**Probability density functions**

**50 XP**

**1. Probability density functions**

So far, we have talked about probabilities of discrete quantities, such as die rolls and number of bus arrivals, but what about continuous quantities?

**2. Continuous variables**

A continuous quantity can take on any value, not just discrete ones. For example, the speed of a train can be 45-point-76 km/h.

**3. Michelson's speed of light experiment**

Continuous variables also have probability distributions. Let's consider an example. In 1879, Albert Michelson performed 100 measurements of the speed of light in air.

**4. Michelson's speed of light experiment**

Each measurement has some error in it; conditions, such as temperature, humidity, alignment of his optics, et cetera, change from measurement to measurement. As a result, any fractional value of the measured speed of light is possible, so it is apt to describe the results with a continuous probability distribution. In looking at Michelson's numbers, shown here in units of megameters/s, or 1000s of kilometers/s, we see this is indeed the case. What probability distribution describes these data? I posit that these data follow the famous Normal distribution. To understand what the Normal distribution is,

**5. Probability density function (PDF)**

let's consider its probability density function, or PDF. This is the continuous analog to the probability mass function, the PMF. It describes the chances of observing a value of a continuous variable. The probability of observing a single value of the speed of light does not make sense, because there is an infinity of numbers,

**6. Normal PDF**

say between 299-point-6 and 300-point-0 megameters per second. Instead, areas under the PDF give probabilities. So, the probability of measuring that the speed of light is greater than

**7. Normal PDF**

300,000 km/s is an area under the normal curve. Parametrizing the PDF based on Michelson's experiments, this is about a 3% chance, since the pink region is about 3% of the total area under the PDF. To do this calculation, we were really just looking at the cumulative distribution function,

**8. Normal CDF**

or CDF, of the Normal distribution. Here is the CDF of the Normal distribution. Remember that the CDF gives the probability the measured speed of light will be less than the value on the x-axis. So, reading off the value at 300,000 km/s,

**9. Normal CDF**

we see that there is a 97% chance that a speed of light measurement is less than that. So, there is about a 3% change it is greater. We will study the Normal distribution in more depth in the coming exercises, but for right now,

**10. Let's practice!**

let's review some of the concepts we've learned about continuous distribution functions.

# Interpreting PDFs

Consider the PDF shown to the right (it may take a second to load!). Which of the following is true?

##### Possible Answers

* ** x is more likely to be less than 10 than to be greater than 10.**

 x is more likely to be **greater than 10** than to be less than 10.

 We cannot tell from the PDF if is more likely to be greater than or less than 10.

 This is not a valid PDF because it has two peaks.

Incorrect. A PDF can have any shape. It just has to be singularly-valued, nonnegative, and the total area under the PDF must be equal to one.

Correct! The probability is given by the area under the PDF, and there is more area to the left of 10 than to the right.

# Interpreting CDFs

On the right is the CDF corresponding to the PDF you considered in the last exercise. Using the CDF, what is the probability that

is greater than 10?

##### Possible Answers

* **0.25 Correct! The value of the CDF at x = 10 is 0.75, so the probability that x < 10 is 0.75. Thus, the probability that x > 10 is 0.25.**
* **0.75**
* 3.75
* 15

Incorrect. Remember, that if the value of the CDF is 0.75 at x = 10, then the probability that x is less than 10 is 0.75.

Incorrect. For a CDF, probability is not given by the area under the curve.

Incorrect. Remember, probabilities must be between zero and one.

**Daily XP450**

**Introduction to the Normal distribution**

**50 XP**

**1. Introduction to the Normal distribution**

The Normal distribution is famous, and we just used it as an example to learn about continuous distributions. We will not go into the story of the Normal distribution in depth,

**2. Normal distribution**

but for the purposes of this course, it suffices to say that it describes a continuous variable whose PDF is symmetric and has a single peak.

**3. Normal distribution**

Indeed, that is what its PDF looks like. The Normal distribution is parametrized by two parameters.

**4. Normal distribution**

The mean determines

**5. Normal distribution**

where the center of the peak is.

