INTRODUCTION TO DATA SCIENCE

UNIT - I

Introduction: Definition of Data Science- Big Data and Data Science hype – and getting past the hype - Datafication - Current landscape of perspectives - Statistical Inference - Populations and samples - Statistical modeling, probability distributions, fitting a model – Over fitting. **Basics of R:** Introduction, R- Environment Setup, Programming with R, Basic Data Types.

1. Introduction: What Is Data Science?

Over the past few years, there's been a lot of hype in the media about "data science" and "Big Data." Today, Data rules the world. This has resulted in a huge demand for Data Scientists.

A Data Scientist helps companies with data-driven decisions, to make their business better. Data science is a field that deals with unstructured, structured data, and semi-structured data. It involves practices like data cleansing, data preparation, data analysis, and much more.

Data science is the combination of: statistics, mathematics, programming, and problem-solving; capturing data in ingenious ways; the ability to look at things differently; and the activity of cleansing, preparing, and aligning data. This umbrella term includes various techniques that are used when extracting insights and information from data.

Big data refers to significant volumes of data that cannot be processed effectively with the traditional applications that are currently used. The processing of big data begins with raw data that isn't aggregated and is most often impossible to store in the memory of a single computer.

Data analytics is the science of examining raw data to reach certain conclusions. Data analytics involves applying an algorithmic or mechanical process to derive insights and running through several data sets to look for meaningful correlations. It is used in several industries, which enables organizations and data analytics companies to make more informed decisions, as well as verify and disprove existing theories or models. The focus of data analytics lies in inference, which is the process of deriving conclusions that are solely based on what the researcher already knows.

Big Data and Data Science Hype

Data science **enables companies not only to understand data from multiple sources but also to enhance decision making**. As a result, data science is widely used in almost every industry, including health care, finance, marketing, banking, city planning, and more.
If you are probably means you have something useful to contribute to making data science into a more legitimate field that has the power to have a positive impact on society.

So, what is eyebrow-raising about Big Data and data science? Let's count the ways:

1. There's a lack of definitions around the most basic terminology. What is "Big Data" anyway? What does

"data science" mean? What is the relationship between Big Data and data science? Is data science the science of Big Data? Is data science only the stuff going on in companies like Google and Facebook and tech companies? Why do many people refer to Big Data as crossing disciplines (astronomy, finance, tech, etc.) and to data science as only taking place in tech? Just how *big* is big? Or is it just a relative term? These terms are so ambiguous, they're well-nigh meaningless.

2. There's a distinct lack of respect for the researchers in academia and industry labs who have been working on this kind of stuff for years, and whose work is based on decades (in some cases, centuries) of work by statisticians, computer scientists, mathematicians, engineers, and scientists of all types. From the way the media describes it, machine learning algorithms were just invented last week and data was never "big" until Google came along. This is simply not the case. Many of the methods and techniques we're using—and the challenges we're facing now—are part of the evolution of everything that's come before. This doesn't mean that there's not new and exciting stuff going on, but we think it's important to show some basic respect for everything that came before.
3. The hype is crazy—people throw around tired phrases straight out of the height of the pre-financial crisis era like "Masters of the Universe" to describe data scientists, and that doesn't bode well. In general, hype masks reality and increases the noise-to-signal ratio. The longer the hype goes on, the more many of us will get turned off by it, and the harder it will be to see what's good underneath it all, if anything.
4. Statisticians already feel that they are studying and working on the "Science of Data." That's their bread and butter. Maybe you, dear reader, are not a statistician and don't care, but imagine that for the statistician, this feels a little bit like how identity theft might feel for you. Although we will make the case that data science is *not* just a rebranding of statistics or machine learning but rather a field unto itself, the media often describes data science in a way that makes it sound like as if it's simply statistics or machine learning in the context of the tech industry.
5. People have said to us, "Anything that has to call itself a science isn't." Although there might be truth in there, that doesn't mean that the term "data science" *itself* represents nothing, but of course what it represents may not be science but more of a craft.

Getting Past the Hype

Data science **enables companies not only to understand data from multiple sources but also to enhance decision making**. As a result, data science is widely used in almost every industry, including health care, finance, marketing, banking, city planning, and more.

Around all the hype, in other words, there is a ring of truth: this *is* something new. But at the same time, it's a fragile, nascent idea at real risk of being rejected prematurely. For one thing, it's being paraded around as a magic bullet, raising unrealistic expectations that will surely be disappointed.

Rachel gave herself the task of understanding the cultural phenom-enon of data science and how others were experiencing it. She started meeting with people at Google, at startups and tech companies, and at universities, mostly from within statistics departments.

From those meetings she started to form a clearer picture of the new thing that's emerging. She ultimately decided to continue the investigation by giving a course at Columbia called "Introduction to Data Science," which Cathy covered on her blog. We figured that by the end of the semester, we, and hopefully the students, would know what all this actually meant. And now, with this book, we hope to do the same for many more people.

Why Now?
We have massive amounts of data about many aspects of our lives, and, simultaneously, an abundance of inexpensive computing power. Shopping, communicating, reading news, listening to music, searching for information, expressing our opinions—all this is being tracked online, as most people know.

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).

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 data"; 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 Facebook, 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 feedback 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.

Considering the impact of this feedback loop, we should start thinking seriously about how it's being conducted, along with the ethical and technical responsibilities for the people responsible for the process.

