Hypothesis Testing in Python

We use statistics in a lot of different ways in data science.
In this lecture, I want to refresh your knowledge of hypothesis testing, which is a core data analysis activity behind experimentation.
We're starting to see experimentation used more and more commonly outside of academic scientific, and in day to day business environments.
Part of the reason for this is the rise of big data and web commerce.
It's now easy to change your digital storefront and deliver a different experience to some of your customers, and then see how those customer actions might differ from one another.

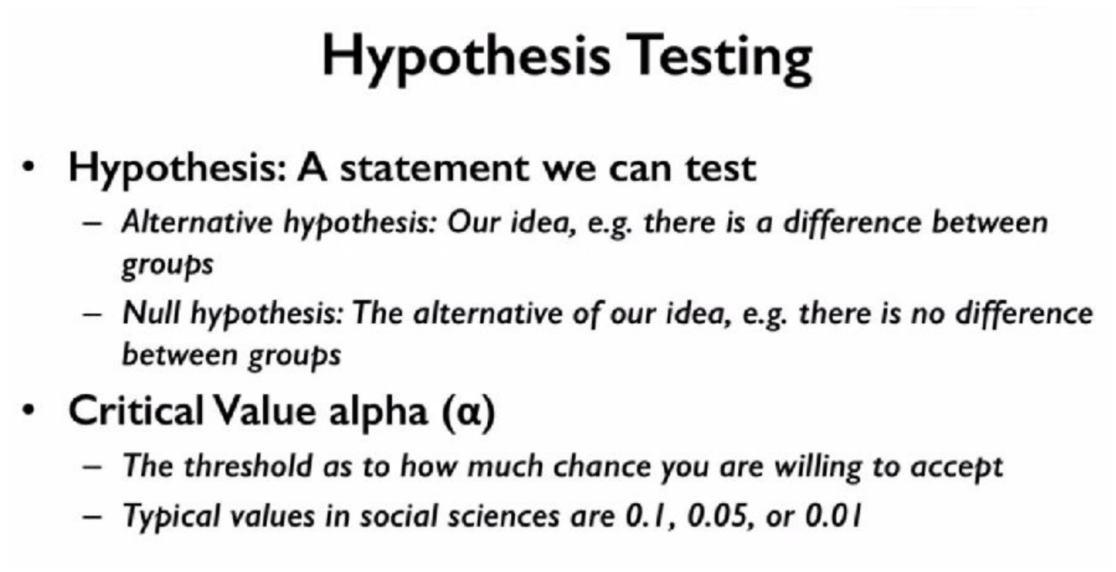
For instance, if you sell books, you might want to have one condition where the cover of the book is featured on the web page and another condition where the focus is on the author and the reviews of the book.
This is often called A/B testing. And while it's not unique to this time in history, it's now becoming so common that if you're using a website, you are undoubtedly part of an A/B test somewhere.

A/B testing is a way to compare two versions of a single variable, typically by testing a subject's response to variant A against variant B, and determining which of the two variants is more effective.

A/B testing is comparing two versions of a webpage or app against each other to determine which one performs better

A hypothesis is a statement that we can test.
I'll pull an example from my own research of educational technology and learning analytics.
Let's say that we have an expectation that when a new course is launched on a MOOC platform, the keenest students find out about it and all flock to it.
Thus, we might expect that those students who sign up quite quickly after the course is launched with higher performance than those students who signed up after the MOOC has been around for a while.
In this example, we have samples from two different groups which we want to compare.
The early sign ups and the late sign ups.

