CSCI3022 S23

Homework 9: Hypothesis Testing and Confidence Intervals

Due Monday, April 10th at 11:59pm to Gradescope

Name: _____Rocky Erdenebat_____

Collaboration Policy

While completing the assignment you are not allowed to consult any source other than the course textbooks/online reference links provided on Canvas, your own class notes, and/or the posted lecture slides/in-class Jupyter notebooks. You may discuss questions you have with your classmates or on Piazza or in office hours, but all work you submit must be your own, which means when writing up your solutions or code, you MUST do it entirely by yourself.

You should be able to easily reproduce from scratch and explain a solution that was your own when asked in office hours by a TA/Instructor or on a quiz/exam without referencing your notes/book/HW.

Do not search/ask for a solution online: You may not actively search for a solution to the problems below from the internet. This includes posting to or using sources like ChatGPT, StackOverflow, StackExchange, Reddit, Chegg, CourseHero, etc.

We are here to help! Visit HW Hours and/or post questions on Piazza!

Copying/consulting from the solution of another classmate or an online solution (or providing a classmate your solution) constitutes a **violation of the course's collaboration policy and the honor code and will result in an F in the course and a trip to the honor council**.

Instructions for Submitting in Correct Format

You must submit a PDF of this Juptyer notebook to Gradescope by the deadline listed above. Submissions that are not a PDF or that are not submitted to Gradescope will not be counted for credit.

Before submitting your PDF, make sure that your LaTeX has rendered correctly in you
Any of your solutions with incorrectly rendered or incompletely rendered LaTeX will be

- There are several ways to quickly make a .pdf out of this notebook for Gradescope submission.
- If you are running Juptyer locally on your computer:
  - Option1 : Select Kernel->Restart & Run All. Then select File -> Print Preview, and then Right-Click -> Print using your default browser and "Print to PDF"
  - Option 2: Select Kernel->Restart & Run All. Then select File -> Download as PDF via LaTeX. This will require your system path find a working install of a TeX compiler
- If you are running using CSEL:
  - Option1 : Go to File ->Save & Export Notebook As-> HTML. Then open the HTML, and then Right-Click -> Print and select "Print to PDF".
  - Option2 : Go to File ->Download. Then use this converter https://htmtopdf.herokuapp.com/ipynbviewer/ to convert ipynb to pdf.

Notes

- For full points you must correctly match your questions to the respective Gradescope problem, and include clear comments in your code. Please note that any LaTeX that is not correctly rendered in your submitted PDF will result in a 0 on the entire problem(s) that involves the unrendered LaTeX.
- You **must show all work and justify ALL answers to receive credit**. Sparse or nonexistent work will receive sparse or nonexistent credit.
- Any relevant data sets are available on Canvas.
- LaTeX Tips: Here is a [reference guide] (https://math.meta.stackexchange.com/questions/5020/mathjax-basic-tutorial-and-quick-reference). **All** of your written commentary, justifications and mathematical work should be in Markdown. I also recommend the wikibook for LaTex.
- Because you can technically evaluate notebook cells is a non-linear order, it's a good idea to do **Kernel** $\rightarrow$ **Restart & Run All** as a check before submitting your solutions.
- It is **bad form** to make your reader interpret numerical output from your code. If a question asks you to compute some value from the data you should show your code output **AND** write a summary of the results in Markdown directly below your code.
- There is *not a prescribed API* for these problems. You may answer coding questions with whatever syntax or object typing you deem fit. Your evaluation will primarily live in the clarity of how well you present your final results, so don't skip over any interpretations!

clarity of how well you present your final results, so don't skip over any interpretations! Your code should still be commented and readable to ensure you followed the given course algorithm.