Datafication
An article on "The Rise of Big Data". have discuss the concept of datafication, and their example is how we quantify friendships with "likes": it's the way everything we do, online or otherwise, ends up recorded for later examination in someone's data storage units. Or maybe multiple storage units, and maybe also for sale.

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.

The Current Landscape

What is data science? Is it new, or is it just statistics or analytics rebranded? Is it real, or is it pure hype? And if it's new and if it's real, what does that mean?

This is an ongoing discussion, but one way to understand what's going on in this industry is to look online and see what current discussions are taking place. This doesn't necessarily tell us what data science is, but it at least tells us what other people think it is, or how they're perceiving it. What is Data Science?" and here's Metamarket CEO Mike Driscoll's answer:

Data science, as it's practiced, is a blend of Red-Bull-fueled hacking and espresso-inspired statistics.

But data science is not merely hacking—because when hackers finish debugging their Bash one-liners and Pig scripts, few of them care about non-Euclidean distance metrics.

And data science is not merely statistics, because when statisticians finish theorizing the perfect model, few could read a tab-delimited file into R if their job depended on it.

Data science is the civil engineering of data. Its acolytes possess a practical knowledge of tools and materials, coupled with a theoretical understanding of what's possible. Driscoll then refers to Drew Conway's Venn diagram of data science from 2010, shown in Figure 1-1.

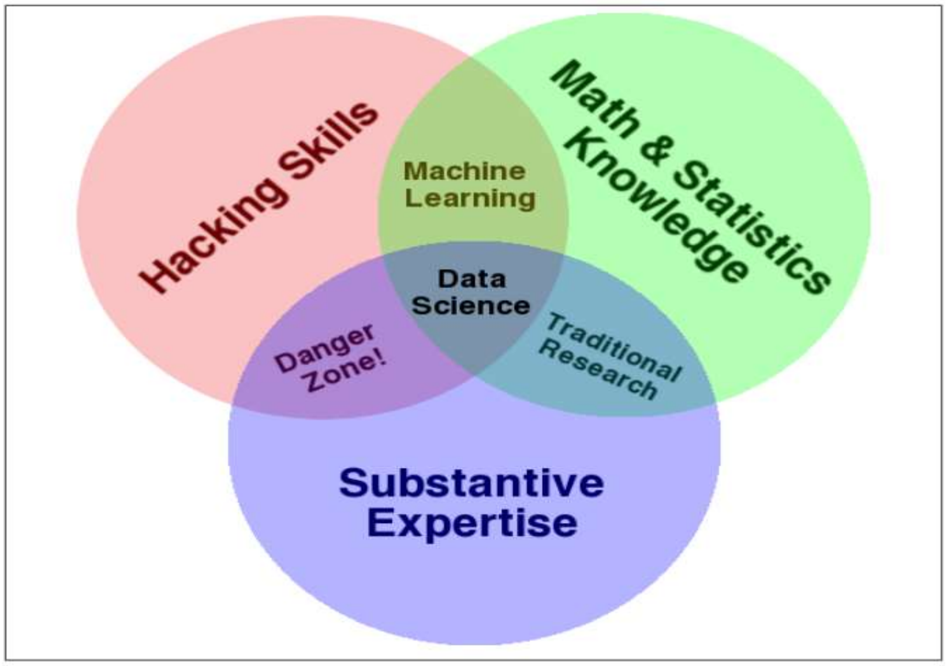


Figure 1-1. Drew Conway's Venn diagram of data science

He also mentions the sexy skills of data geeks from Nathan Yau's 2009 post, "Rise of the Data Scientist", which include:

• Statistics (traditional analysis you're used to thinking about)
• Data munging (parsing, scraping, and formatting data)
• Visualization (graphs, tools, etc.)

But wait, is data science just a bag of tricks? Or is it the logical extension of other fields like statistics and machine learning?

For a slightly different perspective, see ASA President Nancy Geller's 2011 Amstat News article, "Don't shun the 'S' word", in which she defends statistics:

We need to tell people that Statisticians are the ones who make sense of the data deluge occurring in science, engineering, and medicine; that statistics provides methods for data analysis in all fields, from art history to zoology; that it is exciting to be a Statistician in the 21st century because of the many challenges brought about by the data explosion in all of these fields.

Much of the development of the field is happening in industry, not academia. That is, there are people with the job title data scientist in companies, but no professors of data science in academia. (Though this may be changing.).

It makes sense to us that once the skill set required to thrive at Google working with a team on problems that required a hybrid skill set of stats and computer science paired with personal characteristics including curiosity and persistence spread to other Silicon Valley tech companies, it required a new job title. Once it became a pattern, it deserved a name. And once it got a name, everyone and their mother wanted to be one. It got even worse when *Harvard Business Review* declared data scientist to be the "Sexiest Job of the 21st Century".

Both LinkedIn and Facebook are social network companies. Often-times a description or definition of data scientist includes hybrid statistician, software engineer, and social scientist. This made sense in the context of companies where the product was a *social* product and still makes sense when we're dealing with human or user behavior.

Data Science Jobs

Data scientist is the Sexiest Job of the 21st Century. And here's one thing we noticed about most of the job descriptions: companies ask data scientists to be experts in computer science, statistics, communication, data visualization, *and* to have extensive domain expertise. Nobody is an expert in everything, which is why it makes more sense to create teams of people who have different profiles and different expertise—together, as a team, they can specialize in all those things. We'll talk about this more after we look at the composite set of skills in demand for today's data scientists.

