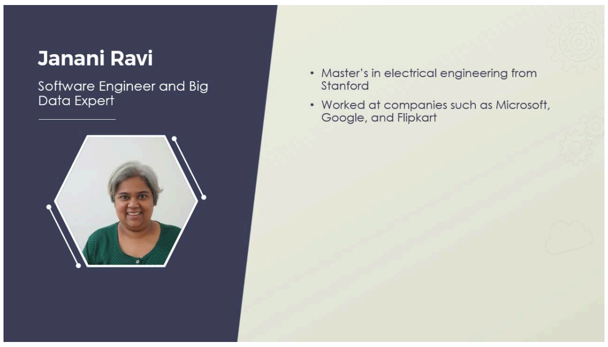
Statistical Analysis and Modeling in R: Working with Probability Distributions

Interpreting data is a core pre-processing step in data analysis and modeling. Use this course to practice using various dynamic statistical tools to explore and understand your data. During this course, you'll explore population distributions to model random variables, work with discrete and continuous probability distributions, and use discrete probability distribution types, such as the uniform, binomial, and Poisson distributions. You'll also examine continuous distributions, such as the normal and the exponential distributions. You'll round the course off by learning how to read and interpret QQ plots, which can be used to compare the distributions of two samples of data. When you're finished, you'll be able to use probability distributions to model events and understand your data.

Table of Contents

1. Video: Course Overview (it_dasamrdj_01_enus_01)



Janani Ravi
Software Engineer and Big Data Expert

- Master's in electrical engineering from Stanford
- Worked at companies such as Microsoft, Google, and Flipkart

In this video, you'll learn more about the course and your instructor. In this course, you'll explore statistical tools you can use to explore and understand your data. You'll learn population distributions to model random variables and work with discrete and continuous probability distributions. You'll learn the characteristic of discrete probability distributions such as the uniform distribution, the binomial distribution, and the Poisson distribution, as well as the normal distribution and the exponential distribution.

- *discover the key concepts covered in this course*

[Video description begins] *Topic title: Course Overview.* [Video description ends]

[Video description begins] *Your host for this session is Janani Ravi. She is a Software Engineer and a Big Data Expert.* [Video description ends]

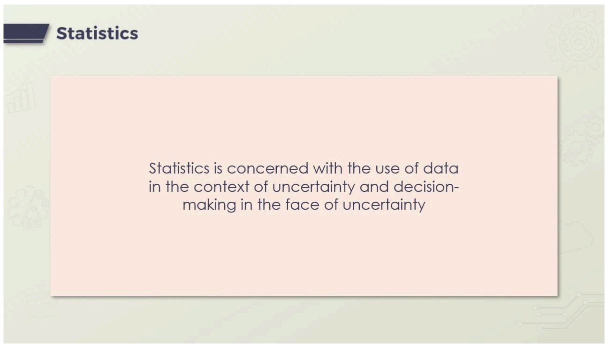
Hi, my name is Janani Ravi and welcome to this course, working with probability distributions using R.

A little about myself. I have a master's degree in electrical engineering from Stanford, and have worked at companies such as Microsoft, Google, and Flipkart. At Google, I was one of the first engineers working on real time collaborative editing in Google Docs, and I hold four patents for its underlying technologies. I currently work on my own startup Loonycorn, a studio for high quality video content. Understanding and exploring your data is a key preprocessing step in data analysis and modelling. An important technique used to understand data is to understand the probability distribution of your data. The probability distribution tells us the range of possible values for a variable, and how often those values occur. Probability distributions thus help us visualize and model uncertainty in a random variable.

[Video description begins] *Screen title: Learning Objectives.* [Video description ends]

In this course, you will explore statistical tools that you can use to explore and understand your data. You will understand the use of population distributions to model random variables and work with both discrete and continuous probability distributions. In this context, you will learn the characteristic of discrete probability distributions such as the uniform distribution, the binomial distribution, and the poisson distribution, as well as continuous probability distributions such as the normal distribution and the exponential distribution. You will round this course off by learning how to read and interpret QQ plots or quantile-quantile plots which can be used to compare the distributions of two samples of data. When you are finished with this course, you should be able to use probability distributions to model events and to understand your data.

2. Video: Statistical Tools for Understanding Data (it_dasamrdj_01_enus_02)



Statistics

Statistics is concerned with the use of data in the context of uncertainty and decision-making in the face of uncertainty

In this video, you'll learn more about statistics. Statistics is a mathematical body of science that pertains to the collection, analysis, interpretation, or explanation, and presentation of data. Statistics is about working with data, understanding your data, exploring your data, and interpreting results from your data. Statistics is concerned with the use of data in the context of uncertainty and decision making in the face of uncertainty.

- *recall the sets of statistical tools used to understand data*

[Video description begins] *Topic title: Statistical Tools for Understanding Data. Presented by: Janani Ravi.* [Video description ends]

We've all heard of statistics as a branch of mathematics, and we often use the term statistics loosely. But what exactly is the study of statistics? Statistics can be thought of as a mathematical body of science that pertains to the collection, analysis, interpretation, or explanation, and presentation of data. There is one thing that's absolutely clear here, is that statistics is all about working with data, understanding your data, exploring your data, and interpreting results from your data. But what do we do when we perform statistical analysis with data? Well, statistics is concerned with the use of data in the context of uncertainty and decision making in the face of uncertainty. Whenever we make decisions, whether it's in our personal lives or in our professional lives, we never have all of the facts needed at our disposal. We only have a sample or a subset of the facts to work with.

These limited facts are the equivalent of data. And statistics allows us to use this data and promotes decision making when things are uncertain, when everything isn't fully known. Let's talk about some of the steps that you would follow to apply statistics to a problem. You have a population that you want to study. Or there may be a process that you want to study. This is what you start with, the population as a whole, or a process. Now, the population could be anything. Maybe you want to conduct a study of all of the six year old children in the world. All of the citizens in a country. All individuals of the female gender who live in a certain town. Now, this population will differ based on the kind of study that you want to conduct.

The next step would be for you to compile data about the population. You'll basically try and survey everyone who's included in the population. You'll collect information that is relevant to what you want to study about the population and compile this data together. Next, you'll then summarize the details that you've collected about the population by using descriptive statistics. As the name implies, descriptive statistics are used to describe your data. They give you a quick overview of what your data looks like. You'll understand details like, what value is your data centered around? Do the values in your data jump around a lot? How are they spread out? These are descriptive statistics. Now, a question that comes up when you apply statistics to a problem is,

what if you don't have access to the population as a whole? If you're basically trying to study all of the six year olds across the world, you're not going to be able to access the data for the entire population. So what do you do in that case? Well, this is where you will choose a subset of data from the population. And this subset is referred to as a sample in statistics. Whenever you're working on a real world problem, you should accept as a fact that you won't have the data associated with the population as a whole. That's almost impossible to collect. You typically work with a sample, which is essentially a subset of the population. The sample contains data that you have access to, that you've managed to collect from the broader population. While performing statistical analysis, it's important that this sample that you're working with should be representative of the population as a whole.

For example, let's say you're running some kind of analysis on males and females. Let's say the population as a whole is comprised of roughly 50% males and 50% females. If the sample that you're working with contains 80% males and 20% females, well, that's a sample that is not representative of the population as a whole. Why does the sample need to be a representative sample? Well, typically when you sample from the population, you're going to use inferential statistics on the sample to draw meaningful conclusions about the population. By using inferential statistics on the sample, you're trying to figure out something about the population as a whole. Now, if your sample is not a representative one, any deductions you make about the population will be flawed.

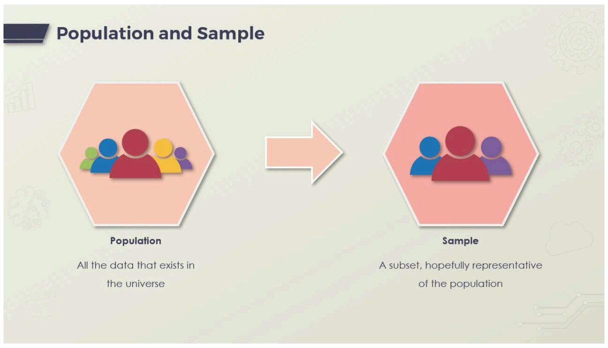
To understand the population from a sample, you need a representative sample. We've now discussed two sets of statistical tools. The first statistical tool is descriptive statistics, which is what we use to describe or summarize whatever data we have available. Descriptive statistics do not allow us to draw conclusions from the data, they only describe and give you an overview of your data. The second statistical tool that we've discussed is inferential statistics. Inferential statistics, as its name implies, allow us to draw inferences about the population using patterns that we've found in our sample in the data that we have to work with. Inferential statistics serve to connect the dots or make deductions about the population. I'll quickly summarize the nature of each of these statistical tools, starting with descriptive statistics. Descriptive statistics are all quantitative.

They quantitatively describe or summarize the data in a meaningful manner. Now, you're familiar with descriptive statistics and you've probably used it in your everyday life, the mean, mode, and median of your data. Those are measures of central tendency, range, variance, standard deviation, those are measures of dispersion. These are all descriptive statistics. All of the statistics that I just mentioned are used to describe data. Understand what value the data is centered around. Understand how the values in your data jump around. This description can then feed into other analysis. You can use the mean value of your data in other analysis as well. Descriptive statistics do not help you draw any conclusions about your data. Instead, you'll use these to get a big picture understanding of your data. In addition to the measures of central tendency and measures of dispersion. You can also summarize descriptive statistics using tables with your data, graphs and comments that you've added to your data.

All of these serve to describe your data. They are all descriptive statistics. When you compute descriptive statistics on your data, they do not assume that your data comes from a larger population. There is no fundamental assumption of the data being a sample of a larger population. Descriptive statistics are actually concerned with the actual properties of whatever data you're working with. Descriptive statistics exist to tell you, this is the data that you have. On the other hand, inferential statistics allow you to make propositions and draw inferences about a population. Propositions are essentially statements about your data that you need to evaluate to see whether they are true or false. You'll make propositions and then use your data to draw inferences about a population as a whole.

Now, the data that you work with for inferential statistics is considered to be a subset of a sample drawn from a larger population. An implicit feature of inferential statistics is that you know you don't have access to the entire population's details, and you can only work with a sample. And hopefully that sample is a representative sample. You'll examine the properties of your sample using different techniques. And then use the sample properties to make inferences or deductions about the population. In this manner, working with only a small sample of data, you can draw conclusions about a larger population. That's what inferential statistics is all about.

3. Video: Population and Sample Metric Comparisons (it_dasamrdj_01_enus_03)


Population and Sample
Population
All the data that exists in the universe
Sample
A subset, hopefully representative of the population

In this video, you’ll learn more about the terms population and sample. The term population refers to all the data that exists in the universe. It's impossible to work with data for the entire population. When you have data to work with, that data will be a sample. A sample represents a subset of the population. This subset is representative of the population as a whole.

- *compare and contrast population metrics with sample metrics*

[Video description begins] *Topic title: Population and Sample Metric Comparisons. Presented by: Janani Ravi.* [Video description ends]

When you perform statistical analysis, you'll often use the terms population and sample. In this video, we'll understand exactly what these terms mean. The term population refers to all the data that exists in the universe. Let's say you're studying plant matter, the population will refer to all of the plants that exist in the universe. If you're studying insects, the population refers to all insects. If you're studying cockroaches, the population refers to all of the cockroaches that exist in the universe. As you can imagine, the population number for anything that you want to meaningfully study will be huge. In reality, it's impossible for you to work with data for the entire population. When you have data to work with, that data will be a sample. A sample represents a subset of the population.

