Karpagam Academy of Higher Education

(Deemed to be University)
(Established Under Section 3 of UGC Act 1956)
Pollachi Main Road, Eachanari (p.o)
Coimbatore – 641 021

Department of CS, CA and IT

Subject Name: Introduction to Data Science **Semester:V**

Subject Code : 16CTU503A **Class : III B.Sc(CT)**

Staff Name : SIVAGAMINATHAN P.G

LECTURE PLAN

S. No	Lecture Duration (Hr)	Topics to be Covered	Support Materials
		UNIT – I	
1.	1	Introduction to Bigdata	W1
2.	1	Introduction to data science	W1
3.	1	Data Scientist's tool box	W1
4.	1	Data Science-discovery of insight	W2
5.	1	Data science as a multidisciplinary field	W2
6.	1	Turning data in to actionable knowledge	W2
7.	1	Sensing and data collection	W3
8.	1	Tools used in building analysis software	W1,W2
9.	1	Version Control, Git and GitHub	W2,W3
10.	1	Recapitulation and discussion of important questions	
	Total no. of Hours		**10**
		UNIT II	
1.	1	Basics Of R programming	S7
2.	1	R data types	S7
3.	1	R objects	S7
4.	1	Reading data and writing data	S7
5.	1	Control structures	S7

6.	1	Functions, scoping rules	S7,W4
7.	1	Dates and times, loop functions	S7,W4
8.	1	Debugging tools	S7,W4
9.	1	Simulation, code profiling	W4
10.	1	Recapitulation and discussion of important questions	
	Total no.of hours		**10**
UNIT – III			
1.	1	Introduction	S2
2.	1	Significance of clear data	W5
3.	1	Obtaining data from web	W5
4.	1	Obtaining data from API's	W5
5.	1	Obtaining data from databases	W5
6.	1	Obtaining data from colleagues	W5
7.	1	Obtaining data in various formats	W7
8.	1	Data cleaning, need for tidy data	W7
9.	1	Recapitulation and Discussion of important questions	
	Total no. of Hours		**9**
UNIT – IV			
1.	1	**Introduction**	
2.	1	Exploratory data analysis	W11
3.	1	Essential techniques for summarizing data	W11
4.	1	Eliminating or sharpening potential hypotheses	W11
5.	1	Boxplot ,bar plot using R	S7
6	1	gg plot using R	S7
7.	1	Univariate analysis, multivariate analysis	W11
8.	1	Multivariate analysis to visualize high dimensional data	W11
9.		Recapitulation and Discussion of important questions	
	Total no. of Hours		**9**

UNIT – V			
1.	1	Introduction	W12
2.	1	Reproducible research, reporting tools	W12
3.	1	Importance of mark down in data science	W12
4.	1	To write a document using R mark down	W12
5.	1	Integrate live R code into a statistical program	W12
6.	1	Compiling R mark down documents, knit r tools	W12
7.	1	Reproducible and repeatable research	W12
8.	1	Previous ESE Question Paper Discussion	
9.	1	Previous ESE Question Paper Discussion	
10.	1	Previous ESE Question Paper Discussion	
	Total no. of Hours		**10**
	Total no of hours planned for this course		**48 hours**

Suggested Readings:

S1: Rachel Schutt, Cathy O'Neil(2013), Doing data science: Straight talk from the frontline, S.Chroff/O'Reilly

S2: Foster Provost, Tom Fawcett(2013), Data Science for Business-What you need to know about Datamining and Data Analytic Thinking, O'Reilly

S3: John W Foreman(2013), Data smart: using data science to transform information into insight, John Wiley and Sons.

S4:Ian Ayres(2007), Super Crunchers: Why Thinking-By-Numbers is the New way to be Smart(1st Edition), Bantam

S5:Eric Seigel(2013), Predictive Analytics: The power of predict who will click, Buy Lie or Die(1st Edition), Wiley

S6: Mathew A.Russell(2013),Mining the social webL:Data mining facebook, Twitter, LinkedIn, Google+, GitHub, and more (2nd Edition), O'Reilly Media

S7: G.Sudhamathy and C.Jothi Venkateswaran, "R Programming An Approach to Data Analytics", MJP Publishers

Websites:

W1 :https://www.planet-data.eu/sites/default/files/presentations/Big_Data_Tutorial_part4.pdf

W2: https://guides.github.com/pdfs/markdown-cheatsheet-online.pdf

W3: https://www.tutorialspoint.com/git/index.htm

W4: https://libguides.library.kent.edu/statconsulting/r

W5: https://www.coursera.org/learn/data-cleaning

W6: https://www.class-central.com › Coursera

W7: datasciencespecialization.github.io/getclean/

W8: https://www.analyticsvidhya.com/.../an-introduction-to-apis-application-programming.

W9: https://www.earthdatascience.org/courses/...analytics/get-data...apis/API-data-access-r

W10: https://www.coursera.org/learn/sql-data-science

W11: https://www.analyticsvidhya.com/blog/tag/exploratory-data-analysis/

W12: https://moodle.epfl.ch/.../Reproducible_Research_in_Computational_Science-Science-...

FACULTY **HOD**

16CTU503A INTRODUCTION TO DATA SCIENCE 4H – 4C

Instruction Hours / week: L: 4 T: 0 P: 0 **Marks:** Int : **40** Ext : **60** Total: **100**

SCOPE

This course gives an introduction to the basics of data science and leave armed with practical experience extracting value from big data.

OBJECTIVES

- Building a comprehensive working knowledge and expertise around various analytical and database tools which is a key step to excel in big data and data science fields.
- The data science course covers topics in a comprehensive manner with applications of R Programming

UNIT-I

Data Scientist's Tool box: Turning data into actionable knowledge, introduction to the tools that will be used in buliding data analysis software; version control, markdown, git, GitHub, R and R-Studio.

UNIT-II

R Programming basics: Overview of R, R datatypes and Objects, reading and writing data, control structures, functions, scoping rules, dates and tiems, loop functions, dubugging tools, simualtion, code profiling.

UNIT-III

Getting and cleaning data: Obtaining data from the web, from API's , from database, and from colleagues in various formats. Basics of data cleaning and making data –tidy.

UNIT-IV

Exploratory data analysis: Essential exploratory techniques for summarzing data, applied before formal modelling commences, eliminating or sharpening potential hypotheses about the world that can be addresses by the data, common multivariate statistical techniques used to visualize high-dimensional data.

UNIT-V

Reproducible Research: Concepts and tools behind reporting modern data analysis in a reproducible manner, To write a document using R markdown, integrate live R code into a literate statistical program, compile R markdown documents using knitr and related tools, and organize a data analysis so that it is reproducible and accessible to others.

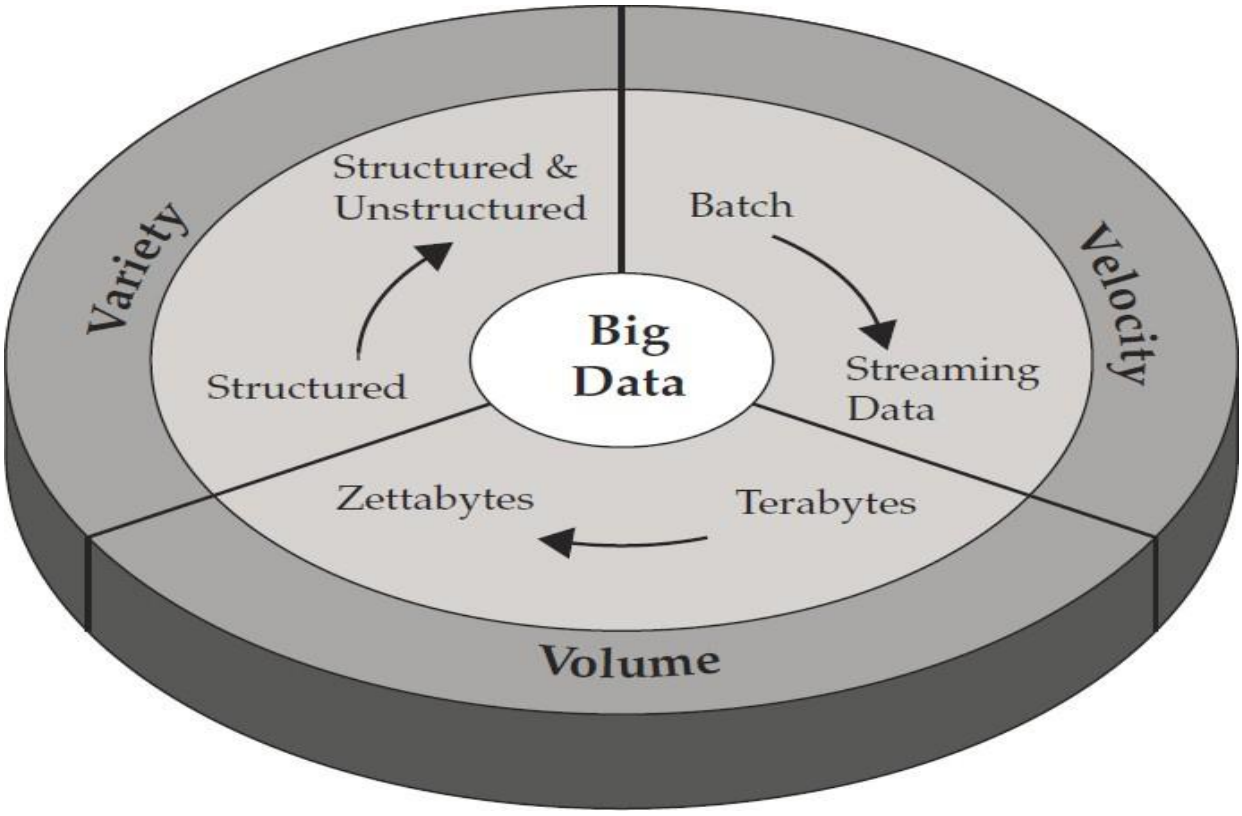
Suggested Readings

1. Rachel Schutt, Cathy O'Neil (2013). Doing data science: Straight Talk from the frontline. S.Chroff/O' Reilly
2. Foster Provost, Tom Fawcett (2013). Data science for Business – What you need to know about Datamining and Data Analytic Thinking. O'Reilly
3. John. W. Foreman (2013). Data Smart: Using Data science to transform inforamtion into insight. John Wiley and Sons.
4. Ian Ayres (2007). Super Crunchers: Why Thinking – By – Numbers is the New way to Be Smart (1st ed.). Bantam
5. Eric Seigel (2013). Predictive Analytics: The Power of Predict who will click, Buy Lie or Die (1st ed.). Wiley.
6. Matthew A.Russel (2013). Mining the social webL: Datamining Facebook, Twitter, LinkedIn, Google+, GitHub, and More (2nd ed.). O'Reilly Media.

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established under section 3 of UGC Act 1956)
Pollachi main road, Eachanari post, Coimbatore -21
Class: III B.Sc(CT)
Code :16CTU503A Introduction to Data Science (2016-2019)

Unit I

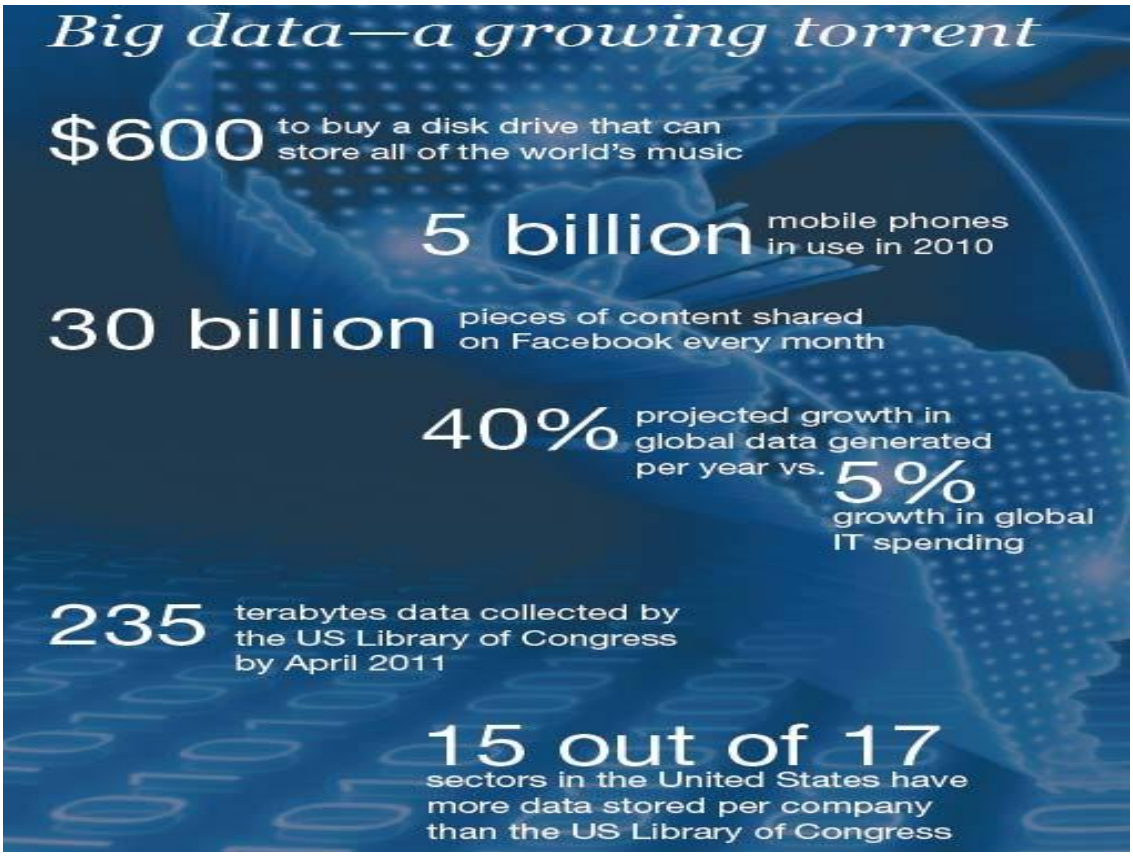
Data scientist tool box: Turning data into actionable knowledge, Introduction to the tools that will be used in building data analysis software: version control, mark down, Git, GitHub, R and R –Studio



What is Big data?

Big Data is defined as high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation.

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established under section 3 of UGC Act 1956)
Pollachi main road, Eachanari post, Coimbatore -21
Class: III B.Sc(CT)
Code :16CTU503A Introduction to Data Science (2016-2019)



Introduction to Bigdata – Organizations are capturing storing and analyzing data that has high volume, velocity, and variety of data comes from variety of sources including social media, machines, logfiles, video, text, image, RFID and GPS. These sources have strained the capabilities of traditional relational database management systems and spawned a host of new technologies, approaches, and platforms. The potential value of big data analytics is great and is clearly established by a growing number of studies. There are keys to success with big data analytics, including clear business speed, strong committed sponsorship, alignment between the business and IT strategies, a fact-based decision making culture, a strong data infrastructure, the right analytical tools and people skilled in the use of analytics. Because of the paradigm shift in the kind of data being analyzed and how this data is used as a 4th generation decision support

system. Though the business value from big data is great, especially for online companies like Google and Facebook, how it is being used is raising significant privacy concerns.

Big data and analytics are intertwined, but analytic techniques such as regression analysis, simulation, and machine learning has been used in DSS. The value in analyzing unstructured data such as email and documents are well understood. The value addition is the advances in technology and software, new sources of data(e.g social media) and business opportunity. It is even spawning a new area of practice and study called "data science" that encompasses the techniques, tools, technologies and processes for making sense out of big data.

Big data is changing existing jobs and creating new ones. For example, market researchers must now be skilled in social media analytics. Data management professionals must be able to store massive amount of data of any structure. The job of data scientist the" high priest" of big data analytics has emerged. Many companies are seeking people with big data skills, many universities, are offering new courses, certificates and degree programs to provide students with the needed skills. Vendors like IBM provide education to faculty and students through their university support programs. At a high level, the requirements for organizational success with bug data analytics are the same as business intelligence [BI], at a deeper level(micro level) there are many nuances to be considered for a cost effective decision making.

Governments and companies are able to integrate personal data from numerous sources and learn much of what you do, where you go, who your friends are, and what are your preferences. This leads to better services which in turn making profit, it also raises privacy concerns. There are legal restrictions on what big data companies such as Facebook, and Google can do with the data they collect. From an evolutionary perspective bigdata is not new, A major reason for creating data warehouses in 1990's was to store large amount of data. Later, terabyte of data was considered as bigdata. Teradata a company has more than 35 customers such as Walmart, Verizon with data warehouses over a petabyte in size. eBay captures a terabytes of data per minute and maintains over 40 petabytes, most of any company in the world. Some people consider 10 terabytes to be bigdata, but any numerical definition is likely to change over time as organizations collect, store and analyse more data.

Big Data is a data analysis methodology enabled by recent advances in technologies that support high-velocity data capture, storage and analysis. Data sources extend beyond the traditional corporate database to include emails, mobile device outputs, and sensor-generated data where data is no longer restricted to structured database records but rather unstructured data having no

standard formatting. Since Big Data and Analytics is a relatively new and evolving phrase, there is no uniform definition; various stakeholders have provided diverse and sometimes contradictory definitions. One of the first widely quoted definitions of Big Data resulted from the Gartner report of 2001. **Gartner** proposed that, Big Data is defined by three V's volume, velocity, and variety. Gartner expanded its definition in 2012 to include veracity, representing requirements about trust and uncertainty pertaining to data and the outcome of data analysis. In a 2012 report, IDC defined the 4th V as value—highlighting that Big Data

Applications need to bring incremental value to businesses. Big Data Analytics is all about processing unstructured information from call logs, mobile-banking transactions, online user generated content such as blog posts and tweets, online searches, and images which can be transformed into valuable business information using computational techniques to unveil trends and patterns between datasets.

Another dimension of the Big Data definition involves technology. Big Data is not only large and complex, but it requires innovative technology to analyze and process. In 2013, the National Institute of Standard and Technology (NIST) Big Data workgroup proposed the following definition of Big Data that emphasizes application of new technology; Big Data exceed the capacity or capability of current or conventional methods and systems, and enable novel approaches to frontier questions previously inaccessible or impractical using current or conventional methods. Business challenges rarely show up in the appearance of a perfect data problem, and even when data are abundant, practitioners have difficulties to incorporate it into their complex decision-making that adds business value. In 2012,

McKinsey & Company conducted a survey of 1,469 executives across various regions, industries and company sizes, in which 49 percent of respondents said that their companies are focusing big data efforts on customer insights, segmentation and targeting to improve overall performance An even higher number of respondents 60 percent said their companies should focus efforts on using data and analytics to generate these insights. Yet, just one-fifth said that their organizations have fully deployed data and analytics to generate insights in one business unit or function, and only 13 percent use data to generate insights across the company. As these survey results show, the question is no longer whether big data can help business, but how can business derive maximum results from big data.

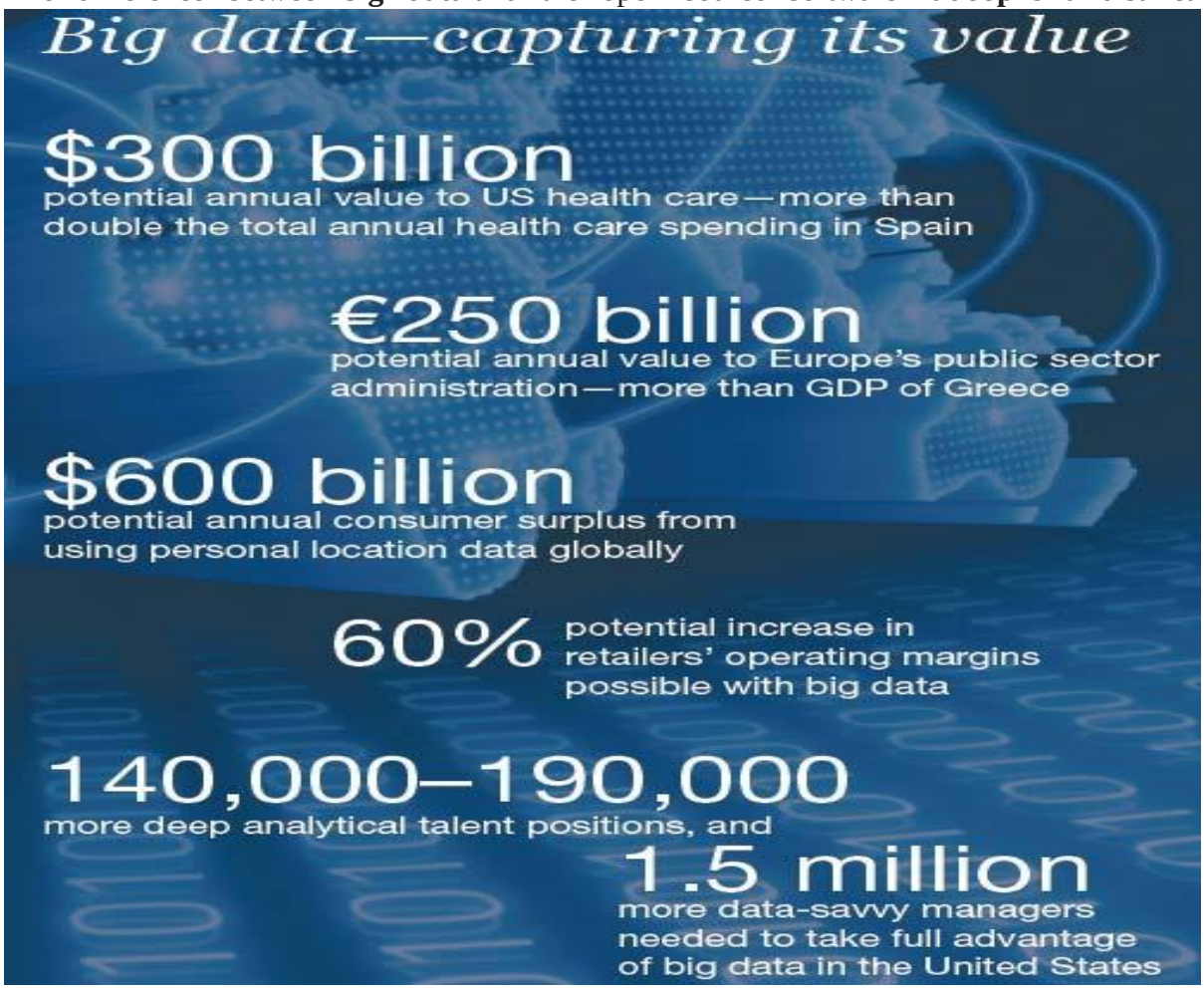
KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established under section 3 of UGC Act 1956)
Pollachi main road, Eachanari post, Coimbatore -21
Class: III B.Sc(CT)
Code :16CTU503A Introduction to Data Science (2016-2019)

What is big data and bigdata analytics?

Big data analytics is the process of examining large and varied data sets -- i.e., big data -- to uncover hidden patterns, unknown correlations, market trends, customer preferences and other useful information that can help organizations make more-informed business decisions.

Is Hadoop related to Big data?

Big Data is nothing but a concept which facilitates handling large amount of **data**sets. **Hadoop** is just a single framework out of dozens of tools. **Hadoop** is primarily used for batch processing. The difference between **big data** and the open source software **Hadoop** is a distinct and



What is Hadoop and bigdata?

Hadoop is an open-source software framework for storing **data** and running applications on clusters of commodity hardware. It provides massive storage for any kind of **data**, enormous processing power and the ability to handle virtually limitless concurrent tasks or jobs.

What are big data tools?

- Apache Hadoop. Apache Hadoop is a java based free software framework that can effectively store large amount of data in a cluster. ...
- Microsoft HDInsight. ...
- NoSQL. ...
- Hive. ...
- Sqoop. ...
- PolyBase. ...
- Big data in EXCEL. ...
- Presto.

Is bigdata and data science are same?

Broadly speaking, Big Data Analytics can be called Data Science, but Data Science cannot be called Big Data Analytics. On the surface, they perform the same operation – i.e. mining useful information from data.Machine Learning is what makes it different from Analytics.

Why big data is so important?

Big data is a given in the health care industry. ... That's why big data analytics technology is so important to heath care. By analyzing large amounts of information – both structured and unstructured – quickly, health care providers can provide lifesaving diagnoses or treatment options almost immediately.

What are the 3V's of bigdata?

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established under section 3 of UGC Act 1956)
Pollachi main road, Eachanari post, Coimbatore -21
Class: III B.Sc(CT)
Code :16CTU503A Introduction to Data Science (2016-2019)

3Vs (volume, variety and velocity) are three defining properties or dimensions of big data. Volume refers to the amount of data, variety refers to the number of types of data and velocity refers to the speed of data processing.

Data Scientist Tool box: <<annexure I attached>>

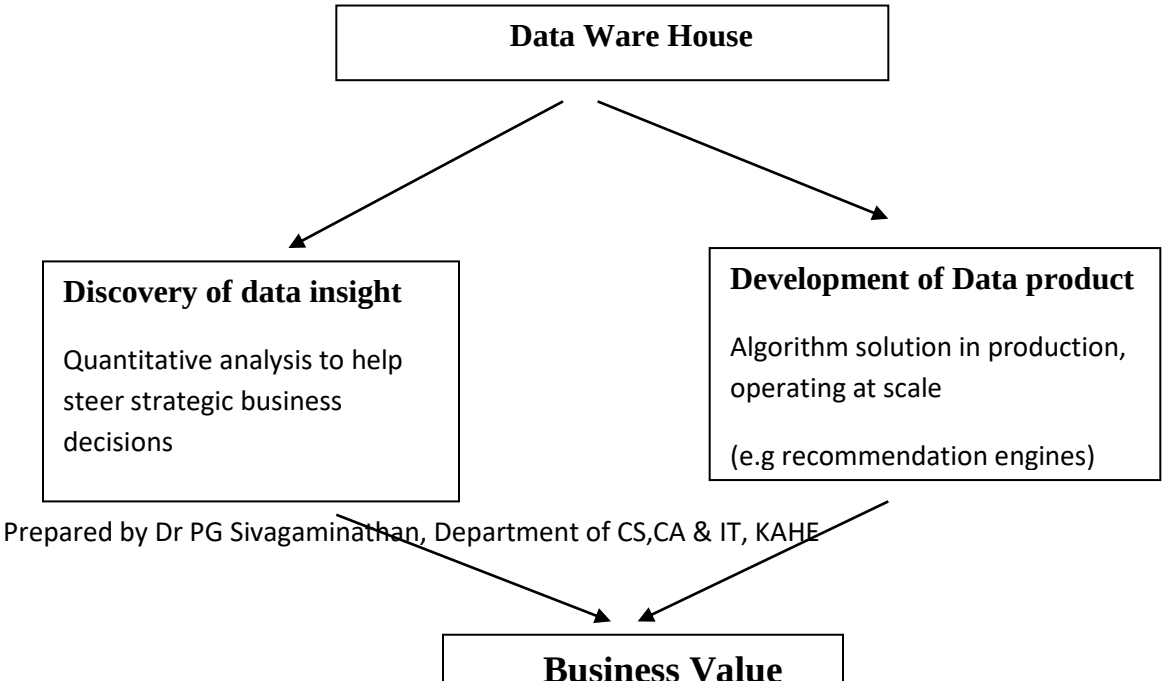
Data Science:Introduction

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from data in various forms, both structured and unstructured similar to data mining.

Data science is a "concept to unify statistics, data analysis, machine learning and their related methods" in order to "understand and analyze actual phenomena" with data. It employs techniques and theories drawn from many fields within the context of mathematics, statistics, information science, and computer science.

Data science is a multidisciplinary blend of data inference, algorithm development, and technology in order to solve analytically complex problems. At the core is data. Troves of raw information, streaming in and stored in enterprise data warehouses by mining it. Advanced capabilities we can build with it. Data science is ultimately about using this data in creative ways to generate business value.

Data Warehouse



Data science – discovery of data insight

Insight is the value obtained through the use of analytics. The insights gained through analytics are incredible powerful, and can be used to grow your business while identifying areas of opportunity.

Example of an insight: **Insight** is sometimes called an epiphany, an "aha" moment or a "eureka' feeling when a solution to a problem presents itself suddenly. Buddhists use meditation to help solve problems using **insight** knowledge or "vipassana nana."

This aspect of data science is all about uncovering findings from data. Diving in at a granular level to mine and understand complex behaviors, trends, and inferences. It's about surfacing hidden insight that can help enable companies to make smarter business decisions. For example:

Netflix data mines movie viewing patterns to understand what drives user interest, and uses that to make decisions on which Netflix original series to produce.

Target identifies what are major customer segments within it's base and the unique shopping behaviors within those segments, which helps to guide messaging to different market audiences.

Proctor & Gamble utilizes time series models to more clearly understand future demand, which help plan for production levels more optimally.

How do data scientists mine out insights? It starts with data exploration. When given a challenging question, data scientists become detectives. They investigate leads and try to

understand pattern or characteristics within the data. This requires a big dose of analytical creativity.

Then as needed, data scientists may apply quantitative technique in order to get a level deeper – e.g. inferential models, segmentation analysis, time series forecasting, synthetic control experiments, etc. The intent is to scientifically piece together a forensic view of what the data is really saying.

This data-driven insight is central to providing strategic guidance. In this sense, data scientists act as consultants, guiding business stakeholders on how to act on findings.

Data science – development of data product

A "data product" is a technical asset that: (1) utilizes data as input, and (2) processes that data to return algorithmically-generated results. The classic example of a data product is a recommendation engine, which ingests user data, and makes personalized recommendations based on that data. Here are some examples of data products:

- Amazon's recommendation engines suggest items for you to buy, determined by their algorithms. Netflix recommends movies to you. Spotify recommends music to you.
- Gmail's spam filter is data product – an algorithm behind the scenes processes incoming mail and determines if a message is junk or not.
- Computer vision used for self-driving cars is also data product – machine learning algorithms are able to recognize traffic lights, other cars on the road, pedestrians, etc.

This is different from the "data insights" section above, where the outcome to that is to perhaps provide advice to an executive to make a smarter business decision. In contrast, a data product is technical functionality that encapsulates an algorithm, and is designed to integrate directly into core applications. Respective examples of applications that incorporate data product behind the scenes: Amazon's homepage, Gmail's inbox, and autonomous driving software.

Data scientists play a central role in developing data product. This involves building out algorithms, as well as testing, refinement, and technical deployment into production systems. In this sense, data scientists serve as technical developers, building assets that can be leveraged at wide scale.

Data science is a blend of skills in three major areas of mathematics expertise, hacking skills using technology and business/strategy acumen.

Mathematics Expertise

At the heart of mining data insight and building data product is the ability to view the data through a quantitative lens. There are textures, dimensions, and correlations in data that can be expressed mathematically. Finding solutions utilizing data becomes a brain teaser of heuristics and quantitative technique. Solutions to many business problems involve building analytic models grounded in the hard math, where being able to understand the underlying mechanics of those models is key to success in building them.

Also, a misconception is that data science all about statistics. While statistics is important, it is not the only type of math utilized. First, there are two branches of statistics – classical statistics and Bayesian statistics. When most people refer to *stats* they are generally referring to *classical stats*, but knowledge of both types is helpful. Furthermore, many inferential techniques and machine learning algorithms lean on knowledge of linear algebra. For example, a popular method to discover hidden characteristics in a data set is SVD, which is grounded in matrix math and has much less to do with classical stats. Overall, it is helpful for data scientists to have breadth and depth in their knowledge of mathematics.

Technology and Hacking

First, let's clarify on that we are *not* talking about hacking as in breaking into computers. We're referring to the tech programmer subculture meaning of hacking – i.e., creativity and ingenuity in using technical skills to build things and find clever solutions to problems.

Why is hacking ability important? Because data scientists utilize *technology* in order to wrangle enormous data sets and work with complex algorithms, and it requires tools far more sophisticated than Excel. Data scientists need to be able to code — prototype quick solutions, as well as integrate with complex data systems. Core languages associated with data science include SQL, Python, R, and SAS. On the periphery are Java, Scala, Julia, and others. But it is not just knowing language fundamentals. A hacker is a technical ninja, able to creatively navigate their way through technical challenges in order to make their code work.

Along these lines, a data science hacker is a solid algorithmic thinker, having the ability to break down messy problems and recompose them in ways that are solvable. This is critical because data scientists operate within a lot of algorithmic complexity. They need to have a strong mental comprehension of high-dimensional data and tricky data control flows. Full clarity on how all the pieces come together to form a cohesive solution.

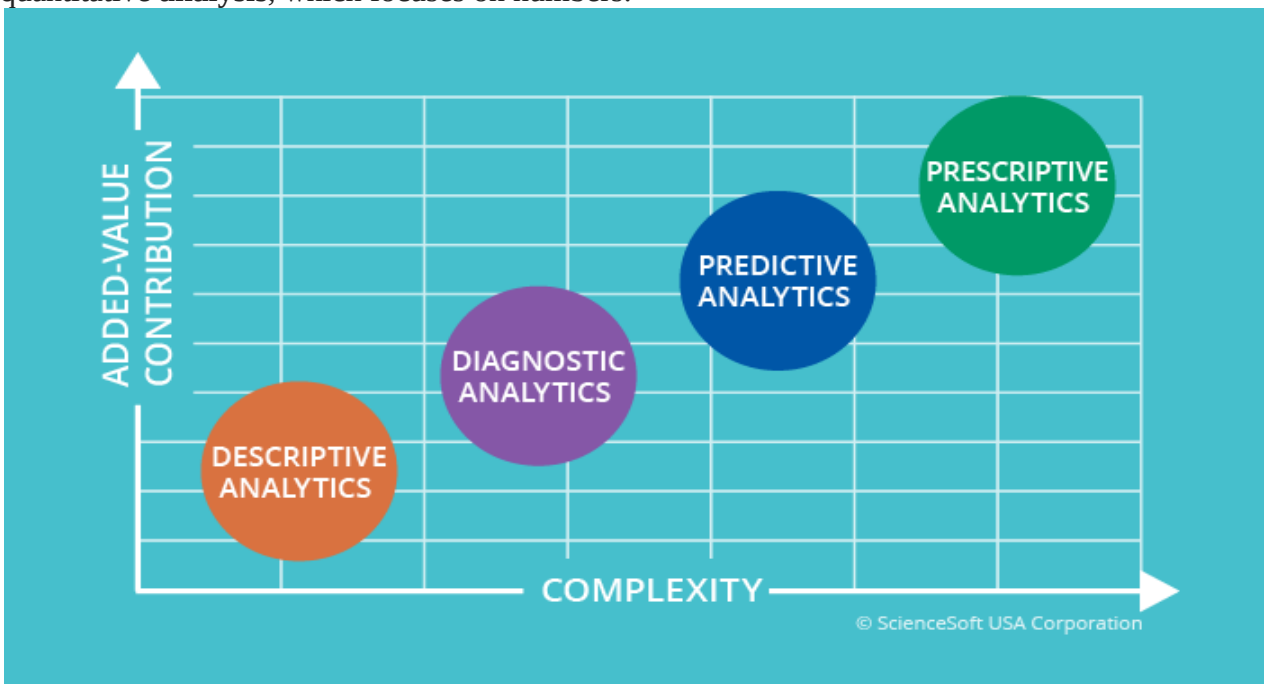
Strong Business Acumen

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established under section 3 of UGC Act 1956)
Pollachi main road, Eachanari post, Coimbatore -21
Class: III B.Sc(CT)
Code :16CTU503A Introduction to Data Science (2016-2019)

It is important for a data scientist to be a **tactical business consultant**. Working so closely with data, data scientists are positioned to learn from data in ways no one else can. That creates the responsibility to translate observations to shared knowledge, and contribute to strategy on how to solve core business problems. This means a core competency of data science is using data to cogently tell a story. No data-puking – rather, present a cohesive narrative of problem and solution, using data insights as supporting pillars, that lead to guidance.

Having this business acumen is just as important as having acumen for technology and algorithms. There needs to be clear alignment between data science projects and business goals. Ultimately, the value doesn't come from data, math, and tech itself. It comes from leveraging all of the above to build valuable capabilities and have strong business influence.

Categories of Analysis:
There are many **types of data analysis**. Some of them are more basic in nature, such as descriptive, exploratory, inferential, predictive, and causal. Some, however, are more specific, such as qualitative **analysis**, which looks for things like patterns and colors, and quantitative **analysis**, which focuses on numbers.



Descriptive Analysis - Descriptive analytics answers the question of *what happened.* For instance, a healthcare provider will learn how many patients were hospitalized last month; a retailer – the average weekly sales volume; a manufacturer – a rate of the products returned for a

past month, etc. Let us also bring an example from our BI consulting practice: a manufacturer was able to decide on focus product categories based on the analysis of revenue, monthly revenue per product group, income by product group, total quality of metal parts produced per month.
Descriptive analytics juggles raw data from multiple data sources to give valuable insights into the past. However, these findings simply signal that something is wrong or right, without explaining why. For this reason, highly data-driven companies do not content themselves with descriptive analytics only, and prefer combining it with other types of data analytics.