We'll need Numpy, Matplotlib, Pandas, and scipy.stats for this notebook, so let's load them.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pylab as plt
import scipy.stats as stats
import statsmodels.stats.api as sms
%matplotlib inline
```

Shortcuts: Problem 1 | Problem 2 | Problem 3 |Problem 4

Back to top

(7 pts) Problem 1: How did we do?

NFL Overtime

On Homework 6 we tested the idea that the first team might be favored in an NFL overtime by estimating the proportion of games where the first team to receive the ball won the overtime compared to the number of games won by the 2nd team to receive the ball. Suppose we had approached this problem as a hypothesis test, wherein we we would propose a change to a new format if the data suggests a bias towards either team with a significance of 10%.

1A) (2 pts) Let p_1 be the probability the first team to receive the ball wins the game and p_2 be the probability the 2nd team to receive the ball wins the game. State the null and alternative hypothesis for this experiment, both in "H_0" notation as stated as plain English sentences.

The null hypothesis is that both teams have an equal probability of winning the game. The Alternative hypothesis is that one team is more likely to win than the other. $H_0 = p_1 = p_2$ $H_a = p_1 \neq p_2$

1B) (i) (1 pt) Use **your exact simulation** results from the "NFL overtime" problem of HW 6 Problem 4 as data (i.e. find the proportion of times Team 1 won in your simulation compared to the proportion of times Team 2 won) to conduct an appropriate hypothesis test (using a built-in function from Python) and output the p-value for your hypothesis test:

Considering games won only, team 1 won 4760 times and team 2 won 3509 times for a total of 8269 games. Assuming the game is fair, each team should expect to win half the time.

In [2]:
```
nflips=8269 #sample size
p=0.5

print(stats.binom.ppf(1-.1, n=nflips, p=p))
```

```
4193.0
```

ii) (1 pt) . What decision would you make as a result of this test (i.e. do you reject the null hypothesis or fail to reject the null hypothesis)?

Because the number of games team 1 won is greater than the critical number of games we can expect to win with a significance of 10%,we reject the null hypothesis and accept the alernative hypothesis.

nb08 ln55

iii)(1 pt) There are a pletheora of different hypothesis tests built into Python. Explain any assumptions that are necessary for the hypothesis test you chose and why/how you know your data satisfies those assumption(s).

We assume that the samples were randomly chosen, the sample size is ≥ 30, and finite variance. We know that our data satisfies those assumptions because that's how they were stimulated was using python random numbers.

1C) (2 pts) Actual NFL game data reveals that since the current overtime rules were implemented the team receiving the ball first has won 86 times and tied 10 times out of 164 total overtime games. Rerun your hypothesis test from part b using this data to determine if the data suggests a bias towards either team with a significance of 10%.
i). What is the p-value?

In [3]:
```
nflips=164 #sample size
p=0.5