And hopefully, this subset is representative of the population as a whole. In just the previous video, we discussed the importance of the sample to be representative of the population. Only with a representative sample can you draw conclusions from the sample and apply those conclusions to the population as a whole. Let's describe the term population in a little more detail. The population refers to all of the data that we're interested in. Let's say we're performing some kind of analysis on the heights of individuals, the population will have the heights of all of the people in the world. In a sense, if you have population data, you have perfect data, you have everything there is to be known about your data. Now, the population can be small or large, the size is immaterial. All of the population examples I've given so far have referenced a very large population.

But let's say you were studying the six year olds in one particular school, the population then would be quite small. So remember, the population is just all of the data for whatever it is that you're studying. So the size of the population does not matter. Now, it's possible for you to apply descriptive statistics to population. Descriptive statistics will give you the measures of central tendency and measures of dispersion for a population. For example, let's say you want to compute the mean, median, and standard deviation on the population as a whole. These values are referred to as parameters. So when you apply descriptive statistics to a population, whatever values you get, they are referred to as population parameters. Applying descriptive statistics to a population will give you parameters and these parameters represent the population, and not a subset of the population.

Now, this term is important because I'm just going to introduce another term for when you apply descriptive statistics to a sample. So remember, descriptive statistics on populations are called parameters and not statistics. An easy way to remember this is when you use the term statistic, there is a measure of uncertainty involved in that term. When you have the population as a whole and you compute descriptive statistics on a population, they are parameters, not statistics. With parameters, there is no uncertainty because parameters represent every point of data that we're interested in. The average height of six year olds in a particular school, if you have access to all of the six year olds, you measure their heights and compute the average, that is the parameter of the population of six year olds in that school. Even though the size of the population doesn't matter, when you're performing real world analysis, you're unlikely to have access to the entire population.

You may not have access to all of the data that you're interested in the real world. There can be many reasons for this. Firstly, it may not be feasible to measure values across the entire population. How will you access every individual that you're interested in and get the data that you want from them? Even if you don't have access to all of the data in the real world, you still need to make decisions, you still need to analyze your data. So your way out is to work with a smaller subset of data, that is a sample of data. And this is a sample that is drawn from the larger population. If you're working with a sample, you can compute descriptive statistics on a sample.

Values of the descriptive statistics, such as mean, median, and standard deviation, when computed on a sample, are referred to as statistics. Here, they're statistics, not parameters. These are statistics and not parameters because there is a measure of uncertainty involved. These statistics are estimates of the actual population parameters. The heights of six year olds all around the world, you may only have access to six year olds in a particular school. And then you'll compute the average height of a six year old in that school and use that as an estimate of the average height of six year olds across the world. Here is an important term that you need to remember. When you compute descriptive statistics on samples, these are referred to as statistics and not parameters.

Now here it becomes very important that the sample that you're working with represents the population. If you're working with a non-representative sample, conclusions drawn from the sample may not apply to the population. And finally, we know that statistics are used to make inferences or generalizations about the population as a whole. From the population, we somehow need to get a sample of data to work with. And the way you do this is by using sampling techniques. Sampling techniques refer to the methodology that you use to select a representative subset of data from the population. There are many different sampling techniques that you could use to get a sample from the larger population. And each technique will have its own strengths and weaknesses. We'll discuss two broad category of techniques. The first is probability sampling.

Probability sampling uses randomization to ensure that every element of the population gets a chance to be a part of the selected sample. The intuitive idea behind probability sampling is if you pick at random, every element has some chance of getting picked and being part of your sample. Another sampling technique is called non-probability sampling. This relies on the researcher's ability to select elements for a sample. Because non-probability sampling relies on an individual and it's not random, it's more likely to be biased. So the sample that you get is likely to be a biased representation of the population.

4. Video: Characteristics of Probability Distribution Types (it_dasamrdj_01_enus_04)

In this video, you'll learn more about probability distributions. Probability distribution is a mathematical function that shows the possible values for a variable and how often those values occur. The variable refers to your data. Your variables have their own range of values, and different values occur with different frequencies or different probabilities of occurrence. A probability distribution is a mathematical function that gives the probabilities of occurrence of different possible outcomes for an experiment.

- *recall the characteristics of discrete and continuous probability distributions*

[Video description begins] *Topic title: Characteristics of Probability Distribution Types. Presented by: Janani Ravi.* [Video description ends]

If you're performing statistical analysis on your data, it's important that you understand probability distributions. You need to understand what probability distributions are, what they represent about your data, and what you can learn from your data if your data is drawn from a certain probability distribution. So what exactly is a probability distribution? It's just a mathematical function that shows the possible values for a variable and how often those values occur. The variable here refers to your data, whether it be the heights of individuals, the scores in mathematics for students of a university, the temperatures at a particular location around the year. Each of these variables have their own range of values, and different values occur with different frequencies or have different probabilities of occurrence.

Another way to describe a probability distribution is, in probability theory and statistics, a probability distribution is the mathematical function that gives the probabilities of occurrence of different possible outcomes for an experiment. You're conducting an experiment and a result can be one of possible outcomes. If you have a function that gives you the probability of occurrence of individual outcomes, that is a probability distribution. We'll use some visualization to cement your understanding of probability distributions in just a bit. But before we get there, let's understand some of the properties of probability distributions. Let's say p(x) represents the likelihood that a random variable has a particular value equal to x. You have some random variable, p(x) is the probability that the value of that variable is equal to x. Now the probability p(x) for any value has to be between 0 and 1.

Negative probabilities and probabilities greater than 1 are meaningless. If you want to represent this in terms of percentages, 0 represents 0%, 1 represents a 100%. So your probability has to lie between 0 and 100%. If you take the probabilities of all of the individual values for your variable, and then sum it all up, the sum of the probabilities should be equal to 1. This can be mathematically represented by sigma of p(x) for all possible values of x. And this should be equal to1. Probability distributions fall under two broad categories or types. The first kind are discrete probability distributions for discrete variables. Discrete variables are also referred to as categorical variables. These are variables that can only take on a discrete subset of values, heads or tails, boy or girl, red, green, or blue.

All of these are examples of a discrete variables. Probability distributions for such a discrete variables are referred to as discrete probability distributions. And then you have variables that can take on continuous values, any value within a range. Probability distributions for such variables are called continuous probability distributions. The price of a car this can vary from 0 to infinity or if not infinity certainly a very large number and the car's price can be any value within that range. This is an example of a continuous variable. The height of an individual can be anywhere between 0 and 7 feet tall. That's also an example of a continuous variable. Let's discuss discrete probability distributions first. These are probability distributions for variables that have discrete or categorical values.

The mathematical function used to represent a probability distribution for a discrete variable is referred to as a probability mass function. So, note this term, this will be different for continuous variables. Examples of discrete variables are coin tosses, the number of people standing in a queue at any point in time, star ratings for restaurants. All of these are categorical variables. In the case of discrete distributions, if you sum up the probabilities of all possible discrete values, that should be equal to one. So if you think of the probability of your coin landing heads or tails that has to sum up to one. There is a no other possibility. It has to be heads, or tails. Let's discuss some of the most common types of discrete distributions.

We'll be working more with these distributions when we come to the demos in this learning path. The first is the binomial distribution that is used to model binary data. So when you only have two possible outcomes, male or female, head or tails, success or failure, all of these are modeled using the binomial distribution. Another discrete distribution is the Poisson distribution used to model counts. Whenever you have the counts for anything, the number of visits to your website in an hour. The number of people standing in the queue. The number of calls to your customer service hotline. The number of advertisements that you see on TV when you watch for an hour. All of these are modeled using the Poisson distribution. And finally, another discrete probability distribution is the uniform distribution that is used to model multiple events that have the same probability of occurrence. Let's say you're tossing a dice, a dice has six faces.

And if it's a fair dice, any of these faces can come up with equal probability. That is a uniform distribution when all outcomes have an equal probability of occurring. From discrete distributions, let's move on to discussing continuous probability distributions. A continuous variable is one where the variable can take on any value within a range. The heights and weights of individuals are examples of continuous variables. The probability function for a continuous distribution is referred to as a probability distribution function. So when you hear the term probability distribution function, your variable should be a continuous variable. Keep this in mind.

Examples here are heights and weights of individuals, scores on a test, blood pressure, all of these are continuous variables. Once again, even in the case of continuous distributions, the sum of probabilities of all values of your variable is equal to one. The most common of all continuous distributions that we study in statistics is the normal distribution. The normal distribution is widely used because it occurs so often in nature. There are other continuous distributions that exist as well. The weibull and lognormal distributions can be used to represent skewed data. Skewed data is data that is not symmetric around a central value, where extreme values are more likely to occur. For the rest of this video, let's focus our attention on understanding the normal distribution.

[Video description begins] *A Normal Distribution graph appears on the screen. It has four gradient lines for different sigma values. The Y-axis values range from 0.0 to 1.0. The X-axis values range from -5 to 5.* [Video description ends]

This distribution occurs widely for natural or physical properties.

On the x axis we represent the possible values for the continuous variable. And on the y axis, we represent the probability that value occurs. A normal distribution is defined using two parameters, the mean or the center of the distribution, and the standard deviation, which tells us how narrow and tall or short and flat the curve is. This image shows you four different normal distributions with different means, and different standard deviations. Why do we discuss the normal distribution over and over again in statistics? That's because this is the probability distribution for many natural phenomena. The normal distribution, as you can see, is a symmetric curve and the mean value of your variable specifies the center of the distribution. In a normal distribution, the highest point in the density curve is the mean of your variable. The standard deviation specifies the spread of the distribution, how tall or flat this curve is.

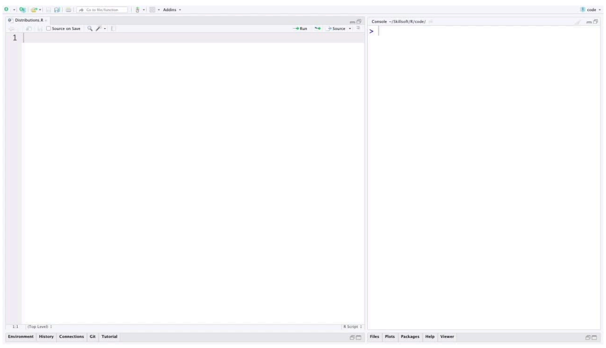
In a normal distribution, the mean, mode, and median of your data are all equal. The mean is the average of your data. The mode is the most commonly occurring value, and the median represents the data point exactly at the center of your data at the 50th percentile. The curve of the normal distribution is shaped like a bell. And the normal distribution curve is often referred to as a bell curve. This curve is symmetric around the mean. 50% of the values in the distribution are to the left of the center, and 50% of the values are to the right of the center. And as is the case for any probability distribution, the total area under the curve is equal to 1. This is where we sum up the probability of occurrence of all possible values.

Here is a curve for a normal distribution centered around a mean of 0. So, the highest point of this curve corresponds to the value 0 on the x axis.

[Video description begins] *Another Normal Distribution graph appears on the screen. It has only one curve inclined at the center at 0. The Y-axis values range from 0.0 to 0.4. The X-axis values range from -3 to 3.* [Video description ends]

This curve basically tells us that values close to the mean are the most likely to occur. As we move away from the mean, the values are less likely to occur. Distance from the mean can be represented using standard deviations, as you can see along the x axis here. I'm going to have arrows highlight the portion of the curve that is one standard deviation away from the mean to the left of the center, as well as to the right of the center. In a normal distribution 34.1 + 34.1, 68.2% of all of the data points fall within one standard deviation of the mean. If you look one more standard deviation away from the mean, you can see that around 95% of the values lie within two standard deviations of the mean. 2 multiplied by 34.1 plus 2 multiplied by 13.6. And finally, 99.7% of the values lie within three standard deviations of the mean. These percentage values that I've just discussed apply to all normal distributions. No matter what the mean and what the standard deviation.

