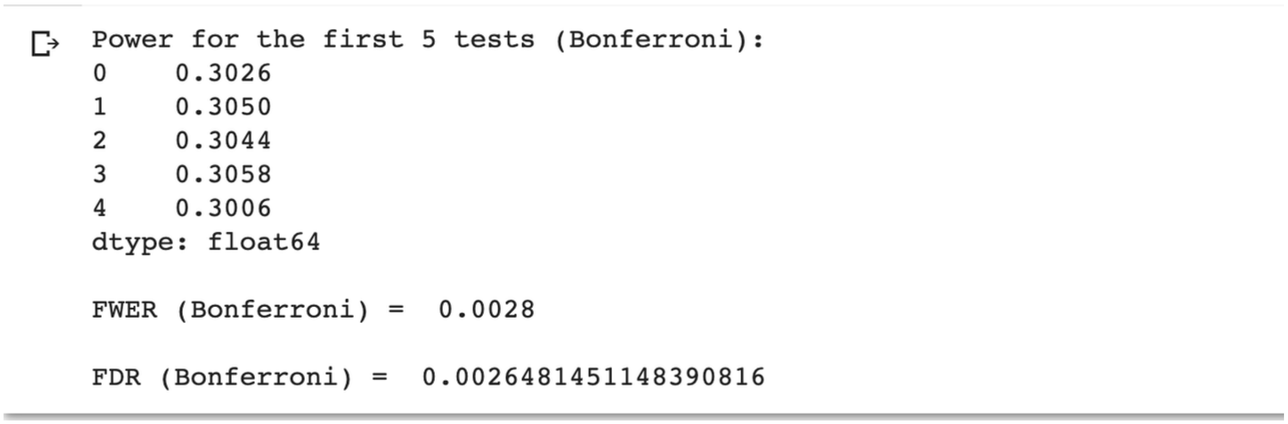
1.1 In-class foundation

1. The dependent test is testing the null hypothesis that there are no differences between the means of the two related groups. Here the covariance exits in the dataset; covariance is between 0 and 1; and the `np.random.multivariate_normal()` returns the set of a correlated random variables.
2. The dataset is random normal distribution; the notebook conducts independent hypothesis, so that it is uncorrelated.
   The power in two experiments after correction do not have big difference, they are stay on a low level (30%); while the FWER has a sharp drop when there is high covariance (0.95), from 5% to 0.28%, be very far away from $\alpha = 0.05$, so we say that it is very conservative.



```
Power for the first 5 tests (Bonferroni):
0    0.3064
1    0.2966
2    0.3174
3    0.3064
4    0.3012
dtype: float64

FWER (Bonferroni) =  0.049

FDR (Bonferroni) =  0.02353666666666647
```