Diagnostic Analysis - At this stage, historical data can be measured against other data to answer the question of *why something happened*. Thanks to diagnostic analytics, there is a possibility to drill down, to find out dependencies and to identify patterns. Companies go for diagnostic analytics, as it gives a deep insight into a particular problem. At the same time, a company should have detailed information at their disposal, otherwise data collection may turn out to be individual for every issue and time-consuming.
Let's take another look at the examples from different industries: a healthcare provider compares patients' response to a promotional campaign in different regions; a retailer drills the sales down to subcategories. Another flashback to our BI projects: in the healthcare industry, customer segmentation coupled with several filters applied (like diagnoses and prescribed medications) allowed measuring the risk of hospitalization.

Predictive Analysis - Predictive analytics tells *what is likely to happen.* It uses the findings of descriptive and diagnostic analytics to detect tendencies, clusters and exceptions, and to predict future trends, which makes it a valuable tool for forecasting. Despite numerous advantages that predictive analytics brings, it is essential to understand that forecasting is just an estimate, the accuracy of which highly depends on data quality and stability of the situation, so it requires a careful treatment and continuous optimization.
Thanks to predictive analytics and the proactive approach it enables, a telecom company, for instance, can identify the subscribers who are most likely to reduce their spend, and trigger targeted marketing activities to remediate; a management team can weigh the risks of investing in their company's expansion based on cash flow analysis and forecasting. One of our case studies describes how advanced data analytics services allowed a leading FMCG company to predict what they could expect after changing brand positioning.
Prescriptive Analysis - The purpose of prescriptive analytics is to literally prescribe *what action to take* to eliminate a future problem or take full advantage of a promising trend. An example of prescriptive analytics from our project portfolio: a multinational company was able to identify opportunities for repeat purchases based on customer analytics and sales history.
This state-of-the-art type of data analytics requires not only historical data, but also external information due to the nature of statistical algorithms. Besides, prescriptive analytics uses sophisticated tools and technologies, like machine learning, business rules and algorithms, which makes it sophisticated to implement and manage. That is why, before deciding to adopt

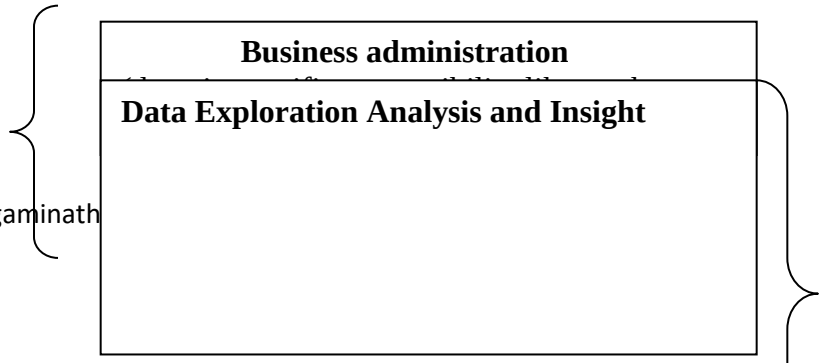
prescriptive analytics, a company should compare required efforts vs. an expected added value.With various types of analytics, companies are free to choose how deep they need to dive in data analysis to satisfy their business needs best. While descriptive and diagnostic analytics offers a reactive approach, predictive and prescriptive analytics makes users proactive approach

What is Analytics?

Analytics has risen quickly in popular business lingo over the past several years; the term is used loosely, but generally meant to describe critical thinking that is quantitative in nature. Technically, analytics is the "science of analysis" — put another way, the practice of analyzing information to make decisions. Is "analytics" the same thing as data science? Depends on context. Sometimes it is synonymous with the definition of data science that we have described, and sometimes it represents something else. A data scientist using raw data to build a predictive algorithm falls into the scope of analytics. At the same time, a non-technical business user interpreting pre-built dashboard reports (e.g. GA) is also in the realm of analytics, but does not cross into the skill set needed in data science. Analytics has come to have fairly broad meaning. At the end of the day, as long as you understand beyond the buzzword level, the exact semantics don't matter much.

What is the difference between an analyst and a data scientist?

"Analyst" is somewhat of an ambiguous job title that can represent many different types of roles (data analyst, marketing analyst, operations analyst, financial analyst, etc). What does this mean in comparison to data scientist?
Data Scientist: Specialty role with abilities in math, technology, and business acumen. Data scientists work at the raw database level to derive insights and build data product.
Analyst: This can mean a lot of things. Common thread is that analysts look at data to try to gain insights. Analysts may interact with data at both the database level or the summarized report level.
Thus, "analyst" and "data scientist" is not exactly synonymous, but also not mutually exclusive. Here is our interpretation of how these job titles map to skills and scope of responsibilities:



Analyst

Advanced Algorithms Machine Learning

Data

Data Product Engineering

Scientist

What is Machine Learning?

Machine learning is a term closely associated with data science. It refers to a broad class of methods that revolve around data modeling to (1) algorithmically make predictions, and (2) algorithmically decipher patterns in data.

Machine learning for making predictions — Core concept is to use tagged data to train predictive models. Tagged data means observations where ground truth is already known. Training models means automatically characterizing tagged data in ways to predict tags for unknown data points. E.g. a credit card fraud detection model can be trained using a historical record of tagged fraud purchases. The resultant model estimates the likelihood that any new purchase is fraudulent. Common methods for training models range from basic regressions to complex neural nets. All follow the same paradigm known as supervised learning.

Machine learning for pattern discovery — Another modeling paradigm known as unsupervised learning tries to surface underlying patterns and associations in data when no existing ground truth is known (i.e. no observations are tagged). Within this broad category of methods, the most commonly used are clustering techniques, which algorithmically detect what are the natural groupings that exist in a data set. For example, clustering can be used to programmatically learn the natural customer segments in a company's user base. Other unsupervised methods for mining underlying characteristics include: principal component analysis, hidden markov models, topic models, and more.

Not all machine learning methods fit neatly into the above two categories. For example, collaborative filtering is a type of recommendations algorithm with elements related to both supervised and unsupervised learning. Contextual bandits are a twist on supervised learning where predictions get adaptively modified on-the-fly using live feedback.

This wide-ranging breadth of machine learning techniques comprise an important part of the data science toolbox. It is up to the data scientist to figure out which tool to use in different circumstances (as well as how to use the tool correctly) in order to solve analytically open-ended problems

Turning data in to actionable knowledge:

The amount of data produced and communicated over the Internet and the Web is rapidly increasing. Everyday around 20 quintillion (10^18) bytes of data are produced.(http://www01.ibm.com/software/data/bigdata/). This data includes textual content (unstructured, semi-structured, structured) to multimedia content (images, video and audio), on a variety of platforms (enterprise, social media, and sensors). One of the fastest growing types of data relates to physical observations, measurements and occurrences in the real world. The growth of physical world data collection and communication is supported by low cost sensor devices such as wireless sensor nodes that can be deployed in different environments, smart phones and other network-enabled appliances. This trend will only accelerate, as it is estimated that by 2020 more than 50 billion devices will be connected to the Internet.

Extending the current Internet and providing connections and communication between physical objects and devices, or "Things" is described under the general term of Internet of Things (IoT). Another often used term is, Internet of Everything (IOE), which recognises the key role of people or citizen sensing, such as through social media, to complement physical sensing implied by IoT. Integrating the real world data into the Web and providing Web-based interactions with the IoT resources is also often discussed under umbrella term of "Web of Things" (WoT). Data collected by different sensors and devices have various types (e.g. temperature, light, sound, and video) and are inherently diverse (the quality and validity of data can vary with different devices through time; data is also mostly location and time dependent) [10]. WoT resources can be ubiquitous and are often constrained in terms of power, memory, processing and communication capabilities. The heterogeneity, ubiquity and dynamic nature of the resources and devices, and the wide range of data, make discovering, accessing, processing, integrating and interpreting the physical world data on the Web a challenging task.
The WoT data, however, is not limited to only sensor device data. Web resident data and knowledge (e.g., Wikipedia, Linked Open Data) and information exchanged over social media and user submitted physical world observations and measurements make up a rich cyber component. Integration of physical, cyber, and social resources enables developing applications and services that can incorporate situation and context-awareness into the decision making mechanisms and can create smarter applications and enhanced services.

WoT data is a type of Big Data that is not only large in scale and volume, but also continuous, with rich spatiotemporal dependency. The resources that produce the data often operate in

dynamic and volatile environments or can collect and communicate the data on an ad-hoc basis. The dynamicity of the environment and data providers make efficient utilisation of the WoT data on a global scale a challenging task. Figure 1 shows the access and process chain of the physical world data. The data is produced and collected using machine sensors, human sensing, smart phones and other devices. The data can be aggregated and summarised, or it can be processed and transformed to higher-level abstract descriptions of situations and events. The collected data – raw, and at times aggregated, and/or abstracted – is communicated over networks. The data can be published and stored temporarily or it can be added to repositories, and the publication interfaces can provide a metadata-enriched representation of the data. The query and discovery services can support search processes for finding the data in large-scale distributed environments. Data aggregation and summarisation can also occur at later stages by combining data from different sources and various types. The aggregation and summarisation highlight the role and importance of creating knowledge from raw data.

<<Data production and access chain>>

Sensing and data collection

The sensor devices, smart phones, social media and citizen sensing resources are some of the key sources for producing and collecting physical world data that can be communicated, integrated and accessed on the Web. These resources can produce large volumes of data in which the quality of the data can also vary over time. The data can be represented as numerical measurement values or as symbolic descriptions of occurrences in the physical world. Determining the quality, validity and trust of data are among the key issues in Big Data collections from the physical world, especially in use-case scenarios where the data is made available by a large number of different (and sometimes unknown) providers. As the physical world data can be related to the environment, people, and events, privacy and security are always key concerns. When the scale of the data and the number of different parties that can access and process the data increase, dealing with these issues becomes more challenging. For example, in a smart city environment where the sensory devices collect data related to citizen activities, and multiple agencies can access these data, ownership, duration of storage and types of use can raise significant privacy and security concerns.

Publication

Different data publishers have various ways of publishing and reporting data, and providing access to sensory data streams. Sensory data can be transient or it can be published and stored in repositories for long-term access and use. As the size and diversity of multimodal physical world data increases, publication and representation of the data in a way that makes discovery and

access more flexible and scalable becomes a challenge. In recent years there have been several efforts focusing on adding enriched metadata to enhance semantic interoperability and to provide machine-readable (and potentially machine interpretable) descriptions of sensory data– a notable example is the W3C Incubator Group on Semantic Sensor Networks [8], [4]. Several other models and semantic annotation frameworks have also been proposed for the physical world data publication and representation [2]. An important work in progress related to the representation and publication of heterogeneous physical world data includes how to automate semantic annotations, interpretation, mapping and mediation between different schema models, and efficiently balancing between express-ability and complexity of descriptions.

Query and Discovery

The query and discovery of physical world data is often based on type, time, location and the entity of interest. However, in large-scale dynamic and distributed environments, defining the region and location of requested data, indexing and querying of distributed data and/or services and data provider sources is not easy. Current solutions are very effective in processing and interpreting textual and audio-visual data. However, large scale distributed data streams that provide numerical location and time dependent data of varying quality related to physical world phenomena and discovery scenarios where the data is location and time dependent, and varies in quality, requires a different set of solutions. In principle, we still do not have fully matured search engines, similar to those on the Web, which can provide for query, indexing, discovery and resolving real-time numerical and descriptive sensory measurements and observations. In order to better manage the tasks of publishing, sharing, analysing and understanding streaming data, researchers are adapting and extending the Semantic Web technologies. In particular, there are several efforts towards the extension of SPARQL for streaming data processing of semantically annotated data [3][1]. By extending query languages to allow continuous queries over semantically annotated data, WoT applications will more easily integrate various streams of realworld data with background domain knowledge available on the Web as Linked Open Data.

Actionable knowledge

WoT data is not only voluminous; it is also continuous, streaming, real time, dynamic and volatile. Consequently, the Big Data analytics for distributed processing of large-scale data (e.g., Hadoop) and programming models that allow automatic parallelisation of the execution of tasks (e.g., MapReduce) [7] will not be effective or adequate. In addition, creating human understandable and/or machine-readable information from raw observation and measurement data

and providing real time processing and response mechanisms are also important. The distribution and efficient scalable processing of data in WoT, in addition to enhanced data publication and dissemination, will be dependent on effective mechanisms for in-network processing, aggregation and summarisation. Creating abstractions from data, or patterns of data, that can provide an aggregated view on the data will be useful. This will require using domain-specific background knowledge to extract meaningful information and actionable knowledge from the WoT data [6]. WoT resources are often dynamic; they can join a network but may later become unavailable due to network or power outage, or the source providing them can move and join a different sub-network. This will add to the challenges of discovery, integration and exploitation of data in conventional systems [9]. Indexing and discovery of resources will require a set of mechanisms that can support mobility and dynamicity in real time data and resource discovery, and can find the data by referring to their relations to objects and entities in the physical world. Unlike Internet search engines that rely on indexing existing data, the publication and integration of data in WoT cannot be separated from data discovery and search. Internet search engines discover available data, whereas in WoT data is not usually available at the time of query, and so discovery and search mechanisms would need to obtain it from suitable resources. The quality and form of resources is another major challenge. For example, during the Fukushima disaster, when people started publishing radiation data, different users provided a wide variety of inconsistent data for similar or nearby locations [5]. Inconsistency can be due to a number of factors such as errors in reading and reporting, the use of different and un-calibrated devices, or different processes of data collection. Discovery and search methods would therefore require learning, feedback and profiling mechanisms for quality-based data queries. In WoT, millions of devices and resources, including citizen sensors (humans reporting what they see or think using social media) participate in collecting and publishing data from the physical world. Going beyond device and resource connectivity on a large-scale, we will need data and semantic connectivity among resources and consumers for supporting the effective utilisation of the networks of the future. With the huge diversity and volumes of data expected in the near future, connectivity at the information level becomes more important than connectivity at the network level to facilitate the effective interpretation and extraction of knowledge (i.e., abstraction) from the WoT Big Data. Developing scalable and flexible analysis and processing models, and learning mechanisms, that can interpret large volumes of dynamic data of diverse quality requires coordination and collaboration between different methods and solutions. This includes different methods and solutions to preprocess the raw sensory data (e.g. aggregation, summarisation and filtering mechanisms), various metadata and annotation models and techniques (e.g. data representation frameworks and languages), data abstraction and pattern recognition methods, and semantic interpretation and online analytical

processing methods. This will provide a value chain for the raw sensory data from various sources to be processed, integrated and interpreted, thus transformed into actionable information, insight and knowledge that lead to improved decisions and human experience. Figure 2 demonstrates different steps that can be envisaged for efficient processing and making use of the WoT data.

Tools used in building analysis software

- Knime. KNIME **Analytics** Platform is the leading open solution **for data**-driven innovation, helping you discover the potential hidden in your **data**, mine **for**fresh insights, or predict new futures. ...
- OpenRefine. ...
- R-Programming. ...
- Orange. ...
- RapidMiner. ...
- Pentaho. ...
- Talend. ...
- Weka.
- Tableau
- Hadoop
- Python

Introduction to the Tools used in data analysis software

Hadoop: The name Hadoop has become synonymous with big data. Hadoop isan open-source big data analytics software framework for distributed storage of very large datasets on computer clusters. All that means you can scale your data up and down without having to worry about hardware failures. Hadoop provides massive amounts of storage for any kind of data, enormous processing power and the ability to handle virtually limitless concurrent tasks or jobs.

The Apache Hadoop software library is a big data framework. It allows distributed processing of large data sets across clusters of computers. It is designed to scale up from single servers to thousands of machines.

Hadoop is not for the data beginner. To truly harness the software'spower, you really need to know Java. It might be a commitment, but Hadoop is certainly worth the effort – since tons of other companies and technologies run off of it or integrate with it. Getting familiar with the Hadoop ecosystem can prove valuable in more ways than one.

HPCC

HPCC is a big data tool developed by LexisNexis Risk Solution. It delivers on a single platform, a single architecture and a single programming language for data processing.

Storm:

Storm is a free and open source big data computation system. It offers distributed real-time, fault-tolerant processing system. With real-time computation capabilities.

Qubole: Data is Autonomous Big data management platform. It is self-managed, self-optimizing tool which allows the data team to focus on business outcomes.

Cassandra: The Apache Cassandra database is widely used today to provide an effective management of large amounts of data.

Pentaho: Pentaho provides big data tools to extract, prepare and blend data. It offers visualizations and analytics that change the way to run any business. This Big data tool allows turning big data into big insights.

Flink: Apache Flink is an open-source stream processing Big data tool. It is distributed, high-performing, always-available, and accurate data streaming applications.

Rapid miner: Rapid Miner is an open source big data tool. It is used for data prep, machine learning, and model deployment. It offers a suite of products to build new data mining processes and setup predictive analysis.

SAS: "Statistical Analysis System") is a software suite developed by **SAS** Institute for advanced **analytics**, multivariate analyses, business intelligence, data management, and predictive **analytics**. **SAS** was developed at North Carolina State University from 1966 until 1976, when **SAS** Institute was incorporated.

Version control

Version Control For Data Science

Discover how to overcome the steep learning curve of version control for data science, while also taking into account best practices and recommendations.

Keeping track of changes that you or your collaborators make to data and software is a critical part of any project, whether it's research, data science, software engineering, ... Being able to reference or retrieve a specific version of the entire project aids in reproducibility for you leading up to publication, when responding to reviewer comments, and when providing supporting information for reviewers, editors, and readers.

The best tools for tracking changes are the version control systems that are used in software development, such as Git, Mercurial, and Subversion. They keep track of what was changed in a file when and by whom, and synchronize changes to a central server so that multiple contributors can manage changes to the same set of files.

While these tools make tracking changes easier, they can have a steep learning curve. To overcome this learning curve, there are two sets of recommendations: a systematic manual approach for managing changes and version control in its full glory, and you can use the first while working towards the second, or just jump in to version control.

Nevertheless, whatever recommendation you will end up choosing, there are some general best practices or recommendations that you best take into account:

1. **Back up (almost) everything created by a human being as soon as it is created.** This includes scripts and programs of all kinds, software packages that your project depends on, and documentation. A few exceptions to this rule are discussed below.

2. **Keep changes small.** Each change should not be so large as to make the change tracking irrelevant. For example, a single change such as Revise script file that adds or changes several hundred lines is likely too large, as it will not allow changes to different components of an analysis to be investigated separately. Similarly, changes should not be broken up into pieces that are too small. As a rule of thumb, a good size for a single change is a group of edits that you could imagine wanting to undo in one step at some point in the future.
3. **Share changes frequently.** Everyone working on the project should share and incorporate changes from others on a regular basis. Do not allow individual investigator's versions of the project repository to drift apart, as the effort required to merge differences goes up faster than the size of the difference. This is particularly important for the manual versioning procedure describe below, which does not provide any assistance for merging simultaneous, possibly conflicting, changes.
4. **Create, maintain, and use a checklist for saving and sharing changes to the project.** The list should include writing log messages that clearly explain any changes, the size and content of individual changes, style guidelines for code, updating to-do lists, and bans on committing half-done work or broken code.
5. **Store each project in a folder that is mirrored off the researcher's working machine** using a system such as Dropbox or a remote repository such as GitHub. Synchronize that folder at least daily. It may take a few minutes, but that time is repaid the moment a laptop is stolen or its hard drive fails.

Markdown : Markdown is a lightweight markup language with plain text formatting syntax. It is designed so that it can be converted to HTMLand many other formats using a tool by the same name.[8] Markdown is often used to format readme files, for writing messages in online discussion forums, and to create rich text using a plain text editor. As the initial description of Markdown

contained ambiguities and unanswered questions, many implementations and extensions of Markdown appeared over the years to answer these issues.

An Example:

Text Using mark down syntax

```
Heading
=======

## Sub-heading

Paragraphs are separated
by a blank line.

Two spaces at the end of a line
produces a line break.

Text attributes _italic_,
**bold**, `monospace`.

Horizontal rule:

---

Bullet list:

  * apples
  * oranges
  * pears

Numbered list:

  1. wash
  2. rinse
```

```
  3. repeat

A [link](http://example.com).

![Image](Image_icon.png)

> Markdown uses email-style > characters for
blockquoting.

Inline <abbr title="Hypertext Markup
Language">HTML</abbr> is supported.
```

Corresponding HTML produced by mark down syntax

```
Heading
=======

## Sub-heading

Paragraphs are separated
by a blank line.

Two spaces at the end of a line
produces a line break.

Text attributes _italic_,
**bold**, `monospace`.

Horizontal rule:

---
```

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established under section 3 of UGC Act 1956)
Pollachi main road, Eachanari post, Coimbatore -21
Class: III B.Sc(CT)
Code :16CTU503A Introduction to Data Science (2016-2019)

```
Bullet list:

  * apples
  * oranges
  * pears

Numbered list:

  1. wash
  2. rinse
  3. repeat

A [link](http://example.com).

![Image](Image_icon.png)

> Markdown uses email-style > characters for
blockquoting.

Inline <abbr title="Hypertext Markup
Language">HTML</abbr> is supported.
```

Text viewed in a Browser

Heading[edit]

Sub-heading[edit]

Paragraphs are separated by a blank line.

Two spaces at the end of a line
produces a line break.

Text attributes *italic*, **bold**, `monospace`.

Horizontal rule:

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established under section 3 of UGC Act 1956)
Pollachi main road, Eachanari post, Coimbatore -21
Class: III B.Sc(CT)
Code :16CTU503A Introduction to Data Science (2016-2019)

Bullet list:

- apples
- oranges
- pears

Numbered list:

1. wash
2. rinse
3. repeat

A link.

Git

Git :is a version control system for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for source code management in software development,[8] but it can be used to keep track of changes in any set of files. As a distributed revision control system it is aimed at speed,[9] data integrity,[10] and support for distributed, non-linear workflows.[11]

As with most other distributed version control systems, and unlike most client–server systems, every Git directory on every computer is a full-fledged repository with complete history and full version tracking abilities, independent of network access or a central server.

Version Control System (VCS) is a software that helps software developers to work together and maintain a complete history of their work.

Listed below are the functions of a VCS:

- Allows developers to work simultaneously.
- Does not allow overwriting each other's changes.
- Maintains a history of every version.

Following are the types of VCS:

- Centralized version control system (CVCS).
- Distributed/Decentralized version control system (DVCS).

we will concentrate only on distributed version control system and especially on Git. Git falls under distributed version control system.

Distributed Version Control System

Centralized version control system (CVCS) uses a central server to store all files and enables team collaboration. But the major drawback of CVCS is its single point of failure, i.e., failure of the central server. Unfortunately, if the central server goes down for an hour, then during that hour, no one can collaborate at all. And even in a worst case, if the disk of the central server gets corrupted and proper backup has not been taken, then you will lose the entire history of the project. Here, distributed version control system (DVCS) comes into picture.

DVCS clients not only check out the latest snapshot of the directory but they also fully mirror the repository. If the server goes down, then the repository from any client can be copied back to the server to restore it. Every checkout is a full backup of the repository. Git does not rely on the central server and that is why you can perform many operations when you are offline. You can commit changes, create branches, view logs, and perform other operations when you are offline. You require network connection only to publish your changes and take the latest changes.

Advantages of Git

Free and open source

Git is released under GPL's open source license. It is available freely over the internet. You can use Git to manage property projects without paying a single penny. As it is an open source, you can download its source code and also perform changes according to your requirements.

Fast and small

As most of the operations are performed locally, it gives a huge benefit in terms of speed. Git does not rely on the central server; that is why, there is no need to interact with the remote server for every operation. The core part of Git is written in C, which avoids runtime overheads associated with other high-level languages. Though Git mirrors entire repository, the size of the data on the client side is small. This illustrates the efficiency of Git at compressing and storing data on the client side.

Implicit backup

The chances of losing data are very rare when there are multiple copies of it. Data present on any client side mirrors the repository, hence it can be used in the event of a crash or disk corruption.

Security

Git uses a common cryptographic hash function called secure hash function (SHA1), to name and identify objects within its database. Every file and commit is check-summed and retrieved by its checksum at the time of checkout. It implies that, it is impossible to change file, date, and commit message and any other data from the Git database without knowing Git.

No need of powerful hardware

In case of CVCS, the central server needs to be powerful enough to serve requests of the entire team. For smaller teams, it is not an issue, but as the team size grows, the hardware limitations of the server can be a performance bottleneck. In case of DVCS, developers don't interact with the

server unless they need to push or pull changes. All the heavy lifting happens on the client side, so the server hardware can be very simple indeed.

Easier branching

CVCS uses cheap copy mechanism, If we create a new branch, it will copy all the codes to the new branch, so it is time-consuming and not efficient. Also, deletion and merging of branches in CVCS is complicated and time-consuming. But branch management with Git is very simple. It takes only a few seconds to create, delete, and merge branches.

DVCS Terminologies

Local Repository

Every VCS tool provides a private workplace as a working copy. Developers make changes in their private workplace and after commit, these changes become a part of the repository. Git takes it one step further by providing them a private copy of the whole repository. Users can perform many operations with this repository such as add file, remove file, rename file, move file, commit changes, and many more.

Working Directory and Staging Area or Index

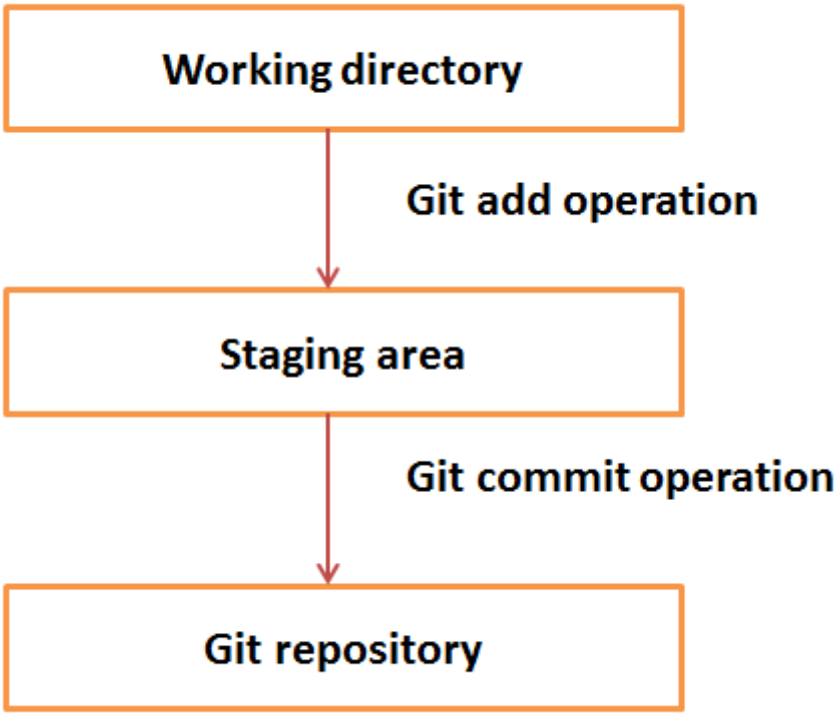
The working directory is the place where files are checked out. In other CVCS, developers generally make modifications and commit their changes directly to the repository. But Git uses a different strategy. Git doesn't track each and every modified file. Whenever you do commit an operation, Git looks for the files present in the staging area. Only those files present in the staging area are considered for commit and not all the modified files.

Let us see the basic workflow of Git.

Step 1 : You modify a file from the working directory.

Step 2 : You add these files to the staging area.

Step 3 : You perform commit operation that moves the files from the staging area. After push operation, it stores the changes permanently to the Git repository.



Suppose you modified two files, namely "sort.c" and "search.c" and you want two different commits for each operation. You can add one file in the staging area and do commit. After the first commit, repeat the same procedure for another file.

```
# First commit

[bash]$ git add sort.c
```

```
# adds file to the staging area

[bash]$ git commit –m “Added sort operation”


# Second commit

[bash]$ git add search.c


# adds file to the staging area

[bash]$ git commit –m “Added search operation”
```

Blobs

Blob stands for **B**inary **L**arge **Ob**ject. Each version of a file is represented by blob. A blob holds the file data but doesn't contain any metadata about the file. It is a binary file, and in Git database, it is named as SHA1 hash of that file. In Git, files are not addressed by names. Everything is content-addressed.

Trees

Tree is an object, which represents a directory. It holds blobs as well as other sub-directories. A tree is a binary file that stores references to blobs and trees which are also named as **SHA1** hash of the tree object.

Commits

Commit holds the current state of the repository. A commit is also named by **SHA1** hash. You can consider a commit object as a node of the linked list. Every commit object has a pointer to the parent commit object. From a given commit, you can traverse back by looking at the parent pointer to view the history of the commit. If a commit has multiple parent commits, then that particular commit has been created by merging two branches.

Branches

Branches are used to create another line of development. By default, Git has a master branch, which is same as trunk in Subversion. Usually, a branch is created to work on a new feature. Once the feature is completed, it is merged back with the master branch and we delete the branch. Every branch is referenced by HEAD, which points to the latest commit in the branch. Whenever you make a commit, HEAD is updated with the latest commit.

Tags

Tag assigns a meaningful name with a specific version in the repository. Tags are very similar to branches, but the difference is that tags are immutable. It means, tag is a branch, which nobody intends to modify. Once a tag is created for a particular commit, even if you create a new commit, it will not be updated. Usually, developers create tags for product releases.

Clone

Clone operation creates the instance of the repository. Clone operation not only checks out the working copy, but it also mirrors the complete repository. Users can perform many operations with this local repository. The only time networking gets involved is when the repository instances are being synchronized.

Pull

Pull operation copies the changes from a remote repository instance to a local one. The pull operation is used for synchronization between two repository instances. This is same as the update operation in Subversion.

Push

Push operation copies changes from a local repository instance to a remote one. This is used to store the changes permanently into the Git repository. This is same as the commit operation in Subversion.

HEAD

HEAD is a pointer, which always points to the latest commit in the branch. Whenever you make a commit, HEAD is updated with the latest commit. The heads of the branches are stored in **.git/refs/heads/** directory.

Revision

Revision represents the version of the source code. Revisions in Git are represented by commits. These commits are identified by **SHA1** secure hashes.

Git - Life Cycle

In this chapter, we will discuss the life cycle of Git. In later chapters, we will cover the Git commands for each operation.

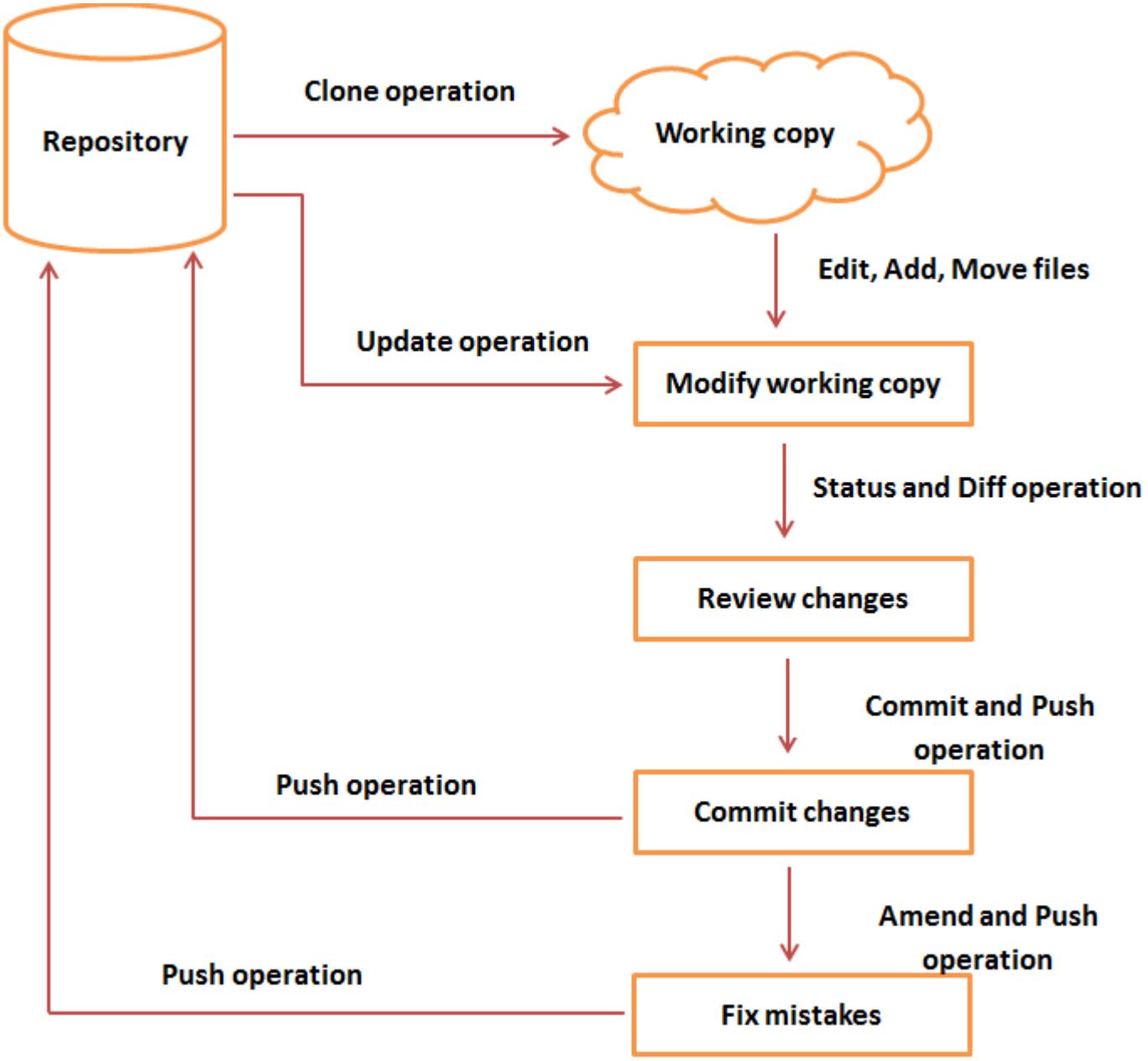
General workflow is as follows:

- You clone the Git repository as a working copy.
- You modify the working copy by adding/editing files.

- If necessary, you also update the working copy by taking other developer's changes.
- You review the changes before commit.
- You commit changes. If everything is fine, then you push the changes to the repository.
- After committing, if you realize something is wrong, then you correct the last commit and push the changes to the repository.

Shown below is the pictorial representation of the work-flow.

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established under section 3 of UGC Act 1956)
Pollachi main road, Eachanari post, Coimbatore -21
Class: III B.Sc(CT)
Code :16CTU503A Introduction to Data Science (2016-2019)



GitHub **GitHub** is a web-based hosting service for software development projects that use the Git revision control system. **GitHub** offers both paid plans for private repositories, and free accounts for open source projects.

What is the difference between Git and GitHub?

Git is a revision control system, a tool to manage your source code history. GitHub is a hosting service for Git repositories. So they are not the same thing: Git the tool, GitHub the service for projects that use Git.

R AND R STUDIO

In fact, they work together. **R** is a programming language for statistical calculation. And **RStudio** is an Integrated Development Environment (IDE) that helps you develop programs in **R**. ... You can use **R** without using **RStudio**, but you can't use**RStudio** without using **R**, so **R** comes first.

What is R?

R is a programming language and free software environment for statistical computing and graphics that is supported by the R Foundation for Statistical Computing.[6] The R language is widely used among statisticians and data miners for developing statistical software[7] and data analysis.[8] Polls, surveys of data miners, and studies of scholarly literature databases show that R's popularity has increased substantially in recent years.

What is R Studio?

RStudio is a free, open source IDE (integrated development environment) for R. (You must install R before you can install RStudio.) Its interface is organized so that the user can clearly view graphs, data tables, R code, and output all at the same time. It also offers an Import-Wizard-like feature that allows users to import CSV, Excel, SAS (*.sas7bdat), SPSS (*.sav), and Stata (*.dta) files into R without having to write the code to do so.

