**Stats**

* Overview

**Statistics**

There are three types of lies - lies, damn lies, and statistics

* + *Benjamin Disraeli*

In this section we will review some basic descriptive statistics, talk about different probability distributions, and give an introduction to hypothesis testing.

**Goals**

* + Build on top of the statistics knowledge from the khan academy prework
  + Understand how to use the uniform, binomial, poisson, and normal distributions to model real-world scenarios
  + Understand in general how hypothesis testing is performed
  + Know when to use a t-test, correlation test, and $χ^2$ test
  + Write python code that simulates experiments in order to calculate an experimental probability
  + Use various statistical distributions in python through scipy.stats
  + Perform hypothesis testing in python code

**A Note on Visualizations,**[**the viz module**](https://ds.codeup.com/stats/viz.py)

Throughout the statistics curriculum you will see a module named viz imported in the code examples. This module contains some complex matplotlib plotting code, and is available for reference [here](https://ds.codeup.com/stats/viz.py). The intent of putting the code in a seperate module is to not distract from the lesson at hand.

**Descriptive Statistics**

Before we move on to more advanced concepts, we'll review some descriptive statistics terminology.

Descriptive statistics, as the name implies, let us *describe* a set of data.

We will discuss two main categories of descriptive statistics, measures of central tendancy, and measures of spread. The measures we will discuss focus on a single variable, for example, age or height. These measures do **not** measure multiple variables at the same time, or the interaction/relationship between variables.

**Measures of Central Tendency**

All of the measures of central tendancy allow us to describe where the middle, or most mass of a data set is.

* + **mean**: the average value, i.e. the sum of all the values divided by the number of values.

$$ \frac{\sum^n\_{i=1}x\_i} n $$

The mean can be subject to influence by large outlier points.

* + **expected value**: similar to the average, except that each value is weighted by its probability

$$ E[X] = \sum^n\_{i=1} x\_ip\_i $$

* + **median**: The center, or middle value. When there are an even number of data points, the average of the two middle points.
  + **mode**: the most frequently occuring value
  + **bi-modal**: when two values tie for the mode
  + **trimmed mean**: The average after removing a certain percentage of outliers. Less sensitive to outliers than the mean.

**Measures of Spread**

The measures of spread allow us to describe how spread out a data set is. These measures can give us an idea of the shape of the data by describing how far points tend to be from the middle.

* + **Min**: smallest value
  + **Max**: largest value
  + **Range**: The difference between the max and the min
  + **Mean Absolute Deviation**: The average of deviations from a central point, usually the mean. People often think and talk in terms of mean absolute deviation. For example, "we sell 100 units a day ± 10%". This measure of spread is the most common measure of spread/variance in conversational language.
  + **Quantile**: The cut points that divide a probability distribution into equally sized continuous intervals. To divide our distribution into n equally sized intervals, there are n-1 quantiles.
    - To divide a distribution into 2 equally sized intervals, the median is the quantile. This is because the median is the middle of the range of values. Half of the values are below the median, and the other half of the values are above the median.
    - To create two evenly sized intervals from the distrubtion [1, 2, 3, 4, 5, 6], the median is 3.5. The first half of values are between 1 and 3.5, and the second half of the values are between 3.5 and the max.
  + **Quartile**: The cut points on a distrubtion to subdivide it into 4 equally sized intervals are called the quartiles. To create 4 equally sized quarters, we need 3 "cut points" called quartiles. The quartiles are commonly abbreviated as Q1, Q2, and Q3.
    - Our quartiles Q1, Q2, and Q3 are numbers that create boundries for our 4 quarters. The quartiles split the distribution into 4 different intervals.
      * 25% of the observations are between the min and Q1.
      * 50% of the observations are between the min and Q2.
      * 75% of the observations are between the min and Q3.
    - How to use the quartiles to subdivide our distribution into four evenly sized quarters.
      * The first quarter contains values between the min and Q1.
      * The second quarter contains values between Q1 and Q2.
      * The third quarter contains values between Q2 and Q3.
      * The fourth quarter contains values between Q3 and the max.
    - Consider x = np.array([1, 2, 3, 4, 5, 6, 7, 8]).
      * Q1 is 2.5 calculated by np.quantile(x, 0.25)
      * Q2 is 4.5 calculated by np.quantile(x, 0.5)
      * Q3 is 6.5, calculated by np.quantile(x, 0.75)
  + **Percentile**: A quantile cut into 100 equally sized intervals

The percentile can be interpreted as the point where a percentage of values fall below it.

For example, the 75% of the values fall below the 75th percentile.

* + **IQR**: The Interquartile Range, Q3-Q1 (75th percentile - 25th percentile)
  + **Variance**: The average squared distance between each point and the mean

$$ \frac{1}{n} \sum (x\_i-\mu)^2 $$

We may divide by n-1 if sample is small to correct for a bias

Note that the units of the variance are the units of the variable being measured squared.

* + **Standard Deviation**: The square root of the variance

$$ \sqrt{\frac{1}{n}\sum(x\_i-\mu)^2} $$

Measures the absolute variability, so the units can be compared to the original values

* + **Skew**
    - Symmetric
    - Left-skewed: A set of data values in which the mean is generally less than the median. The left tail of the distribution is longer than the right tail of the distribution.
    - Right-skewed: A set of data values in which the mean is generally greater than the median. The right tail of the distribution is longer than the left tail of the distribution.
* Simulation

**Simulation**

* + In this lesson, we will work through several examples of using random numbers to simulate real-world scenarios.
  + For reference, the [viz module](https://ds.codeup.com/stats/viz.py) contains the visuals used for these lessons.

%matplotlib inline

import numpy as np

import pandas as pd

import viz # curriculum example visualizations

np.random.seed(29)

**Generating Random Numbers with Numpy**

The numpy.random module provides a number of functions for generating random numbers.

* + np.random.choice: selects random options from a list
  + np.random.uniform: generates numbers between a given lower and upper bound
  + np.random.random: generates numbers between 0 and 1
  + np.random.randn: generates numbers from the standard normal distribution
  + np.random.normal: generates numbers from a normal distribution with a specified mean and standard deviation

**Example Problems**

**Carnival Dice Rolls**

You are at a carnival and come across a person in a booth offering you a game of "chance" (as people in booths at carnivals tend to do).

You pay 5 dollars and roll 3 dice. If the sum of the dice rolls is greater than 12, you get 15 dollars. If it's less than or equal to 12, you get nothing.

Assuming the dice are fair, should you play this game? How would this change if the winning condition was a sum greater than *or equal to* 12?

To simulate this problem, we'll write the python code to simulate the scenario described above, then repeat it a large amount of times.

One way we can keep track of all the simulations is to use a 2-dimensional matrix. We can create a matrix where each row represents one "trial". Each row will have 3 columns, representing the 3 dice rolls.

n\_trials = nrows = 10\_000

n\_dice = ncols = 3

rolls = np.random.choice([1, 2, 3, 4, 5, 6], n\_trials \* n\_dice).reshape(nrows, ncols)

rolls

array([[6, 4, 5],

[6, 3, 1],

[1, 2, 2],

...,

[6, 2, 1],

[3, 4, 3],

[4, 2, 4]])

Here we used the choice function to randomly select an element out of the list of the number 1-6, effectively simulating a dice roll. The second argument supplied to choice is the total number of dice to roll. Once we have generated all the dice rolls, we use the .reshape method to create our matrix with 3 columns and 10,000 rows.

Now that we have all of the simulated dice rolls, we want to get the sum of the dice rolls for each trial. To do this, we can use the .sum function and specify that we want the sum of every row (as opposed to the sum of all the numbers, or the sum by column) with the axis key word argument.

sums\_by\_trial = rolls.sum(axis=1)

sums\_by\_trial

array([15, 10, 5, ..., 9, 10, 10])

Let's pause here for a minute and visualize the data we have:

viz.simulation\_example1(sums\_by\_trial)

The area shaded in lightblue represents our chance of winning, that is, the number of times that the sum of 3 dice rolls is greater than 12.

We can now convert each value in our array to a boolean value indicating whether or not we won:

wins = sums\_by\_trial > 12

wins

array([ True, False, False, ..., False, False, False])

To calculate an overall win rate, we can treat each win as a 1 and each loss as 0, then take the average of the array:

win\_rate = wins.astype(int).mean()

win\_rate

0.2633

Now that we know our win rate, we can calculate the expected profit:

expected\_winnings = win\_rate \* 15

cost = 5

expected\_profit = expected\_winnings - cost

expected\_profit

-1.0505000000000004

So we would expect, based on our simulations, on average, to lose a little over a dollar everytime we play this game.

To answer the last part of the question, we can recalculate our win rate based on the sums being greater than or equal to 12:

wins = sums\_by\_trial >= 12

win\_rate = wins.astype(int).mean()

expected\_winnings = win\_rate \* 15

cost = 5

expected\_profit = expected\_winnings - cost

expected\_profit

0.5860000000000003

If our win condition changes to the sum being greater than or equal to 12, then, based on our simulations, on average, we expect to win about 58 cents.

**No Rest or Relaxation**

There's a 30% chance my son takes a nap on any given weekend day. What is the chance that he takes a nap at least one day this weekend? What is the probability that he doesn't nap at all?

Let's first do a little bit of setup:

p\_nap = .3

ndays = ncols = 2

n\_simulated\_weekends = nrows = 10\*\*5

To simulate the results from many weekends, we'll create a 2 x 10,000 matrix, with 2 being the number of days in a weekend and 10,000 being the number of simulations we want to run.

To determine whether or not a nap is taken on a given day, we'll generate a random number between 0 and 1, and say that it is a nap if it is less than our probability of taking a nap.

data = np.random.random((nrows, ncols))

data

array([[0.46762045, 0.70078355],

[0.18897809, 0.54312897],

[0.253291 , 0.43836437],

...,

[0.15008559, 0.37577491],

[0.34690321, 0.58934311],

[0.97135998, 0.57219933]])

naps = data < p\_nap

naps

array([[False, False],

[ True, False],

[ True, False],

...,

[ True, False],

[False, False],

[False, False]])

Now that we have each day as either true or false, we can take the sum of each row to find the total number of naps for the weekend. When we sum an array of boolean values, numpy will treat True as 1 and False as 0.

naps.sum(axis=1)

array([0, 1, 1, ..., 1, 0, 0])

Now we have the results of our simulation, an array where each number in the array represents how many naps were taken in a two day weekend.

viz.simulation\_example2(naps)

We can use this to answer our original questions, what is the probability that at least one nap is taken?

(naps.sum(axis=1) >= 1).mean()

0.50998

What is the probability no naps are taken?

(naps.sum(axis=1) == 0).mean()

0.49002

**One With Dataframes**

Let's take a look at one more problem:

What is the probability of getting at least one 3 in 3 dice rolls?

To simulate this, we'll use a similar strategy to how we modeled the dice rolls in the previous example, but this time, we'll store the results in a pandas dataframe so that we can apply a lambda function that will check to see if one of the rolls was a 3.

n\_simulations = nrows = 10\*\*5

n\_dice\_rolled = ncols = 3

rolls = np.random.choice([1, 2, 3, 4, 5, 6], nrows \* ncols).reshape(nrows, ncols)

(pd.DataFrame(rolls)

.apply(lambda row: 3 in row.values, axis=1)

.mean())

0.42324

Let's break down what's going on here:

* + First we assign values for the number of rows and columns we are going to use
  + Next we create the rolls variable that holds a 3 x 10,000 matrix where each element is a randomly chosen number from 1 to 6
  + Lastly we create a dataframe from the rolls
    - pd.DataFrame(rolls) converts our 2d numpy matrix to a pandas DataFrame
    - .apply(... applies a function to each **row** in our dataframe, because we specified axis=1, the function will be called with each row as it's argument. The body of the function checks to see if the value 3 is in the values of the row, and will return either True or False
    - .mean() takes our resulting series of boolean values, and treats True as 1 and False as 0, to give us the average rate of Trues, in this case, the simulated probability of getting a 3 in 3 dice rolls.

**Exercises**

Using the [repo setup directions](https://ds.codeup.com/fundamentals/git/), setup a new local and remote repository named statistics-exercises. The local version of your repo should live inside of ~/codeup-data-science. This repo should be named statistics-exercises

Do your work for this exercise in either a python file named simulation.py or a jupyter notebook named simulation.ipynb.

* + How likely is it that you roll doubles when rolling two dice?
  + If you flip 8 coins, what is the probability of getting exactly 3 heads? What is the probability of getting more than 3 heads?
  + There are approximitely 3 web development cohorts for every 1 data science cohort at Codeup. Assuming that Codeup randomly selects an alumni to put on a billboard, what are the odds that the two billboards I drive past both have data science students on them?
  + Codeup students buy, on average, 3 poptart packages with a standard deviation of 1.5 a day from the snack vending machine. If on monday the machine is restocked with 17 poptart packages, how likely is it that I will be able to buy some poptarts on Friday afternoon? (Remember, if you have mean and standard deviation, use the np.random.normal) *You'll need to make a judgement call on how to handle some of your values*
  + Compare Heights
    - Men have an average height of 178 cm and standard deviation of 8cm.
    - Women have a mean of 170, sd = 6cm.
    - Since you have means and standard deviations, you can use np.random.normal to generate observations.
    - If a man and woman are chosen at random, what is the likelihood the woman is taller than the man?
  + When installing anaconda on a student's computer, there's a 1 in 250 chance that the download is corrupted and the installation fails. What are the odds that after having 50 students download anaconda, no one has an installation issue? 100 students?

What is the probability that we observe an installation issue within the first 150 students that download anaconda?

How likely is it that 450 students all download anaconda without an issue?

* + There's a 70% chance on any given day that there will be at least one food truck at Travis Park. However, you haven't seen a food truck there in 3 days. How unlikely is this?

How likely is it that a food truck will show up sometime this week?

* + If 23 people are in the same room, what are the odds that two of them share a birthday? What if it's 20 people? 40?

Be sure to add, commit, and push your work.

**Bonus Exercises**

* + [Mage Duel](https://gist.github.com/ryanorsinger/2996446f02c1bf30fcb3f8fdb88bd51d)
  + [Chuck a Luck](https://gist.github.com/ryanorsinger/eac1d7b7e978f90b8390bdc056312123)
* Probability Distributions

**Probability Distributions**

Probability distributions are mathematical functions that we can use to model real-world processes. These distributions provide the probabilities of occurrence of different possible outcomes in an experiment.

In this lesson we will discuss four of the most common distributions: uniform, normal, binomial, and poisson. We will also discuss how to work with these distributions with python and scipy.

import matplotlib.pyplot as plt

import numpy as np

from scipy import stats

import viz # curriculum viz example code

np.random.seed(123)

**Types of Distributions**

* + Uniform distributions have equal likelihoods among all outcomes, like a fair coin.
  + Binomial distributions are all about determining a binary outcome of an event. Success/failure, for example.
  + Normal distributions model a continuous random variable.
  + Poisson distributions model a certain amount of events occuring over a time interval
  + There are [many more](https://www.itl.nist.gov/div898/handbook/eda/section3/eda366.htm) distribution shapes. This lesson will focus on the first four.

**Working With Distribution Objects from scipy.stats**

* + Consider the situation at hand and determine the appropriate distribution type.
  + Create the distribution object using the stats module from scipy.
  + Ask yourself what information you have and what information you need.
  + Utilize the diagram below to call the appropriate distribution. *Hey Python, you do the calculus!*
  + Because these distribution objects represent the distribution itself and not specific numbers, use the rvs method if you need to generate actual random numbers (for visualizing or using to produce simulation experiments)

**Diagram to Select the Appropriate Distribution Method**

**Uniform Distribution**

The uniform distribution can be used to model events where the outcome is discrete and each outcome has an equally likely chance of happening.

An example of an event that can be modeled with the uniform distribution is the outcome of rolling a 6-sided die.

Since the uniform distribution is conceptually simple, we will use it as an example of how to work with different distributions in scipy.

**Working with distributions in SciPy**

Scipy provides many different ways of interacting with various statistical distributions through it's stats module.

We will discuss the following distribution methods:

* + rvs
  + pmf / pdf
  + cdf / ppf
  + sf / isf

Before using any of the above methods, a distribution must be created, and any parameters that define that distribution must be defined (we'll see more examples of this as we discuss other types of distributions). To represent rolling a dice, we'll use the randint distribution and specify that the outcomes are from 1 - 6

die\_distribution = stats.randint(1, 7)

While our example here is fairly simple, and the values we are calculating could easily be calculated by hand, the same principles that we get into here will apply to all the distributions we talk about where the calculations are not so easy.

Now that we have created the distribution object, we will explore some different ways to use it.