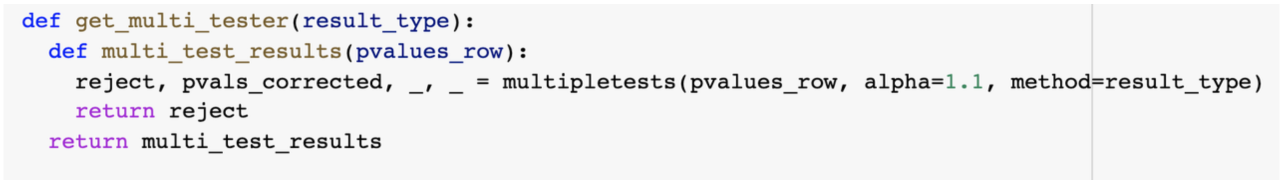
cov = 0



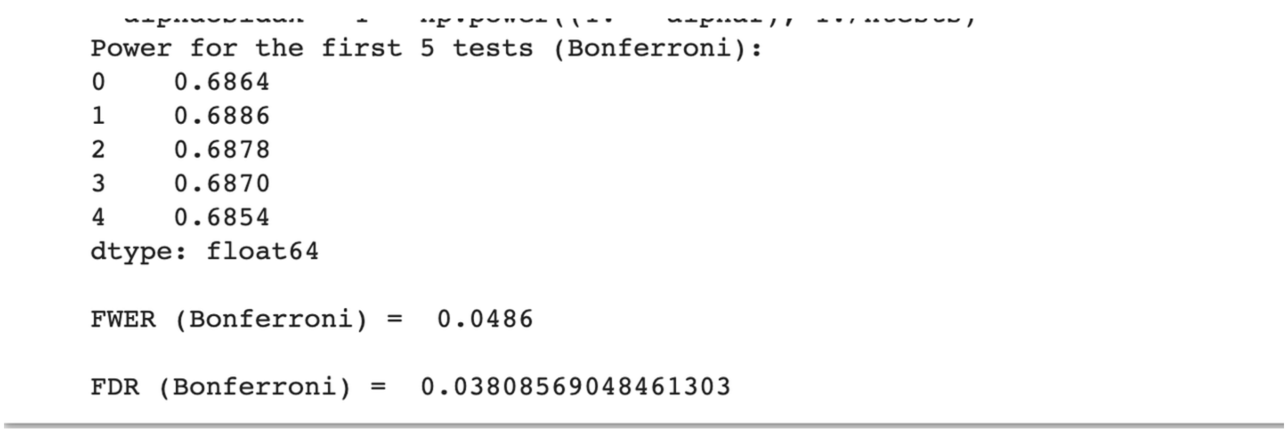
```
Power for the first 5 tests (Bonferroni):
0     0.3026
1     0.3050
2     0.3044
3     0.3058
4     0.3006
dtype: float64

FWER (Bonferroni) =  0.0028

FDR (Bonferroni) =  0.0026481451148390816
```

cov = 0.95

3. set alpha = 1.1 to make FWER near to 0.05; we can see that even the alpha is very high, the power is not too high, only 0.69. Even if the fwer is controlled, this results in a loss of power. With respect to fwer control, the Bonferroni correction can be conservative if there are a large number of tests and/or the test statistics are positively correlated. The correction comes at the cost of increasing the probability of reducing statistical power.



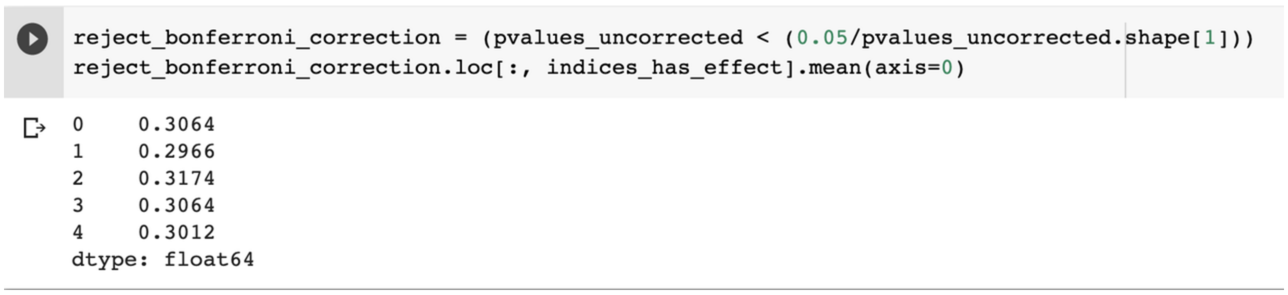
```python
def get_multi_tester(result_type):
  def multi_test_results(pvalues_row):
    reject, pvals_corrected, _, _ = multipletests(pvalues_row, alpha=1.1, method=result_type)
    return reject
  return multi_test_results
```



```
Power for the first 5 tests (Bonferroni):
0     0.6864
1     0.6886
2     0.6878
3     0.6870
4     0.6854
dtype: float64

FWER (Bonferroni) =  0.0486

FDR (Bonferroni) =  0.03808569048461303
```

4.

```
reject_bonferroni_correction = (pvalues_uncorrected < (0.05/pvalues_uncorrected.shape[1]))
reject_bonferroni_correction.loc[:, indices_has_effect].mean(axis=0)
```