print(stats.binom.ppf(1-.1, n=nflips, p=p))
```

```
90.0
```

ii). What decision would you make as a result of this test?

Because the first team is within the rejection region of the test $86 \leq 90$, we accept the null hypothesis and assume both teams are equally likely to win regardless of who gets the ball first.

Back to top

[18 points] Problem 2- A/B Test for Conversion Rates

In this problem we'll go over the process of analyzing an A/B test when comparing proportion data (i.e. conversion rates). For our data, we'll use a dataset from Kaggle (`ab_data.csv`), which contains the results of an A/B test on what seems to be 2 different designs of a website page (old_page vs. new_page). Here's what we'll do:

1. Designing our experiment
2. Collecting and preparing the data
3. Visualising the results
4. Testing the hypothesis
5. Drawing conclusions

To make it a bit more realistic, here's a potential **scenario** for our study:

> Let's imagine you work on the product team at a medium-sized **online e-commerce business**. The UX designer worked really hard on a new version of the product page, with the hope that it will lead to a higher conversion rate. The product manager (PM) told you that the **current conversion rate** is about **13%** on average throughout the year, and that the team would be happy with an **increase of 2%**, meaning that the new design will be considered a success if it raises the conversion rate to 15%.

Before rolling out the change, the team would be more comfortable testing it on a small number of users to see how it performs, so you suggest running an **A/B test** on a subset of your user base users.

Designing our experiment

Choosing the variables

For our A/B test we'll need **two groups**:

- A `control` group - They'll be shown the old design
- A `treatment` (or experimental) group - They'll be shown the new design

This will be our *Independent Variable*. The reason we have two groups even though we know the baseline conversion rate is that we want to control for other variables that could have an effect on our results, such as seasonality: by having a `control` group we can directly compare their results to the `treatment` group, because the only systematic difference between the groups is the design of the product page, and we can therefore attribute any

differences in results to the designs.

For our *Dependent Variable* (i.e. what we are trying to measure), we are interested in capturing the `conversion rate`. A way we can code this is by each user session with a binary variable:

- `0` - The user did not buy the product during this user session
- `1` - The user bought the product during this user session

This way, we can easily calculate the mean for each group to get the conversion rate of each design.

Step 1: Formulating a hypothesis

First things first, we want to make sure we formulate a hypothesis at the start of our project. This will make sure our interpretation of the results is correct as well as rigorous.

Let p_1 be the conversion rate in the treatment group (shown the new design) and p_0 be the conversion rate in the control group (shown the old design).
We don't know if the new design will perform better or worse (or the same?) as our current design.

(Part A) (1 pt) State the null and alternative hypothesis for this A/B Test (using "H_0" and "H_A" notation)

H_0 is that there is no effect between the two designs. H_a is that there is a difference between the two designs.

Step 2: Choosing a significance level

For this test we'll set a **significance level** of $\alpha = 0.05$.

The α value is a threshold we set, by which we say "if the probability of observing a result as extreme or more (p-value) is lower than α, then we reject the null hypothesis".

This means that whatever conversion rate we observe for our new design in our test, we want to be 95% confident it is statistically different from the conversion rate of our old design, before we decide to reject the Null hypothesis H_0.

Note that α also gives us our Type I Error rate (i.e. the probability of rejecting the null hypothesis when it is in fact true).

Choosing a sample size

It is important to note that since we won't test the whole user base (our population), the conversion rates that we'll get will inevitably be only *estimates* of the true rates.

The number of people (or user sessions) we decide to capture in each group will have an effect on the precision of our estimated conversion rates: **the larger the sample size**, the more precise our estimates (i.e. the smaller our confidence intervals), **the higher the chance to detect a difference** in the two groups, if present.

On the other hand, the larger our sample gets, the more expensive (and impractical) our study becomes.

So how many people should we have in each group?

The sample size we need is estimated through something called *Power analysis*, and it depends on a few factors:

- **Type I Error Rate** (α) - The significance level we set earlier to 0.05.
- **Type II Error rate** (β) - In this case, we will choose a commonly used Type II error rate cutoff of 0.2
- **Power of the test** ($1 - \beta$) - This represents the probability of finding a statistical difference between the groups in our test when a difference is actually present. In our case, the power is $1 - 0.2 = 0.8$, (here's more info on statistical power, if you are curious)
- **Effect size** - How big of a difference we expect there to be between the conversion rates. Since our team would be happy with a difference of 2%, we can use 13% and 15% to calculate the effect size we expect.

Having set the `power` parameter to 0.8 in practice means that if there exists an actual difference in conversion rate between our designs, assuming the difference is the one we estimated (13% vs. 15%), we have about 80% chance to detect it as statistically significant in our test with the sample size we calculated.

Luckily, **built-in function available in Python takes care of all these calculations for us.** You can read more about these functions here: https://www.statsmodels.org/dev/generated/statsmodels.stats.power.NormalIndPower.html

Run the cell below to determine the required number of samples based on our Type I and Type II error thresholds:

In [4]:
```
effect_size = sms.proportion_effectsize(0.13, 0.15)    # Calculating effect size ba


required_n = sms.NormalIndPower().solve_power(
    effect_size,
    power=0.8,
    alpha=0.05,
    ratio=1
    )                                                  # Calculating sample size ne