**6. Normal distribution**

The standard deviation

**7. Normal distribution**

is a measure of how wide the peak is, or how spread out the data are. Note that

**8. Insert title here...**

the mean and

**9. Insert title here...**

standard deviation are the names of the parameters of the Normal distribution. Do not confuse these with the mean and standard deviation that we computed directly from the data when doing exploratory data analysis. I know that nomenclature is confusing, but it is important to keep straight.

**10. Comparing data to a Normal PDF**

Now, if we add a histogram of the Michelson measurements, we see that the measured speed of light in air looks to be Normally distributed. Comparing the histogram to the PDF suffers from binning bias, so it is better to compare the ECDF of the data to the theoretical CDF of the Normal distribution. To compute the theoretical CDF, we can use np dot random dot normal to draw samples and then we can compute the CDF.

**11. Checking Normality of Michelson data**

As was the case with sampling out of the binomial distribution, we need to provide parameters, in this case the mean and standard deviation, to parametrize the Normal distribution we are sampling out of. The mean and standard deviations computed from the data are good estimates, so we'll compute them and pass them into np dot random dot normal to take our samples. We then use the function we already wrote to compute the ECDFs of the data and of the Normally distributed theoretical samples we just drew.

**12. Checking Normality of Michelson data**

Finally, we plot the theoretical and empirical CDFs on the same plot.

**13. Checking Normality of Michelson data**

With the absence of binning bias, it is much clearer that the Michelson data are approximately Normally distributed.

**14. Let's practice!**

Now that you can sample out of a Normal distribution, let's practice using it!

# The Normal PDF

In this exercise, you will explore the Normal PDF and also learn a way to plot a PDF of a known distribution using hacker statistics. Specifically, you will plot a Normal PDF for various values of the variance.

##### Instructions

**100 XP**

* Draw 100,000 samples from a Normal distribution that has a mean of 20 and a standard deviation of 1. Do the same for Normal distributions with standard deviations of 3 and 10, each still with a mean of 20. Assign the results to samples\_std1, samples\_std3 and samples\_std10, respectively.
* Plot a histograms of each of the samples; for each, use 100 bins, also using the keyword arguments normed=True and histtype='step'. The latter keyword argument makes the plot look much like the smooth theoretical PDF. You will need to make 3 plt.hist() calls.
* Hit submit to make a legend, showing which standard deviations you used, and show your plot! There is no need to label the axes because we have not defined what is being described by the Normal distribution; we are just looking at shapes of PDFs.
* # Draw 100000 samples from Normal distribution with stds of interest: samples\_std1, samples\_std3, samples\_std10
* # Make histograms
* # Make a legend, set limits and show plot
* \_ = plt.legend(('std = 1', 'std = 3', 'std = 10'))
* plt.ylim(-0.01, 0.42)
* plt.show()

Great work! You can see how the different standard deviations result in PDFs of different widths. The peaks are all centered at the mean of 20.

# Draw 100000 samples from Normal distribution with stds of interest: samples\_std1, samples\_std3, samples\_std10

samples\_std1 = np.random.normal(20, 1, size=100000)

samples\_std3 = np.random.normal(20, 3, size=100000)

samples\_std10 = np.random.normal(20, 10, size=100000)

# Make histograms

\_ = plt.hist(samples\_std1, bins=100, normed=True, histtype='step')

\_ = plt.hist(samples\_std3, bins=100, normed=True, histtype='step')

\_ = plt.hist(samples\_std10, bins=100, normed=True, histtype='step')

# Make a legend, set limits and show plot

\_ = plt.legend(('std = 1', 'std = 3', 'std = 10'))

plt.ylim(-0.01, 0.42)

plt.show()

# Draw 100000 samples from Normal distribution with stds of interest: samples\_std1, samples\_std3, samples\_std10 samples\_std1 = np.random.normal(20, 1, size=100000) samples\_std3 = np.random.normal(20, 3, size=100000) samples\_std10 = np.random.normal(20, 10, size=100000) # Make histograms \_ = plt.hist(samples\_std1, bins=100, normed=True, histtype='step') \_ = plt.hist(samples\_std3, bins=100, normed=True, histtype='step') \_ = plt.hist(samples\_std10, bins=100, normed=True, histtype='step') # Make a legend, set limits and show plot \_ = plt.legend(('std = 1', 'std = 3', 'std = 10')) plt.ylim(-0.01, 0.42) plt.show()

# The Normal CDF

Now that you have a feel for how the Normal PDF looks, let's consider its CDF. Using the samples you generated in the last exercise (in your namespace as samples\_std1, samples\_std3, and samples\_std10), generate and plot the CDFs.