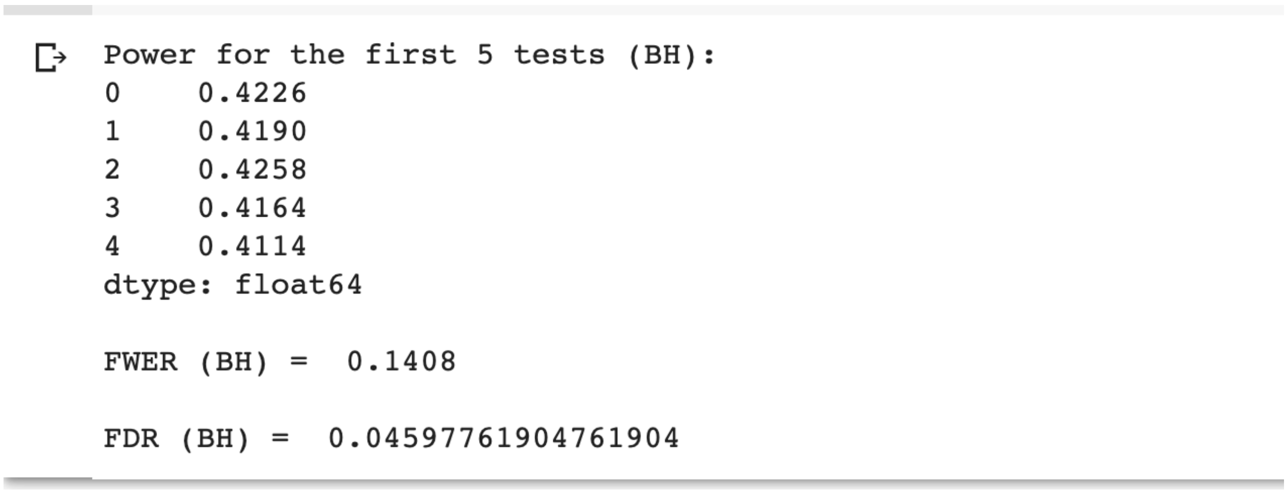


```
reject_bonferroni_correction = (pvalues_uncorrected < (0.05/pvalues_uncorrected.shape[1]))
reject_bonferroni_correction.loc[:, indices_has_effect].mean(axis=0)
```

```
0    0.3064
1    0.2966
2    0.3174
3    0.3064
4    0.3012
dtype: float64
```

