Sub Code / Sub Name:	– Introduction to Data Science	
Unit: 01	**Lecture No.: 1**	**Date: 23-09-2022**
Topic Name: Introduction		
Name of the Faculty: Mahesh Hundekar		

Data science is a broad concept, which presupposes gathering data from multiple sources and extracting critical information with the help of machine learning (ML), artificial intelligence (AI), predictive analytics, and sentiment analysis.

What people might not know is that the "datafication" of our offline behavior has started as well, mirroring the online data collection revolution (more on this later). Put the two together, and there's a lot to learn about our behavior and, by extension, who we are as a species. It's not just Internet data, though—it's finance, the medical industry, pharmaceuticals, bioinformatics, social welfare, government, education, retail, and the list goes on. There is a growing influence of data in most sectors and most industries. In some cases, the amount of data collected might be enough to be considered "big"; in other cases, it's not.

But it's not only the massiveness that makes all this new data interesting (or poses challenges). It's that the data itself, often in real time, becomes the building blocks of data products. On the Internet, this means Amazon recommendation systems, friend recommendations on Face-book, film and music recommendations, and so on. In finance, this means credit ratings, trading algorithms, and models. In education, this is starting to mean dynamic personalized learning and assessments coming out of places like Knewton and Khan Academy. In government, this means policies based on data. We're witnessing the beginning of a massive, culturally saturated feed- back loop where our behavior changes the product and the product changes our behavior. Technology makes this possible: infrastructure for large-scale data processing, increased memory, and bandwidth, as well as a cultural acceptance of technology in the fabric of our lives. This wasn't true a decade ago.

Datafication:

datafication as a process of "taking all aspects of life and turning them into data." As examples, they mention that "Google's augmented-reality glasses datafy the gaze. Twitter datafies stray thoughts. LinkedIn datafies professional networks."

Datafication is an interesting concept and led us to consider its importance with respect to people's intentions about sharing their own data. We are being datafied, or rather our actions are, and when we

"like" someone or something online, we are intending to be datafied, or at least we should expect to be. But when we merely browse the Web, we are unintentionally, or at least passively, being datafied through cookies that we might or might not be aware of. And when we walk around in a store, or even on the street, we are being datafied in a completely unintentional way, via sensors, cameras, or Google glasses.

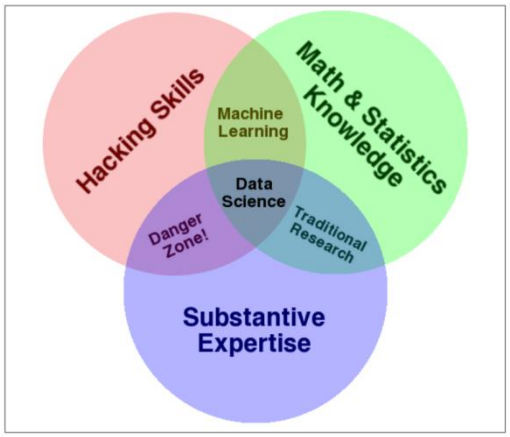

Figure 1-1. Drew Conway's Venn diagram of data science

Statistical Inference:

As we commute to work on subways and in cars, as our blood moves through our bodies, as we're shopping, emailing, procrastinating at work by browsing the Internet and watching the stock market, as we're building things, eating things, talking to our friends and family about things, while factories are producing products, this all at least potentially produces data.

The point here is that the processes in our lives are actually data-generating processes. This overall process of going from the world to the data, and then from the data back to the world, is the field of statistical inference. More precisely, statistical inference is the discipline that concerns itself with the development of procedures, methods, and theorems that allow us to extract meaning and information from data that has been generated by stochastic (random) processes.

Populations and Samples:

In classical statistical literature, a distinction is made between the population and the sample. The word population immediately makes Statistical Thinking in the Age of Big Data us think of the entire world's population of 7 billion people. But put that image out of your head, because in statistical inference

population isn't used to simply describe only people. It could be any set of objects or units, such as tweets or photographs or stars. If we could measure the characteristics or extract characteristics of all those objects, we'd have a complete set of observations, and the convention is to use N to represent the total number of observations in the population.

When we take a sample, we take a subset of the units of size n in order to examine the observations to draw conclusions and make inferences about the population.

Suppose your population was all emails sent last year by employees at a huge corporation, BigCorp. Then a single observation could be a list of things: the sender's name, the list of recipients, date sent, text of email, number of characters in the email, number of sentences in the email, number of verbs in the email, and the length of time until first reply.

Populations and Samples of Big Data

But, wait! In the age of Big Data, where we can record all users' actions all the time, don't we observe everything? Is there really still this notion of population and sample? If we had all the email in the first place, why would we need to take a sample?

There are multiple aspects of this that need to be addressed.

Sampling solves some engineering challenges

In the current popular discussion of Big Data, the focus on en- terprise solutions such as Hadoop to handle engineering and computational challenges caused by too much data overlooks sampling as a legitimate solution. At Google, for example, soft- ware engineers, data scientists, and statisticians sample all the time.