**Random Values**

We can generate random values based on the distribution with the rvs method. We can pass

* + no arguments to get a single random value
  + a single integer to get that many random values
  + a tuple with the dimensions of a matrix of random values

die\_distribution.rvs()

6

die\_distribution.rvs(5)

array([3, 5, 3, 2, 4])

die\_distribution.rvs((5, 5))

array([[3, 4, 2, 2, 1],

[2, 2, 1, 1, 2],

[4, 6, 5, 1, 1],

[5, 2, 4, 3, 5],

[3, 5, 1, 6, 1]])

Generating random values can be a good way to visualize the distribution.

n = 10\_000

x = die\_distribution.rvs(n)

plt.hist(x, bins=range(0, 9), align='left', width=1, edgecolor='black')

plt.title(f'Outcome of {n:,} Dice Rolls')

Text(0.5, 1.0, 'Outcome of 10,000 Dice Rolls')

**PMF / PDF**

The **probability mass function (pmf)** (*probability density function (pdf)* for continuous distributions) is a function that gives us the probability of any single outcome. For example, we could use the pmf to give us the probability of rolling a 3 with our dice rolling distribution:

die\_distribution.pmf(3)

0.16666666666666666

**SciPy Distribution API**

All of the functions that we discuss here can accept a single value or a list of values and will produce either a single number, or a numpy array of results that correspond to the inputs.

For example, we can calculate multiple pmfs at once like this:

die\_distribution.pmf([1, 2, 3])

**CDF / PPF**

The **cumulative distribution function** tells us the likelihood of a single outcome or all the results below it. For our dice rolling example, this might be something like "what is the probability of rolling a 3 or lower?"

die\_distribution.cdf(3)

0.5

We can visualize this by plotting each of the outcomes against the likelihood of that outcome, and shading the area that we wish to know the probability of:

X = outcome of one dice roll

viz.distributions\_example1(die\_distribution)

Here, the probability of rolling a 3 or lower is represented by the area of the distribution that is shaded, in our case, .5.

The **percent point function (ppf)** (also known as the quantile function) can be thought of as the inverse of the cdf.

The ppf accepts a probability, and gives us the value that is associated with that probability:

die\_distribution.ppf(5/6)

5.0

**SF / ISF**

The **survival function (sf)** tells us what the probability of our random variable falling above a certain value is. This is the same as 1 minus the cdf of the same value.

We can use this to answer questions like: "What is the likelihood we roll a value higher than 4?"

die\_distribution.sf(4)

0.33333333333333337

Visualizing the above calculation:

viz.distributions\_example2(die\_distribution)

Like the ppf, the **inverse survival function (isf)** will give us a value when we provide a probability.

For example: "There is a 1/3 chance a dice roll will be higher than what value?"

die\_distribution.isf(1/3)

4.0

**Binomial Distribution**

The binomial distribution lets us model the number of successes after a number of trials, given a certain probability of success. The classic example of this is the number of heads you would expect to see after flipping a coin a certain number of times.

A binomial distribution is defined by a number of trials, and a probability of success. These two pieces of information are what we need in order to model a problem with the binomial distribution.

The binomial distribution assumes that each trial is independent of the others.

Let's take an example:

You are taking a multiple choice test consisting of 30 questions that you forgot to study for. Each question has 4 possible answers and you will choose one at random. What is the probability you get more than 10 of the questions right?

Here we have a probability of success, 0.25, and a number of trials, 30. We'll define X as the number of questions we get right on the test. We want to know the probability that X > 10, which tells us we want to use the survival function.

stats.binom(30, .25).sf(10)

0.10572812269266013

Let's visualize what we are doing here as well. Here is what a binomial distribution looks like for our problem, that is, with n = 30 and p = .25.

viz.distributions\_example3()

When we ask for the probability of more than 10 successes (i.e. questions correct), we are looking for the area in the pmf distribution where x is greater than 10.

viz.distributions\_example4()

Conceptually speaking, the survival function is equal to the sum of all of the areas shaded in green above.

The definition of a "success" can vary quite a bit. Really we just need a random variable with two outcomes, and we define one of the two outcomes as a success. For example, we could use the binomial distribution to help answer the following question:

Suppose there is a 5% chance that a Codeup student will show up late to class. With a class of 20, what is the likelihood that everyone shows up on time?

Here we define a "success" as a student showing up late, and this outcome has a probability of 0.05. We want to know what the probability of 0 successes is. We now have all the pieces we need to use the binomial distribution to answer the question.

stats.binom(20, .05).pmf(0)

0.3584859224085419

Again, a visualization can be helpful:

viz.distributions\_example5()

Let's visualize what various binomial distributions look like with some simulations:

viz.distributions\_example6()

**Bernoulli Distribution**

A binomial distribution with an n of 1 is referred to as a **Bernoulli Distribution**.

**Normal Distribution**

The normal distribution models a continuous random variable where the further away from the mean you are, the less likely the outcome. This is commonly referred to as the "bell curve", and many continous variables tend to follow a normal distribution.

A normal distribution is defined by a mean and a standard deviation. The **standard normal distribution** is a normal distribution with a mean of 0 and standard deviation of 1.

Suppose that a store's daily sales are normally distributed with a mean of 12,000 dollars and standard deviation of 2000 dollars. How much would the daily sales have to be to be in the top 10% of all days?

Here we are given the mean and standard deviation, and are asked to find the value that corresponds to the top 10%. Here, since we know the probability and want a value, we can use the percent point function to find our answer.

μ = 12000

σ = 2000

sales = stats.norm(μ, σ)

top\_10\_percent\_cutoff = sales.ppf(.9)

print('${:,.2f}'.format(top\_10\_percent\_cutoff))

$14,563.10

Let's visualize what we are doing as well:

viz.distributions\_example7(μ, σ)

Using the same data as before, let's answer another question:

How likely is it that the store sells less than 10,000 dollars one day?

Here we want to know the probability that our random variable takes on a value less than 10,000, so we can use the cdf.

p = sales.cdf(10\_000)

print(f'Ony any given day, there\\'s a {p:.1%} chance we sell less than $10,000.')

Ony any given day, there's a 15.9% chance we sell less than $10,000.

Here is what various normal distributions look like with different means and standard deviations:

viz.distributions\_example8()

**Poisson Distribution**

The poisson distribution lets us model a situation where a certain number of events happen over a specified time interval[**1**](https://ds.codeup.com/stats/probability-distributions/#fn:1). The number of events that happen is a discrete measure, and this distribution can tell us the likelihood of a certain number of events occuring over the time period.

The poisson distribution assumes that the events are indpendent of each other and independent of the time since the last event. We must also know the average rate to use a poisson distribution.

Some examples of real-world processes that can be modeled with a poisson distribution are:

* + The number of emails sent by a mail server in a day
  + The number of phone calls received by a call center per hour
  + The number of decay events per second from a radioactive source

Let's dive into a specific example:

Codeup knows that, on average, students consume 5 lbs of coffee per week. How likely is it that the coffee consumption for this week is only 3 lbs?

stats.poisson(5).pmf(3)

0.1403738958142805

viz.distributions\_example10()

What is the likelihood that more than 7 lbs of coffee are consumed?

stats.poisson(5).sf(7)

0.13337167407000744

viz.distributions\_example11()

Let's take a look at what the poisson distribution looks like with different average rates:

viz.distributions\_example12()

**Further Reading**

* + [Stats: Probability Rules](https://people.richland.edu/james/lecture/m170/ch05-rul.html)
  + [Wikipedia: Poisson Distribution](https://en.wikipedia.org/wiki/Poisson_distribution)
  + [Details on how to calculate the probability of at least one event occuring](https://www.quora.com/If-there-are-two-independent-events-that-could-occur-A-with-a-chance-of-20-and-B-with-a-chance-of-60-what-is-the-probability-that-at-least-one-of-these-events-will-occur-Explain-how-you-calculated-the-answer)
  + [Probability exercise and quiz questions](https://www.analyticsvidhya.com/blog/2017/04/40-questions-on-probability-for-all-aspiring-data-scientists/)
  + [Statistics How To: Statistics for the rest of us](https://www.statisticshowto.datasciencecentral.com/)

**Exercises**

Do your work for this exercise in either a python script named probability\_distributions.py or a jupyter notebook named probability\_distributions.ipynb.

For the following problems, use python to simulate the problem and calculate an experimental probability, then compare that to the theoretical probability.

* + A bank found that the average number of cars waiting during the noon hour at a drive-up window follows a Poisson distribution with a mean of 2 cars. Make a chart of this distribution and answer these questions concerning the probability of cars waiting at the drive-up window.
    - What is the probability that no cars drive up in the noon hour?
    - What is the probability that 3 or more cars come through the drive through?
    - How likely is it that the drive through gets at least 1 car?
  + Grades of State University graduates are normally distributed with a mean of 3.0 and a standard deviation of .3. Calculate the following:
    - What grade point average is required to be in the top 5% of the graduating class?
    - What GPA constitutes the bottom 15% of the class?
    - An eccentric alumnus left scholarship money for students in the third decile from the bottom of their class. Determine the range of the third decile. Would a student with a 2.8 grade point average qualify for this scholarship?
    - If I have a GPA of 3.5, what percentile am I in?
  + A marketing website has an average click-through rate of 2%. One day they observe 4326 visitors and 97 click-throughs. How likely is it that this many people or more click through?
  + You are working on some statistics homework consisting of 100 questions where all of the answers are a probability rounded to the hundreths place. Looking to save time, you put down random probabilities as the answer to each question.
    - What is the probability that at least one of your first 60 answers is correct?
  + The codeup staff tends to get upset when the student break area is not cleaned up. Suppose that there's a 3% chance that any one student cleans the break area when they visit it, and, on any given day, about 90% of the 3 active cohorts of 22 students visit the break area. How likely is it that the break area gets cleaned up each day? How likely is it that it goes two days without getting cleaned up? All week?
  + You want to get lunch at La Panaderia, but notice that the line is usually very long at lunchtime. After several weeks of careful observation, you notice that the average number of people in line when your lunch break starts is normally distributed with a mean of 15 and standard deviation of 3. If it takes 2 minutes for each person to order, and 10 minutes from ordering to getting your food, what is the likelihood that you have at least 15 minutes left to eat your food before you have to go back to class? Assume you have one hour for lunch, and ignore travel time to and from La Panaderia.
  + Connect to the employees database and find the average salary of current employees, along with the standard deviation. For the following questions, calculate the answer based on modeling the employees salaries with a normal distribution defined by the calculated mean and standard deviation then compare this answer to the actual values present in the salaries dataset.
    - What percent of employees earn less than 60,000?
    - What percent of employees earn more than 95,000?
    - What percent of employees earn between 65,000 and 80,000?
    - What do the top 5% of employees make?

**Hint** If you're looking at this exercise and wondering "How do I get pandas to talk the database, again?", remember that you'll need 3 things: your .gitignore, your env.py, and to use pd.read\_sql. Copy over your .gitignore and env.py from your data science libraries exercises folder, and connect to the employees database like so:

import pandas as pd

import env

url = f'mysql+pymysql://{env.user}:{env.password}@{env.host}/employees'

pd.read\_sql('SELECT \* FROM departments', url)

* + While time is the most common interval used, the poisson distribution can also be used for intervals like distance, area, or volume. [↩](https://ds.codeup.com/stats/probability-distributions/#fnref:1)
* Hypothesis Testing
  + [Overview](https://www.canva.com/design/DAFydlGRL_w/5nc0gOERA7FDG4AU_nSu-A/view?utm_content=DAFydlGRL_w&utm_campaign=designshare&utm_medium=link&utm_source=viewer)

**Hypothesis Testing**

Hypothesis testing is the process of comparing one hypothesis to another and using statistics to help evaluate the hypothesis. It is part of the branch of statistics known as Inferential Statistics.

In this lesson we will introduce some broad concepts related to hypothesis testing, and in future lessons we will dive into a few specific hypothesis tests.

The terms covered in this lesson are summarized in the table below:

| **Term** | **Formula / Symbol** | **Description** |
| --- | --- | --- |
| Null Hypothesis | $H\_0$ | The "default" hypothesis; usually no change, no effect, etc |
| Alternative Hypothesis | $H\_1$ or $H\_a$ | The "other" hypothesis; states there is some relationship |
| Significance Level, False Positive Rate | $α$ | P(FP) = P(Type I Error) |
| Statistical Power | $1−β$ | P(Reject $H\_0$ when $H\_0$ is false) |
| False Negative Rate | $β$ | P(FN) = P(Type II Error) |
| p-value | $p$ | P(We observed this result due to chance |

**Sample vs. Population**

According to [[Scribbr.com](http://Scribbr.com)]([https://www.scribbr.com/methodology/population-vs-sample/#:~:text=A population is the entire,t always refer to people.)](https://www.scribbr.com/methodology/population-vs-sample/#:~:text=A%20population%20is%20the%20entire,t%20always%20refer%20to%20people.)), "A **population** is the entire group that you want to draw conclusions about. A **sample** is the specific group that you will collect data from. The size of the sample is always less than the total size of the population. In research, a population doesn’t always refer to people. It can mean a group containing elements of anything you want to study, such as objects, events, organizations, countries, species, organisms, etc."

Let's say we have a dataset that contains information about 600 statistics students: their hair and eye color and their sex.

One thing we know is we have a sample of *some* population.

What we don't know is what population it is a representative sample of.

Some questions we may ask:

* + - Is this a representative sample of *all* statistics students?
    - ...of *all* students?
    - ...of statistics students in a certain region of the world?
    - ...of college statistics students in the US?
    - ...of people in the U.S.?
    - ...(and on and on)

Let's take, for example, the question of "Is this a representative sample of people in the U.S.?"

We can use statistical testing to determine the probability that this sample is a sample of the general U.S. population. My hypothesis is that it is not of the general U.S. population because within this group of students, I found that 36% of them have blue eyes. However, according to [heffingtons](https://heffingtons.com/interesting-facts-about-eye-color/), the estimated population proportion of blue eyes in the U.S. is 27%.

* + - Brown Eyes: 45%
    - Blue Eyes: 27%
    - Hazel Eyes: 18% (Note: Hazel eyes consist of shades of brown and green.)
    - Green Eyes: 9%
    - Other: 1%

But is this difference significant? I.e. If I took a sample of 600 people across the U.S., what is the probability that 36% of that sample had blue eyes? I'm going to say that if the probability of that happening is < 5%, then I will conclude that this is a sample of a different population. If it is > 5%, I will conclude that the 9% difference was just by chance and this still appears to be a sample of the U.S. population as a whole.

Another option is to look at hair and eye color together and look for relationships. Possible questions could be:

* + - Are brown eyes more likely to be associated with brown or black hair?
    - Are green eyes more likely to be associated with blond or red hair?

Here we would be testing relationships, to see if two categorical variables are dependent on each other. In that case we could use something like a Chi-Square test for independence.

**Null and Alternate Hypothesis**

When performing formal statistical hypothesis testing, the question being asked needs to be phrased as a **null hypothesis** ($H\_0$) and an **alternative hypothesis** ($H\_a$) . The null hypothesis is the "status quo" and usually reflects no change or no difference, while the alternative hypothesis says that there is a difference or change.

Some examples:

$H\_0$: There is no difference between right-handed people and left-handed individual's heights. $H\_a$: There is a difference between right-handed people and left-handed individual's heights. $H\_0$: The amount of sleep a student gets the night before an exam makes no difference on the student's exam score. $H\_a$: Less sleep the night before an exam leads to a lower exam score.

The results of a hypothesis test will lead us to either **reject the null hypothesis** or **fail to reject the null hypothesis**. Strictly speaking, this does not tell us that the alternative hypothesis is true.

The alternative hypothesis can either be that there is a difference or that the difference is either greater or less than. This tells us whether we are setting up a **two-tailed** (for any difference) or **one-tailed** (for a specific difference) test.

General hypothesis tests process:

* + - Choose the right type of test for your data / question (elaborated on in future lessons)
    - Form hypotheses and set a desired confidence level
    - Calculate the appropriate test statistics and p-value
    - Conclude based on the above statistics

**Hypothesis Test Results**

Once we determine the correct hypothesis test, we choose a **confidence interval**, a range of values that contains the true value a certain percent of the time. By choosing a confidence interval, we set our **significance level**, $\alpha$ (alpha) as well. $\alpha$ is defined as 1 - our confidence level. Typical values for our confidence interval are 95%, 99% and 99.9%.

**p-value**

One of the values we will obtain from a hypothesis test is a **p-value**. The calculation differs depending on the specific type of test we are running, but is interpreted the same way.

The p-value is the chance that we obtained the results we did (or would obtain more extreme results) if the null hypothesis is true.

For example, imagine we were testing the hypothesis that Codeup students that drink coffee have higher grades. Our hypotheses would be:

* + - $H\_0$: There is no difference in grade for coffee and non-coffee drinkers.
    - $H\_a$: Coffee drinkers have higher grades than non-coffee drinkers.

Let's imagine we end up with a p-value of .05. This means that if it's true that there is no difference in grades, and we ran the experiment 20 times, we would expect 1 out of the 20 experiments to tell us that there is a difference in grades, purely due to chance.