##### Instructions

**100 XP**

* Use your ecdf() function to generate x and y values for CDFs: x\_std1, y\_std1, x\_std3, y\_std3 and x\_std10, y\_std10, respectively.
* Plot all three CDFs as dots (do not forget the marker and linestyle keyword arguments!).
* Hit submit to make a legend, showing which standard deviations you used, and to show your plot. There is no need to label the axes because we have not defined what is being described by the Normal distribution; we are just looking at shapes of CDFs.
* # Generate CDFs
* # Plot CDFs
* # Make a legend and show the plot
* \_ = plt.legend(('std = 1', 'std = 3', 'std = 10'), loc='lower right')
* plt.show()

# Generate CDFs

x\_std1, y\_std1 = ecdf(samples\_std1)

x\_std3, y\_std3 = ecdf(samples\_std3)

x\_std10, y\_std10 = ecdf(samples\_std10)

# Plot CDFs

\_ = plt.plot(x\_std1, y\_std1, marker='.', linestyle='none')

\_ = plt.plot(x\_std3, y\_std3, marker='.', linestyle='none')

\_ = plt.plot(x\_std10, y\_std10 , marker='.', linestyle='none')

# Make a legend and show the plot

\_ = plt.legend(('std = 1', 'std = 3', 'std = 10'), loc='lower right')

plt.show()

# Generate CDFs x\_std1, y\_std1 = ecdf(samples\_std1) x\_std3, y\_std3 = ecdf(samples\_std3) x\_std10, y\_std10 = ecdf(samples\_std10) # Plot CDFs \_ = plt.plot(x\_std1, y\_std1, marker='.', linestyle='none') \_ = plt.plot(x\_std3, y\_std3, marker='.', linestyle='none') \_ = plt.plot(x\_std10, y\_std10 , marker='.', linestyle='none') # Make a legend and show the plot \_ = plt.legend(('std = 1', 'std = 3', 'std = 10'), loc='lower right') plt.show()

Great work! The CDFs all pass through the mean at the 50th percentile; the mean and median of a Normal distribution are equal. The width of the CDF varies with the standard deviation.

**Daily XP700**

**The Normal distribution: Properties and warnings**

**50 XP**

**1. The Normal distribution: Properties and warnings**

Here is the

**2. Image: Deutsche Bundesbank**

10 Deutschmark bill. It was retired in 2002 with the adoption of the Euro by Germany. The man pictured on this bill is Karl Gauss, one of the greatest mathematicians of all time. Clearly, the Germans think highly of Gauss to put him on their currency.

**3. The Gaussian distribution**

If we zoom in to the center of the bill, we see which of Gauss's barrel full of accomplishments they think most highly of. There it is, the Normal distribution! It is often also called the Gaussian distribution, after its inventor, so you will often hear it referred to this way. So, yes, the Normal distribution is very important and very widely used. It is so prevalent, it is worth talking about some more. In practice, it is used to describe most symmetric peaked data you will encounter. Furthermore, for many of the statistical procedures you have heard of, Normality assumptions about the data are present. Indeed, it is a very powerful distribution that seems to be ubiquitous in nature, not just in the field of statistics. That said, there are important caveats about the distribution and we need to be careful when using it. First off, often times things you may think are Normally distributed are not.

**4. Length of MA large mouth bass**

Consider for example large mouth bass in Massachusetts lakes measured in 1994 and 1995 by the Massachusetts Department of Environmental Protection. If we look at a histogram of the length of the 316 fish they measured, they appear to be Normally distributed. Indeed, when we look at

**5. Length of MA large mouth bass**

the ECDF overlayed with a theoretical Normal CDF, the measurements look close to Normally distributed. There are some systematic differences, though,

**6. Length of MA large mouth bass**

especially on the left tail. So this is not quite a Normal distribution, but we might not be making too big of an error by treating it as so.