5. Video: Sampling and Analyzing Uniform Distribution Data (it_dasamrdj_01_enus_05)



In this video, you'll learn more about RStudio to write R code. You'll see RStudio is a freely available integrated development environment or IDE for R. RStudio includes a console, a syntax-highlighting editor that supports direct code execution. RStudio also has tools for plotting history, debugging, workspace management, and so on. You'll discover RStudio Desktop is freely available for anyone to download and use. You'll need to download RStudio on your computer.

- *sample and analyze data that follows uniform distribution*

[Video description begins] *Topic title: Sampling and Analyzing Uniform Distribution Data. Presented by: Janani Ravi.* [Video description ends]

In this learning path, I assume that you're familiar with basic R programming and you know how to manipulate and transform

[Video description begins] *An RStudio IDE appears on the screen. It is divided in two parts. The first part, on the left has the code editor. It has a tab named Distributions.R. On the right, a Console pane is open. Its path is: ~/Skillsoft/R/code/.* [Video description ends]

a data stored in R data frames or tibbles. For all of the demos in this learning path, we'll use RStudio to write R code. RStudio is a freely available integrated development environment or IDE for R. RStudio includes a console, a syntax highlighting editor that supports direct code execution. RStudio also has tools for plotting history, debugging, workspace management and so on. The RStudio Desktop is freely available for anyone to download and use. I already have RStudio downloaded and installed on my local machine and I have the R programming language running.

The version of R that I'm using is version 4.0.3, which is the latest version at the time of this recording. If you have a newer version of R, then all of the programs that we write should work just fine. Earlier in this learning path, when we spoke of the different categories of statistical tools that we can use with our data, descriptive statistics, and inferential statistics, we also discussed that we often don't have the data associated with the entire population. We only work with a sample of data. And an important attribute of the data sample that we need to consider before we perform statistical analysis is the probability distribution of the data. The probability distribution, as we discussed, is the mathematical function that gives us the probability of occurrence of different possible outcomes of an experiment. We know that one experiment can have different possible outcomes or results. And we want to plot the probability of each of these results.

That's what a probability distribution is. And probability distributions are what we are going to explore here in this demo. Within RStudio, I have a new R script called Distributions.R which I have created and saved in a folder on my local machine. My current working directory is within the Skillsoft/R/code directory. You can see this in the top right. That's where my Console is pointing. I'll write code using the R script file on the left side of my screen. And the result of executing this code will be available in the Console output on the right side of my screen. Now, before I start any script, I'm going to invoke the function, rm(list = ls()).

[Video description begins] *She adds a code in line 1. It reads: rm(list = ls()).* [Video description ends] The result of executing this line of code is basically to get rid of any R objects that currently exist in my R programming memory.

This allows me to start the code in my script afresh older objects won't be present. If you're working on a Windows machine, hit Ctrl+Enter to execute this line of code. On a Mac machine, hit Cmd+Enter. When we work with data, it's important that we be able to visualize the data and visualize any relationships that exist in the data.

[Video description begins] *She adds a code in line 3. It reads: install.packages("ggplot2").* [Video description ends]

For this, we need a plotting library, and the plotting library that we'll be using in this learning path is ggplot2. ggplot2 is the most modern plotting library available in R. And it makes plotting very easy, allowing us to layer complex plots using components. I'm going to go ahead and install the ggplot2 package on my local machine. Now it so happens that I already have ggplot to installed. So in order to reinstall this, I need to restart my R kernel.

Click on Yes and your package will be downloaded and

[Video description begins] *A prompt titled Updating Loaded Packages appears. At the bottom, Cancel, No, and Yes buttons are present.* [Video description ends]

installed within your current local R programming environment. We'll now include this package in this current program by invoking the library function.

[Video description begins] *She adds a code in line 4. It reads: library(ggplot2).* [Video description ends]

This will allow us to invoke the functions that ggplot2 has to offer from our R program. Let's now sample some data points from a uniform distribution. This can be done using the runif function. This is part of the stats package in R which ought to be installed by default when you set up R along with RStudio. The uniform distribution is a discrete probability distribution used to model

[Video description begins] *She adds a code in line 6. It reads: uniform.dist.data <- data.frame(val = runif(10, 1, 100)).* [Video description ends]

discrete or categorical variables. These are variables that can only take on values from a discrete, prespecified set.

This discrete probability distribution is one where all outcomes have the same probability. Let's say we have an event that has three possible outcomes. If all three outcomes have the same probability of occurrence, the variable is set to have a uniform distribution. Let's take a look at this runif function in R which generate data points that belong to a uniform distribution. Now runif here has taken in three input arguments. You can see this on line 6. The first input argument 10 gives us the number of data points that we want to generate. The second input argument 1 is the mean range of the values that the data points can take out of the possible outcomes. And 100 is the max of the range of the values that the data can take on. The way that runif function works in R is it will not generate either of the extreme values.

Here, the extreme values are 1 and 100 unless max is equal to min. Here, that's not the case or max minus min is very small compared to min. And in particular for the default specifications of the other input arguments to runif, the extreme values are unlikely to be generated. Here, I've created a dataframe, which contains 10 data points sampled from a uniform distribution. Let's see what this data looks like. The easiest way to view the distribution of this data is to visualize it as a histogram.

[Video description begins] *She adds a code in line 8. It reads: hist(uniform.dist.data$val).* [Video description ends]

The histogram is a univariate visualization that allows us to view the shape of our data. Now, we see a histogram pop up at the bottom right of our screen.

[Video description begins] *A Plots pane opens at the bottom-right corner. It has a toolbar with the following options: Zoom, Expert, Publish, etc. Below it, there is a histogram. The y-axis shows the frequency from 0-4. The x-axis shows the uniform.dist.data$val from 0-100.* [Video description ends]

Click on the zoom icon there just above the plot and that will allow you to view this histogram on full screen. Let's take a look at the x-axis of this histogram.

You can see that the x-axis ranges from 0 to 100. The possible outcomes or the possible values for our data lie in this range. You can see that the plot has divided the range into buckets, 0 to 20, 20 to 40, 40 to 60, and so on. Now, we generated 10 data points sampled from a uniform distribution. The height of the bars of the histogram give us the count of data points that fall in each of these buckets. So there are four data points in the range 0 to 20, 1 data point in the range 20 to 40, and so on. Now, the data doesn't really seem to be uniformly distributed. There are different counts of records in each of these buckets. Well, that's because we have just 10 data points. Let's go back to our code and plot this uniform distribution using ggplot. That's the plotting mechanism that we'll use for most of this course.

[Video description begins] *She adds a code in line 9. It reads: ggplot(uniform.dist.data, aes(x=val)) + geom_histogram().* [Video description ends]

We invoke the ggplot function, pass in our uniformly distributed data.

We pass in the aesthetic mapping. The x-axis corresponds to the value of the data points and we want to view this data in the form of a histogram + geom_histogram().

[Video description begins] *Another histogram appears in the Plots pane. The y-axis shows the count from 0.0-2.0. The x-axis shows the val from 0-100.* [Video description ends]

Now ggplot uses different default values for the buckets on the x-axis, which is why our plot is a little different from what we saw earlier with the hist function.

[Video description begins] *She clicks the Zoom button. The histogram appears in full-screen.* [Video description ends]

When we sample our data from a uniform distribution, we would expect that every outcome, that is, every value in this range has an equal probability of occurrence. That would mean that all of the bars that you see here in this histogram should have equal heights, but that's clearly not the case. And the reason for that is we've only generated 10 data points. Let's change this, I'm going to use the runif function once again. And I want to generate data points from a uniform distribution in the range 1 to 100.

But this time around, I'm going to generate a 100 data points. 100 is the first input argument that I've passed in to runif on line 11.

[Video description begins] *She adds a code in line 11. It reads: uniform.dist.data <- data.frame(val = runif(100, 1, 100)).* [Video description ends]

That's the main change here in my code. The first input argument for number of data points is now 100. Let's go ahead and plot the histogram of this uniformly distributed data. So we have more data points and values in the range 1 to 100. I'll use ggplot for this.

[Video description begins] *She adds a code in line 12. It reads: ggplot(uniform.dist.data, aes(x=val)) + geom_histogram(). Another histogram appears in the Plots pane. The y-axis shows the count from 0-8. The x-axis shows the val from 0-100.* [Video description ends]

And you can see that this histogram is a little more dense.

[Video description begins] *She clicks the Zoom button. The histogram appears in full-screen.* [Video description ends]

The x-axis ranges from 0 to 100. And we have many more bars here because we have more data points, so we have more records in each bucket represented. And the bars are still not of equal height or almost equal height. A 100 data points is clearly still a small number.

[Video description begins] *She adds a code in line 14. It reads: uniform.dist.data <- data.frame(val = runif(100000, 1, 100)).* [Video description ends]

Distributions are really much clearer when you have a large sample of data, that is a large number of data points in your sample.

I'm going to invoke the runif function. I want data to be generated from a uniform distribution, the range 1 to 100, but I want a 100000 data points. Notice 100000 as the first input argument on line 14. We have a much larger number of data points here. Hopefully, the uniform nature of our distribution will become clearer when we plot this data.

[Video description begins] *She adds a code in line 15. It reads: ggplot(uniform.dist.data, aes(x=val)) + geom_histogram(). Another histogram appears in the Plots pane. The y-axis shows* *the count from 0-3000. The x-axis shows the val from 0-100.* [Video description ends]

And that is indeed the case.

[Video description begins] *She clicks the Zoom button. The histogram appears in full-screen.* [Video description ends]

If you look at the bars in this histogram, you can see that in the range 0 to 100, all of the bars have almost the same height. This means the number of data points in each bucket is almost the same, which in turn means that the probability of occurrence of each value in the specified range 1 to 100 is almost exactly the same. It's actually exactly the same for a uniform distribution. But because this is a sample that we're working with, you know that it's almost the same and our data is uniformly distributed.

Let's run this once again with 1000000 data points.

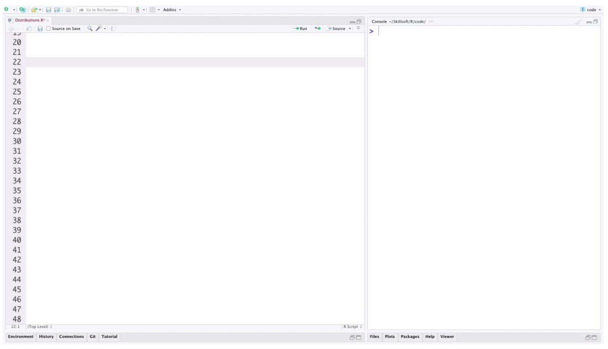
[Video description begins] *She adds a code in line 17. It reads: uniform.dist.data <- data.frame(val = runif(1000000, 1, 100)).* [Video description ends]

Observe that the first input argument on line 17 is 1000000 data points. That's the number of points we'll generate. Larger the size of our sample, easier it is for us to view the uniform nature of this data. Observe the bars in this histogram.