Based on our previously set confidence interval and p-value, we decide whether to reject the null hypothesis or not. If our p-value is less than $\alpha$, we reject the null hypothesis, otherwise, we fail to reject the null hypothesis.

Note that p-values don't tell us anything about effect size.

**Hypothesis Testing Errors**

There are two types of errors we will encounter with hypothesis testing:

* + - A **type I** error is when we reject the null hypothesis, but, in reality, the null hypothesis is true.
    - A **type II** error is when we fail to reject the null hypothesis when it is actually false.

The table below shows us the possible outcomes of a hypothesis test.

|  | **$H\_0$ is true** | **$H\_0$ is false** |
| --- | --- | --- |
| Accept $H\_0$ | True Negative | False Negative (Type II Error) |
| Reject $H\_0$ | False Positive (Type I Error) | True Positive |

**In Practice**

This question and others are what we answer using hypothesis testing.

Thinking about the telco-churn dataset, here are some questions we might initially have of the data:

* + - Do customers churn because their bills are too high?
    - Is their internet too slow? Maybe there's something wrong with certain internet options?
    - Do customers get charged more the longer they are there?

Questions do not usually start off in an organized way. So let's restructure these questions into something that will help us know how to test the questions.

* + - Do those who churn spend more than those who do not churn?
    - Are certain internet types more or less likely to churn?
    - Is there a linear relationship between tenure and average monthly charges?

Let's explicitly call out the associated variables and their datatype for these questions.

* + - Do those who churn (has\_churned, boolean) spend more each month (avg\_monthly\_spend, numeric) than those who do not churn?
    - Are customers with Fiber (has\_fiber, boolean) more likely to churn (has\_churned, boolean) than those without?
    - Are sr. citizens (is\_senior\_citizen, boolean) more likely to churn (has\_churned, boolean)?
    - Are customers without auto payment (has\_autopayment, boolean) more likely to churn (has\_churned, boolean)?
    - Do customers who churn (has\_churned, boolean) have lower tenure (tenure\_months, numeric)?
    - Is there a linear relationship between tenure (tenure\_months, numeric) and total charges (ttl\_charges, numeric)?
    - Is there a linear relationship between tenure (tenure\_months, numeric) and average monthly charges (avg\_monthly\_charges, numeric)?

The types of tests we run depends on the question and data types:

* + - Do those who churn (has\_churned, boolean) spend more each month (avg\_monthly\_spend, numeric) than those who do not churn? (boolean x numeric: comparison of means (t-test) across the 2 groups)
    - Are customers with Fiber (has\_fiber, boolean) more likely to churn (has\_churned, boolean) than those without? (boolean x boolean: comparison of proportions/relationships)
    - Are sr. citizens (is\_senior\_citizen, boolean) more likely to churn (has\_churned, boolean)? (boolean x boolean: comparison of proportions/relationships)
    - Are customers without auto payment (has\_autopayment, boolean) more likely to churn (has\_churned, boolean)? (boolean x boolean: comparison of proportions/relationships)
    - Do customers who churn (has\_churned, boolean) have lower tenure (tenure\_months, numeric)? (boolean x numeric: comparison of means (t-test) across the 2 groups)
    - Is there a linear relationship between tenure (tenure\_months, numeric) and total charges (ttl\_charges, numeric)? (numeric x numeric: linear correlation between two continuous values, does one affect the other. (pearson's correlation))
    - Is there a linear relationship between tenure (tenure\_months, numeric) and average monthly charges (avg\_monthly\_charges, numeric)? (numeric x numeric: linear correlation between two continuous values, does one affect the other. (pearson's correlation))

**Exercises**

Do your work for this exercise in a jupyter notebook named hypothesis\_testing.ipynb.

For each of the following questions, formulate a null and alternative hypothesis (be as specific as you can be), then give an example of what a true positive, true negative, type I and type II errors would look like. Note that some of the questions are intentionally phrased in a vague way. It is your job to reword these as more precise questions that could be tested.

* + - Has the network latency gone up since we switched internet service providers?
    - Is the website redesign any good?
    - Is our television ad driving more sales?
  + [Comparison of Groups](https://www.canva.com/design/DAFzCBrS-8M/vQwI3BqzVb_SPidOZTX67w/view?utm_content=DAFzCBrS-8M&utm_campaign=designshare&utm_medium=link&utm_source=viewer)

**Comparison of Groups**

**The Chi-Square Test of Independence**

While there are other $\chi^2$ ('$k \bar{i}$') tests, we will be using the *test of independence*. This test can be used to compare two categorical variables. It uses contingency tables to test the hypothesis that one group is independent of another. A contingency table displays frequencies for two categorical variables. This test helps us answer questions like:

* + - Is whether or not a customer churns independent of their subscription plan?
    - Are doctors less likely to smoke?
    - Does playing on the home field give a soccer team an advantage?

For all hypothesis tests, we will follow the same process:

* + - Form hypotheses and set a desired confidence level
    - Calculate the appropriate test statistics and p-value
    - Conclude based on the above statistics

To calculate the test statistic for $\chi^2$ manually (example in bonus content below):

* + - Create contingency table of observed values
    - Create contingency table of expected values
    - $\chi^2 = \sum \frac{(O-E)^2}{E}$, where $O$ is the observed values and $E$ is the expected values

To calculate the test statistic for $\chi^2$ using python:

* + - Create contingency table of observed values
    - Use scipy.stats.chi2\_contingency to generate a contingency table of expected values, test-statistic and p-value

For this lesson, we will look at the dataset on cars that we explored previously.

`import pandas as pd import numpy as np

from pydataset import data from scipy import stats

mpg = data('mpg') mpg['transmission'] = mpg.trans.str[:-4] # a little cleaning goes a long way mpg.head()`

|  | **manufacturer** | **model** | **displ** | **year** | **cyl** | **trans** | **drv** | **cty** | **hwy** | **fl** | **class** | **transmission** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | audi | a4 | 1.8 | 1999 | 4 | auto(l5) | f | 18 | 29 | p | compact | auto |
| 2 | audi | a4 | 1.8 | 1999 | 4 | manual(m5) | f | 21 | 29 | p | compact | manual |
| 3 | audi | a4 | 2.0 | 2008 | 4 | manual(m6) | f | 20 | 31 | p | compact | manual |
| 4 | audi | a4 | 2.0 | 2008 | 4 | auto(av) | f | 21 | 30 | p | compact | auto |
| 5 | audi | a4 | 2.8 | 1999 | 6 | auto(l5) | f | 16 | 26 | p | compact | auto |

**Example 1**

We will investigate the question of whether the car's drive is independent of transmission type.

**1. Form hypotheses and set a desired confidence level**

* + - $H\_0$ (Null Hypothesis): drive is independent of transmission type
    - $H\_a$ (Alternative Hypothesis): drive is dependent on transmission type

State our alpha to set our confidence interval.

alpha = 0.05

**2. Calculate the appropriate test statistics and p-value**

Create a contingency table of observed values from the dataframe's two columns of interest. We will use the pandas crosstab function to accomplish this.

observed = pd.crosstab(mpg.drv, mpg.transmission) observed

| **transmission** | **auto** | **manual** |
| --- | --- | --- |
| drv |  |  |
| 4 | 75 | 28 |
| f | 65 | 41 |
| r | 17 | 8 |

We will send the contingency table of observed values into the chi2\_contingency function.

The chi2\_contingency function returns 4 items (*in this order*):

* + - the **test statistic**: $\chi^2$
    - the **p-value**: the probability of seeing these proportions by chance
    - the **degrees of freedom**: equivalent to (rows - 1) \* (columns - 1)
    - the contingency table of the **expected values**, which represents what the values would be if everything was proportional and there was no relationship between the 2 variables.

chi2, p, degf, expected = stats.chi2\_contingency(observed)

Let's look at our results:

`# print 'Observed Values' followed by a new line print('Observed Values\n')

**print the values from the 'observed' dataframe**

print(observed.values)

**print --- and then a new line, 'Expected Values', followed by another new line**

print('---\nExpected Values\n')

**print the expected values array**

print(expected.astype(int))

**print a new line**

print('---\n')

**print the chi2 value, formatted to a float with 4 digits.**

print(f'chi^2 = {chi2:.4f}')

**print the p-value, formatted to a float with 4 digits.**

print(f'p = {p:.4f}')`

Observed Values

[[75 28]

[65 41]

[17 8]]

---

Expected Values

[[69 33]

[71 34]

[16 8]]

---

chi^2 = 3.1368

p = 0.2084

**3. Conclude based on the above statistics**

We can see by comparing the contingency tables that the observed values are very close to the expected values. We can confirm that, with the data available, there does not appear to be a significant relationship between type of transmission and type of drive.

if p < alpha: print('We reject the null hypothesis') else: print('We fail to reject the null hypothesis')

We fail to reject the null hypothesis

**Example 2**

We will now investigate the question of whether the car's class is independent of number of cylinders. Number of cylinders, while it is represented numerically, is a discrete variable. It is not continuous and there are a limited number of options.

**1. Form hypotheses and set a desired confidence level**

* + - $H\_0$ (Null Hypothesis): class is independent of cylinders.
    - $H\_a$ (Alternative Hypothesis): class is dependent on cylinders.

State our alpha to set our confidence interval.

alpha = 0.05

**2. Calculate the appropriate test statistics and p-value**

Create a contingency table of observed values from the dataframe's two columns of interest.

observed = pd.crosstab(mpg['class'], mpg.cyl) observed

| **cyl** | **4** | **5** | **6** | **8** |
| --- | --- | --- | --- | --- |
| class |  |  |  |  |
| 2seater | 0 | 0 | 0 | 5 |
| compact | 32 | 2 | 13 | 0 |
| midsize | 16 | 0 | 23 | 2 |
| minivan | 1 | 0 | 10 | 0 |
| pickup | 3 | 0 | 10 | 20 |
| subcompact | 21 | 2 | 7 | 5 |
| suv | 8 | 0 | 16 | 38 |

When we run a chi-square test of independence, we are testing whether there is a relationship between **at least** 2 categories of one variable with **at least** 2 categories of the other variable. The results do not tell us where the relationship lies, or that there is a relationship between all categories. We can do additional testing to figure out where the relationship lies, if necessary.

Therefore, when we run a chi-square test with these 2 variables, the resulting p-value will tell us whether there is a relationship between at least 2 classes of vehicles with at least 2 types of cylinders.

chi2, p, degf, expected = stats.chi2\_contingency(observed) p

1.5351076620141742e-20

**3. Conclude based on the above statistics**

def eval\_results(p, alpha, group1, group2): ''' this function will take in the p-value, alpha, and a name for the 2 variables you are comparing (group 1 and group 2) ''' if p < alpha: print(f'p-value: {p}') print(f'We reject the null hypothesis.') print(f'There exists some relationship between {group1} and the {group2}.') else: print(f'p-value: {p}') print(f'We fail to reject the null hypothesis.') print(f'There is not a significant relationship between {group1} and {group2}.')

eval\_results(p, alpha, group1='class', group2='cylinders')

p-value: 1.5351076620141742e-20

We reject the null hypothesis.

There exists some relationship between class and the cylinders.

Now we know there is *some* relationship. But where does that relationship exist?

**Continuing to investigate**

print("Expected DataFrame") print(pd.DataFrame(expected.astype('int'), index=observed.index, columns=observed.columns)) print("\\n") print("Observed DataFrame") print(observed)

Expected DataFrame

cyl 4 5 6 8

class

2seater 1 0 1 1

compact 16 0 15 14

midsize 14 0 13 12

minivan 3 0 3 3

pickup 11 0 11 9

subcompact 12 0 11 10

suv 21 1 20 18

Observed DataFrame

cyl 4 5 6 8

class

2seater 0 0 0 5

compact 32 2 13 0

midsize 16 0 23 2

minivan 1 0 10 0

pickup 3 0 10 20

subcompact 21 2 7 5

suv 8 0 16 38

There appear to be more than expected results in 2-seater vehicles with 8-cylinders, compact cars with 4-cylinders, midsized cars with 6-cylinders, minivans with 6-cylinders, pickups with 8-cylinders, subcompact cars with 4-cylinders, and finally SUVs with 8-cylinders.

If I want to identify where these are significant relationships, I can compare each of these "interesting groups", such as 2-seater vehicles vs. non-2-seater vehicles with 8-cylinders vs. non-8-cylinders.

`# create a variable that is a 1 if a vehicle is a 2 seater, and a 0 otherwise. mpg['class\_2seater'] = (mpg['class'] == '2seater').astype('int')

**create a variable that is a 1 if the vehicle is 8 cylinders and a 0 otherwise.**

mpg['cyl\_8'] = (mpg['cyl'] == 8).astype('int')

**generate a crosstab of these 2 new variables**

observed = pd.crosstab(mpg['class\_2seater'], mpg['cyl\_8'])

observed`

| **cyl\_8** | **0** | **1** |
| --- | --- | --- |
| class\_2seater |  |  |
| 0 | 164 | 65 |
| 1 | 0 | 5 |

# run chi-square test chi2, p, degf, expected = stats.chi2\_contingency(observed)

# evaluate results eval\_results(p, alpha, group1='2 seater cars', group2='8 cylinders')

p-value: 0.0030157710558452104

We reject the null hypothesis.

There exists some relationship between 2 seater cars and the 8 cylinders.

Compare one more interesting group: SUVs and 8-cylinders. We have already created the 8-cylinder boolean variable, so we just need to create a boolean variable for SUV.

`# create a new variable that is a 1 if the class is an SUV and a 0 otherwise. mpg['class\_SUV'] = (mpg['class'] == 'suv').astype('int')

**generate a crosstab**

observed = pd.crosstab(mpg.class\_SUV, mpg.cyl\_8) observed`

| **cyl\_8** | **0** | **1** |
| --- | --- | --- |
| class\_SUV |  |  |
| 0 | 140 | 32 |
| 1 | 24 | 38 |

`# run chi-square test chi2, p, degf, expected = stats.chi2\_contingency(observed)

**evaluate results**

eval\_results(p, alpha, group1='SUVs', group2='8 cylinders')`

p-value: 8.702491537516895e-10

We reject the null hypothesis.

There exists some relationship between SUVs and the 8 cylinders.

I can repeat these steps to evaluate each interesting group I came up with in my analysis of the expected vs. the observed to verify significant relationships. If the stakes are low, I can also choose to verify the groups that seem the least obvious (taking into account the difference of the values AND the sample size), and if those are significant, then we can pretty safely assume that the other interesting groups are significant.

**Exercises**

Continue working in your hypothesis\_testing notebook.

* + - Use the following contingency table to help answer the question of whether using a Macbook and being a Codeup student are independent of each other.

|  | **Codeup Student** | **Not Codeup Student** |
| --- | --- | --- |
| Uses a Macbook | 49 | 20 |
| Doesn't Use A Macbook | 1 | 30 |

* + - Choose another 2 categorical variables from the mpg dataset.
      * State your null and alternative hypotheses.
      * State your alpha.
      * Perform a $chi^2$ test of independence.
      * State your conclusion
    - Use the data from the employees database to answer these questions:
      * Is an employee's gender independent of whether an employee works in sales or marketing? (only look at current employees)
      * Is an employee's gender independent of whether or not they are or have been a manager?

**Bonus Content - Manual Calculation**

**Example 1**

For this example, we will look at the dataset on cars that we explored previously.

As we did above, we will investigate the question of whether the car's drive is independent of transmission type.

* + - $H\_0$ (Null Hypothesis): drive is independent of transmission type.
    - $H\_a$ (Alternative Hypothesis): drive is dependent on transmission type.

mpg = data('mpg') mpg['transmission'] = mpg.trans.str[:-4] # a little cleaning goes a long way mpg.head()

|  | **manufacturer** | **model** | **displ** | **year** | **cyl** | **trans** | **drv** | **cty** | **hwy** | **fl** | **class** | **transmission** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | audi | a4 | 1.8 | 1999 | 4 | auto(l5) | f | 18 | 29 | p | compact | auto |
| 2 | audi | a4 | 1.8 | 1999 | 4 | manual(m5) | f | 21 | 29 | p | compact | manual |
| 3 | audi | a4 | 2.0 | 2008 | 4 | manual(m6) | f | 20 | 31 | p | compact | manual |
| 4 | audi | a4 | 2.0 | 2008 | 4 | auto(av) | f | 21 | 30 | p | compact | auto |
| 5 | audi | a4 | 2.8 | 1999 | 6 | auto(l5) | f | 16 | 26 | p | compact | auto |

**Expected Values**

To begin with, we will calculate the values we would expect to see if the two groups are independent.

For each subgroup, we calculate the proportion of the total that it is, then multiply each subgroup's proportion by the proportion from every other subgroup to determine the expected values.

To start with, we'll calculate the proportions for transmission type:

`n = mpg.shape[0]

transmission\_proportions = mpg.transmission.value\_counts() / n transmission\_proportions`

auto 0.67094

manual 0.32906

Name: transmission, dtype: float64

This tells us that cars with automatic transmissions make up ~ 67% of the total, and cars with manual transmissions make up ~ 33% of the total.