2. Statistical Inference

Every data scientist should do once they've gotten data in hand for any data-related project: exploratory data analysis (EDA).

When you're developing your skill set as a data scientist, certain foundational pieces need to be in place first statistics, linear algebra, some programming. Even once you have those pieces, part of the challenge is that you will be developing several skillsets in parallel simultaneously—data preparation and munging, modeling, coding, visualization, and communication—that are interdependent. As we progress, we need to start somewhere, and will begin by getting grounded in statistical inference. it's always helpful to go back to fundamentals and remind ourselves of what statistical inference and thinking is all about. And further still, in the age of Big Data, classical statistics methods need to be revisited and reimagined in new contexts.

The world we live in is complex, random, and uncertain. At the same time, it's one big data-generating machine.

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.

Data represents the traces of the real-world processes, and exactly which traces we gather are decided by our data collection or sampling method. After separating the process from the data collection, we can see clearly that there are two sources of randomness and uncertainty.

A **mathematical model** for both uncertainty and randomness is offered by probability theory
A world/processes is defined by one or more variables. The model of the world is defined by a function:
Model = f (x) or f (x, y, z) (A multivariable function)
The function is unknown, model is unclear, at least initially. Typically, our task is to come up with the model, given the data.

The 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 or observations (the overall dataset)* immediately makes us think of the entire US population of 300 million people, or the entire world's population of 7 billion people.

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.
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.

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. There are different ways you might go about getting this subset of data, and you want to be aware of this sampling mechanism because it can introduce *biases* into the data, and distort it, so that the subset is not a "mini-me" shrunk-down version of the population. Once that happens, any conclusions you draw will simply be wrong and distorted.

In the BigCorp email example, you could make a list of all the employees and select 1/10th of those people *at random* and take all the email they ever sent, and that would be your sample. Alternatively, you could sample 1/10th of all email sent each day at random, and that would be your sample. Both these methods are reasonable, and both methods yield the same sample size. But if you took them and counted how many email messages each person sent, and used that to estimate the underlying *distribution* of emails sent by all individuals at BigCorp, you might get entirely different answers.

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 and comfortable with dealing a variety of types of data, including:

• Traditional: numerical, categorical, or binary
• Text: emails, tweets, *New York Times* articles
• Records: user-level data, timestamped event data, json formatted log files
• Geo-based location data: briefly touched on in this chapter
with NYC housing data
• Network
• Sensor data (not covered in this book)
• Images (not covered in this book)
These new kinds of data require us to think more carefully about what sampling means in these contexts.

Big Data and Statistical Inferences

A few ways to think about Big Data:

- **"Big" is a moving target**. Constructing a threshold for Big Data such as 1 petabyte is meaningless because it makes it sound absolute.
- **"Big" is when you can't fit it on one machine**. Different individuals and companies have different computational resources available to them, so for a single scientist data is big if she can't fit it on one machine because she has to learn a whole new host of tools and methods once that happens.
- **Big Data is a cultural phenomenon**. It describes how much data is part of our lives
- **The 4 Vs:** Volume, variety, velocity, and value. Many people are circulating this as a way to characterize Big Data.

An article "The Rise of Big Data." In it, they argue that the Big Data revolution consists of three things:

• Collecting and using a lot of data rather than small samples
• Accepting messiness in your data
• Giving up on knowing the causes

They describe these steps in a rather grand fashion by claiming that Big Data doesn't need to understand cause given that the data is so enormous.

Sample size N
For statistical inference $N < ALL$
For Bigdata $N=ALL$
For some typical bigdata analysis $N=1$, world model through the eyes of the twitter user

Modeling

A model is our attempt to understand and represent the nature of reality through a particular lens, be it architectural, biological, or mathematical.
$y = f(x)$
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.

Statistical Modeling

Before you get too involved with the data and start coding, it's useful to draw a picture of what you think the underlying process might be with your model. What comes first? What influences what? What causes what? What's a test of that?

But different people think in different ways. Some prefer to express these kinds of relationships in terms of math. The mathematical expressions will be general enough that they have to include parameters, but the values of these parameters are not yet known.

In mathematical expressions, the convention is to use Greek letters for parameters and Latin letters for data. So, for example, if you have two columns of data, *x* and *y*, and you think there's a linear relationship, you'd write down
$y = \beta_0 + \beta_1 x$.
You don't know what $\beta_0$and $\beta_1$are in terms of actual numbers yet, so they're the parameters.
Other people prefer pictures and will first draw a diagram of data flow, possibly with arrows, showing how things affect other things or what happens over time. This gives them an abstract picture of the relationships before choosing equations to express them.

So try writing down a linear function (more on that in the next unit). When you write it down, you force yourself to think: does this make *any* sense? If not, why? What would make *more sense*? 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. Some of the building blocks of these models are *probability distributions*.

Probability distributions

Probability distributions are the foundation of statistical models. When we get to linear regression and Naive Bayes, you will see how this happens in practice. One can take multiple semesters of courses on probability theory, and so it's a tall challenge to condense it down for you in a small section.

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 humans, following a *normal* distribution—a bell-shaped curve, also called a Gaussian distribution, named after Gauss.