References

https://www.planet-data.eu/sites/default/files/presentations/Big_Data_Tutorial_part4.pdf

https://www.datameer.com/pdf/big-data-analytics-ebook.pdf?mkt_tok

https://guides.github.com/pdfs/markdown-cheatsheet-online.pdf

https://www.tutorialspoint.com/git/index.htm

https://libguides.library.kent.edu/statconsulting/r

questions

Big Data is defined as high-volume, high-velocity and/or high-variety information assets that demand ----------

Knowledge workers in any pyramid structure of organization termed as _____

Organizations are capturing, storing and analyzing data that has high volume, high velocity, and variety of _____

Business value from big data is great. However some companies have significant privacy concerns

1 TB of data is equivalent to----------

one million web pages are equivalent to----------

_____ is all about diving deep at a granular level to mine and understand complex behaviors, trends, and

_____ identifies major customer segments and unique shopping behaviors within the segment

A _____ is called as recommendation engine in data science

_____ used for self driving cars is also a data product able to sense traffic lights, pedestrians, and other

_____ is a software that helps software developers to work together and maintain a complete hist

_____ is a light weight markup language with plain text formatting syntax.

_____is sometimes called an epiphany, an "aha" moment or a "eureka' feeling when a solution

_____ is not for the data beginner

Rapid Miner is an _____ source big data tool

SAS stands for_____

SAS was developed by _____

VCS stands for_____

Centralized version control system (CVCS) uses a _____server to store all files and enables team collaboratic

The core part of Git is written in _____

BLOB stands for _____

_____ is a language for statistical calculation

_____ is a free source IDE

we cant use rstudio without _____programming language

_____ is a pointer, which always points to the latest commit in the branch.

_____ provides big data tools to extract, prepare and blend data.

_____database is widely used today to provide an effective management of large amounts of

_____ is an open source big data tool.

_____is a free and open source big data computation system.

_____delivers on a single platform, a single architecture and a single programming language fo

_____is designed to scale up from single servers to thousands of machines.

The name _____ has become synonymous with big data

KNIME _____ Platform is the leading open solution **for data**-driven innovation

The _____ of physical world data is often based on type, time, location and the entity of interest.

The data can be represented as numerical measurement values or as_____ descriptions of oc

The amount of data produced and communicated over the Internet and the Web is rapidly _____

_____ is a term closely associated with data science.

_____ means observations where ground truth is already known.

_____ means automatically characterizing tagged data in ways to predict tags for unknown data p

The _____ estimates the likelihood that any new purchase is fraudulent.

The purpose of _____ is to literally prescribe *what action to take* to eliminate a future problem or tak

_____uses the findings of descriptive and diagnostic analytics to detect tendencies, clusters and exceptions,

_____is a possibility to drill down, to find out dependencies and to identify patterns.

_____answers the question of *what happened.*

_____is a blend of skills in three major areas of mathematics expertise, hacking skills

using technology and business/strategy acumen

_____ play a central role in developing data product.

_____data mines movie viewing patterns to understand what drives user interest, and uses that

_____identifies what are major customer segments within it's base and the unique shopping behavi

_____ utilizes time series models to more clearly understand future demand, which help plan for production levels more optimally

Modulo division in R is computed using the symbol_____

_____ is a software that helps software developers to work together and maintain a complete histor

_____ identifies major customer segments and unique shopping behaviors within the segment.

_____is a version control system for tracking changes in computer files and coordinating work on those file

A _____ is called as recommendation engine in data science

Machine readable file formats for data processing are

A web scraper can download all web pages except _____

_____ is a way for servers to communicate requesting specific resource such as documents, images or video.

Organizations are capturing, storing and analyzing data that has high volume, high velocity, and variety of _____

_____ used for self driving cars is also a data product able to sense traffic lights, pedestrians, and other c

Extending current internet and providing connection and communication between physical objects and devices c

Which command in R used to search for help pages containing word “plotting” _____-

a	b
quality	**cost-effective**
MIS	ESS
information	knowledge
hindustan lever and protector and gamble	**google and facebook**
1000pb	1024mb
1 lakhs	100 lakhs
big data	DSS
neflix	**target**
data insight	**data product**
netflix	big data
data insight	**version control system**
knitr	**mark down**
insight	big data
hadoop	data insight
open	closed
statistical analysis system	statistical analytics system
cambridge university	**north carolina state university**
version control system	various control system
central	main
c	c++
binary large optimizing	**binary large objects**
R	C
visual studio	cinfig studio
c	R
SHAI	HEAD
cassandra	**Pentaho**
pentaho	flink
pentaho	cassandra
rapid miner	flink
cassandra	rapid miner
SHAI	pentaho
pentaho	rapid miner
refering	interface
data refining	actionable knowledge
logical	**symbolic**
decreasing	low
data learning	query
training models	**tagged data**
resultant model	tagged data
resultant model	training models
predictive analytics	**prescriptive analytics**
predictive analytics	prescriptive analytics
predictive analytics	prescriptive analytics
descriptive analysis	diagnostic analysis
big data	Data science

data scientists	programmers
target	**netflix**
target	netflix
proctor and gamble	target
%	%%
data insight	**version control system**
netflix	**target**
git hub	git
data insight	data product
csv &json	HTML and PDF
wikipedia	CAPTCHA codes
FTP	knit r
information	knowledge
netflix	big data
WOT	IOT
?”plotting”	**??”plotting”**

c	d
controlability	accuracy
OLAP	**DSS**
data	business value
amazon and verizone	none of the above
1024yb	**none of the above**
10 lakhs	1 crore
data science	business intelligence
spotify	none of the above
google	none of the above
computer vision	none of the above
pentaho	rapid miner
r language	none of the above
target	data
r language	DSS
both a& b	none of the above
statistical analyzing system	statistical accurate system
american university	none of the above
version command system	various connecting system
open	all of the bove
java	html
binary large object	basic large objects
c++	python
rstudio	all of the bove
both a&b	none of these
TAIL	TARGET
flink	rapid miner
rapid miner	**Apache Cassandra**
Rapid Miner	flink
apache cassandra	**Storm**
HPCC	Storm
apache cassandra	**apache hadoop**
Hadoop	open refine
analyzing	**Analytics**
query and discovery	open refine
alphabetical	none of the above
increasing.	high
Machine learning	none of the above
resultant model	none of the above
Training models	all of the bove
either a or b	both a &b
predictive analysis	prescriptive analysis
predictive analysis	prescriptive analysis
Diagnostic Analysis	none of the above
either a or b	none of the above
both a &b	none of the above

both a &b	none of the above
data miner	all of above
data miner	data analysist
netflix	b alone
%/%	%&%
pentaho	rapid miner
spotify	spam filter
markdown	knitr
google	none of the above
XML&excel files	**a &c**
pay walls	**all the above**
HTML	json
data	business value
computer vision	none of the above
knowledge	none of the above
??plotting	?plotting

ot

Class: III B.Sc(CT)
Subject Code:16CTU503A Introduction to Data Science Batch(2016-2019)

Unit II : R programming basics: Overview of R, R datatypes and Objects, reading and writing data, control structures, functions, scoping rules, dates and times, loop functions, debugging tools, simulation, code profiling

R Programming Basics:

Introducing R: R is a programming language and R refers to the software that is used to run R program. Ross Ihaka and Robert Gentleman from University of Auckland, created R language in 1990 s. R language is based on S language.

R is a free open source that has cross platform compatibility. R is a most advanced statistical programming tool which produces outstanding graphical outputs. R is flexible and comprehensive for beginners. R easily relates to other languages such as C,C++, Java, Python, Hadoop etc..R can handle huge data in flat files even in semi structured or in structured form.

Initiating R

Open R GUI, find the command prompt and type command below and hit enter

>sum(1:5)

[1] 15

In the above command sum() is a function that takes argument 1:5 which means a vector consists of a sequence of integers from 1 to 5. R allows to move arrow key to move upward to see previous results.

Help in R

There are many ways to get help from R. If a function name or a dataset name is known then we can type ? followed by name. If name is not known, need to give ?? followed by a term related to the search.

Keywords, special characters and two separate terms of search need to be enclosed in double or single quotes. The symbol # is comment statement.

>?mean #help page for mean function

>?"+" #help page for addition function

>?"if" #help page for if opens

>??plotting #searches for help pages containing word "plotting"

>>??"regression model" #searches for "regression model" phrase

Alternate ways of getting help

>help("mean"

>help("+")

>help("if)

>help.search("plotting")

>help.search("regression model")

Assigning variables

The results of an operation can be stored for reuse. The values can be assigned either using "=" or "<-"

There is no need for variable declaration. The variable type is automatically assumed based on the value given.

>X<-1:3

>X

[1] 1 2 3

>Y=4:6

[1] 4 5 6 7

>X+3*Y-2

[1] 11 15 19

Variable name consists of letters, numbers,dots and underscores, but a name should only start with an alphabet.

To create global variables we use "<< -"

For global assignment assign() function is used by including an extra attribute globalenv().

>assign("F",3*8)

>assign("G",6*9,globalenv())

>F

[1] 24

>printf(G)

[1] 54

Overview of R

Basic Mathematical Operations

The "+"plus operator is used to add numbers or add two vectors. Vector represents an ordered set of values.vectors are mainly used to analyze statistical data. Colon operator creates a sequence.

>7:12 + 12:17

[1] 19 21 23 24 2527 29

>c(3,1,8,6,7) + c(9,2,5,5,7,1)

[1] 12 3 13 13 8

The vectors and c() function in R helps us to avoid loops. The statistical function in R can take vector arguments and produces results. Sum() function takes vector arguments and produce results.

>sum(7:10)

[1] 34

>mean(7:10)

[1] 8.5

>median(7:10)

[1] 8.5

Similarly subtraction and multiplication operation is done as follows

>c(5,6,1,9) -2

[1] 3 4 -1 7

>c(5,6,1,9) –c(4,2,0,7)

[1] 1 4 1 2

The exponent operator is "^" or "**"

>identical(2^3,2**3)

[1] TRUE

The division operator is of three types

Ordinary division is "/"

Integer division is "%/%"

Modulo division is "%%" symbol.

>5:9/2

[1] 2.5 3.0 3.5.0.5

>5,9%/%2

[1] 2 3 3

>5:9%%2

[1] 1 0 1 0 1

The other mathematical functions are the trigonometry functions like sin(), cos(), tan(), asin(), acos(), atan(). Logarithmic and exponential functions log(), exp(),loglp(), expm1().

Operator "==" used to compare values. For checking inequalities "!=" is used. All relational operators are applicable.

>c("Week","WEEK","week","weak"=="week"

[1] FALSE FALSE TRUE FALSE

Packages in R

R packages are installed in an online repository called CRAN(Comprehensive R Archive Network). A package is a collection of R functions and datasets. Currently there are 10756 available packages.The list of all available packages in the CRAN repository can be viewed from the website https://cran.r-project.org/web/packages/available_packages_by_name.html To find the list of functions in a package (say "stats") then type ls("package:stats") or the command library(help=stats) in the command prompt.We can list and see all the packages which are already loaded using search() function.

>search()

[1] "GlobalEnv" "package.cluster" "tools:rstudio""package.stats"

[5] "package.graphics "packages:grDevices "package.utils" "package.datsets"

[9] "package.methods" "Autoloads" "package:base"

The function R.home("library" retrieves the location on the machine that stores all R default packages.Alternate command to do the same using .Library at prompt.

Assignment and printing in one line:

>L< - sum(4:8);L

[1] 30

OR

>(M< -sum(5:9)

[1] 35

R Datatypes and Objects:

In contrast to other programming languages like C and java, R variables are not declared as a datatype. The variables are assigned with R-Objects and the data type of R-Objects becomes the data type of variables.There are many types of R Objects viz – Vectors, Arrays, Matrices, Lists, Data Frames, Strings and Factors.

- Numeric
- Integer
- Complex
- Logical
- Character

Numeric : Decimal values are called numeric in R. It is the default computational data type.

```
>x=10.5

>x

[1] 10.5

>class(x)       #print the classname of x
```

An integer value is confirmed by using is.integer() function.Example

```
>k=1

>k

[1] 1

>class(k)

[1] "numeric"

>is.integer(k)

[1] FALSE
```

Integer

In order to create an integer variable in R, the as.integer(0 function is invoked as below.

```
>y=as.integer(3)
```

```
>y

[1] 3

>class(y)

[1] "integer"

>is.integer(y)

[1] TRUE
```

Force a numeric value in to an integer with as.integer() function.

```
>as.integer(3.14)

[1] 3

>as.integer("abc")

[1] NA

>as.integer(TRUE)

[1] 1

>as.integer(FALSE)

[1] 0
```

Complex

A complex number is expressed as an imaginary value i

```
>z=3+4i

>z

[1] 3+4i

>class(z)
```

[1] “complex”

>sqrt(-1)

[1] NaN #warning message

>sqrt(as.complex(-1))

[1] 0+1i

Logical

When two variables are compared, logical values are created. The logical operators are and, “|”, and negation “!”

>a=4;b=7

>p=a>b

>p

[1] FALSE

>class(p)

[1] “logical”

>a=TRUE;b=FALSE

>a AND OP b

[1] FALSE

>a | b

[1] TRUE

> !a

[1] FALSE

Character

The character object is used to represent string values in R. Objects can be converted in to character values using as.character() function. A paste() function is used to concatenate two character values.

>s=as.character("8.78")

>s

[1] "8,78"

>fname="kirtilal"

>lname="kalidas"

>paste(fname,lname)

[1] "kirtilal kalidas"

Vectors

A sequence of data elements of the same type is called as Vector. Members in a vector are called as components or members. The vector() function creates a vector of a specified type and length. The result is zero or FALSE or empty string.

>vector("numeric",3)

[1] 0 0 0

>vector("logical",5)

[1] FALSE FALSE FALSE FALSE FALSE

>vector("character",2)

[1] "" ""

Seq()

>seq(1:5)

[1] 1 2 3 4 5

>seq.int(5,12)

[1] 5 6 7 8 9 10 11 12

>seq.int(10,5,-1.5)

[1] 10.0 8.5 7.0 5.5

>length(1:7)

[1] 7

>length(c("aa","ccc","eeee"))

[1] 3

rep() function creates of a vector with repeated elements.

>rep(1:3,4)

[1] 1 2 3 1 2 3 1 2 3 1 2 3 9999999

Accessing the keyboard and monitor

In R, there are a series of functions that can be used to request an input from the user, including **readline(), cat(), and scan()**. But, I find the readline() function to be the optimal function for this task.

a) Reading from the keyboard

To read the data from the keyboard we use three different functions: scan(), readline(), print().

i) scan() –

Read Data Values: Read data into a vector or list from the console or file.

Keywords: File, connection

For Example:

```
> z <-scan()
1: 125
3: 2
4:
```

```
Read 3 items
[php]> z
[1]1252
```

ii) readline()

Read Text Lines from a Connection: Read some or all text lines from a connection.

Keywords: File, connection

We can use readline() for inputing a line from the keyboard in the form of a string:

For Example:

```
> w <-readline()
xyz vw u
> w
[1]"xyz vw u"
```

iii) print()

Print Values: **print**, prints its argument and returns it. It is a generic function which means that new printing methods can be easily added for new classes.

Keywords: print

printing to the screen: In interactive mode, one can print the value of that variable by just typing the variable name or expression. In batch mode, one can use the print() function, e.g.

print(x)

The argument might be an object. So it is a little better to use cat() instead of print(), as the last one can print only one expression and its result is numbered, which may be a nuisance to us. Here is an example written below:

```
>print("xyz")
[1]"xyz"
>cat("xyz \n")
xyz
```

The arguments to cat() will be printed out with intervening spaces, for instance

```
> g <-62
>cat(g,"xyz","gs\n")
```

```
3. 62 xyz gs
```

Reading and Writing files

There are so many methods to read and write files in R programming:

a) Reading a data or matrix from a file

Usually we use function read.table() to read data. The default value of a header is 'FALSE' and hence when we do not have a header, we need not say such. **R factors**are also called as character strings. For turning this "feature" off, you can include the argument as.is=T in your call to read.table().

When you have a spreadsheet export file, i.e. having a type.csv where the fields are divided by commas in place of spaces, use read.csv() in place of read.table(). To read spreadsheet files we can use read.xls.

When you read in a **matrix** using read.table(), the resultant object will become a **data frame**, even when all the entries got to be numeric. A case exists which may followup call towards as.matrix() in a matrix.

For example:

let say the matrix x becomes:

```
1. 101
2. 111
3. 110
4. 110
5. 001
```

We need to read it into a matrix form like this

```
> x <-matrix(scan("x"),nrow=5,byrow=T)
```

b) Reading a single File One Line at a Time

We can use readLines() for this, but we need to produce a connection first, by calling the file().

For Example:

```
1. > c <-file("z","r")
2. >readLines(c,n=1)
3. [1]"1 3"
```

c) Writing a Table to a File

In R, write.table() function is being used to work. It is same as read.table() and which writes a data frame instead of reading one. In the case of writing a matrix, to a file, we need not have to know row or column names like:

```
write.table(xc,"xcnew",row.names=F,col.names=F)
```

Control structures:

Control structures in R contains conditionals, loop statements like any other programming languages. Loops are very important and forms backbone to any programming languages.Before we get into the control structures in R, just type as below in Rstudio:

?control

If else statement:

#See the code syntax below for if else statement

```
if(x>1){
 print("x is greater than 1")
 }else{
  print("x is less than 1")
  }
```

#See the code below for nested if else statement

```
x=10
  x=10
 if(x>1 & x<7){
    print("x is between 1 and 7")}else if(x>8 & x< 15){
```

```
    print("x is between 8 and 15")
  }
```

[1] "x is between 8 and 15"

For loops:

As we know for loops are used for iterating items

```
#Below code shows for  loop implementation
x = c(1,2,3,4,5)
 for(i in 1:5){
   print(x[i])
 }
```

[1] 1

[1] 2

[1] 3

[1] 4

[1] 5

While loop :

```
#Below code shows while loop in R
x = 2.987
while(x <= 4.987) {
   x = x + 0.987
   print(c(x,x-2,x-1))
 }
```

```
[1] 3.974 1.974 2.974

[1] 4.961 2.961 3.961

[1] 5.948 3.948 4.948
```

Repeat Loop:

The repeat loop is an infinite loop and used in association with a break statement.

#Below code shows repeat loop:

```
a = 1

 repeat { print(a) a = a+1 if(a > 4) break }

[1] 1

[1] 2

[1] 3

[1] 4
```

Break statement:

A break statement is used in a loop to stop the iterations and flow the control outside of the loop.

#Below code shows break statement:

```
x = 1:10

 for (i in x){

    if (i == 2){

       break

    }

    print(i)
```

```
 }
```

[1] 1

Next statement:

Next statement enables to skip the current iteration of a loop without terminating it.

#Below code shows next statement

```
x = 1: 4
 for (i in x) {
    if (i == 2){
       next}
    print(i)
 }
```

[1] 1

[1] 3

[1] 4

Functions:

Creating a function in R:

function() is a built-in R function whose job is to create functions. In the below example function() takes one parameter x, executes a for loop logic.

The function object thus created using function() is assigned to a variable ('words.names'). Now this created function will be called using the variable 'word.names'

#Below code shows us, how a function is created in R:

Syntax:

```
function_name = function(parameters,..){ code}
```

```
words = c("R", "datascience", "machinelearning","algorithms","AI")

words.names = function(x) {

   for(name in x){

      print(name)

   }

}
```

Function Definition

An R function is created by using the keyword function. The basic syntax of an R function definition is as follows −

```
function_name <- function(arg_1, arg_2, ...) {

  Function body

}
```

Function Components:

The different parts of a function are −

Function Name − This is the actual name of the function. It is stored in R environment as an object with this name.

Arguments − An argument is a placeholder. When a function is invoked, you pass a value to the argument. Arguments are optional; that is, a function may contain no arguments. Also arguments can have default values.

Function Body − The function body contains a collection of statements that defines what the function does.

Return Value − The return value of a function is the last expression in the function body to be evaluated.

R has many in-built functions which can be directly called in the program without defining them first. We can also create and use our own functions referred as user defined functions.

Built-in Function

Simple examples of in-built functions are seq(), mean(), max(), sum(x) and paste(...) etc. They are directly called by user written programs. You can refer most widely used R functions.

```
# Create a sequence of numbers from 32 to 44.

print(seq(32,44))

# Find mean of numbers from 25 to 82.

print(mean(25:82))

# Find sum of numbers frm 41 to 68.

print(sum(41:68))
```

When we execute the above code, it produces the following result −

```
[1] 32 33 34 35 36 37 38 39 40 41 42 43 44

[1] 53.5

[1] 1526
```

User-defined Function

We can create user-defined functions in R. They are specific to what a user wants and once created they can be used like the built-in functions. Below is an example of how a function is created and used.

```
# Create a function to print squares of numbers in sequence.

new.function <- function(a) {
```

```
  for(i in 1:a) {
    b <- i^2
    print(b)
  }
}
```

Calling a Function

```
# Create a function to print squares of numbers in sequence.
new.function <- function(a) {
  for(i in 1:a) {
    b <- i^2
    print(b)
  }
}

# Call the function new.function supplying 6 as an argument.
new.function(6)
```

When we execute the above code, it produces the following result −

```
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
```

[1] 36

Calling a Function without an Argument

```
# Create a function without an argument.

new.function <- function() {
  for(i in 1:5) {
    print(i^2)
  }
}


# Call the function without supplying an argument.

new.function()
```

When we execute the above code, it produces the following result −

[1] 1

[1] 4

[1] 9

[1] 16

[1] 25

Calling a Function with Argument Values (by position and by name)

The arguments to a function call can be supplied in the same sequence as defined in the function or they can be supplied in a different sequence but assigned to the names of the arguments.

```
# Create a function with arguments.

new.function <- function(a,b,c) {
  result <- a * b + c
```

```
  print(result)
}


# Call the function by position of arguments.
new.function(5,3,11)
# Call the function by names of the arguments.
new.function(a = 11, b = 5, c = 3)
```

When we execute the above code, it produces the following result −

```
[1] 26
[1] 58
```

Calling a Function with Default Argument

We can define the value of the arguments in the function definition and call the function without supplying any argument to get the default result. But we can also call such functions by supplying new values of the argument and get non default result.

```
# Create a function with arguments.
new.function <- function(a = 3, b = 6) {
  result <- a * b
  print(result)
}
# Call the function without giving any argument.
new.function()
# Call the function with giving new values of the argument.
```

new.function(9,5)

When we execute the above code, it produces the following result −

[1] 18

[1] 45

Lazy Evaluation of Function

Arguments to functions are evaluated lazily, which means so they are evaluated only when needed by the function body.

```
# Create a function with arguments.

new.function <- function(a, b) {
  print(a^2)
  print(a)
  print(b)
}
```

We will see three groups of function in action

1.General function

2.Maths function

3.Statistical function

4.General functions

We are already familiar with general functions like cbind(), rbind(),range(),sort(),order() functions. Each of these functions has a specific task, takes arguments to return an output. Following are important functions one must know-

diff() function

If you work on time series, you need to stationary the series by taking their lag values. A stationary process allows constant mean, variance and autocorrelation over time. This mainly improves the prediction of a time series. It can be easily done with the function diff(). We can build a random time-series data with a trend and then use the function diff() to stationary the series. The diff() function accepts one argument, a vector, and return suitable lagged and iterated difference.

Note: We often need to create random data, but for learning and comparison we want the numbers to be identical across machines. To ensure we all generate the same data, we use the set.seed() function with arbitrary values of 123. The set.seed() function is generated through the process of pseudorandom number generator that make every modern computers to have the same sequence of numbers. If we don't use set.seed() function, we will all have different sequence of numbers.

```
set.seed(123)

## Create the data

x = rnorm(1000)

ts <- cumsum(x)

## Stationary the serie

diff_ts <- diff(ts)

par(mfrow=c(1,2))

## Plot the series

plot(ts, type='l')

plot(diff(ts), type='l')
```

length() function

In many cases, we want to know the length of a vector for computation or to be used in a for loop. The length() function counts the number of rows in vector x. The following codes import the cars dataset and return the number of rows.

Note: length() returns the number of elements in a vector. If the function is passed into a matrix or a data frame, the number of columns is returned.

```
dt <- cars

## number columns

length(dt)
```

Output:

```
## [1] 1

## number rows

length(dt[,1])
```

Output:

```
## [1] 50
```

Math functions

R has an array of mathematical functions.

Operator	Description
abs(x)	Takes the absolute value of x
log(x,base=y)	Takes the logarithm of x with base y; if base is not specified, returns the natural logarithm

exp(x) Returns the exponential of x

sqrt(x) Returns the square root of x

factorial(x) Returns the factorial of x (x!)

```
# sequence of number from 44 to 55 both including incremented by 1
x_vector <- seq(45,55, by = 1)
#logarithm
log(x_vector)
```

Output:

```
##  [1] 3.806662 3.828641 3.850148 3.871201 3.891820 3.912023 3.931826
##  [8] 3.951244 3.970292 3.988984 4.007333
#exponential
exp(x_vector)
#squared root
sqrt(x_vector)
```

Output:

```
##  [1] 6.708204 6.782330 6.855655 6.928203 7.000000 7.071068 7.141428
##  [8] 7.211103 7.280110 7.348469 7.416198
#factorial
factorial(x_vector)
```

Output:

```
##  [1] 1.196222e+56 5.502622e+57 2.586232e+59 1.241392e+61 6.082819e+62

##  [6] 3.041409e+64 1.551119e+66 8.065818e+67 4.274883e+69 2.308437e+71

## [11] 1.269640e+73
```

Statistical functions

R standard installation contains wide range of statistical functions. In this tutorial, we will briefly look at the most important function..

Basic statistic functions

Operator	Description
mean(x)	mean of x
median(x)	Median of x
var(x)	Variance of x
sd(x)	Standard deviation of x
scale(x)	Standard scores (z-scores) of x
quantile(x)	The quartiles of x
summary(x)	Summary of x: mean, min, max etc..

```
speed <- dt$speed

speed

# Mean speed of cars dataset

mean(speed)
```

Output:

```
## [1] 15.4

# Median speed of cars dataset

median(speed)
```

Output:

```
## [1] 15
```

Variance speed of cars dataset

```
var(speed)
```

Output:

```
## [1] 27.95918
```

Standard deviation speed of cars dataset

```
sd(speed)
```

Output:

```
## [1] 5.287644
```

Standardize vector speed of cars dataset

```
head(scale(speed), 5)
```

Output:

```
##          [,1]
## [1,] -2.155969
## [2,] -2.155969
## [3,] -1.588609
## [4,] -1.588609
## [5,] -1.399489
```

Quantile speed of cars dataset

```
quantile(speed)
```

Output:

```
##   0%  25%  50%  75% 100%
##    4   12   15   19   25
# Summary speed of cars dataset
summary(speed)
```

Output:

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     4.0    12.0    15.0    15.4    19.0    25.0
```

Up to this point, we have learned a lot of R built-in functions.

Note: Be careful with the class of the argument, i.e. numeric, Boolean or string. For instance, if we need to pass a string value, we need to enclose the string in quotation mark: "ABC" .

Write function in R

In some occasion, we need to write our own function because we have to accomplish a particular task and no ready made function exists. A user-defined function involves a name, arguments and a body.

```
function.name <- function(arguments)
{
   computations on the arguments
   some other code
}
```

Note: A good practice is to name a user-defined function different from a built-in function. It avoids confusion.

Environment Scoping

In R, the environment is a collection of objects like functions, variables, data frame, etc. R opens an environment each time Rstudio is prompted. The top-level environment available is the global environment, called R_GlobalEnv. And we have the local environment.

We can list the content of the current environment.

```
ls(environment())
```

Output

```
## [1] "diff_ts"        "dt"            "speed"         "square_function"

## [5] "ts"            "x"             "x_vector"
```

You can see all the variables and function created in the R_GlobalEnv.

Scoping rules:

Variable Scope

A variable is pairing of a name and a value.Variables can be created by assignment. The assignment x = 10 indicates that the value "10" and the name "x" are to be paired. R has both local and global variables. Global variables are created by assignments at top level. Local variables are created by assigning values within functions.

Example: Same Name, Different Variables

There are two "x" variables in the following code.

```
x = 10 # <-- global
fun =
function() {
x = 20 # <-- local to fun
}
```

Inside "fun," the value of "x" is "20."
Outside "fun," the value of "x" is"10."
Changes made to one "x" do not affect the other.

Visibility of Global Variables

Inside a function, the values of both local and global variables are visible.

```
> y = 20
> fun =
function() {
x = 30
x + y
}
> fun()
[1] 50
```

Here, "fun" combines the value of a local variable, "x," and the value of a global variable, "y."

Nested Functions

When R functions are nested, the variables of the outer functions are visible within the nested functions.

```
> outer =
function() {
inner =
function() {
y = 20 # y is local to inner
x + y
}
x = 10 # x is local to outer
inner()
}
> outer()
[1] 30
```

Scoping Rules

_ The scoping rules of a (computer) language govern how the value of variables can be determined.

_ The scope of a variable is the region of code where that variable has meaning.

_ In R, a local variable has meaning only within the function it is local to (including any functions that are
nested within that function).

_ When a value is sought for a given name, the local scope is inspected for a value, then any nesting scopes (i.e. functions) and finally a global value is sought.

Function Parameters, Arguments and Variables

_ Assignment is not the only way that variables are created.

_ The association of function's arguments with the function's formal parameters also creates variables.

_ In this case the variable name is the formal parameter name and the (initial) value of the variable is
corresponding function argument.

Example: Parameters and Arguments
_ In the function definition
fun =
function(a, b)
a^2 + b^2
the formal parameters of the function are “a” and “b.”
_ In the function call
fun(10, 20)
the arguments to the function are “10”and “20.”
_ The arguments and parameters are paired as variables.

A Simple Example
The following function adds a local variable “u” to a global variable “x.”
> add.x.to = function(u) x + u
> x = 10
> add.x.to(20)
[1] 30
> x = 20
> add.x.to(20)
[1] 40

More Complexity
Now let’s embed the “add.x.to” function inside another function.
> add =
function(x, y) {
add.x.to = function(u) x + u
add.x.to(y)
}
> add(1, 2)
[1] 3
Now “x” is no longer global. It is a variable that is local to the
function “add.”

Closures
_ The function returned by “make.add.to” is said to close over the value of “x.”

_ The function retains access to the variable "x" and can use its value whenever it is called.
_ Functions that do this are called closures.
_ Forming closures is one of R's most important capabilities.
_ It is also one of the least-understood and least-used.

Why Are Closures Useful?

_ In the last example, we could have just used a global variable "x" to hold the data values for the likelihood.
_ However, if we need several likelihood functions, they can't all keep their data in the same "x."
_ Closures provide a way for functions to have their own private copies of data.
_ This provides a very powerful programming tool.
_ It gets used a lot in other programming languages, but not so much in R.

Alternative Way of Creating Closures

_ There are a number of ways of creating closures that provide a more direct way of creating closures.
_ The most direct of these are based on the "with" and "local" functions.
_ They both work by directly providing a set of variables that can be closed over.
_ (The use of function calls to return closures can have some rather nasty side-effects.)

Creating Closures using "with"

_ The "with" function makes the elements of a named list available as variables to evaluate and expression.
_ If this expression creates a function, it closes over the variables made from the list.

```
> L = with(list(x = PoissonData),
function(lambda)
prod(dpois(x, lambda)))
```

_ This produces a closure that performs exactly like the earlier one created with "mk.pois.lk."

Creating Closures using "local"

_ Another direct way of creating closures is to establish local scope using the "local" function.
_ Again, a function returned from this local scope closes over the variables defined in that scope.

```
> L = local({
x = PoissonData
function(lambda)
prod(dpois(x, lambda))
})
```

_ This is the method I most commonly use for creating closures (although I do sometimes use the other
methods too.)

Handling date-times in R

Date-time variables are a pain to work with in any language. We'll discuss some of the common issues and how to overcome them. Before we examine the combination of dates and times, let's focus on dates. Even by themselves dates can be a pain. One idea is to refuse to use them. Generally dates are interally stored as integers, and dependingz on the operations you'll need, you could choose to do the same. As an added bonus, converting dates to integers may be useful in de-identifying your dataset.

One easy way to do this is by having a lookup table.

```
date.lookup <- format(seq(as.Date("2000-01-01"), as.Date("2010-12-31"), by = "1 day"))
match("2004-01-19", date.lookup)
## [1] 1480
date.lookup[1337]
## [1] "2003-08-29"
```

If that works for you, then that works for me and we're done here... let's assume you actually want dates though.

Date Class

The simplest data type to use for dates is the "Date" class. As mentioned, these will be internally stored as integers. In my example, day one was 2000-01-01. The speci_c date used to index your dates is called the origin. Typically programming languages use a default origin of 1970-01-01, though it is really day zero, not day one (negative values are perfectly valid).

```
# create a date
as.Date("2012-08-30")
## [1] "2012-08-30"
# specify the format
as.Date("08/30/2012", format = "%m/%d/%Y")
## [1] "2012-08-30"
# use a different origin, for instance importing values from Excel
as.Date(41149, origin = "1900-01-01")
## [1] "2012-08-30"
1
# take a difference
Sys.Date() - as.Date("1970-01-01")
## Time difference of 15582 days
# alternate method with specified units
difftime(Sys.Date(), as.Date("1970-01-01"), units = "days")
## Time difference of 15582 days
# see the internal integer representation
unclass(Sys.Date())
```

```
## [1] 15582
```

POSIXt Date-Time Class

Dates are pretty simple and most of the operations that we could use for them will also apply to date-time variables. There are at least three good options for date-time data types: built-in POSIXt, chron package, lubridate package. I rarely support using external packages (spoiler: I use POSIXt), if for no other reason than I don't like forcing dependencies on my code. But it can also be burdensome to write code that someone else has already written, so add-on packages are worth examining.

There are two POSIXt types, POSIXct and POSIXlt. "ct" can stand for calendar time, it stores the number of seconds since the origin. "lt", or local time, keeps the date as a list of time attributes (such as "hour" and "mon").

```
# current time as POSIXct
unclass(Sys.time())
## [1] 1.346e+09
# and as POSIXlt
unclass(as.POSIXlt(Sys.time()))
## $sec
## [1] 9.166
##
## $min
## [1] 54
##
## $hour
## [1] 13
##
## $mday
## [1] 30
##
## $mon
## [1] 7
##
## $year
## [1] 112
##
## $wday
## [1] 4
2
##
## $yday
## [1] 242
##
```

```
## $isdst
## [1] 1
##
## attr(,"tzone")
## [1] "" "CST" "CDT"
```

Format

Thus far we've made the bold assumption that we want our date-times in the format YYYY-MM-DD. It's not hard to set the format when creating or printing date-times. In fact it's not a bad idea to always set the
format so that you can be certain that your input/output looks like you expect. View the strftime help for an idea on how to specify format.

```
# create POSIXct variables
as.POSIXct("080406 10:11", format = "%y%m%d %H:%M")
## [1] "2008-04-06 10:11:00 CDT"
as.POSIXct("2008-04-06 10:11:01 PM", format = "%Y-%m-%d %I:%M:%S %p")
## [1] "2008-04-06 22:11:01 CDT"
as.POSIXct("08/04/06 22:11:00", format = "%m/%d/%y %H:%M:%S")
## [1] "2006-08-04 22:11:00 CDT"
# convert POSIXct variables to character strings
format(as.POSIXct("080406 10:11", format = "%y%m%d %H:%M"), "%m/%d/%Y %I:%M
%p")
## [1] "04/06/2008 10:11 AM"
as.character(as.POSIXct("080406 10:11", format = "%y%m%d %H:%M"), format = "%m-%d-
%y %H:%M")
## [1] "04-06-08 10:11"
```

Example 1

We've _nally covered the basics, let's run some practical examples.

```
# when do I turn 1 billion seconds old?
billbday <- function(bday, age = 10^9, format = "%Y-%m-%d %H:%M:%S") f
x <- as.POSIXct(bday, format = format) + age
togo <- round(difftime(x, Sys.time(), units = "days"))
if (to go > 0) f
msg <- sprintf("You will be %s seconds old on %s, which is %s days from now.", age,
format(x, "%Y-%m-%d"), togo)
g else f
msg <- sprintf("You turned %s seconds old on %s, which was %s days ago.", age, format(x,
"%Y-%m-%d"), -1 * togo)
g
3
if (age > 125 * 365.25 * 86400)
```

```
msg <- paste(msg, "Good luck with that.")
print(msg)
format(x, "%Y-%m-%d")
g
billbday("1981-04-13 15:00:00")
## [1] "You will be 1e+09 seconds old on 2012-12-20, which is 112 days from now."
## [1] "2012-12-20"
```

Data .frames Hate You

That was easy enough, let's use a data.frame.

```
dts <- data.frame(day = c("20081101", "20081101", "20081101", "20081101", "20081101",
"20081102",
"20081102", "20081102", "20081102", "20081103"), time = c("01:20:00", "06:00:00",
"12:20:00",
"17:30:00", "21:45:00", "01:15:00", "06:30:00", "12:50:00", "20:00:00", "01:05:00"), value =
c("5",
"5", "6", "6", "5", "5", "6", "7", "5", "5"))
dts1 <- paste(dts$day, dts$time)
dts2 <- as.POSIXct(dts1, format = "%Y%m%d %H:%M:%S")
dts3 <- as.POSIXlt(dts1, format = "%Y%m%d %H:%M:%S")
dts.all <- data.frame(dts, ct = dts2, lt = dts3)
# lt changed to POSIXct!
str(dts.all)
## 'data.frame': 10 obs. of 5 variables:
## $ day : Factor w/ 3 levels "20081101","20081102",..: 1 1 1 1 1 2 2 2 2 3
## $ time : Factor w/ 10 levels "01:05:00","01:15:00",..: 3 4 6 8 10 2 5 7 9 1
## $ value: Factor w/ 3 levels "5","6","7": 1 1 2 2 1 1 2 3 1 1
## $ ct : POSIXct, format: "2008-11-01 01:20:00" "2008-11-01 06:00:00" ...
## $ lt : POSIXct, format: "2008-11-01 01:20:00" "2008-11-01 06:00:00" ...
```

Not even a warning? Stupid date-times, I don't care if the documentation states that you should use ct's
for data.frames. Let's try again.