Now we'll do the same for drive types.

drive\_proportions = mpg.drv.value\_counts() / n drive\_proportions

f 0.452991

4 0.440171

r 0.106838

Name: drv, dtype: float64

To find the overall proportions, we multiply all the combinations of proportions together.

For example, to find the expected proportion of automatic drive cars with 4-wheel drive, we would multiply those two proportions together.

.67∗.44=.2984.67∗.44=.2984

So we would expect about 29.84% of the total cars to be automatic and 4-wheel drive.

Below we show some code that will loop through all of the proportions and perform this calculation for all combinations of groups.

`expected = pd.DataFrame()

for transmission\_group, t\_prop in transmission\_proportions.items(): for drive\_group, d\_prop in drive\_proportions.items(): expected.loc[drive\_group, transmission\_group] = t\_prop \* d\_prop

expected.sort\_index(inplace=True) expected`

|  | **auto** | **manual** |
| --- | --- | --- |
| 4 | 0.295328 | 0.144843 |
| f | 0.303930 | 0.149061 |
| r | 0.071682 | 0.035156 |

If we wanted to convert these proportions to expected number of values, we can multiply by the total number of observations:

expected \*= n expected

|  | **auto** | **manual** |
| --- | --- | --- |
| 4 | 69.106838 | 33.893162 |
| f | 71.119658 | 34.880342 |
| r | 16.773504 | 8.226496 |

**Observed Values**

Now we have the expected proportions, we need to calculate the actual proportions so that we can compare them. To do this, we'll use the crosstab function from pandas.

observed = pd.crosstab(mpg.drv, mpg.transmission) observed

| **transmission** | **auto** | **manual** |
| --- | --- | --- |
| drv |  |  |
| 4 | 75 | 28 |
| f | 65 | 41 |
| r | 17 | 8 |

**Calculate Chi-Square**

Now we can calculate our test statistic, $\chi^2$

chi2 = ((observed - expected)\*\*2 / expected).values.sum() chi2

3.136769245971112

We also need to find our degrees of freedom for the distribution. The degrees of freedom are given by:

(nrows−1)×(ncols−1)(nrows−1)×(ncols−1)

Where nrows and ncols are the number of rows and columns in our contingency table.

`nrows, ncols = observed.shape

degrees\_of\_freedom = (nrows - 1) \* (ncols - 1)`

Now, based on the test statistic and degrees of freedom, we could lookup the corresponding p-value from a pre-calculated table, or use scipy's chi2 distribution.

stats.chi2(degrees\_of\_freedom).sf(chi2)

0.20838152534979645

With this high of a p-value, we fail to reject our null hypothesis.

**Manual Calculation, Example 2**

**Observed Values**

Suppose we have the following contingency table:

|  | **Product A** | **Product B** |
| --- | --- | --- |
| Churn | 100 | 50 |
| No Churn | 120 | 28 |

And we want to know if a customer churning is independent of which product offering they have.

**Expected Values**

We have all the information that we need to run a $\chi^2$ test, because we can calculate the population proportions from the above table.

* + - Find the proportions for Product A, Product B, Churn, and No Churn

|  | **Product A** | **Product B** |  |
| --- | --- | --- | --- |
| Churn | 100 | 50 | 150 |
| No Churn | 120 | 28 | 148 |
|  | 220 | 78 | 298 |

* + - Calculate the proportions
      * Product A = 220 / 298 = .738
      * Product B = 78 / 298 = .262
      * Churn = 150 / 298 = .503
      * No churn = 148 / 298 = .497
    - Multiply these together to produce a contingency table of expected values

First we calculate proportions:

|  | **Product A** | **Product B** |
| --- | --- | --- |
| Churn | 0.372 | 0.132 |
| No Churn | 0.367 | 0.130 |

Then we can also see the actual expected number:

|  | **Product A** | **Product B** |
| --- | --- | --- |
| Churn | 110.7 | 39.3 |
| No Churn | 109.3 | 38.7 |

**Calculate Chi-Square**

* + - Calculate the test statistic and compute a p-value

`index = ['Churn', 'No Churn'] columns = ['Product A', 'Product B']

observed = pd.DataFrame([[100, 50], [120, 28]], index=index, columns=columns) n = observed.values.sum()

expected = pd.DataFrame([[.372, .132], [.367, .130]], index=index, columns=columns) \* n

chi2 = ((observed - expected)\*\*2 / expected).values.sum()

nrows, ncols = observed.shape

degrees\_of\_freedom = (nrows - 1) \* (ncols - 1)

p = stats.chi2(degrees\_of\_freedom).sf(chi2)

print('Observed') print(observed) print('---\nExpected') print(expected) print('---\n') print(f'chi^2 = {chi2:.4f}') print(f'p = {p:.4f}')`

Observed

Product A Product B

Churn 100 50

No Churn 120 28

---

Expected

Product A Product B

Churn 110.856 39.336

No Churn 109.366 38.740

---

chi^2 = 7.9656

p = 0.0048

* + Correlation

**Correlation**

Correlation tests are used to check if two samples are related. They are often used for feature selection and multivariate analysis in data preprocessing and exploration.

**Pearson's Correlation Coefficient**

The goal of this test is to answer the question: Do two continuous samples have a linear relationship?

For all hypothesis tests, we will follow the same process:

* + - Form hypotheses and set a desired confidence level
    - Calculate the appropriate test statistics and p-value
    - Conclude based on the above statistics

To answer this question, we will take the following steps:

* + - Calculate the Pearson correlation coefficient, $r\_{xy}$
    - Calculate the corresponding t-values
    - Test whether the t-values are significant or not

or in one step:

* + - Use stats.pearsonr

In this lesson, we will be looking at a dataset of student scores on an exam.

`import pandas as pd import numpy as np import matplotlib.pyplot as plt import seaborn as sns

from math import sqrt

from scipy import stats from pydataset import data

url = "<https://gist.githubusercontent.com/ryanorsinger/2c13a71421037af127e9fa7fa1463cad/raw/3eb443414078b51af33fdb2d211159e5f3e220ab/exam_scores.csv>" df = pd.read\_csv(url)

df.head(3)`

|  | **exam\_score** | **hours\_studied** | **study\_strategy** | **handedness** | **coffee\_consumed** | **hours\_slept** |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 100.591011 | 9.126291 | flashcards | left | 0 | 11 |
| 1 | 95.637086 | 9.677438 | flashcards | left | 1 | 10 |
| 2 | 53.200296 | 4.550207 | NaN | right | 5 | 6 |

sns.pairplot(df, corner=True) plt.suptitle("sns.pairplot visualizes continuous variable relationships") plt.show()

**Example**

Is there a correlation between the number of hours studied and exam score?

**1. Form hypotheses and set a desired confidence level**

When performing a correlation test, our null hypothesis is that there is no linear correlation between the two variables.

$H\_0$: There is no linear correlation between the number of hours studied and the score on the exam.

$H\_a$: There is a linear relationship between the number of hours studied and the score on the exam.

We will choose some values for our confidence interval and, based on that value, our alpha.

n = df.shape[0] # number of observations degf = n - 2 # degrees of freedom: the # of values in the final calculation of a statistic that are free to vary. conf\_interval = .95 # desired confidence interval α = 1 - conf\_interval

**2. Calculate the appropriate test statistics and p-value**

*Pearson Correlation Coefficient*

The correlation coefficient, $r\_{xy}$ is a unitless continuous numerical measure between -1 and 1, where 1 = perfect correlation and -1 = perfect negative correlation.

We will calculate the correlation between hours studied and exam score.

$$ r\_{xy} = \frac{\frac{1}{n}\sum(x\_i - \bar{x})(y\_i - \bar{y}) }{s\_x s\_y} $$

`x = df.hours\_studied y = df.exam\_score

def stdev(x): variance = ((x - x.mean()) \*\* 2).sum() / n return sqrt(variance)

r\_xy = (((x - x.mean()) \* (y - y.mean())).sum() / n) / (stdev(x) \* stdev(y)) r\_xy`

0.8351498542413306

*Calculate the corresponding p-value*

We can calculate a t-statistic for our correlation coefficient in order to inform us how likely it is that we observed this result due to chance. We will then use this t-statistic to find our p-value.

The t-value can be positive for positive correlations and negative for negative correlations, and is given by:

$$ t = \frac{r\_{xy}\sqrt{n-2}}{\sqrt{1-r^2\_{xy}}} $$

Where $n-2$ is our degrees of freedom.

t = (r\_xy \* sqrt(n - 2)) / sqrt(1 - r\_xy\*\*2) t

9.359998377263368

Once we have our t-statistic, we can find our p-value by looking up the t-statistic in a t-table, or by using scipy's t distribution:

p = stats.t.sf(t, df=degf) \* 2 # \*2 for a two-tailed test p

2.0762953315463266e-11

**3. Conclude based on the above statistics**

Lastly, we compare our p value to our alpha that we selected earlier:

p < α

True

Since p is less than alpha, we reject our null hypothesis that there is no linear correlation between the number of hours studied and exam score.

**The Easy Way**

All of the work that we did above is also provided by scipy's stats module using the pearsonr function.

**1. Form hypotheses and set a desired confidence level**

$H\_0$: There is no linear correlation between the number of hours studied and the score on the exam.

$H\_a$: There is a linear relationship between the number of hours studied and the score on the exam.

alpha = 0.05

**2. Calculate the appropriate test statistics and p-value**

corr, p = stats.pearsonr(x, y) corr, p

(0.8351498542413308, 2.0762953315462545e-11)

**3. Conclude based on the above statistics**

if p < alpha: print('We reject the null hypothesis') else: print('We fail to reject the null hypothesis')

We reject the null hypothesis

**Correlation Gotchas**

When working with correlation, keep in mind:

* + - Correlation is not causality.
    - Correlation measures *linear* relationship between the 2 variables. However, there may be other types of relationships, such as a quadratic or absolute value relationship.
    - Correlations can be misleading when confounding variables are ignored.
    - Correlation tells you nothing about how large the relationship is.

**Correlation is Not Causation**

Correlation means that two variables are associated, but doesn't tell us whether one causes the other or not.

**Nonlinear Relationship**

Here we'll look at an example of a nonlinear relationship:

x = np.linspace(-3, 3) y = x \*\* 2 plt.scatter(x, y, s=10, c='firebrick', alpha=.8);

We see that there is clearly a direct relationship between x and y, however it is not a linear relationship, so a correlation test will not give us significant results.

r, p = stats.pearsonr(x, y) print(f'r = {r:.5f}') print(f'p = {p:.1f}')

r = -0.00000

p = 1.0

**Confounding Variables**

We must be careful because correlation doesn't tell the whole story of a dataset. That is, correlation just looks at two variables in isolation, and doesn't account for any others. For example, a certain subgroup could have a strong correlation while another does not, or a third variable could be influencing both of the variables.

In our exam score data, if we look at coffee consumption and exam score, we see that they are strongly negatively correlated:

`r, p = stats.pearsonr(df.coffee\_consumed, df.exam\_score)

plt.scatter(df.coffee\_consumed, df.exam\_score) plt.title('Exam Score vs # Cups of Coffee') plt.text(8, 80, f'r = {r:.3f}') plt.show()`

However, it is probably the case that a third variable here has more influence, the number of hours that each student slept before the exam. We know that getting a good night's sleep improves exam scores, and if you don't get a good night's sleep, you are probably likely to drink more coffee.

`r\_sleep\_coffee, \_ = stats.pearsonr(df.hours\_slept, df.coffee\_consumed) r\_sleep\_score, \_ = stats.pearsonr(df.hours\_slept, df.exam\_score)

print('Correlation between hours slept and coffee consumed') print(f' r = {r\_sleep\_coffee:.3f}') print('Correlation between hours slept and exam score') print(f' r = {r\_sleep\_score:.3f}')`

Correlation between hours slept and coffee consumed

r = -1.000

Correlation between hours slept and exam score

r = 0.994

**Scale of the Relationship**

The correlation coefficient shows us how linearly correlated two variables are, but doesn't tell us the scale. That is one variable could increase/decrease in lock step with another, but the size of the change could be tiny.

Imagine our data was a little different and we tracked several student's coffee intake before an exam. The coffee consumption could be perfectly correlated with exam score, but to such a small amount that it doesn't matter.

`cups\_of\_coffee = [0, 1, 2, 3, 4] exam\_score = [80, 80.1, 80.2, 80.3, 80.4] r, p = stats.pearsonr(cups\_of\_coffee, exam\_score)

plt.plot(cups\_of\_coffee, exam\_score) plt.title('Exam Score vs Coffee Consumption') plt.ylim(0, 100) plt.xticks(range(5)) plt.xlabel('Coffee Consumed (cups)') plt.ylabel('Exam Score')

plt.text(1, 70, f'r = {r:.2f}') plt.text(1, 63, f'p = {p:e}') plt.show()`

**What about nonlinear correlations between variables?**

* + - Use [Spearman's R](https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.spearmanr.html), for a nonparametric test of linearity, as long as we have monotonicity.

**Further Reading**

* + - [Examples of Spurious Correlations](http://www.tylervigen.com/spurious-correlations)
    - [The Harvard Business Review: Beware Spurious Correlations](https://hbr.org/2015/06/beware-spurious-correlations)

**Exercises**

Continue working in your hypothesis\_testing notebook.

* + - Answer with the type of stats test you would use (assume normal distribution):
      * Is there a relationship between the length of your arm and the length of your foot?
      * Does smoking affect when or not someone has lung cancer?
      * Is gender independent of a person’s blood type?
      * Does whether or not a person has a cat or dog affect whether they live in an apartment?
      * Does the length of time of the lecture correlate with a student's grade?
    - Use the telco\_churn data.
      * Does tenure correlate with monthly charges?
      * Total charges?
      * What happens if you control for phone and internet service?
    - Use the employees database.
      * Is there a relationship between how long an employee has been with the company and their salary?
      * Is there a relationship between how long an employee has been with the company and the number of titles they have had?
    - Use the sleepstudy data.
      * Is there a relationship between days and reaction time?
  + Comparison of Means

**Comparing Means**

We can compare two groups to see if they differ in some measure (represented by a continuous variable) significantly by comparing each group's mean value of that continuous variable. In other words, we compare subgroups of a categorical value by comparing the mean of a continuous variable across each of those subgroups.

The statistical question you are asking: Are these two groups from the same population?

A test comparing means can help us answer questions like:

* + - Are the salaries of the marketing department higher than the company average?
    - Do customers who receive marketing emails spend more money than customers who do not receive marketing emails?
    - Are sales for product A higher when we run a promotion for it than when we do not run a promotion for it?

For a comparing means test, there are different options based on your data. In this lesson we will explore both parametric and non-parametric tests.

**Parametric Vs. Non-Parametric**

The difference between **Parametric** and **Non-Parametric** tests is that parametric tests rely on a distribution.

There are other assumptions about the data made when running these tests, but let's talk about this assumption of normality first.If your sample is > 500, you don't need to worry about normality assumption due to the **central limit theorem**. If your sample is < 30 and not at all normal, then use a non-parametric test. In between, "it depends". The smaller your sample, the more normal your distribution needs to be.

But what is the Central Limit Theorem and why does it matter?

**Central Limit Theorem**

The central limit theorem tells us that the **sampling distribution** for a random variable is normally distributed, even if the underlying random variable is not. In other words, the distribution of the average values of many samples taken of that one sample, is normal.

Let's take the example of rolling a dice. We know that rolling a die is a random process, but imagine an experiment where we roll 10 dice and take the average roll. If we performed this experiment many times and plotted the resulting calculated averages of 10 dice rolls, we would expect to see a normal distribution.

`import matplotlib.pyplot as plt import numpy as np np.random.seed(123)

n\_dice\_per\_experiment = ncols = 10 n\_experiments = nrows = 100

data = np.random.randint(1, 7, (nrows, ncols))

data[:4]`

array([[6, 3, 5, 3, 2, 4, 3, 4, 2, 2],

[1, 2, 2, 1, 1, 2, 4, 6, 5, 1],

[1, 5, 2, 4, 3, 5, 3, 5, 1, 6],

[1, 2, 4, 5, 5, 5, 2, 6, 4, 3]])

calculated\_averages = data.mean(axis=1) calculated\_averages

array([3.4, 2.5, 3.5, 3.7, 3.2, 4.7, 3.4, 3.2, 3.4, 3. , 3.7, 3.5, 4. ,

4.5, 2.9, 2.8, 2.9, 3.8, 3.2, 3.6, 3.4, 4. , 3.5, 3.3, 3.6, 3. ,

4.3, 3. , 3.3, 3.2, 4.2, 3.9, 3.2, 3.8, 3.6, 3.4, 2.7, 3.4, 3.2,

3.1, 3.2, 3. , 4.1, 3.3, 3. , 4.1, 3.4, 3.3, 2.9, 3.8, 3.4, 3.6,

3.9, 3.2, 4.7, 3.8, 3.5, 2.8, 3.8, 4.7, 4.1, 3.5, 2.8, 3.7, 3.4,

3.7, 3.6, 4.1, 3.6, 3.5, 2.9, 4. , 3.2, 3.4, 4.4, 2.9, 3.2, 3.9,

1.9, 3. , 3. , 3.5, 3.3, 4.4, 3.4, 4.2, 4.1, 3.2, 3.5, 3.5, 3.8,

3.1, 3.6, 2.9, 2.9, 4. , 3.3, 3.1, 3.4, 3.6])

plt.hist(calculated\_averages) plt.xlabel(f'Average of {n\_dice\_per\_experiment} dice rolls') plt.ylabel('# of Occurrences') plt.title(f'Outcome of averaging {n\_dice\_per\_experiment} dice rolls {n\_experiments} times') plt.show()

This concept allows us to make calculations using the normal distribution based on the values we calculate from our samples.