**7. Mass of MA large mouth bass**

Now, let's consider the mass of the bass. One might think that since the length of these bass is close to Normally distributed, the mass should be also. When we overlay the theoretical Normal CDF on the ECDF of the data, it is not even close. Seeing this immediately shows us that our initial thought was incorrect. Another important issue to keep in mind when using the Normal distribution is

**8. Light tails of the Normal distribution**

the lightness of its tails. If we look at the Normal distribution, the probability of being

**9. Light tails of the Normal distribution**

more than four standard deviations from the mean is very small. This means that when you are modeling data as Normally distributed, outliers are extremely unlikely. Real data sets often have extreme values, and when this happens, the Normal distribution might not be the best description of your data.

**10. Let's practice!**

These caveats are not meant to scare you. I bring them up to remind you to always think carefully about assumptions that go into your analyses. The Normal distribution is still of great use, and use it you will in the next exercises!

What are the mean and standard deviation, respectively, of the Normal distribution that was on the 10 Deutschmark banknote, shown to the right?

##### Possible Answers

* **mean = 3, std = 1**
* mean = 3, std = 2
* mean = 0.4, std = 1
* mean = 0.6, std = 6

**Daily XP800**

##### Exercise

##### Exercise

# Are the Belmont Stakes results Normally distributed?

Since 1926, the Belmont Stakes is a 1.5 mile-long race of 3-year old thoroughbred horses. [Secretariat](https://en.wikipedia.org/wiki/Secretariat_(horse)) ran the fastest Belmont Stakes in history in 1973. While that was the fastest year, 1970 was the slowest because of unusually wet and sloppy conditions. With these two outliers removed from the data set, compute the mean and standard deviation of the Belmont winners' times. Sample out of a Normal distribution with this mean and standard deviation using the np.random.normal() function and plot a CDF. Overlay the ECDF from the winning Belmont times. Are these close to Normally distributed?

Note: Justin scraped the data concerning the Belmont Stakes from the [Belmont Wikipedia page](https://en.wikipedia.org/wiki/Belmont_Stakes).

##### Instructions

**100 XP**

* Compute mean and standard deviation of Belmont winners' times with the two outliers removed. The NumPy array belmont\_no\_outliers has these data.
* Take 10,000 samples out of a normal distribution with this mean and standard deviation using np.random.normal().
* Compute the CDF of the theoretical samples and the ECDF of the Belmont winners' data, assigning the results to x\_theor, y\_theor and x, y, respectively.
* Hit submit to plot the CDF of your samples with the ECDF, label your axes and show the plot.
* # Compute mean and standard deviation: mu, sigma
* # Sample out of a normal distribution with this mu and sigma: samples
* # Get the CDF of the samples and of the data
* # Plot the CDFs and show the plot
* \_ = plt.plot(x\_theor, y\_theor)
* \_ = plt.plot(x, y, marker='.', linestyle='none')
* \_ = plt.xlabel('Belmont winning time (sec.)')
* \_ = plt.ylabel('CDF')
* plt.show()

# Compute mean and standard deviation: mu, sigma

mu = np.mean(belmont\_no\_outliers)

sigma = np.std(belmont\_no\_outliers)

# Sample out of a normal distribution with this mu and sigma: samples

samples = np.random.normal(mu, sigma, size=10000)

# Get the CDF of the samples and of the data

x\_theor, y\_theor = ecdf(samples)

x, y = ecdf(belmont\_no\_outliers)

# Plot the CDFs and show the plot

\_ = plt.plot(x\_theor, y\_theor)

\_ = plt.plot(x, y, marker='.', linestyle='none')

\_ = plt.xlabel('Belmont winning time (sec.)')