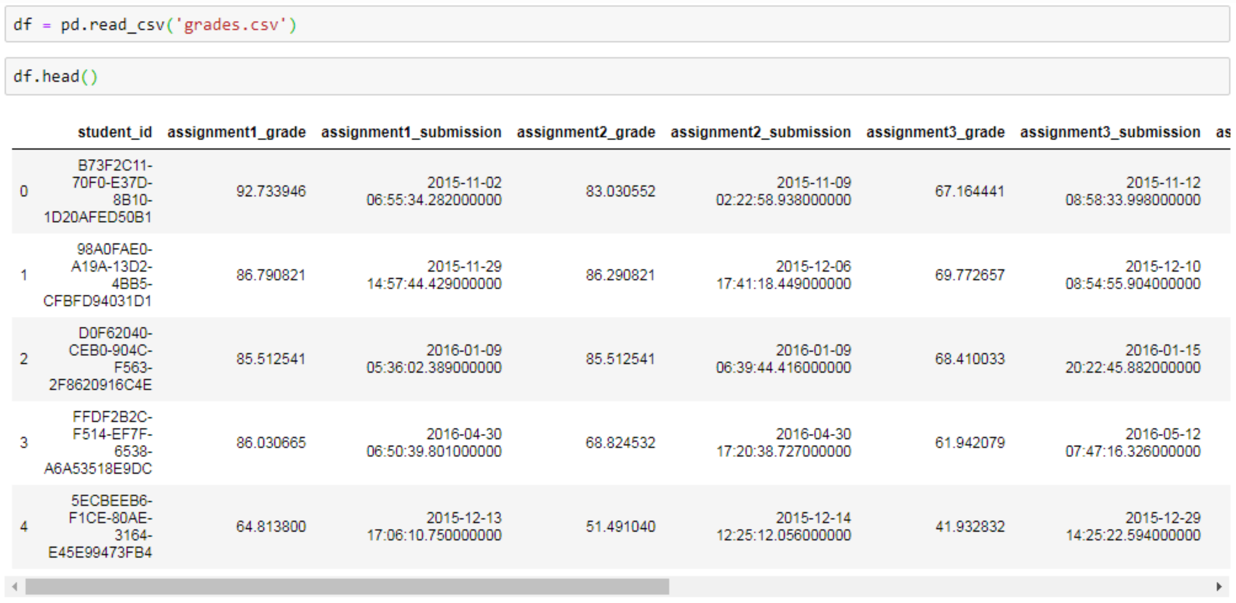
When we do hypothesis testing, we hold out that our hypothesis as the alternative and we create a second hypothesis called the null hypothesis, which in this case would be that there is no difference between groups.
We then examine the groups to determine whether this null hypothesis is true or not.



Hypothesis Testing

- Hypothesis: A statement we can test
  - *Alternative hypothesis: Our idea, e.g. there is a difference between groups*
  - *Null hypothesis: The alternative of our idea, e.g. there is no difference between groups*
- Critical Value alpha (α)
  - *The threshold as to how much chance you are willing to accept*
  - *Typical values in social sciences are 0.1, 0.05, or 0.01*

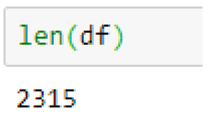
If we find that there is a difference between groups, then we can reject the null hypothesis and we accept our alternative.
There are subtleties in this description. We aren't saying that our hypothesis is true per se, but we're saying that there's evidence against the null hypothesis.
So, we're more confident in our alternative hypothesis.
Let's see an example of this. We can load a file called grids.csv.
If we take a look at the dataframe inside, we see we have six different assignments. Each with a submission time and it looks like there just under 3,000 entries in this data file.



```
df = pd.read_csv('grades.csv')
```

```
df.head()
```

	student_id	assignment1_grade	assignment1_submission	assignment2_grade	assignment2_submission	assignment3_grade	assignment3_submission	as
0	B73F2C11-70F0-E37D-8B10-1D20AFED50B1	92.733946	2015-11-02 06:55:34.282000000	83.030552	2015-11-09 02:22:58.938000000	67.164441	2015-11-12 08:58:33.998000000	
1	98A0FAE0-A19A-13D2-4BB5-CFBFD94031D1	86.790821	2015-11-29 14:57:44.429000000	86.290821	2015-12-06 17:41:18.449000000	69.772657	2015-12-10 08:54:55.904000000	
2	D0F62040-CEB0-904C-F563-2F8620916C4E	85.512541	2016-01-09 05:36:02.389000000	85.512541	2016-01-09 06:39:44.416000000	68.410033	2016-01-15 20:22:45.882000000	
3	FFDF2B2C-F514-EF7F-6538-A6A53518E9DC	86.030665	2016-04-30 06:50:39.801000000	68.824532	2016-04-30 17:20:38.727000000	61.942079	2016-05-12 07:47:16.326000000	
4	5ECBEEB6-F1CE-80AE-3164-E45E99473FB4	64.813800	2015-12-13 17:06:10.750000000	51.491040	2015-12-14 12:25:12.056000000	41.932832	2015-12-29 14:25:22.594000000	



```
len(df)
```

```
2315
```

For the purpose of this lecture, let's segment this population into two pieces. Those who finish the first assignment by the end of December 2015 and those who finish it sometimes after that.

```
early = df[df['assignment1_submission'] <= '2015-12-31']
late = df[df['assignment1_submission'] > '2015-12-31']
```

I just made this date up and it gives us two dataframes, which are roughly the same size.