```
dts.all <- dts
dts.all$ct <- dts2
dts.all$lt <- dts3
str(dts.all)
## 'data.frame': 10 obs. of 5 variables:
## $ day : Factor w/ 3 levels "20081101","20081102",..: 1 1 1 1 1 2 2 2 2 3
## $ time : Factor w/ 10 levels "01:05:00","01:15:00",..: 3 4 6 8 10 2 5 7 9 1
## $ value: Factor w/ 3 levels "5","6","7": 1 1 2 2 1 1 2 3 1 1
## $ ct : POSIXct, format: "2008-11-01 01:20:00" "2008-11-01 06:00:00" ...
## $ lt : POSIXlt, format: "2008-11-01 01:20:00" "2008-11-01 06:00:00" ...
```

```
unclass(dts.all$ct)
## [1] 1.226e+09 1.226e+09 1.226e+09 1.226e+09 1.226e+09 1.226e+09 1.226e+09 1.226e+09
## [9] 1.226e+09 1.226e+09
4
## attr(,"tzone")
## [1] ""
unclass(dts.all$lt)
## $sec
## [1] 0 0 0 0 0 0 0 0 0 0
##
## $min
## [1] 20 0 20 30 45 15 30 50 0 5
##
## $hour
## [1] 1 6 12 17 21 1 6 12 20 1
##
## $mday
## [1] 1 1 1 1 1 2 2 2 2 3
##
## $mon
## [1] 10 10 10 10 10 10 10 10 10 10
##
## $year
## [1] 108 108 108 108 108 108 108 108 108 108
##
## $wday
## [1] 6 6 6 6 6 0 0 0 0 1
##
## $yday
## [1] 305 305 305 305 305 306 306 306 306 307
##
## $isdst
## [1] 1 1 1 1 1 1 0 0 0 0
##
```

That worked? Suspicious behaviour now may yield suspicious behaviour later. Let's try rounding our
times.

```
# round sets to POSIXlt and fails!
dts.all[, "ct"] <- round(dts.all[, "ct"], units = "hours")
## Warning: provided 9 variables to replace 1 variables
# force back to POSIXct
```

```
dts.all[, "ct"] <- as.POSIXct(round(dts2, units = "hours"))
# rounding our POSIXlt column also fails
dts.all[, "lt"] <- round(dts3, units = "hours")
## Warning: provided 9 variables to replace 1 variables
# while this works
dts.all$lt <- round(dts3, units = "hours")
```

Let's try to replace one element of one row.

5

```
# this fails
dts.all[1, "lt"] <- as.POSIXlt("2008-11-01 01:00:00")
## Warning: provided 9 variables to replace 1 variables
## Error: 'origin' must be supplied
# this works
dts.all$lt[1] <- as.POSIXlt("2008-11-01 01:00:00")
# this works?
dts.all[1, "lt"] <- as.POSIXct("2008-11-01 01:00:00")
```

There's magic in the $. But it seems we can use this trick to force things (that shouldn't be forced?).

So maybe there's a reason we're supposed to use POSIXct times in data.frames. While tricks are discour-
aged, and it might take some e_ort, it's certainly nice to have access to our list attributes.

```
time1 <- dts.all$lt[1]
time2 <- dts.all$lt[2]
# presumably do something with the data from time1 to time2
while (time1 < time2) f
time1$hour <- time1$hour + 1
g
print(sprintf("%s -- %s", time1, time2))
## [1] "2008-11-01 06:00:00 -- 2008-11-01 06:00:00"
```

Daylight Savings and Time Zones [also hate you]

```
# another example
time1 <- dts.all$lt[5]
time2 <- dts.all$lt[7]
while (time1 < time2) f
time1$hour <- time1$hour + 1
# notice how hour is the only attribute to change
print(unlist(time1))
g
## sec min hour mday mon year wday yday isdst
## 0 0 23 1 10 108 6 305 1
## sec min hour mday mon year wday yday isdst
```

```
## 0 0 24 1 10 108 6 305 1
## sec min hour mday mon year wday yday isdst
## 0 0 25 1 10 108 6 305 1
## sec min hour mday mon year wday yday isdst
## 0 0 26 1 10 108 6 305 1
## sec min hour mday mon year wday yday isdst
## 0 0 27 1 10 108 6 305 1
## sec min hour mday mon year wday yday isdst
## 0 0 28 1 10 108 6 305 1
## sec min hour mday mon year wday yday isdst
## 0 0 29 1 10 108 6 305 1
## sec min hour mday mon year wday yday isdst
## 0 0 30 1 10 108 6 305 1
6
## sec min hour mday mon year wday yday isdst
## 0 0 31 1 10 108 6 305 1
## sec min hour mday mon year wday yday isdst
## 0 0 32 1 10 108 6 305 1
print(sprintf("%s -- %s", time1, time2))
## [1] "2008-11-02 08:00:00 -- 2008-11-02 07:00:00"
# DST and tzone are confusing they're still equal
time1 == time2
## [1] TRUE
# combining them seems to work
c(time1, time2)
## [1] "2008-11-02 07:00:00 CST" "2008-11-02 07:00:00 CST"
# default to right tzone
as.POSIXlt(as.POSIXct(time1))
## [1] "2008-11-02 07:00:00 CST"
```

Remember our _rst solution, where we decided to skip dates altogther and convert everything to integer values? It sure would be nice if we didn't have to worry about daylight savings or local time zones. We can
use universal time (UTC) for that. Actually, like specifying the format, it's a good idea to always specify your time zone even if you don't use UTC.

```
# universal time
dts4 <- round(as.POSIXlt(dts1, format = "%Y%m%d %H:%M:%S", tz = "UTC"), units =
"hours")
dts4
## [1] "2008-11-01 01:00:00 UTC" "2008-11-01 06:00:00 UTC" "2008-11-01 12:00:00 UTC"
## [4] "2008-11-01 18:00:00 UTC" "2008-11-01 22:00:00 UTC" "2008-11-02 01:00:00 UTC"
## [7] "2008-11-02 07:00:00 UTC" "2008-11-02 13:00:00 UTC" "2008-11-02 20:00:00 UTC"
```

```
## [10] "2008-11-03 01:00:00 UTC"
# central standard time
round(as.POSIXlt(dts1, format = "%Y%m%d %H:%M:%S", tz = "CST"), units = "hours")
## [1] "2008-11-01 01:00:00 CST" "2008-11-01 06:00:00 CST" "2008-11-01 12:00:00 CST"
## [4] "2008-11-01 18:00:00 CST" "2008-11-01 22:00:00 CST" "2008-11-02 01:00:00 CST"
## [7] "2008-11-02 07:00:00 CST" "2008-11-02 13:00:00 CST" "2008-11-02 20:00:00 CST"
## [10] "2008-11-03 01:00:00 CST"
```

Example 2

Let's examine a function that will _ll in values when we have missing dates.

```
# POSIXct data.frame
mydata.ct <- data.frame(date = dts.all$ct, value = dts.all$value)
# POSIXlt data.frame
mydata.lt <- data.frame(date = NA, value = dts.all$value)
7
mydata.lt$date <- dts.all$lt
# POSIX.lt data.frame using UTC
mydata.lt.utc <- data.frame(date = NA, value = dts.all$value)
mydata.lt.utc$date <- dts4
fill.dates <- function(x) f
dates <- seq(x[1, "date"], x[nrow(x), "date"], by = "hours")
x2 <- data.frame(date = rep(NA, length(dates)), value = NA)
x2$date <- as.POSIXlt(dates)
x2$value[match(as.character(x$date), as.character(x2$date))] <- x$value
for (i in which(is.na(x2$value))) x2[i, "value"] <- x2[i - 1, "value"]
x2
g
data.ct <- fill.dates(mydata.ct)
data.lt <- fill.dates(mydata.lt)
data.lt.utc <- fill.dates(mydata.lt.utc)
cbind(data.ct[21:30, ], data.lt[21:30, ], data.lt.utc[21:30, ])
## date value date value date value
## 21 2008-11-01 21:00:00 2 2008-11-01 21:00:00 2 2008-11-01 21:00:00 2
## 22 2008-11-01 22:00:00 1 2008-11-01 22:00:00 1 2008-11-01 22:00:00 1
## 23 2008-11-01 23:00:00 1 2008-11-01 23:00:00 1 2008-11-01 23:00:00 1
## 24 2008-11-02 00:00:00 1 2008-11-02 00:00:00 1 2008-11-02 00:00:00 1
## 25 2008-11-02 01:00:00 1 2008-11-02 01:00:00 1 2008-11-02 01:00:00 1
## 26 2008-11-02 01:00:00 1 2008-11-02 01:00:00 1 2008-11-02 02:00:00 1
## 27 2008-11-02 02:00:00 1 2008-11-02 02:00:00 1 2008-11-02 03:00:00 1
## 28 2008-11-02 03:00:00 1 2008-11-02 03:00:00 1 2008-11-02 04:00:00 1
## 29 2008-11-02 04:00:00 1 2008-11-02 04:00:00 1 2008-11-02 05:00:00 1
## 30 2008-11-02 05:00:00 1 2008-11-02 05:00:00 1 2008-11-02 06:00:00 1
```

Chron Package

Let's examine the chron package. "chron" is short for chronological, not chronic pain, which would be logical
given the amount of pain programmers have had to deal with implementing date-times over the years (last
joke, I promise). It's not a base package, so you'll have to install it, as will everyone else that runs your code.
It's important to note that chron expects numeric values to be time since origin. If you want 101010 to
mean 2010-10-10, you'll need to set it as a character string _rst (otherwise it's 2246-07-23).

```
library(chron)
# fails on a factor
chron(dts$day, format = "ymd")
## Error: object dates. must be numeric or character
chron(as.character(dts$day), format = "ymd")
## [1] 081101 081101 081101 081101 081101 081102 081102 081102 081102 081103
chron.dts <- chron(date = as.character(dts$day), time = dts$time, format = c("ymd", "h:m:s"))
cdts.all <- dts
cdts.all$cd <- chron.dts
# I hate two-digit year format; methods to display 4 are awful
format(as.POSIXlt(cdts.all$cd, tz = "UTC"), "%Y%m%d %H:%M:%S")
8
## [1] "20081101 01:20:00" "20081101 06:00:00" "20081101 12:20:00" "20081101 17:30:00"
## [5] "20081101 21:45:00" "20081102 01:15:00" "20081102 06:30:00" "20081102 12:50:00"
## [9] "20081102 20:00:00" "20081103 01:05:00"
# options(chron.year.abb=FALSE) # don't use options as a global variable
```

So what does chron have to o_er? In my opinion, very little, it mostly replicates what's already there.

Lubridate Package

Lubridate is another package you may wish to install. It's documentation boasts that it "makes working with dates fun instead of frustrating". It does include many functions unavailable to us with POSIXt objects.More importantly the basic stu_ is easy to do.

```
library(lubridate)
ymd_hms("2012-12-31 23:59:59")
## [1] "2012-12-31 23:59:59 UTC"
ldate <- mdy_hms("12/31/2012 23:59:59")
ldate + seconds(1)
## [1] "2013-01-01 UTC"
month(ldate) <- 8
```

Understanding the di_erent objects used to represent time is the key to using lubridate.

Instants

An instant is a specific moment in time.

```
now()
## [1] "2012-08-30 13:54:09 CDT"
# last day of the month
ceiling_date(now(), unit = "month") - days(1)
## [1] "2012-08-31 CDT"
# is it morning?
am(now())
## [1] FALSE
```

Intervals, Durations, Periods

An interval is the time in between two instants. Know it exists, but assume you don't care. Durations are exact time spans, measured in seconds. It is the absolute time between two events. Periods are used to handle relative time measurements between two events.

```
# how do you want to handle leap years?
ldate - dyears(1)
9
## [1] "2011-09-01 23:59:59 UTC"
ldate - years(1)
## [1] "2011-08-31 23:59:59 UTC"
# or DST?
force_tz(ymd_hms(as.character(dts.all$lt[5])), "America/Chicago") + dhours(6)
## [1] "2008-11-02 03:00:00 CST"
force_tz(ymd_hms(as.character(dts.all$lt[5])), "America/Chicago") + hours(6)
## [1] "2008-11-02 04:00:00 CST"
# within is a nice function to use with intervals
now() %within% new_interval(ymd("2012-07-01"), ymd("2013-06-30"))
## [1] TRUE
```

I don't know about fun, but you could make a very strong argument to use lubridate to handle your date-time variables.

Bonus Example: Determine Your Format

This simple function will try to determine your date-time format. You'll have problems if the format isn't consistent. This function could easily be extended, but as is, it may be helpful.

```
# FUNCTION guessDateFormat @x vector of character dates/datetimes @returnDates return
# actual dates rather than format convert character datetime to POSIXlt datetime, or at
# least guess the format such that you could convert to datetime
guessDateFormat <- function(x, returnDates = FALSE, tzone = "") f
x1 <- x
# replace blanks with NA and remove
```

```
x1[x1 == ""] <- NA
x1 <- x1[!is.na(x1)]
if (length(x1) == 0)
return(NA)
# if it's already a time variable, set it to character
if ("POSIXt" %in% class(x1[1])) f
x1 <- as.character(x1)
g
dateTimes <- do.call(rbind, strsplit(x1, " "))
for (i in ncol(dateTimes)) f
dateTimes[dateTimes[, i] == "NA"] <- NA
g
# assume the time part can be found with a colon
timePart <- which(apply(dateTimes, MARGIN = 2, FUN = function(i) f
any(grepl(":", i))
g))
# everything not in the timePart should be in the datePart
datePart <- setdiff(seq(ncol(dateTimes)), timePart)
# should have 0 or 1 timeParts and exactly one dateParts
if (length(timePart) > 1 || length(datePart) != 1)
stop("cannot parse your time variable")
10
timeFormat <- NA
if (length(timePart)) f
# find maximum number of colons in the timePart column
ncolons <- max(nchar(gsub("[^:]", "", na.omit(dateTimes[, timePart]))))
if (ncolons == 1) f
timeFormat <- "%H:%M"
g else if (ncolons == 2) f
timeFormat <- "%H:%M:%S"
g else stop("timePart should have 1 or 2 colons")
g
# remove all non-numeric values
dates <- gsub("[^0-9]", "", na.omit(dateTimes[, datePart]))
# sep is any non-numeric value found, hopefully / or -
sep <- unique(na.omit(substr(gsub("[0-9]", "", dateTimes[, datePart]), 1, 1)))
if (length(sep) > 1)
stop("too many seperators in datePart")
# maximum number of characters found in the date part
dlen <- max(nchar(dates))
dateFormat <- NA
```

```
# when six, expect the century to be omitted
if (dlen == 6) f
if (sum(is.na(as.Date(dates, format = "%y%m%d"))) == 0) f
dateFormat <- paste("%y", "%m", "%d", sep = sep)
g else if (sum(is.na(as.Date(dates, format = "%m%d%y"))) == 0) f
dateFormat <- paste("%m", "%d", "%y", sep = sep)
g else stop("datePart format [six characters] is inconsistent")
g else if (dlen == 8) f
if (sum(is.na(as.Date(dates, format = "%Y%m%d"))) == 0) f
dateFormat <- paste("%Y", "%m", "%d", sep = sep)
g else if (sum(is.na(as.Date(dates, format = "%m%d%Y"))) == 0) f
dateFormat <- paste("%m", "%d", "%Y", sep = sep)
g else stop("datePart format [eight characters] is inconsistent")
g else f
stop(sprintf("datePart has unusual length: %s", dlen))
g
if (is.na(timeFormat)) f
format <- dateFormat
g else if (timePart == 1) f
format <- paste(timeFormat, dateFormat)
g else if (timePart == 2) f
format <- paste(dateFormat, timeFormat)
g else stop("cannot parse your time variable")
if (returnDates)
return(as.POSIXlt(x, format = format, tz = tzone))
format
g
# generate some dates
mydates <- format(as.POSIXct(sample(31536000, 20), origin = "2011-01-01", tz = "UTC"),
"%m/%d/%Y %H:%M")
mydates
## [1] "02/07/2011 06:51" "11/21/2011 17:03" "09/17/2011 22:42" "02/16/2011 13:45"
## [5] "12/14/2011 19:11" "09/08/2011 09:22" "12/06/2011 14:06" "02/02/2011 11:00"
## [9] "03/27/2011 06:12" "01/05/2011 15:09" "04/15/2011 04:17" "10/20/2011 14:20"
11
## [13] "11/13/2011 21:46" "02/26/2011 03:24" "12/29/2011 11:02" "03/17/2011 02:24"
## [17] "02/27/2011 13:51" "06/27/2011 08:36" "03/14/2011 10:54" "01/28/2011 14:14"
guessDateFormat(mydates)
## [1] "%m/%d/%Y %H:%M"
12
```

Loop functions:

R for Loop

Loops are used in programming tssso repeat a specific block of code. In this article, you will learn to create a for loop in R programming.

A for loop is used to iterate over a vector in R programming.

Syntax of for loop

```
for (val in sequence)
{
   statement
}
```

Here, sequence is a vector and val takes on each of its value during the loop. In each iteration, statement is evaluated.

Flowchart of for loop

Example: for loop

Below is an example to count the number of even numbers in a vector.

```
x <- c(2,5,3,9,8,11,6)
```

```
count <- 0

for (val in x) {

   if(val %% 2 == 0)  count = count+1

}

print(count)
```

Output

```
[1] 3
```

In the above example, the loop iterates 7 times as the vector x has 7 elements.

In each iteration, val takes on the value of corresponding element of x.

We have used a counter to count the number of even numbers in x. We can see that x contains 3 even numbers.

Debugging tools:

What is R Debug?

A grammatically correct program may give us incorrect results due to logical errors. In case, if such errors (i.e. bugs) occur, we need to find out why and where they occur so that you can fix them. The procedure to identify and fix bugs is called “debugging”.

There are number of R debug Functions, such as:

1.traceback()

2.debug()

3.browser()

4.trace()

5.recover()

Fundamental Principles of R Debugging

Programmers often find that they spend more time debugging a program than actually writing it. Good debugging skills are invaluable.

a.The essence of debugging

b.The principle of confirmation

c.Fixing a bugging program is a process of confirming, one by one, that the many things you believe to be true about code actually are true. d.When we find one of our assumptions is not true, we have found a clue to the location of a bug.

For Example:

```
x <- y^2 + 3*g(z, 2)

w <- 28

if (w+q > 0) u <- 1 else v <- -10
```

Start Small:

At least at the beginning of the r debug process, stick to small simple test cases. Working with large data objects may make it harder to think about the problem. Of course, we should eventually test our code in large, complicated cases, but start small.

Debug in a Modular

Top-Down Manner:

Most good software developers agree that code should be written in a modular manner. Our first-level code should not be long with much of it consisting of functions calls. And those functions should not be too lengthy and should call another function if necessary. This makes code easier at the writing stage and also easier for others to understand when it comes time for the code to be extended.We should debug in a top-down manner, too. Suppose we have a set the debug status of our function

f() and f() contains this line.

For Example:

```
Y <- g(x,8)
```

Do not call debug (g) yet. Execute that line and see g() returns the value we expect. If it does then we have to just avoid the time-consuming process of single-stepping Through g(). If g() returns the wrong value, then now is the time to call debug(g).

Antibugging:

If we have a section of a code in which a variable x should be positive. We can insert this line:

```
Stopifnot(x>0)
```

if there is a bug earlier in the code that renders x equals to, say -3, the call to stopifnot() will bring things right there, with an error message like this:

```
Error: x > 0 is not TRUE
```

R Debug Functions

Now we will discuss the above-mentioned functions for debugging in R.

1. traceback()

If our code has already crashed and we want to know where the offending line is, try traceback(). This will (sometimes) show where abouts in the code the problem occurred.When an R function fails, an error is printed to the screen. Immediately after the error, you can call traceback() to see in which function the error occurred. The traceback() function prints the list of functions that were called before the error occurred. The functions are printed in reverse order.

For Example:

```
f<-function(x) {
r<-x-g(x)
```

```
r
}
g<-function(y) {
r<-y*h(y)
r
}
h<-function(z) {
r<-log(z)
if(r<10)
r^2
else
r^3
}
> f(-1)
Error in if (r < 10) r^2 else r^3: missing value where TRUE/FALSE needed
In addition: Warning message:
In log(z) : NaNs produced
> traceback()
3: h(y)
2: g(x)
1: f (-1)
```

2. debug():

The function debug() in R allows one to step through the execution of a function, line by line. At any point, we can print out values of variables or produce a graph of the results within the function. While debugging, we can simply type "c" to continue to the end of the current section of code. traceback() does not tell us where in the function the error occurred. In order to know which line causes the error, we may want to step through the function using debug().

For Example:

Compute the sum of squared error SS = $\sum_{i=1}^{n}(x_i-\mu)$

```
## compute sum of squares

SS<-function(mu,x) {

d<-x-mu

d2<-d^2

ss<-sum(d2)

ss

}

# set seed to get reproducible results

> set.seed(100)

> x<-rnorm(100)

> SS(1,x)

[1] 202.5615

## now start debugging

> debug(SS)

> SS(1,x)

debugging in: SS(1, x)

debug: {
```

```
d <- x - mu

d2 <- d^2

ss <- sum(d2)

ss

}

attr(,"srcfile")

C:\Users\jihk\doc\courses\StatisticalComputing\lecture5\debugex2.R

Browse[1]>
```

After you see the "Browse[1]>" prompt, you can do different things:

Typing n executes the current line and prints the next one;

By typing Q, we can quit the debugging;

Typing where tells where you are in the function call stack;

By typing ls(), we can list all objects in the local environment;

Typing an object name or print(<object name>) tells us current value of the object. If your object has name n, c or Q, we have to use print() to see their values.

3. browser():

The R debug function 'browser()' stops the execution of a function until the user allows it to continue. This is useful if we don't want to step through all the code, line-by-line, but we want it to stop at a certain point so we can check out what is going on. Inserting a call to the browser() in a function will pause the execution of a function at the point where the browser() is called. Similar to using debug() except we can control where execution gets paused.

For Example:

```
h<-function(z) {
```

browser() ## a break point inserted here

r<-log(z)

if(r<10)

r^2

else

r^3

}

> f(-1)

Called from: h(y)

Browse[1]> ls()

[1] "z"

Browse[1]> z

[1] -1

Browse[1]> n

debug: r <- log(z)

Browse[1]> n

debug: if (r < 10) r^2 else r^3

Browse[1]> ls()

[1] "r" "z"

Warning message:

In log(z) : NaNs produced

Browse[1]> r

[1] NaN

Browse[1]> c

Error in if (r < 10) r^2 else r^3 : missing value where TRUE/FALSE needed

>

4. trace():

Calling trace() on a function allows the user to insert bits of code into a function. The syntax for R debug function trace() is a bit strange for first time users. It might be better off using debug().

For Example:

> as.list(body(h))

[[1]]

'{'

[[2]]

r <- log(z)

[[3]]

if (r < 10) r^2 else r^3

attr(,"srcfile")

C:\Users\jihk\doc\courses\StatisticalComputing\lecture5\debugex1.R

> trace("h",quote(if(is.nan(r)) {browser()}), at=3, print=FALSE)

> f(1)

[1] 1

> f(-1)

Called from: eval(expr, envir, enclos)

Browse[1]> ls()

[1] "r" "z"

Warning message:

In log(z) : NaNs produced

Browse[1]> r

[1] NaN

Browse[1]> z

[1] -1

Browse[1]> c

...

>

> trace("h",quote(if(z<0) {z<-1}), at=2, print=FALSE)

[1] "h"

> f(-1)

[1] -1

Test Your R Knowledge

5. recover():

When we are debugging a function, recover() allows us to check variables in upper level functions.

For Example:

> trace("h",quote(if(is.nan(r)) {recover()}), at=3, print=FALSE)

[1] "h"

> f(-1)

Enter a frame number, or 0 to exit

1: f(-1)

2: g(x)

3: h(y)

Selection: 1

Called from: eval(expr, envir, enclos)

Browse[1]> ls()

[1] "x"

Warning message:

In log(z) : NaNs produced

Browse[1]> x

[1] -1

Browse[1]> c

Enter a frame number, or 0 to exit

1: f(-1)

2: g(x)

3: h(y)

Selection: 2

Called from: eval(expr, envir, enclos)

Browse[1]> ls()

[1] "y"

Browse[1]> y

[1] -1

Browse[1]> c

Enter a frame number, or

Enter 0 to exit

1: f(-1)

2: g(x)

3: h(y)

Selection: 3

Called from: eval(expr, envir, enclos)

Browse[1]> ls()

[1] "r" "z"

Browse[1]> r

[1] NaN

Browse[1]> z

[1] -1

Browse[1]> c

Enter a frame number, or

Enter 0 to exit

1: f(-1)

2: g(x)

3: h(y)

Selection: 0

Error in if (r < 10) r^2 else r^3: missing value where TRUE/FALSE needed

recover() can be used as an error handler, set using options() (e.g.options(error=recover)).

When a function throws an error, execution is halted at the point of failure. We can browse the function calls and examine the environment to find the source of the problem.

This was all on R debug and R debug functions.

Simulation:

Code Profiling:

Profiler is a tool for helping you to understand how R spends its time. It provides a interactive graphical interface for visualizing data from Rprof, R's built-in tool for collecting profiling data and, profvis, a tool for visualizing profiles from R. In this document, we'll understand how to profile code using the profiler and walk through a couple examples to help diagnose and fix performance problems. Here's an example of the profiler in use. We'll create a scatter plot of the diamonds data set, which has about 54,000 rows, fit a linear model, and draw a line for the model. If you copy and paste this code into your R console, it'll open the same profiler interface that you see in this document.

```
library(profvis)

profvis({
  data(diamonds, package = "ggplot2")


  plot(price ~ carat, data = diamonds)

  m <- lm(price ~ carat, data = diamonds)

  abline(m, col = "red")

})
```

In the profiler interface, on the top is the code, and on the bottom is the flame graph. If the panels are too narrow, minimizing the console pane will help. In the flame graph, the horizontal direction represents time in milliseconds, and the vertical direction represents the call stack. Looking at the bottom-most items on the stack, almost 0.7 seconds are spent in plot, a much smaller amount of time is spent in lm, and almost no time at all is spent in abline – it doesn't even show up on the flame graph.

Traveling up the stack, plot called plot.formula, which called do.call, and so on. Going up a few more levels, we can see that plot.defaultcalled a number of functions: first deparse, and later, plot.xy. Similarly, lm calls a number of different functions.

On the top, we can see the amount of time and memory spent on each line of code. This tells us, unsurprisingly, that most of the time is spent on the line with plot, and a little bit is spent on the line with lm.

Using the profiler

The profiler is composed by two main tabs:

-Flame graph

-Data

Using the flame graph view

The flame graph tab is interactive. You can try the following:

As you mouse over the flame graph, information about each block will show in the info box.

Yellow flame graph blocks have corresponding lines of code on the left. (White blocks represent code where profvis doesn't have the source code – for example, in base R and in R packages. See the FAQ if you want package code to show up in the code panel.) If you mouse over a yellow block, the corresponding line of code will be highlighted. Note that the highlighted line of code is where the yellow function is calledfrom, not the content of that function. If you mouse over a line of code, all flame graph blocks that were called from that line will be highlighted.

Click on a block or line of code to lock the current highlighting. Click on the background, or again on that same item to unlock the highlighting. Click on another item to lock on that item.

Use the mouse scroll wheel or trackpad's scroll gesture to zoom in or out in the x direction.

Click and drag on the flame graph to pan up, down, left, right.

Double-click on the background to zoom the x axis to its original extent.

Double-click on a flamegraph block to zoom the x axis the width of that block.

Each block in the flame graph represents a call to a function, or possibly multiple calls to the same function. The width of the block is proportional to the amount of time spent in that function. When a function calls another function, another block is added on top of it in the flame graph.

The profiling data has some limitations: some internal R functions don't show up in the flame graph, and it offers no insight into code that's implemented in languages other than R (e.g. C, C++, or Fortran).

Using the data view

The data view provides a top-down tabular view of the profile. Click the `code` column to expand the call stack under investigation and the following columns to reason about resource allocation:

Calls: This column represents the total number of calls performed. Even for fast calls, executing code several times can degrade performance and is worth investigating the total calls from this column is what you would expect to see.

Memory: Memory allocated or deallocated (for negative numbers) at a give stack. This is represented in megabytes and aggregated over all the call stacks over the code in the given row.

Time: Time spent in milliseconds. This field is also aggregated over all the call stacks executed over the code in the given row.

How profiling data is collected

The profiler uses data collected by Rprof, which is part of the base R distribution. At each time interval (default interval is 10ms), the profiler stops the R interpreter, looks at the current function call stack, and records the stack trace and memory into a file. Because it works by sampling, the result isn't deterministic. Each time you profile your code, the result will be

slightly different. Memory results are also influenced by the non-deterministic behavior of the garbage collector, which will make memory deallocations seem somewhat random.

Profiling examples

We'll use the profiler to optimize some simple examples. Please keep in mind that R's sampling profiler is non-deterministic, and that the code in these examples is run and profiled when this knitr document is executed, so the numeric timing values may not exactly match the text.

Profiling time

In this first example, we'll work with a data frame that has 151 columns. One of the columns contains an ID, and the other 150 columns contain numeric values. What we will do is, for each numeric column, take the mean and subtract it from the column, so that the new mean value of the column is zero.

```
times <- 4e5

cols <- 150

data <-

 as.data.frame(x = matrix(rnorm(times * cols, mean = 5),

 ncol = cols))

data <- cbind(id = paste0("g", seq_len(times)), data)


profvis({

 # Store in another variable for this run

 data1 <- data


 # Get column means

 means <- apply(data1[, names(data1) != "id"], 2, mean)
```

```
 # Subtract mean from each column
 for (i in seq_along(means)) {
 data1[, names(data1) != "id"][, i] <-
 data1[, names(data1) != "id"][, i] - means[i]
 }
})
```

Most of the time is spent in the apply call, so that's the best candidate for a first pass at optimization. apply calls as.matrix and aperm. These two functions convert the data frame to a matrix and transpose it – so even before we've done any useful computations, we've spent a large amount of time transforming the data.

We could try to speed this up in a number of ways. One possibility is that we could simply leave the data in matrix form (instead of putting it in a data frame in line 4). That would remove the need for the as.matrix call, but it would still require aperm to transpose the data. It would also lose the connection of each row to the id column, which is undesirable. In any case, using apply over columns looks like it will be expensive because of the call to aperm.

An obvious alternative is to use the colMeans function. But there's also another possibility. Data frames are implemented as lists of vectors, where each column is one vector, so we could use lapply or vapply to apply the mean function over each column. Let's compare the speed of these four different ways of getting column means.

```
profvis({
  data1 <- data
  # Four different ways of getting column means
  means <- apply(data1[, names(data1) != "id"], 2, mean)
  means <- colMeans(data1[, names(data1) != "id"])
```

```
  means <- lapply(data1[, names(data1) != "id"], mean)

  means <- vapply(data1[, names(data1) != "id"], mean, numeric(1))

})
```

The profiler data helped us to identify performance bottlenecks, and understanding of the underlying data structures allowed us to approach the problem in a more efficient way.

Profiling memory

This example addresses some more advanced issues. This time, it will be hard to directly see the causes of slowness, but we will be able to see some of their side-effects, most notably the side-effects from large amounts of memory allocation.

Suppose you have a data frame that contains a column for which you'd like to take a cumulative sum (and you don't know about R's built-in cumsumfunction). Here's one way to do it:

```
profvis({

  data <- data.frame(value = runif(3e4))


  data$sum[1] <- data$value[1]

  for (i in seq(2, nrow(data))) {

    data$sum[i] <- data$sum[i-1] + data$value[i]

  }

})
```

This takes over 2 seconds to calculate the cumulative sum of 30,000 items. That's pretty slow for a computer program. Looking at the profvis visualization, we can see a number of notable features:

Almost all the time is spent in one line of code, line 6. Although this is just one line of code, many different functions that are called on that line.

In the flame graph, you'll see that some of the flame graph blocks have the label $, which means that those samples were spent in the $ function for indexing into an object (in R, the expression x$y is equivalent to `$`(x, "y")).

Because $ is a generic function, it calls the corresponding method for the object, in this case $.data.frame. This function in turn calls [[, which calls [[.data.frame. (Zoom in to see this more clearly.)

Other flame graph cells have the label $<-. The usual syntax for calling this function is x$y <- z; this is equivalent to `$<-`(x, "y", z). (Assignment with indexing, as in x$y[i] <- z is actually a bit more complicated.)

Finally, many of the flame graph cells contain the entire expression from line 6. This can mean one of two things:

R is currently evaluating the expression but is not inside another function call.

R is in another function, but that function does not show up on the stack. (A number of R's internal functions do not show up in the profiling data. See more about this in the FAQ.)

This profiling data tells us that much of the time is spent in $ and $<-. Maybe avoiding these functions entirely will speed things up. To do that, instead of operating on data frame columns, we can operate on temporary vectors. As it turns out, writing a function that takes a vector as input and returns a vector as output is not only convenient; it provides a natural way of creating temporary variables so that we can avoid calling $ and $<- in a loop.

```
profvis({

  csum <- function(x) {

    if (length(x) < 2) return(x)


    sum <- x[1]

    for (i in seq(2, length(x))) {
```

```
    sum[i] <- sum[i-1] + x[i]
  }
  sum
 }
 data$sum <- csum(data$value)
})
```

Using this csum function, it takes just over half a second, which is about 4x as fast as before.

It may appear that no functions are called from line 7, but that's not quite true: that line also calls [, [<-, -, and +.

The [and [<- functions don't appear in the flame graph. They are internal R functions which contain C code to handle indexing into atomic vectors, and are not dispatched to methods. (Contrast this with the first version of the code, where $ was dispatched to $.data.frame).

The - and + functions can show up in a flame graph, but they are very fast so the sampling profiler may or may not happen to take a sample when they're on the call stack.

You probably have noticed the gray blocks labeled <GC>. These represent times where R is doing garbage collection – that is, when it is freeing chunks of memory that were allocated but no longer needed. If R is spending a lot of time freeing memory, that suggests that R is also spending a lot of time allocating memory. This is another common source of slowness in R code.

In the csum function, sum starts as a length-1 vector, and then grows, in a loop, to be the same length as x. Every time a vector grows, R allocates a new block of memory for the new, larger vector, and then copies the contents over. The memory allocated for the old vector is no longer needed, and will later be garbage collected.

To avoid all that memory allocation, copying, and garbage collection, we can pre-allocate a correctly-sized vector for sum. For this data, that will result in 29,999 fewer allocations, copies, and deallocations.