[Video description begins] *She adds a code in line 18. It reads: ggplot(uniform.dist.data, aes(x=val)) + geom_histogram(). Another histogram appears in the Plots pane. The y-axis shows the count from 0-30000. The x-axis shows the val from 0-100.* [Video description ends]

Let's head over to full screen. You can see that the bars are almost the same height. The probability of occurrence of the values 1 to 100 in our data is almost exactly the same for each value. You can see clearly that this data is uniformly distributed.

6. Video: Sampling and Analyzing Binomial Distribution Data (it_dasamrdj_01_enus_06)



In this video, you’ll learn more about probability distributions. You’ll learn another discrete probability distribution, the binomial distribution. The binomial distribution summarizes the probability that the outcome will be one of two independent values. You’ll learn a variable that is binomially distributed can have only two possible outcomes, heads or tails, true or false, here or there. These two outcomes must be independent. One outcome should not be more possible because of the previous outcome.

- *sample and analyze data which follows binomial distribution*

[Video description begins] *Topic title: Sampling and Analyzing Binomial Distribution Data. Presented by: Janani Ravi.* [Video description ends]

[Video description begins] *The RStudio IDE appears on the screen. In the code editor, a tab named Distributions.R is open. On the right, a Console pane is open. Its path is: ~/Skillsoft/R/code/.* [Video description ends]

After having understood the uniform distribution, you're now ready to move on to other probability distributions. In this video, we'll explore another discrete probability distribution, the binomial distribution. The binomial distribution summarizes the likelihood or probability that the outcome will be one of two independent values. Now the most important detail to note here is a variable that is binomially distributed can have only two possible outcomes, heads or tails, true or false, here or there. Another detail to observe here is that these two outcomes need to be independent. One outcome should not be more possible because of the previous outcome. For example, when you toss a coin heads or tails, this is by binomially distributed data.

The easiest way to imagine a binomial distribution is to consider coin tosses. We know that heads and tails are both independent outcomes or independent events. Another way to define a binomial distribution, is that, it describes the outcome of N independent trials in an experiment. Independent trials in that, one trial does not depend on the previous one, and does not depend on any future trials. Each trial can only have two possible outcomes, success or failure. The probability distribution of the N independent trials that you conduct will be a binomial distribution. As you can see here, this is basically, N coin tosses. Each coin toss can be heads or tails. Each coin toss is independent of previous coin tosses and future coin tosses. You can generate samples from a binomially distributed data using R's rbinom function.

This function is also available in the stats package in R that is installed by default when you set up and work with R.

[Video description begins] *She adds a code in line 22. It reads: rbinom(20, size = 1, prob = 0.5).* [Video description ends]

Before we move on to how the rbinom function works and how it's invoked there is one more term that you should be familiar with when we are working with binomial distributions, and that is the Bernoulli trial. Bernoulli trials refer to random experiments that can have exactly two possible outcomes. As you can see, a Bernoulli trial is nothing but a binomial trial. Coin tosses are essentially Bernoulli trials. Every toss of a coin can produce only one of two possible outcomes, heads or tails. Variables that can be used to represent the outcomes of a Bernoulli trials are binomially distributed. Now that you understand all of these terms, let's turn our attention to the rbinom function.

The rbinom function generates values or outcomes that are outputs of Bernoulli trials. Trials that can have one of two possible values, win or lose, zero or one. Here when we invoke the rbinom function, observe the input arguments we pass in. This is on line 22. The first input argument is 20. This input argument refers to the number of observations in our trial. The number of observations here refers to the number of times we flip our coin or coins. 20 here indicates that we flip our coin 20 times one after the other. The second input argument, which is size = 1 refers to the number of trials per observation. This refers to the number of coins that we flip for each observation. size = 1 means we flip exactly one coin for each observation.

So one coin flip 20 times. The last input argument here, 0.5, refers to the probability of success. Let's say success means your coin turns up heads. What is the probability of success? Here observe that we made the assumption that this is a fair coin. The probability of success is 0.5, which means the probability of failure is also 0.5. Let's run this code by hitting Cmd+Enter or Ctrl+ Enter, and see the outcomes of 20 coin flips. We flipped one coin 20 times and you can see the outcome here within the console window. Of our 20 observations, certain flips had the coin turn up heads, let's say that's 1. Other flips had the coin turn up tails, that's 0. There's an almost even split between heads and tails.

That's because the probability of success we've specified as 0.5 or 50%. Coin tosses are an easy binomial experiment to imagine. Let's consider a different one. Let's say you have a factory which makes 1,000 widgets per day.

[Video description begins] *She adds a code in line 24. It reads: rbinom(10, size = 1000, prob = 0.03).* [Video description ends]

There is a 3% chance that the widgets are defective. How many widgets will be rejected each day for the next 10 days? When you first hear of the scenario of this factory producing widgets, it doesn't seem similar at all to coin tosses. But it is, let's see how. Now we can toss any number of coins. A factory can produce any number of widgets. Here, this factory produces 1,000 widgets. Now every widget can have only two possible outcomes. It has been successfully produced and it's a good widget or there is a problem with it and it will be rejected.

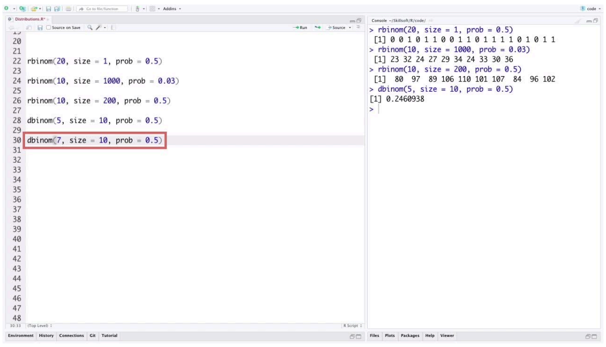
The number of observations here will be the number of days for which you want the reject data. The number of these over which you want to model the number of rejected widgets. Let's take a look at the rbinom function here with this factory producing widgets example. We invoke rbinom on line 24. The first input argument is 10. This is equal to the number of observations. We want to observe the factory's output over 10 days, which is why I specified 10 as the input argument. How many widgets does the factory produce in a single day? 1,000 which is why size is set to 1000.

The number of trials per observation is the number of widgets produced per day. Remember one day is one observation. Now the third input argument I've previously specified as the probability of success, but success is basically what you want to model. Here I want to model the number of widgets rejected. The probability that a widget will be rejected is 0.03. So here, I'll define success as knowing the number of widgets that are rejected. So probability is 3%. Let's go ahead and hit Enter, and you can see that roughly 20 to 40 widgets will be rejected per day. Obviously, when you run your code, your output will be a little different, but the overall range and values should be very similar. This is binomially distributed data. I'm now going to invoke the rbinom function once again and I'm going to go back to my coin tosses example.

[Video description begins] *She adds a code in line 26. It reads: rbinom(10, size = 200, prob = 0.5).* [Video description ends]

The first input argument here to the rbinom function is 10. Remember, 10 is the number of observations. This means that I'm tossing whatever number of coins I have 10 times. So I'm conducting my experiment 10 times. Next input argument is size = 200. The size tells me how many coins I actually toss. So for one observation, how many coins do I toss? 200 coins, and finally, the probability is 0.5. The probability of success, that is a coin turning up heads. That's what we'll assume as success here. We can also assume a coin turning up tails as success is 50%. When you run this code, you'll get 10 values at your output. Each value will tell you the number of coins out of 200 that turned up heads for that observation. So in the first observation 80 coins were heads out of 200. Second observation 97 then 89, 106, 110, and so on. You can see that because the probability of success is 50% at any point, roughly 100 coins turn up heads.

7. Video: Computing Probabilities in Binomial Distributions (it_dasamrdj_01_enus_07)



```
22 rbinom(20, size = 1, prob = 0.5)

24 rbinom(10, size = 1000, prob = 0.03)

26 rbinom(10, size = 200, prob = 0.5)

28 dbinom(5, size = 10, prob = 0.5)

30 dbinom(7, size = 10, prob = 0.5)
```

```
> rbinom(20, size = 1, prob = 0.5)
 [1] 0 0 1 0 1 1 0 0 1 1 0 1 1 1 1 0 1 0 1 1
> rbinom(10, size = 1000, prob = 0.03)
 [1] 23 32 24 27 29 34 24 33 30 36
> rbinom(10, size = 200, prob = 0.5)
 [1]  80  97  89 106 110 101 107  84  96 102
> dbinom(5, size = 10, prob = 0.5)
[1] 0.2460938
>
```

In this video, you'll learn more about computing probabilities in binomial distributions. You'll remember the rbinom function from the earlier video gives you data points generated from a binomial distribution with certain parameters. The dbinom function in R also works with binomial distributions. This is the function that gives you the density. You'll look at an example here.

- *calculate probabilities of events in the binomial distribution*

[Video description begins] *Topic title: Computing Probabilities in Binomial Distributions. Presented by: Janani Ravi.* [Video description ends]

The rbinom function that we saw in the earlier video gives us data points generated from a binomial distribution with certain parameters.

[Video description begins] *The RStudio IDE appears on the screen. In the code editor, a tab named Distributions.R is open. On the right, a Console pane is open. Its path is: ~/Skillsoft/R/code/.* [Video description ends]

The dbinom function in R also works with binomial distributions. This is the function that gives us the density. Let's understand what that means by considering an example.

[Video description begins] *She adds a code in line 28. It reads: dbinom(5, size = 10, prob = 0.5).* [Video description ends]

Here I invoke the dbinom function. What we are trying to figure out, is if we flip a fair coin 10 times, what is the probability of getting exactly five heads? Out of 10 flips, we want the probability of getting exactly 5 heads? The first input argument 5 is the number of heads or successes. Size is equal to 10, means we are flipping that coin 10 times, that's a way to think about it. Probability is equal to 0.5 is the probability of getting your coin to show up as heads, that is 50%.

The dbinom function will give us a single probability score as the result, 0.246. If you flip a fair coin 10 times, you have exactly a 24.6% chance of getting 5 heads. Based on our probability input argument for each toss, the chance that the coin will show up heads is 50%. Let's use the dbinom to compute the probability of hitting

[Video description begins] *She adds a code in line 30. It reads: dbinom(7, size = 10, prob = 0.5).* [Video description ends]

exactly 7 heads out of 10. So we're going to flip a coin 10 times, size is equal to 10 and I want exactly 7 heads. That is the first input argument. It's a fair coin, the probability of having our coin show up heads is 0.5 or 50%. And when you execute this, you'll see that this is about 11.7% or 0.117. You can see from the result here that the probability of getting exactly 7 heads is less than the probability of getting exactly 5 heads, 11.7% versus 24.6%.

The dbinom function can accept a sequence as an input argument as well. This allows us to compute the probabilities for a number of different outcomes with one invocation.

[Video description begins] *She adds a code in line 32. It reads: dbinom(0:7, size = 10, prob = 0.5).* [Video description ends]

Here I am want to know the probabilities of getting exactly 0 heads, exactly 1 head, exactly 2 heads, exactly 3 heads, all the way up to 7 heads for 10 coin flips. Once again, this is a fair unbiased coin, the probability of heads is 50%. Once I execute this bit of code, you'll see that I'll get 8 probability scores in my result. The probability of getting 0 heads across 10 coin flips is 0.0009. That is the first probability score in our result list. That's a really low score, because that means that all 10 flips should show up tails. Then we can see that the probability of getting exactly 7 heads is 0.1171. This is a score that we've computed before. That is the last score in our results list.

Another variant of the binom function is the pbinom, which gives us the probability distribution. This allows us to compute cumulative probabilities. Here once again, we are flipping a fair coin 10 time size is equal to 10. We want to know the probability of getting 5 or fewer heads. This will be the sum of the probability that we get 0 heads plus the probability that we get 1 head, the probability that we get 2 heads, up to the probability that we get 5 heads.