Other common shapes have been named after their observers as well (e.g., the Poisson distribution and the Weibull distribution), while other shapes such as Gamma distributions or exponential distributions are named after associated mathematical objects.

Natural processes tend to generate measurements whose empirical shape could be approximated by mathematical functions with a few parameters that could be estimated from the data.

Not *all* processes generate data that looks like a *named* distribution, but many do. We can use these functions as building blocks of our models. It's beyond the scope of the book to go into each of the distributions in detail, but we provide them in Figure 2-1 as an illustration of the various common shapes, and to remind you that they only have names because someone observed them enough times to think they deserved names. There is actually an infinite number of possible distributions.

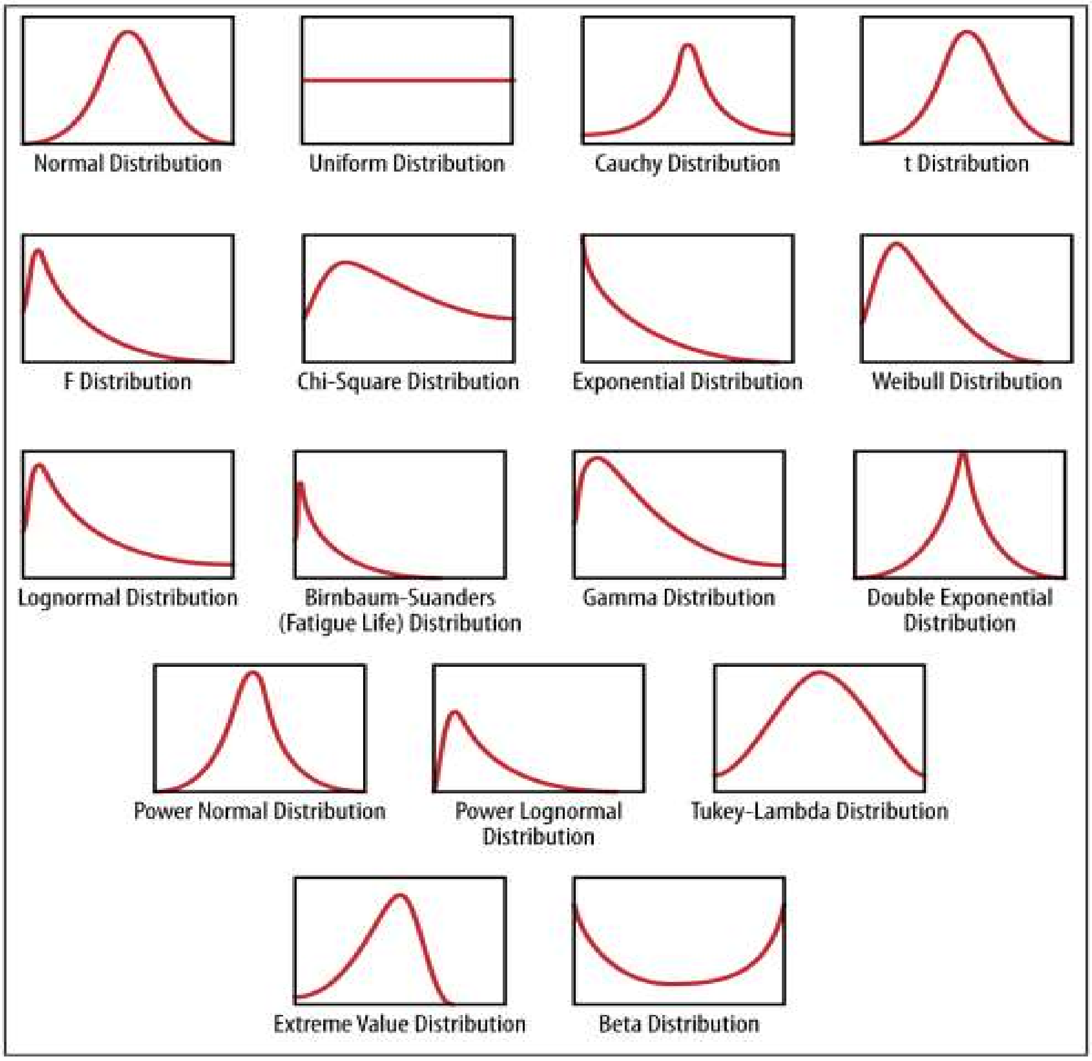


Figure 2-1. A bunch of continuous density functions (aka probability distributions)

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 μ is 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.

Normal, Uniform, Cauchy, t, F-, Chi Square, exponential, Weibull, lognormal etc. They are known as continuous density functions. Any random variable x or y can be assumed to have probability distribution p(x), if it maps it to a positive real number. For a probability density function, if we integrate the function to find the area under the curve it is 1, allowing it to be interpreted as probability.

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 *estimators*, 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.