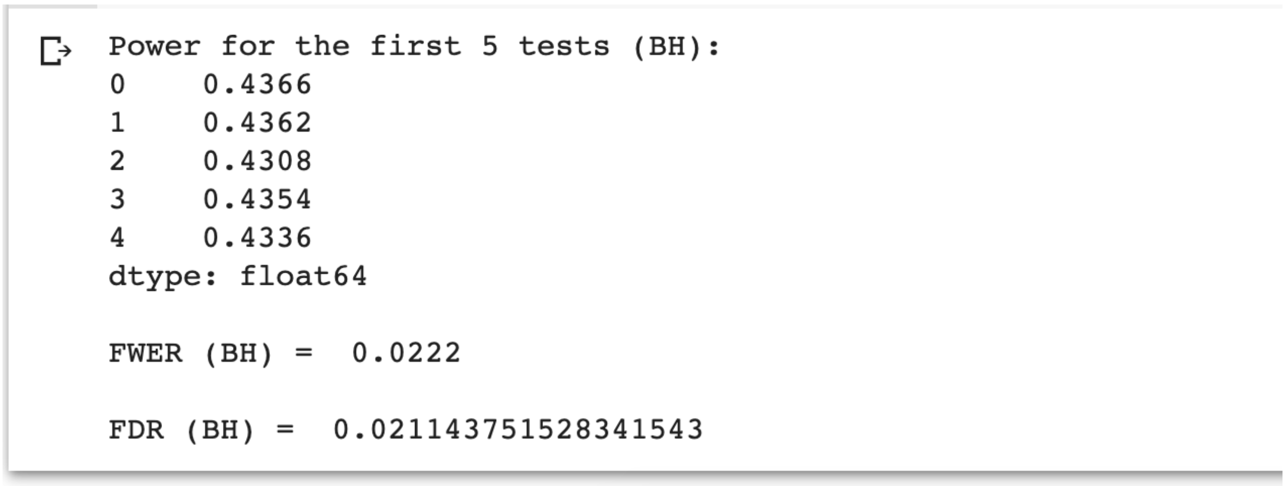
1.2 Homework extension

1. When cov = 0, The FDR is 0.046. We set an error rate of 0.05, so BH has successfully kept the error rate below 0.05. The FWER for the BH correction is about 0.14. The BH is not designed to control the FWER, so we should not be surprised that the FWER is above 0.05. Power is 0.42, not a high performance.
   When cov = 0.95, The FDR is 0.02, lower than that when cov= 0. The FWER for the BH correction is about 0.02, has a better performance than that when cov =0 ; power is 0.43 still not a high performance, make no big difference with that when cov = 0.
   Since the fwer and fdr under Bonferroni correction decreases by 10 times when switch cov=0 to cov =0.95 and power do not have big difference; while the fwer and fdr under BH decrease by 7 times and 2 times, and power also has no big change, we can say that Bonferroni correction is more sensitive to dependence in the test.



```
Power for the first 5 tests (BH):
0    0.4226
1    0.4190
2    0.4258
3    0.4164
4    0.4114
dtype: float64

FWER (BH) =  0.1408

FDR (BH) =  0.04597761904761904
```

cov = 0



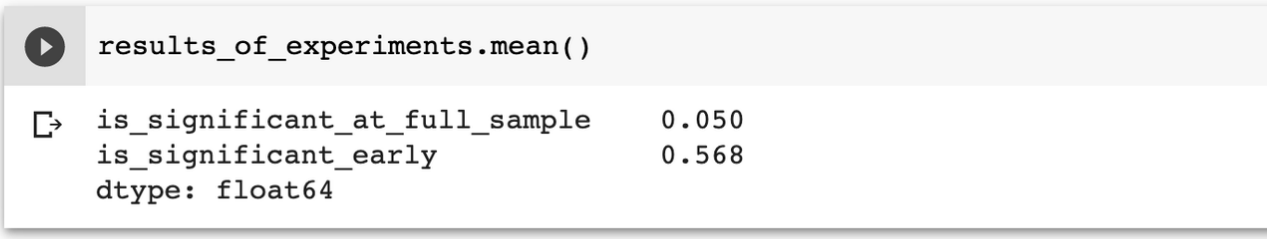
```
Power for the first 5 tests (BH):
0    0.4366
1    0.4362
2    0.4308
3    0.4354
4    0.4336
dtype: float64

FWER (BH) =  0.0222

FDR (BH) =  0.021143751528341543
```

cov = 0.95

2. Depending on the correlation structure of the tests, the Bonferroni correction could be extremely conservative, leading to a high rate of false negatives and reducing the statistical power. When doing Bonferroni correction, we correct p_i value by dividing the original α-value by the number of analyses on the dependent variable, which $p_i = \alpha/M$, and new $\alpha_{\text{critical}} = 1 - (1 - p_i)^k$. This will in actuality result in a true alpha of significant less than 0.05, therefore raising the rate of false negatives, failing to identify an unnecessarily high percentage of actual significant differences in the data. So when adding correlation, P(A⋂B) becomes more larger than P(A)*P(B). ($P(A) = \{P_A > \alpha/M\}, P(B) = \{P_B > \alpha/M\}$, so false negative becomes larger, i.e. power becomes smaller.
3. For example, suppose we sample from students in a school and are interested whether higher Math score predict higher history scores . By accident we sampled students that suggest there is such a relationship whereas in fact the population does not have such a relationship: we picked a sample that produced a False Alarm. Now, we test a second hypothesis from the same sample, namely, that higher Math score predict higher astronomy score. Suppose there is a perfect correlation between math score and astronomy score; our second analysis will produce another False Alarm.
4. $\alpha * \alpha = 0.05, \alpha = 0.224$
5. Each element on the principal diagonal of a correlation matrix is the correlation of a random variable with itself, which always equals to 1.
6. In our case, cov > 0, they are positively correlated P(A|B) > P(A), so $P\{\bigcap(p_i > \alpha/M)\} \geq \prod\{P(p_i > \alpha/M\}$.