required_n = np.ceil(required_n)                       # Rounding up to next who

print("required number of samples is:", required_n)
```

```
required number of samples is: 4720.0
```

Step 3: Collecting and preparing the data

Great stuff! So now that we have our required sample size, we need to collect the data. Usually at this point you would work with your team to set up the experiment, likely with the help of the Engineering team, and make sure that you collect enough data based on the sample size needed.

However, since we'll use a dataset that we found online, in order to simulate this situation we'll:

1. Read the data into a pandas DataFrame
2. Check and clean the data as needed
3. Randomly sample `n=4720` rows from the DataFrame for each group **

*__Note__: Normally, we would not need to perform step 3, this is just for the sake of the exercise

The data file contains the following columns:

- `user_id` - The user ID of each session
- `timestamp` - Timestamp for the session
- `group` - Which group the user was assigned to for that session { `control`, `treatment` }
- `landing_page` - Which design each user saw on that session { `old_page`, `new_page` }
- `converted` - Whether the session ended in a conversion or not (binary, `0` =not converted, `1` =converted)

We'll actually only use the `group` and `converted` columns for the analysis.

(Part B) (1 pt) Read the `ab_data.csv` file into a pandas DataFrame, view the first 5 rows and get the info about the dataframe. How many rows are in the DataFrame?

In [5]:
```
dfData = pd.read_csv('/home/jovyan/3022/homework/hw9/ab_data.csv')
dfData.head(5)
#dfData.info()
```

Out[5]:

	user_id	timestamp	group	landing_page	converted
0	851104	2017-01-21 22:11:48.556739	control	old_page	0
1	804228	2017-01-12 08:01:45.159739	control	old_page	0
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0
3	853541	2017-01-08 18:28:03.143765	treatment	new_page	0
4	864975	2017-01-21 01:52:26.210827	control	old_page	1

In [6]:
```
dfData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 294478 entries, 0 to 294477
Data columns (total 5 columns):
 #   Column        Non-Null Count   Dtype 
---  ------        --------------   ----- 
 0   user_id       294478 non-null  int64 
 1   timestamp     294478 non-null  object
 2   group         294478 non-null  object
 3   landing_page  294478 non-null  object
 4   converted     294478 non-null  int64 
dtypes: int64(2), object(3)
memory usage: 11.2+ MB
```

There's a total of 294478 entries

(Part C) (2 pts) Part of data cleaning is checking that columns in your dataset correctly group users into control and treatment based on your critiera. Check to make sure that each user was correctly categorized into the control or treatment group based on the page they saw. In other words, check if any users who saw the old_page were incorrectly assigned to "control" and vice-versa. Clean the "group" column to correctly assign any incorrectly categorized users.

In [7]:
```
# df = pandas.read_csv("test.csv")
# df.loc[df.ID == 103, 'FirstName'] = "Matt"
# df.loc[df.ID == 103, 'LastName'] = "Jones"
dfData = pd.read_csv('/home/jovyan/3022/homework/hw9/ab_data.csv')