**Comparing Tests that Compare Means**

| **Goal** | **$H\_0$** | **Data Needed** | **Parametric Test** | **Assumptions\*** | **Non-parametric Test** |
| --- | --- | --- | --- | --- | --- |
| Compare observed mean to theoretical one | $\mu\_{obs}=\mu\_{th}$ | array-like of observed values & float of theoretical | 1 sample t-test (stats.ttest\_1samp) | Normally Distributed\*\* | 1 sample Wilcoxon signed rank test (stats.wilcoxon) |
| Compare two observed means (independent samples) | $\mu\_{a}=\mu\_{b}$ | 2 array-like samples | 2 sample t-test (Independent t-test)(stats.ttest\_ind) | Independent, Normally Distributed\*\*, Equal Variances\*\*\* | Mann-Whitney's test (stats.mannwhitneyu) |
| Compare several observed means (independent samples) | $\mu\_{a}=\mu\_{b}=\mu\_{n}$ | n array-like samples | ANOVA (stats.f\_oneway) | Independent, Normally Distributed\*\*, Equal Variances | Kruskal-Wallis test (stats.kruskal) |

* + - If assumptions can't be met, the equivalent non-parametric test can be used.
    - \*Normal Distribution assumption can be met by having a large enough sample (due to Central Limit Theorem), or the data can be scaled using a Gaussian Scalar.
    - \*\*The argument in the stats.ttest\_ind() method of equal\_var can be set to False to accommodate this assumption.

All the comparing means tests will follow the regular process with an additional step:

* + - Form hypotheses and set a desired confidence level
    - **Verify assumptions**
    - Calculate the appropriate test statistics and p-value
    - Conclude based on the above statistics

When concluding based on our p-values, we have to take into account whether it is a 1-tail or 2-tail test (different from 1 sample or 2 sample t-test). A 1-tailed test means our alternative hypothesis is checking if one subgroup is larger than the other OR our alternative hypothesis is checking if one subgroup is smaller than the other. A 2-tail test is checking if there is any difference in subgroups, regardless of direction.

Our null hypothesis corresponds to a t-score of 0 (if there is no difference in the means, the numerator in our t calculation would come out to 0). If the null hypothesis is true, our t-scores will follow a normal distribution and will be centered around 0. That is, if the null hypothesis is true, and we ran our experiment many many times, we would expect to get slightly different t-statistics each time. If we plotted the resulting t-statistics we would expect to see an approximately normal curve.

Therefore, for a 2-tailed test, we take the p-value as is and compare it to alpha as normal. For a 1-tailed test, we divide our p-value in half (since it is only looking in one direction) **AND** compare our t-statistic to zero (greater than zero to test if it is higher or less than zero to test if it is lower).

**One Sample T-Test**

Goal: Compare observed mean to theoretical one.

* + - Form hypotheses and set a desired confidence level

|  |  |  |
| --- | --- | --- |
| Null Hypothesis | $H\_0$ | $\mu\_{obs}=\mu\_{th}$ |
| Alternative Hypothesis (2-tail, significantly different) | $H\_a$ | $\mu\_{obs}!=\mu\_{th}$ |
| Alternative Hypothesis (1-tail, significantly smaller) | $H\_a$ | $\mu\_{obs} < \mu\_{th}$ |
| Alternative Hypothesis (1-tail, significantly larger) | $H\_a$ | $\mu\_{obs} > \mu\_{th}$ |

* + - Verify Assumptions
      * Normal Distribution
        + Or at least 500 observations (CLT)
        + Or at least 30 observations and "kinda" normal
    - Calculate the appropriate test statistics and p-value
      * scipy.stats.ttest\_1samp
        + inputs

an array of all observed values of the subgroup

the population mean

* + - * + outputs

t-statistic

p-value

* + - Conclude based on the above statistics
      * For 2-tail, p compared to $\alpha$
      * For 1-tail higher than, $p/2$ compared to $\alpha$ and $t > 0$
      * For 1-tail lower than, $p/2$ compared to $\alpha$ and $t < 0$

We will use the telco\_churn database to work through examples.

`import pandas as pd import matplotlib.pyplot as plt import seaborn as sns import scipy.stats as stats import env

db\_url = f'mysql+pymysql://{env.user}:{env.password}@{env.host}/telco\_churn'  
df = pd.read\_sql('SELECT \* FROM customers', db\_url)`

df.columns

Index(['customer\_id', 'gender', 'senior\_citizen', 'partner', 'dependents',

'tenure', 'phone\_service', 'multiple\_lines', 'internet\_service\_type\_id',

'online\_security', 'online\_backup', 'device\_protection', 'tech\_support',

'streaming\_tv', 'streaming\_movies', 'contract\_type\_id',

'paperless\_billing', 'payment\_type\_id', 'monthly\_charges',

'total\_charges', 'churn'],

dtype='object')

I believe customers who churn are charged more (monthly average) than the overall average monthly charges.

**Example 1**

Is the mean of monthly charges of customers who churn significantly higher than the mean across all customers?

We will use a 1 sample (comparing 1 group to the average), 1-tailed ("significantly higher") t-test.

**1. Form hypotheses and set a desired confidence level**

$H\_0$: Mean of monthly charges of churned customers <= Mean of monthly charges of all customers

$H\_a$: Mean of monthly charges of churned customers > Mean of monthly charges of all customers

alpha = .05

**2. Verify assumptions**

df.churn.value\_counts()

No 5174

Yes 1869

Name: churn, dtype: int64

Our churn sample is large enough, as is overall count, to meet the assumptions of normal distributions.

**3. Calculate the appropriate test statistics and p-value**

* + - scipy.stats.ttest\_1samp

`# an array of all observed values of the subgroup churn\_sample = df[df.churn == 'Yes'].monthly\_charges

**the population mean**

overall\_mean = df.monthly\_charges.mean()

t, p = stats.ttest\_1samp(churn\_sample, overall\_mean)

print(t, p/2, alpha)`

16.96540308050567 1.8703196496911995e-60 0.05

**4. Conclude based on the above statistics**

For a 1-tailed test where our alternative hypothesis is testing for "greater than", we evaluate $𝑝/2 < \alpha$ and $𝑡 > 0$.

if p/2 > alpha: print("We fail to reject the null hypothesis.") elif t < 0: print("We fail to reject null hypothesis.") else: print("We reject the null hypothesis.")

We reject the null hypothesis.

**Example 2**

Is there any difference in monthly charges for customers who churned and customers who don't churn?

We will use a 1 sample (comparing 1 group to the average), 2-tailed (significantly different) t-test.

**1. Form hypotheses and set a desired confidence level**

$H\_0$: Mean of monthly charges of churned customers = Mean of monthly charges of all customers

$H\_a$: Mean of monthly charges of churned customers != Mean of monthly charges of all customers

$\alpha$: See above

**2. Verify assumptions**

See above

**3. Calculate the appropriate test statistics and p-value**

t, p = stats.ttest\_1samp(churn\_sample, overall\_mean) t, p

(16.96540308050567, 3.740639299382399e-60)

**4. Conclude based on the above statistics**

Since we are looking at any difference in monthly charges (regardless of higher or lower), we will use a 2-tail test. Is p-value less than alpha?

p < alpha

True

We reject the null hypothesis.

**Example 3**

The mean monthly charges of customers who have turned is less than the mean monthly charges of all customers.

We will use a 1 sample (comparing 1 group to the average), 1-tailed (significantly lower) t-test.

**1. Form hypotheses and set a desired confidence level**

$H\_0$: Mean of monthly charges of churned customers >= Mean of monthly charges of all customers

$H\_a$: Mean of monthly charges of churned customers < Mean of monthly charges of all customers

$\alpha$: See above

**2. Verify assumptions**

See above

**3. Calculate the appropriate test statistics and p-value**

t, p = stats.ttest\_1samp(churn\_sample, overall\_mean) t, p

(16.96540308050567, 3.740639299382399e-60)

**4. Conclude based on the above statistics**

Is 1/2 of p-value < alpha AND t-stat < 0?

print(p/2 < alpha) print(t < 0)

True

False

We fail to reject the null hypothesis.

**Independent T-Test (a.k.a. Two Sample T-Test)**

Goal: Compare mean of group a to mean of group b.

* + - Form hypotheses and set a desired confidence level

|  |  |  |
| --- | --- | --- |
| Null Hypothesis | $H\_0$ | $\mu\_a == \mu\_b$ |
| Alternative Hypothesis (2-tail, significantly different) | $H\_a$ | $\mu\_a! == \mu\_b$ |
| Alternative Hypothesis (1-tail, a is significantly smaller than b) | $H\_a$ | $\mu\_a < \mu\_b$ |
| Alternative Hypothesis (1-tail, a is significantly larger than b) | $H\_a$ | $\mu\_a > \mu\_b$ |

* + - Verify Assumptions:
      * Normal Distribution
        + Or at least 500 observations (CLT)
        + Or at least 30 observations and "kinda" normal
      * Independent samples
      * Equal Variances (or set method argument to False when not)
    - Calculate the appropriate test statistics and p-value
      * scipy.stats.ttest\_ind
        + inputs

an array of observed values from one subgroup

an array of observed values from the other subgroup

* + - * + outputs

t-statistic

p-value

* + - Conclude based on the above statistics
      * For 2-tail, p compared to $\alpha$
      * For 1-tail higher than, $p/2$ compared to $\alpha$ and $t > 0$
      * For 1-tail lower than, $p/2$ compared to $\alpha$ and $t < 0$

**Example 1**

I believe customers who churn are charged more (monthly average) than customers who don't churn. Is the mean of monthly charges of customers who churn significantly higher than the mean of those who don't churn?

We will use a 2 sample (comparing 2 groups to each other), 1-tailed (significantly higher) t-test.

#an array for each subgroup churn\_sample = df[df.churn == 'Yes'].monthly\_charges no\_churn\_sample = df[df.churn == 'No'].monthly\_charges

**1. Form hypotheses and set a desired confidence level**

$H\_0$: Mean of monthly charges of churned customers <= Mean of monthly charges of customers who haven't churned

$H\_a$: Mean of monthly charges of churned customers > Mean of monthly charges of customers who haven't churned

alpha = .05

**2. Verify Assumptions**

* + - Independent Samples?
      * YES! No observations in the churn sample exist in the no-churn sample.
    - Normal Distribution?
      * YES! Plenty of observations
    - Equal Variances (the scipy methods we will use has an argument to handle when variances aren't equal)?

print(churn\_sample.var()) print(no\_churn\_sample.var())

608.4141833954315

966.7527670734293

* + - NO! So we will set the argument of equal\_var to False.

**3. Calculate the appropriate test statistics and p-value**

t, p = stats.ttest\_ind(churn\_sample, no\_churn\_sample, equal\_var=False) t, p

(18.407526676414673, 8.59244933154705e-73)

**4. Conclude based on the above statistics**

print("is p/2 < alpha? ", p / 2 < alpha) print("is t > 0? ", t > 0)

is p/2 < alpha? True

is t > 0? True

if p / 2 > alpha: print("We fail to reject the null hypothesis") elif t < 0: print("We fail to reject the null hypothesis") else: print("We reject the null hypothesis")

We reject the null hypothesis

**Example 2**

Are charges of customers who churn *significantly different* than those who do not churn?

We will use a 2 sample (comparing 2 groups to each other), 2-tailed (significantly different) t-test.

**1. Form hypotheses and set a desired confidence level**

$H\_0$: charges of customers who churn equals that of those who don't churn.

$H\_a$: charges of customers who churn is not equal to that of those who don't churn.

alpha = See above

**2. Verify assumptions**

See above

**3. Calculate the appropriate test statistics and p-value**

t, p = stats.ttest\_ind(churn\_sample, no\_churn\_sample, equal\_var=False) t, p

(18.407526676414673, 8.59244933154705e-73)

**4. Conclude based on the above statistics**

Is the p-value less than alpha?

print("Reject null hypothesis? ", p < alpha)

Reject null hypothesis? True

**Example 3**

Are charges of customers who churn *significantly less* than those who do not churn?

We will use a 2 sample (comparing 2 groups to each other), 1-tailed (significantly lower) t-test.

**1. Form hypotheses and set a desired confidence level**

$H\_0$: charges of customers who churn equals or greater than that of those who don't churn.

$H\_a$: charges of customers who churn is less than that of those who don't churn.

alpha: See above

**2. Verify assumptions**

See above

**3. Calculate the appropriate test statistics and p-value**

t, p = stats.ttest\_ind(churn\_sample, no\_churn\_sample, equal\_var=False) t, p

(18.407526676414673, 8.59244933154705e-73)

**4. Conclude based on the above statistics**

Is p/2 < alpha AND is t < 0?

print("Is p/2 < alpha? ", p / 2 < alpha) print("Is t < 0? ", t < 0)

Is p/2 < alpha? True

Is t < 0? False

We fail to reject the null hypothesis.

**ANOVA Analysis of Variance**

Goal: Compare means of groups a, b & c.

* + - Form hypotheses and set a desired confidence level

|  |  |  |
| --- | --- | --- |
| Null Hypothesis | $H\_0$ | $\mu\_a == \mu\_b == \mu\_c$ |
| Alternative Hypothesis (significantly different) | $H\_a$ | $\mu\_a! = \mu\_b! = \mu\_c$ |

* + - Verify Assumptions:
      * Normal Distribution
        + Or at least 500 samples (CLT)
        + Or at least 30 observations and "kinda" normal
      * Independent samples
      * Equal Variances
    - Calculate the appropriate test statistics and p-value
      * scipy.stats.f\_oneway
        + inputs

an array of observed values for every subgroup

* + - * + outputs

f-statistic

p-value

* + - Conclude based on the above statistics

**Example 1**

Is the sepal length significantly different across the different species of iris?

Since there are three different species of iris, there are three subgroups that we are comparing and will use the ANOVA test.

df = sns.load\_dataset('iris') df.species.value\_counts()

setosa 50

versicolor 50

virginica 50

Name: species, dtype: int64

df.sepal\_length.describe()

count 150.000000

mean 5.843333

std 0.828066

min 4.300000

25% 5.100000

50% 5.800000

75% 6.400000

max 7.900000

Name: sepal\_length, dtype: float64

**1. Form hypotheses and set a desired confidence level**

$H\_0$: population means of the sepal length for the three species, versicolor, virginica & setosa, are all equal.

$H\_a$: population means of the sepal length for the three species, versicolor, virginica & setosa, are NOT all equal.

alpha = .05

**2. Verify assumptions**

versicolor\_sepal\_length = df[df.species == 'versicolor'].sepal\_length virginica\_sepal\_length = df[df.species == 'virginica'].sepal\_length setosa\_sepal\_length = df[df.species == 'setosa'].sepal\_length

Independent samples? YES!

Normal Distribution?

versicolor\_sepal\_length.plot.hist();

virginica\_sepal\_length.plot.hist();

setosa\_sepal\_length.plot.hist();

* + - YES! The distributions are mostly normal

Equal Variances?

print(versicolor\_sepal\_length.var()) print(virginica\_sepal\_length.var()) print(setosa\_sepal\_length.var())

0.2664326530612246

0.40434285714285706

0.12424897959183666

* + - YES! The variance is very small so the differences are minor.

**3. Calculate the appropriate test statistics and p-value**

f, p = stats.f\_oneway(versicolor\_sepal\_length, virginica\_sepal\_length, setosa\_sepal\_length) f, p

(119.26450218450472, 1.6696691907693648e-31)

**4. Conclude based on the above statistics**

if p < alpha: print("We reject the null hypothesis") else: print("We fail to reject the null hypothesis")

We reject the null hypothesis

**Example 2**

Is the horsepower of a car different in cars from different origins?

df = sns.load\_dataset('mpg') df.origin.value\_counts()

usa 249

japan 79

europe 70

Name: origin, dtype: int64

df.head() df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 398 entries, 0 to 397

Data columns (total 9 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 mpg 398 non-null float64

1 cylinders 398 non-null int64

2 displacement 398 non-null float64

3 horsepower 392 non-null float64

4 weight 398 non-null int64

5 acceleration 398 non-null float64

6 model\_year 398 non-null int64

7 origin 398 non-null object

8 name 398 non-null object

dtypes: float64(4), int64(3), object(2)

memory usage: 28.1+ KB

Drop nulls

df = df[~df['horsepower'].isna()]

usa\_hp = df[df.origin == 'usa'].horsepower japan\_hp = df[df.origin == 'japan'].horsepower eu\_hp = df[df.origin == 'europe'].horsepower

**1. Form hypotheses and set a desired confidence level**

$H\_0$: hp is the same across all origins

$H\_a$: hp is not the same across all origins

$\alpha$ is already set to .05

**2. Verify assumptions**

Independent: YES!