```
profvis({
```

```
  csum2 <- function(x) {
    if (length(x) < 2) return(x)

    sum <- numeric(length(x))  # Preallocate
    sum[1] <- x[1]
    for (i in seq(2, length(x))) {
      sum[i] <- sum[i-1] + x[i]
    }
    sum
  }
  data$sum <- csum2(data$value)
})
```

This version of the code, with csum2, is around 60x faster than our original code. These performance improvements were possible by avoiding calls to $ and $<-, and by avoiding unnecessary memory allocation and copying from growing a vector in a loop.

Why do some function calls not show in the profiler?

As noted earlier, some of R's built-in functions don't show in the flame graph. These include functions like <-, [, and $. Although these functions can occupy a lot of time, they don't show on the call stack. (In one of the examples above, $ does show on the call stack, but this is because it was dispatched to $.data.frame, as opposed to R's internal C code, which is used for indexing into lists.)

In some cases the side-effects of these functions can be seen in the flamegraph. As we saw in the example above, using these functions in a loop led to many memory allocations, which led to garbage collections, or <GC> blocks in the flame graph.

How do I share a profile?

Right now the easiest way to do this is to generate the profile in RStudio, and publish to RPubs using the publish toolbar button:

How do I get code from an R package to show in the code panel?

In typical use, only code written by the user is shown in the code panel. (This is code for which source references are available.) Yellow blocks in the flame graph have corresponding lines of code in the code panel, and when moused over, the line of code will be highlighted. White blocks in the flame graph don't have corresponding lines in the code panel. In most cases, the calls represented by the white blocks are to functions that are in base R and other packages.

The profiler can also show code that's inside an R package. To do this, source refs for the package code must be available. There are two general ways to do this: you can install the package with source refs, or you can use devtools::load_all() to load a package from sources on disk.

Why does the flame graph hide some function calls for Shiny apps?

When profiling Shiny applications, the flame graph will hide many function calls by default. They're hidden because they aren't particularly informative for optimizing code, and they add visual complexity. This feature requires Shiny 0.12.2.9006 or greater.

Why does Sys.sleep() not show up in profiler data?

The R profiler doesn't provide any data when R makes a system call. If, for example, you call Sys.sleep(5), the R process will pause for 5 seconds, but you probably won't see any instances of Sys.sleep in the profiler visualization – it won't even take any horizontal space. For these examples, we've used the pause function instead, which is part of the profvis package. It's similar to Sys.sleep, except that it does show up in the profiling data.

Web References:

https://www.coursera.org/learn/data-cleaning

https://www.class-central.com › Coursera

datasciencespecialization.github.io/getclean/

https://towardsdatascience.com/big-data-what-is-web-scraping-and-how-to-use-it-74e7...

Class: III B.Sc(CT)
Subject Code:16CTU503A Introduction to Data Science Batch(2016-2019)

ttps://towardsdatascience.com/simple-beginning-to-web-scraping-ef2f2287aff9
https://blog.exploratory.io/scrape-data-from-web-pages-50e45b2b150a
https://www.analyticsvidhya.com/.../an-introduction-to-apis-application-programming.

Questions	Opt1	Opt2	Opt3	Opt4	Opt5
Which of the following is used for reading tabul	read.csv	dget	readLines	get	
Which of the following is used for reading in sav	unserialize	load	get	read	
Which of the following statement would read fil	data <- rea	read.data <	data <- rea	data.read <- read(“foo.	
Which of the following function is identical to re	read.csv	read.data	read.tab	read.table	
Which of the following code would read 100 rov	initial <- re	tabAll <- re	initial <- re	initial <- read.table(“da	
Which of the following code opens a connection	data <- rea	data <- rea	data <- rea	data <- readcsv(“foo.tx	
Which of the following extracts first element fro	x[10].	x[1].	x[0].	x[11].	
Point out the correct statement :	There are t	The [opera	The [[oper	The ((operator is used	
Which of the following extracts first four elemer	x[0:4].	x[0:3].	x[1:4].	x[1:3].	
What would be the output of the following code	“a” “b” “c”	“a” “c” “c”	“a” “c” “b”	“a” “b” “b”	
Point out the wrong statement :	$ operator	The [opera	The $ oper	The [[operator is used	
What would be the output of the following code	3	2	1	0	
What would be the output of the following code	1 3 5	2 3 5	3 3 5	file	
Which of the following code extracts the secon	x[2,].	x[1, 2].	x[, 2].	x[2, 2].	
Point out the wrong statement :	$ operator	The [[oper	The $ oper	There are three operat	
Which of the following code extracts 1st elemer	x[[c(2, 1)]].	x[[c(1, 2)]].	x[[c(2, 1,1)]	x[[c(2, 0,1)]].	
_____ , for dumping a textual representati	dput	save	dump	serialize	
_____ , for outputting a textual representatio	dput	save	dump	serialize	
_____ , for saving an arbitrary number of R	dput	save	dump	serialize	
_____ , for converting an R object into a binar	dput	save	dump	serialize	
_____ string indicating how the columns are	sep	colClasses	nrows	file	
_____ character vector indicating the cl	sep	colClasses	nrows	file	
_____ the number of rows in the dataset	sep	colClasses	nrows	file	
_____ logical indicating if the file has a hea	sep	colClasses	nrows	header	
_____ character string indicating the comm	sep	colClasses	comment.c	header	
Partial matching of names is allowed with ____	[and $	[[and [	[[and [$	[[and $	
The _____ operator can take an integer sequen	$	[[	[	((	
The _____ operator can be used to extract singl	$	[	[[	((	
The ____ operator to extract elements by name	$	[	[[	((	
The _____ function can be useful for reading	Load()	readLines()	read()	readpage()	
Text files can be read line by line using the ____	Load()	readpage()	read()	readLines()	
The _____ package is recently developed by	readr	dplyr	read	dr	
The _____ and _____ functions are useful	dump() and	dump() and	dget() and	dump() and dp()	
_____ opens a connection to a file	file	gzfile	bzfile	url	
_____ opens a connection to a file compre	file	gzfile	bzfile	url	
_____ opens a connection to a file compre	file	gzfile	bzfile	url	
_____ opens a connection to a webpage	file	gzfile	bzfile	url	
The ____ function has a number of arguments t	f()	close()	file()	open()	
_____ open file in read only mode	“r”	“a”	“w”	"ab"	
_____ open a file for writing (and initializing	“r”	“a”	“w”	"ab"	
_____ open a file for appending	“r”	“a”	“w”	"ab"	
The_____ operator can be used to extract	$	[	[[	((	
What would be the output of the following code	1 2 3 4	0 1 2 3	1 2 3 4 5	1 2 3 5	
What would be the output of the following code	2 3 5	1 3 3 5	1 2 3	1 2 3 4 5	
What would be the output of the following code	1	3	2	4	
What would be the output of the following code	13	14	15	16	

The _____ function is used to convert in	dput()	save()	serialize()	dump()
Matrices can be subsetted in the usual way with	subset	subsetting	indices	sets
The main functions for converting R objects into	save(), save	save(), save	save(), unse	unserialize(), save.imag
The _____ function is one of the most cor	read.csv()	read.table(	read.data()	read()
_____, a character vector indicating th	sep	header	file	colClasses
The inverse of dump() is _____ function	file()	dput()	source()	dum()
Vectors are basic objects in R and they can be su	((	[	[]	[[
The _____ function is identical to read.table	read.csv()	read.table(	read()	read.data()
Factors are important in statistical modeling and	l() and gl()	lm() and gl	lme() and g	m() and gm()
We can also create an empty list of a prespecifie	create()	file()	vector()	list()
The sequence does not have to be in order; you	specified	legel	unarbitrary	arbitrary
The [[operator can be used to extract _____	no	all	single	double
The $ operator can only be used with _____ r	different	literal	same	unique
A common task in data analysis is removing ___	missing val	segments	changing va	names

Opt6	Answer	
	read.csv	
	load	
.txt")	data <- read.table("foo.txt")	
	read.csv	
atatable.txt'	initial <- read.table("datatable.txt", nrows = 100)	
t")	data <- read.csv("foo.txt")	
	x[1].	
to extract e	There are three operators that can be used to extract subsets of R objects	
	x[1:4].	
	"a" "c" "c"	
to extract e	The $ operator is used to extract elements of a list or a data frame	
	3	
	1 3 5	
	x[, 2].	
ors that car	The $ operator can be used to extract multiple elements from a list	
	x[[c(2, 1)]].	
	dump	
	dput	
	save	
	serialize	
	sep	
	colClasses	
	nrows	
	header	
	comment.char	
	[[and $	
	[[	
	[[	
	$	
	readLines()	
	readLines()	
	readr	
	dump() and dput()	
	file	
	gzfile	
	bzfile	
	url	
	file()	
	"r"	
	"w"	
	"a"	
	[	
	1 2 3 4	
	1 2 3 4 5	
	2	
	14	

serialize()
indices
ge(), and se save(), save.image(), and serialize()
read.table()
colClasses
source()
[
read.csv()
lm() and glm().
vector()
arbitrary
single
literal
missing values

Unit III: Getting and Cleaning data: Obtaining data from the web, from API's, from databases, and from colleagues in various formats. Basics of data cleaning and making data - tidy

Getting Data from the Web

You've tried everything else, and you haven't managed to get your hands on the data you want. You've found the data on the web, but, alas — no download options are available and copy-paste has failed you. Fear not, there may still be a way to get the data out. For example you can:

- Get data from web-based APIs, such as interfaces provided by online databases and many modern web applications (including Twitter, Facebook and many others). This is a fantastic way to access government or commercial data, as well as data from social media sites.
- Extract data from PDFs. This is very difficult, as PDF is a language for printers and does not retain much information on the structure of the data that is displayed within a document. Extracting information from PDFs is beyond the scope of this book, but there are some tools and tutorials that may help you do it.
- Screen scrape web sites. During screen scraping, you're extracting structured content from a normal web page with the help of a scraping utility or by writing a small piece of code. While this method is very powerful and can be used in many places, it requires a bit of understanding about how the web works.

With all those great technical options, don't forget the simple options: often it is worth to spend some time searching for a file with machine-readable data or to call the institution which is holding the data you want.

In this chapter we walk through a very basic example of scraping data from an HTML web page.

What is machine-readable data?

The goal for most of these methods is to get access to machine-readable data. Machine readable data is created for processing by a computer, instead of the presentation to a human user. The structure of such data relates to contained information, and not the way it is displayed eventually.

Examples of easily machine-readable formats include CSV, XML, JSON and Excel files, while formats like Word documents, HTML pages and PDF files are more concerned with the visual layout of the information. PDF for example is a language which talks directly to your printer, it's concerned with position of lines and dots on a page, rather than distinguishable characters.

Scraping web sites: what for?

Everyone has done this: you go to a web site, see an interesting table and try to copy it over to Excel so you can add some numbers up or store it for later. Yet this often does not really work, or the information you want is spread across a large number of web sites. Copying by hand can quickly become very tedious, so it makes sense to use a bit of code to do it.

The advantage of scraping is that you can do it with virtually any web site — from weather forecasts to government spending, even if that site does not have an API for raw data access.

What you can and cannot scrape

There are, of course, limits to what can be scraped. Some factors that make it harder to scrape a site include:

- Badly formatted HTML code with little or no structural information e.g. older government websites.
- Authentication systems that are supposed to prevent automatic access e.g. CAPTCHA codes and paywalls.
- Session-based systems that use browser cookies to keep track of what the user has been doing.
- A lack of complete item listings and possibilities for wildcard search.
- Blocking of bulk access by the server administrators.

Another set of limitations are legal barriers: some countries recognize database rights, which may limit your right to re-use information that has been published online. Sometimes, you can choose to ignore the license and do it anyway — depending on your jurisdiction, you may have special rights as a journalist. Scraping freely available Government data should be fine, but you may wish to double check before you publish. Commercial organizations — and certain NGOs — react with less tolerance and may try to claim that you're "sabotaging" their systems. Other information may infringe the privacy of individuals and thereby violate data privacy laws or professional ethics.

Tools that help you scrape

There are many programs that can be used to extract bulk information from a web site, including browser extensions and some web services. Depending on your browser, tools like Readability (which helps extract text from a page) or DownThemAll (which allows you to download many files at once) will help you automate some tedious tasks, while Chrome's Scraper extension was explicitly built to extract tables from web sites. Developer extensions like FireBug (for Firefox, the same thing is already included in Chrome, Safari and IE) let you track exactly how a web site is structured and what communications happen between your browser and the server.

ScraperWiki is a web site that allows you to code scrapers in a number of different programming languages, including Python, Ruby and PHP. If you want to get started with scraping without the hassle of setting up a programming environment on your computer, this is the way to go. Other web services, such as Google Spreadsheets and Yahoo! Pipes also allow you to perform some extraction from other web sites.

How does a web scraper work?

Web scrapers are usually small pieces of code written in a programming language such as Python, Ruby or PHP. Choosing the right language is largely a question of which community you have access to: if there is someone in your newsroom or city already working with one of these languages, then it makes sense to adopt the same language.

While some of the click-and-point scraping tools mentioned before may be helpful to get started, the real complexity involved in scraping a web site is in addressing the right pages and the right elements within these pages to extract the desired information. These tasks aren't about programming, but understanding the structure of the web site and database.

When displaying a web site, your browser will almost always make use of two technologies: HTTP is a way for it to communicate with the server and to request specific resource, such as documents, images or videos. HTML is the language in which web sites are composed.

The anatomy of a web page

Any HTML page is structured as a hierarchy of boxes (which are defined by HTML "tags"). A large box will contain many smaller ones — for example a table that has many smaller divisions: rows and cells. There are many types of tags that perform different functions — some produce boxes, others tables, images or links. Tags can also have additional properties (e.g. they can be unique identifiers) and can belong to groups called 'classes', which makes it possible to target and capture individual elements within a document. Selecting the appropriate elements this way and extracting their content is the key to writing a scraper.

Viewing the elements in a web page: everything can be broken up into boxes within boxes.

To scrape web pages, you'll need to learn a bit about the different types of elements that can be in an HTML document. For example, the <table> element wraps a whole table, which

has <tr> (table row) elements for its rows, which in turn contain <td> (table data) for each cell. The most common element type you will encounter is , which can basically mean any block of content. The easiest way to get a feel for these elements is by using the developer toolbar in your browser: they will allow you to hover over any part of a web page and see what the underlying code is.

Tags work like book ends, marking the start and the end of a unit. For example *signifies the start of an italicized or emphasized piece of text and* signifies the end of that section

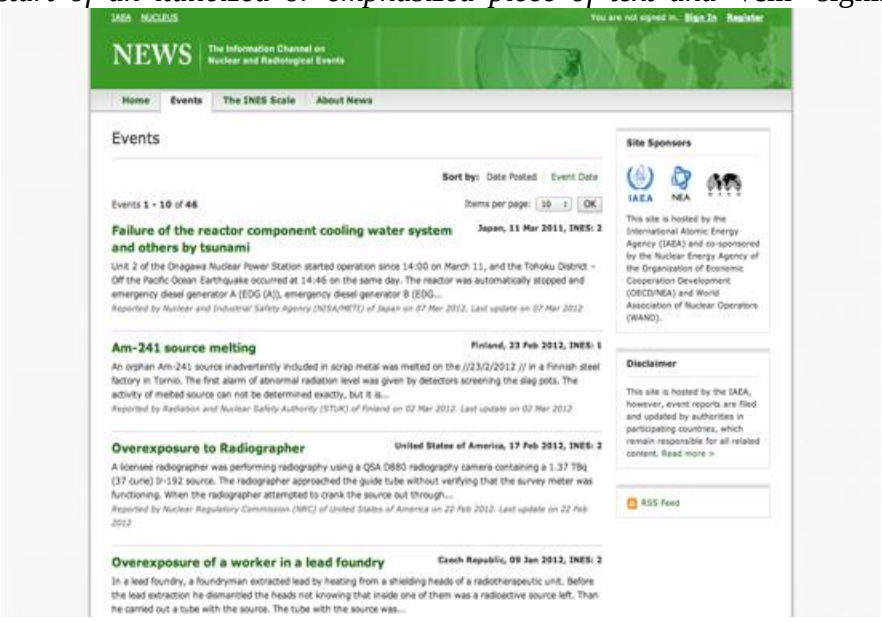


Figure 57. The International Atomic Energy Agency's (IAEA) portal (news.iaea.org)

An example: scraping nuclear incidents with Python

NEWS is the International Atomic Energy Agency's (IAEA) portal on world-wide radiation incidents (and a strong contender for membership in the Weird Title Club!). The web page lists incidents in a simple, blog-like site that can be easily scraped.

To start, create a new Python scraper on ScraperWiki and you will be presented with a text area that is mostly empty, except for some scaffolding code. In another browser window, open

the IAEA site and open the developer toolbar in your browser. In the "Elements" view, try to find the HTML element for one of the news item titles. Your browser's developer toolbar helps you connect elements on the web page with the underlying HTML code.

Investigating this page will reveal that the titles are <h4> elements within a <table>. Each event is a <tr> row, which also contains a description and a date. If we want to extract the titles of all events, we should find a way to select each row in the table sequentially, while fetching all the text within the title elements.

In order to turn this process into code, we need to make ourselves aware of all the steps involved. To get a feeling for the kind of steps required, let's play a simple game: In your ScraperWiki window, try to write up individual instructions for yourself, for each thing you are going to do while writing this scraper, like steps in a recipe (prefix each line with a hash sign to tell Python that this not real computer code). For example:

```
# Look for all rows in the table
# Unicorn must not overflow on left side.
```

Try to be as precise as you can and don't assume that the program knows anything about the page you're attempting to scrape.

Once you've written down some pseudo-code, let's compare this to the essential code for our first scraper:

```
importscraperwiki
fromlxml import html
```

In this first section, we're importing existing functionality from libraries — snippets of pre-written code. scraperwiki will give us the ability to download web sites, while lxml is a tool for the structured analysis of HTML documents. Good news: if you are writing a Python scraper with ScraperWiki, these two lines will always be the same.

```
url = "http://www-news.iaea.org/EventList.aspx"
doc_text = scraperwiki.scrape(url)
doc = html.fromstring(doc_text)
```

Next, the code makes a name (variable): url, and assigns the URL of the IAEA page as its value. This tells the scraper that this thing exists and we want to pay attention to it. Note that the URL itself is in quotes as it is not part of the program code but a *string*, a sequence of characters.

We then use the url variable as input to a function, scraperwiki.scrape. A function will provide some defined job — in this case it'll download a web page. When it's finished, it'll assign its output to another variable, doc_text. doc_text will now hold the actual text of the website — not the visual form you see in your browser, but the source code, including all the tags. Since this form is not very easy to parse, we'll use another function, html.fromstring, to generate a special representation where we can easily address elements, the so-called document object model (DOM).

```
for row in doc.cssselect("#tblEventstr"):
link_in_header = row.cssselect("h4 a").pop()
event_title = link_in_header.text
printevent_title
```

In this final step, we use the DOM to find each row in our table and extract the event's title from its header. Two new concepts are used: the for loop and element selection (.cssselect). The for loop essentially does what its name implies; it will traverse a list of items, assigning each a temporary alias (row in this case) and then run any indented instructions for each item.

The other new concept, element selection, is making use of a special language to find elements in the document. CSS selectors are normally used to add layout information to HTML elements and can be used to precisely pick an element out of a page. In this case (Line. 6) we're selecting #tblEventstr which will match each <tr> within the table element with the ID tblEvents (the hash simply signifies ID). Note that this will return a list of <tr>elements.

As can be seen on the next line (Line. 7), where we're applying another selector to find any <a> (which is a hyperlink) within a <h4> (a title). Here we only want to look at a single element (there's just one title per row), so we have to pop it off the top of the list returned by our selector with the .pop() function.

Note that some elements in the DOM contain actual text, i.e. text that is not part of any markup language, which we can access using the [element].text syntax seen on line 8. Finally, in line 9, we're printing that text to the ScraperWiki console. If you hit run in your scraper, the smaller window should now start listing the event's names from the IAEA web site.

Methodology:Data cleaning

1. Data cleaning

All data sources potentially include errors and missing values – data cleaning addresses these anomalies. Not cleaning data can lead to a range of problems, including linking errors, model misspecification, errors in parameter estimation and incorrect analysis leading users to draw false conclusions.

- The impact of these problems is magnified in the S-DWH environment1 due to the planned re-use of data: if the data contain untreated anomalies, the problems will repeat. The other key data cleaning requirement in a S-DWH is storage of data before cleaning and after every stage of cleaning, and complete metadata on any data cleaning actions applied to the data.
- The main data cleaning processes are editing, validation and imputation. Editing and validation are sometimes used synonymously – in this manual we distinguish them as editing describing the identification of errors, and validation their correction. The remaining process, imputation, is the replacement of missing values.

- Different data types have distinct issues regards data cleaning, so data-specific processing needs to be built into a S-DWH.

1. Census data – although census data do not usually contain a high percentage of anomalies, the• sheer volume of responses, allied with the number of questions, so data cleaning needs to be automatic wherever possible
2. Sample survey data – business surveys generally have less responses, more variables, and more• anomalies than social surveys – and are more complex due to the continuous

nature of the variables (compared to categorical variables for social surveys) – so data cleaning needs to be very differently defined for business and social surveys

3. Administrative data – traditional data cleaning techniques do not work for administrative data• due to the size of the datasets and the underlying data collection (which legally and/or practically precludes recontact to validate responses), so data cleaning needs to be automatic wherever possible

2.Editing:

Editing can take place at different levels, and use different methods – the choice is known as the data editing strategy. Different data editing strategies are needed for each data type – there is no "one size fits all" solution for a S-DWH.

Macro- and micro-editing :

Editing can be at the micro level, editing individual records, or the macro level, editing (aggregate) outputs.

Macro-editing is generally subjective – eye-balling the output, in isolation and/or relative to• similar outputs/previous time periods, or calculating measures of growth and applying rules of thumb to decide whether they are realistic or not. This type of editing would not suit the S-DWHenvironment, as outputs are separated by two layers from inputs, and given the philosophy of re-use of data it would be difficult to define a process where "the needs of the one (output) outweigh the needs of the many". Hence nothing more is said about these methods.

Micro-editing methods are numerous and well-established, and are appropriate for a S-DWH• where editing should only take place in the sources layer. Hence these are the focus here.

Hard and soft edits

Editing methods – known as rules – detect errors, but once a response fails the treatment varies dependent on the rule type.

Hard edits (some validity, consistency, logical and statistical) do not require validation and can• be treated automatically – see below.

Soft edits (all remaining) require external validation – see section

Automatic editing

Automatic editing, mentioned in section 1 as a key option for census data, is also commonly used for business survey data as a cost- and burden-saving measure when responses fail hard edits. Given the high costs associated with development of a S-DWH, automatic editing should be implemented wherever possible – at least during initial development. However, another advantage of automatic editing applies both during development and beyond – it will lead to more timely data, as there will be less time spent validating failures, which will benefit all dependent outputs.

Selective editing

Selective (also significance) editing, like automatic editing, is a cost- and burden-saving measure. It reduces the amount of overall validation required by automatically treating the least important edit rule failures as if they were not failures – the remaining edit rule failures are sent for validation.

The decision whether to validate or not is driven by the selective editing score – all failures with scores above the threshold are sent for validation, all those with scores below are not validated.

The selective editing score is based on the actual return in period t (yt), the expected return E(yt) (usually the return yt-1 in the previous period, but can also be based on administrative data), the weight in period t (wt) – which is 1 for census and administrative data – and the estimated domain total in the previous period (Yt-1): $\boldsymbol{wt}\,|\boldsymbol{yt} - \boldsymbol{E}(\boldsymbol{yt})|\,\boldsymbol{Yt}{-}\mathbf{1}$

The selective editing threshold is set subjectively to balance cost versus quality: the higher the threshold, the better the savings but the worse the quality. In a S-DWH context, as responses can be used for multiple outputs, it is impossible to quantify the quality impact, so selective editing is of questionable utility. It should definitely be out of scope for all data in the backbone of the S-DWH.

Validation Data

validation takes place once responses fail edit rules, and are not treated automatically. The process involves human intervention to decide on the most appropriate treatment for each failure – based on three sources of information (in priority order):

- primary – answer given during a telephone call querying the response, or additional written• information (eg the respondent verified the response when recontacted)

- secondary – previous responses from the same respondent (eg if the current response, although a failure, follows the same pattern as previous responses then the response would be confirmed)
- tertiary – current responses from similar respondents (eg if there are more than one respondents in a household, their information could explain the response that failed the edit rule)

  In addition to these objective sources of information, there is also a valuable subjective source – the experience of the staff validating the data (eg historical knowledge of the reasons for failures).

4.Imputation

The final stage of data cleaning is imputation for partial missing response (item non-response) – the solution for total missing response (unit non-response) is estimation (see 1.3). To determine what imputation method to use requires understanding of the nature of the missing data.

Type of Missingness

Missing data can be characterized as 3 types:

- MCAR (missing completely at random) – the missing responses are a random subsample of the overall sample
- MAR (missing at random) – the rate of missingness varies between identifiable groups, but within these groups the missing responses are MCAR
- NMAR (not missing at random) – the rate of missingness varies between identifiable groups, and within these groups the probability of being missing depends on the outcome variable

Getting Data using API:

API is short for Application Programming Interface. Basically, it means a way of accessing the functionality of a program from inside another program. So instead of performing an action using an interface that was made for humans, a point and click GUI for instance, an API allows a program to perform that action automatically.

The power of this concept becomes only visible, when you imagine that you can mesh the calling of an API in the program with anything else that program might want to do. Some examples from data science:

Retrieve data and produce a visualization from it that gets updated every time someone looks at it

Have tweets automatically translated and entities reported

Have additional nodes in a computer cluster launched as soon as tasks become cumbersome, to ensure fast data processing

While an API can be any defined interface between two programs, today APIs usually refer to a special kind of APIs that are based on the WWW's HyperText Transfer Protocol (HTTP) that is also used by web servers and browsers to exchange data. Indeed, one might consider browsing the web as using APIs: a program (the browser) uses a defined set of commands and conventions to retrieve data (the webpage) from a remote server (the website) and renders it locally in the browser (the thing you see).

All web-based 2 APIs have always the same structure: they consist of a URL to a domain and a path to an endpoint. For instance: http://example.com/api where http://example.com is the URL and /api is the path to the endpoint.

Web-based APIs that are used for data science come usually in two flavors that are named after the HTTP verbs defined 3:

GET – sends a set of parameters, a query to an endpoint and then receives an answer.

POST – sends a data payload to an endpoint to be processed at the remote system, usually receiving only a success message as an answer.

There are other verbs defined in HTTP, like DELETE, but they are less common in APIs. The by far largest group of APIs makes only use of the GET verb. Let's look at that flavor in greater detail.

A canonical example would be an API that allows to retrieve data from some data base and the API's query can be used to narrow down the selection. Let's say an API provides access to newspaper articles. By specifying the parameter year the API returns not all articles, but only

those that were written in a specific year. Let's say we are only interested in articles written in 2014. The corresponding API call would, thus, look like: http://example.com/api?year=2014. We already know which part is the URL and which part is the path to the endpoint. What's new is the query year=2014. Note that it's separated from the path by a question mark. In this example, year is the name of the parameter, and 2014 is its value.Upon receiving the API call, the remote system crafts an answer. The answer can be in any format. It could be a image file, or a movie, or text, or … In recent years, JSON has become the most common answer format by far. JSON is a simple text file that uses special characters and conventions to bring structure into its contents. You can find more info at the Wikipedia page on JSON. For now it suffices to know that is a popular format to store data, that can potentially be nested and delivered together with metadata. And that R can process it quite easily.The big problem with APIs is that they are always designed by humans. So APIs vary wildly in logical structure and the quality of documentation. This unfortunately means, that there is no simple catch-all solution for working with APIs and all programs will need to be custom tailored to the API used. This also means that using an API almost always requires programming to some degree.

Accessing APIs from R

we'll use R to retrieve data from an API and process it. The API we'll query provides data on EU legislative documents. More specifically, we are interested in which week day is most popular for EU energy legislative documents to go into force. Ok, perhaps that's not a mind blowing research question, but one that will allow us to demonstrate the using of APIs and the required data processing quite nicely.

Required package:

There are many facilities in R that can be used to access APIs. The one package that I find most useful is Hadley4's httr. It allows for easy crafting of API calls and also handling the more intricate aspects of APIs like authentication.Working with JSON data is facilitated a lot by the jsonlite package. It does a good job translating JSON's nested data structures into sensible R objects. Well, most of the time, anyway.Since in this example we are going to work with dates, let's use another of Hadley's packages: lubridate. If you work with dates frequently, it's a package that might be a valuable addition to your toolbox.If you don't yet have these packages installed, you can use this R code to obtain them:

```
install.packages(c("httr", "jsonlite", "lubridate"))
```

Databases using R

dplyr as a database interface

The dplyr package simplifies data transformation. It provides a consistent set of functions, called verbs, that can be used in succession and interchangeably to gain understanding of the data iteratively.

dplyr is able to interact with databases directly by translating the dplyr verbs into SQL queries. This convenient feature allows you to 'speak' directly with the database from R. Other advantages of this approach are:

- Run data exploration routines over all of the data, instead of importing part of the data into R.
- Use the SQL Engine to run the data transformations. In effect, computation is being pushed to the database.
- Collect into R only a targeted dataset.
- All of your code is in R. Because dplyr is used to communicate with the database, there is no need to alternate between languages or tools to perform the data exploration.
- Connect to a database
- At the center of this approach is the DBI package. This package acts as 'middle-ware' between packages to allow connectivity with the database from the user or other packages. It provides a consistent set of functions regardless of the database type being accessed. The dplyr package depends on the DBI package for communication with databases.

There are packages that enables a direct connection between the an open-source database and R. Currently, such packages exist for the following databases: MySQL, SQLite, PostgreSQL, and bigquery.

Most commercial databases, like Oracle and Microsoft SQL Server, offer ODBC drivers that allow you to connect your tool to the database. Even though there are R packages that allow you to use ODBC drivers, the connection will most likely not be compatible with DBI. The new odbc package solves that problem by providing a DBI backend to any ODBC driver connection.

If you are interested in creating your own package that connects DBI to a database, please review the article DBI Backend.

SQL Translations for dplyr

A complementary package called dbplyr contains the translations of the vendor-specific SQL for dplyr to use. As of today, we have translations for the following databases:

- Microsoft SQL Server
- Oracle
- Apache Hive
- Apache Impala
- PostgreSQL

In the development version of dbplyr, we also have translations for:

- Amazon Redshift

Tidy Data:

A huge amount of effort is spent cleaning data to get it ready for analysis, but there has been little research on how to make data cleaning as easy and effective as possible.This paper tackles a small, but important, component of data cleaning: data tidying.Tidy datasets are easy to manipulate, model and visualize, and have a specific structure: each variable is a column, each observation is a row, and each type of observational unitis a table. This framework makes it easy to tidy messy datasets because only a small set of tools are needed to deal with a wide range of un-tidy datasets. This structure also makes it easier to develop tidy tools for data analysis, tools that both input and output tidy datasets. The advantages of a consistent data structure and matching tools are demonstrated with a case study free from mundane data manipulation chores.The principles of tidy data provide a standard way to organize data values within a dataset. A standard makes initial data cleaning easier because you do not need to start from scratch and reinvent the wheel every time. The tidy data standard has been designed to facilitate

initial exploration and analysis of the data, and to simplify the development of data analysistools that work well together. Current tools often require translation. You have to spend timemunging the output from one tool so you can input it into another. Tidy datasets and tidy tools work hand in hand to make data analysis easier, allowing you to focus on the interestingdomain problem, not on the uninteresting logistics of data.

Defining tidy data:

Like families, tidy datasets are all alike but every messy dataset is messy in its own way. Tidydatasets provide a standardized way to link the structure of a dataset (its physical layout)with its semantics (its meaning). In this section, I will provide some standard vocabulary fordescribing the structure and semantics of a dataset, and then use those definitions to define tidy data.

Tidy data :

Tidy data is a standard way of mapping the meaning of a dataset to its structure. A dataset is messy or tidy depending on how rows, columns and tables are matched up with observations,variables and types. In tidy data:

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

Tidying messy datasets

Real datasets can, and often do, violate the three precepts of tidy data in almost every way imaginable. While occasionally you do get a dataset that you can start analyzing immediately, this is the exception, not the rule. This section describes the five most common problems with messy datasets, along with their remedies:

. Column headers are values, not variable names.
. Multiple variables are stored in one column.
. Variables are stored in both rows and columns.
. Multiple types of observational units are stored in the same table.
. A single observational unit is stored in multiple tables.

Surprisingly, most messy datasets, including types of messiness not explicitly described above,can be tidied with a small set of tools: melting, string splitting, and casting. The followingsections illustrate each problem with a real dataset that I have encountered, and show howto tidy them. The complete datasets and the R code used to tidy them are available online.Tidy data is only worthwhile if it makes analysis easier. This section discusses tidy tools, tools that take tidy datasets as input and return tidy datasets as output. Tidy tools are useful because the output of one toolcan be used as the input to another. This allows you to simply and easily compose multipletools to solve a problem. Tidy data also ensures that variables are stored in a consistent,explicit manner. This

makes each tool simpler, because it does not need a Swiss Army knifeof parameters for dealing with different dataset structures.

Tools can be messy for two reasons: either they take messy datasets as input (messy-input tools) or they produce messy datasets as output (messy-output tools).
Messy-input tools are typically more complicated than tidy-input tools because they need to include some parts ofthe tidying process. This can be useful for common types of messy datasets, but it typically makes the function more complex, harder to use and harder to maintain. Messy-output tools are frustrating and slow down analysis because they cannot be easily composed and you must constantly think about how to convert from one format to another.

UNIT-3

SI.NO
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46

QUESTION

_____ is a language for printers and does not retain much information on the structure of the data that is displaye

_____extracting structured content from a normal web page with the help of a scraping utility.

Machine _____ ___data is created for processing by a computer, instead of the presentation to a human user.

Extraction of information in_____ is very difficult.

_____method is very powerful and can be used in many places, it requires a bit of understanding about h

_____ are more concerned with the visual layout of the information.

_____is a language which talks directly to your printer.

_____ systems that are supposed to prevent automatic access.

_____system use browser cookies to keep track of what the user has been doing.

Example for authentication systems are_____

_____tool helps to extract text from a page.

_____tool allows you to download many files at once.

_____extension was explicitly built to extract tables from web sites.

_____track exactly how a web site is structured and what communications happen between your browser

_____is a web site that allows you to code scrapers in a number of different programming languages.

_____are small pieces of code written in a programming language such as Python, Ruby or PHP.

_____is a way to communicate with server and to request specific resource like documents, images or

_____is the language in which web sites are composed.

_____is the International Atomic Energy Agency's portal on world-wide radiation incidents.

_____will be presented with a text area that is mostly empty, except for some scaffolding code.

_____methodology includes all data sources potentially errors and missing values.

_____data do not usually contain a high percentage of anomalies, sheer volume of responses.

_____data is business surveys generally have less responses, more variables,more anomalies than social s

_____ data is traditional data cleaning techniques do not work for administrative data due to size of the

Missing data can be characterized as ____types.

_____the missing responses are a random subsample of the overall sample.

_____sends a set of parameters, a query to an endpoint and then receives an answer.

_____sends data to be processed at the remote system,receiving only a success message as an answer.

_____is able to interact with databases directly by translating the dplyr verbs into SQL queries.

The _____ package provides a concise set of operations for managing data frames.

Which of the following return a subset of the columns of a data frame ?

_____data is a standard way of mapping the meaning of a dataset to its structure.

_____ generate summary statistics of different variables in the data frame, possibly within strata.

The _____ operator is used to connect multiple verb actions together into a pipeline.

_____ add new variables/columns or transform existing variables.

The dplyr package can be installed from GitHub using the _____ package.

The dplyr package can be installed from CRAN using :

Which of the following object is masked from 'package:stats' ?

The _____ function can be used to select columns of a data frame that you want to focus on.

_____ function is similar to the existing subset() function in R but is quite a bit faster.

Columns can be arranged in descending order too by using the special ____ operator.

The _____ operator allows you to string operations in a left-to-right fashion.

There is an SQL interface for relational databases via the _____ package.

dplyr can be integrated with the _____ package for large fast tables.

Which of the following function is similar to summarize ?

_____can take place at different levels, and use different methods – the choice is known as the data editing

Editing can take place at different levels, and use different methods is known as_____strategy.
Editing can be in _____forms.
Editing individual records is known as _____editing.
Editing aggregate outputs is known as _____editing.
_____methods are numerous and well-established, and are appropriate for a S-DWH.
_____is generally subjective – eye-balling the output, in isolation and/or relative to similar outputs.
_____edits do not require validation and can be treated automatically.
_____edits (all remaining) require external validation.
_____editing is commonly used for business survey data as cost and measure when responses fail hard e
_____editing is also like automatic editing, is a cost- and burden-saving measure.
_____data takes place once responses fail edit rules, and are not treated automatically.
The process involves human intervention to decide on the most appropriate treatment for each failure based c
_____source answer given during a telephone call querying the response, or additional written information
The final stage of data cleaning is_____ for partial missing response.

OPTION1	OPTION2	OPTION3	OPTION4	ANSWER
API	PDF	XML	CSV	PDF
screen scraping	Google	Yahoo	firefox	screen scraping
readable	writable	both A&B	none of these	readable
PDF	API	excel files	XML	PDF
CAPTCHA	paywalls	screen scraper	none of these	screenscraper
HTML pages	PDF files	excel files	both A&B	both A&B
XML	HTML	PDF	JSON	PDF
commercial	Authentication	session-based	both A&B	Authentication
session-based	commercial	Authentication	both A&B	session-based
CAPTCHA codes	paywalls	both A&B	none of these	both A&B
firebug	Readability	Scraper	DownThemAll	Readablility
Scraper	Readability	DownThemAll	firebug	DownThemAll
firefox	scraper	firebug	none of these	Sraper
firebug	scraper	firefox	paywalls	firebug
Scraperwiki	firebug	chrome	firefox	Scraperwiki
web scraper	CAPTCHAcodes	firebug	XML	web scraper
PHP	HTTP	XML	HTML	HTTP
HTML	XML	JSON	none of these	HTML
IAEA site	NEWS	web scraper	scraperwiki	NEWS
firebug	IAEA site	scraper wiki	firefox	Scraper wiki
Data cleaning	Macro editing	Micro editing	Data editing	Data cleaning
census	survey	Adminstrative	none of these	census
census	survey	Adminstrative	none of these	survey
census	survey	Adminstrative	none of these	Adminstrative
1	2	3	4	3
MACR	MCAR	MAR	NMAR	MCAR
GET	POST	HOLD	SET	GET
GET	POST	HOLD	SET	POST
dplyr	dpl	dplr	dbyr	dplyr
dpl	dplyr	dyr	dplr	dplyr
select	retrieve	get	hold	select
Tidy	Hard	Soft	none of these	Tidy
subset	summarize	rename	filter	summarize
pipe	piper	start	end	pipe
mutate	add	append	arrange	mutate
dtool	dev	devtools	none of these	devtools
install("dplyr")	install.packages("dplyr")	installed("dplyr")	install.dplyr	install.packages("dplyr"
filter	union	difference	setdifference	filter
get	hold	select	rename	select
filter	set	subet	rename	filter
desc()	asc()	descending	des()	desc()
%>%	%<%	%>%>	>%>%>	%>%
DBI	DIB	DB	DB2	DBI
data.table	read.table	table.data	table.read	data.table
rename()	group_by()	group()	subset()	group_by()
Editing	Macro editing	Micro editing	none of these	Editing

data editing	data cleaning	Micro editing	Macro editing	data editing
one	two	three	none of these	two
micro level	macro level	primary level	secondary level	micro level
micro level	macro level	primary level	none of these	macro level
Micro-editing	primary-editing	data-editing	Macro-editing	Micro-editing
Micro-editing	Macro-editing	primary level	none of these	Macro-editing
Hard	Soft	Primary	Data	Hard
Hard	Soft	Primary	Data	Soft
Automatic	Selective	Data	Hard	Automatic
Selective	Hard	Soft	Data	Selective
Validation	Verificaion	Micro level	Macro level	Validation
two	four	three	one	three
primary	secondary	tertiary	none of these	primary
imputation	item response	unit response	missingness	imputation

')

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University Established Under Section 3 of UGC Act 1956)
Pollachi Main Road, Eachanari (p.o)
Coimbatore -21
Class: III B.Sc(CT)
Subject Code:16CTU503A Introduction to Data Science Batch(2016-2019)

Unit IV – Exploratory Analysis: Essential exploratory techniques for summarizing data, applied before formal modeling commences, eliminating or sharpening potential hypotheses about the world that can be addressed by the data, common multivariate statistical techniques use to visualize high-dimensional data.

Introduction :

In statistics, **exploratory data analysis** (**EDA**) is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task. Exploratory data analysis was promoted by John Tukey to encourage statisticians to explore the data, and possibly formulate hypotheses that could lead to new data collection and experiments. EDA is different from initial data analysis (IDA),[1] which focuses more narrowly on checking assumptions required for model fitting and hypothesis testing, and handling missing values and making transformations of variables as needed. EDA encompasses IDA.

Exploratory Data Analysis (EDA) is an approach/philosophy for data analysis that employs a variety of techniques (mostly graphical) to

1. maximize insight into a data set;
2. uncover underlying structure;
3. extract important variables;
4. detect outliers and anomalies;
5. test underlying assumptions;
6. develop parsimonious models; and
7. determine optimal factor settings.

What is meant by data exploration?

Data exploration is the first step in **data** analysis and typically involves summarizing the main characteristics of a dataset. It is commonly conducted using visual analytics tools, but can also be done in more advanced statistical software, such as R.

What is data explosion?

The information **explosion** is the rapid increase in the amount of published information or **data** and the effects of this abundance. As the amount of available **data** grows, the problem of managing the information becomes more difficult, which can lead to information overload.

What is the purpose of exploratory analysis?

In statistics, **exploratory data analysis** (EDA) is an approach to analyzing **data** sets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily EDA is for seeing what the **data** can tell us beyond the formal modeling or hypothesis testing task.

Exploratory analysis

Exploratory Data Analysis is one of the important steps in the data analysis process. Here, the focus is on making sense of the data in hand – things like formulating the correct questions to ask to your dataset, how to manipulate the data sources to get the required answers, and others. This is done by taking an elaborate look at trends, patterns, and outliers using a visual method. Exploratory Data Analysis is a crucial step before you jump to machine learning or modeling of your data. It provides the context needed to develop an appropriate model – and interpret the results correctly.

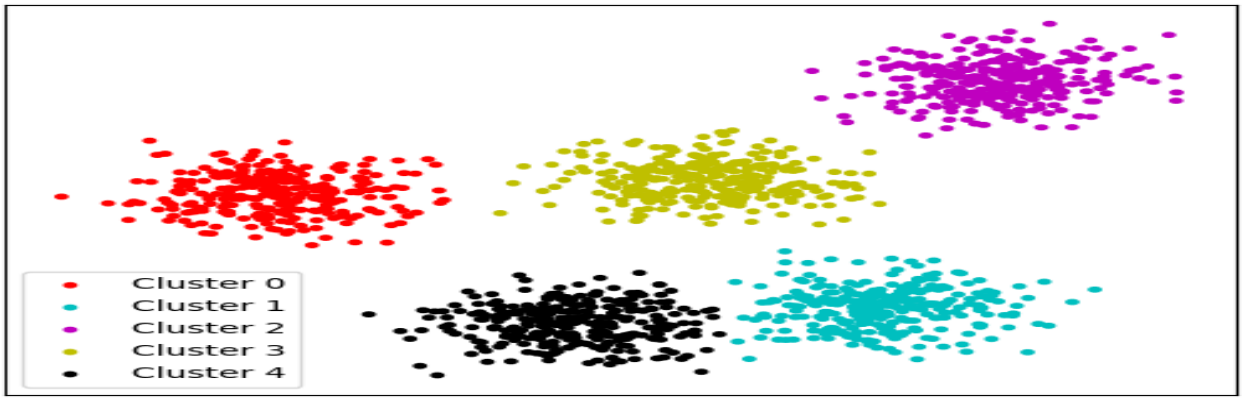
EDA is often seen and described as a philosophy more than science because there are no hard-and-fast rules for approaching it. The purpose of Exploratory Data Analysis is essential to tackle specific tasks such as:Spotting missing and erroneous data;

- Mapping and understanding the underlying structure of your data;
- Identifying the most important variables in your dataset;
- Testing a hypothesis or checking assumptions related to a specific model;
- Establishing a parsimonious model (one that can explain your data using minimum variables);
- Estimating parameters and figuring the margins of error.

Tools and Techniques used in Exploratory Data Analysis

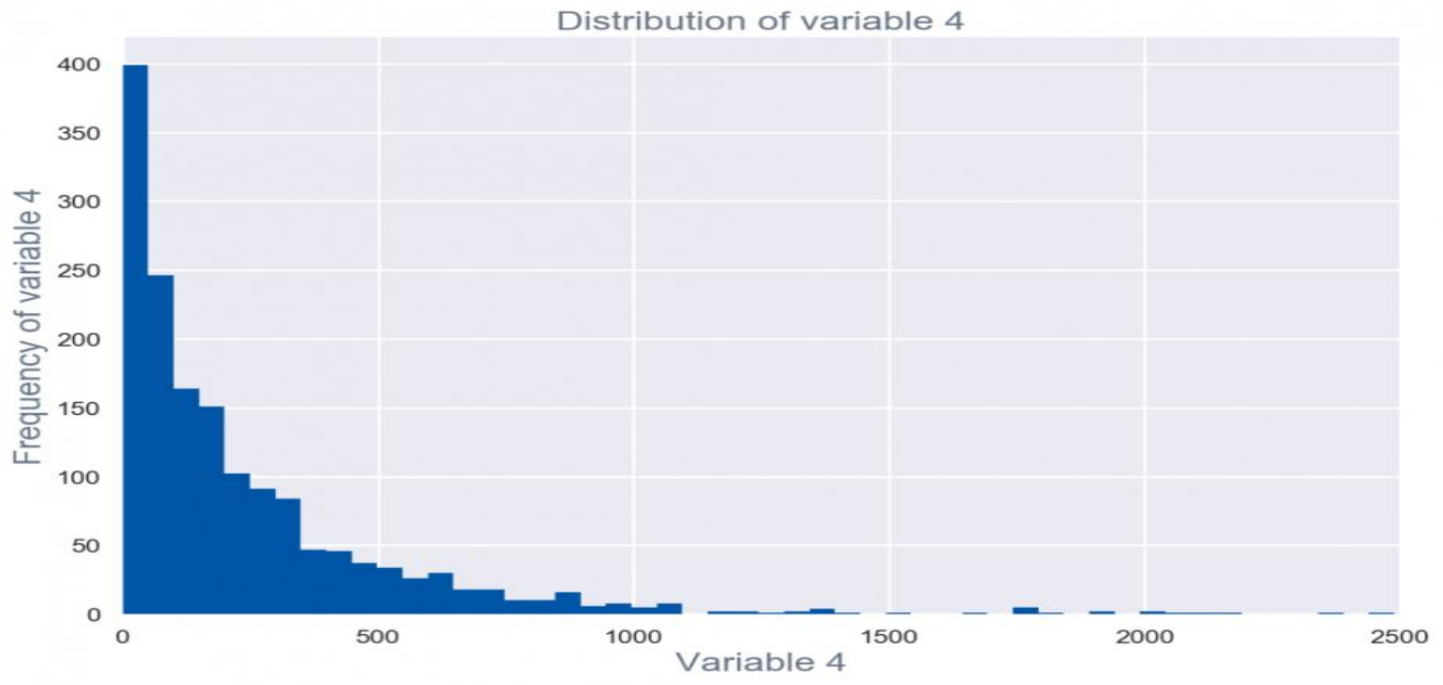
S-Plus and R are the most important statistical programming languages used to perform Exploratory Data Analysis. These languages come bundled with a plethora of tools that help you perform specific statistical functions like:

Classification and dimension reduction techniques



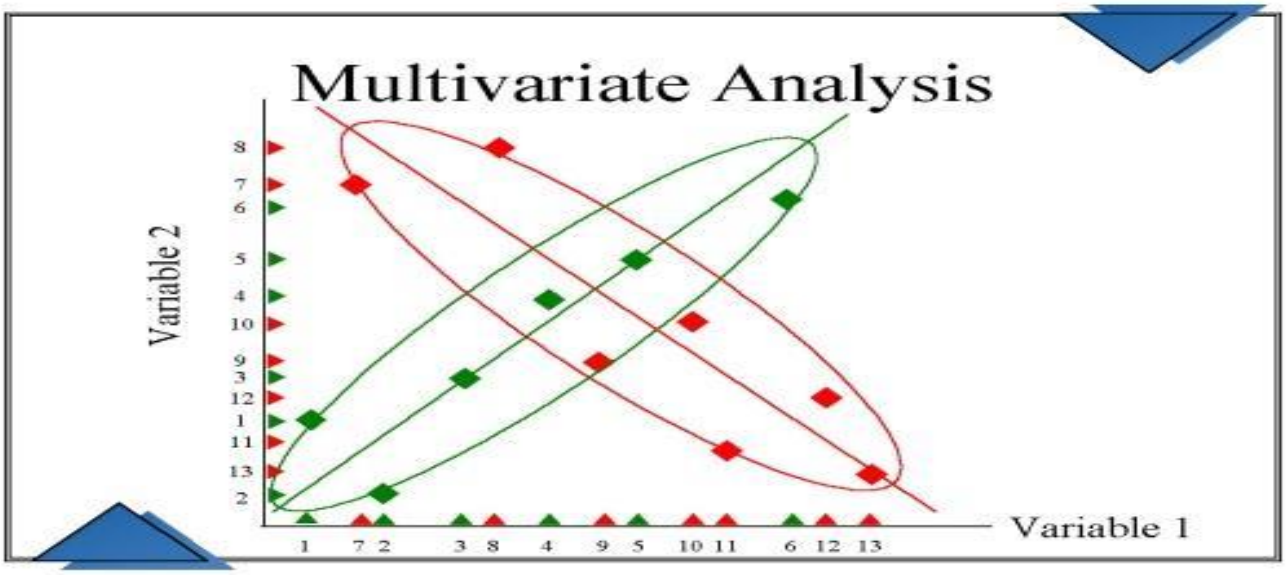
Classification is essentially used to group together different datasets based on a common parameter/variable. The data we're talking about is multi-dimensional, and it's not easy to perform classification or clustering on a multi-dimensional dataset. Hence, to help with that, Dimensionality Reduction techniques like PCA and LDA are performed – these reduce the dimensionality of the dataset without losing out on any valuable information from your data.

Univariate Visualization:



Univariate visualisations are essentially probability distributions of each and every field in the raw dataset – with summary statistics. Univariate visualisations use frequency distribution tables, bar charts, histograms, or pie charts for the graphical representation.

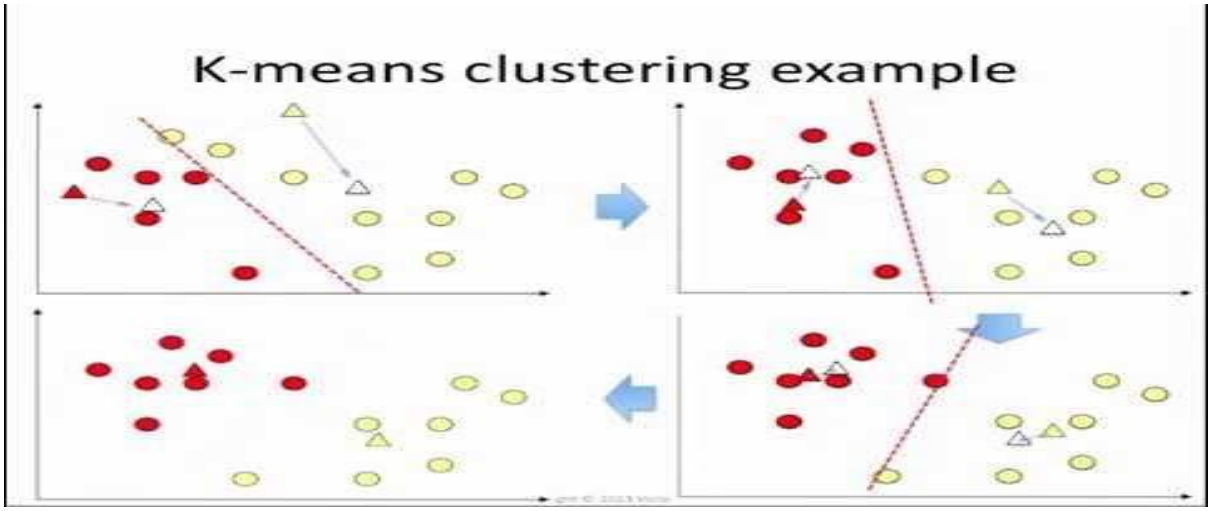
Multivariate analysis:



Multivariate visualizations help in understanding the interactions between different data-fields. It involves observation and analysis of more than one statistical outcome variable at any given time.

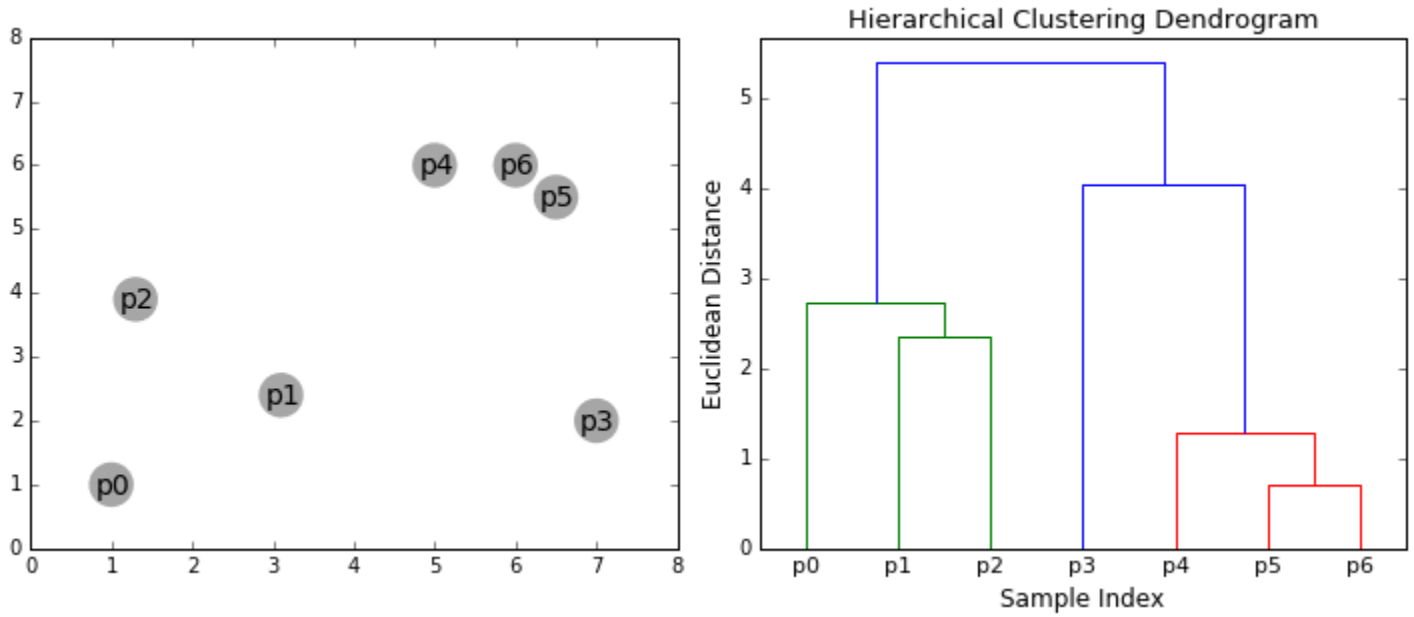
Clustering Methods:

1)K-means Clustering



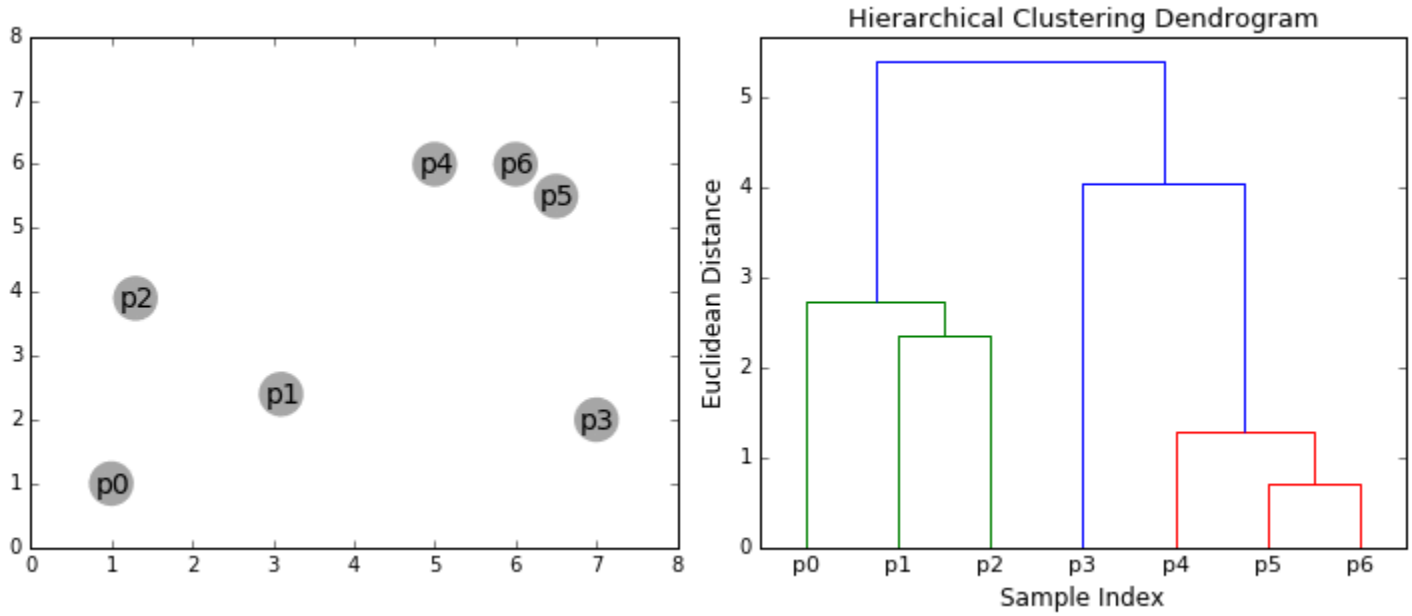
K-means clustering is basically used to create "centers" for each cluster based on the nearest mean. It's an iterative technique that keeps creating and re-creating clusters – until the clusters formed stop changing with iterations. It can be used for finding outliers in a dataset

2) Agglomerative Hierarchial Clustering: Hierarchical clustering algorithms actually fall into 2 categories: top-down or bottom-up. Bottom-up algorithms treat each data point as a single cluster at the outset and then successively merge (or *agglomerate*) pairs of clusters until all clusters have been merged into a single cluster that contains all data points. Bottom-up hierarchical clustering is therefore called *hierarchical agglomerative clustering* or *HAC*. This hierarchy of clusters is represented as a tree (or dendrogram). The root of the tree is the unique cluster that gathers all the samples, the leaves being the clusters with only one sample. Check out the graphic below for an illustration before moving on to the algorithm steps



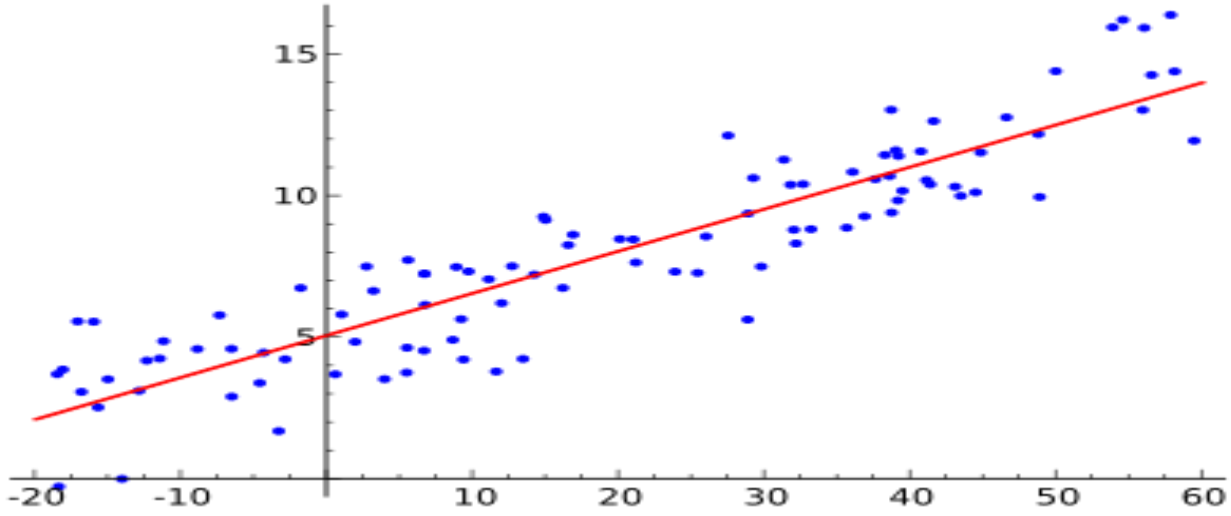
1. we begin by treating each data point as a single cluster i.e if there are X data points in our dataset then we have X clusters. We then select a distance metric that measures the distance between two clusters. As an example we will use *average linkage* which defines the distance between two clusters to be the average distance between data points in the first cluster and data points in the second cluster.
2. On each iteration we combine two clusters into one. The two clusters to be combined are selected as those with the smallest average linkage. I.e according to our selected distance metric, these two clusters have the smallest distance between each other and therefore are the most similar and should be combined.

3. Step 2 is repeated until we reach the root of the tree i.e we only have one cluster which contains all data points. In this way we can select how many clusters we want in the end, simply by choosing when to stop combining the clusters i.e when we stop building the tree!



Agglomerative Hierarchical Clustering

Predictive Models



predictive modeling is a method that uses statistics to predict outcomes. Although most predictions aim to predict what'll happen in the future, predictive modeling can also be applied to any unknown event, regardless of when it's likely to occur. For example, this technique can be used to detect crime and identify suspects even after the crime has happened. The most common way of performing predictive modeling is using linear regression

Basic fundamentals of statistics

Poisson distribution

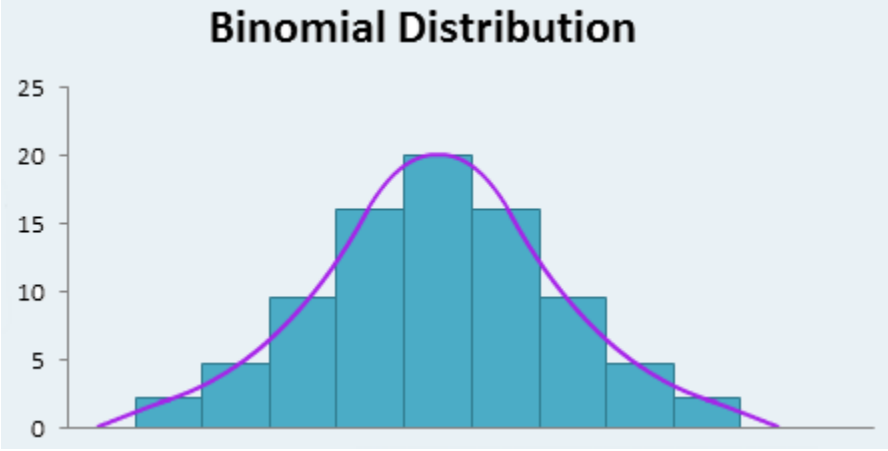
$$P(X = x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

The Poisson distribution is one of the most essential tools in statistics. It's used for to calculate the number of events that are likely to occur in a time interval. For instance, how many phone calls are likely to occur in any particular period of time.

The funny looking symbol in this equation (λ) is known as lambda. It is used to represent the average number of events occurring per time interval.

Another good example where Poisson distribution finds use is to calculate loss in manufacturing. Suppose a machine produces sheets of metal and has X flaws per yard. Suppose, for instance, the error rate was 2 per yard of the sheet – then using Poisson distribution, we can calculate the probability that exactly two errors will occur in a yard.

Binomial Distribution



Experiment of flipping an unbiased coin thrice. Can you tell the probability of the coin showing heads on all three flips? First, from basic combinatorics, we can find out that there are eight possible combinations of results when flipping a coin thrice. Now, we can plot the probabilities of having 0,1,2, or 3 heads. That plot will give us our required binomial distribution for this problem. When graphed, you'll notice that it looks very similar to a typical normal distribution curve, in theory, both are very similar. While Binomial Distribution is for discrete values (a limited number of coin flips), Normal Distribution takes care of continuous values.

How does exploratory analysis help your business and where does fit in?

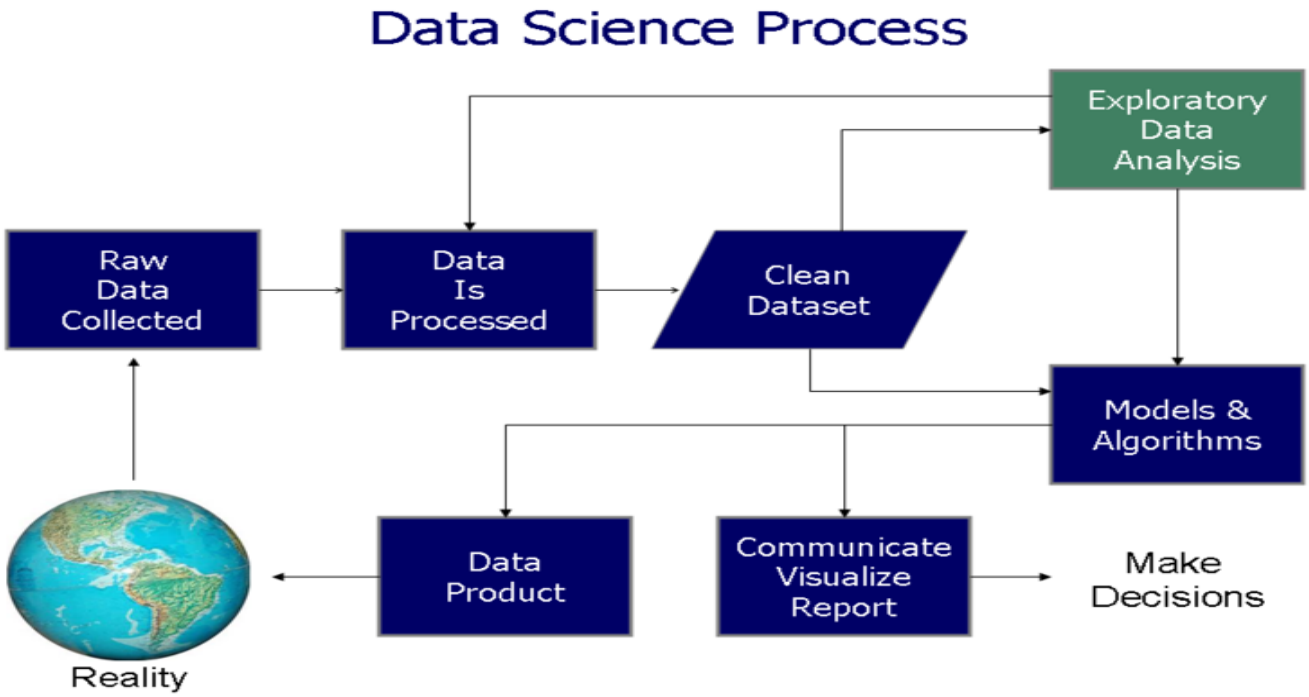
Exploratory Data Analysis provides utmost value to any business by helping scientists understand if the results they've produced are correctly interpreted and if they apply to the required business contexts. Other than just ensuring technically sound results, Exploratory Data Analysis also benefits stakeholders by confirming if the questions they're asking are right or not. Exploratory Data Science often turns up with unpredictable insights – ones that the stakeholders or data scientists wouldn't even care to investigate in general, but which can still prove to be highly informative about the business.

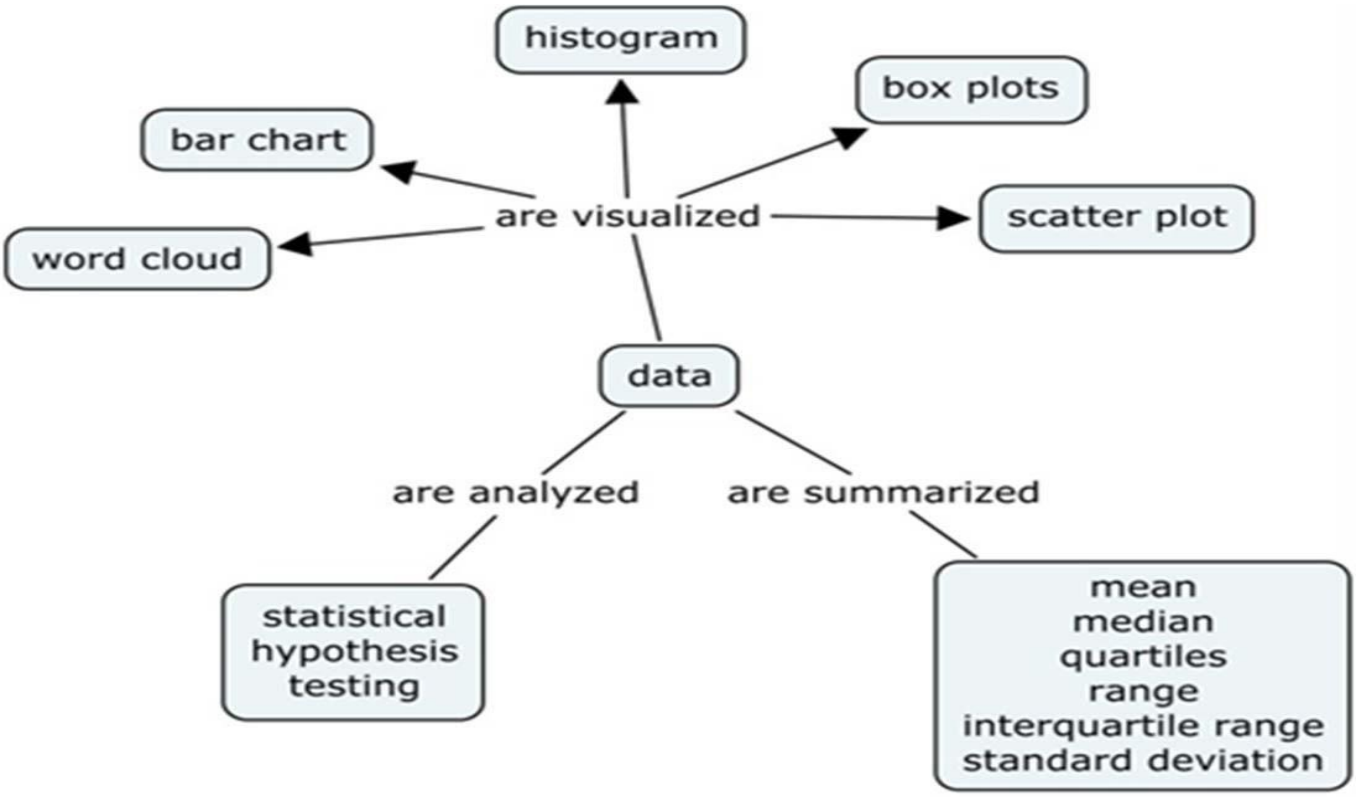
There are a number of data connectors that help organisations incorporate Exploratory Data Analysis directly into their Business Intelligence software. You can also set this up to allow data to flow the other way too, by

building and running statistical models in (for example) R that use BI data and automatically update as new information flows into the model.

Potential use-cases of Exploratory Data Analysis are wide-ranging, but ultimately, it all boils down to this – Exploratory Data Analysis is all about getting to know and understand your data before making any assumptions about it, or taking any steps in the direction of Data Mining. It helps you avoid creating inaccurate models or building accurate models on the wrong data.

Performing this step right will give any organisation the necessary confidence in their data – which will eventually allow them to start deploying powerful machine learning algorithms. However, ignoring this crucial step can lead you to build your Business Intelligence System on a very shaky foundation.





Steps of Data Exploration and Preparation

Remember the quality of your inputs decide the quality of your output. So, once you have got your business hypothesis ready, it makes sense to spend lot of time and efforts here. With my personal estimate, data exploration, cleaning and preparation can take up to 70% of your total project time.

Below are the steps involved to understand, clean and prepare your data for building your predictive model:

1. Variable Identification
2. Univariate Analysis

3. Bi-variate Analysis
4. Missing values treatment
5. Outlier treatment
6. Variable transformation
7. Variable creation

Finally, we will need to iterate over steps 4 – 7 multiple times before we come up with our refined model.

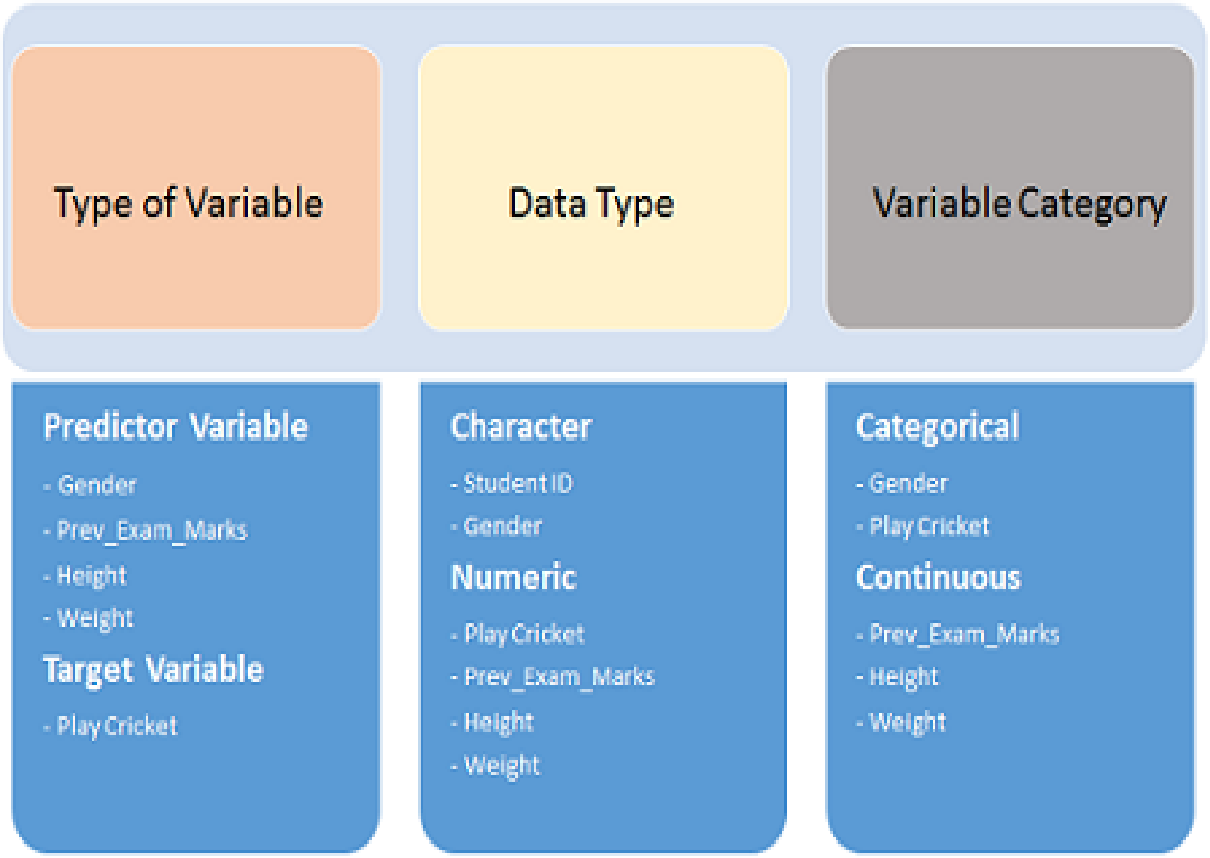
Let's now study each stage in detail:-

Variable Identification

First, identify **Predictor** (Input) and **Target** (output) variables. Next, identify the data type and category of the variables.

Let's understand this step more clearly by taking an example.

Example:- Suppose, we want to predict, whether the students will play cricket or not (refer below data set). Here you need to identify predictor variables, target variable, data type of variables and category of variables.



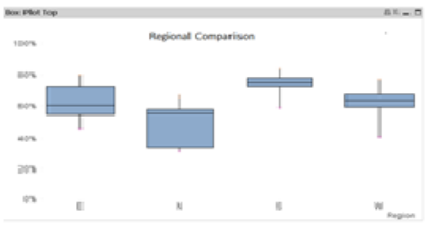
Student_ID	Gender	Prev_Exam_Marks	Height (cm)	Weight Caregory (kgs)	Play Cricket
S001	M	65	178	61	1
S002	F	75	174	56	0
S003	M	45	163	62	1
S004	M	57	175	70	0
S005	F	59	162	67	0

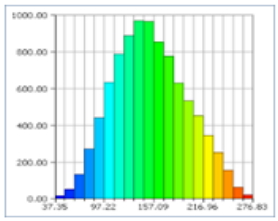
Univariate Analysis

At this stage, we explore variables one by one. Method to perform uni-variate analysis will depend on whether the variable type is categorical or continuous. Let's look at these methods and statistical measures for categorical and continuous variables individually:

Continuous Variables:- In case of continuous variables, we need to understand the central tendency and spread of the variable. These are measured using various statistical metrics visualization methods as shown below:

Central Tendency	Measure of Dispersion	Visualization Methods
Mean	Range	Histogram
Median	Quartile	Box Plot
Mode	IQR	
Min	Variance	
Max	Standard Deviation	
	Skewness and Kurtosis	



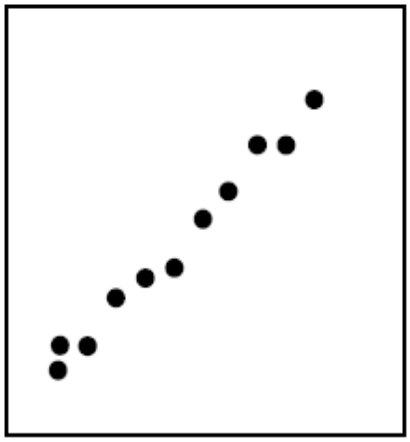


Bi-variate Analysis

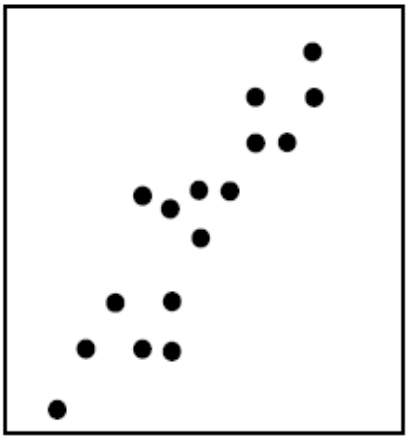
Bi-variate Analysis finds out the relationship between two variables. Here, we look for association and disassociation between variables at a pre-defined significance level. We can perform bi-variate analysis for any combination of categorical and continuous variables. The combination can be: Categorical & Categorical, Categorical & Continuous and Continuous & Continuous. Different methods are used to tackle these combinations during analysis process.

Let's understand the possible combinations in detail:

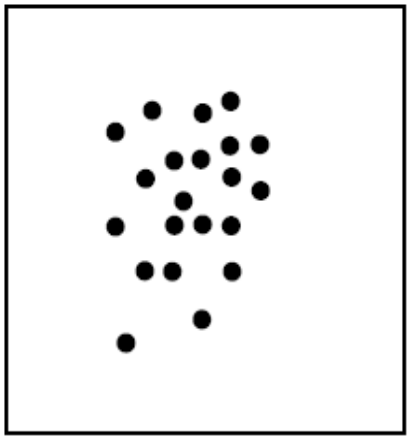
Continuous & Continuous: While doing bi-variate analysis between two continuous variables, we should look at scatter plot. It is a nifty way to find out the relationship between two variables. The pattern of scatter plot indicates the relationship between variables. The relationship can be linear or non-linear.



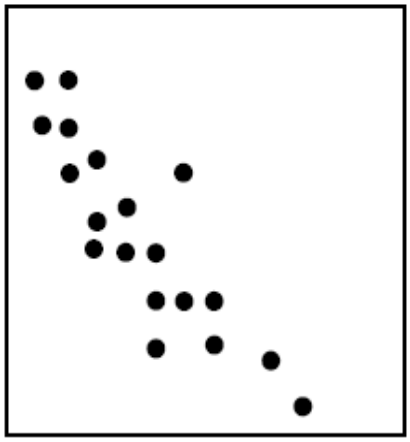
Strong positive correlation



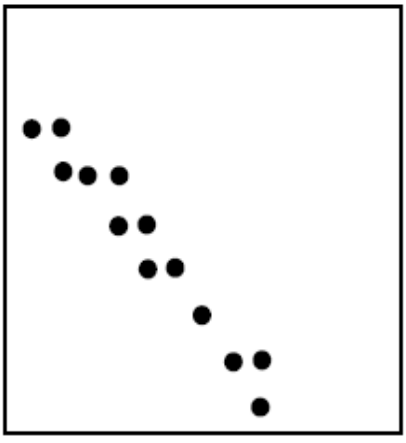
Moderate positive correlation



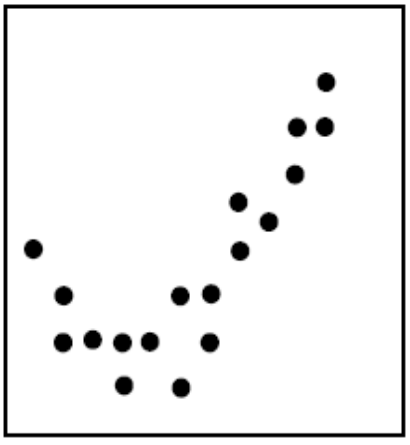
No correlation



Moderate negative correlation



Strong negative correlation



Curvilinear relationship

Scatter plot shows the relationship between two variable but does not indicates the strength of relationship amongst them. To find the strength of the relationship, we use Correlation. Correlation varies between -1 and +1.

- -1: perfect negative linear correlation
- +1:perfect positive linear correlation and
- 0: No correlation

orrelation can be derived using following formula:

Correlation = Covariance(X,Y) / SQRT(Var(X)* Var(Y))

Various tools have function or functionality to identify correlation between variables. In Excel, function CORREL() is used to return the correlation between two variables and SAS uses procedure PROC CORR to identify the correlation. These function returns Pearson Correlation value to identify the relationship between two variables:

X	65	72	78	65	72	70	65	68
Y	72	69	79	69	84	75	60	73

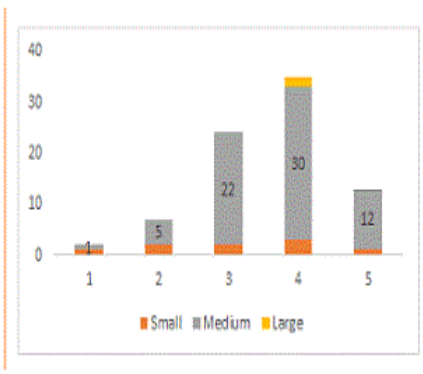
Metrics	Formula	Value
Co-Variance (X,Y)	=COVAR(E6:L6,E7:L7)	18.77
Variance (X)	=VAR.P(E6:L6)	18.48
Variance (Y)	=VAR.P(E7:L7)	45.23
Correlation	=G10/SQRT(G11*G12)	0.65

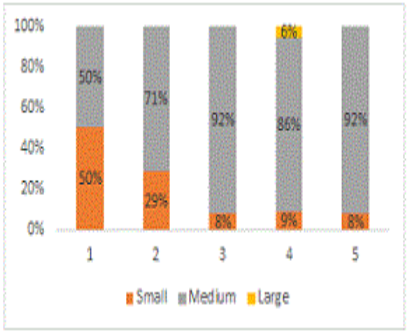
Categorical & Categorical: To find the relationship between two categorical variables, we can use following methods:

- **Two-way table:** We can start analyzing the relationship by creating a two-way table of count and count%. The rows represents the category of one variable and the columns represent the categories of the other variable. We show count or count% of observations available in each combination of row and column categories.
- **Stacked Column Chart:** This method is more of a visual form of Two-way table

Frequency Row Pct	Product Category 1	2	3	4	5	Total
Small	1 11.11	2 22.22	2 22.22	3 33.33	1 11.11	9
Medium	1 1.43	5 7.14	22 31.43	30 42.86	12 17.14	70
Large	0 0.00	0 0.00	0 0.00	2 100.00	0 0.00	2
Total	2	7	24	35	13	81

Frequency Missing = 77





- **Chi-Square Test:** This test is used to derive the statistical significance of relationship between the variables. Also, it tests whether the evidence in the sample is strong enough to generalize that the relationship for a larger population as well. Chi-square is based on the difference between the expected and observed frequencies in one or more categories in the two-way table. It returns probability for the computed chi-square distribution with the degree of freedom.

Probability of 0: It indicates that both categorical variable are dependent

Probability of 1: It shows that both variables are independent.

Probability less than 0.05: It indicates that the relationship between the variables is significant at 95% confidence. The chi-square test statistic for a test of independence of two categorical variables is found by:

$X^2 = \sum (O - E)^2 / E$ where *O* represents the observed frequency. *E* is the expected frequency under the null hypothesis and computed by:

$$E = \frac{row\ total \times column\ total}{sample\ size}$$

From previous two-way table, the expected count for product category 1 to be of small size is 0.22. It is derived by taking the row total for Size (9) times the column total for Product category (2) then dividing by the sample size (81). This is procedure is conducted for each cell. Statistical Measures used to analyze the power of relationship are:

- Cramer's V for Nominal Categorical Variable
- Mantel-Haenszed Chi-Square for ordinal categorical variable.

Different data science language and tools have specific methods to perform chi-square test. In SAS, we can use **Chisq** as an option with **Proc freq** to perform this test.

Categorical & Continuous: While exploring relation between categorical and continuous variables, we can draw box plots for each level of categorical variables. If levels are small in number, it will not show the statistical significance. To look at the statistical significance we can perform Z-test, T-test or ANOVA.

- **Z-Test/ T-Test:-** Either test assess whether mean of two groups are statistically different from each other or not.

- **ANOVA:-** It assesses whether the average of more than two groups is statistically different.

Example: Suppose, we want to test the effect of five different exercises. For this, we recruit 20 men and assign one type of exercise to 4 men (5 groups). Their weights are recorded after a few weeks. We need to find out whether the effect of these exercises on them is significantly different or not. This can be done by comparing the weights of the 5 groups of 4 men each.

Till here, we have understood the first three stages of Data Exploration, Variable Identification, Uni-Variate and Bi-Variate analysis. We also looked at various statistical and visual methods to identify the relationship between variables.

Now, we will look at the methods of Missing values Treatment. More importantly, we will also look at why missing values occur in our data and why treating them is necessary.

Missing Value Treatment

Why missing values treatment is required?

Missing data in the training data set can reduce the power / fit of a model or can lead to a biased model because we have not analysed the behavior and relationship with other variables correctly. It can lead to wrong prediction or classification.

Name	Weight	Gender	Play Cricket/ Not
Mr. Amit	58	M	Y
Mr. Anil	61	M	Y
Miss Swati	58	F	N
Miss Richa	55		Y
Mr. Steve	55	M	N
Miss Reena	64	F	Y
Miss Rashmi	57		Y
Mr. Kunal	57	M	N

Gender	#Students	#Play Cricket	%Play Cricket
F	2	1	50%
M	4	2	50%
Missing	2	2	100%

Name	Weight	Gender	Play Cricket/ Not
Mr. Amit	58	M	Y
Mr. Anil	61	M	Y
Miss Swati	58	F	N
Miss Richa	55	F	Y
Mr. Steve	55	M	N
Miss Reena	64	F	Y
Miss Rashmi	57	F	Y
Mr. Kunal	57	M	N

Gender	#Students	#Play Cricket	%Play Cricket
F	4	3	75%
M	4	2	50%

Notice the missing values in the image shown above: In the left scenario, we have not treated missing values. The inference from this data set is that the chances of playing cricket by males is higher than females. On the other hand, if you look at the second table, which shows data after treatment of missing values (based on gender), we can see that females have higher chances of playing cricket compared to males.

Why my data has missing values?

We looked at the importance of treatment of missing values in a dataset. Now, let's identify the reasons for occurrence of these missing values. They may occur at two stages:

1. **Data Extraction**: It is possible that there are problems with extraction process. In such cases, we should double-check for correct data with data guardians. Some hashing procedures can also be used to make sure data extraction is correct. Errors at data extraction stage are typically easy to find and can be corrected easily as well.
2. **Data collection**: These errors occur at time of data collection and are harder to correct. They can be categorized in four types:
    - **Missing completely at random:** This is a case when the probability of missing variable is same for all observations. For example: respondents of data collection process decide that they will declare their earning after tossing a fair coin. If an head occurs, respondent declares his / her earnings & vice versa. Here each observation has equal chance of missing value.
    - **Missing at random:** This is a case when variable is missing at random and missing ratio varies for different values / level of other input variables. For example: We are collecting data for age and female has higher missing value compare to male.
    - **Missing that depends on unobserved predictors:** This is a case when the missing values are not random and are related to the unobserved input variable. For example: In a medical study, if a particular diagnostic causes discomfort, then there is higher chance of drop out from the study. This missing value is not at random unless we have included "discomfort" as an input variable for all patients.
    - **Missing that depends on the missing value itself:** This is a case when the probability of missing value is directly correlated with missing value itself. For example: People with higher or lower income are likely to provide non-response to their earning.

Which are the methods to treat missing values ?

1. **Deletion:** It is of two types: List Wise Deletion and Pair Wise Deletion.
    - In list wise deletion, we delete observations where any of the variable is missing. Simplicity is one of the major advantage of this method, but this method reduces the power of model because it reduces the sample size.
    - In pair wise deletion, we perform analysis with all cases in which the variables of interest are present. Advantage of this method is, it keeps as many cases available for analysis. One of the disadvantage of this method, it uses different sample size for different variables.

List wise deletion

Gender	Manpower	Sales
M	25	343
F	.	280
M	33	332
M	.	272
F	25	.
M	29	326
	26	259
M	32	297

Pair wise deletion

Gender	Manpower	Sales
M	25	343
F	.	280
M	33	332
M	.	272
F	25	.
M	29	326
	26	259
M	32	297

- Deletion methods are used when the nature of missing data is "**Missing completely at random**" else non random missing values can bias the model output.

1. **Mean/ Mode/ Median Imputation:** Imputation is a method to fill in the missing values with estimated ones. The objective is to employ known relationships that can be identified in the valid values of the data set to assist in estimating the missing values. Mean / Mode / Median imputation is one of the most frequently used methods. It consists of replacing the missing data for a given attribute by the mean or

median (quantitative attribute) or mode (qualitative attribute) of all known values of that variable. It can be of two types:-

- **Generalized Imputation:** In this case, we calculate the mean or median for all non missing values of that variable then replace missing value with mean or median. Like in above table, variable "**Manpower"** is missing so we take average of all non missing values of "**Manpower"** (**28.33**) and then replace missing value with it.
- **Similar case Imputation:** In this case, we calculate average for gender "**Male"** (29.75) and "**Female**" (25) individually of non missing values then replace the missing value based on gender. For "**Male**", we will replace missing values of manpower with 29.75 and for "**Female**" with 25.

2. **Prediction Model**: Prediction model is one of the sophisticated method for handling missing data. Here, we create a predictive model to estimate values that will substitute the missing data. In this case, we divide our data set into two sets: One set with no missing values for the variable and another one with missing values. First data set become training data set of the model while second data set with missing values is test data set and variable with missing values is treated as target variable. Next, we create a model to predict target variable based on other attributes of the training data set and populate missing values of test data set.We can use regression, ANOVA, Logistic regression and various modeling technique to perform this. There are 2 drawbacks for this approach:
   - The model estimated values are usually more well-behaved than the true values
   - If there are no relationships with attributes in the data set and the attribute with missing values, then the model will not be precise for estimating missing values.
3. **KNN Imputation:** In this method of imputation, the missing values of an attribute are imputed using the given number of attributes that are most similar to the attribute whose values are missing. The similarity of two attributes is determined using a distance function. It is also known to have certain advantage & disadvantages.

- **Advantages:**
  - k-nearest neighbour can predict both qualitative & quantitative attributes
  - Creation of predictive model for each attribute with missing data is not required
  - Attributes with multiple missing values can be easily treated
  - Correlation structure of the data is taken into consideration
- **Disadvantage:**
  - KNN algorithm is very time-consuming in analyzing large database. It searches through all the dataset looking for the most similar instances.
  - Choice of k-value is very critical. Higher value of k would include attributes which are significantly different from what we need whereas lower value of k implies missing out of significant attributes.

After dealing with missing values, the next task is to deal with outliers. Often, we tend to neglect outliers while building models. This is a discouraging practice. Outliers tend to make your data skewed and reduces accuracy. Let's learn more about outlier treatment.

Techniques of Outlier Detection and Treatment

What is an Outlier?

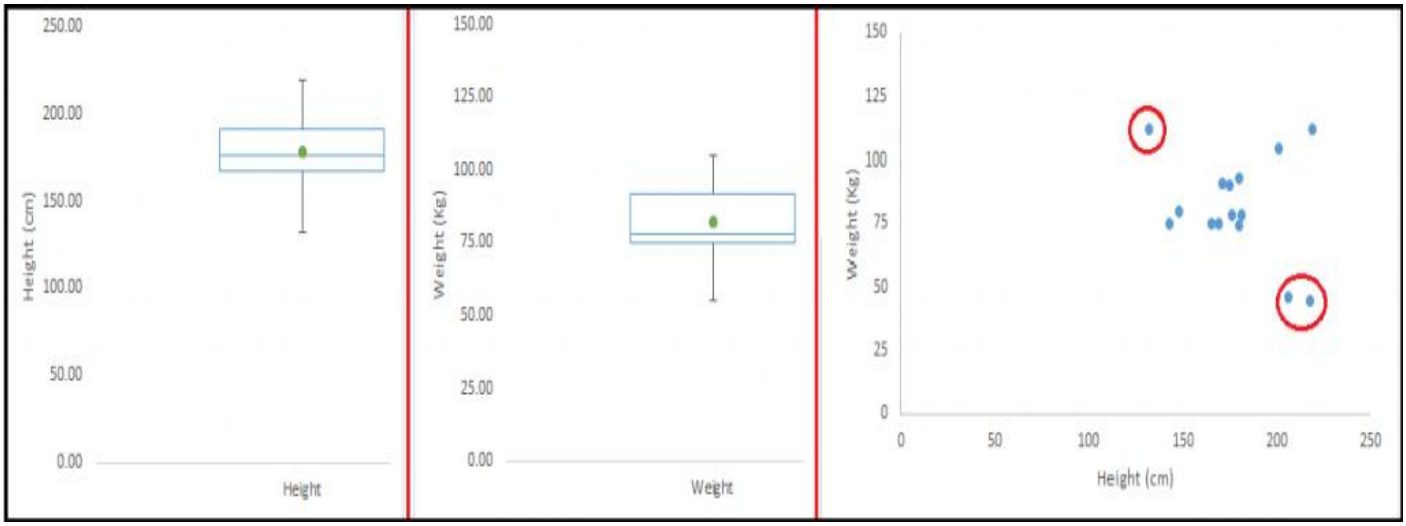
Outlier is a commonly used terminology by analysts and data scientists as it needs close attention else it can result in wildly wrong estimations. Simply speaking, Outlier is an observation that appears far away and diverges from an overall pattern in a sample.

Let's take an example, we do customer profiling and find out that the average annual income of customers is $0.8 million. But, there are two customers having annual income of $4 and $4.2 million. These two customers annual income is much higher than rest of the population. These two observations will be seen as Outliers.

What are the types of Outliers?

Outlier can be of two types: **Univariate** and **Multivariate**. Above, we have discussed the example of univariate outlier. These outliers can be found when we look at distribution of a single variable. Multi-variate outliers are outliers in an n-dimensional space. In order to find them, you have to look at distributions in multi-dimensions.

Let us understand this with an example. Let us say we are understanding the relationship between height and weight. Below, we have univariate and bivariate distribution for Height, Weight. Take a look at the box plot. We do not have any outlier (above and below 1.5*IQR, most common method). Now look at the scatter plot. Here, we have two values below and one above the average in a specific segment of weight and height.



Key Concepts:

Displaying data
Displaying distributions
Displaying info about the variables
Just as with Non-Graphical EDA, Graphical EDA has the same four points as a focal point. These are:

- measures of central tendency, i.e. the mean, the media and mode,
- measures of spread, i.e. variability, variants and standard deviation,
- the shape of the distribution, and
- the existence of outliers.

Types of displays

The distribution of a variable tells us what values the variable takes and how often each value occurs.

Quantitative Line graph across time *1 variable*: Histograms, Boxplots, Stem and Leaf plots, Quantile normal plot *2 variables*: Scatterplots	**Categorical** *1 variable*: Pie charts, Bar graphs *2 or more*: Bar graphs, Pictograms, Contingency Tables
Categorical & Quantitative: boxplot	

Exploratory Data Analysis (EDA)

5.1 Introduction

EDA will help in visualization and transformation to explore your data in a systematic way, a task that statisticians call Exploratory Data Analysis, or EDA for short. EDA is an interactive cycle that involves:

1. Forming questions about your data.
2. Searching for answers by visualizing, transforming, and modeling your data.
3. Using what you discover to refine your questions about the data, or to choose new questions to investigate

EDA is not a formal process with a strict set of rules: you must be free to investigate every idea that occurs to you. Instead, EDA is a loose set of tactics that are more likely to lead to useful insights. This chapter will

teach you a basic toolkit of these useful EDA techniques. Our discussion will lead to a model of data science itself, the model that I've built this book around.

This chapter will point you towards many other interesting packages, more so than any other chapter in the book.

Also recommend the ggplot2 book https://amzn.com/331924275X. The 2nd edition was recently published so it's up-to-date. Contains a lot more details on visualisation. Unfortunately it's not free, but if you're at a university you can get electronic version for free through SpringerLink. This book doesn't contain as much visualisation as it probably should because you can use ggplot2 book as a reference as well.

5.1.1 Prerequisites

In this chapter we'll combine what you've learned about dplyr and ggplot2 to iteratively ask questions, answer them with data, and then ask new questions.