As you've seen, the pandas dataframe object has a variety of statistical functions associated with it.
If we call the mean function directly on the dataframe, we see that each of the means for the assignments are calculated.

```
early.mean()
```

```
assignment1_grade    74.972741
assignment2_grade    67.252190
assignment3_grade    61.129050
assignment4_grade    54.157620
assignment5_grade    48.634643
assignment6_grade    43.838980
dtype: float64
```

```
late.mean()
```

```
assignment1_grade    74.017429
assignment2_grade    66.370822
assignment3_grade    60.023244
assignment4_grade    54.058138
assignment5_grade    48.599402
assignment6_grade    43.844384
dtype: float64
```

Note that the date time values are ignored as panda's knows this isn't a number, but an object type.
If we look at the mean values for the late dataframe as well, we get surprisingly similar numbers.
There are slight differences, though. It looks like the end of the six assignments; the early users are doing better by about a percentage point.

So, is this enough to go ahead and make some interventions to actually try and change something in the way we teach?
When doing hypothesis testing, we have to choose a significance level as a threshold for how much of a chance we're willing to accept.

This significance level is typically called alpha.
It can vary greatly, depending on what you're going to do with the result and the amount of noise you expect in your data.

A critical value is a line on a graph that splits the graph into sections. One or two of the sections is the “rejection region”; if your test value falls into that region, then you reject the null hypothesis.

For instance, in social sciences research, a value of 0.05 or 0.01 is often used, which indicates a tolerance for a probability of between 5% and 1% of chance.
In a physics experiment where the conditions are much more controlled and thus, the burden of proof is much higher, you might expect to see alpha levels of 10 to the negative 5 or 100,000th of a percentage.

You can think of the significance level from the perspective of interventions as well and this is something I run into regularly with my research.
What am I going to do when I find out that two student populations are different?
For instance, if I'm going to send an email nudge to encourage students to continue working on their homework, that's a pretty low-cost intervention.
Emails are cheap and while I certainly don't want to annoy students, one extra email isn't going to ruin their day.
But what if the intervention is a little more involved, like having our tutorial assistant follow up with a student via phone?
This is all of a sudden much more expensive for both the institution and for the student.
So, I might want to ensure a higher burden of proof.

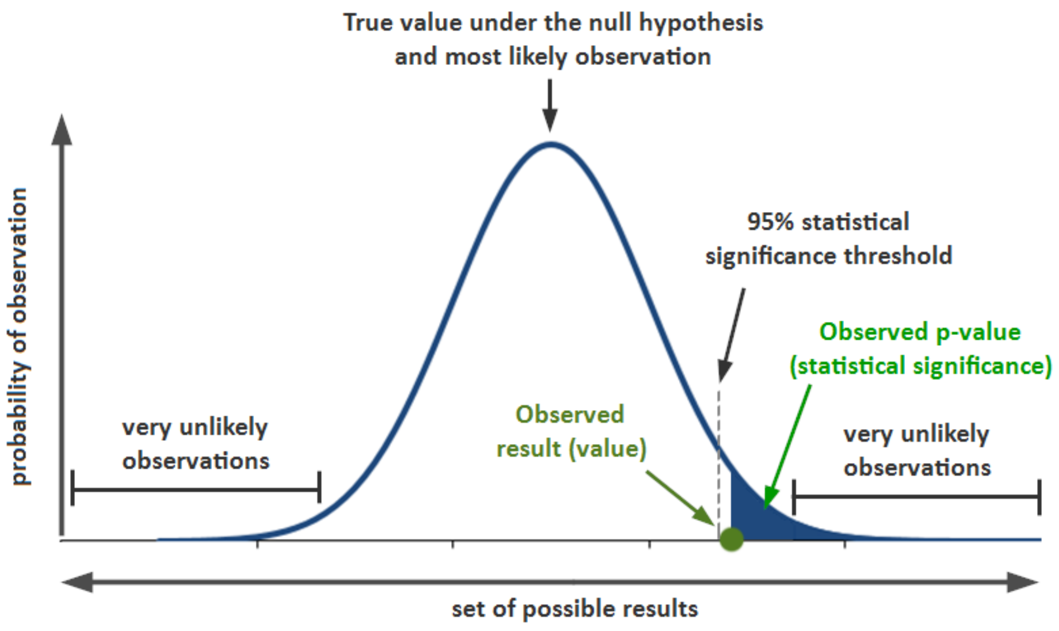
So the threshold you set for alpha depends on what you might do with the result, as well.
For this example, let's use a threshold of 0.05 for our alpha or 5%.
Now, how do we actually test whether these means are different in Python?
The SciPy library contains a number of different statistical tests and forms a basis for hypothesis testing in Python.
A Ttest is one way to compare the means of two different populations.
In the SciPy library, the Ttest end function will compare two independent samples to see if they have different means.