\_ = plt.ylabel('CDF')

plt.show()

# Compute mean and standard deviation: mu, sigma mu = np.mean(belmont\_no\_outliers) sigma = np.std(belmont\_no\_outliers) # Sample out of a normal distribution with this mu and sigma: samples samples = np.random.normal(mu, sigma, size=10000) # Get the CDF of the samples and of the data x\_theor, y\_theor = ecdf(samples) x, y = ecdf(belmont\_no\_outliers) # Plot the CDFs and show the plot \_ = plt.plot(x\_theor, y\_theor) \_ = plt.plot(x, y, marker='.', linestyle='none') \_ = plt.xlabel('Belmont winning time (sec.)') \_ = plt.ylabel('CDF') plt.show()

The theoretical CDF and the ECDF of the data suggest that the winning Belmont times are, indeed, Normally distributed. This also suggests that in the last 100 years or so, there have not been major technological or training advances that have significantly affected the speed at which horses can run this race.

# What are the chances of a horse matching or beating Secretariat's record?

Assume that the Belmont winners' times are Normally distributed (with the 1970 and 1973 years removed), what is the probability that the winner of a given Belmont Stakes will run it as fast or faster than Secretariat?

##### Instructions

**100 XP**

* Take 1,000,000 samples from the normal distribution using the np.random.normal() function. The mean mu and standard deviation sigma are already loaded into the namespace of your IPython instance.
* Compute the fraction of samples that have a time less than or equal to Secretariat's time of 144 seconds.
* Print the result.
* # Take a million samples out of the Normal distribution: samples
* # Compute the fraction that are faster than 144 seconds: prob
* # Print the result
* print('Probability of besting Secretariat:', prob)

Great work! We had to take a million samples because the probability of a fast time is very low and we had to be sure to sample enough. We get that there is only a 0.06% chance of a horse running the Belmont as fast as Secretariat.

# Take a million samples out of the Normal distribution: samples

samples = np.random.normal(mu, sigma, size=1000000)

# Compute the fraction that are faster than 144 seconds: prob

prob = np.sum(samples <= 144)/len(samples)

# Print the result

print('Probability of besting Secretariat:', prob)

# Take a million samples out of the Normal distribution: samples

samples = np.random.normal(mu, sigma, size=1000000)

# Compute the fraction that are faster than 144 seconds: prob

prob = np.sum(samples <= 144)/len(samples)

# Print the result

print('Probability of besting Secretariat:', prob)

Probability of besting Secretariat: 0.000675

<script.py> output:

Probability of besting Secretariat: 0.000635

**The Exponential distribution**

**50 XP**

**1. The Exponential distribution**

Just as there are many named discrete distribution, there are many named continuous distributions as well. For example, let's take a another trip to Poissonville and stand at a bus stop. We know that the number of buses that will arrive per hour are Poisson distributed. But the amount of time between arrivals of buses is Exponentially distributed.

**2. The Exponential distribution**

The Exponential distribution has this story: the waiting time between arrivals of a Poisson process are exponentially distributed. It has a single parameter, the mean waiting time. This distribution is not peaked,

**3. The Exponential PDF**

as we can see from its PDF. As an example,

**4. Possible Poisson process**

As an example, we can look at the time between all incidents involving nuclear power since 1974. We might expect incidents to be well-modeled by a Poisson process, ie, the timing of one incident is independent of all others. So, the time between incidents should be Exponentially distributed.

**5. Exponential inter-incident times**

We can compute and plot the CDF we would expect based on the mean time between incidents and overlay that with the ECDF from the real data. We take our usual approach where we draw many samples out of the Exponential distribution, using the mean inter-incident time as the parameter. We make the plot and label the axes.

**6. Exponential inter-incident times**

We see that it is close to being Exponentially distributed, indicating the nuclear incidents can indeed be modeled as a Poisson process. The Exponential and Normal are just two of many examples of continuous distributions. Importantly, in many cases you can just simulate your story to get the CDF. Remember, you have the power of a computer. If you can simulate a story, you can get its distribution!

**7. Let's practice!**

Now, let's play with the Exponential distribution in some exercises.

# Matching a story and a distribution

How might we expect the time between Major League no-hitters to be distributed? Be careful here: a few exercises ago, we considered the probability distribution for the number of no-hitters in a season. Now, we are looking at the probability distribution of the time between no hitters.

#### Possible Answers

* Normal
* **Exponential**
* Poisson
* Uniform

# Waiting for the next Secretariat

Unfortunately, Justin was not alive when Secretariat ran the Belmont in 1973. Do you think he will get to see a performance like that? To answer this, you are interested in how many years you would expect to wait until you see another performance like Secretariat's. How is the waiting time until the next performance as good or better than Secretariat's distributed? Choose the best answer.

#### Possible Answers

* Normal, because the distribution of Belmont winning times are Normally distributed.
* Normal, because there is a most-expected waiting time, so there should be a single peak to the distribution.
* Exponential: It is very unlikely for a horse to be faster than Secretariat, so the distribution should decay away to zero for high waiting time.
* **Exponential: A horse as fast as Secretariat is a rare event, which can be modeled as a Poisson process, and the waiting time between arrivals of a Poisson process is Exponentially distributed.**

Correct! The Exponential distribution describes the waiting times between rare events, and Secretariat is rare!

# If you have a story, you can simulate it!

Sometimes, the story describing our probability distribution does not have a named distribution to go along with it. In these cases, fear not! You can always simulate it. We'll do that in this and the next exercise.

In earlier exercises, we looked at the rare event of no-hitters in Major League Baseball. Hitting the cycle, when a batter gets all four kinds of hits in a single game, is another rare baseball event. Like no-hitters, this can be modeled as a Poisson process, so the time between hits of the cycle are also Exponentially distributed.

How long must we wait to see a no-hitter and then a batter hit the cycle? The idea is that we have to wait some time for the no-hitter, and then after the no-hitter, we have to wait for hitting the cycle. Stated another way, what is the total waiting time for the arrival of two different Poisson processes in succession? The total waiting time is the time waited for the no-hitter, plus the time waited for the hitting the cycle.

Now, you will write a function to sample out of the distribution described by this story.

##### Instructions

**100 XP**

* Define a function with call signature successive\_poisson(tau1, tau2, size=1) that samples the waiting time for a no-hitter and a hit of the cycle.
  + Draw waiting times tau1 (size number of samples) for the no-hitter out of an exponential distribution and assign to t1.
  + Draw waiting times tau2 (size number of samples) for hitting the cycle out of an exponential distribution and assign to t2.
  + The function returns the sum of the waiting times for the two events.
* def successive\_poisson(tau1, tau2, size=1):
* """Compute time for arrival of 2 successive Poisson processes."""
* # Draw samples out of first exponential distribution: t1
* t1 = \_\_\_\_(\_\_\_\_, \_\_\_\_)
* # Draw samples out of second exponential distribution: t2
* t2 = \_\_\_\_(\_\_\_\_, \_\_\_\_)
* return t1 + t2

Great work! We'll put the function to use in the next exercise.

def successive\_poisson(tau1, tau2, size=1):

    """Compute time for arrival of 2 successive Poisson processes."""

    # Draw samples out of first exponential distribution: t1

    t1 = np.random.exponential(tau1, size=1)

    # Draw samples out of second exponential distribution: t2

    t2 = np.random.exponential(tau2, size=1)

    return t1 + t2

def successive\_poisson(tau1, tau2, size=1): """Compute time for arrival of 2 successive Poisson processes.""" # Draw samples out of first exponential distribution: t1 t1 = np.random.exponential(tau1, size=1) # Draw samples out of second exponential distribution: t2 t2 = np.random.exponential(tau2, size=1) return t1 + t2

# Distribution of no-hitters and cycles

Now, you'll use your sampling function to compute the waiting time to observe a no-hitter and hitting of the cycle. The mean waiting time for a no-hitter is 764 games, and the mean waiting time for hitting the cycle is 715 games.

##### Instructions

**100 XP**

* Use your successive\_poisson() function to draw 100,000 out of the distribution of waiting times for observing a no-hitter and a hitting of the cycle.
* Plot the PDF of the waiting times using the step histogram technique of a previous exercise. Don't forget the necessary keyword arguments. You should use bins=100, normed=True, and histtype='step'.
* Label the axes.
* Show your plot.
* # Draw samples of waiting times: waiting\_times
* # Make the histogram
* # Label axes
* # Show the plot

Great work! Notice that the PDF is peaked, unlike the waiting time for a single Poisson process. For fun (and enlightenment), I encourage you to also plot the CDF.

# Draw samples of waiting times: waiting\_times

waiting\_times = successive\_poisson(764, 715, size=100000)

# Make the histogram

\_ = plt.hist(waiting\_times, bins=100, normed=True, histtype='step')

# Label axes

\_ = plt.xlabel('waiting times')

\_ = plt.ylabel('PDF')

# Show the plot

plt.show()

# Draw samples of waiting times: waiting\_times waiting\_times = successive\_poisson(764, 715, size=100000) # Make the histogram \_ = plt.hist(waiting\_times, bins=100, normed=True, histtype='step') # Label axes \_ = plt.xlabel('waiting times') \_ = plt.ylabel('PDF') # Show the plot plt.show()

**oughts**

**50 XP**

**1. Final thoughts**

Congratulations! You have completed the first course in statistical thinking.

**2. You now can…**

You can now use Python to construct instructive plots and informative summary statistics to explore your data. You have also built the intellectual and computational infrastructure to think probabilistically. You've accomplished a lot in a short time. Impressive! The knowledge you learned in this course really shines when you directly apply it to statistical inference problems.

**3. In the sequel, you will…**

In the next course, you will work with real data sets in Python to infer parameter values by a variety of methods including linear regression. You will use hacker statistics to compute confidence intervals to help you couch the conclusions you draw from your data in the appropriate probabilistic language. You will perform hypothesis tests, such as A/B tests, to help you discern differences between data sets. It's a great opportunity to use and expand upon what you learned in this first course in statistical thinking.

**4. Let's practice!**

I encourage you to take that opportunity! See you there!

2

### Intro to Statistical Inference and Statistical Modelling in Python

Top of Form

Bottom of Form

Save



2 - 3 Hours

45 Points

We’d like you to **complete sections four and five** on this LinkedIn Learning course: Introduction to Statistical Inference and Introduction to Statistical Modeling. Please complete the quizzes corresponding to these two sections.

You’ll learn how to do the following concepts from the Art of Statistics in Python:

* Confidence Intervals
* Hypothesis Testing
* Statistical Modeling
* Fitting Models
* Bootstrapping (you’ll take a closer look at bootstrapping in the third subunit)

Top of Form

Save



2 - 3 Hours

45 Points

We’d like you to **complete sections four and five** on this LinkedIn Learning course: Introduction to Statistical Inference and Introduction to Statistical Modeling. Please complete the quizzes corresponding to these two sections.

You’ll learn how to do the following concepts from the Art of Statistics in Python:

* Confidence Intervals
* Hypothesis Testing
* Statistical Modeling
* Fitting Models
* Bootstrapping (you’ll take a closer look at bootstrapping in the third subunit)

Bottom of Form

3

### Case Study - Frequentist Inference: A and B

Save

3 - 4 Hours

64 Points

In this case study, you’ll learn more about frequentist inference. There are two parts to the case study. In part A, you’ll learn the Pythonic implementation of the concepts underlying frequentist inference. In Part B, you’ll apply those implementations to a real-world scenario.

You’ll be implementing the following concepts as you work through the case study:

* The z-statistic
* The t-statistic
* The Central Limit Theorem
* Estimation of the population standard deviation and mean, from a sample
* Sampling distribution of a test statistic, particularly for the mean
* Calculating confidence intervals and p-values
* Hypothesis testing

This assignment will require some degree of independence to complete. Please utilize your Google skills if you get stuck. If you're feeling like you can't find a path forward, feel free to reach out to your Slack community and your mentor.

**Submission Steps**

1. Please download [**the project files**](https://www.springboard.com/archeio/download/7b215cdf8aec4390a98371887e07f315/)
2. Complete Part A
3. Complete Part B
4. Add both completed parts to a folder in your GitHub repo
5. Submit a link to your folder here