Overfitting

Overfitting is **a concept in data science, which occurs when a statistical model fits exactly against its training data**. When this happens, the algorithm unfortunately cannot perform accurately against unseen data, defeating its purpose 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

Data Science Process

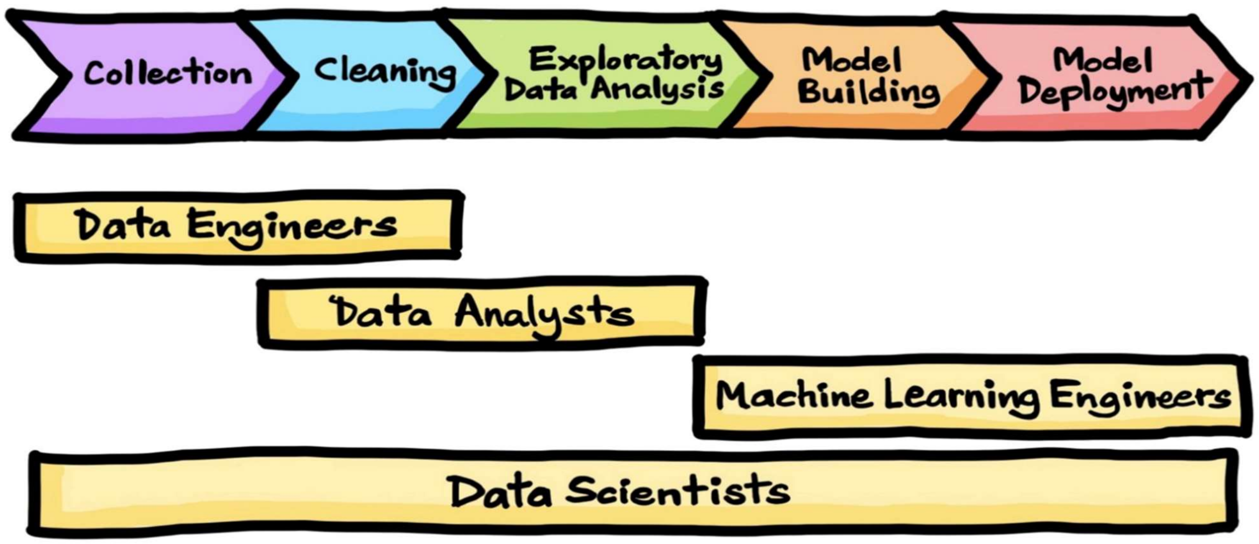
While data scientists often disagree about the implications of a given data set, virtually all data science professionals agree on the need to follow the data science process, which is a structured framework used to complete a data science project. There are many different frameworks, and some are better for business use cases, while others work best for research use cases.

This post will explain the most popular data science process frameworks, which ones are the best for each use case, and the fundamental elements comprising each one.

What Is the Data Science Process?

The data science process is a systematic approach to solving a data problem. It provides a structured framework for articulating your problem as a question, deciding how to solve it, and then presenting the solution to stakeholders.

Data Science Life Cycle

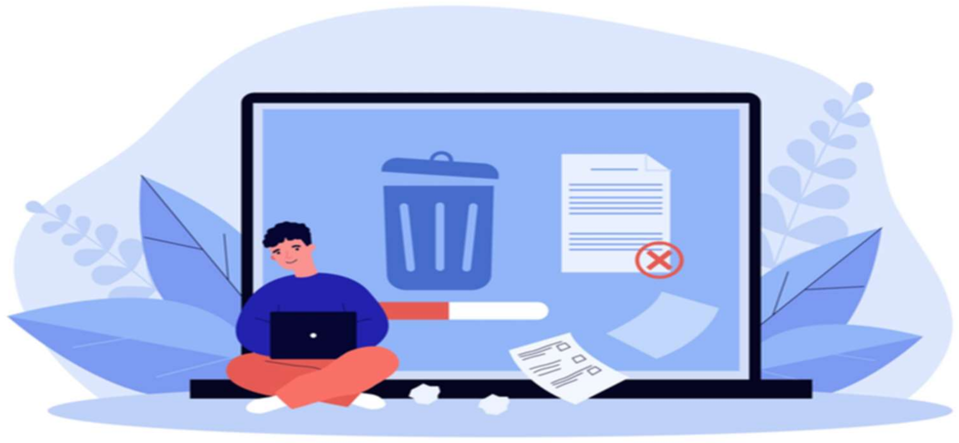


Another term for the data science process is the data science life cycle. The terms can be used interchangeably, and both describe a workflow process that begins with collecting data, and ends with deploying a model that will hopefully answer your questions. The steps include:

Framing the Problem
Understanding and framing the problem is the first step of the data science life cycle. This framing will help you build an effective model that will have a positive impact on your organization.

Collecting Data
The next step is to collect the right set of data. High-quality, targeted data—and the mechanisms to collect them—are crucial to obtaining meaningful results. Since much of the roughly 2.5 quintillion bytes of data created every day come in unstructured formats, you'll likely need to extract the data and export it into a usable format, such as a CSV or JSON file.

Cleaning Data



Most of the data you collect during the collection phase will be unstructured, irrelevant, and unfiltered. Bad data produces bad results, so the accuracy and efficacy of your analysis will depend heavily on the quality of your data.

Cleaning data eliminates duplicate and null values, corrupt data, inconsistent data types, invalid entries, missing data, and improper formatting.

This step is the most time-intensive process, but finding and resolving flaws in your data is essential to building effective models.

Exploratory Data Analysis (EDA)
Now that you have a large amount of organized, high-quality data, you can begin conducting an exploratory data analysis (EDA). Effective EDA lets you uncover valuable insights that will be useful in the next phase of the data science lifecycle.

Model Building and Deployment
Next, you'll do the actual data modeling. This is where you'll use machine learning, statistical models, and algorithms to extract high-value insights and predictions.