Note that most statistical tests expect that the data conforms to a certain distribution, a shape.
So, you shouldn't apply such tests blindly and should investigate your data first.
If we want to compare the assignment grades for the first assignment between the two populations, we could generate a Ttest by passing these two series into the Ttest_ind function.
The result is a two with a test statistic and a p-value.
The p-value here is much larger than our 0.05.

```
from scipy import stats
stats.ttest_ind?
```

```
stats.ttest_ind(early['assignment1_grade'], late['assignment1_grade'])
```

```
Ttest_indResult(statistic=1.400549944897566, pvalue=0.1614828301606057)
```



The p-value corresponds to the probability of observing sample data at least as extreme as the actually obtained test statistic. Small p-values provide evidence against the null hypothesis. The smaller (closer to 0) the p-value, the stronger is the evidence against the null hypothesis.
When you perform a statistical test a p-value helps you determine the significance of your results in relation to the null hypothesis.

The null hypothesis states that there is no relationship between the two variables being studied (one variable does not affect the other). It states the results are due to chance and are not significant in terms of supporting the idea being investigated. Thus, the null hypothesis assumes that whatever you are trying to prove did not happen.

The alternative hypothesis is the one you would believe if the null hypothesis is concluded to be untrue.

The alternative hypothesis states that the independent variable did affect the dependent variable, and the results are significant in terms of supporting the theory being investigated (i.e. not due to chance).

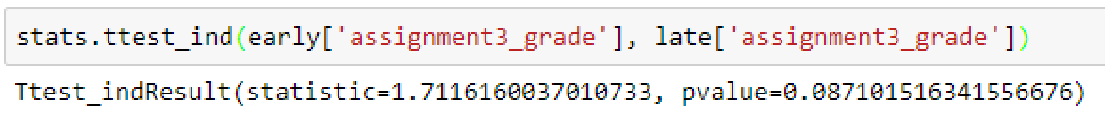
So we cannot reject the null hypothesis, which is that the two populations are the same.
In more late terms, we would say that there's no statistically significant difference between these two sample means.
Let's check with assignment2 grade.

```
stats.ttest_ind(early['assignment2_grade'], late['assignment2_grade'])
```

```
Ttest_indResult(statistic=1.3239868220912567, pvalue=0.18563824610067967)
```

No, that's much larger than 0.052.

How about with assignment3?