Normal: YES!

usa\_hp.hist();

japan\_hp.hist();

eu\_hp.hist();

Equal Variance: no

usa\_hp.var()

1591.8336567413864

japan\_hp.var()

317.5238558909445

eu\_hp.var()

406.3397717295875

**3. Calculate the appropriate test statistics and p-value**

stats.kruskal(usa\_hp, japan\_hp, eu\_hp)

KruskalResult(statistic=105.59475799843663, pvalue=1.1759521262123952e-23)

**4. Conclude based on the above statistics**

if p < alpha: print('We reject the null hypothesis') else: print('We fail to reject the null hypothesis')

We reject the null hypothesis

Using Kruskal-Wallis test, non-parametric test for ANOVA, shows us that the mean HP of the cars from the 3 origins is significantly different.

**Exercises**

Continue working in your hypothesis\_testing notebook.

* + - Answer with the type of test you would use (assume normal distribution):
      * Is there a difference in grades of students on the second floor compared to grades of all students?
      * Are adults who drink milk taller than adults who don't drink milk?
      * Is the price of gas higher in Texas or in New Mexico?
      * Are there differences in stress levels between students who take data science vs students who take web development vs students who take cloud academy?
    - Ace Realty wants to determine whether the average time it takes to sell homes is different for its two offices. A sample of 40 sales from office #1 revealed a mean of 90 days and a standard deviation of 15 days. A sample of 50 sales from office #2 revealed a mean of 100 days and a standard deviation of 20 days. Use a .05 level of significance.
    - Load the mpg dataset and use it to answer the following questions:
      * Is there a difference in fuel-efficiency in cars from 2008 vs 1999?
      * Are compact cars more fuel-efficient than the average car?
      * Do manual cars get better gas mileage than automatic cars?
  + More Examples

**Statistical Testing Examples in Python**

In this lesson, we'll take a look at some examples of using statistical testing to analyze a dataset.

We'll be using the tips dataset, which contains information about each tip a waiter received while working in a restauraunt.

We'll focus on three tests for these specific purposes:

* + - **chi2**: to compare two categorical variables
    - **pearson r**: to compare two continuous variables
    - **t-test**: to compare one categorical and one continuous variable

Before we begin we will load in the data set and take a quick glance at it.

`%matplotlib inline import pandas as pd from scipy import stats from pydataset import data import viz

tips = data('tips')`

tips.shape

(244, 7)

tips.head()

|  | **total\_bill** | **tip** | **sex** | **smoker** | **day** | **time** | **size** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 16.99 | 1.01 | Female | No | Sun | Dinner | 2 |
| 2 | 10.34 | 1.66 | Male | No | Sun | Dinner | 3 |
| 3 | 21.01 | 3.50 | Male | No | Sun | Dinner | 3 |
| 4 | 23.68 | 3.31 | Male | No | Sun | Dinner | 2 |
| 5 | 24.59 | 3.61 | Female | No | Sun | Dinner | 4 |

**chi2**

We use a chi2 test to compare two categorical variables. For this example, we will compare the sex variable with the smoker column. Our null hypothesis is that membership in these groups is independent, more formally:

$H\_0$: sex is indep of whether or not someone is a smoker

First we need to generate a *contingency table*, which is another word for a cross tabulation, and can easily be generated with pandas.

contingency\_table = pd.crosstab(tips.sex, tips.smoker) contingency\_table

| **smoker** | **No** | **Yes** |
| --- | --- | --- |
| sex |  |  |
| Female | 54 | 33 |
| Male | 97 | 60 |

The way the chi2 test works is to compare the actual contingency table of the actual values against the table that we would predict to be the case if group membership is independent. When we perform the test, one of the returned values will be the expected values in the contingency table.

To perform the test, we simply pass the contingency table that we created with pandas to the chi2\_contingency function from scipy.

test\_results = stats.chi2\_contingency(contingency\_table) test\_results

(0.0,

1.0,

1,

array([[53.84016393, 33.15983607],

[97.15983607, 59.84016393]]))

The function returns several values:

* + - the chi2 test statistic
    - the p value
    - the degrees of freedom
    - the matrix of expected values

We'll focus in on the p value and the matrix of expected values:

\_, p, \_, expected = test\_results

Now we can look at p to decide whether to reject / fail to reject H0.

p

1.0

With such a high p-value, we fail to reject the null hypothesis.

Less formally, it seems as though two groups are independent of each other. We can see an intuitive proof of this by comparing the expected values agains what we actually observed:

`# Here we'll do some data frame manipulation with pandas to get the two tables

**into a more comparable form**

expected = pd.DataFrame(expected, index=['Female', 'Male'], columns=['Non-Smoker', 'Smoker'])

contingency\_table.columns = ['Non-Smoker', 'Smoker'] contingency\_table.index.name = ''

contingency\_table['group'] = 'Actual' expected['group'] = 'Expected'

(pd.concat([contingency\_table, expected]) .reset\_index() .rename({'index': 'sex'}, axis=1) .set\_index(['group', 'sex']))`

|  |  | **Non-Smoker** | **Smoker** |
| --- | --- | --- | --- |
| group | sex |  |  |
| Actual | Female | 54.000000 | 33.000000 |
|  | Male | 97.000000 | 60.000000 |
| Expected | Female | 53.840164 | 33.159836 |
|  | Male | 97.159836 | 59.840164 |

The table above shows us that the actual values are very close to the expected values, thus our failure to reject the null hypothesis.

**Pearson R**

We can use the pearson r test to compare two continuous variables and see if they are linearly correlated, and the strength of the correlation. We will use the only two continuous variables from our dataset, total\_bill and tip. Our null hypothesis for this test is that there is no correlation between the two variables, more formally:

$H\_0$: There is not linear correlation between the total bill and the tip amount.

To perform the test, we can pass the two Series that contain the values we are looking at to the pearsonr function form scipy's stats module.

test\_results = stats.pearsonr(tips.total\_bill, tips.tip) test\_results

(0.6757341092113641, 6.6924706468642374e-34)

This test gives us back two pieces of information:

* + - the test statistic, in this case, the r value
    - the p value

`r, p = test\_results

print(f'p is {p:.10f}')`

p is 0.0000000000

Since p is 0 within 10 decimal places, we can safely reject the null hypothesis of no linear correlation. Less formally, it seems as though the total bill and tip amount are very related.

The test statistic for this test is the r value, which tells us how strongly correlated the two variables are.

We can see this with a visualization:

viz.more\_examples\_example1()

**T Test**

We can use the t-test to compare a categorical feature with a continous feature.

We will focus on two kinds of t-tests:

* + - 1 sample: compares the mean for a subgroup against the population mean
    - 2 sample: compares the means for two subgroups

In both cases, our null hypothesis is the same: there is no difference in the means.

To demonstrate, we'll try to answer two different questions:

* + - Is the total bill amount different for smokers?
    - Is the size of the tip different for parties of 2 and parties of 4?

**1 Sample T Test**

Is the total bill amount different for smokers?

To answer this question, we need two pieces of information, which we will pass along to scipy:

* + - the total bill amounts for all the smokers
    - the overall total bill mean

We will feed both of these into the ttest\_1samp function from scipy's stats module.

Our null hypothesis is that there is no difference, more formally:

$H\_0$: The average bill for smokers is no different than the population mean.

`smokers\_total\_bills = tips[tips.smoker == 'Yes'].total\_bill overall\_total\_bill\_mean = tips.total\_bill.mean()

test\_results = stats.ttest\_1samp(smokers\_total\_bills, overall\_total\_bill\_mean) test\_results`

Ttest\_1sampResult(statistic=0.951796790928544, pvalue=0.3436939512284921)

This function gives us back two pieces of information, the test statstic and the p-value.

In this case, because our p-value is so high, we fail to reject our null hypothesis. Less formally, we conclude that smoker might not have a significant difference in their total bill.

Again a visualization can be helpful:

viz.more\_examples\_example2()

**2 Sample T Test**

Is the total bill amount different for parties of 2 vs 4?

$H\_0$: The average size of the tip left by parties of 2 and parties of 4 is the same.

For this example, we'll need to create two seperate datasets that contain the values for the continuous variable for each subgroup. In our case, this means we need all the tip values for parties of 2 and the tip values for parties of 4.

We'll pass these to the ttest\_ind function from scipy's stats module.

parties\_of\_2 = tips[tips['size'] == 2] parties\_of\_4 = tips[tips['size'] == 4] test\_results = stats.ttest\_ind(parties\_of\_2.tip, parties\_of\_4.tip) test\_results

Ttest\_indResult(statistic=-7.462130391296251, pvalue=2.924028981378475e-12)

Like before, the function returns the test statistic and the p-value. Here, with such a small p-value, we reject the null hypothesis. We think there is a significant difference in the average tip amount left by parties of 2 and parties of 4.

Let's visualize this as well:

viz.more\_examples\_example3()

**Summary of Hypothesis Testing and Distributions**

**Exercise**

Continue working in your hypothesis\_testing notebook.

Choose several continous and categorical variables that were not covered in the lesson and perform each type of test on them. You may use another data set if you wish.

**Tableau**

* Overview

**Introduction to Tableau**

**Tableau Products**

* 1. [Tableau Public](https://public.tableau.com/en-us/s/): Tableau Public is completely free to use. Data can be downloaded by anyone, so do NOT use confidential data. On the same token, it's a good place to find data to play with. In fact, all of the details behind the public visualizations you see can be accessed by anyone. If you want help getting started on a project: find something you like, open it in your Tableau instance, and modify it to make it your own. It's a great way to learn Tableau. Limitations of Tableau Public are that your workbooks cannot be saved locally, there are fewer ways to connect data, and you are limited to 10 million rows per workbook.
  2. [Tableau Desktop](https://www.tableau.com/products/desktop): Tableau Desktop is used to create dashboards or stories, like Tableau Public. It connects to data sources directly and workbooks can be saved locally. It also is not free... or cheap.
  3. [Tableau Server](https://www.tableau.com/products/server): Tableau Server is used when a company wants to publish data and dashboards internally. The dashboards are created using Tableau Desktop and pushed to Tableau Server, where they are shared across the internal network. The server must be dedicated to Tableau.
  4. [Tableau Cloud](https://www.tableau.com/products/cloud-bi):Tableau Cloud is a mix of Tableau Server and Tableau Public. Like Tableau Public, the server is hosted by Tableau. Unlike Tableau Public, your data and visualizations are private unless you specify otherwise. Like Tableau Server, you would build visualizations on Tableau Desktop and deploy them to a server.
  5. [Tableau Prep](https://www.tableau.com/products/prep): Tableau Prep is a newer product. It is a GUI-based data wrangling application. It includes Tableau Prep Builder and Tableau Prep Conductor. The builder is for creating data flows. The conductor is for sharing flows and managing them across the organization. The ideal user for this software is a data analyst with limited programming skills.

**Tableau File Types**

* 1. .twb - Tableau Workbook: Stores visualization without source data.
  2. .tds - Tableau Data Source: Stores the server address, password, and other information required to access a datasource.
  3. .tbm - Tableau Bookmark: Stores a connection to a worksheet in another Tableau workbook.
  4. .tde - Tableau Data Extract: Stores Tableau data as a filtered and aggregated extract.
  5. .twbx - Tableau Packaged Workbook: Stores extracted data and visualizations for viewing in Tableau or Tableau Reader.

**Download Public Dashboards**

Tableau Public allows you to download it on your computer as Tableau Workbook (TWBX), PDF, Crosstab, Data and Image. To do so, hover over the download button at the bottom right corner and select file type. Not all workbooks allow for downloading as a Tableau workbook.

**Tableau Resources**

* 1. [Install Tableau Public](https://public.tableau.com/en-us/s/)
  2. [How-to Videos, Sample Data, & Live Training](https://public.tableau.com/en-us/s/resources)
  3. [Viz Gallery](https://public.tableau.com/en-us/s/gallery)
  4. [Leading Authors & Artists](https://public.tableau.com/en-us/s/authors#!/)

**Examples of Charts**

[Horizon Chart: Unemployment](https://public.tableau.com/profile/technical.product.marketing#!/vizhome/UnemploymentHorizionChart_1/HorizonChart)

[Positive + Negative Bar Chart: Gender Pay Gap](https://public.tableau.com/profile/iting#!/vizhome/170414_tax2014-15_genderpaygap/malejobs)

[Arrow Chart & Comet Chart: Sales](https://public.tableau.com/profile/tableaubims#!/vizhome/ArrowChartCometChart/Arrowchart)

**Examples of Dashboards**

[Dashboard: Twitter Analytics](https://public.tableau.com/en-us/s/gallery/adams-twitter-analytics?gallery=featured)

[Dashboard: Top Places to Retire](https://public.tableau.com/profile/zillow.real.estate.research#!/vizhome/TopPlacestoRetire/Dashboard1)

[How-to: Custom Filters](https://public.tableau.com/s/blog/2015/06/using-custom-shapes-dashboard-filters)

[How-to: Add Visual Elements](https://public.tableau.com/s/blog/2017/12/how-add-illustrations-your-dashboard-and-why-you-should-care)

**Examples of Stories**

[2018 World Series](https://public.tableau.com/en-us/s/gallery/journey-2018-world-series?gallery=votd)

[St. Mungo Hospital Annual Report](https://public.tableau.com/profile/sean.oslin#!/vizhome/StMungoHospitalAnnualReportPart1/Dashboard1)

[Mobile example: Which countries has the most opiod prescriptions in 2015?](https://public.tableau.com/profile/raycom.news.network#!/vizhome/Whichcountieshadthemostopioidprescriptionsin2015MobileVersion/Dashboard2)

[Deloitte Technology Fast 500](https://public.tableau.com/en-us/s/gallery/2018-deloitte-technology-fast-500?gallery=featured)

[Holiday Shopping](https://public.tableau.com/en-us/s/gallery/online-holiday-shopping?gallery=featured)

[IT Trends 2018](https://public.tableau.com/en-us/s/gallery/it-trends-2018?gallery=featured)

[A Data Viz Storyboard about Data Visualizations](https://public.tableau.com/profile/andy.kriebel#!/vizhome/VisualVocabulary/VisualVocabulary)

**Exercises**

* 1. [Create](https://public.tableau.com/app/discover) a Tableau Public account.
  2. [Download](https://www.tableau.com/products/public/download) Tableau Public.
  3. Log in to Tableau Public.
  4. [Explore](https://public.tableau.com/app/discover) public dashboards and download one.
* Connecting to Data

**Connecting to Data**

Tableau can connect to a broad range of data storage systems, softwares and file types. Tableau Public limits the connections, but the most common are available and should satisfy most of what you want to do. The first step in building a visualization in tableau is connecting to the data. Tableau offers flexibility in how you connect (maintain a live connection vs. create a data extract), in the data types that can be read, as well as in the merging of multiple data sources and datasets. Upon opening the Tableau app (and not a workbook), you will start on the connection page. We will spend this lesson here and on the *Data Source* tab, which you will see after completing a connection.

**Lesson Goals**

* 1. Know where to get help
  2. Connect to a data source
  3. Filter your data
  4. Prepare your data

**Primary Types of Data Sources**

**Local Files**

* 1. Excel: .xls, .xlsx
  2. Text: .csv, .txt, .tsv, .tab
  3. JSON Files: .json
  4. PDF Files: .pdf
  5. Spatial Files: Esri File Geodatabases (gdb\*.zip), Esri Shapefiles (.shp), MapInfo Tables (.tab), MapInfo Interchange Format (.mif), GeoJSON (.geojson), TopoJSON (.json, .topojson), KML (.kml)
  6. Statistical Files: SAS (.sas7bdat), SPSS (.sav), R (.rda, .rdata)

**Servers**

* 1. Google Sheets
  2. OData: mysql database server, e.g.
  3. Web data connector

**Other data connectors** (not available in Tableau Public)

* 1. [Other Data Sources](https://www.tableau.com/products/desktop?_ga=2.168784937.943005315.1582736711-305547270.1582736711&_fsi=g1y4KoAD#data-sources)

**Connection Method: Live vs. Data Extract**

"Tableau Data Extracts are snapshots of data optimized for aggregation and loaded into system memory to be quickly recalled for visualization. Extracts tend to be much faster than live connections, especially in more complex visualizations with large data sets, filters, calculations, etc." (Medrano, 2016). If data is being updated regularly, refreshing the extract is necessary in order to stay up to date with the most recent data.On the other hand, live connections will provide real-time updates and those changes will be reflected in your visualizations, with any changes in the data source reflected in Tableau. While extracts are always optimized for performance, the databases you may access may not be. That is to say that With live connections, your queries will only perform as well as the database can. Generally, you will be using extracts. When you publish a dashboard that needs real-time updates (which is the small minority of the time), you may decide to use a live connection. However, even in those cases, when building the visualizations on your desktop, you will likely use an extract. All of that said, with Tableau Public, you do not have the option to create a data extract.