How much data you need at hand really depends on what your goal is: for analysis or inference purposes, you typically don't need to store all the data all the time. On the other hand, for serving purposes you might: in order to render the correct information in a UI for a user, you need to have all the information for that particular user, for example.

Bias

Even if we have access to all of Facebook's or Google's or Twitter's data corpus, any inferences we make from that data should not be extended to draw conclusions about humans beyond those sets of users, or even those users for any particular day.

Sampling

Let's rethink what the population and the sample are in various contexts.

In statistics we often model the relationship between a population and a sample with an underlying mathematical process. So we make sim- plifying assumptions about the underlying truth, the mathematical structure, and shape of the underlying generative process that created the data. We observe only one particular realization of that generative process, which is that sample.

New kinds of data

Gone are the days when data is just a bunch of numbers and categorical variables. A strong data scientist needs to be versatile andcomfortable with dealing a variety of types of data, including:

- Traditional: numerical, categorical, or binary
- Text: emails, tweets, articles
- Records: user-level data, time stamped event data, json formatted log files
- Geo-based location data
- Network
- Sensor data
- Images.

n = 1

At the other end of the spectrum from N=ALL, we have n=1, by which we mean a sample size of 1. In the old days a sample size of 1 would be ridiculous; you would never want to draw inferences about an en- tire population by looking at a single individual. And don't worry, that's still ridiculous. But the concept of n=1 takes on new meaning in the age of Big Data, where for a single person, we actually can record tons of information about them, and in fact we might even sample from all the events or actions they took (for example, phone calls or keystrokes) in order to make inferences about them. This is what user level modeling is about.

What is a model?

Humans try to understand the world around them by representing it in different ways. Architects capture attributes of buildings through blueprints and three-dimensional, scaled-down versions. Molecular biologists capture protein structure with three-dimensional visuali- zations of the connections between amino acids. Statisticians and data scientists capture the uncertainty and randomness of data-generating processes with mathematical functions that express the shape and structure of the data itself. A model is our attempt to understand and represent the nature of reality through a particular lens, be it architectural, biological, or mathematical. A model is an artificial construction where all extraneous detail has been removed or abstracted. Attention must always be paid to these abstracted details after a model has been analyzed to see what might have been overlooked. In the case of proteins, a model of the protein backbone with side chains by itself is removed from the laws of quantum mechanics that govern the behavior of the electrons, which ultimately dictate the structure and actions of proteins. In the case of a statistical model, we may have mistakenly excluded key variables, included irrelevant ones, or assumed a mathematical structure divorced from reality.

But how do you build a model?

You start simply and keep building it up in complexity, making assumptions, and writing your assumptions down. You can use full-blown sentences if it helps—e.g., "I assume that my users naturally cluster into about five groups because when I hear the sales rep talk about them, she has about five different types of people she talks about"—then taking your words and trying to express them as equations and code. Remember, it's always good to start simply. There is a trade-off in modeling between simple and accurate. Simple models may be easier to interpret and understand. Oftentimes the crude, simple model gets you 90% of the way there and only takes a few hours to build and fit, whereas getting a more complex model might take months and only get you to 92%. You'll start building up your arsenal of potential models throughout this book. Some of the building blocks of these models are probability distributions.

Probability distributions

Back in the day, before computers, scientists observed real-world phenomenon, took measurements, and noticed that certain mathematical shapes kept reappearing. The classical example is the height of hu- mans, following a normal distribution—a bell-shaped curve, also called a Gaussian distribution, named after Gauss.

They are to be interpreted as assigning a probability to a subset of possible outcomes, and have corresponding functions. For example, the normal distribution is written as:

$$N(x|\mu,\sigma) \sim \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

The parameter μis the mean and median and controls where the distribution is centered (because this is a symmetric distribution), and the parameter σ controls how spread out the distribution is. This is the general functional form, but for specific real-world phenomenon, these parameters have actual numbers as values, which we can estimate from the data.