dfData.loc[dfData.landing_page == "old_page", "group"] = "control"
dfData.loc[dfData.landing_page == "new_page", "group"] = "treatment"
dfData.head(5)
```

Out[7]:

	user_id	timestamp	group	landing_page	converted
0	851104	2017-01-21 22:11:48.556739	control	old_page	0
1	804228	2017-01-12 08:01:45.159739	control	old_page	0
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0
3	853541	2017-01-08 18:28:03.143765	treatment	new_page	0
4	864975	2017-01-21 01:52:26.210827	control	old_page	1

(Part D) (2 pts) Each row in the dataset represents a user session. Before we go ahead and sample the data to get our subset, check to make sure there are no users that have been sampled multiple times. If there are, delete them from the dataset (i.e. don't keep any of their entries). How many distinct users (entries) are now in your cleaned dataset?

In [8]:
```
dfData.drop_duplicates()
dfData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 294478 entries, 0 to 294477
Data columns (total 5 columns):
 #   Column        Non-Null Count   Dtype 
---  ------        --------------   ----- 
 0   user_id       294478 non-null  int64 
 1   timestamp     294478 non-null  object
 2   group         294478 non-null  object
 3   landing_page  294478 non-null  object
 4   converted     294478 non-null  int64 
dtypes: int64(2), object(3)
memory usage: 11.2+ MB
```

Sampling

(Part E) (2 pts) Now that our DataFrame is nice and clean, we can proceed and randomly sample `n=4720` entries for each of the groups. Use the pandas' `DataFrame.sample()` method to do this:

In [9]:
```
dfControl = dfData.loc[dfData.group == "control"].sample(4720)
dfTreatment = dfData.loc[dfData.group == "treatment"].sample(4720)
```

Visualizing the results

The first thing we can do is to calculate some **basic statistics** to get an idea of what our samples look like.

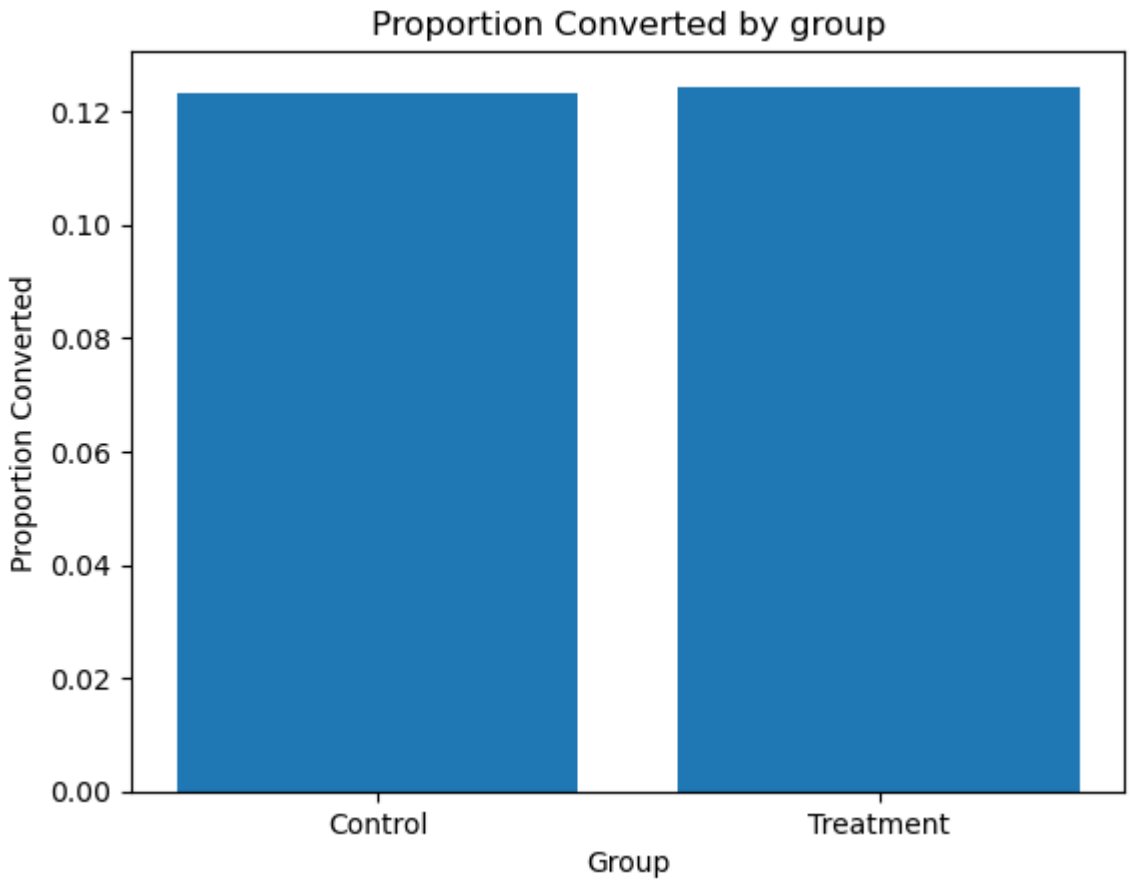
(Part F) (1 pt) Calculate the conversion rate for the control vs the treatment group in your sample:

In [10]:
```
controlConverted = dfControl[dfControl.converted == 1].count()
treatConverted = dfTreatment[dfTreatment.converted == 1].count()
c1 = dfControl['converted'].value_counts()[1]
c2 = dfTreatment['converted'].value_counts()[1]
p1 = c1/4720
p2 = c2/4720
print("Proportion of converted amongst Control group = {}".format(p1))
print("Proportion of converted amongst Treatment group = {}".format(p2))
```