Lastly, you'll communicate your findings to stakeholders. Every data scientist needs to build their repertoire of visualization skills to do this.

Your stakeholders are mainly interested in what your results mean for their organization, and often won't care about the complex back-end work that was used to build your model. Communicate your findings in a clear, engaging way that highlights their value in strategic business planning and operation.

Basics of R:

R programming introduction: R is a scripting language for statistical data manipulation, statistical analysis, graphics representation and reporting. It was inspired by, and is mostly compatible with, the statistical language S developed by AT&T. R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team.

The core of R is an interpreted computer language which allows branching and looping as well as modular programming using functions. R allows integration with the procedures written in the C, C++, .Net, Python or FORTRAN languages for efficiency.

R is freely available under the GNU General Public License, and pre-compiled binary versions are provided for various operating systems like Linux, Windows and Mac.

FEATURES OF R: R is a programming language and software environment for statistical analysis, graphics representation and reporting. The following are the important features of R

- R is a well-developed, simple and effective programming language which includes conditionals,loops, user defined recursive functions and input and output facilities.
- R has an effective data handling and storage facility.
- **It incorporates features found in object-oriented and functional programminglanguages.**
- R provides a suite of operators for calculations on arrays, lists, vectors and matrices.
- R provides a large, coherent and integrated collection of tools for data analysis.
- R provides graphical facilities for data analysis and display either directly at the computer orprinting at the papers.
- R provides special data types that includes the Numerical, Integer , Character, Logical, Complex
- R provides very powerful data structures it contains Vectors, Matrices, List, Arrays, Data frames,Classes.
- Because R is open source software, it‘s easy to get help from the user community. Also, a lot ofnew functions are contributed by users, many of whom are prominent statisticians.

As a conclusion, R is world‘s most widely used statistics programming language.

How to Run R: R operates in two modes: *interactive* and *batch mode*.

1. Interactive mode: The one typically used is interactive mode. In this mode, you type in commands, R displays results, you type in more commands, and so on.

2. Batch mode: Batch mode does not require interaction with the user. It‘s useful for production jobs, such as when a program must be run periodically; say once per day, because you can automate the process.

R Command Prompt: Once you have R environment setup, then it‘s easy to start your R command prompt by just typing the following command at your command prompt −

```
C:\Users\CSELAB>R
```

This will launch R interpreter and you will get a prompt > where you can start typing your program as follows −

```
>myString<-"WELCOME R PROGRAMMING"
>print(myString)
[1] "WELCOME R PROGRAMMING"
```

Here first statement defines a string variable myString, where we assign a string "Hello, World!" and then next statement print() is being used to print the value stored in variable myString.

Object orientation : can be explained by example. Consider statistical regression. When you perform a regression analysis with other statistical packages, such as SAS or SPSS, you get a mountain of output on the screen. By contrast, if you call the lm() regression function in R, the function returns an object containing all the results—the estimate coefficients, their standard errors, residuals, and so on. You then pick and choose, programmatically, which parts of that object to extract. You will see that R‘s approach makes programming much easier, partly because it offers a certain uniformity of access to data.

Functional Programming:

As is typical in functional programming languages, a common theme in R programming is avoidance of explicit iteration. Instead of coding loops, you exploit R‘s functional features, which let you express iterative behavior implicitly. This can lead to code that executes much more efficiently, and it can make a huge timing difference when running R on large data sets.

The functional programming nature of the R language offers many advantages:
- Clearer, more compact code
- Potentially much faster execution speed
- Less debugging, because the code is simpler
- Easier transition to parallel programming.

Comments: Comments are like helping text in your r program and they are ignored by the interpreter while executing your actual program. single comment is written using # in the beginning of the statement as follows

```
# My first program in R Programming
```

R does not support multi-line comments but you can perform a trick which is something as follows −

```
if(FALSE){
"This is a demo for multi-line comments and it should be put inside either a single
```

```
      OR double quote"
}
myString<-"Hello, World!" print(myString)
```

Variables in R

Variables are used to store data, whose value can be changed according to our need. Unique name given to variable (function and objects as well) is identifier.

Rules for writing Identifiers in R

1. Identifiers can be a combination of letters, digits, period (.) and underscore (_).
2. It must start with a letter or a period. If it starts with a period, it cannot be followed by a digit.
3. Reserved words in R cannot be used as identifiers.

Valid identifiers in R

total, Sum, .fine.with.dot, this_is_acceptable, Number5

Invalid identifiers in R

tot@l, 5um, _fine, TRUE, .0ne

Best Practices

Earlier versions of R used underscore (_) as an assignment operator. So, the period (.) was used extensively in variable names having multiple words. Current versions of R support underscore as a valididentifier but it is good practice to use period as word separators. For example, a.variable.name is preferred over –a_variable_name– or alternatively we could use camel case as aVariableName

Constants in R

Constants, as the name suggests, are entities whose value cannot be altered. Basic types ofconstant are numeric constants and character constants.

Numeric Constants

All numbers fall under this category. They can be of type integer, double or complex. This can be checkedwith the typeof() function. Numeric constants followed by L are regarded as integer and those followed by
i are regarded as complex.