**Connect to a File**

* 1. When connecting to a file, select the file type, navigate to the file you wish to connect to in your directory, and *open*.
  2. On the left side, you will see Connections, Files & New Union. Also, listed under *Files*, you will also see any other files in your current working directory that can be read by tableau.
  3. In the main section, you will see a sample (1,000 rows default) of the file you have connected to. Think about this sample size like running a SQL query that limits your results to 1,000. You can increase that number, and it will also increase the time required to load the data into view. You really just want a sample large enough to understand the data that is contained in each field. For this lesson, we will connect to cc\_institution\_details.csv.

**Join Files**

**Join Columns** You can connect and merge multiple data sources, such as multiple files or spreadsheets. Tableau joins are similar to joining tables in a mySQL database. You can do an inner, left, right, or full outer join. A box will display asking for the field(s) to join on. For this lesson, we will add a new connection to the file, merged\_2013\_PP.csv.

**Union Rows** You can append rows from 2 sources if your columns map to each other.

**Filter Data**

In this section, we will be focused on the filter option box. You can get there by looking in the top right of the *Data Source* page. There you can add a filter.

Let's work through an example. Say we want to add a new filter to keep only 4 year universities.

* 1. To do this, we will add a filter to the *Level* field.
  2. Filter (in top right) -> Add -> Add... -> Select field *Level* -> ...
  3. Set the criteria.

**General & Wildcard Filters**

There are many ways to identify the criteria for your filters. Under the **General** tab, you have ways to set criteria for exact matches to values in the *Level* field. Under the **Wildcard** tab, you can set criteria for wildcard string matching. We will look at the Condition tab in our next example. For now, let's filter to include only 4-year institutions.

**Conditional Filters**

For the next example, we will filter states. Add a new filter, and select *State* as the field. Under the **Condition** tab, let's add a condition 'By field'. Let's say we want to only include states which have a median *Student Count* of less than 1,000. We will select 'By field' and fill in the parameters to say *Student Count* Median << 1000. You should end with only Florida and Utah remaining. What you have done is filter out *States*. You have not filtered out schools that have a *Student Count* > 1000. To test this, complete your filter by selecting OK. You should have 206 rows remaining. Now, go back to your filters, edit the existing filter to not filter by a condition, but instead filter by name of state. Include only Florida and Utah. You will see you end up with the same number of observations.

**'Top' Filters**

Similar to Conditional filtering, you can filter the states by ranking in any of the columns. For example, say you want to explore colleges that are in the top 10 states in terms of *Median SAT Value*. You would filter *State* by going to the **Top** tab, and then filter Top 10 by *Median SAT Value* followed by your aggregate method, such as average. Note: Median SAT Value is the median SAT value for that school. When we are aggregating by average, we are taking the average of all of the colleges' *Median SAT Value* in that state. Then Tableau ranks those averages and returns the top 10 states.

**Prepare Data**

In this section, we will be focused on the table containing a sample view of our data. Each column header has 3 groups of activities contained in that tiny little box. The first is the icon in the top left of the column header representing the type of data. Second is the drop down arrow in the top right of the column header. Finally is the sort option in the bottom right of the column header box. The drop down menu arrow and the sort icon on the right side will not appear until you hover over the column header.

**Data Types**

For each column, there is an icon that represents the type of data it contains, or, more accurately, the type of data Tableau has concluded it contains. The options are: Number (decimal), Number (whole), Date & Time, Date, String, and Boolean. When you see a globe icon, this means the data type is a string or a number with a *Geographic Role* assigned to it. Possible geographic roles include airport, area code, city, country/region, county, state/province, zip code, latitude, and longitude. If the data type is a string, you will not see latitude or longitude as options under geographic role. To update the data type of a column, click the icon and select the new data type. The most common types that need correcting are ID columns that default to a number or a year that also defaults to a number. A good way to think about what data type a column should be is to consider how the column will be used in your visualization. If it might make sense to aggregate the field by summing or averaging, for example, then that field should be a number type. If the only way you would aggregate that column would be through a count, then it is likely that column should be in a string. If the column represents a date, such as year, then, obviously, change it to date format. What happens when you change a column *year* to a date data type?

**Other Column Options**

The quick menu in the top right of each column provides the following options...

* 1. **Rename**: Rename the column name
  2. **Reset Name**: Reset the name of the column to the name in the source data. Tableau renames the fields, upon import, to a more audience friendly name. Often, when building the visualizations, and especially calculated fields, it is easier to have names not separated by spaces. In these cases, it might be easier to revert back to the original names.
  3. **Copy Values**: By selecting the column and then 'copy values', you are copying the values in the column to the clipboard. It is the same as typing command-C.
  4. **Hide**: Hide will hide the column from view. You can still reference the column, such as in calculated fields.
  5. **Aliases** (strings only): Aliases are useful when you want a value in a field to be a more user-friendly value. For example, if *State* was in abbreviated form, but you want to make sure the entire state is spelled out on your visualizations, then aliasing would be useful. The alias does not change anything about the data in the background, only what is displayed.
  6. **Create Calculated Field...**: Create a new field based on existing fields. As an example, we will create a new field, award\_per\_natl\_delta, that is the difference between awards\_per\_natl\_value and awards\_per\_value. In the column *Awards Per Natl Value*, select Create Calculated Field from the drop down menu in the top right corner of the column header. Add the new column name, Awards Per Natl Delta. In the formula box, enter the formula: [Awards Per Value] - [Awards Per Natl Value] and click OK. You now have a new column where the data type icon, instead of '#' for number, it shows '=#' to represent a calculated number. You will also notice the lack of a blue bar bordering the top of the column header, indicating the column was not in the original data.
  7. **Create Group...**: Created Group is used for grouping categories into larger, higher hierarchical groups. For example, if I wanted to have a way to identify states by region, I could use the *State* column to create groups and then add each state to a group, such as south, northeast, southwest, northwest, and central. To do so, go to "Create Group" in the drop down menu for the *State* column. Begin selecting the states you wish to go in the first group. To select multiple states, hold the Command key down. Once your first group values are selected, click 'group'. You can then name the group. Move on to the next group. When you are done, title the new field and click ok. You can know a field is a result of grouping another field by the paperclip that has been added to the data type icon.
  8. **Split** (strings only): Split strings at a common delimiter into multiple columns. Let's take the field, *Counted Pct*, as an example. *Counted Pct* is defined as the "share entering undergraduate class who were first-time, full-time, degree-seeking students, meaning that they generally would be part of a tracked cohort of potential graduates. The entering class of 2007 is displayed for 4-year institutions; 2010 for 2-year institutions" (<https://data.world/databeats/college-completion/workspace/data-dictionary>). Currently, the data is in the format pct|yy, such at 54.8|10. 54.8% of those entering that 2-yr college's undergraduate class of 2010 were first-time, full-time, degree-seeking students. I would like to be able to use that percentage, so if I split the column into 2, then I can turn one into numeric and the other into date. I can then hide the original column.
  9. **Custom Split** (strings only): With custom split, you can specify the delimiter to split on. You can also specify how many columns to split off.
  10. **Create Bins...** (numbers only): Bins are useful when you have a continuous variable and you want to reduce the noise and bin the values close to each other together. For example, if you wanted to create a field that bins the number of students into equally sized groups, you could use "Create Bins" to do that.
  11. **Pivot** (when selecting multiple fields): Use as you would when creating a pivot table in excel or Python.
  12. **Merge Mismatched Fields** (when selecting multiple fields): Does what you would think! Use this when you want to merge fields into a single field, such as *City* and *State* into *City State*.
  13. **Describe**: Describe is a useful resource for a quick view into the column. The domain is not loaded by default. To see the domain of the field, click 'Load' in the Describe window. If you look at the description for *Awards Per Natl Delta*, a calculated field, you will see the formula that is used to calculate that field.

**Column Sorting**

Click the horizontal bar graph icon in the column header to sort that column. It will cycle through ascending, descending, and original sort as you click the icon. When you hover over the column header, you can tell if it is sorted by the way the horizontal bar graph image is sorted.

**Getting Help**

* 1. F1 or the Help menu header: you have options to get support, watch videos, see sample workbooks, sample gallery, as well as customizing settings & preferences.
  2. [onlinehelp.tableau.com](http://onlinehelp.tableau.com)

**Exercises**

* 1. In your mySQL client, query the database telco\_normalized, joining all tables together into a single table which you will then export to a csv and save on your local drive. Do NOT use the telco\_churn database.
  2. Connect to your csv through the tableau client that is installed on your laptop. Reminder: a csv is considered a text file.
  3. Hide any redundant columns (payment\_type\_id, contract\_type\_id, internet\_service\_type\_id are all represented through the columns with their descriptive names).
  4. Dimensions are something you could group by to see aggregated measures, like average payment by gender, or total charges by customer\_id. Measures are the fields you would perform calculations on. Ensure all possible measures are stored number datatype, and all dimensions are NOT stored as number data types. E.g. Senior Citizen will need to be changed to a string.
  5. Create aliases for values in the following fields. Follow the examples to make similar aliases. The goal is to make the values easily interpretable to the user.
     + Senior Citizen: "Is Senior Citizen", "Not Senior Citizen"
     + Partner
     + Dependents
     + Paperless Billing
     + Phone Service
     + Multiple Lines: "No": "Single line", "Yes": "Multiple lines", "No phone service": "No Phone service"
     + Online Security: "No": "Internet without Online Security", "Yes": "Online Security", "No internet service": "No internet service"
     + Online Backup
     + Device Protection
     + Tech Support
     + Streaming TV
     + Streaming Movies
  6. Create a new grouped field from payment type. Group the automatic payments by selecting both of those payment types, and name the group "Automatic Payment". Then group the non-auto payments and name the group "Manual Payment". You will then have a new field at the end of your table titled "Payment Type (group)".
  7. Once your table is ready, export it to a csv for a backup in case your tableau file gets deleted! (data -> export data to csv)

**Data Resources**

* 1. <https://ds.codeup.com/appendix/open_data/>
  2. <https://public.tableau.com/en-us/s/resources>

**References**

Medrano, Diego (April 14, 2016), Tableau Online tips: Extracts, live connections, & cloud data, <https://www.tableau.com/about/blog/2016/4/tableau-online-tips-extracts-live-connections-cloud-data-53351>

* Creating Custom Fields

**Creating Custom Calculations & Fields**

Custom-calculated fields are used to segment data, convert data types, aggregate data, filter results, and calculate ratios.

Example scenarios include:

* 1. Data is missing, such as net profit, when you have revenue and expenses.
  2. Transform values, such as year-over-year growth. (Quick Table Calculation)
  3. Quickly categorize data, such as: IF [net profit] > 0 THEN 'growth' ELSE 'decline' END.

**Lesson Goal**

* 1. Create basic calculations

**Basic Calculations**

Basic calculations allow you to transform values or members at the data source level of detail (a row-level calculation) or at the visualization level of detail (an aggregate calculation).

* 1. Example row-level calculations, or those at the data source level of detail, include: splitting an email address into two columns, username & domain, computing a date difference between start date and end date, or computing the profit of each sale by taking the difference of cost and sales. The calculation in Tableau would look like: [sales\_price] - [cost]. In pandas, the same type of calculation would look something like: df['profit'] = df.sales - df.cost.
  2. Example aggregate calculations, or those at the visualization level of detail include: summing the profit or computing the median sales price. How these are grouped when aggregated depends on the dimensions in your visualization. So, the results of the calculation will change with your visualization. The calculation in Tableau would look like: MEDIAN([sales\_price]). What this concept could look like in pandas, assuming you want to compute the median sales price for each product category: df.groupby(['product\_category'])['sales\_price'].median(). In the basic calculations, though, the group by is updated every time you alter your visualization by adding or removing a dimension (i.e. a group by field).
     + *NOTE: This may be addressed in the calculation using a*[*FIXED Level of Detail*](https://help.tableau.com/current/pro/desktop/en-us/calculations_calculatedfields_lod.htm#fixed)*which computes an aggregate using only the specified dimension(s).*

**Simple Calculations**

Create a [simple calculated field](https://help.tableau.com/current/pro/desktop/en-us/calculations_calculatedfields_formulas.htm) with the following steps:

* 1. Analysis > Create Calculated Field OR in the field names on the left, click on the down arrow of a field you wish to use in the calculation and select 'Create Calculated Field'.
  2. Name the new field
  3. Enter a formula.

**Ad Hoc Calculations**

[Ad hoc calculations](https://help.tableau.com/current/pro/desktop/en-us/calculations_calculatedfields_adhoc.htm) are calculations that you can create and update as you work with a field on a shelf in the view. Ad-hoc calculations are also known as type-in or in-line calculations. Double-click on a field in your chart (in the rows or columns list, e.g.) and begin.

**Quick Table Calculations**

To create [quick table calculations](https://help.tableau.com/current/pro/desktop/en-us/calculations_tablecalculations_quick.htm), such as year-over-year difference, click on the menu arrow of a field that exists in your chart/table and select 'Quick Table Calculation', then follow the options.

**The Format of Calculations**

To understand the format of calculations, we will use this example from the [Tableau help site](https://help.tableau.com/current/pro/desktop/en-us/functions_operators.htm):

IF [Profit per Day] > 2000 THEN "Highly Profitable"

ELSEIF[Profit per Day] <= 0 THEN "Unprofitable"

ELSE "Profitable"

END

// this function labels all profits over 2000 as highly profitable, those <= 0 as unprofitable and all others as profitable.

* 1. **Functions**: [Functions](https://help.tableau.com/current/pro/desktop/en-us/functions.htm) transform the values or members in a field. The functions in the example include IF, THEN, ELSEIF, ELSE, and END.
  2. **Fields**: [Fields](https://help.tableau.com/current/pro/desktop/en-us/functions_operators.htm#Fields) are dimensions or measures from your data source, i.e. columns. The fields in the example include Profit per Day.
  3. **Operators**: [Operators](https://help.tableau.com/current/pro/desktop/en-us/functions_operators.htm#operator-syntax) are symbols to denote an operation. Operators in our example include > and <=.
  4. **Literal Expressions**: [Literal expressions](https://help.tableau.com/current/pro/desktop/en-us/functions_operators.htm#literal-expression-syntax) are constant values that are represented “as is”, such as a string you want to match or return in an if statement. The literal expressions in our example include "Profitable", "Unprofitable", "Highly Profitable", 2000, and 0.
  5. **Parameters**: [Parameters](https://help.tableau.com/current/pro/desktop/en-us/functions_operators.htm#add-parameters-to-a-calculation) are placeholder variables that can be inserted into calculations to replace constant values. See more on [creating parameters](https://help.tableau.com/current/pro/desktop/en-us/parameters_create.htm#create-a-parameter).
  6. **Comments**: [Comments](https://help.tableau.com/current/pro/desktop/en-us/functions_operators.htm#add-comments-to-a-calculation) You should add comments to any calculations beyond the very basic and simple. Comments are preceded by //. The comment in our example is: "// this function labels all profits over 2000 as highly profitable, those <= 0 as unprofitable and all others as profitable."
  7. **Data Types**: [Data Types in Calculations](https://help.tableau.com/current/pro/desktop/en-us/functions_operators.htm#understanding-data-types-in-calculations) include string, date/datetime, number, and boolean.