[Video description begins] *She adds a code in line 34. It reads: pbinom(5, size = 10, prob = 0.5).* [Video description ends]

Let's go ahead and execute this code. And let's see the probability of getting 5 or fewer heads in 10 coin flips of a fair coin. And you can see that it is 0.62, or roughly 62%. One last example here using the pbinom function. We want to know for 10 coin flips, the probability of getting 7 or fewer heads.

[Video description begins] *She adds a code in line 36. It reads: pbinom(7, size = 10, prob = 0.5).* [Video description ends]

Now remember, 7 or fewer heads should have a higher probability than 5 or fewer heads. Remember the pbinom computes cumulative probability scores. And you can see that the result here is 0.945 or 94.5%. We are now ready to move on to using ggplot to visualize binomially distributed data.

[Video description begins] *She adds a code in line 40. It reads: num.heads <- 0:20.* [Video description ends]

We'll continue using coin tosses as an example because that's by far the easiest to imagine. Let's assume that the number of heads possible is a sequence from 0 to 20. So I initialize the num heads variable as a vector containing the values 0 to 20. Both inclusive. Now I'll use the dbinom function to generate probability values that we get a certain number of heads from 0 to 20, for a certain number of coin tosses.

Observed that I've specified size is equal to 20 on line 41 here

[Video description begins] *She adds a code in line 41. It reads: binom.dis.data <- data.frame(val = dbinom(num.heads, size = 20, prob = 0.5)).* [Video description ends]

for my invocation of the dbinom function. I'm going to toss a fair coin probability is equal to 0.5, 20 times. And I'm going to compute the probability that 0 flips will show up heads, 1 flip will show up heads, 2 flips will show up heads, up to all 20 flips showing up heads. That is the sequence I've specified in the num heads variable. This will generate 21 different probability scores that I'll store in a dataframe called binomial distributed data. I'll then use ggplot to visualize this data in the form of a line chart. So on the x-axis, we'll have the number of heads from 0 to 20.

[Video description begins] *She adds a code in line 43. It reads: ggplot(binom.dis.data, aes(x=num.heads, y=val)) + geom_line().* [Video description ends]

On the y-axis, we'll have the probability that our 20 coin flips turned up that number of heads. And I'm going to plot this density function as a line plot. And there you can see the plot of my binomially distributed data.

[Video description begins] *A plot graph appears in the Plots pane. The y-axis shows the val and the x-axis shows the num.heads.* [Video description ends]

Viewing this and full screen, you can see that the x-axis corresponds to the number of heads that we see for 20 coin flips. The y-axis corresponds to the probability that we see that number of heads. Because this is a fair coin, the probability of seeing 10 heads is the highest point in our density curve here. Let's now visualize the binomial distribution for a slightly different sequence.

[Video description begins] *She adds a code in line 45. It reads: num.heads <- 300:700.* [Video description ends]

I will compute the probability scores that we have 300 to 700 showings of heads for our coin tosses. So, how many times do we toss our coin? Let's take a look at the dbinom function invocation here on line 46.

[Video description begins] *Code line 46 reads: binom.dis.data <- data.frame(x = dbinom(num.heads, size = 1000, prob = 0.5)).* [Video description ends]

I have specified size is = 1000, which means we'll toss a coin a 1000 times. Probability is equal to 0.5, indicating this is a fair coin. The dbinom function will compute probability scores, for 300 heads, 301 heads, all the way to 700 heads.

Let's take a look at this data using ggplot.

[Video description begins] *She adds a code in line 48. It reads: ggplot(binom.dis.data, aes(num.heads, y=x)) + geom_line().* [Video description ends]

I'm going to use the geom-point layer to plot this probability density curve. On the x-axis we'll have num heads. On the y-axis, we'll have the actual probabilities for that number of heads. Let's do this in full screen.

[Video description begins] *Another plot graph appears in the Plots pane. The y-axis shows the x and the x-axis shows the num.heads.* [Video description ends]

And here you can see that the highest probability of about 0.025, that is 2.5%, is for 500 heads out of a 1,000 tosses. This should not be surprising, because our coin is a fair coin. Now, so far in all of our binomially distributed data, we assume that each outcome is equally likely. For a coin toss, we assumed a fair coin. The probability of heads was 50%, the probability of tails was 50% as well. Now let's try 20 coin tosses and let's compute the probability of 0 to 20 heads for a coin that is not fair.

[Video description begins] *She adds a code in line 50. It reads: num.heads <- 0:20.* [Video description ends]

I invoke the dbinom function on line 51 to generate probabilities for binomially distributed data.

I'm going to toss a coin 20 times and compute the probability of having 0 heads, 1 head, all the way through to 20 heads. But observe here that the coin is not a fair coin.

[Video description begins] *She adds a code in line 51. It reads: binom.dis.data <- data.frame(x = dbinom(num.heads, size = 20, prob = 0.8)).* [Video description ends]

The probability of this coin showing up heads is 0.8 or 80%, that is the third input argument to the dbinom function invocation. Once again, I'm going to plot this probability density curve using ggplot.

[Video description begins] *She adds a code in line 53. It reads: ggplot(binom.dis.data, aes(num.heads, y=x)) + geom_line().* [Video description ends]

And let's see what the binomial distribution for an unfair coin looks like. You can see that the distribution is skewed towards the right.

[Video description begins] *Another plot graph appears in the Plots pane. The y-axis shows the x and the x-axis shows the num.heads.* [Video description ends]

You can see the highest point of this density curve with a probability of about 0.23 corresponds to getting 16 heads, that is 80% of the 20 coin tosses. This should intuitively make sense to you because the probability of getting heads in 1 flip is 80%. Let's look at one last example here where we flip an unfair coin.

[Video description begins] *She adds a code in line 57. It reads: num.heads <- 0:20.* [Video description ends]

Once again, we'll have 20 coin flips and we'll compute the probability of the coin showing up heads 0 times up to 20 times.

I invoke the dbinom function size is equal to 20, probability is 0.3.

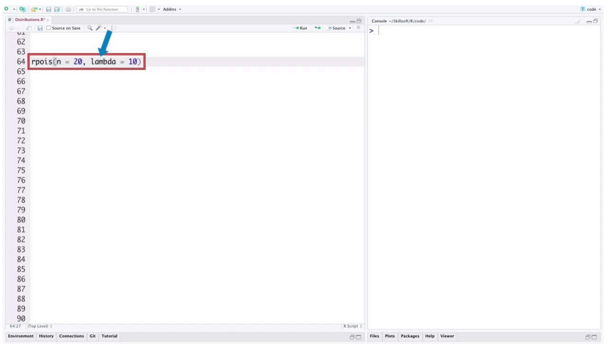
[Video description begins] *She adds a code in line 51. It reads: binom.dis.data <- data.frame(x = dbinom(num.heads, size = 20, prob = 0.3)).* [Video description ends]

The probability of this coin showing up heads is 30%. This is an unfair coin. I plot this probability density curve using ggplot once again. And a quick look should show you that this is a binomial distribution that

[Video description begins] *She adds a code in line 60. It reads: ggplot(binom.dis.data, aes(num.heads, y=x)) + geom_line().* [Video description ends]

skews towards the left. You can see that the highest point of this density curve corresponds to 6 heads showing up across 20 coin flips. Which makes sense because the probability of getting 6 heads out of 20 is 30% which is exactly the probability that we had specified of 1 coin flip showing up as heads.

8. Video: Sampling and Analyzing Poisson Distribution Data (it_dasamrdj_01_enus_08)



```
64 rpois(n = 20, lambda = 10)
```

In this video, you'll learn more about the Poisson distribution. The Poisson distribution is also a discrete probability distribution. You'll learn the Poisson distribution is commonly used to model the number of expected events for a process, as long as you know the average rate at which events occur during a unit of time. Onscreen, you'll see to generate data points from a Poisson distribution using a specific example.

- *sample and analyze data which follows uniform distribution*

[Video description begins] *Topic title: Sampling and Analyzing Poisson Distribution Data. Presented by: Janani Ravi.* [Video description ends]

In this video, we'll understand and explore the Poisson distribution. The Poisson distribution is also a discrete probability distribution.

[Video description begins] *The RStudio IDE appears on the screen. In the code editor, a tab named Distributions.R is open. On the right, a Console pane is open. Its path is: ~/Skillsoft/R/code/.* [Video description ends]

The Poisson distribution is commonly used to model the number of expected events for a process, given we know the average rate at which events occur during a unit of time. For example, let's say we are modeling the number of visitors to a website. We know that we have roughly ten visitors to a website in a single minute, say. The Poisson distribution can be used to estimate how many customers we will have visit our website every minute over the next, say, 60 minutes. So given that we know that the average rate of visits is ten customers a minute, how will the next 60 minutes play out? Let's see how we can generate data points from a Poisson distribution using the same example that we just discussed,

[Video description begins] *She adds a code in line 64. It reads: rpois(n = 20, lambda = 10).* [Video description ends]

the number of customers to a website per minute.

Now, the Poisson distribution requires a parameter lambda. Lambda here gives us the rate of occurrence, how often is an event expected to occur within a particular time window. The time window that we are considering here is a minute. So how many customers do we expect per minute, we have specified 10. Lambda here is the second input argument to the rpois function that allows us to generate data from a Poisson distribution. The first input argument, n=20, allows us to simulate the number of visitors for the next 20 minutes, given the average rate is 10 visitors per second. And when you run this code you can see the results of the simulation on the right console window. Based on our Poisson distribution, roughly 16 visitors are expected the first minute, then 9, then 13, and so on. The numbers that you get might be different, but they're drawn from a Poisson distribution with lambda = 10.

It's pretty clear that using the Poisson distribution will allow us to estimate how many servers we need to have running with our website in order to handle the customer load. Let's consider another example.

[Video description begins] *She adds a code in line 65. It reads: rpois(n = 7, lambda = 50).* [Video description ends]

Let's say we know that the number of customer complaints per day for an e-commerce site is 50. How many complaints can we expect to receive over the next seven days? I'm going to use the rpois function to generate estimates for the next seven days, lambda = 50, meaning I expect on average 50 complaints per unit time. The unit time here is a single day or 24 hours. Or the next seven days, n = 7, let's look at how many customers we expect. You can see that the customers range from 42 to about 57. Being able to estimate the number of customer complaints over a week will allow us to predict how much staff we need to monitor our phones.

The rpois function allows us to generate data points from a Poisson distribution with certain parameters. If you want to compute the actual probability values of events from a Poisson distribution, you should use the dpois function.

[Video description begins] *She adds a code in line 67. It reads: dpois(7, lambda = 10).* [Video description ends]

Here, let's assume that customer complaints come in at about 10 per hour. So our unit time is one hour, and lambda, the average rate of occurrence, is 10. Now what I want to compute is, what is the probability of having exactly seven customers call us within the span of an hour? So the first input argument I've specified to the dpois function is 7. And the result shows me that having exactly 7 customer complaints in an hour is 0.09 or 9%. Given that the average rate of customer calls is at about 10 per hour, what is the probability that we get exactly 10 customer calls in any hour?

[Video description begins] *She adds a code in line 69. It reads: dpois(10, lambda = 10).* [Video description ends]

So lambda = 10, as you can see on line 69, and the first input argument n is also equal to 10.