>typeof(5) [1] "double"	>typeof(5L) [1] "integer"	>typeof(5i) [1] "complex"

Character Constants: Character constants can be represented using either single quotes (') or double quotes (") as delimiters.

```
 > 'example'
[1] "example"

>typeof("5")
 [1] "character"
```

Built-in Constants

Some of the built-in constants defined in R along with their values is shown below.

```
> LETTERS
```

```
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P"
"Q" "R" "S"[20] "T" "U" "V" "W" "X" "Y" "Z"

>letters
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r"
"s"[20] "t" "u" "v" "w" "x" "y" "z"

>pi
 [1] 3.141593

>  month.name
 [1] "January"   "February"  "March"      "April"     "May"       "June"       [7] "July"
                                          "August""September" "October"   "November"
"December"
>month.abb
 [1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov" "Dec"
```

But it is not good to rely on these, as they are implemented as variables whose values can be changed.

```
>pi
[1] 3.141593

>pi<- 56
>pi [1] 56
```

Maths in R Programming: R Having Number Of Mathematical Calculations in R .It Contains Given Below

Program	Output	Program	Output
>a<-99 >print(a)	[1] 99	> z<-c(1:10)	>z [1] 1 2 3 4 5 6 7 8 9 10
> 2+2	[1] 4	> z1<-c(-5:5)	> z1 [1] -5 -4 -3 -2 -1 0 1 2 3 4 5
> 3-2	[1] 1	>seq(-5,5)	[1] -5 -4 -3 -2 -1 0 1 2 3 4 5
> 6/2	[1] 3	> seq(-5,5,by=2)	[1] -5 -3 -1 1 3 5
> 5*5	[1] 25	>seq(from=1,to=10,by=10)	[1] 1
>log(10)	[1] 2.302585	>seq(from=1,to=100,by=10)	[1] 1 11 21 31 41 51 61 71 81 91
>exp(1)	[1] 2.718282	>seq(from=0,to=100,by=10)	[1] 0 10 20 30 40 50 60 70 80 90 100
>exp(2)	[1] 7.389056	>rep("rao",5)	[1] "rao" "rao" "rao" "rao" "rao"
>exp(2)*exp(1)	[1] 20.08554	>rep(1:5,4)	[1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
>log10(6)	[1] 0.7781513	>rep("R programming",9)	[1] "R programming" "R programming" "R programming" "R programming" "R programming" "R programming" "R programming" "R programming" "R programming"
>log(16,4)	[1] 2		
>pi	[1] 3.141593		
>sin(2)	[1] 0.9092974		
>floor(9)	[1] 9		
>floor(12.5)	[1] 12		
>ceiling(13.5)	[1] 14		
>round(15.5)	[1] 16		
>abs(-1)	[1] 1		
>sqrt(100)	[1] 10		

R Session:

Let‘s make a simple data set (in R parlance, a *vector*) consisting of the numbers 1, 2, and 4, and name it x:

```
> x <- c(1,2,4)
```

The standard assignment operator in R is **<-**. You can also use =, but this is discouraged, as it does not work in some special situations. Note that there are no fixed types associated with variables. Here, we‘ve assigned a vector to x, but later we might assign something of a different type to it. The c stands for *concatenate*. Here, we are concatenating the numbers 1, 2, and 4. More precisely, we are concatenating three one-element vectors that consist of those numbers. This is because any number is also considered to be a one-element vector.

Now we can also do the following:

> q <- c(x,x,8) which sets q to (1,2,4,1,2,4,8) (yes, including the duplicates).

Now let‘s confirm that the data is really in x. To print the vector to the screen, simply type its name. If you type any variable name (or, more generally, any expression) while in interactive mode, R will print out the value of that variable (or expression). Programmers familiar with other languages such as Pythonwill find this feature familiar. For our example, enter this:

```
> x
[1] 1 2 4
```

Yep, sure enough, x consists of the numbers 1, 2, and 4.

Individual elements of a vector are accessed via [ ]. Here‘s how we can print out the third element of x:

```
> x[3]
[1] 4
```

As in other languages, the selector (here, 3) is called the *index* or *subscript*. Those familiar with ALGOL-family languages, such as C and C++, should note that elements of R vectors are indexed starting from 1, not 0.

Subsetting is a very important operation on vectors. Here‘s an example: > x <- c(1,2,4)

```
> x[
2:3]
[1] 2
4
> mean(
x) [1]
2.333333
> sd(x)
[1] 1.527525
```

Introduction to Functions: As in most programming languages, the heart of R programming consists of writing *functions*. A function is a group of instructions that takes inputs, uses them to compute other values, and returns a result.
As a simple introduction, let‘s define a function named oddcount(), whose purpose is to count the odd numbers in a vector of integers.

```
# counts the number of odd integers in x
> oddcount <- function(x) {
+ k <- 0 # assign 0 to k
+ for (n in x) {
+ if (n %% 2 == 1) k <- k+1 # %% is the modulo operator + }
+ return(k)
+ }
> oddcount(c(1,3,5))
[1] 3
> oddcount(c(1,2,3,7,9))
[1] 4
```

Arithmetic Operators: These operators are used to carry out mathematical operations like addition and multiplication. Here is a list of arithmetic operators available in R.

	Arithmetic Operators in R
Operator	**Description**
+	Addition
-	Subtraction
*	Multiplication
/	Division
^	Exponent
%%	Modulus (Remainder from division)
%/%	Integer Division (no Fractional part)

An example run

```
> x <-5
> y <-16
>x+
y
[1]2
1
>x-y
[1]-
11
>x*
y
[1]8
0
>y/x
[1]3.
2
>y%/%x
[1]3
>y%%
x[1]1
>y^x
[1]10485
76
```

Relational Operators: Relational operators are used to compare between values. Here is a list ofrelational operators available in R.