2 Early-stopping

1. Set N = 5000, B = 500, false positive rate is 0.05 at full sample, and 0.568 at early stopping. So the larger sample size can make early stopping less of a problem.

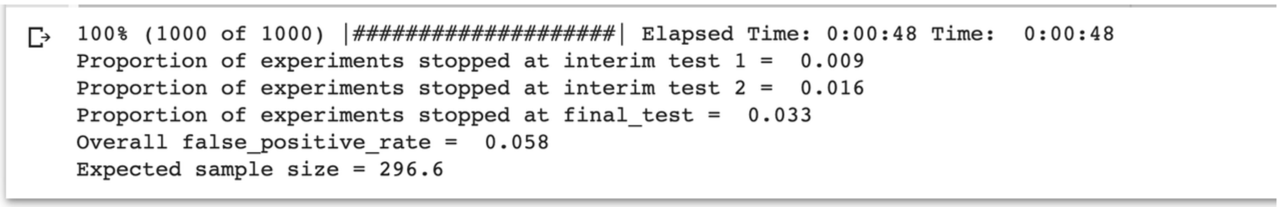


```
results_of_experiments.mean()
```

```
is_significant_at_full_sample    0.050
is_significant_early             0.568
dtype: float64
```

2. overall false positive rate is 0.058, so it controls the false positive rate.

```
OBrienFleming_BOUNDARY_3_ANALYSES = [0.006, 0.0151, 0.0471]
```

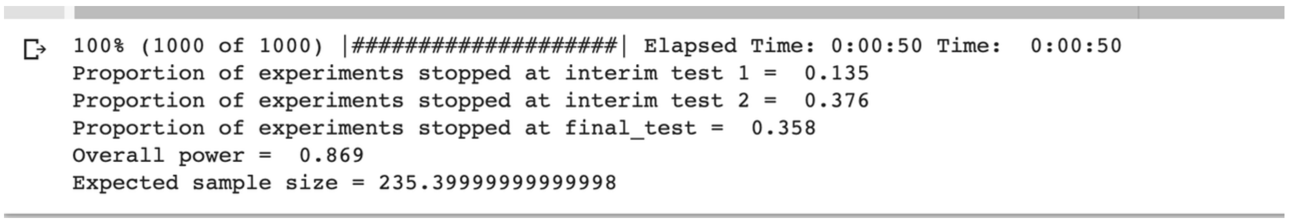


```
100% (1000 of 1000) |####################| Elapsed Time: 0:00:48 Time:  0:00:48
Proportion of experiments stopped at interim test 1 =  0.009
Proportion of experiments stopped at interim test 2 =  0.016
Proportion of experiments stopped at final_test =  0.033
Overall false_positive_rate =  0.058
Expected sample size = 296.6
```

ATE = 0

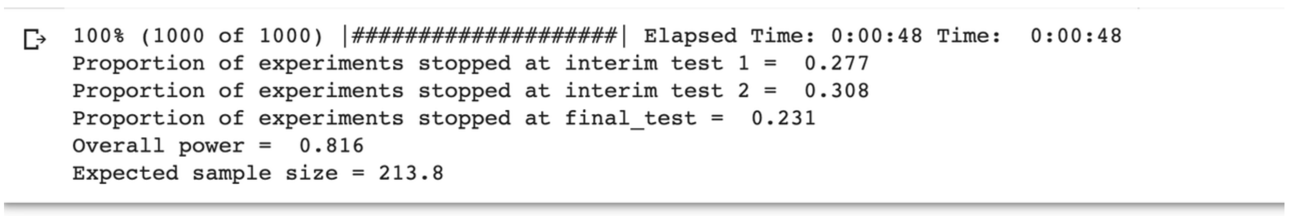
Compare power and sample size:
power: fixed sample size > O'Brien-Fleming boundary, O'Brien-Fleming boundary > Pocock boundary
sample size: fixed sample size > O'Brien-Fleming boundary, O'Brien-Fleming boundary > Pocock boundary
So, by applying sequential analysis can allows you to correctly maintain the overall false positive rate and strong power, even the power may a slight decrease (but still at good performance lovel), and saves the sample size.