I've invoked the dpois function, and this gives me a probability score of 12.5%, for getting exactly 10 customer calls. For the same average rate of 10 customer complaints per hour, what is that the probability that we'll get exactly two customer complaints in any hour? Invoke the dpois function passing 2 and lambda = 10,

[Video description begins] *She adds a code in line 71. It reads: dpois(2, lambda = 10).* [Video description ends]

and this gives us a probability of 0.002 or 0.2%. If you want to compute cumulative probability scores for the Poisson distribution, you can use the ppois function. Given that we know that 20 customer complaints come in for any hour, what is the probability of getting seven or fewer complaints in any hour? Now this will be the sum of probabilities of getting exactly zero complaints, plus the probability of getting exactly one complaint plus the probability of getting exactly two complaints, all the way through to seven. Let's invoke the ppois function and see what the probability of getting 7 or fewer complaints for an average of 20 complaints an hour looks like.

The probability here is 0.0007.

[Video description begins] *She adds a code in line 73. It reads: ppois(7, lambda = 20).* [Video description ends]

Now, let's assume that you know that the average number of customer complaints you get on your hotline is 25 an hour, so based on this average, you've staffed your hotline with 25 people in order to answer the average number of complaints that come in. But what is the chance that a customer who comes in will have to wait?

[Video description begins] *She adds a code in line 75. It reads: 1 - ppois(25, lambda = 25).* [Video description ends]

Well, that is the probability that you get more than 25 complaints in any hour. If you want to compute the probability that a customer will have to wait when he calls your hotline, you need to compute the probability that you get more than 25 calls in an hour. Now, the probability that you get 25 or fewer calls in an hour is given by ppois, n = 25, that's what we've passed in, our lambda or average rate is 25.

Now, 1 minus this probability score will give us the probability that we get more than 25 complaints in any hour. And this works out to 0.447 or 44.7%. So the probability that a customer who calls in has to wait is 44.7% if you only have 25 people manning your phones. So for an average rate of 25 calls an hour, you probably want more people manning phones to bring down the probability that a customer has to wait. Let's now use ggplot to visualize the Poisson distribution curve. Let's have our num.complaints range from 0 to 40. So for a lambda of 10, that is an average rate of complaints equal to 10,

[Video description begins] *She adds a code in line 79. It reads: num.complaints <- 0:40.* [Video description ends]

let's see what is the probability for zero complaints, one complaint, two complaints, all the way up to 40 complaints.

[Video description begins] *She adds a code in line 80. It reads: poisson.dis.data <- data.frame(val = dpois(num.complaints, lambda = 10)).* [Video description ends]

We'll invoke the dpois function to get the probability scores for our Poisson distributed data. And we'll store this in a dataframe.

Now that I have the probabilities for the different values in a Poisson distribution, I can plot this using ggplot. On the x-axis I have the sequence of the number of complaints, on the y-axis I have the probability scores. And I plot this as a line plot using geom_line.

[Video description begins] *She adds a code in line 82. It reads: ggplot(posson.dis.data, aes(x=num.complaints, y=val)) + geom_line().* [Video description ends]

And here is our curve representing a Poisson distribution on full screen.

[Video description begins] *A plot graph appears in the Plots pane. The y-axis shows the val and the x-axis shows the num.complaints.* [Video description ends]

You can see that the highest probability score corresponds to around 10 complaints, that is the lambda that we had specified, the average rate at which complaints come in, in an hour. You can visualize multiple Poisson distributions on the same plot for different values of lambda.

[Video description begins] *She adds a code in line 84. It reads: num.complaints <- 0:40.* [Video description ends]

So let's have our complaint sequence be from 0 to 40. I'll use multiple dpois functions to generate the probability density functions for Poisson distributions for different values of lambda.

[Video description begins] *She adds a set of code in lines 85-89. Code line 85 is: poisson.dis.data <- data.frame(val1 = dpois(num.complaints, lambda = 5),. Code line 86 is: val2 = dpois(num.complaints, lambda = 8),. Code line 87 is: val3 = dpois(num.complaints, lambda = 12),. Code line 88 is: val4 = dpois(num.complaints, lambda = 15),. Code line 89 is: val5 = dpois(num.complaints, lambda = 10)).* [Video description ends]

So within a data frame, we have val1 corresponding to a Poisson distribution for lambda = 5, val2 for lambda 8, all the way up to val5 for lambda = 10. Go ahead and run this code. This will give us the probability scores for these different distributions within a single dataframe. We can then use ggplot to plot these probability curves in different colors.

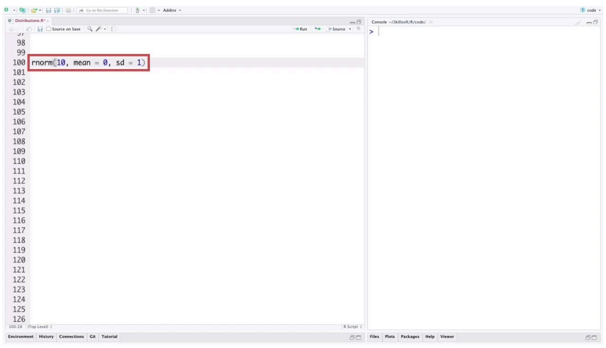
[Video description begins] *She adds a set of code in lines 91-96. Code line 91 is: ggplot(poisson.dis.data, aes(x=num.complaints)) +. Code line 92 is: geom_line(aes(y = val1, color="red")) +. Code line 93 is: geom_line(aes(y = val2, color="blue")) +. Code line 94 is: geom_line(aes(y = val3, color="green")) +. Code line 95 is: geom_line(aes(y = val4, color="yellow")) +. Code line 96 is: geom_line(aes(y = val5, color="orange")).* [Video description ends]

While viewing this in full screen, you can see how the shape of the Poisson distribution changes based on the value of lambda.

[Video description begins] *A plot graph appears in the Plots pane. It displays five different values using five lines of different colors. The y-axis shows the val and the x-axis shows the num.complaints.* [Video description ends]

For a lower rate of lambda of 5, we have a long and narrow density curve. For higher values of lambda, we have shorter and broader density curves.

9. Video: Examining Normal and Exponential Distributions (it_dasamrdj_01_enus_09)



```
rnorm(10, mean = 0, sd = 1)
```

In this video, you'll learn more about the normal distribution, a continuous probability distribution. The normal distribution is a continuous probability distribution that's symmetric around the mean or the average of your data. The probability density curve of a normal distribution is known as a bell curve because it's shaped like a bell. In a normal distribution, data values that are close to the mean occur more often than values far away from the mean.

- *examine and interpret normal distributions and exponential distributions*

[Video description begins] *Topic title: Examining Normal and Exponential Distributions. Presented by: Janani Ravi.* [Video description ends]

After having explored a number of different kinds of discrete probability distributions, let's now explore a continuous probability distribution and that is the normal distribution.

[Video description begins] *The RStudio IDE appears on the screen. In the code editor, a tab named Distributions.R is open. On the right, a Console pane is open. Its path is: ~/Skillsoft/R/code/.* [Video description ends]

The normal distribution is a continuous probability distribution that is symmetric around the mean or the average of your data. The probability density curve of a normal distribution is often referred to as a bell curve, because it's shaped like a bell. In a normal distribution, data values which are very close to the mean occur far more often than values which are far away from the mean. The normal distribution is extremely important in the real world. Because it fits many natural phenomena, heights of individuals, weights of individuals, blood pressure, IQ scores, all of these follow the normal distribution. Other terms for the normal distribution are Gaussian distribution and the bell curve.

A normal distribution is defined using two parameters, the mean or the average value of the data points of the distribution, that is a measure of central tendency, and the standard deviation of the distribution. The standard deviation, as you know, is a measure of variance.

[Video description begins] *She adds a code in line 100. It reads: rnorm(10, mean = 0, sd = 1).* [Video description ends]

Now let's sample 10 data points from a normal distribution which has a mean = 0 and standard deviation = 1. Such a normal distribution that these parameters mean = 0 and standard deviation = 1 is referred to as a Standard Normal Distribution. Here are 10 data points sampled from a Standard Normal Distribution. You can see that we have data on either side of 0 so we have positive points, as well as negative points. Let's now use ggplot to visualize the probability density curve of our standard normal distribution.

[Video description begins] *She adds a set of code in lines 102-104. Code line 102 is: ggplot(data = data.frame(x = c(-5, 5)), aes(x)) +. Code line 103 is: stat_function(fun = dnorm, n = 101, args = list(mean = 0, sd = 1)) +. Code line 104 is: scale_y_continuous(breaks = NULL).* [Video description ends]

Now I've invoked ggplot, but there are a number of other functions that I've used as well.

The first input argument is a data frame containing the x-axis values. I want my x-axis to range from -5 to 5. Next on line 103, I've invoked the stat_function. The stat_function is a geometric function available in the ggplot tool library that allows you to plot a function as a continuous curve. The normal distribution is a continuous probability distribution, which is why I've used stat_function to plot this curve. The input arguments to the stat_function is the function that we want to plot, that is dnorm. That will give me the probability scores for normally distributed data. The second input argument n = 101, gives us the number of data points to interpolate along. I want to generate 101 data points. The args input argument are the arguments that I want to pass into the dnorm function mean = 0, standard deviation = 1.

This is the standard normal distribution. The scale by continuous function will plot my y-axis as a continuous curve. As you can see here on screen, let's view our normally distributed data on full screen. The curve here is in the shape of a bell, you can see that the highest probability corresponds to the value 0. Take a look at 0 on the x-axis, you can see that it maps to the highest point on this curve. As we move away from the value 0 on either side of this bell curve, you can see that the probabilities fall. So the probabilities taper off the further away we go from the mean. You can have a normal distribution centered at any mean and with any standard deviation. On line 106, here you can see I've used ggplot to visualize

[Video description begins] *She adds a set of code in lines 106-108. Code line 106 is: ggplot(data = data.frame(x = c(-10, 34)), aes(x)) +. Code line 107 is: stat_function(fun = dnorm, n = 101, args = list(mean = 12, sd = 5)) +. Code line 108 is: scale_y_continuous(breaks = NULL).* [Video description ends]

normally distributed data with mean = 12 and standard deviation = 5.

This specification is on line 107. Going to plot the x-axis from -10 to 34, this is on line 106. And of course, I use stat_function to plot this as a continuous curve. Let's view this in full screen and you can see that the highest point on the curve corresponds to an x value of 12. That is the mean of our data. We've already seen that the mean parameter that we specify for our normally distributed data specifies the center or the line along which our data is distributed.

[Video description begins] *She adds a set of code in lines 110-114. Code line 110 is: ggplot(data = data.frame(x = c(-5, 5)), aes(x)) +. Code line 111 is: stat_function(fun = dnorm, n = 101, args = list(mean = 0, sd = 1)) +. Code line 112 is: stat_function(fun = dnorm, n = 101, args = list(mean = 0, sd = 2)) +. Code line 113 is: stat_function(fun = dnorm, n = 101, args = list(mean = 0, sd = 3)) +. Code line 114 is: scale_y_continuous(breaks = NULL).* [Video description ends]

The standard deviation similarly gives you an idea of how your data is spread out. Here I'm going to use ggplot to plot three different normal distributions on the same plot. You can see the specifications of each normal distribution on lines 111, 112 and 113.

I've interpolated across 101 data points for each distribution. Each distribution has a mean = 0. As you can see from the input arguments, args, the standard deviation of each distribution is a little different. It's equal to 1 for the first distribution, 2 for the second distribution, and 3 for the third distribution. When you visualize this plot, you will get three different curves. Let's move to full screen here. The tallest and narrowest curve is our normally distributed data which has a standard deviation of 1. A lower standard deviation implies a taller, narrower curve. The second curve is for standard deviation 2. Larger the standard deviation, shorter and flatter the curve. And the third bottom-most curve is for standard deviation equal to 3. That is the widest or broadest of all curves.