```
Proportion of converted amongst Control group = 0.12309322033898305
Proportion of converted amongst Treatment group = 0.12436440677966101
```

(Part G) (1 pt) Create a bar plot (using `plt.bar`) to compare the conversion rate for the control vs the treatment groups.

In [11]:
```
x = ['Control', 'Treatment']
y = [p1, p2]
#print(c1,c2)
plt.bar(x, y)
plt.title('Proportion Converted by group')
plt.xlabel('Group')
plt.ylabel('Proportion Converted')
plt.show()
```



Step 4. Testing the hypothesis

Is the difference in conversion rates between the control and treatment groups statistically significant?

The last step of our analysis is testing our hypothesis.

Again, Python makes all the calculations very easy.

(Part H(i)) (2 pts) Conduct an appropriate hypothesis test using an appropriate built-in function from Python and output the p-value for your hypothesis test:

In [12]:
```
2*(1-stats.binom.cdf(c2,4720,c1)+stats.binom.pmf(c2, 4720,c1))
```

Out[12]:
```
nan
```

(Part H(ii)) (1 pts) Explain any assumptions that are necessary for the hypothesis test you used and why/how you know your data satisfies those assumption(s).

(Part I) (2 pts) Use a built-in Python function to get a 95% confidence interval for the difference in conversion rates between the new page and the old page. Again, list any assumptions about the data in using the built-in function you chose.

In [ ]:
```

```

Step 5. Drawing conclusions

(3 pts) What do you conclude based on the results of Part H and Part I above? Comment on both what the p-value AND the confidence interval tells you. What do you recommend to your team - is the new design considered a success (i.e. should it be implemented or not)?

Back to top

(10 pts) Problem 3 - Alcohol Content of Red vs White Wine

Load the data in **`winequality-red.csv`** and **`winequality-white.csv`** into Pandas DataFrames and take a look at the data. A description of this dataset can be found on Berkeley's Machine Learning Repository.

Part A (2 pts) The characteristic that we'll be interested in is the wine's alcohol content. Create side-by-side boxplots to compare the wine alcohol content for red wine vs white wine based on this sample data. Include the mean alcohol content for each type of wine in their corresponding boxplot by setting `showmeans=True` .

In [ ]:

Visually it's not clear whether or not the difference is actually significant. We'd like to test whether there is a statistically significant difference in mean alcohol content between red and white wine.

Part C (1 pt) State the null and alternative hypotheses:

Part C (2 pts) Conduct an appropriate hypothesis test at a 5% significance level using an appropriate built-in function from Python and output the p-value for your hypothesis test:

In [ ]:

Part D(1 pt) What is the conclusion of your test based on your p-value in part C?

In [ ]:

Part E (1 pt) Explain any assumptions that are necessary for the hypothesis test you chose and why/how you know your data satisfies those assumption(s).

Part F (2 pts) What is the 95% confidence interval for the difference between mean alcohol content in red vs white wine (i.e. for $\mu_{red} - \mu_{white}$)? (You can use built-in functions to calculate this, but again explain any assumptions that are necessary to use the built-in function you chose).