O'Brien-Fleming boundary, ATE = 0.05:



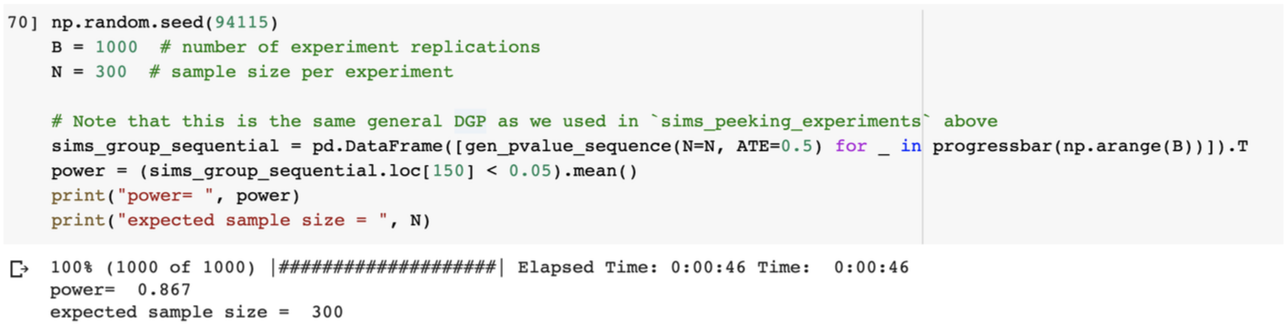
```
100% (1000 of 1000) |####################| Elapsed Time: 0:00:50 Time:  0:00:50
Proportion of experiments stopped at interim test 1 =  0.135
Proportion of experiments stopped at interim test 2 =  0.376
Proportion of experiments stopped at final_test =  0.358
Overall power =  0.869
Expected sample size = 235.39999999999998
```

Pocock boundary, ATE = 0.05:



```
100% (1000 of 1000) |####################| Elapsed Time: 0:00:48 Time:  0:00:48
Proportion of experiments stopped at interim test 1 =  0.277
Proportion of experiments stopped at interim test 2 =  0.308
Proportion of experiments stopped at final_test =  0.231
Overall power =  0.816
Expected sample size = 213.8
```

Fixed sample size, ATE = 0.05:



```
70] np.random.seed(94115)
    B = 1000  # number of experiment replications
    N = 300  # sample size per experiment

    # Note that this is the same general DGP as we used in `sims_peeking_experiments` above
    sims_group_sequential = pd.DataFrame([gen_pvalue_sequence(N=N, ATE=0.5) for _ in progressbar(np.arange(B))]).T
    power = (sims_group_sequential.loc[150] < 0.05).mean()
    print("power= ", power)
    print("expected sample size = ", N)
```

```
100% (1000 of 1000) |####################| Elapsed Time: 0:00:46 Time:  0:00:46
power=  0.867
expected sample size =  300
```

3. If just literally looking at the result will not affect the result unless the researchers take some actions . The early stopping problem occurs when the researcher stops the experiment based on results before the planned sample size is reached. For example, the researcher might check the results of the experiment each day. When the *t*-test happens to have a p-value below 0.05 the researcher stops the experiment and declares a statistically significant result.

4. Such as experiments in medicine field, the treatment group takes a kind of new drugs, we are not sure if there will have serious negative effect on human health. In such situation, peeking is needed for researchers to decides whether this experiment should go on or just be ended. In the airbnb dashboard, they have multiple outcome variables(10 metrics), multiple subgroups analyzed (195 countries), multiple comparisons between subgroups (2 treatments), and multiple repetitions of same test over time (31 days), that is totally 120,900 tests. Multiple test is a problem. When doing multiple hypothesis test, false positive rate will goes up. In this case, P(at least one significant result) = 1 - (1 - 0.05)^120900 = 100%, we have a 100% chance of observing at least one significant result, even if all of the tests are actually not significant, which means the probability of getting a significant result is totally by chance.