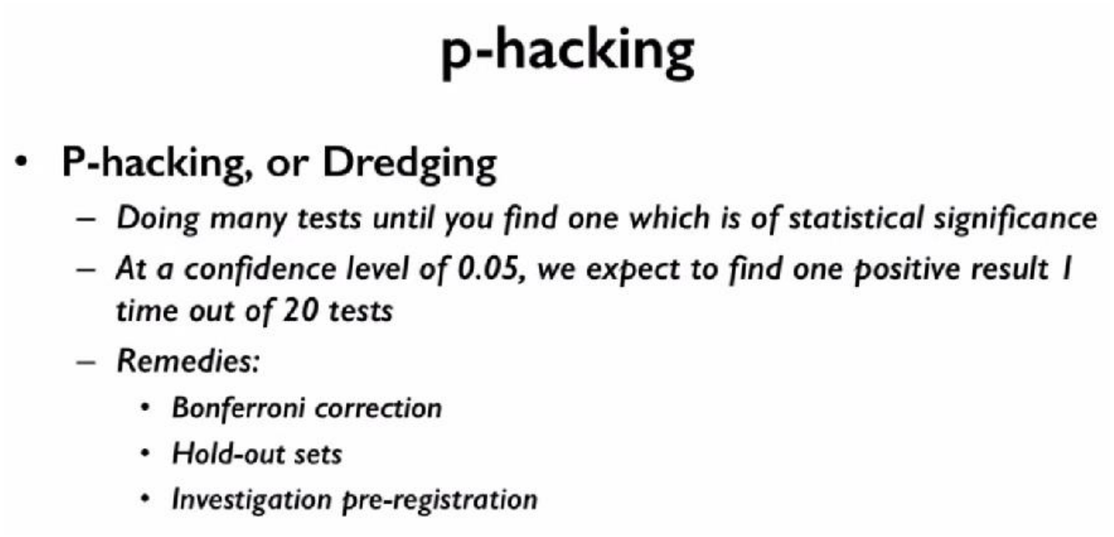


```
stats.ttest_ind(early['assignment3_grade'], late['assignment3_grade'])
```

```
Ttest_indResult(statistic=1.7116160037010733, pvalue=0.087101516341556676)
```

Well, that's much closer, but still beyond our threshold value.
It's important to stop here and talk about serious process from with how we're handling this investigation of the difference between these two populations.
When we set the alpha to be 0.05, we're saying that we expect it that there will be positive result, 5% of the time just to the chance.

As we run more and more Ttests, we're more likely to find a positive result just because of the number of Ttests we have run.



p-hacking

- P-hacking, or Dredging
  - *Doing many tests until you find one which is of statistical significance*
  - *At a confidence level of 0.05, we expect to find one positive result 1 time out of 20 tests*
  - *Remedies:*
    - *Bonferroni correction*
    - *Hold-out sets*
    - *Investigation pre-registration*

P-hacking (also known as data dredging, data fishing, data snooping) is the use of data mining to discover patterns which are presented as statistically significant, but the analysis is done by exhaustively searching various combinations of variables for correlation.

When a data scientist run many tests in this way, it's called p-hacking or dredging and it's a serious methodological issue.

P-hacking results in spurious correlations instead of generalizable results.
There are a couple of different ways you can deal with p-hacking.

- The first is called the Bonferroni correction.
  In this case, you simply tighten your alpha value, the threshold of significance based on the number of tests you're running.
  So if you choose 0.05 with 1 test, you want to run 3 test, you reduce alpha by multiplying 0.05 by one-third to get a new value of 0.01 sub.

I personally find this approach to be very conservative.

- Another option is to hold out some of your data for testing to see how generizable your result is.

In this case, we might take half of our data for each of the two data frames; run our Ttest with that.

Form specific hypothesis based on the result of these tests, then run very limited tests on the rest of the data.

This method is actually heavily used in machine learning when building predictive models where it's called cross fold validation.

- A final method which has come about is the pre-registration of your experiment.

In this step, you would outline what you expect to find and why, and describe the test that would backup a positive proof of this.
You register it with a third party, in academic circles; this is often a journal who determines whether it's a reasonable test or rather not.

You then run your study and report the results, regardless as to whether they were positive or not.
Here, there is a larger burden on connecting to existing theory since you need to convince reviewers that the experiment is likely to test fully a given hypothesis.

In this lecture, we've discussed just some of the basics of hypothesis testing in Python.
I introduced you to the SciPy library, which you can use for T testing.
We've discussed some of the practical issues which arise from looking for statistical significance.
There's much more to learn about hypothesis testing. For instance, there are different tests used, depending on the shape of your data and different ways to report results instead of just p-values such as confidence intervals.
But I hope this gives you a start to comparing the means of two different populations, which is a common task for data scientists.

This lecture also completes the lectures for the first course in the Applied Data Science with Python Specialization.
We've covered the basics of Python programming, some more advanced features like maps, lambdas and miscomprehensions.
How to read and manipulate data using the panda's library, including querying, joining, grouping and processing of dataframes and the creation of pivot tables.
And now we've talked a little bit about statistics in Python and dug deeper into the Num-Py and SciPy toolkit.