In [ ]:

Part G(1 pt) Intepret/explain what the confidence interval values mean in the context of this problem.

In [ ]:

Back to top

(15 pts) Problem 4- Interpreting Confidence Intervals

The following two parts are NOT Related:

Part A (Theory):

Suppose you obtain a 95% confidence interval of $[8.81, 9.82]$ for the mean of some unknown distribution. For each of the following, explain why or why not the situation described is correct, given the technical definition of a 95% confidence interval we went over in class.

(i) (2 pts) If you had no other evidence regarding the true mean of the distribution, you could say there is a 95% chance that its true mean falls between 8.81 and 9.82.

(ii) (2 pts) If a class of 100 students all construct 95% confidence intervals for the mean of this particular distribution, then we expect about 95 of their CIs to contain the true mean, and about 5 of them to miss the true mean.

(iii) (2 pts) There is a 95% probability that any given random variable sampled from this distribution will be between 8.81 and 9.82.

i. This is incorrect, basically if we ran this test over and over and got a confidence interval, 95% of them would cover the true mean. This single confidence interval doesn't mean we can say thing for certainty. ii. This is correct, that's what the idea of a confidence interval is. iii. No, we're choosing the mean of a sample. not just a single random variable.

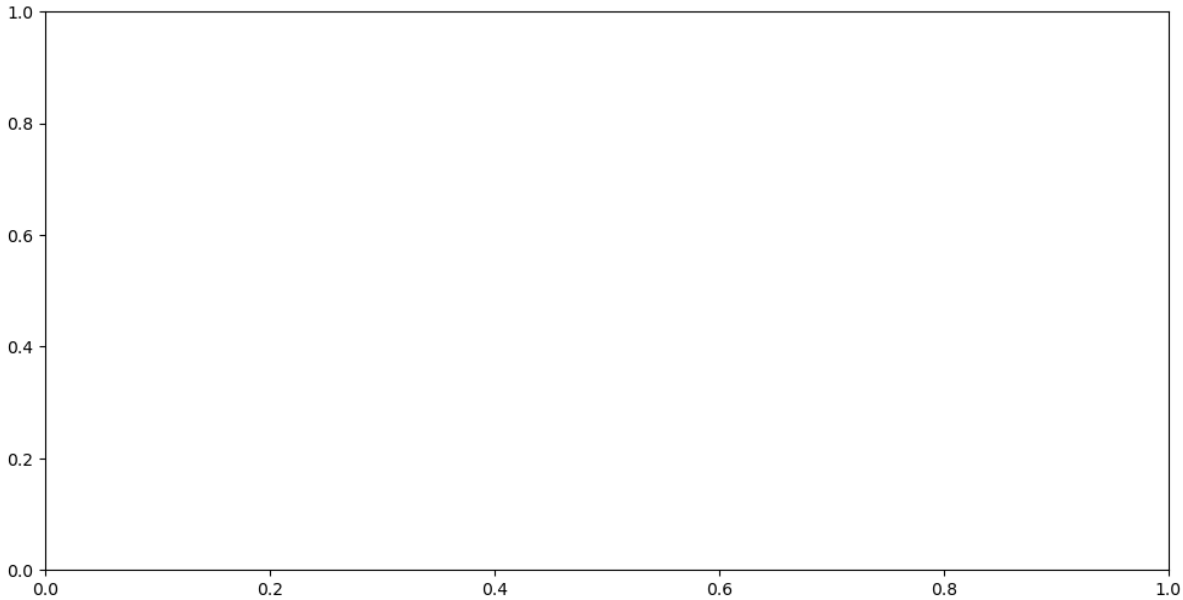
Part B: Simulation:

The so-called Chi-squared distribution is important in several statistical tests. It's also fairly asymmetric, and thus interesting for investigating confidence intervals. It is implemented in scipy.stats as chi2.

In [13]:
```
from scipy.stats import chi2
```

(i) (2 pts): Complete the following code cell to plot a histogram of 10,000 random samples from the Chi-squared distribution with degrees of freedom $d = 3$.

In [14]:
```
d = 3
x = chi2.rvs(d, size=10000)
fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(12,6))
# TODO
```



(ii)(1 pt): Look up the mean and variance of the Chi-squared distribution with degrees of freedom $d = 3$ and report them here (you will need these in the next step).

(iii) (5 pts): In this part you'll write a function to investigate the *coverage properties* of a confidence interval for the mean of the Chi-squared distribution. Complete the following function to randomly sample $m = 500$ sample means with sample size $n = 100$ for the Chi-squared distribution with 3 degrees of freedom. For each random sample, compute the 95% confidence interval for the mean. Your function should do two things:

1. Report the proportion of confidence intervals that successfully cover the true mean of the distribution
2. Make a plot of 50 randomly selected confidence intervals. Overlay the intervals on the line $y = \mu$ where μ is the true mean of the distribution. Color confidence intervals black if they cover the true mean, and red if they don't.

In [15]:
```
def confidence_intervals(m=500, n=100):
    # TODO
```

```
  Cell In[15], line 2
    # TODO
          ^
SyntaxError: incomplete input
```

(iv)(1 pt): Does the proportion of confidence intervals that cover the true mean of the distribution agree with the theory described in class? Explain.

In [ ]: