Hypothesis Testing in Python
by datacamp

A/B testing
involves splitting users into control and treatment groups

Hypothesizing about the mean
going to use StackOverflow survey data with users who identify as Data Scientists
a hypothesis
mean annual compensation of the population of data scientists is $110,000
point estimate, another name for summary statistic (sample statistic):

```
mean_comp_samp = stack_overflow['coverted_comp'].mean()
```

output is 119574
**different, but is this a 'meaningful' difference?
to determine this:
generate a boostrap distribution of sample means
we do this by resampling the dataset, calculating the sample mean for that resample, then repeating those steps to create a list of sample means

```
import numpy as np
so_boo_distn = [ ]
for i in range(5000):
        so_boot_distn.append(np.mean(stack_overflow.sample(frac=1, replace=True)
['converted_comp']))
```

sample
calculate mean
append list
repeat

Standard error

```
std_error = np.std(so_boot_distn, ddof=1)
```

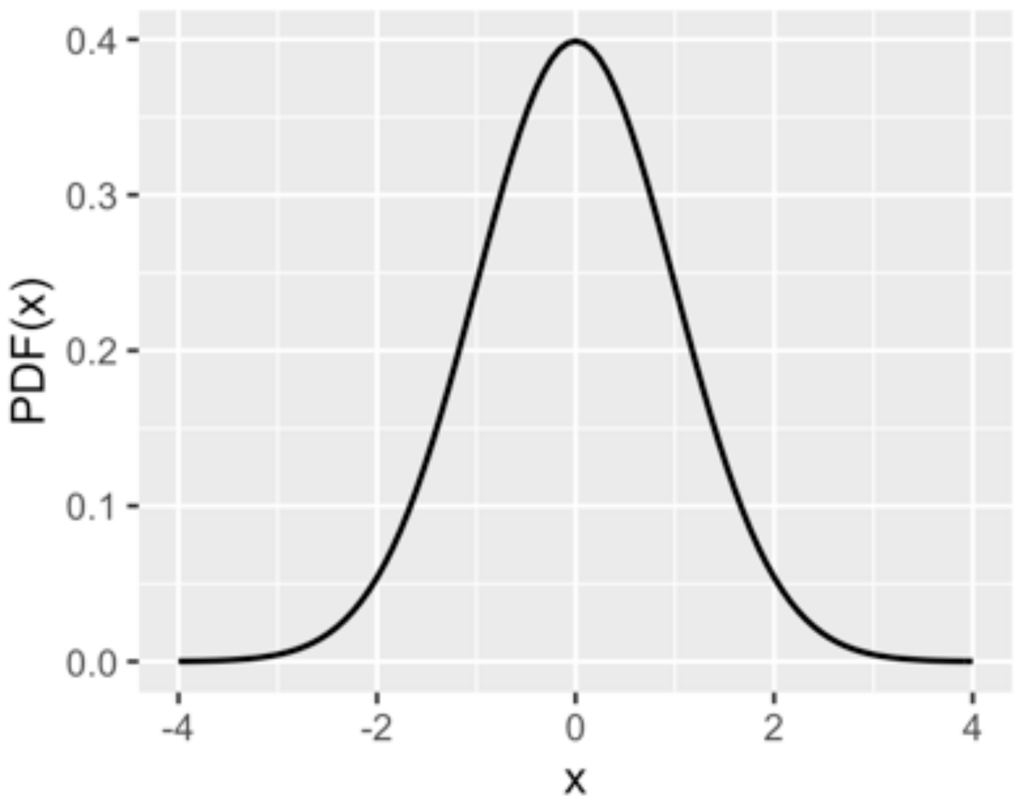
a common way of standardizing values:
standardized value = (value-mean)/standard deviation
for hypothesis testing we use a variation (called z-score)
z score = (sample statistic - hypothesized parameter value) / standard error
example (continuation from above)

```
mean_comp_hyp = 110000
z score = (mean_comp_samp - mean_comp_hyp) / std_error
```

our example output > 1.707
**Is this high or low?
**the goal of hypothesis testing is determining whether a sample statistic is close

to or far away from an expected (or ‘hypothesized’) values

Standard normal (z) distribution
mean = 0
std = 1



example
```
# Hypothesize that the proportion is 6%
late_prop_hyp = 0.06

# Calculate the standard error
std_error = np.std(late_shipments_boot_distn)

# Find z-score of late_prop_samp
z_score = (late_prop_samp - late_prop_hyp) / std_error

# Print z_score
print(z_score)
```

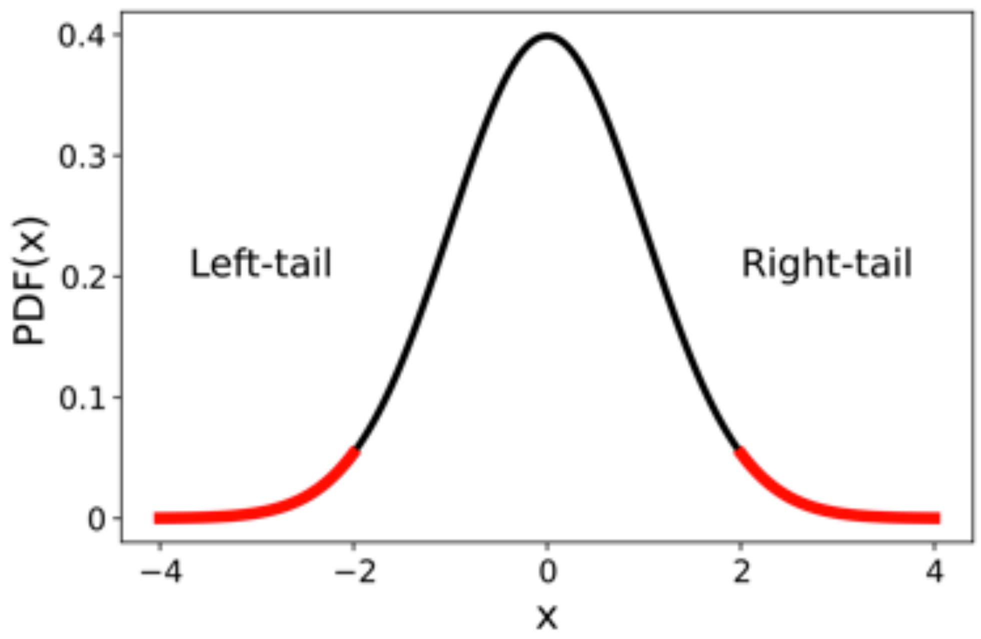
p-values
Hypothesis can be compared to a criminial trial
two possible outcomes > defendant committed the crime or they did not

two possible verdicts > guilty or not guilty
*initially the defendant is assumed to be not guilty
prosecution must present evidence ‘beyond reasonable doubt’ for a guilty verdict

Definitions
hypothesis is a statement about an unknown population parameter
**we don’t know the true value of this population parameter
we can only make inferences
hypothesis test is a test of two competiing hypotheses
null hypothesis (Ho) is the existing idea
called ‘H naught’ British for zero
alternative hypothesis (Ha) is the new ‘challenger’ idea of the researcher
called ‘H-A’

Example problem
Ho: the proportion of data scientists starting programming as children is 35%
Ha: the proportion of data scientists starting programming as children is greater than 35%
either Ha or Hnaught is true (not both)
initially Hnaught is assumed to be true
this only changes if the sample provides enough evidence to reject it
rather than saying we accept the alternative hypothesis, it is convention to say:
reject the null hypothesis
or fail to reject the null hypothesis
***if the evidence from the sample is ‘significant’ that Ha is true, reject Hnaught, else choose Hnaught
’significance level’ is the ‘beyond a reasonable doubt’ for hypothesis testing

One-tailed and two-tailed tests
hypothesis tests check if the sample statistics lie in the tails of the null distribution



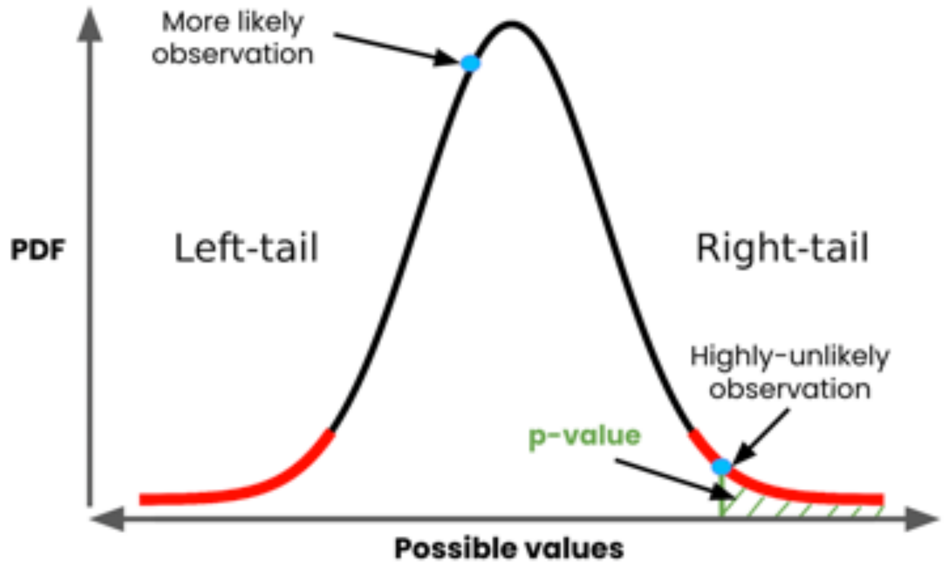
the tails of the distribution are the left and right edges of its probability density function (PDF)
hypothesis tests determine whether sample statistics lie in the tails of the null distribution
which is the distribution of the statistic if the null hypothesis was true
**three types of tests
phrasing of the alternative hypothesis determines which type

1. alternative different from null > two-tailed
    1. ie we are looking for any difference, any extreme values, greater or less than
2. alternative greater than null or exceeds null > right-tailed
3. alternative less than null or fewer than null > left-tailed

our example
Ha: proportion of data scientists starting programming as children is 'greater' than 35%
**we need a right-tailed test

p-values
probability of obtaining a result, 'assuming' the null hypothesis is true
measures the strength of support for the null hypothesis
p-values are probabilities so they are always between 0 and 1
large p-values mean our statistic is producing a result that is likely not in a tail of our null distribution
large p-values state chance could be a good explanation for the result
small p-values mean our statistic is producing a result likely in the tail of our null distribution

large p-value > large support “for” Hnaught
large p-value > statistic likely not in the tail of the null distribution
small p-value > strong evidence “againgst” Hnaught
small p-value > statistic likely in the tail of the null distribution
‘small’ means ‘close to zero’
‘p’ in p-value is for probability



Another z-score calculation example
for our example of null hypothesis of data scientists first programming as children as 35%
prop_child_samp = (stack_overflow[‘age_first_code_cut’] == ‘child’).mean()
prop_child_hyp = 0.35
std_error = np.std(first_code_boot_distn, ddof=1)
z_score = (prop_child_samp - prop_child_hyp) / std_error

**left-tailed test use norm.cdf()
**right-tailed test use 1 - norm.cdf()
#cdf is the cummulative distribution function
#we pass the z-score to the standard normal CDF
#we use 1 - norm.cdf() because our Ha is greater than 35%
#since our Ha is using ‘greater’, we use a right-tailed test
**think left for 0.025 on the bell curve and right for 0.975 on the bell curve
from scipy.stats import norm
1 - norm.cdf(z_score, loc=0, scale=1)

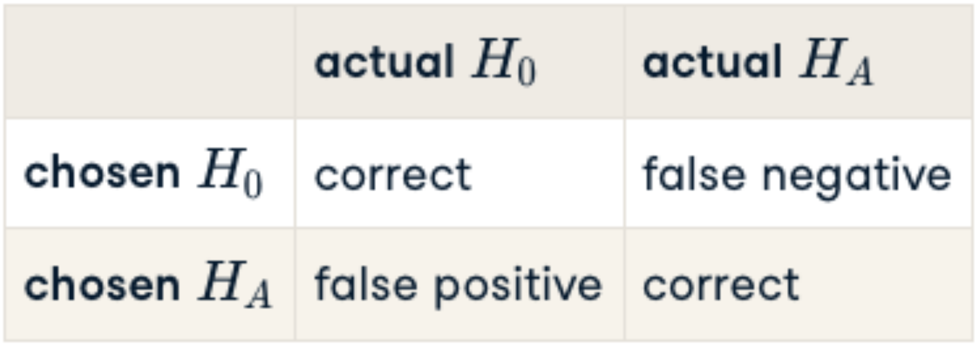
p-value recap
p-values quantify evidence for the null hypothesis
large p-values > fail to reject null hypothesis
small p-value > reject null hypothesis

What defines the cutoff between a small p-value and a large one?
Significance level determines the threshold
commonly 1, 5, or 10%, with 5% being the most common
this level depends on the field and dataset
significance level of a hypothesis test is labeled with the alpha symbol
significance level is this point for ‘beyond a reasonable doubt’
**if p < alpha, reject Hnaught, else fail to reject Hnaught
**alpha should be set prior to conducting the hypothesis test

Again
Calculating the p-value
alpha = 0.05
prop_child_samp = (stack_overflow[“age_first_code_cut’] == ‘child’).mean()
prop_child_hyp = 0.35
#remember standard error is obtained from the bootstrap distribution
std_error = np.std(first_code_boo_distn, ddof=1)
z_score = (prop_child_samp - prop_child_hyp) / std_error
#once again our Ha is ‘greater’ than, so we are checking a right-tail test
#1 - norm.cdf() for right-tail testx
p_value = 1 - norm.cdf(z-score, loc=0, scale=1)

Confidence intervals
to get a sense of the potential values of the population parameter
CI = 1 - alpha
p value of 0.05 gives a CI of 95%
again get CIs with np. quantiles
lower = np.quantile(first_code_boot_distn, 0.025)
upper = np.quantile(first_code_boot_distn, 0.975)
print((lower, upper))
output > a range of plausible values for the population proportion of data scientists that programmed as children
here it is 0.37, 0.41

Types of errors



	actual H_0	actual H_A
chosen H_0	correct	false negative
chosen H_A	false positive	correct

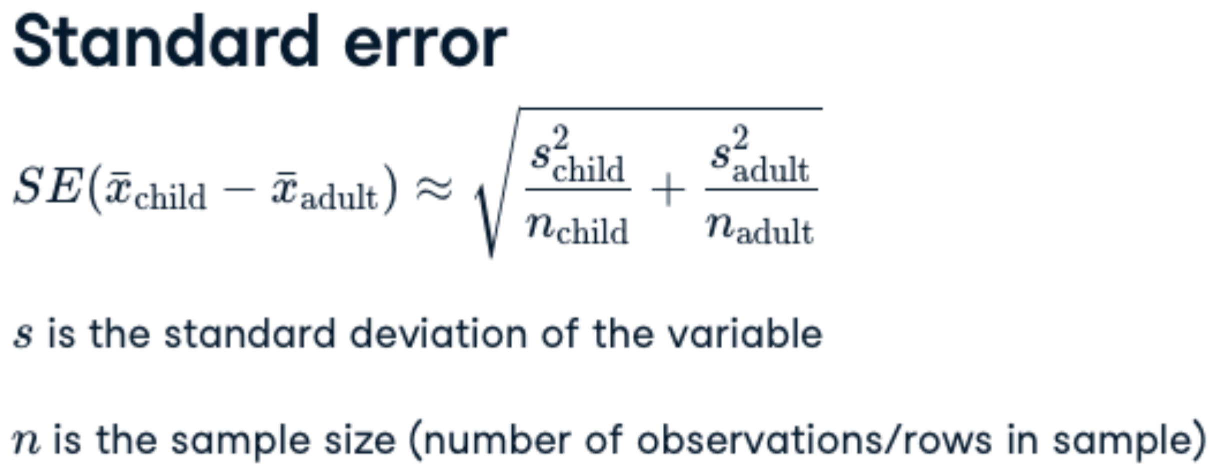
**False positives are Type I errors
**False negatives are Type II errors
'possible' errors in our example
if p<=alpha, we reject Hnaught:
a false positive (Type I) error: data scientists didn't start coding as children at a higher rate
if p>alpha, we fail to reject Hnaught:
a false negative (Type II) error: data scientists started coding as children at a higher rate

Performing t-tests
z-score is a test statistic for a 'single' variable
now we will compare sample statistics across groups in a variable
our question - are users who first programmed as a child better compensated than those that started as adults?
we will assess using numerical variable 'converted_comp' and categorical variable with vals (child and adult) in 'age_first_code_cut'
Hypotheses
Hnaught: The mean compensation (in USD) is the same for those that coded first as a child and those that coded first as an adult
Hnaught: mu with subscript child = mu with subscript adult
or
Hnaught: mu subscript child - mu subscript adult = 0
Ha: The mean compensation (in USD) is greater for those that coded first as a child compared to those that coded first as an adult
Ha: mu with subscript child > mu with subscrip adult
or
Ha: mu subscript child - mu subscript adult > 0

Calculating groupwise summary statistics
to calculate summary statistics for each group, groupby categorical variable then compute on the numerical variable
stack_overflow.groupby('age_first_code_cut')['converted_comp'].mean()
output > adult 111000 and child 132000

is the difference statistically significant or is it just sampling variability?

Test statistics
remember sample mean estimates the population mean
xbar denotes a sample mean
xbar subscript child - xbar subscript adult = a test statistic
**the difference between these two sample means is the test statistic for the hypothesis test
z-scores are a type of standardized test statistic
again
z-score = (sample stat - population parameter) / standard error
t = (difference in sample stats - difference in population parameters) / standard error
our example
t = ((xbar subscript child - xbar subscript adult) - (mu subscript child - mu subscript adult)) / SE(xbar subscript child - xbar subscript adult)
bootstrapping tends to be a good option to get the standard error
an easier way is to approximate it
SE(xbar subscript child - xbar subscript adult) approx equals sqrt((s subscript child squared / n subscript child) + (s subscript adult squared / n subscript adult)
where s is the standard deviation of the variable and
n is the sample size (number of observations/rows in sample)



Standard error

$$SE(\bar{x}_{\text{child}} - \bar{x}_{\text{adult}}) \approx \sqrt{\frac{s_{\text{child}}^2}{n_{\text{child}}} + \frac{s_{\text{adult}}^2}{n_{\text{adult}}}}$$

s is the standard deviation of the variable

n is the sample size (number of observations/rows in sample)

Assuming the null hypothesis is true
we can assume that the population means are equal
so their difference is zero
this removes the population term from the numerator when calculating the test statistic

$$t = \frac{(\bar{x}_{\text{child}} - \bar{x}_{\text{adult}}) - (\mu_{\text{child}} - \mu_{\text{adult}})}{SE(\bar{x}_{\text{child}} - \bar{x}_{\text{adult}})}$$

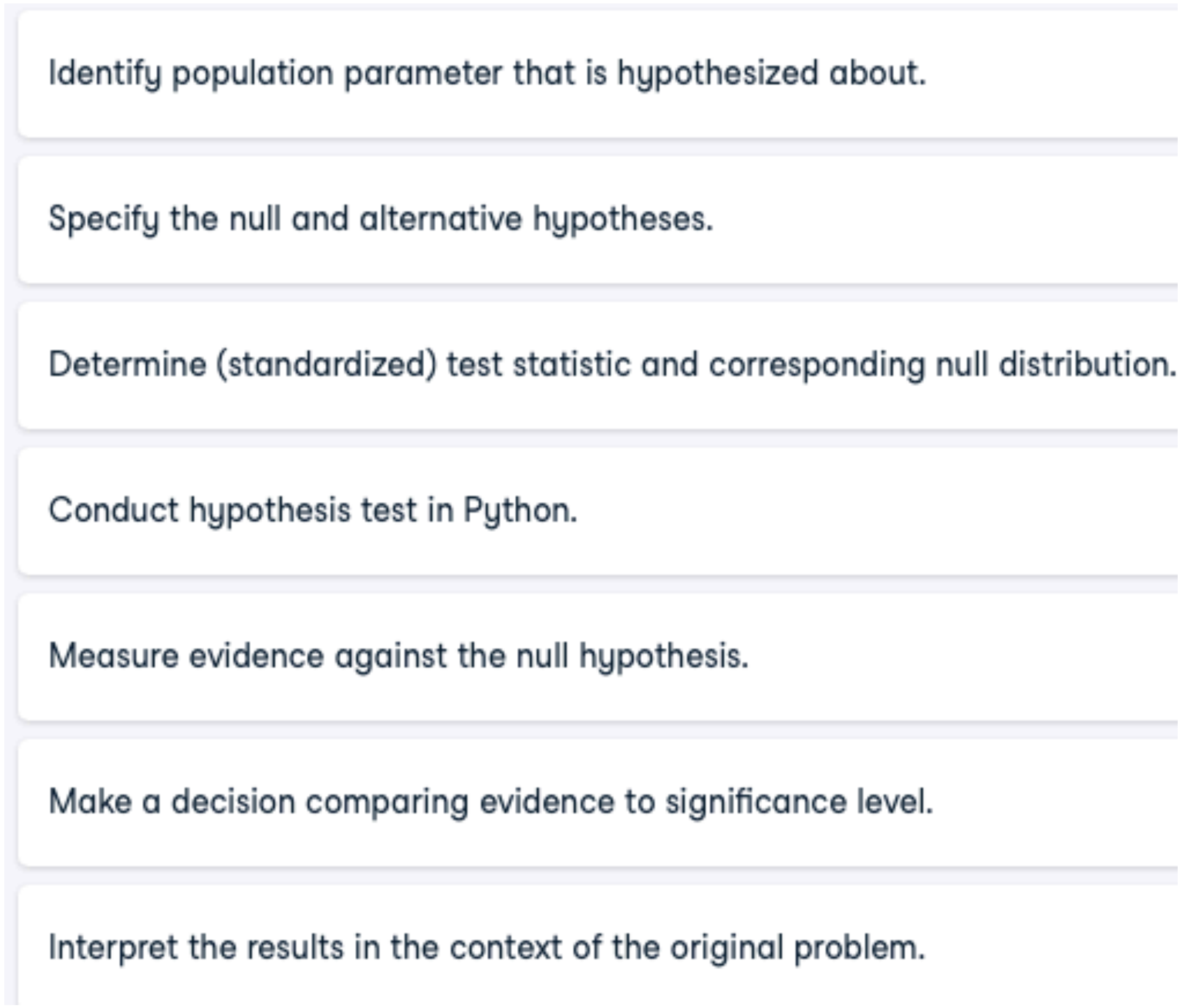
$$H_0\colon \mu_{\text{child}} - \mu_{\text{adult}} = 0 \quad \rightarrow \quad t = \frac{(\bar{x}_{\text{child}} - \bar{x}_{\text{adult}})}{SE(\bar{x}_{\text{child}} - \bar{x}_{\text{adult}})}$$

this also gives us a way of calculating the test statistic only using calculations on the sample dataset

$$t = \frac{(\bar{x}_{\text{child}} - \bar{x}_{\text{adult}})}{\sqrt{\frac{s_{\text{child}}^2}{n_{\text{child}}} + \frac{s_{\text{adult}}^2}{n_{\text{adult}}}}}$$

Calculations assuming the null hypothesis is true
need the mean, std, and number of observations for each group to fill in the formula for t
xbar = stack_overflow.groupby(‘age_first_code_cut’)[‘converted_comp’].mean()
s = stack_overflow.groupby(‘age_first_code_cut’)[‘converted_comp’].std()
n = stack_overfllow.groupby(‘age_first_code_cut’)[‘converted_comp’].count()
numerator = xbar_child - xbar_adult
denominator = np.sqrt(s_child **2 / n_child + s_adult ** 2 / n_adult)
#denominator is like a weighted hypotenuse
t_stat = numerator / denominator

Hypothesis testing workflow


Identify population parameter that is hypothesized about.

Specify the null and alternative hypotheses.

Determine (standardized) test statistic and corresponding null distribution.

Conduct hypothesis test in Python.

Measure evidence against the null hypothesis.

Make a decision comparing evidence to significance level.

Interpret the results in the context of the original problem.

Hypothesis testing is a statistical method used to make inferences and draw conclusions about a population based on a sample of data. It allows researchers and analysts to assess whether there is enough evidence to support or reject a specific claim or hypothesis about a population parameter.

The typical process of hypothesis testing involves the following steps:

1. **Formulate the Hypotheses:** The first step is to define two competing hypotheses: the null hypothesis (H0) and the alternative hypothesis (H1). The null hypothesis represents the default assumption or the status quo, often stating that there is no significant effect, relationship, or difference between variables. The alternative hypothesis represents the claim or theory that the researcher wants to support or find evidence for.

2. **Select a Test Statistic:** Based on the data and the nature of the hypotheses, a suitable test statistic is chosen. The test statistic is a numerical value calculated

from the sample data and provides information about the degree of support for the hypotheses.

3. **Set the Significance Level:** The significance level (often denoted as α) determines the threshold for deciding whether to reject the null hypothesis. It represents the probability of making a Type I error, which is the incorrect rejection of a true null hypothesis.

4. **Compute the P-Value:** The p-value is a probability value that measures the likelihood of obtaining the observed data, or more extreme data, under the assumption that the null hypothesis is true. A small p-value (typically less than the significance level α) suggests that the observed data is unlikely to have occurred by chance alone, leading to the rejection of the null hypothesis.

5. **Make a Decision:** Based on the p-value and the significance level, the null hypothesis is either rejected or not rejected. If the p-value is less than α, there is strong evidence against the null hypothesis, and it is rejected in favor of the alternative hypothesis. If the p-value is greater than or equal to α, there is not enough evidence to reject the null hypothesis.

6. **Draw Conclusions:** Once a decision is made, conclusions are drawn about the population based on the results of the hypothesis test. The conclusion may provide evidence for the alternative hypothesis or suggest that there is insufficient evidence to support the alternative hypothesis.

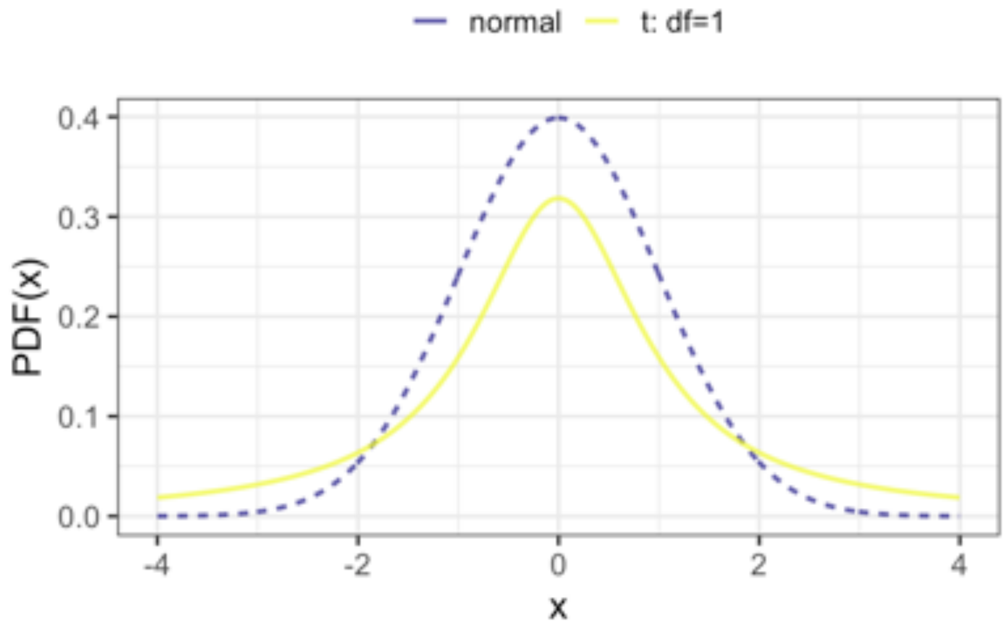
Remember that the outcome of hypothesis testing does not prove that one hypothesis is true or false; it only provides evidence to support or reject a hypothesis based on the available data and statistical methods.

Example

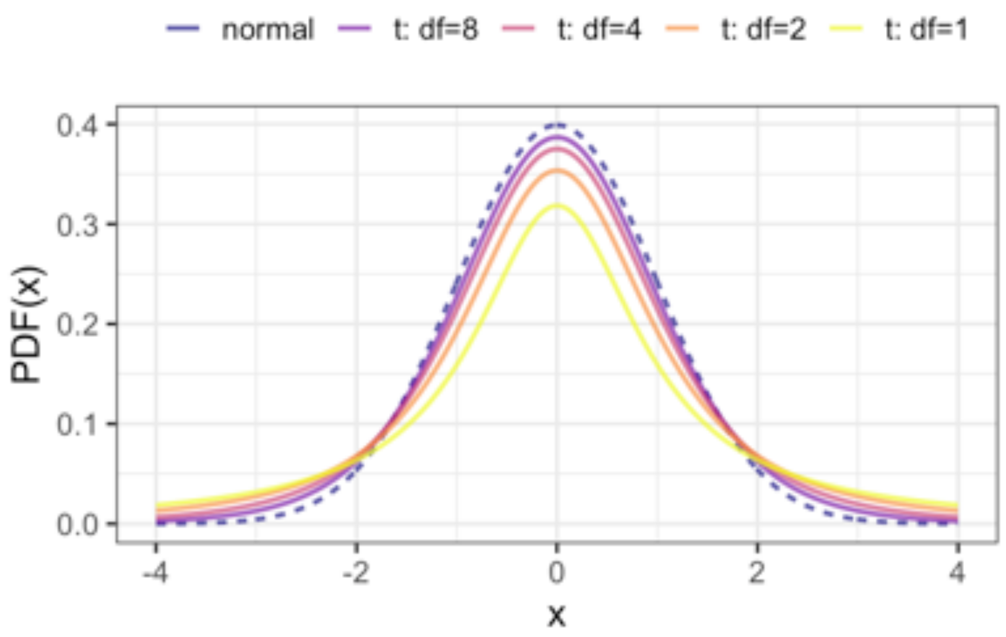
```
# Calculate the numerator of the test statistic
numerator = xbar_no - xbar_yes

# Calculate the denominator of the test statistic
denominator = np.sqrt(s_no ** 2 / n_no + s_yes ** 2 / n_yes)

# Calculate the test statistic
t_stat = numerator / denominator

# Print the test statistic
print(t_stat)
```

Calculating p-values from t-statistics

t-distributions
t statistic follows a t-distribution
test statistic is t or t statistic
t distributions have a parameter named degrees of freedom or df
**t-distributions for small degrees of freedom have fatter tails than normal distribution but otherwise they look similar



larger degrees of freedome brings the t-distribution closer to the normal distribution
**normal distributin is t-distribution with infinite df
**degrees of freedom are defined as the maximum number of logically independent values in the data sample



***Calculating and understanding degrees of freedom
example
dataset has 5 independent observations
we know that 4 of them have values 2, 6, 8, and 5
we also know sample mean is 5
because we know the sample mean, we can figure out the last value, which must be 4
**although all 5 observations in the sample were independent, we know an additional fact about the sample (its mean)
**so we only have 4 degrees of freedom

Another example
our previous example
df = n subscript child + n subscript adult - 2
here there is as many degrees of freedom as observations
minus 2 because we know 2 sample statistics (the mean for each group)

Calculating p-values from t-statistics
when using an approximation for the test statistic standard error by using sample information we add more uncertainty to a problem
the increased uncertainty makes it a t problem instead of a z problem
z-statistic: needed when using one sample statistic to estimate a population parameter
t-statistic: needed when using multiple sample statistics to estimate a population parameter
t-distribution allows for more uncertainty when using multiple estimates in a single statistic calculation

need to transform the test statistic using the t-distribution CDF instead of the normal distribution CDF
**use t-distribution CDF not normal CDF
from scipy.stats import t
1 - t.cdf(t_stat, df=df)
#see our ‘df’ above

Paired t-tests
example
Question > was the percentage of Republican candidate votes lower in 2008 than 2012?
Hnaught: mu2008 - mu2012 = 0
Ha: m2008 - mu2012 < 0
alpha = 0.05 significance level
**data is paired > each voter percentage refers to the same county
**means they are not independent
from two samples to one
for paired analyses, rather than considering the two variables separately, we can consider a single variable of the difference
sample_data = repub_votes_potus_08_12
sample_data[‘diff’] = sample_data[‘repub_percent_08’] - sample_data[‘repub_percent_12’]
sample_data[‘diff’].hist(bins=20)
calculate sample statistics of the difference
xbar_diff = sample_data[‘diff’].mean()
*revise the hypotheses
new hypotheses:
Hnaught: mu diff = 0
Ha: mu diff < 0
Test statistic

$$t = \frac{x_{\text{diff}} - \mu_{\text{diff}}}{\sqrt{\frac{s_{diff}^2}{n_{\text{diff}}}}}$$

$$df = n_{diff} - 1$$

we have one statistic so the number of degrees of freedom is the number of pairs minus one
xbar_diff = sample_data[‘diff’].mean()

mu diff = 0 (**remember mu diff is the null hypothesis and the assumption that it is true so it basically says there is no difference, ie 0)
n_diff = len(sample_data)
s_diff = sample_data[‘diff’].std()
t_stat = (xbar_diff-0) / np.sqrt(s_diff**2/n_diff)
degrees_of_freedom = n_diff - 1
from scipy.stats import t
p_value = t.cdf(t_stat, df=n_diff-1)

an easier way
import pingouin
pingouin.ttest(x=sample_data[‘diff’], y=0, alternative=‘less’)
first argument is x and represents the data for the first sample, in this case it is the Series of differences (can also be an array or DF)
y argument represents the data for the second sample in a two sample t-test, in this example we use y=0 to represent the difference value from the null hypothesis
**in a one-sample test y is set to None
‘alternative’ argument respresents the alternative hypothesis and can be specified as two-sided, less, or greater
two-sided for two-tailed, greater for right-tailed, and lesser for left-tailed

for paired data
pingouin.ttest(x=sample_data[‘repub_percent_08’], y=sample_data[‘repub_percent_12’], paired=True, alternative=‘less’)

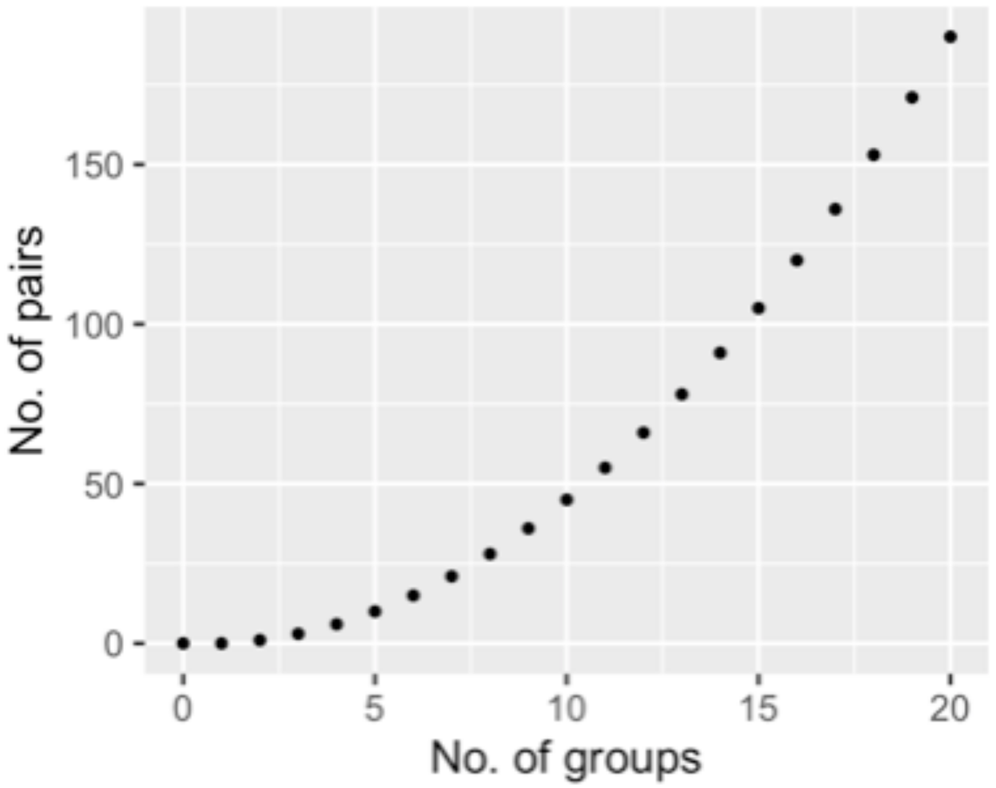
Anova tests
what is there is more than two groups?
Visualizing multiple distributions
Is mean annual compensation different for different levels of job satisfaction?
visualize distributions with box plots
import seaborn as sns
import matplotlib.pyplot as plt
sns.boxplot(x=‘converted_comp’, y=‘job_sat’, data=stack_overflow)
plt.show()
ANOVA (Analysis of variance)
- a test for differences between groups
- define significane level

example
alpha = 0.2 (not common, just for example to help explain the implications on comparing different numbers of groups later on)
pingouin.anova(data=stack_overflow, dv=‘converted_comp’, between=‘job_sat’)
‘dv’ argument represents the dependent variable
‘between’ argument represents the column of groups to calculate between

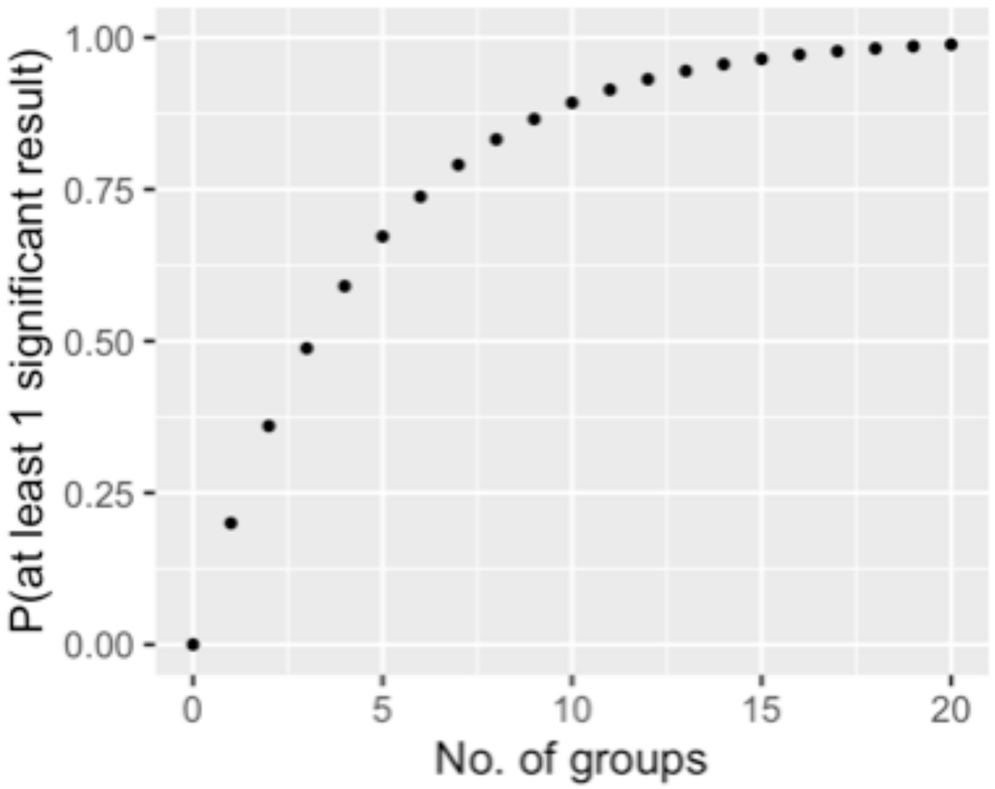
output:
p-unc represents the p-value
in this example it is less than 20% which means that at least two of the categories of job satisfaction have significant differences between their compensations levels,
**this doesn’t tell us which two
to identify which categories are different we use pairwise tests
we compare all five job satisfaction categories
testing on each pair in turn
in this example we have ten tests to perform

- $\mu_{\text{very dissatisfied}} \neq \mu_{\text{slightly dissatisfied}}$
- $\mu_{\text{very dissatisfied}} \neq \mu_{\text{neither}}$
- $\mu_{\text{very dissatisfied}} \neq \mu_{\text{slightly satisfied}}$
- $\mu_{\text{very dissatisfied}} \neq \mu_{\text{very satisfied}}$
- $\mu_{\text{slightly dissatisfied}} \neq \mu_{\text{neither}}$
- $\mu_{\text{slightly dissatisfied}} \neq \mu_{\text{slightly satisfied}}$
- $\mu_{\text{slightly dissatisfied}} \neq \mu_{\text{very satisfied}}$
- $\mu_{\text{neither}} \neq \mu_{\text{slightly satisfied}}$
- $\mu_{\text{neither}} \neq \mu_{\text{very satisfied}}$
- $\mu_{\text{slightly satisfied}} \neq \mu_{\text{very satisfied}}$

can run all ten of these tests in one go
pingouin.pairwise_tests(data=stack_overflow, dv=‘converted_comp’, between=‘job_sat’, padjust=‘none’)
output:
a DF that shows the pairs being compared as A column and B column
our output shows us a three p-unc (ie p-values) less than the alpha .2
*as the number of groups increase, the number of pairs and hypothesis tests increases quadratically

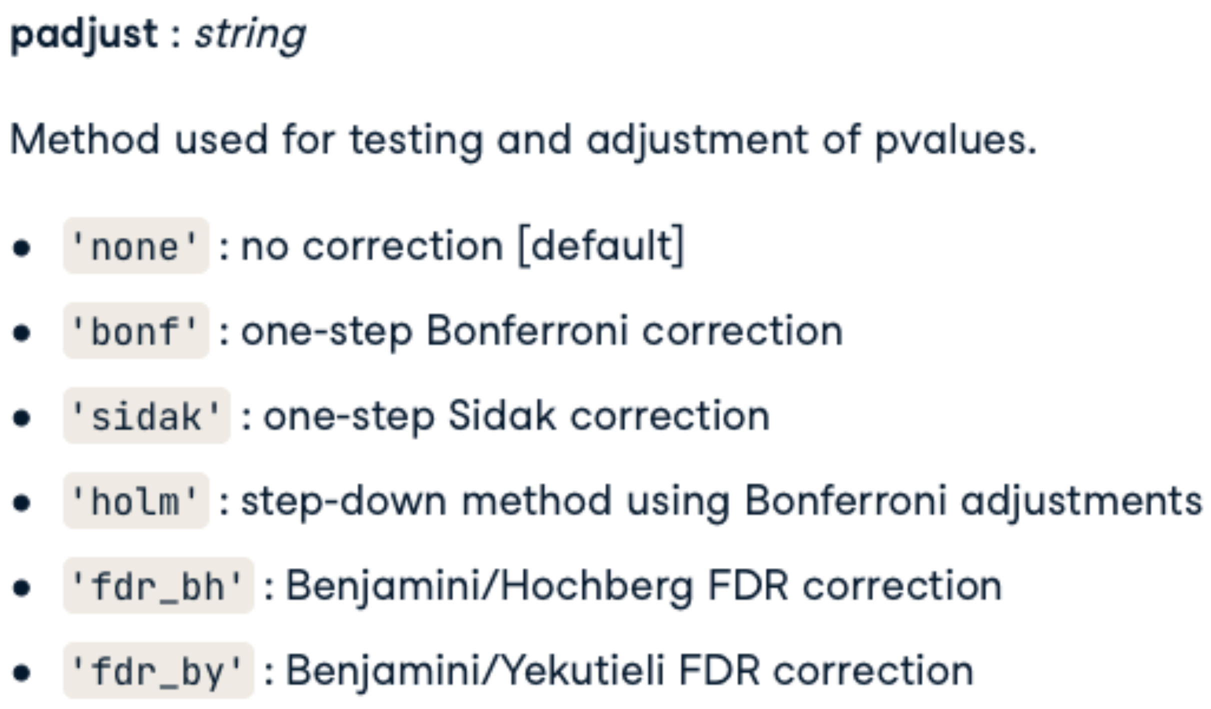


*the more tests we run, the higher the chance that at least one of them will give a false positive significant result
example our test with significance level 0.2 has a 0.2 chancee of a false positive
after ten tests the probability of a false positive is near 0.7 or 70%
If we wer to increase our groups to say 20 we'd have almost a guarantee .99 that we'll get at least one false positive



A solution for this?
one possible solution is to apply an adjustment to increase the p-values
this reduces the chance of getting a false positive
a common adjustment is called the Bonferroni correction
**we do this with the ‘padjust’ argument
in this case padjust=‘bonf’
**you now compare p-unc to the p-corr column
in our example now only two of the pairs appear to have significant differences

More methods:


padjust : *string*

Method used for testing and adjustment of pvalues.

- `'none'` : no correction [default]
- `'bonf'` : one-step Bonferroni correction
- `'sidak'` : one-step Sidak correction
- `'holm'` : step-down method using Bonferroni adjustments
- `'fdr_bh'` : Benjamini/Hochberg FDR correction
- `'fdr_by'` : Benjamini/Yekutieli FDR correction

One-sample proportion tests
Chapter 1 recap

- the hypothesis test measured whether or not an unknown population proportion was equal to some value
- we used bootstrapping on the sample to estimate the standard error of the sample statistic
- standard error was then used to calculata a standardized test statistic (z-score)
- z-score was used to get a p-value
- p-value was used to decide whether or not to reject the null hypothesis

Now we are going to calculate the test statistic without using the bootstrap distribution
why dot this?
a bootstrap distribution can be computationally intensive
Standardized test statistic for proportions
p is for population proportion (unknown population parameter)
phat is for sample proportion (sample statistic)
po is for hypothesized population proportion
here the test statistic is also a z-score

$$z = \frac{\hat{p} - \text{mean}(\hat{p})}{\text{SE}(\hat{p})} = \frac{\hat{p} - p}{\text{SE}(\hat{p})}$$

*the mean of a sampling distributin of sample means, denoted here by phat, is p, the population proportion
under the null hypothesis, the unknown proportion p is assumed to be the hypothesized population proportion p-zero
assuming Ho is true, p = po, so:

$$z = \frac{\hat{p} - p_0}{\text{SE}(\hat{p})}$$

$SE_{\hat{p}} = \sqrt{\frac{p_0 * (1 - p_0)}{n}} \rightarrow$ Under $H_0, SE_{\hat{p}}$ depends on hypothesized p_0 and sample size n

Assuming H_0 is true,

$$z = \frac{\hat{p} - p_0}{\sqrt{\frac{p_0 * (1 - p_0)}{n}}}$$

- Only uses sample information ($\hat{p}$ and n) and the hypothesized parameter (p_0)

Standard Error Review
In statistics, the standard error (SE) is a measure of the variability or precision of a sample statistic, such as the sample mean or sample proportion, in comparison to the true population parameter it estimates. It provides an indication of how much the sample statistic is expected to vary from one sample to another if multiple samples were taken from the same population.

The standard error is especially important when making inferences about the population based on sample data. It allows us to quantify the uncertainty or margin of error associated with our estimates.

For the sample mean (x̄), the formula for standard error is:

```
SE = s / √n
```

where:
- `s` is the sample standard deviation, which measures the spread of data points around the sample mean.
- `n` is the sample size, which represents the number of observations in the sample.

For the sample proportion (p̂) in a binomial distribution, the formula for standard error is:

```
SE = √(p̂ * (1 - p̂) / n)
```

where:
- `p̂` is the sample proportion (the number of successes divided by the sample size).
- `n` is the sample size.

Key points about standard error:

1. **Sample Size:** The standard error is inversely proportional to the square root of the sample size. As the sample size increases, the standard error decreases, indicating that larger samples tend to produce more precise estimates.

2. **Population Variability:** The standard error depends on the variability of the population. If the population has higher variability, the standard error will be larger, indicating less precision in the estimate.

3. **Confidence Intervals:** Standard error is used to calculate confidence intervals, which provide a range of values within which the true population parameter is likely to lie with a certain level of confidence.

4. **Hypothesis Testing:** In hypothesis testing, the standard error is used to calculate test statistics, such as t-statistics or z-scores, which are then used to assess the significance of the sample statistic and make decisions about the null hypothesis.

In summary, the standard error is a fundamental concept in statistics that quantifies the uncertainty associated with sample statistics. It is a critical component in inferential statistics, providing valuable information about the accuracy and precision of estimates made from sample data.

T-statistic and T-distribution Review

The t-statistic and the t-distribution are closely related concepts used in statistics, especially in hypothesis testing and confidence interval estimation. However, they have different meanings and roles in statistical analysis:

1. **T-Statistic:**
The t-statistic is a single value calculated from sample data that measures how many standard errors the sample mean is away from the hypothesized population mean (under the assumption that the null hypothesis is true). It is used primarily in hypothesis testing to assess whether there is a significant difference between the sample mean and a hypothesized population mean.

The formula for the t-statistic is typically:
```
t = (sample mean - hypothesized population mean) / (standard error of the sample mean)
```

The t-statistic is used in t-tests, which are statistical tests used to compare the means of two samples (two-sample t-test) or a sample mean against a known value (one-sample t-test).

2. **T-Distribution:**
The t-distribution is a probability distribution that is used when the sample size is small and the population standard deviation is unknown. It is similar in shape to the standard normal distribution (z-distribution), but its shape varies depending on the sample size (degrees of freedom, df).

The t-distribution is characterized by its degrees of freedom, which is the number of independent pieces of information used to estimate a parameter. For example, in a one-sample t-test, the degrees of freedom are equal to the sample size minus 1 (df = n - 1).

The t-distribution is used to calculate critical values for hypothesis testing or construct confidence intervals for population parameters when the population standard deviation is unknown.

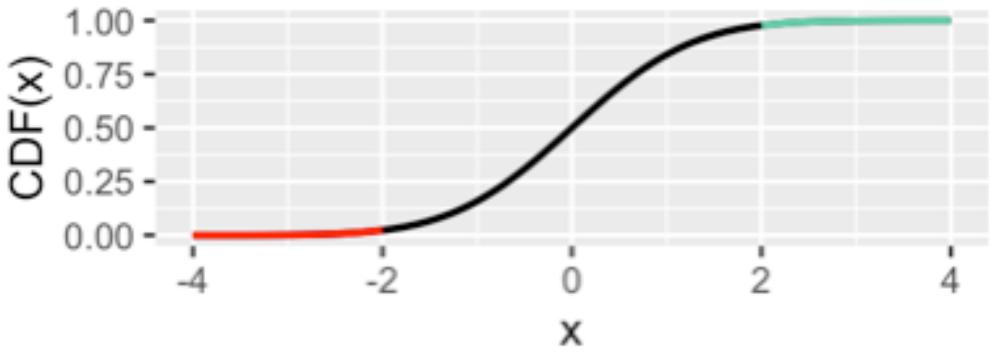
In summary, the t-statistic is a specific value calculated from sample data to assess the significance of a sample mean in comparison to a hypothesized population mean, whereas the t-distribution is a probability distribution used to determine critical values and construct confidence intervals for sample means when the sample size is small and the population standard deviation is unknown.

Both the t-statistic and the t-distribution play essential roles in statistical inference, especially when dealing with small samples or situations where the population standard deviation is not known.

Why z instead of t?
standard deviation of sample, s, is calculated from the sample mean, xbar
that means for the t-statistic that xbar is used in the numerator to estimate the population mean
and in the denominator to estimate the population standard deviation
**the dual usage increases the uncertainty in our estimate of the population parameter
**now don’t confuse t-statistic with t-distribution
remember t-distribution is essentially normal distribution with fatter tails becoming more like the normal distribution with each added on independent variable
we can use t-distributions to account for this extra uncertainty
t-distribution provides extra caution against mistakenly rejecting the null hypothesis
**for proportion, we only use phat in the numerator so we avoiding the problem with uncertainty so z-distribution is fine

One-sample proportion test example
StackOverflow age categories
Hnaught: Proportion of Stack Overflow users under thirty = 0.5
Ha: Proportion of Stack Overflow users under thity not equal 0.5
alpha = 0.01
stack_overflow[‘age_cat’].value_counts(normalize=True)
variables needed for the z score
p_hat = (stack_overflow[‘age_cat’] == ‘Under 30’).mean()
p_0 = 0.5 #this is according to the null hypothesis
#n is the number or rows in the dataset
n = len(stack_overflow
calculated z score
z = (p_hat - p_0) / sqrt(p_0 * (1 - p_0) / n)
numerator = p_hat - p_0
denominator = np.sqrt(p_0 * (1-p_0) / n)
z_score = numerator / denominator
calculate the p-value
from scipy.stats import norm
Left-tailed (‘less than’)
p_value = norm.cdf(z_score)
Right-tailed (‘greater than’)
p_value = 1 - norm.cdf(z_score)

Two-tailed ('not equal')
corresponds to the z-score and the other to its negative on the other side of the distribution
p_value = norm. cdf(-z_score) + 1 - norm.cdf(z_score)
*since the normal distribution PDF is symmetric, this simplifies to twice the right-tailed p-value since the z score is positive
p_value = 2 * (1 - norm.cdf(z_score))



Two-sample proportion tests
comparing two proportions
Hnaught: Proportion of hobbyist users is the same for those under thirty as those at least thirty
Hnaught: p>=30 - p<30 = 0
Ha: Proportion of hobbyist users is different for those under thirty to those at least thirty
Ha: p>=20 - p<30 not equal to 0
set alpha to 0.05
get the numbers for the z-score
p_hats = stack_overflow.groupby('age_cat')
['hobbyist'].value_counts(normalize=True)
#remember that the normalize argument set to True sends relative frequencies (proportions or percents) of unique values
#normalize set to False returns absolute counts of unique values
n = stack_overflow.groupby('age_cat')['hobbyist'].count()
#here we used count() instead of len() because we are focusing on rows where hobbyist is Yes
#remember count() is used to count occurrences of specific elements or non-missing (non-null) values, whereas len() is a general-purpose function to find the length or size of various data structures, including the total number or elements or characters
**to isolate the hobbyist propotions from p_hats, we us pandas' multiIndex subsetting

passing a tuple of the outer column and inner column values

```
p_hat_at_least_30 = p_hats[(‘At least 30’, ‘Yes’)]
p_hat_under_30 = p_hats[(‘Under 30’, ‘Yes’)]
print(p_hat_at_least_30, p_hat_under_30)
```

additional simpler pandas subsetting to get the number of observations

```
n_at_least_30 = n[‘At least 30’]
n_under_30 = n[‘Under 30’]
print(n_at_least_30, n_under_30)
```

calculate the z-score

- z-score equation for a *proportion test*:

$$z = \frac{(\hat{p}_{\geq 30} - \hat{p}_{<30}) - 0}{\text{SE}(\hat{p}_{\geq 30} - \hat{p}_{<30})}$$

- Standard error equation:

$$\text{SE}(\hat{p}_{\geq 30} - \hat{p}_{<30}) = \sqrt{\frac{\hat{p} \times (1 - \hat{p})}{n_{\geq 30}} + \frac{\hat{p} \times (1 - \hat{p})}{n_{<30}}}$$

- $\hat{p}$ → weighted mean of $\hat{p}_{\geq 30}$ and $\hat{p}_{<30}$

$$\hat{p} = \frac{n_{\geq 30} \times \hat{p}_{\geq 30} + n_{<30} \times \hat{p}_{<30}}{n_{\geq 30} + n_{<30}}$$

- Only require $\hat{p}_{\geq 30}, \hat{p}_{<30}, n_{\geq 30}, n_{<30}$ from the sample to calculate the z-score

```
p_hat = (n_at_least_30 * p_hat_at_least_30 + n_under_30 * p_hat_under_30) / (n_at_least_30 + n_under_30)
std_error = np.sqrt(p_hat * (1-p_hat) / n_at_least_30 + p_hat * (1-p_hat) / n_under_30)
z_score = (p_hat_at_least_30 - p_hat_under_30) / std_error
print(z_score)
```

Easier way
proportion tests using proportions_ztest() from statsmodels
function requires two objects as NumPy arrays

```
stack_overflow.groupby(‘age_cat’)[‘hobbyist’].value_counts
n_hobbyists = np.array([812, 1021])
n_rows = np.array([812 + 238, 1021 + 190])
from statsmodels.stats.proportion import proportions_ztest
z_score, p_value = proportions_ztest(count=n_hobbyists, nobs=n_rows, alternative=‘two-sided’)
#testing for a difference, so we specify the alternative argument to two-sided
```

```
# Calculate the pooled estimate of the population proportion
p_hat = (p_hats["reasonable"] * ns["reasonable"] + p_hats["expensive"] *
ns["expensive"]) / (ns["reasonable"] + ns["expensive"])

# Calculate p_hat one minus p_hat
p_hat_times_not_p_hat = p_hat * (1 - p_hat)

# Divide this by each of the sample sizes and then sum
p_hat_times_not_p_hat_over_ns = p_hat_times_not_p_hat / ns["expensive"] +
p_hat_times_not_p_hat / ns["reasonable"]

# Calculate the standard error
std_error = np.sqrt(p_hat_times_not_p_hat_over_ns)

# Calculate the z-score
z_score = (p_hats["expensive"] - p_hats["reasonable"]) / std_error

# Calculate the p-value from the z-score
p_value = 1 - norm.cdf(z_score)

# Print p_value
print(p_value)
```

Chi-square test of independence
Anova extends t-tests to more than two groups
Chi-square tests of independence extend proportion tests to more than two groups

Statistical independence is the proportion of successess in the response variable is the same across all categories of the explanatory variable
in the previous hypothesis test, the proportion test had a positive result
the small p-value suggested that there was evidence that hte hobbyis and age category are associated
*if the proportion of hobbyists was the same for each age category, then the variables would be considered statistically independent

Test for independence of variables

```
import pingouin
expected, observed, stats = pingouin.chi2_independence(data=stack_overflow,
x='hobbyist', y='age_cat', correction=False)
print(stats)
```

the 'correction' argument specifies whether or not to apply Yate's continuity correction

Yate’s continuity correction is a fudge factor for when the sample size is very small and the degrees of freedom is one
*our example technically does not need it since each group has over one hundred observations
output > three different pandas dataframes
chi2 value is the squared result of our z-score

$$\chi^2 \text{ statistic} = 17.839570 = (-4.223691463320559)^2 = (z\text{-score})^2$$

Another example
use same Stack Overflow dataset
two variables - age category and job satisfaction
age category has two categories
job satisfaction has five categories
Declare hypotheses:
Hnaught: Age categories are independent of job satisfactions levels
Ha: Age categores are not independent of job satisfaction levels
Set alpha:
alpha = 0.1
Choose test statistic:
chi-square
*it quantifies how far away the observed results are from the expected values if independence was true
asks - Assuming independence, how far away are the observed results from the expected values?
Exploratory visualization:
we choose to use a proportional stacked bar plot

```
#calculate proportion in each age group
props = stack_overflow.groupby(‘job_sat’)
[‘age_cat’].value_counts(normalize=True)
#’unstack’ method converts this table into wide format
wide_props = props.unstack()
#’stacked’ argument produces a proportional stacked bar plot
wide_props.plot(kind=‘bar’, stacked=True)
```

**if the age category was independent of job satisfaction, then the split between age categories would be at the same height in each bar
in our example there is some variation
but now we need to see if that variation is significant
to do this we use the chi square independence test

again use pingouin package

expected, observed, stats = pingouin.chi2_independence(data=stack_overflow, x=‘job_sat’, y=‘age_cat’)
print(stats)
**we leave out the collection argument because our dof is 4
dof is calculated
(no. of response categories - 1) * (no. of explanatory categories -1)
our example
(2-1) * (5-1) = 4
our result
p-value is .23 so we cannot reject Hnaught
so we conclude that age categories are independent of job satisfaction

*if we swap the variables in our visualization, we see that the splits for each bar are in similar places
*if we run the chi-square test with the variables swapped, then the results are identical
because of this we phrase our questions as “are variables X and Y independent?” rather than “is variable X independent from variable Y?”

What about direction and tails?
chi-square test statistic is based on the square of observed and expected counts
square numbers are non-negative
this means that chi-square tests tend to be right-tailed tests
*left-tailed chi-square tests are used in statistical forensics to detect if a fit is suspiciousely good because the data was fabricated
*chi-square tests of variance can be two-tailed
both of these are very niche things, the majority of the time chi-square will be right-tailed

Chi-square goodness of fit tests
last example compared proportions in two categorical variables
this example will use another variant in chi-square test to compare a single categorical variable to a hypothesized distribution
How do you feel when you discover that you’ve already visited the top resource?
this is called a purple link
#get the counts of each group in the purple_link column
purple_link_counts = stack_overflow[‘purple_link’].value_counts()
#re-name left most column
#assign counts to n
#sort by purple_link, so the respones are in alphabetical order
purple_link_counts =
purple_link_counts.rename_axis(‘purple_link’).reset_index(name=’n’).sort_values(‘purple_link’)

Declare hypotheses:
Hnaught: the sample matches the hypothesized distribution
Ha: the sample does not match the hypothesized distribution

```
hypothesized = pd.Dataframe({‘purple_link’: [‘Amused’, ‘Annoyed’, ‘Hello, old friend’, ‘Indifferent’],
        ‘prop’: [1/6, 1/6, 1/2, 1/6]})
```

Test statistic:
chi-squared - measures how far the observed sample distribution of proportions is from the hypothesized distribution
Set alpha > 0.01
Hypothesized counts by category:

```
n_total = len(stack_overflow)
hypothesized[’n’] = hypothesized[‘prop’] * n_total
```

Visualize counts:
natural way to visualize the counts of a categorical variable is with a bar plot

```
#set horizontal axis to ‘purple_link’ and vertical axis to ’n’
#addl a label for a legend
plt.bar(purple_link_counts[‘purple_link’], purple_link_counts[’n’], color=‘red’, label=‘Observed’)
#do the same for hypothesized counts
plt.bar(hypothesized[‘purple_link’], hypothesized[’n’], alpha=0.5, color=‘blue’, label=‘Hypothesized’)
plt.legend()
plt.show()
```

Now run a hypothesis test to see if the difference is statistically significant
one-sample chi-square test also called a ‘goodness of fit test’
tests how well our hypothesized data fits the observed data

```
from scipy.stats import chisquare
chisquare(f_obs=purple_link_counts[’n’], f_exp=hypothesized[’n’])
#two required arguments to chisquare method > array-like object for the observed counts and one for the hypothesized counts
```

output > a p-value less than 0.01
we conclude the sample distribution of proportions is different from the hypothesized distribution

Assumptions in hypothesis testing
Needs Randomness

1. Assumption - samples are random subsets of larger populations
2. Consequence - sample is not representative of population
3. How to check this - understand how your data was collected and/or speak to a domain expert

Independence of observations

1. Assumption - each observation (row) in the dataset is independent
2. Consequence - increased chance of false negative/positive error
    1. example - paired t-test > not accounting for dependencies results in an increased chance of false negative/positive errors
3. How to check this - understand how our data was collected
    1. accounting for dependencies ideally needs to be discussed before data collection

Large sample size

1. Assumption - sample is big enough to mitigate uncertainty, so that the central limit theorem applies
    1. and the sample distribution can be assumed to be normally distributed
2. Consequence - wider confidence intervals; increased chance of false negative/positive errors
    1. smaller samples incur greater uncertainty, which may mean CLT does not apply
    2. increased certainty means wider CIs on the parameter we are trying to estimate
    3. if CLT does not apply, calculations on the sample and any conclusions could be nonsense
3. How to check this? - depends on the test

What constitutes a large sample size for t-test?
One sample t-test > general concensus is at least 30 observations
Two sample t-test > at least 30 observations in each sample
Anova > also at least 30 observations in each sample
Paired samples > at least 30 paires of observations across the samples
**key we cannot compensate for one minority group sample by making the majority group sample bigger
*important thing is that the null distribution appears normal
*this often is the case around 30

What constitutes a large sample size for proportion tests?
if probability of success is close to zero or one, then we need a bigger sample
one sample: considered big enough if contains at least 10 successes and 10 failures
number of successes in sample is greater than or equal to 10
n*phat >= 10
number of failures in sample is greater than or equal to 10
n*(1-phat) >= 10
where n = sample size and phat = proportion of successes in sample

two samples: need ten successes and ten failures from each sample
number of successes in each sample is greater than or equal to 10
n1*phat1 >= 10
n2*phat2 >= 10
number of failures in each sample is greater than or equal to 10
n1*(1-phat1) >= 10
n2*(1-phat2) >= 10

What constitutes large enought for chi-square test?
requires five successes and five failures in each group
ni*phati >= 5 for all values of i
ni*(1-phati) >= 5 for all values of i

Sanity check
visualize with a histogram
if the bootstrap distribution doesn’t look normal, assumptions likely aren’t valid
if we don’t see a bell-shaped normal curve, then one of the assumptions hasn’t been met
if this is the case, we should revisit the data collection process and see if any of the three assumptions of randomness, independence, and sample size do not hold

Non-parametric tests
What do we if the assumptions for the hypothesis test we’ve seen aren’t met?
the tests that we’ve seen so far z-test, t-test, and ANOVA are parametric tests
all are based on the assumption that the population is normally distributed
non-parametric tests do not make normal distribution assumptions or the sample size requirements

many non-parametric test use ‘ranks’ of the data
simple example
x = [1, 15, 3, 10, 6]
from scipy.stats import rankdata
rankdata(x)
ouput > ([1, 5, 2, 4, 3])

Non-parametric tests are more reliable than parametric tests for ‘small sample sizes’ and when data ‘isn’t normally distributed’
one of the first non-parametric procedures developed:
Wilcoxon-signed rank test
requires us to calculate the absolute differences in the pairs of data and then rank them
example
#calcualate differences

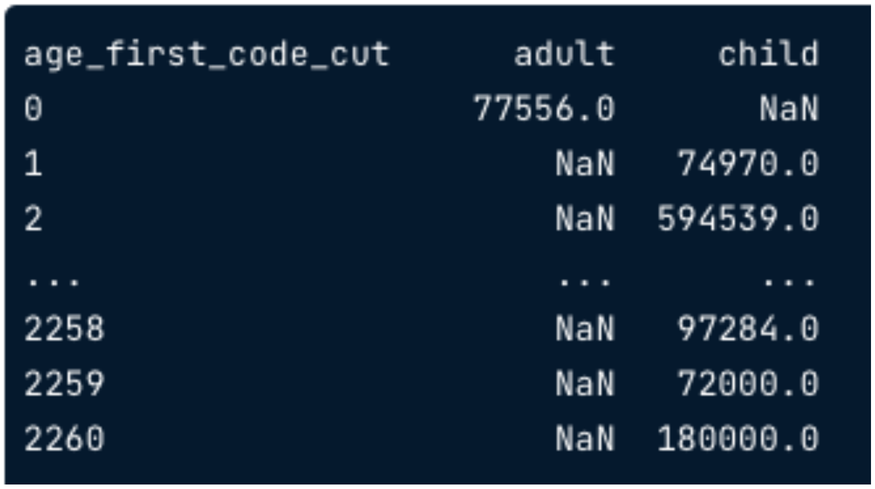
repub_votes_small[‘diff’] = repub_votes_small[‘repub_percent_08’] - repub_votes_small[‘repub_percent_12’]
print(repub_votes_small)
#take the absolute value of the differences
repub_votes_small[‘abs_diff’] = repub_votes_small[‘diff’].abs()
print(repub_votes_small)
#then rank the absolute differences using the ‘rankdata’ method
from scipy.stats import rankdata
repub_votes_small[‘rank_abs_diff’] = rankdata(repub_votes_small[‘abs_diff’])
print(repub_votes_small)
#calculate test statistic ‘W’
#W uses the signs of the diff column to split the ranks into two groups (one for rows with negative differences and one for positive differences)
T_minus = 1 + 4 + 5 + 2 + 3 #sum of the ranks with negative differences
T_plus = 0
**Test statistic W is the smaller of T-minus and T-plus
in our example it is 0 or T_plus

Using pingouin
alpha = 0.01
pingouin.wilcoxon(x=repub_votes_potus_08_12_small[‘repub_percent_08’], y=repub_votes_potus_08_12_small[‘repub_percent_12’], alternative=‘less’)
output > W-val = 0, p-val = 0.03
**this p-value is over 10 times larger than the p-value then if we did a t-test
we should feel more confident with this result given the small sample size
*the Wilcoxon test indicates that we do not have evidence that the 2008 Republican percentages are smaller thatn the 2012 percentages using this small sample of five rows
we faile to reject Hnaught 0.03 > 0.01

Non-parametric ANOVA and unpaired t-tests
non-parametric alternatives to tests of independent numeric samples
Wilcoxon-Mann-Whitney test also known as the Mann Whitney U test
roughly speaking is a t-test on ranked data
can perform hypothesis tests on the ranks of a numeric input
test is similar to the Wilcoxon test but works on unpaired data

Example
evaluate the relationship between converted compensation and age respondents began coding
first focus on these two columns, creating a new DF with just these columns
age_vs_comp = stack_overflow[[‘converted_comp’, ‘age_first_code_cut’]]
we then need to convert the data from long to wide format

do this with the ‘pivot’ method
‘pivot’ method unlike ‘pivot_table’ method does not aggregate, instead it returns the raw values for each group across the rows
age_vs_comp_wide = age_vs_comp.pivot(columns=‘age_first_code_cut’, values=‘converted_comp’)
output >



```
age_first_code_cut      adult     child
0                     77556.0       NaN
1                         NaN   74970.0
2                         NaN  594539.0
...                       ...       ...
2258                      NaN   97284.0
2259                      NaN   72000.0
2260                      NaN  180000.0
```

adult value of NaN corresponds to child
child value of NaN corresponds to adult

Using pingouin
alpha = 0.01
import pingouin
pingouin.mwu(x=age_vs_comp_wide[‘child’], y=age_vs_comp_wide[‘adult’], alternative=‘greater’)
#Ha is that people who code as children have ’higher’ income or ‘greater’ income so we use a right-tailed test
output > p-val is 1.9 to -19power, significantly smaller than the significance level

Kruskal-Wallis test
is to Wilcoxon-Mann-Whitney test as ANOVA is to t-test
Kruskal-Wallis test extends teh Wilcoxon-Mann-Whitney test to more than two groups
Kruska-Wallis test is a non-parametric ANOVA test
example - investigate if there is a difference in converted_comp between job satisfaction groups
alpha = 0.01
pingouin.kruskal(data=stack_overflow, dv=‘converted_comp’, between=‘job_sat’)
**unlike the Wilcoxon-Mann-Whitney test, we don’t need to pivot our data here since the kruskal method works on long data
output > p-unc 5.77 to -15power

smaller than our significance level
provides evidence that at least one of the mean compensation totals is different than the others across these five job satisfaction groups

Additional example

```
# Select the weight_kilograms and late columns
weight_vs_late = late_shipments[['weight_kilograms', 'late']]

# Convert weight_vs_late into wide format
weight_vs_late_wide = weight_vs_late.pivot(columns='late',
                                           values='weight_kilograms')

# Run a two-sided Wilcoxon-Mann-Whitney test on weight_kilograms vs. late
wmw_test = pingouin.mwu(x=weight_vs_late_wide['Yes'],
y=weight_vs_late_wide['No'], alternative='two-sided')

# Print the test results
print(wmw_test)
```