From normal distributions, let's move on to another continuous probability distribution function and that is the exponential distribution. In probability theory and statistics, the exponential distribution is the probability distribution of the time between events in a Poisson point process. A Poisson point process is a process in which events occur continuously and independently at a constant average rate. So the parameter you specify for an exponential distribution is the rate of occurrence. I'll now generate exponentially distributed data and let's visualize this distribution using ggplot. I'll set up a sequence x, which contains data from 0 to 20.

[Video description begins] *She adds a code in line 118. It reads: x <- seq(0, 20, length.out = 1000).* [Video description ends]

I'll generate a total of 1000 points. I'll then invoke the dexp function to generate the exponential values corresponding to this data.

[Video description begins] *She adds a code in line 119. It reads: exp.dis.data <- data.frame(x = x, px = dexp(x, rate=0.65)).* [Video description ends]

The exponential distribution is closely related to the Poisson distribution that we studied earlier. The exponential distribution is used to predict the amount of waiting time until the next event in a Poisson process. So how much time do you have to wait till you get the next customer complaint call or the next visitor to your website? If you remember the Poisson distribution, we specified the average rate per unit time. 10 customer complaints in an hour, 50 visitors to your website in a day, that was the rate. But when we talk of the exponential distribution, we tend to speak in terms of time elapsed, how much time elapsed between events? So you can either say that you get three customers per hour or one customer every 20 minutes or one-third of an hour.

The rate that I have specified for the dexp function of 0.65 means it takes the reciprocal of this that is equal to 1.54 units of time till the next event occurs. So the average time elapsed between events in a Poisson process is equal to 1.54 units of time. That is what this rate signifies. Now that we get our exponentially distributed data, let's use ggplot to plot our probability density curve.

[Video description begins] *She adds a code in line 121. It reads: ggplot(exp.dis.data, aes(x=x, y=px)) + geom_line(). A plot graph appears in the Plots pane.* [Video description ends]

And here you can see the shape of the curve for our exponential distribution. You can see that it starts off with a high probability and then tapers off for the higher values of x. Let's now visualize an exponential distribution for a different rate. Once again, I'll generate a 1000 data points in the range 0 to 20. I'll then generate exponentially distributed data with a rate = 0.35.

[Video description begins] *She adds a code in line 123. It reads: x <- seq(0, 20, length.out = 1000).* [Video description ends]

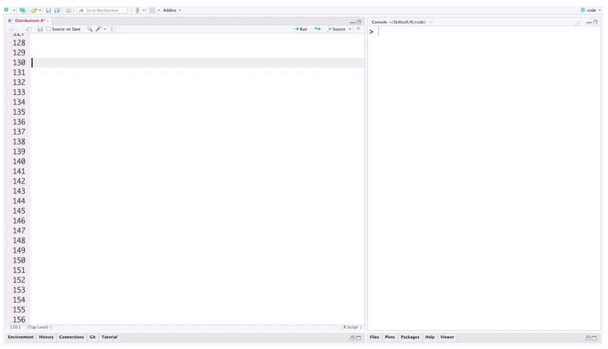
[Video description begins] *She adds a code in line 124. It reads: exp.dis.data <- data.frame(x = x, px = dexp(x, rate=0.35)).* [Video description ends]

This rate implies that the average time elapsed between events is 1 divided by 0.35, that is roughly equal to 2.86, so 2.86 units of time between events. Once we've generated this exponentially distributed data, let's now use ggplot to plot this curve.

[Video description begins] *She adds a code in line 126. It reads: ggplot(exp.dis.data, aes(x=x, y=px)) + geom_line(). A plot graph appears in the Plots pane.* [Video description ends]

And here is the curve for our exponential distribution. Because the rate is lower, notice that the curve tapers off much more gradually.

10. Video: Interpreting QQ Plots Using R (it_dasamrdj_01_enus_10)



In this video, you’ll learn more about QQ plots. QQ Plots are used to assess whether the data you're working with plausibly came from some theoretical distribution. In statistical analysis, you might assume your data is drawn from a population that is normally distributed. This is a requirement for your statistical test. An easy way to test whether this data was from a normal distribution is to use QQ plots.

- *interpret QQ plots for normally and non-normally distributed data*

[Video description begins] *Topic title: Interpreting QQ Plots Using R. Presented by: Janani Ravi.* [Video description ends]

At this point in time, we've explored a number of different kinds of probability distributions, both for discrete data as well as continuous data.

[Video description begins] *The RStudio IDE appears on the screen. In the code editor, a tab named Distributions.R is open. On the right, a Console pane is open. Its path is: ~/Skillsoft/R/code/.* [Video description ends]

Now no discussion of probability distributions is complete without covering QQ plots. QQ Plots are what you'll use to assess whether the data that you're working with plausibly came from some theoretical distribution. Now, what does this mean? Now in statistical analysis, you might assume that your data is drawn from a population that is normally distributed. And this is a requirement for your statistical test. How do you test whether this data was indeed from a normal distribution? An easy way to do this is to use QQ plots. The QQ plot is a graphical tool which will help us assess whether a sample is drawn from a normal distribution. It's just a visual check, it's not airtight proof. So using the QQ plot is somewhat subjective.

But the QQ plot allows us to see at just a glance if our assumption is plausible. And if not, the QQ plot will allow us to see how the assumption has been violated, and what data points contribute to the violation. The qqnorm function in R allows us to check whether the data points that we are working with comes from a population that is normally distributed.

[Video description begins] *She adds a code in line 130. It reads: qqnorm(rnorm(1000, mean = 0, sd = 1).* [Video description ends]

qqnorm accepts as an input argument the data. Now the data that I'm going to pass into this first invocation of qqnorm is data that is drawn from a normal distribution. I'm going to use rnorm to generate these data points, going to generate a 1,000 points. And this is drawn from a normal distribution with mean = 0 and standard deviation = 1.

This is the standard normal distribution.

[Video description begins] *A Normal Q-Q Plot graph appears in the Plots pane. The y-axis shows the Sample Quantiles and the x-axis shows the Theoretical Quantiles.* [Video description ends]

And here you can see that the QQ plot generated using qqnorm is a scatterplot created by plotting two sets of quantiles against one another. Quantiles are basically the percentiles in your data, the points in your data below with a certain proportion of your data fall. On the x-axis, the QQ plot has plotted the theoretical quantiles for normally distributed data. On the y-axis, we have the sample quantiles. That is the quantiles based on the actual data that we passed in. Now, the fact that all of the data points lie along a 45-degree line tells us that our data, the sample that we passed in, is normally distributed data. If the data that we had passed into qqnorm was not normally distributed, the points would not be along this 45-degree line. I'm going to go back to my code and clear this QQ plot that I've plotted by invoking dev.off. This will clear my device.

[Video description begins] *She adds a code in line 131. It reads: dev.off().* [Video description ends]

I'm going to invoke the qqnorm function and pass in data once again, which is drawn from a normally distributed population.

[Video description begins] *She adds a code in line 133. It reads: qqnorm(rnorm(1000, mean = 10, sd = 1.8).* [Video description ends]

However, this is a normal distribution with mean = 10 and standard deviation = 1.8. This is not the standard normal distribution, but it is a normal distribution and you can see the QQ plot of this data is also aligned at a 45-degree angle. So all of the points in our scatterplot lie along this 45-degree angle,

[Video description begins] *Another Normal Q-Q Plot graph appears in the Plots pane. The y-axis shows the Sample Quantiles and the x-axis shows the Theoretical Quantiles.* [Video description ends]

indicating that our sample data has been drawn from a normally distributed population. Let's go back to our code here and I'm going to use dev.off to clear my plotting screen.

[Video description begins] *She adds a code in line 134. It reads: dev.off().* [Video description ends]

Before I move on to my next plot, I'm going to plot a QQ plot of normally distributed data mean = 0, standard deviation = 1. This is the standard normal distribution.

And you can see from the plot on the lower right of the screen that

[Video description begins] *She adds a code in line 136. It reads: qqnorm(rnorm(1000, mean = 0, sd = 1).* [Video description ends]

the scatterplot is at a 45-degree angle. If you want to compare your QQ plot with a line that corresponds to

[Video description begins] *Another Normal Q-Q Plot graph appears in the Plots pane. The y-axis shows the Sample Quantiles and the x-axis shows the Theoretical Quantiles.* [Video description ends]

a theoretical distribution, you can use the qqline function. The qqline function will draw a line through the quantiles representing

[Video description begins] *She adds a code in line 137. It reads: qqline(rnorm(1000, mean = 0, sd = 1).* [Video description ends]

a theoretical distribution. Here the distribution that we have specified is a standard normal distribution mean = 0, sd = 1, you can see this on line 137. Now, this qqline function will overlay the line passing through the theoretical quantiles overlaid on the QQ plot that we previously generated. Let's view this in the full screen.

[Video description begins] *Another Normal Q-Q Plot graph appears in the Plots pane. The y-axis shows the Sample Quantiles and the x-axis shows the Theoretical Quantiles. It displays a line inclined at 45 degree.* [Video description ends]

You can now compare the scatterplot of our data with the theoretical line that represents a normal distribution. You can see our data points lie along this line, indicating our sample has been drawn from a normal distribution.

We can use this combination of the qqnorm function and the qqline function to interpret something from our data. For example, if we have two data sets which come from populations whose distributions differ only by a shift in location, that is, a shift in the mean value of that data, you can interpret this from the QQ plot and qqline.

[Video description begins] *She adds a code in line 138. It reads: dev.off().* [Video description ends]

I'll first use qqnorm to plot R data drawn from a normal distribution with mean = 1 and standard deviation = 1.

[Video description begins] *She adds a code in line 140. It reads: qqnorm(rnorm(1000, mean = 1, sd = 1).* [Video description ends]

Observe that mean here is equal to 1. I want to now compare my data drawn from a normal distribution with mean = 1, standard deviation = 1 with data drawn from a standard normal distribution. So I use qqline to plot a line that will pass through the theoretical quantiles for the standard normal distribution mean = 0 and standard deviation = 1.

The data that I've generated on line 140 has mean = 1.

[Video description begins] *She adds a code in line 141. It reads: qqline(rnorm(1000, mean = 0, sd = 1).* [Video description ends]

The data that I've generated on line 141 that I want to compare the original data with has mean = 0. So the first data set has been shifted in relation to the second data set. And you can visualize this in our QQ plot.

[Video description begins] *Another Normal Q-Q Plot graph appears in the Plots pane. The y-axis shows the Sample Quantiles and the x-axis shows the Theoretical Quantiles.* [Video description ends]

You can see that the data points of our scatterplot are shifted, displaced, above the 45-degree reference line. The 45-degree reference line represents the standard normal distribution. Our distribution has mean = 1 which is why the QQ plot of our distribution has been displaced such that it lies above the reference line. What if we want to compare two normal distributions which have the same mean but different standard deviations? That's exactly what we'll do next.

[Video description begins] *She adds a code in line 142. It reads: dev.off().* [Video description ends]

I'm going to generate a QQ plot of normally distributed data with mean = 0 and standard deviation = 2.

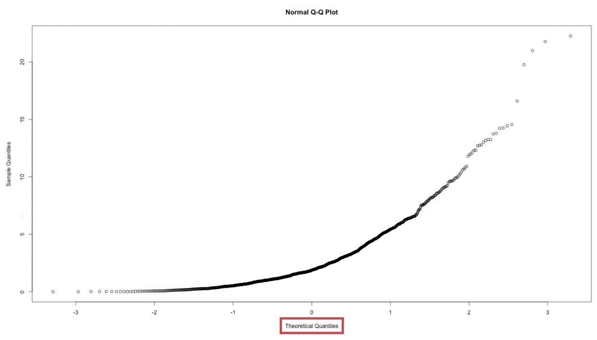
[Video description begins] *She adds a code in line 144. It reads: qqnorm(rnorm(1000, mean = 0, sd = 2).* [Video description ends]

When I run this code, a QQ plot will be generated. I want to compare this data with another normal distribution which has the same mean, mean = 0, but a different standard deviation. The standard deviation for the data that I represent using a line is 1.

[Video description begins] *She adds a code in line 145. It reads: qqline(rnorm(1000, mean = 0, sd = 1).* [Video description ends]

So I'm comparing the data set that I've generated on line 144 with the reference data set that I've generated on line 145. They differ only in the standard deviations. Let's overlay this plot and let's zoom in so that we see this on full screen. And you can see how our data represented by the scatterplot is kind of displaced in angle compared to the reference line. The only difference in the two distributions is that they have different standard deviations, and we can interpret this from the QQ plot.

11. Video: Using QQ Plots in R to Compare Datasets (it_dasamrdj_01_enus_11)



In this video, you’ll work with QQ norm and non-normally distributed data. You'll use QQ norm to get a visual check on whether your data points are normally distributed. First, you’ll use the rexp or rexp function to generate data points drawn from an exponential distribution. You’ll generate a total of 1000 data points. The rate for this exponential distribution is 0.35. Because it’s artificially generated data, you know the data is exponentially distributed.

- *use QQ plots to compare samples from different distributions*

[Video description begins] *Topic title: Using QQ Plots in R to Compare Datasets. Presented by: Janani Ravi.* [Video description ends]

We've already seen that QQ plots can be used to see whether the data points that we have in our sample come from a certain theoretical distribution. Now so far, we've only worked with data that has been normally distributed. Now let's work with QQ norm and non-normally distributed data. We'll use QQ norm to get a visual check on whether our data points are normally distributed.

[Video description begins] *She adds a code in line 150. It reads: qqnorm(rexp(1000, rate = 0.35)).* [Video description ends]

The first thing I'm going to do here is use the rexp or rexp function to generate data points that are drawn from an exponential distribution. I'm going to generate a total of 1000 data points. The rate for this exponential distribution is 0.35. Because this is artificially generated data, I know my data is exponentially distributed. I'm going to plot this data using qqnorm. And we'll take a look at the scatterplot that we'll get and we'll see how we can interpret this scatter plot.

I'm going to zoom in and bring this scatterplot to the full screen.

[Video description begins] *A Normal Q-Q Plot graph appears in the Plots pane. The y-axis shows the Sample Quantiles and the x-axis shows the Theoretical Quantiles.* [Video description ends]

Now along the x-axis, we have the theoretical quantiles for normally distributed data. Along the y-axis, we have the sample quantiles for the data set that we've passed in. We know that data is exponentially distributed. When you view this normal QQ plot, what you're comparing is your current dataset, which is exponentially distributed data, against the quantiles of a normal distribution. So our generated data is exponentially distributed and it has been plotted against quantiles of a normal distribution. An interpretation that we can draw from this QQ plot here, because of the curve that you see in the data points, is that the dataset that we're comparing against the normal is skewed in some way.

And we know that exponential data is skewed if you compare it with normally distributed data. It's pretty clear here from this scatter plot that the data points corresponding to our exponentially distributed data do not lie along the mean 45-degree diagonal. Now if you remember how the QQ plot of data that is a normally distributed looks, you might remember that our data points lie along a 45-degree diagonal as you can see from the inset picture here. If your data points were normally distributed, the QQ plot would show your points along this 45-degree diagonal as in the inset picture. But because our current data is exponentially distributed, you can see the data points are not along this 45-degree line. This is how you know that your data is not normally distributed. It's not along the 45 diagonal. I'm going to clear my output here using dev.off.

[Video description begins] *She adds a code in line 151. It reads: dev.off().* [Video description ends]

Back to our code, we'll compare yet another distribution with the normal distribution.

I use the qqnorm function once again. qqnorm is what I'll use to compare my data against the normal distribution. My data points are generated using the chi-square distribution. rchi square is the name of the function that I invoke. I'll generate a 1000 data points from the chi-square distribution

[Video description begins] *She adds a code in line 153. It reads: qqnorm(rchisq(1000, df = 7)).* [Video description ends]

with 7 degrees of freedom, df = 7, that is a parameter that the chi-square function requires.

[Video description begins] *Another Normal Q-Q Plot graph appears in the Plots pane. The y-axis shows the Sample Quantiles and the x-axis shows the Theoretical Quantiles.* [Video description ends]

Now, the chi-square function also generates a skewed dataset. And if you look at our QQ plot, you can see that the data points lie along a curve. They are not along the 45-degree normal as would be the case if the dataset is normally distributed. The theoretical quantiles on the x-axis are for normally distributed data. The sample quantiles on the y-axis correspond to our generated data which follows the chi-square distribution.

Just because they artificially generated it, we know the distribution of our data. So far, the R functions that we've explored for QQ plots are the qqnorm and the qqline functions.

[Video description begins] *She adds a code in line 154. It reads: dev.off().* [Video description ends]

These compare our datasets against the normal distribution. But what if you want to compare two datasets together to see whether they are from the same distribution? So you don't want to compare against the normal. You want to be able to provide two sets of data as inputs and see whether those points come from the same distribution. That's exactly when you would choose to use the qqplot function in R.

[Video description begins] *She adds a code in line 156. It reads: qqplot(rnorm(1000, mean = 0, sd = 1, rnorm(1000, mean = 0, sd = 1)).* [Video description ends]

The qqplot function invocation that you see here on line 156 accepts two sets of data points. Now both of the data points that I've specified are generated from normally distributed data.

The first rnorm invocation has 1000 data points from the standard normal distribution, mean = 0, standard deviation = 1. The second rnorm invocation also has 1000 data points from a standard normal distribution, mean = 0, standard deviation = 1. The qqplot will allow us to check whether both R datasets are drawn from the same distribution. In this particular example, they are, both of them are drawn from the standard normal distribution with mean = 0 and standard deviation = 1. Let's take a look at the qqplot here.

[Video description begins] *Another Normal Q-Q Plot graph appears in the Plots pane. The y-axis and the x-axis shows rnorm.* [Video description ends]

And you can see that all of the points in our scatterplot lie along the 45-degree diagonal. Here are the quantiles of the data passed in as the first input argument are plotted along the x-axis and the quantiles of the data which we pass in at the second input argument are plotted along the y-axis. Now both of the datasets are drawn from a standard normal distribution, which is why the QQ plot is basically along the 45-degree diagonal. Going back to our code, let's now compare two different distributions. Turn off our plot using dev.off.

[Video description begins] *She adds a code in line 157. It reads: dev.off().* [Video description ends]

Now I'm going to use qqplot to plot data from a chi-square distribution and compare it with the standard normal distribution.

[Video description begins] *She adds a code in line 159. It reads: qqplot(rchisq(500, df = 7), rnorm(1000, mean = 0, sd = 1)).* [Video description ends]

The first input argument to qqplot is a dataset that contains 500 data points from a chi-square distribution. This is a chi-square distribution with 7 degrees of freedom, that is our first input. The second input to qqplot is data which comprise of 1000 points drawn from the standard normal distribution with mean = 0, and standard deviation = 1. So I'm comparing datasets drawn from two different distributions. And we'll use the qqplot function to see whether the points are from the same distribution.

[Video description begins] *Another Normal Q-Q Plot graph appears in the Plots pane. The y-axis shows rnorm and the x-axis shows rchisq.* [Video description ends]

The fact that the qqplot does not have the data points along the 45 degree diagonal, but instead the data points lie along a curve, tells us that the data points are not drawn from the same distribution. We of course already knew this because we artificially generated our data.

But this plot here, this shape of this qqplot makes it very clear that the two datasets that we are working with are not drawn from the same distribution. That comes from the fact that our scatterplot points do not lie along the 45-degrees diagonal. Let's do this once again and compare data from two different distributions. I'll use dev.off to clear my plotting window.

[Video description begins] *She adds a code in line 160. It reads: dev.off().* [Video description ends]

Once again, I'll use qqplot to compare two datasets.

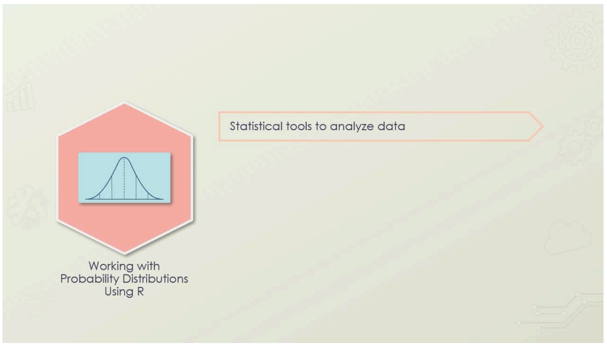
[Video description begins] *She adds a code in line 162. It reads: qqplot(rcauchy(500), rnorm(1000, mean = 0, sd = 1)).* [Video description ends]

The first data set comprises of 500 points drawn from a cauchy distribution using the function rcauchy. The second dataset comprises of 1000 points drawn from a standard normal distribution mean = 0, standard deviation = 1. Now, because this is artificially generated data, we know what the qqplot will show. If you look at the qqplot, the data points in the scatterplot do not lie along the 45-degree diagonal, telling us that the data that we've passed

[Video description begins] *Another Normal Q-Q Plot graph appears in the Plots pane. The y-axis shows rnorm and the x-axis shows rcauchy.* [Video description ends]

into qqplot are not drawn from the same underlying distribution.

12. Video: Course Summary (it_dasamrdj_01_enus_12)


Statistical tools to analyze data
Working with Probability Distributions Using R

In this video, you'll summarize what you've learned in this course. You've explored the statistical tools that analysts use to understand and interpret their data. You covered descriptive statistics used to summarize data and inferential statistics which are used to make deductions from a sample about the population the sample represents. You also learned about probability distributions and how they help you understand data. You explored discrete probability distributions which are used with categorical data.

- *summarize the key concepts covered in this course*

[Video description begins] *Topic title: Course Summary.* [Video description ends]

This brings us to the very end of this course on Working with Probability Distributions Using R. We started this course off by exploring the statistical tools that analysts use to understand and interpret their data. In this context, we covered descriptive statistics used to summarize data and inferential statistics which is used to make deductions from a sample about the population that the sample represents. We then spoke of the characteristics of the population and the sample drawn from the population and the importance of the sample being a representative sample in order to allow us to make inferences about the population. Next, we discussed what exactly probability distributions are and how they help us understand our data. We explored discrete probability distributions which are used with categorical data such as coin tosses. And we also understood continuous probability distributions used with continuous variables.

We worked with and understood the use of discrete probability distributions such as the uniform distribution, the binomial distribution and the poison distribution. We also implemented continuous probability distributions in R, such as the normal distribution and the exponential distribution. Finally, we plotted and interpreted QQ plots which allowed us to compare the distribution of our data with the normal distribution and also allowed us to compare two samples to see whether they've been drawn from the same underlying population distribution. Now that you've completed and understood the concepts that we've covered in this course, you have the foundations you need to understand and interpret your data. You're now ready to move on to understanding and interpreting statistical tests. That's in the course coming up next.

Course File-based Resources

- Course Assets