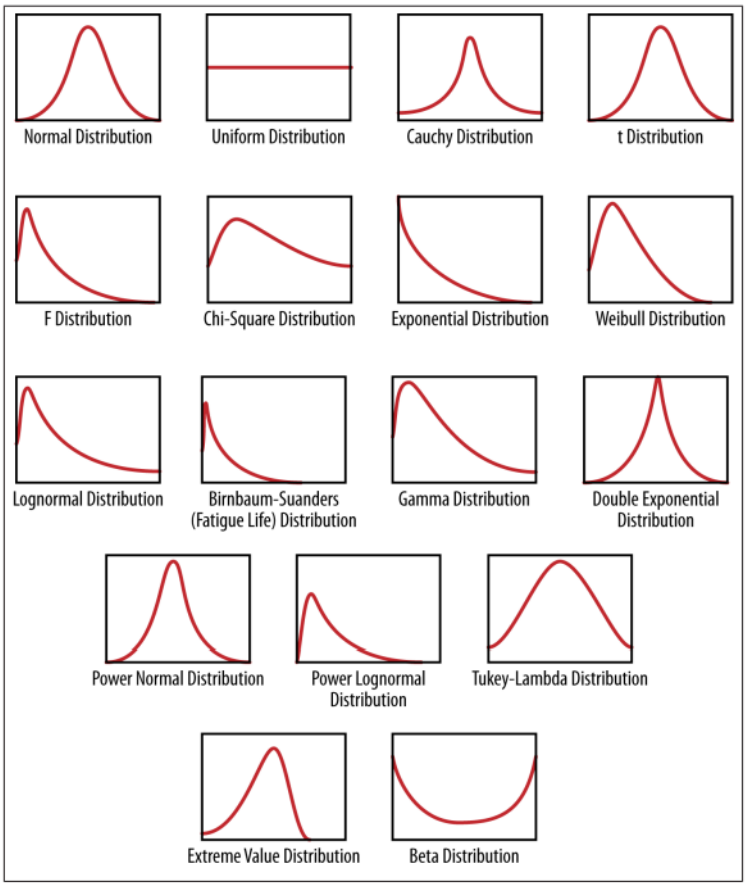

Figure 2-1. A bunch of continuous density functions (aka probability distributions)

Fitting a model

Fitting a model means that you estimate the parameters of the model using the observed data. You are using your data as evidence to help approximate the real-world mathematical process that generated the data. Fitting the model often involves optimization methods and algorithms, such as maximum likelihood estimation, to help get the parameters. In fact, when you estimate the parameters, they are

actually estima- tors, meaning they themselves are functions of the data. Once you fit the model, you actually can write it as y =7.2+4.5x, for example, which means that your best guess is that this equation or functional form expresses the relationship between your two variables, based on your assumption that the data followed a linear pattern. Fitting the model is when you start actually coding: your code will read in the data, and you'll specify the functional form that you wrote down on the piece of paper. Then R or Python will use built-in optimization methods to give you the most likely values of the parameters given the data. As you gain sophistication, or if this is one of your areas of expertise, you'll dig around in the optimization methods yourself. Initially you should have an understanding that optimization is taking place and how it works, but you don't have to code this part yourself—it underlies the R or Python functions.

Overfitting

Throughout the book you will be cautioned repeatedly about overfitting, possibly to the point you will have nightmares about it. Overfitting is the term used to mean that you used a dataset to estimate the parameters of your model, but your model isn't that good at capturing reality beyond your sampled data. You might know this because you have tried to use it to predict labels for another set of data that you didn't use to fit the model, and it doesn't do a good job, as measured by an evaluation metric such as accuracy.

What is R Software?

R is a programming language and free software developed by Ross Ihaka and Robert Gentleman in 1993. R possesses an extensive catalog of statistical and graphical methods. It includes machine learning algorithms, linear regression, time series, statistical inference to name a few. Most of the R libraries are written in R, but for heavy computational tasks, C, C++ and Fortran codes are preferred.

Data analysis with R is done in a series of steps; programming, transforming, discovering, modeling and communicate the results

- **Program**: R is a clear and accessible programming tool
- **Transform**: R is made up of a collection of libraries designed specifically for data science
- **Discover**: Investigate the data, refine your hypothesis and analyze them
- **Model**: R provides a wide array of tools to capture the right model for your data
- **Communicate**: Integrate codes, graphs, and outputs to a report with R Markdown or build Shiny apps to share with the world

What is R used for?

- Statistical inference
- Data analysis
- Machine learning algorithm

Installation

- The R system consists of two major parts:
    - The base system (what you need to install R for the first time).
    - The collection of contributed packages.

- Sources, binaries and documentation for R can be obtained via CRAN (Comprehensive R Archive Network).
- Choose a location in http://cran.r-project.org/mirrors.html.
- For most users (Windows, MacOS X and some Linux distributions) is sufficient to download and install precompiled binaries for base distribution (following the instructions by the installer).
- Run the setup program (.exe in Windows, .app in Mac).
- By default, R is installed into %ProgramFiles%R.

Creating Variables in R

- Variables are containers for storing data values.
- R does not have a command for declaring a variable. A variable is created the moment you first assign a value to it. To assign a value to a variable, use the <- sign. To output (or print) the variable value, just type the variable name:

Data Types

- In programming, data type is an important concept.
- Variables can store data of different types, and different types can do different things.
- In R, variables do not need to be declared with any particular type, and can even change type after they have been set:
-

Basic Data Types

Basic data types in R can be divided into the following types:

- numeric - (10.5, 55, 787)
- integer - (1L, 55L, 100L, where the letter "L" declares this as an integer)
- complex - (9 + 3i, where "i" is the imaginary part)
- character (a.k.a. string) - ("k", "R is exciting", "FALSE", "11.5")
- logical (a.k.a. boolean) - (TRUE or FALSE)

We can use the class() function to check the data type of a variable:

Examples:

```
# numeric
x <- 10.5
class(x)

# integer
x <- 1000L
class(x)

# complex
x <- 9i + 3
class(x)

# character/string
x <- "R is exciting"
class(x)

# logical/boolean
x <- TRUE
class(x)
```