Relational Operators in R			
Operator	**Description**	==	**Equal to**
<	**Less than**	!=	**Not equal to**
>	**Greater than**		
<=	**Less than or equal to**		
>=	**Greater than or equal to**		

An example run

```
> x <-5
> y <-16
```

> x<y [1] TRUE	> x>y [1] FALSE	> x<=5 [1] TRUE	> y>=20 [1] FALSE	>x !=5 [1] FALSE	>y==16 [1] TRUE

Operation on Vectors: The above mentioned operators work on vectors. The variables used above werein fact single element vectors. We can use the function c() (as in concatenate) to make vectors in R. All operations are carried out in element-wise fashion. Here is an example.

```
> x <-c(2,8,3)
> y <-c(6,4,1)
```

>x+y [1]8 12 4	> x>y [1]FALSE TRUE TRUE

When there is a mismatch in length (number of elements) of operand vectors, the elements in shorter one is recycled in a cyclic manner to match the length of the longer one. R will issue a warning if the length of the longer vector is not an integral multiple of the shorter vector.

```
> x <-c(2,1,8,3)
> y <-c(9,4)
>x+y# Element of y is recycled to 9,4,9,4
[1]115177

>x-1# Scalar 1 is recycled to 1,1,1,1
[1]1072
>x+c(1,2,3)
[1]33114 Warning message:
In x +c(1,2,3): longer object length is not a multiple of shorter object length
```

Logical Operators: Logical operators are used to carry out Boolean operations like AND, OR etc.

	Logical Operators in R
Operator	**Description**
!	Logical NOT
&	Element-wise logical AND
&&	Logical AND
|	Element-wise logical OR
||	Logical OR

Operators & and | perform element-wise operation producing result having length of the longer operand. But && and || examines only the first element of the operands resulting into a single length logical vector. Zero is considered FALSE and non-zero numbers are taken as TRUE. An example run.

```
>x <-c(TRUE,FALSE,0,6) >   y <-c(FALSE,TRUE,FALSE,TRUE)
>!x
[1]FALSE  TRUETRUE FALSE
>x&y
[1] FALSE FALSEFALSE  TRUE
>x&&y
[1] FALSE
>x|y
[1]  TRUETRUE FALSE  TRUE
 > x||y
[1] TRUE
```

Assignment Operators: These operators are used to assign values to variables.

Assignment Operators in R	
Operator	Description
<-, <<-, =	Leftwards assignment
->, ->>	Rightwards assignment

The operators <- and = can be used, almost interchangeably, to assign to variable in the same environment. <<- is used for assigning to variables in the parent environments (more like global assignments). The rightward assignments, although available are rarely used.

```
> x <-5
>x [1]5
> x =9
>x [1]9
>10-> x
>x [1]10
```

R Program to Add Two Vectors:

We can add two vectors together using the + operator. One thing to keep in mind while adding (or other arithmetic operations) two vectors together is the recycling rule. If the two vectors are of equal length then there is no issue. But if the lengths are different, the shorter one is recycled (repeated) until its length is equal to that of the longer one. This recycling process will give a warning if the longer vector is not an integral multiple of the shorter one.

Source Code

```
>x
[1]3 6 8
 >y
[1]2 9
0

> x + y
[1]5 15
8

> x +1# 1 is recycled to (1,1,1)
[1]4 7 9

> x +c (1, 4)# (1,4) is recycled to (1,4,1) but warning issued
[1]4 10 9 Warning message:
In x +c(1,4): longer object length is not a multiple of shorter object length
```

As we can see above the two vectors x and y are of equal length so they can be added together without difficulty. The expression x + 1 also works fine because the single 1 is recycled into a vector of three 1's. Similarly, in the last example, a two element vector is recycled into a three element vector. But a warning is issued in this case as 3 is not an integral multiple of 2.

Data types in R: To make the best of the R language, you'll need a strong understanding of the basic data types and data structures and how to operate on those.

- character
- numeric (real or decimal)
- integer
- logical
- complex

Example	**Type**
"a", "swc"	character
2, 15.5	numeric
2 (Must add a L at end to denote integer)	intege
rTRUE, FALSE	logica
l	
1+4i	complex

```
TRUE
[1] TRUE
>class(TRUE)
[1] "logical"
>class(FALSE)
[1] "logical"
> T
```

```
[1] TRUE
> F
[1] FALSE
>class(2)
[1] "numeric"
>class(2L)
[1] "integer"
>class(2i)
[1] "complex"
>class("i love R programming")
[1] "character"
```

UNIT-I QUESTIONS:

1. What is Data Science and Big Data? Why Data Science is Hype now a days? Why not in the earlier.
2. What is Datafication? Describe the Current Landscape of Perspectives.
3. Express the statistical modeling and probability normal distribution
4. What is a model and mathematical model? What do you mean by fitting a model? Describe Overfitting.
5. Describe the Data Science Process
6. Express the following R Programming Fundamentals
   - a) Variables and their types
   - b) Types of Operators
   - c) Built-in Variables and functions
   - d) Math functions
7. Explain the conditional control and Iterative control statements with examples in R.
8. Describe different data structures in R language with examples