**Exercises**

Creating Custom Fields:

* 1. Create a new calculated field from total charges. Call it "Estimated Tenure (months)". It computes the number of months the customers were around from monthly charges and total charges. Round your tenure value to the nearest whole number.
  2. Create a new calculated field from "Churn Month". Call it "Customer Status". It computes whether or not a customer has churned based on the "Churn Month" field using an IF statement. Your function should look like: IF isnull([field1]) THEN 'churned' ELSE 'active' END.
  3. Create a new calculated field from "Tech Support". Call it "with Tech Support". Using an IF ELSE statement, write the function to return a 1 if the customer has tech support ([Tech Support = 'Yes']) else a 0.
  4. Repeat the process in number 3 above for the following fields: 'Streaming TV', 'Streaming Movies', 'Online Backup', 'Online Security'.
  5. Create a new calculated field from "with Tech Support". Call it "Add-On Count". In this field, you will add all the fields you created in numbers 3 and 4 above. It will look like: [with Tech Support] + [with Streaming TV] + [with ...]
  6. Convert all of your new fields to dimensions. (Click the down arrow to the right of the field name and select convert to dimension)

**Appendix**

**Other Types of Calculations**

* 1. Create LOD expression
  2. Create table calculation
  3. Become familiar with [Tableau functions](https://help.tableau.com/current/pro/desktop/en-us/functions.htm)
  4. Become familiar with Tableau documentation on [Calculated Fields](https://help.tableau.com/current/pro/desktop/en-us/calculations_calculatedfields_create.htm)

**Level of Detail Expressions**

[Level of Detail (LOD) expressions](https://help.tableau.com/current/pro/desktop/en-us/calculations_calculatedfields_lod.htm), like basic calculations, allow you to compute values at the data source level and the visualization level. However, LOD calculations give you even more control at the level of granularity you want to compute. They can be performed at a more granular level (INCLUDE), a less granular level (EXCLUDE), or an entirely independent level (FIXED) with respect to the granularity of the visualization. The function in Tableau would look like: { FIXED [product\_category]:(MEDIAN([sales\_price]))}. In pandas, this would look like: df.groupby(['product\_category'])['sales\_price'].median(). This is just like the example in basic calculations, but this time, in Tableau, the group by field stays static in the visualizations. So if you create a visualization that removes the product\_category dimension and adds product\_subcategory as a dimension, the value of the median sales\_price will remain at the product\_category level. So you will see multiple subcategories with the same value.

**Table Calculations**

[Table calculations](https://help.tableau.com/current/pro/desktop/en-us/calculations_tablecalculations.htm) allow you to transform values at the level of detail of the visualization only. For example, if you wanted to add year-over-year growth to your visualization, you could do this in 2 ways. The first way is to add in the Create Calculated Field formula box. It would look like this: ATTR([sales]) - LOOKUP(ATTR([sales]), -1). You could also create this calculation by clicking on the menu arrow of the field sales and selecting 'Quick Table Calculation'. Common use cases for table calculations include ranking, cumulative total, rolling averages, and inter-row calculations such as year-over-year.

* Creating Charts

**Creating & Customizing Charts**

Before we can create dashboards and stories, we have to create charts. As of now, we should have one tab labeled *data source*, and a blank sheet labeled *sheet 1*. We will now move to *sheet 1*. Every chart we wish to create, we will do so on a new sheet such as this one. Tableau refers to the tabs used for indiviual charts as *worksheets*. In this lesson, we will create a variety of charts on multiple worksheets. One way to create a new tab is to click on the tab that displays a bar chart icon with a '+' in the bottom right corner. The two tabs that follow it are for creating a new dashboard and creating a new story, respectively. Another way you can create a new tab is by duplicating existing tab (right click on tab name) and then editing variables, chart types, titles, etc. So, let's get building.

**Lesson Goals**

* 1. Creating charts
  2. Sorting and filtering data
  3. Displaying the data underlying a workbook

**Data Table**

Tableau can create a basic table, like a pivot table you would create in Excel but with more flexibility, that groups and aggregates. You can specify how it aggregates, which fields to aggregate, and the format of the numeric data. If you explore the 'Analysis' tab, you can change the table values to return a 'percentage of' and then specify whether to compute percentage of row, column, or entire table, for example.

**Box Plot**

"A box and whisker plot—also called a box plot—displays the five-number summary of a set of data. The five-number summary is the minimum, first quartile, median, third quartile, and maximum." (Khan Academy, 2020)

A purpose of box plot is to see how the points are distributed across the quartiles, as well as any outliers that exist in the data. For a box plot, you need 0 or more dimensions and 1 or more measures.

**Bar Plot**

**Heatmap**

To demonstrate the heatmap, we will duplicate our basic table tab, and then we can use the "Show Me" menu on the top right to change the visualization from a table to a heatmap or a highlight table (which I think of as a heatmap, also). In this case the highlight table gives us what we need.

**Scatter Plot**

A scatter plot needs 2 measures, or numeric variables. By adding dimensions the scatter plot goes from 1 data point to multiple data points, depending on the number of distinct values or combinations in the dimensions.

Dimensions:

For example, let's take retention rate (x) and cohort size (y). We have to select a way to aggregate each of those. Let's aggregate both retention rate and cohort size by median. That value really doesn't mean anything when everything is aggregated together into a single point. So, we need to bring in some dimensions. Think, *group by*. Dimensions can be visualized separately through color, shape, size, or other means, *or* they can just be pulled in so that individual points are not lost and the aggregate measure only goes as far as each value in that dimension. Back to our example, we can add the dimension of *Control* which indicates whether the school is public institution or a private one. That is a good one to indicate by color, as there are a limited number of distinct values. But what if we want to see a data point for each college? We can drag that dimension into our chart. It will add the college name in the info box that displays when hovering over the point and our chart will now have a dot for each college. The measure we are aggregating is now the actual value you would find in the original dataset.

Analysis:

On the top left of the window, there is a tab that exists behind the *Data* tab (where all the columns are listed) called *Analytics*. From here (or from Analysis Menu item at top of page) we can add trend lines, regression formula, among other bits of information.

**Line Chart**

A line chart needs 2 continuous variables. A line chart can be used to analyze the correlation of 2 variables. It can be used in time series analysis and many other use cases as well.

**Map**

Tableau chart types: **symbol maps**, **maps**

These both require a geographic element. They can take 0 or more dimensions. **Symbol maps** can take 0 to 2 measures, while **maps** take 0 to 1 measures.

Drag longitude to columns and latitude to rows. A default symbol map of a map will populate. Drag *Student Count* to the center of the map and aggregate by median. What's wrong? Why doesn't anything happen? (hint: group by) Add *City* to group by. Where is Portland, OR? I can't find the dot representing portland. Why could this be?

First we should confirm that the data contains Portland. There are multiple ways to do this. One way is to add *city* as a filter and search to include only Portland. Another way is to filter by State and drill closer into the state. Upon doing that, we see a very little dot of "Portland". That seems counter to what I would expect. Big city should mean bigger dot. This is one way maps can be confusing and/or deceiving. It really shouldn't when we think about how we are aggregating. We are taking the median, and larger cities are going to have more, smaller colleges, thus driving down the median. We could use sum if we want to see total students in the city. We could also count the number of colleges to get different information.

Let's change the **symbol map** to **map**. We will then colored the state by the number of colleges that exist in that state. What is wrong with this visualization (misleading, deceptive, confusing, ...)?

**Display Underlying Data**

To display or export the underlying data:

To create a **crosstab** from a chart, right click on tab and select "duplicate as crosstab"

**Sorting and filtering data**

**Filter**

**Sort**

**Titles, Labels, Captions, Summary**

**Titles**

You might notice that your title matches the name of your tab. Rename your tab, and you will notice that these two strings are connected, by default. Now double click on the title of the page, above the chart. Now you can see how these strings are linked. To customize the chart name, enter the new name in the box provided. If you would like to maintain the synchronization between the tab and the chart title, rename the tab, but keep the chart title as < Sheet Name >

**Captions**

To add a caption, go to *Worksheet* menu header, and select *Show Caption*. Captions are useful for key points, notes about the data, what you want the user to take from your chart, how to read the chart, etc. When you are answering a question through a viz, also answer the question in your caption.

**Axes**

You can customize the x & y axes by selecting one and right clicking.

**Summary**

Right click and select *Summary* to view the summary card. You can also view it by *Worksheet* -> *Show Summary*.

**Exercises**

**Plot 1: Scatter plot**

* 1. Create the plot:
     + Create a Scatter Plot with the X axis representing 'Estimated Tenure (months)' and y axis representing 'Monthly Charges'.
     + In order to create a scatter plot of all the customers' tenure x monthly charges, you need to add the customer ids to the chart. Drag the 'Customer ID' field to the "Detail" box in "Marks".
     + Next, add some color to represent whether or not a customer has churned. Do this by dragging 'Customer Status' to the "Color" box in "Marks".
  2. Add a caption:
     + What do you notice from your plot? Add the caption box to your window by right clicking on a gray area in your window until you see the option to select "Caption". Select it so that it has a check mark next to it.
     + Once the caption box appears, double click inside it to bring up the edit box. Keep the default text, but after it, add any takeaways you found in this visualization about how churn seems to relate to tenure and monthly charges.
  3. Format the axes:
     + Format the axis labels: right click in the area of the y-axis and select format.
     + To your left you will see "scale". Format the number labels to be currency (custom). Remove the decimals.
  4. Add a Title:
     + Double click in the title space, currently titles "Sheet 1", and edit your title.
     + Make your title indicative of what you want the user to take away from this chart. A question is often a useful way to title a chart. For example, “Are Monthly Charges Related to When/Whether a Customer Will Churn?”

**Plot 2: Bar plot**

* 1. Create the plot:
     + Make a bar plot showing how many customers have churned and how many are active.
     + In a bar plot, you will have 1-2 dimensions and 1 measure, such as count of total records. In this case, make the x-axis 'Customer Status' and the y-axis count of the total records.
     + Add a dimension to color, internet service type, so you can view number of customers and their status with respect to type of internet
  2. Add a caption with your takaways.
  3. Format the axes:
     + Change the title of 'count of .csv' to something meaningful, such as "Number of Customers". Do this by right clicking on the y-axis and selecting "Edit Axis". There you have the option to change the title.
  4. Add a title:
     + What do you want the user to takeaway from your chart? Make your point clear and hard to miss! An example is: "Customers with No Internet Service Do Not Stick Around"

**Plot 3: Duplicate Plot 2**

* 1. Right Click on sheet 2 and select "duplicate"
  2. Convert your y-axis into a percentage:
     + Select Analyze (top menu item) -> percentage of -> columns

**Plot 4: Explore Tech Support and Add-Ons**

Explore the number of add-ons with and without tech support and how these relate to churn. Control for internet service type.

* 1. Add a new chart.
  2. Add to the columns, Internet Service Type and Add-On Count.
  3. Add to rows 'with Tech Support' and your record count values (e.g. mine is titled 'Telco\_from\_sql.csv (Count)')
  4. Select the best type of plot for your data and insights.
  5. Give your chart a title that indicates what you take away from the chart.

Does having tech support have a relationship with churn? If so, what is it? Is it what you would expect?

**Save your workbook to Tableau Public**

**References**

Khan Academy, (2020), Retrieved from <https://www.khanacademy.org/math/statistics-probability/summarizing-quantitative-data/box-whisker-plots/a/box-plot-review>

* Creating Dashboards

**Creating a Dashboard**

A [dashboard](https://help.tableau.com/current/pro/desktop/en-us/dashboards.htm) is a collection of several views or charts. In Tableau, it is another tab, just like the worksheets for charts and the data tab, but specified for dashboards. When data is modified or a chart in a worksheet is modified, the dashboard is modified, and vice versa.

Remember all we have discussed on knowing your audience and best practices in storytelling as you create your dashboards. Who is your audience, what is the purpose of the dashboard, what is the setting or what kind of device will it be viewed on, etc.

**Lesson Goals**

* 1. Create a dashboard
  2. Set default display to be tablet or phone
  3. Add charts to dashboard
  4. Group layout containers
  5. Add text objects
  6. Add interactivity through highlighting and filters

**Dashboard Display**

* 1. [Dashboard Size](https://help.tableau.com/current/pro/desktop/en-us/dashboards_organize_floatingandtiled.htm#dashboard-size-options): options include fixed size (default: dashboard remains the same size), range (size scales between a min and max value), and automatic (resizes to fit the window used).
  2. [Dashboard Device Layouts](https://help.tableau.com/current/pro/desktop/en-us/dashboards_dsd_create.htm): You can make use of templates and create dashboard layouts based on the type of device the dashboard will be viewed on (phone, tablet, desktop).
  3. [Accessible Dashboards](https://help.tableau.com/current/pro/desktop/en-us/accessibility_dashboards.htm): Tableau also has features for creating accessible dashboards that you can look into further using the hyperlink.
  4. [Managing Sheets in Dashboard Display](https://help.tableau.com/current/pro/desktop/en-us/environ_workbooksandsheets_sheets_hideshow.htm): When you are deploying a dashboard or a story, you will want to manage your sheets so that you product is clean and simple.

**Dashboard Objects**

To [add an object](https://help.tableau.com/current/pro/desktop/en-us/dashboards_create.htm#add-an-object), select an item from the bottom left corner and drag to the main area on the page. To set the options, click the object container to select it and then click the arrow in the upper corner to open the shortcut menu.

* 1. **Horizontal and Vertical objects** provide [layout containers](https://help.tableau.com/current/pro/desktop/en-us/dashboards_refine.htm#Use_a_layout_container) that let you group related objects together and fine-tune how your dashboard resizes when users interact with them. Your layout containers can be customized in ways such as [grouping containers together](https://help.tableau.com/current/pro/desktop/en-us/dashboards_organize_floatingandtiled.htm#group-items-using-layout-containers), [evenly distributing layout containers' items](https://help.tableau.com/current/pro/desktop/en-us/dashboards_organize_floatingandtiled.htm#evenly-distribute-a-layout-containers-items), [tiling or floating items](https://help.tableau.com/current/pro/desktop/en-us/dashboards_organize_floatingandtiled.htm#tile-or-float-dashboard-items), and [adding padding, borders, or background colors to items](https://help.tableau.com/current/pro/desktop/en-us/dashboards_organize_floatingandtiled.htm#add-padding-borders-and-background-colors-around-items).
  2. **Text objects** can provide headers, explanations, and other information.
  3. **Image objects** add to the visual flavor of a dashboard, and you can link them to specific target URLs.
  4. **Web Page objects** display target pages in the context of your dashboard. Be sure to review these web security options, and be aware that some web pages don't allow themselves to be embedded—Google is one example.
  5. **Blank objects** help you adjust spacing between dashboard items.
  6. **Navigation objects** let your audience navigate from one dashboard to another, or to other sheets or stories. You can display text or an image to indicate the button's destination to your users, specify custom border and background colors, and provide informational tooltips.
  7. **Export objects** let your audience quickly create a PDF file, PowerPoint slide, or PNG image of a dashboard. Formatting options are similar to Navigation objects.
  8. **Extension objects** let you add unique features to dashboards or integrate them with applications outside Tableau ([more](https://help.tableau.com/current/pro/desktop/en-us/dashboard_extensions.htm))

**Dashboard Interactivity**

Dashboards can become interactive and more flexible through the use of filters and parameters. Also, enabling the use of highlighting allows for exploration in the dashboard. One was you can add interactivity is in upper corner of sheet, enable the Use as Filter option to use selected marks in the sheet as filters for other sheets in the dashboard. You can find other actions possible [here](https://help.tableau.com/current/pro/desktop/en-us/actions_dashboards.htm).

**Exercises**

* 1. Create a Dashboard with at least three visualizations that you’ve created so far.
  2. Create a text object as a header with the name “TelcoCo KPI Report” Add [this image](https://drive.google.com/file/d/1OmmavSYhDbDxZNLvqnDMQVoN9zm09qKj/view?usp=sharing) as an object on your dashboard.
  3. Determine which of your visualizations are most valuable, and remove at least one, then change the format into a phone layout.
     + How did you decide to drop one of your charts? Think about the ordination of value that each provides and describe your thought process.
* Creating Stories

**Creating a Story**

A Tableau Story is a sequence of visualizations that work together to convey information. You can create stories to tell a data narrative, provide context, demonstrate how decisions relate to outcomes, or to simply make a compelling case. ~ Stories on [help.tableau.com](http://help.tableau.com)

A story is a separate sheet, just like data, charts, and dashboards. Stories contain multiple pages within them, known as story points. Users can navigate between the points by clicking on the menu at the top of the story.

**Story Elements**

* 1. Adding a new story point: Select Blank to add a new blank point, or Duplicate to use a copy of the current point as the beginning of a new one.
  2. The Story pane: Use this pane to drag dashboards, sheets, and text descriptions to your story sheet. You can also show/hide the title and set the size of the story.
  3. The Layout pane: Change the style of the navigator object and choose whether to show/hide the navigation arrows.
  4. The Story menu: Use this menu to format the story font styles, shading, and borders.
  5. The Story toolbar: Hover your mouse over the navigation bar to make it appear. Use it to delete points, revert changes, update points, or save the point to a new story.
  6. The navigator: Use the navigator to select different story points. You can also change the order of story points by dragging and dropping them. Your audience will use the navigator to engage with your story.

[Source](https://help.tableau.com/current/pro/desktop/en-us/story_workspace.htm)

**7 Types of Data Stories**

Tableau talks about 7 types of data stories in a table you can see [here](https://help.tableau.com/current/pro/desktop/en-us/story_best_practices.htm#the-seven-types-of-data-stories).

**Build a Story**

For details on how to create and format stories, check out the [Tableau documentation](https://help.tableau.com/current/pro/desktop/en-us/story_create.htm).

**Exercises**

* 1. Create a story titled “TelcoCo Key Performance Presentation”.
  2. Create an initial point with two text objects, as a preview for your presentation.
  3. Make a second point with your most valuable (by personal assessment) visualization.
  4. Make a third point containing the desktop variant of your dashboard.