```
library(ggplot2)
library(dplyr)
#>
#> Attaching package: 'dplyr'
#> The following objects are masked from 'package:stats':
#>
#>     filter, lag
#> The following objects are masked from 'package:base':
#>
#>     intersect, setdiff, setequal, union
```

1. What type of **covariation** occurs **between** my variables?

The rest of this chapter will look at these two questions. I'll explain what variation and covariation are, and I'll show you several ways to answer each question. To make the discussion easier, let's define some terms:

- A **variable** is a quantity, quality, or property that you can measure.
- A **value** is the state of a variable when you measure it. The value of a variable may change from measurement to measurement.
- An **observation** is a set of measurements made under similar conditions (you usually make all of the measurements in an observation at the same time and on the same object). An observation will contain several values, each associated with a different variable. I'll sometimes refer to an observation as a data point.

- *Tabular data* is a set of values, each associated with a variable and an observation. Tabular data is *tidy* if each value is placed in its own "cell", each variable in its own column, and each observation in its own row.

  For now, assume all the data you see in this book is be tidy. You'll encounter lots of other data in practice, so we'll come back to these ideas again in tidy data where you'll learn how to tidy messy data.

5.3 Variation

"What type of variation occurs within my variables?"

Variation is the tendency of the values of a variable to change from measurement to measurement. You can see variation easily in real life; if you measure any continuous variable twice—and precisely enough, you will get two different results. This is true even if you measure quantities that are constant, like the speed of light (below). Each of your measurements will include a small amount of error that varies from measurement to measurement.

Table: (#tab:variation)The speed of light is a universal constant, but variation due to measurement error obscures its value. In 1879, Albert Michelson measured the speed of light 100 times and observed 30 different values (in km/sec).

299850 300000 299960 299830 299880 299880 299890 299910 299890 299870 299740 299980 299940
299790 299880 299910 299810 299920 299840 299870 299900 299930 299960 299810 299880 299850
299810 299890 299780 299810 300070 299650 299940 299880 299860 299870 299820 299860 299810
299740 299930 299760 299880 299880 299720 299840 299800 299880 299760 299810 299850 299810
299800 299830 299720 299840 299770 299720 299810 299940 299950 300000 299850 299800 299620
299850 299760 299840 299790 299950 299980 300000 299880 299790 299860 299840 299740 299850
299810 299800 299980 299960 299900 299760 299970 299840 299750 299850 299820 299810 299880
299960 299840 299800 299950 299840 299760 299780 299850 299870

Discrete and categorical variables can also vary if you measure across different subjects (e.g. the eye colors of different people), or different times (e.g. the energy levels of an electron at different moments).

Every variable has its own pattern of variation, which can reveal interesting information. The best way to understand that pattern is to visualize the distribution of the values that you observe for the variable.

Visualizing Distributions:

A variable is **categorical** if it can only have a finite (or countably infinite) set of unique values. In R, categorical variables are usually saved as factors, integers, or character strings. To examine the distribution of a categorical variable, use a bar chart.

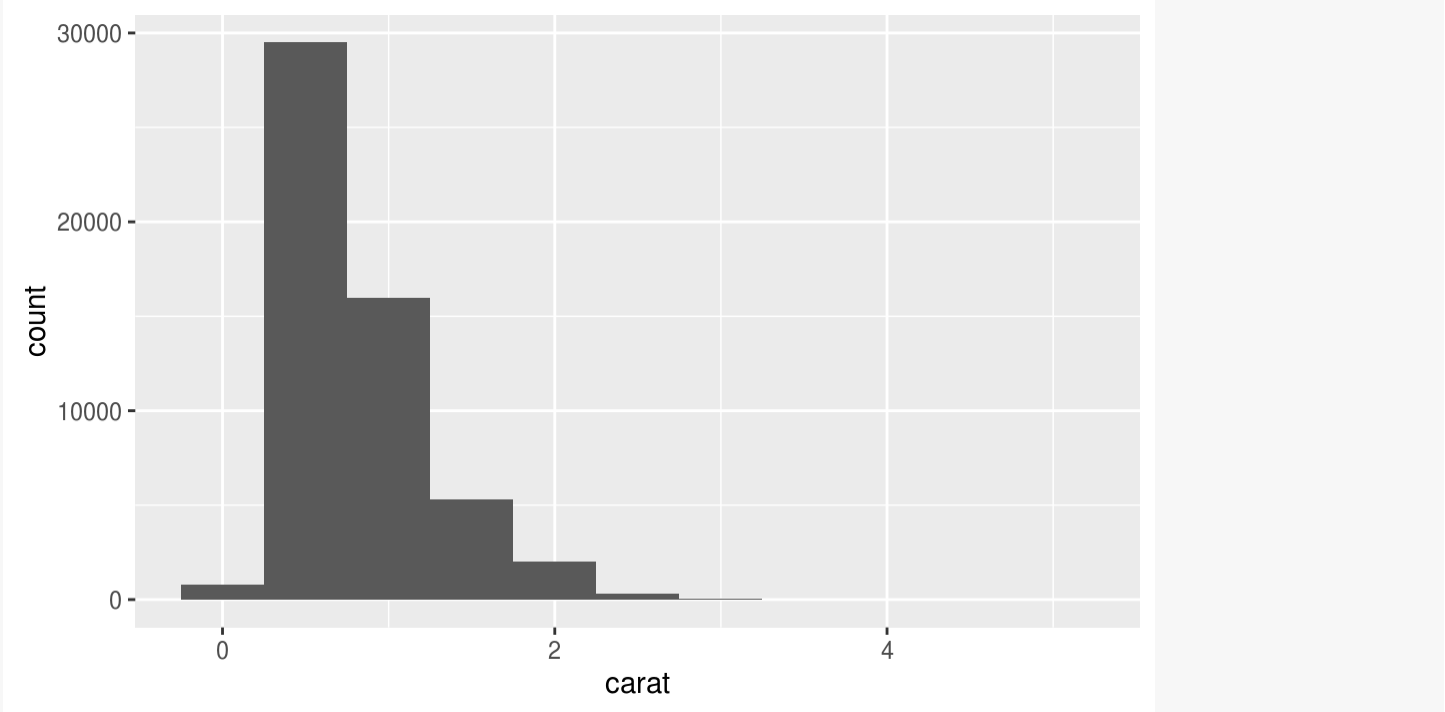
```
gplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut))
```

The height of the bars displays how many observations occurred with each x value. You can compute these values manually with dplyr::count().

```
diamonds %>% count(cut)
#> # A tibble: 5 x 2
#>         cut     n
#>       <ord> <int>
#> 1      Fair  1610
#> 2      Good  4906
#> 3 Very Good 12082
#> 4   Premium 13791
#> 5     Ideal 21551
```

A variable is **continuous** if you can arrange its values in order *and* an infinite number of unique values can exist between any two values of the variable. Numbers and date-times are two examples of continuous variables. To examine the distribution of a continuous variable, use a histogram.

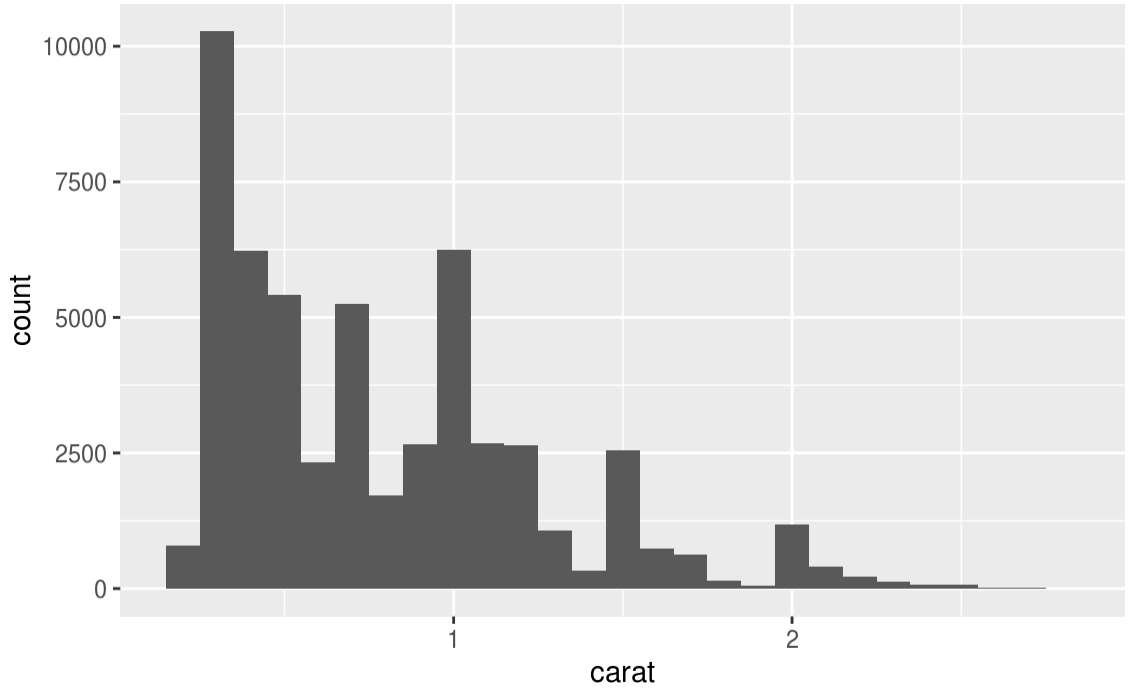
```
ggplot(data = diamonds) +
  geom_histogram(aes(x = carat), binwidth = 0.5)
```



\A histogram divides the x axis into equally spaced intervals and then uses a bar to display how many observations fall into each interval. In the graph above, the tallest bar shows that almost 30,000 observations have a carat. carat value between 0.25 and 0.75, which are the left and right edges of the bar.
You can set the width of the intervals in a histogram with the binwidth argument, which is measured in the units of the xx variable. You should always explore a variety of binwidths when working with histograms, as different binwidths can reveal different patterns. For example, here is how the graph above looks when we zoom into just the diamonds with a binwidth of less than three and choose a smaller binwidth.

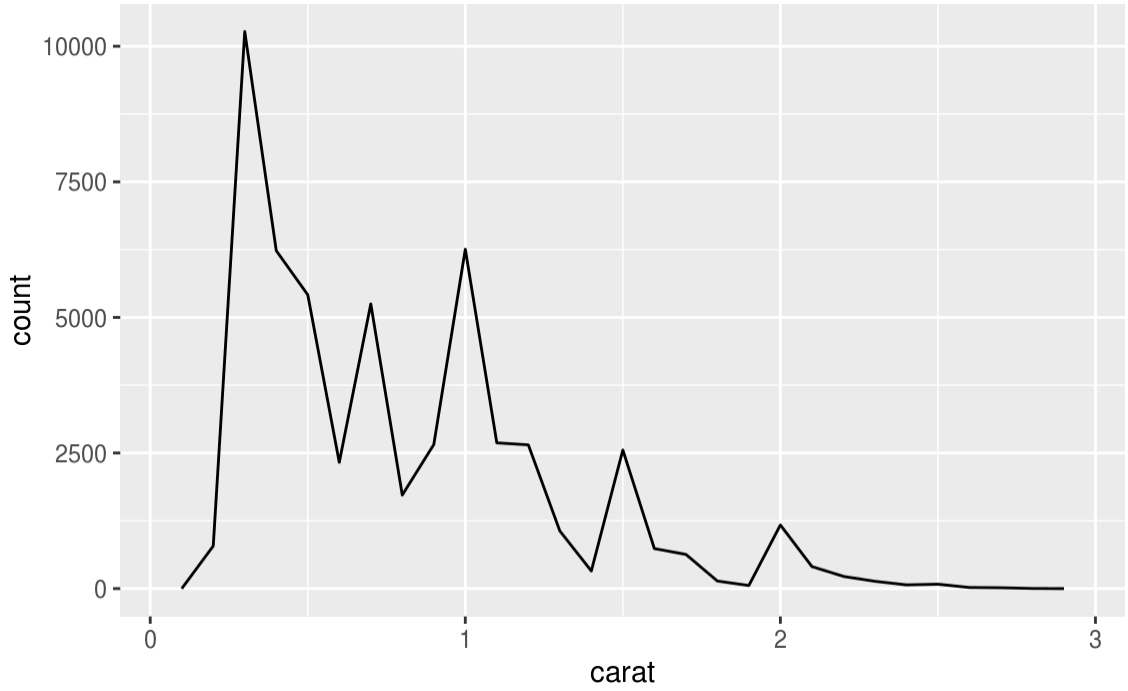
```
smaller <- diamonds %>% filter(carat < 3)

ggplot(data = smaller, mapping = aes(x = carat)) +
  geom_histogram(binwidth = 0.1)
```



If you wish to overlay multiple histograms in the same plot, I recommend using geom_freqpoly() instead of geom_histogram(). geom_freqpoly() performs the same calculation as geom_histogram(), but instead of displaying the counts with bars, uses lines instead. It's much easier to understand overlapping lines than bars.

```
ggplot(data = smaller, mapping = aes(x = carat)) +
  geom_freqpoly(binwidth = 0.1)
```



Now that you can visualize variation, what should you look for in your plots? And what type of follow-up questions should you ask? I've put together a list below of the most useful types of information that you will find in your graphs, along with some follow up questions for each type of information. The key to asking good follow up questions will be to rely on your **curiosity** (What do you want to learn more about?) as well as your **skepticism** (How could this be misleading?).

Typical values

In both bar charts and histograms, tall bars reveal common values of a variable. Shorter bars reveal rarer values. Places that do not have bars reveal values that were not seen in your data. To turn this information into useful questions, look for anything unexpected:

Which values are the most common? Why?

Which values are the rare? Why? Does that match your expectations?

Can you see any unusual patterns? What might explain them?

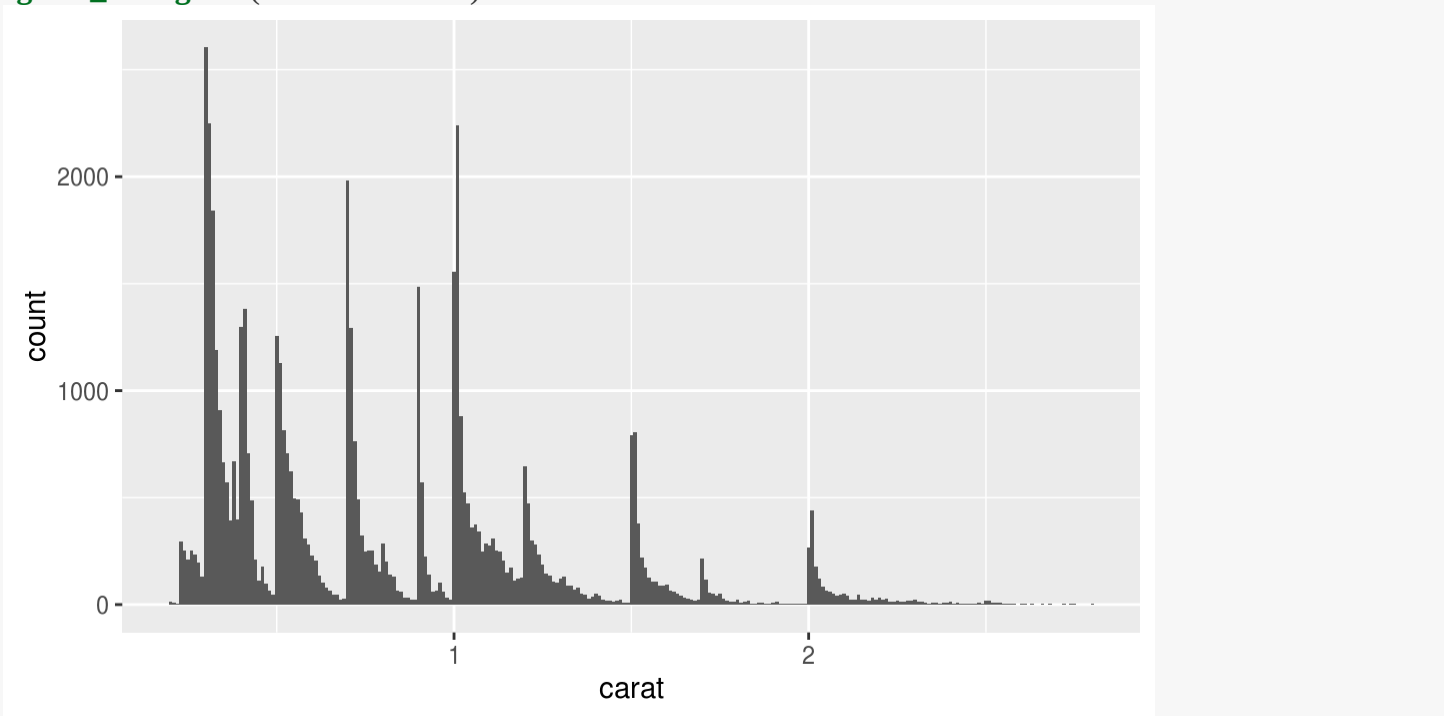
As an example, the histogram below suggests several interesting questions:

Why are there more diamonds at whole carats and common fractions of carats?

Why are there more diamonds slightly to the right of each peak than there are slightly to the left of each peak?

Why are there no diamonds bigger than 3 carats?

```
ggplot(data = smaller, mapping = aes(x = carat)) +
  geom_histogram(binwidth = 0.01)
```

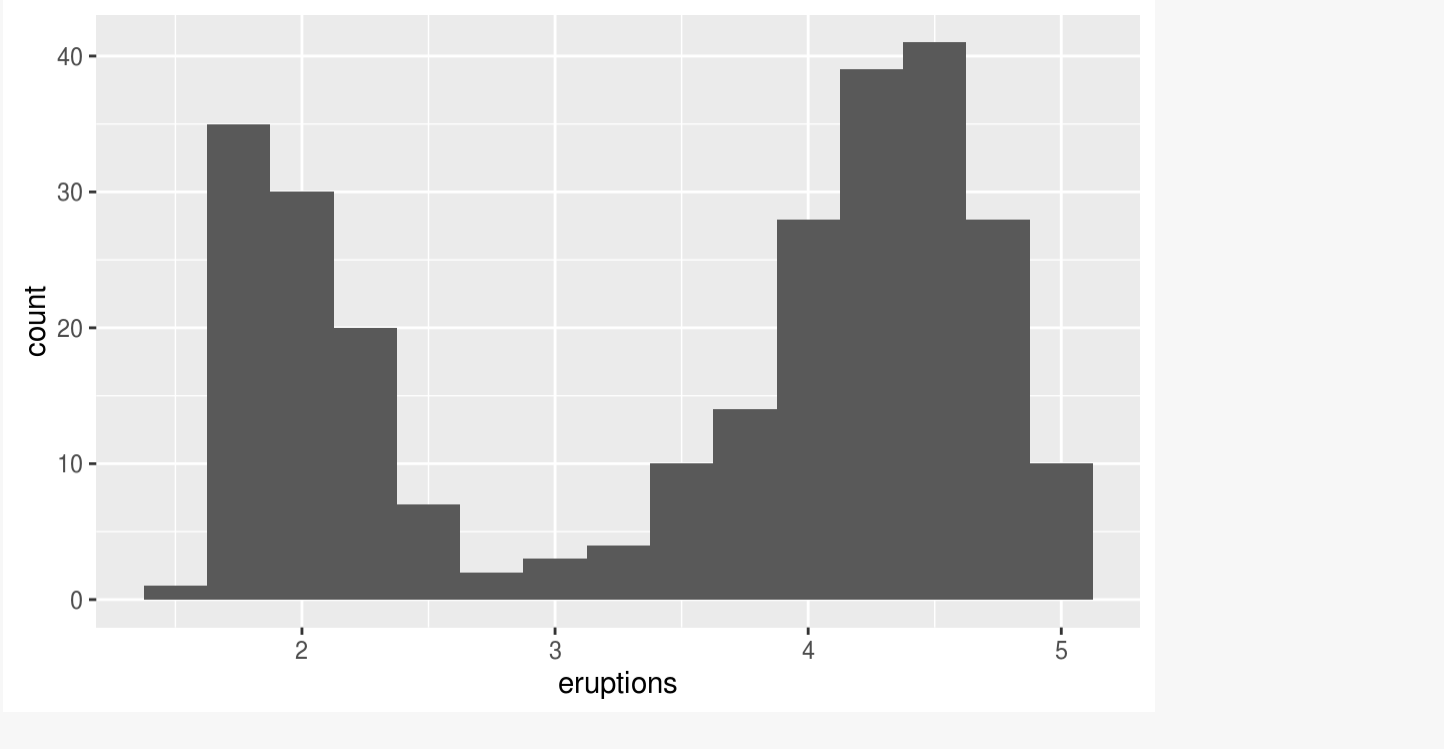


Clusters of similar values suggest that subgroups exist in your data. To understand the subgroups, ask:

- How are the observations within each cluster similar to each other?
- How are the observations in separate clusters different from each other?
- How can you explain or describe the clusters?
- Why might the appearance of clusters be misleading?

The histogram shows the length (in minutes) of 272 eruptions of the Old Faithful Geyser in Yellowstone National Park. Eruption times appear to be clustered in to two groups: there are short eruptions (of around 2 minutes) and long eruption (4-5 minutes), but little in between.

```
ggplot(data = faithful, mapping = aes(x = eruptions)) +
  geom_histogram(binwidth = 0.25)
```

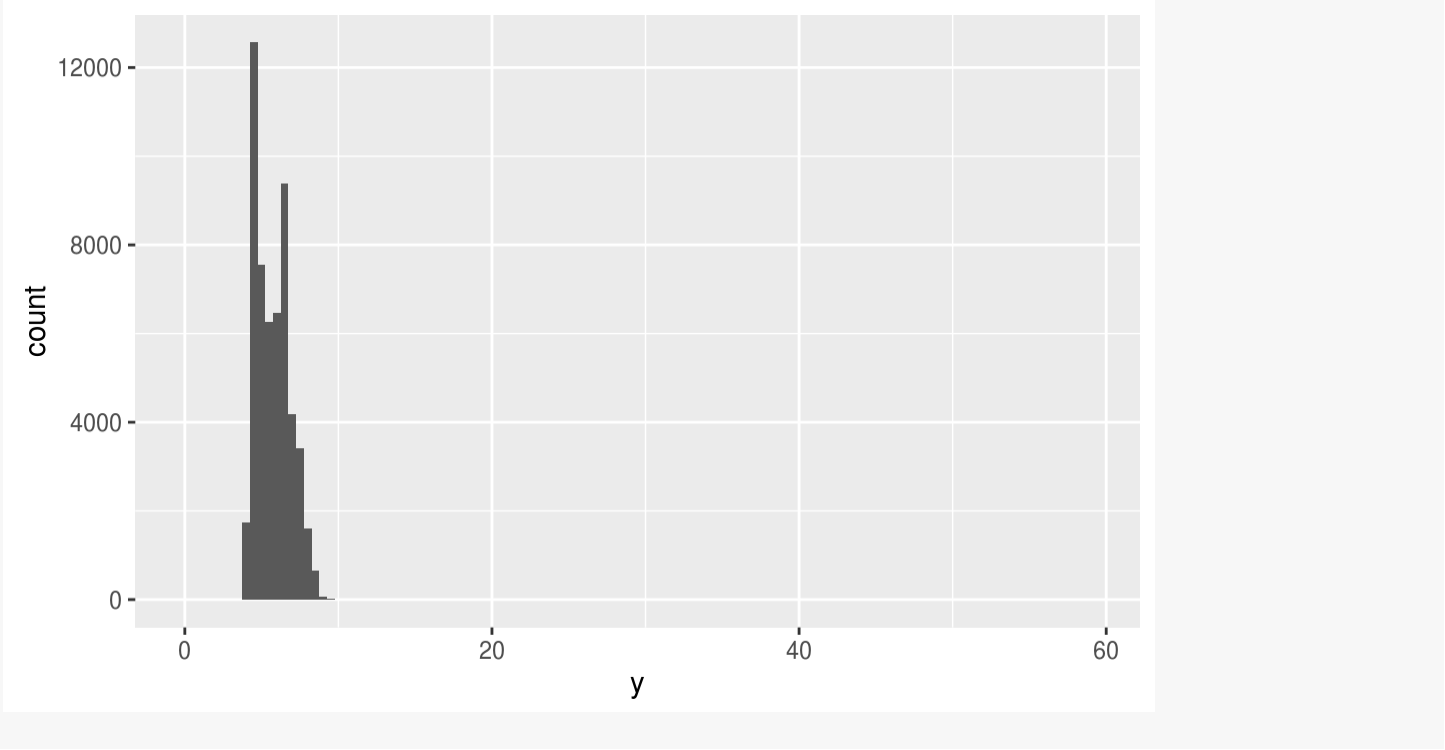


Unusual values

Outliers are observations that are unusual; data points that are don't seem to fit the pattern. Sometimes outliers are data entry errors; other times outliers suggest important new science. When you have a lot of data, outliers are sometimes difficult to see in a histogram.

For example, take this distribution of the `x` variable from the diamonds dataset. The only evidence of outliers is the unusually wide limits on the x-axis.

```
ggplot(diamonds) +
  geom_histogram(aes(x = y), binwidth = 0.5)
```



This is because there are so many observations in the common bins that the rare bins are so short that you can't see them (although maybe if you stare intently at 0 you'll spot something). To make it easy to see the unusual vaues, we need to zoom into to small values of the y-axis with coord_cartesian():

```
ggplot(diamonds) +
  geom_histogram(aes(x = y), binwidth = 0.5) +
  coord_cartesian(ylim = c(0, 50))
```

Web References:

http://r4ds.had.co.nz/exploratory-data-analysis.html

www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/

https://upgrad.com/blog/exploratory-data-analysis-and-its-importance-to-your-business/

https://www.kdnuggets.com/2018/06/7-simple-data-visualizations-should-know-r.html

https://www.analyticsvidhya.com/blog/2015/04/comprehensive-guide-data-exploration-r/

https://www.educba.com/data-exploration-in-r/

https://www.analyticsvidhya.com/.../exploratory-data-analysis-johns-hopkins-university

Questions

Expansion of EDA _____

Expansion of IDA _____

EDA is an approach to --_____ data sets to summarize their main characteristics, often with visua

_____ is the first step in data analysis and typically involves summarizing the main characteris

The _____ is the rapid increase in the amount of published information or data and the effects of thi

_____ is an approach to analyzing data sets to summarize their main characteristics, often with visual

Exploratory Data Analysis is one of the important steps in the _____

_____are the most important statistical programming languages used to perform Exploratory Data Ana

Languages come bundled with a plethora of tools that help you perform specific statistical functions.

Dimensionality Reduction techniques like PCA and LDA are performed these _____ the dimensionali

_____ techniques like PCA and LDA are performed

_____ use frequency distribution tables, bar charts, histograms, or pie charts for the graphical rep

_____ help in understanding the interactions between different data-fields.

___-means clustering

The most common way of performing predictive modeling is using _____ regression

The _____ distribution is one of the most essential tools in statistics.

_____ that help organisations incorporate Exploratory Data Analysis directly into their Business Intelli

Data connectors that help organisations incorporate Exploratory Data Analysis directly into their _____

Method to perform uni-variate analysis will depend on whether the variable type is _____.

_____Analysis finds out the relationship between two variables.

We can start analyzing the relationship by creating a _____ of count and count%.

_____method is more of a visual form of Two-way table

_____ test is used to derive the statistical significance of relationship between the variables.

In SAS, we can use _____ as an option with Proc freq to perform this test.

_____assesses whether the average of more than two groups is statistically different.

_____is possible that there are problems with extraction process.

_____errors occur at time of data collection and are harder to correct.

_____ is a case when the probability of missing variable is same for all observations.

_____ is a case when variable is missing at random and missing ratio varies for different values / level o

_____is a case when the missing values are not random and are related to the unobserved input varia

_____is a case when the probability of missing value is directly correlated with missing value itself.

In _____ deletion, we delete observations where any of the variable is missing.

In _____ deletion, we perform analysis with all cases in which the variables of interest are present.

Deletion methods are used when the nature of missing data is _____ else non random missing values o

_____ test assess whether mean of two groups are statistically different from each other or not.

In _____case, we calculate the mean or median for all non missing values of that variable then replace

In _____case, we calculate average for gender Male and Female individually of non missing values then

_____is a method to fill in the missing values with estimated ones.

_____ is one of the sophisticated method for handling missing data.

There are ____ drawbacks for prediction model approach

In _____ method of imputation, the missing values of an attribute are imputed using the given number o

k-nearest neighbour can predict _____ attributes.

There are _____ types of deletion

Data collection can be categorized in _____ types

Bi-variate Analysis finds out the relationship between _____ variables.

At univariate analysis stage, we explore variables one by _____.

_____ is a commonly used terminology by analysts and data scientists

Outlier can be of ____ types.

Outlier can be of _____ and _____.
A _____ is a quantity, quality, or property that you can measure.
A _____ is the state of a variable when you measure it.
An _____ is a set of measurements made under similar conditions.
_____ is the tendency of the values of a variable to change from measurement to measurement.
A variable is categorical if it can only have a _____ set of unique values.
Which of the following is apply function in R ?
Functions are defined using the _____ directive and are stored as R objects
_____ require you to pass a function whose argument is a vector of parameters
The _____ function is used to plot negative likelihood.
_____ loop over a list and evaluate a function on each element
lappy functions takes _____ arguments in R language.
_____ keeps track of the function call stack at regularly sampled intervals and tabulates how much time is sp

option A	option B
exploratory data analysis (EDA)	exploratory digital analysis (EDA)
initial digital analysis (IDA)	initial data analysis (IDA)
capture	delete
information exploration	data exploration
information exploration	information explosion
initial data analysis(IDA)	initial digital analysis(IDA)
data analysis process	data analytic process
S-Plus and R	R-plus and S
S-plus and R	R-plus and S
increase	reduce
Dimensionality Reduction	Dimension Reduction
Univariate visualisations	Universal visualisations
Univariate visualisations	Universal visualisations
K	S
binary	linear
Poisson	linear
data connectors	data connection
Business Intelligence system.	Business Intelligence software.
categorical	continuous
categorical	continuous
one-way table	three-way table
Stacked Row Chart	Stacked Column Chart
Stacked Row Chart	Stacked Column Chart
ChiRec	ChiTri
chi-square test	Z-test
Data collection	Data distribution
Data collection	Data distribution
Missing completely at random	Missing at random
Missing that depends on the missing value itself	Missing completely at random
Missing that depends on unobserved predictors	Missing that depends on the missing value itself
Missing at random	Missing that depends on unobserved predictors
data wise	pair wise
pair wise	list wise
Missing completely at random	Missing at random
Z-Test/ T-Test	Z-Test/ K-Test
Prediction Model	Similar case Imputation
Generalized Imputation	Similar case Imputation
Imputation	KNN Imputation
Generalized Imputation	Prediction Model
4	6
KNN Imputation	Generalized Imputation
quantitative	qualitative
1	2
2	4
3	2
one	two
Outlier	Variation
2	4

Bivariate and Multivariate.	Univariate and Bivariate.
variable	constant
constant	observation
constant	value
Variation	variable
infinite	finite
apply()	tapply()
function()	funct()
optimize()	optimise()
plot()	graph()
apply()	lapply()
two	three
summaryRprof()	Rprof()

option C
exploratory data analyst (EDA)
initial data analysts (IDA)
analyzing
data explosion
data explosion
exploratory data analysis (EDA)
information analysis process
S-plus and S
S-plus and S
equal
Dimensionality increment
Multivariate visualizations
Multivariate visualizations
R
both a&b
binary
data science
exploratory data analysis
categorical or continuous.
Univariate
four-way table
Two-way table
Two-way table
ChiCri
ANOVA
Data existing
Data existing
Missing that depends on unobserved predictors
Missing at random
Missing completely at random
Missing that depends on the missing value itself
list wise
combo wise
Missing that depends on unobserved predictors
K-Test/ T-Test
Generalized Imputation
KNN Imputation
Generalized Imputation
KNN Imputation
2
Similar case Imputation
both qualitative & quantitative
3
1
8
fourth
Prerequisites
8

Univariate and Multivariate.
identifier
value
observation
value
limited
fapply()
functions()
opt()
graph.plot()
sapply()
four
system.time()

option D	Answer
exploratory digital analysts (EDA)	exploratory data analysis (EDA)
initial digital analysts (IDA)	initial data analysis (IDA)
formatting	analyzing
information explosion	data exploration
data exploration	information explosion
exploratory digital analysis (EDA)	exploratory data analysis (EDA)
information analytic process	data analysis process
R-plus and R	S-Plus and R
R-plus and R	S-plus and R
none of these	reduce
Dimension increment	Dimensionality Reduction
Multiple visualizations	Univariate visualisations
Multiple visualizations	Multivariate visualizations
T	K
none of these	linear
binomial	Poisson
datasets	data connectors
exploratory data analysts	Business Intelligence software
none of these	categorical or continuous
Bi-variate	Bi-variate
two-way table	two-way table
Chi-Square Test	Stacked Column Chart
Chi-Square Test	Chi-Square Test
Chisq	Chisq
T-test	ANOVA
Data extraction	Data extraction
Data extraction	Data collection
Missing that depends on the missing value itself	Missing completely at random
Missing that depends on unobserved predictors	Missing at random
Missing at random	Missing that depends on unobserved predictors
Missing completely at random	Missing that depends on the missing value itself
combo wise	list wise
data wise	pair wise
Missing that depends on the missing value itself	Missing completely at random
Z-Test/ V-Test	Z-Test/ T-Test
KNN Imputation:	Generalized Imputation
Prediction Model	Similar case Imputation
Similar case Imputation	Imputation
Similar case Imputation	Prediction Model
10	2
Prediction Model	KNN Imputation
none of these	both qualitative & quantitative
4	2
3	4
10	2
half	one
variable	Outlier
16	2

none of these
variation
constant
variable
observation
unlimited
sapply()
func()
oplt()
plot.graph()
mapply()
five
prof()

Univariate and Multivariate.
variable
value
observation
Variation
finite
tapply()
function()
opt()
plot()
apply()
four
Rprof()

unit 5

s.no

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46

question
_____provide all the necessary data and the computer codes to run the analysis again, re-creating the
_____study that arrives at the same scientific findings as another study,
____measurement can be obtained with stated precision by the same team using the same measuremer
_____measurement can be ob-tained with stated precision by a different team, a different measuring sy
___Package can be used to blend the subject and single document defines the content and the analysis
_____ & _____ Packages contain alternative approaches to embedding R code into various markups.
____Can create LaTeX documents from scratch.
_____ contains a function to correctly escape special
____Creates resumes.
Standardized exams can be created using the ____ package.
____Package can process HTML files directly
____can also work with HTML by way of the R2HTML package
____can create HTML format documents from scratch.
The packages ____ & ____ have general tools for working with documents in this format.
____package can write R objects to the AsciiDoc format.
___can lead to an excessive dependence of our results on small details and a situation
_____Should be loaded at the top of the script
R supports plenty of_____ formats
Our_____ tells a story about the past.
R mark down documents are fully_____.
One of the neat tools available via a variety of packages in R is the creation of beautiful tables using data
The_____ provides for the development of a lot of interesting questions.
The _____package was written to combine elements of RMarkdown and R code within a single docume
_____runs all the bits of code in the file.
_____generates a markdown file, including bits of the original document and it's output.
_____ converts the markdown document into html.
R Markdown is the file with the file extension_____
knitr package will then transform the file into a Markdown file with the extension_____
Rstudio will load another package called markdown to transform the file into_____
_____includes a powerful and flexible system for creating dynamic reports
R includes a powerful and flexible system for creating_____ reports
_____enables the embedding of R code within LaTeX documents to generate a PDF file
system.time function returns an object of class _____ which contains two useful bits of information.
_____ time is time charged to the CPU(s) for the R expression.
The elapsed time may be _____ than the user time if your machine has multiple cores/processors
Parallel processing is done via _____ package can make the elapsed time smaller than the user time.
You can time _____ expressions by wrapping them in curly braces within the call to system.time().
The profiler can be turned off by passing _____ to Rprof().
The _____ function will first print out the function call stack when an error occurrs.
In simulating linear model can also simulate from_____ where the errors are no longer from a Nor
Simulating _____ numbers is useful but sometimes we want to simulate values that come from a specific mod
The function call stack is the _____ of functions that was called before the error occurred.
In which case the _____function tried to evaluate the formula y ~ x and realized the object y did not exist.
_____ time charged to the CPU(s) for this expression
What will be the output of the following code ? > set.seed(10)> x <- rbinom(100, 1, 0.5)> str(x)
_____ distribution is commonly used to model data that come in the form of counts.

What will be the output of the following code ? > rpois(10, 1)
Which of the following code represents count with mean of 2 ?
The _____ function draws randomly from a specified set of (scalar) objects allowing you to sample from arbitrar
_____ is an important (and big) topic for both statistics and for a variety of other areas where there is a need
Setting the _____ number generator seed via set.seed() is critical for reproducibility
The _____ function tabulates the R profiler output and calculates how much time is spend in which functior
Interactive debugging tools _____,_____,_____,_____ and _____ can be used to find problem
_____ allows you to modify the error behavior so that you can browse the function call stack
_____ suspends the execution of a function wherever it is called and puts the function in debug mode
debug() flags a function for _____ mode in R mode.
What would be the output of the following code ? > mean(x) Error in mean(x) : object 'x' not found> traceback()
The recover() function will first print out the function call stack when an _____ occurs.
_____ is a systematic way to examine how much time is spent in different parts of a program.
Which of the following is primary tool for debugging ?
_____ allows you to insert debugging code into a function a specific places
_____ evaluate the cumulative distribution function for a Normal distribution
_____ generate random Poisson variates with a given rate

option 1
Author
Replication
Replication
Reproducible
knitr
knitr & brew
brew
resumer
Hmisc
exame
knitr
Sweave
brew
markdown & knitr
ascii
complexity
Package
Static
Information
Rroducible
R studio
Data
Knitr
Knitr
Knitr
Knitr
. Rmd
. Rmd
. html
R
Static
Sweave
debug_time
elapsed
smaller
parallel
smaller
0
debug()
generalized model
arbitrary
arbitrary
debug()
sample.time
int [1:100] 1 0 0 1 0 0 0 0 1 0 ...
Gaussian

[1] 7 0 1 1 2 1 1 4 1 2
rpois(10, 2)
sam()
Simulation
arbitrary
prof()
trace, debug, browser, backtrace, and recover
debug()
debug()
debug
1: mean(x)
Error
Profiling
debug()
debug()
dnorm
dnorm

option 2
programmer
Reproducible
Reproducible
Repeatability
brew
brew & R.rsp
R.rsp
Hmisc
exams
Hmisc
Hmisc
brew
lazyweave
rmarkdown & mark
numeric
Entropy
function
Dynamic
Data
Reproducible
R
Dates
R
Knitr HTML
Knitr HTML
Knitr HTML
. md
. md
. Rmd
Rstudio
Dynamic
Knitr
proc_time
user
greater
statistics
longer
1
trace()
generalized linear model
sample
sample
trace()
user time
int [1:100] 10 0 01 1 0 0 01 0 1 0 ...
Parametric

[1] 0 8 1 1 2 1 1 4 1 2
rpois(10, 20)
seed()
samplie
sample
summaryRprof()
traceback, debug, browser, trace, and recover
trace()
trace()
run
Null
Warning
Monitoring
trace()
trace()
rnorm
rnorm

option 3
Analyzer
Repeatability
Repeatability
Replication
R.rsp
R.rsp & knitr
lazyweave
exams
resumer
resumer
exam
scratch
knitr
markdown & rmarkdown
constant
Reproducibility
procedure
Both A&B
Files
Responsible
. Net
Dataset
R studio
R studio
R studio
R studio
. Rpubs
. rd
. md
Knitr
Rstudio
Rstudio
procedure_time
response
equal to
distributed
error
2
recover()
linear model
random
random
eval()
elapsed time
int [1:100] 1 03 0 1 0 0 0 02 1 0 ...
Poisson

[1] 0 0 1 1 2 1 1 4 1 2
rpois(20, 2)
sample()
distribution
random
Rprof()
traceback, debug, browser, trace, and request
recover()
recover()
compile
0
Messages
Logging
browser()
browser()
pnorm
pnorm

option 4
Researcher
none of these
None of these
Reproducibility
lazyweave
brew & lazyweave
knitr
knitr
lazyweave
brew
lazyweave
knitr
Sweave
none of these
integer
Repeatability
characters
None of these
Forms
Response
Vb
Files
R Markdown
HTML
HTML
HTML
. rd
. Rpubs
. rd
None
None
Rpubs
proced_time
request
not equal to
equal
warning
NULL
traceback()
ungeneralized linear model
sequence
sequence
traceback()
system.time
int [1:100] 1 2 3 1 1 0 0 0 1 0 ...
Paradox

[1] 0 9 1 1 2 1 1 5 1 2
rpois(2, 20)
samp()
normal
sequence
Rpro()
traceback, debug, browser, request, and recover
traceback()
browser()
recover
1
stop
Scheduling
traceback()
traceback()
rpois
rpois

Ans
Author
Replication
Repeatability
Reproducibility
knitr
brew & R.rsp
lazyweave
Hmisc
resumer
exam
knitr
Sweave
lazyWeave
markdown & rmarkdown
ascii
Complexity
package
Both A&B
Data
Reproducible
R
Dataset
Knitr
Knitr HTML
Knitr HTML
Knitr HTML
. Rmd
. md
.html
R
Dynamic
Sweave
proc_time
elapsed
smaller
parallel
longer
NULL
recover()
generalized linear model
random
sequence
eval()
user time
int [1:100] 1 0 0 1 0 0 0 0 1 0 ...
Poisson

[1] 0 0 1 1 2 1 1 4 1 2
rpois(10, 2)
sample()
Simulation
random
summaryRprof()
traceback, debug, browser, trace, and recover
recover()
browser()
run
1: mean(x)
Error
Profiling
debug()
trace()
pnorm
rpois

Unit V: Reproducible research: Concepts and tools behind reporting modern data analysis in a reproducible manner, To write a document using R mark down, integrate live R code into a literate statistical program, compile R mark down documents using Knit r and related tools, and organize a data analysis so that it is reproducible and accessible to others.

What is Reproducible research: Authors provide all the necessary data and the computer codes to run the analysis again, re-creating the results.

What Replication: A study that arrives at the same scientific findings as another study, collecting new data (possibly with different methods) and completing new analyses.
Repeatability (Same team, same experimental setup.) The measurement can be obtained with stated precision by the same team using the same measurement procedure, the same mea- suring system, under the same operating conditions, in the same location on multiple trials.For computational experiments, this means that a researcher can reliably repeat her own.

Reproducibility (Different team, different experimental setup.) The measurement can be ob-tained with stated precision by a different team, a different measuring system, in a different location on multiple trials. For computational experiments, this means that an independent group can obtain the same result using artifacts which they develop completely independently.

Literate programming

The goal of reproducible research is to tie specific instructions to data analysis and experimental data so that scholarship can be recreated, better understood and verified. Packages in R for this purpose can be split into groups for: literate programming, package reproducibility, code/data formatting tools, format convertors, and object caching.

Literate Programming

The primary way that R facilitates reproducible research is using a document that is a combination of content and data analysis code. The Sweave function (in the base R utils package) and the knitr package can be used to blend the subject matter and R code so that a single document defines the content and the analysis. The brew and R.rsp packages contain alternative approaches to embedding R code into various markups.

The resources for literate programming are best organized by the document type/markup language:

LaTeX

Both Sweave and knitr can process LaTeX files. lazyWeave can create LaTeX documents from scratch.

Object Conversion Functions:

- *summary tables/statistics* : Hmisc, NMOF, papeR, quantreg, rapport, reporttools, sparktex, tables, xtable, ztable
- *tables/cross-tabulations* : Hmisc, lazyWeave, knitLatex, knitr, reporttools, ztable
- *graphics* : animation, Hmisc, grDevices:::pictex, sparktex, tikzDevice
- *statistical models/methods* : apsrtable, memisc, quantreg, rms, stargazer, suRtex, TeachingSampling, texreg, xtable, ztable
- *bibtex* : bibtex and RefManageR
- *others* : latex2exp converts LaTeX equations to plotmath expressions.

Miscellaneous Tools

- Hmisc contains a function to correctly escape special characters. resumer creates resumes. Standardized exams can be created using the exams package.

HTML

The knitr package can process HTML files directly. Sweave can also work with HTML by way of the R2HTML package. lazyWeave can create HTML format documents from scratch.

Object Conversion Functions:

- *summary tables/statistics* : stargazer
- *tables/cross-tabulations* : DT, formattable, htmlTable, HTMLUtils, hwriter, knitr, lazyWeave, SortableHTMLTables, texreg, ztable
- *statistical models/methods* : rapport, stargazer, xtable
- *others* : knitcitations, RefManageR

Miscellaneous Tools: htmltools has various tools for working with HTML. tufterhandout for creating Tufte-style handouts

Markdown

The knitr package can process markdown files without assistance. The packages markdown and rmarkdown have general tools for working with documents in this format. lazyWeave can create markdown format documents from scratch. Also, the ascii package can write R objects to the AsciiDoc format.

Object Conversion Functions:

- *summary tables/statistics* : papeR
- *tables/cross-tabulations* : DT, formattable, htmlTable, knitr, lazyWeave, papeR
- *statistical models/methods* : pander, papeR, rapport, texreg
- *others* : RefManageR

Miscellaneous Tools: tufterhandout for creating Tufte-style handouts. kfigr allows for figure indexing in markdown documents.

OpenDocument Format (ODF)

Object Conversion Functions:

- *statistical models/methods* : rapport

Microsoft Formats

The ReporteRs (formerly R2DOCX) package can create docx and pptx files. R2wd (windows only) can also create Word documents from scratch and R2PPT (also windows only) can create PowerPoint slides. The rtf package does the same for Rich Text Format documents.

Miscellaneous Tools

Package Reproducibility

R also has tools for ensuring that specific packages versions can be required for analyses. checkpoint, rbundler and packrat install packages required for a project to a local archive as they existed at a specified point in time. This allows specific package versions to be maintained over time and different users. The miniCRAN package facilitates the creation of local CRAN-like repositories.

Formatting Tools

formatR, highlight, highr, and SweaveListingUtils can be used to color and/or format R code.

Packages humanFormat, lubridate, prettyunits, and rprintf have functions to better format data.

Format Convertors

pander can be used for rendering R objects into Pandoc's markdown. knitr has the function pandoc that can call an installed version of Pandoc to convert documents between formats such as Markdown, HTML, LaTeX, PDF and Word. tth facilitates TeX to HTML/MathML conversions.

Object Caching Packages

When using Sweave and knitr it can be advantageous to *cache* the results of time consuming code chunks if the document will be re-processed (i.e. during debugging). knitr facilitates object caching and the Bioconductor package weaver can be used with Sweave.

Non-literate programming packages to facilitating caching/archiving are R.cache and archivist.

Reproducible research

The problems/challenges outlined in the previous section can be grouped into three categories:

1. computational resources: the need to handle large data volumes and long computation times;
2. validation: a need for new, well-validated analysis methods;
3. reproducibility: due to the workflow complexity, it can be difficult for others (or yourself in a few months time) to reproduce the analysis.

Foundations of science

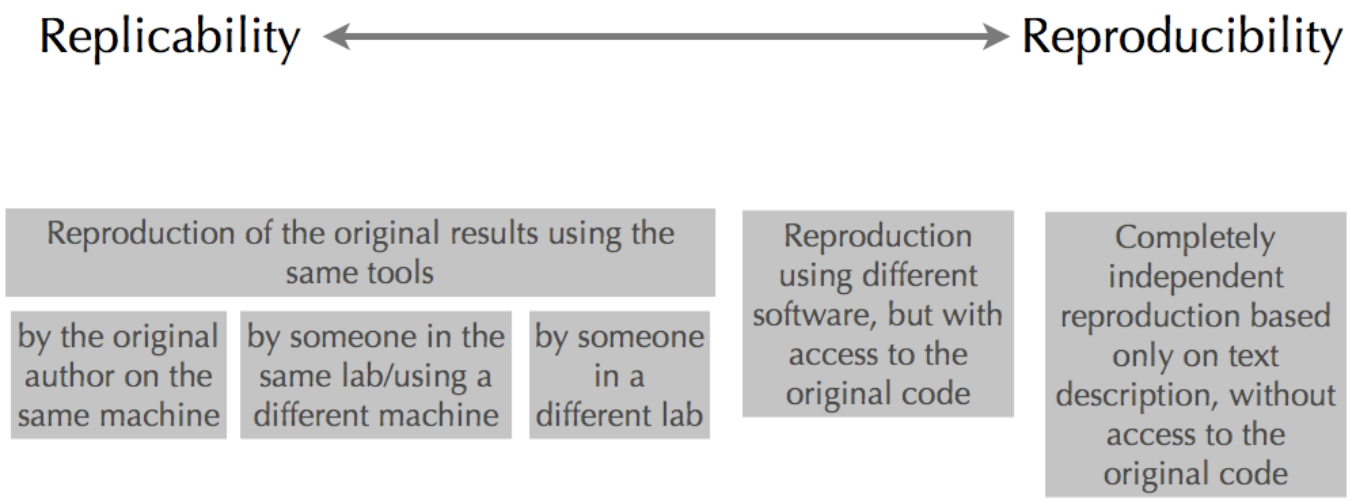
Reproducibility is one of the foundation stones of the scientific method. From this follows the obligation for scientists to include enough information in their publications to enable others to independently reproduce the finding.

But how much information is *enough* information?
There is increasing concern that for science that relies on computation (i.e. almost all of current science), traditional publishing methods do not allow enough information to be included, making independent reproduction either extremely difficult or impossible.

What is worse, in many cases scientists cannot even reproduce their own results, leading to what some are calling a *"credibility crisis"* for computation-based research.

A note on terminology: reproduction, replication and reuse

The term *"reproducibility"* covers a wide range of activities, from a completely independent repeat of the experiment, using only the "Materials and Methods" section of the article, with no access to the original authors' source code, to the original scientist re-running the same code on the same machine. Some of the other points on the spectrum are illustrated in the following figure:



Completely independent reproduction is of course the gold standard. Some commentators have expressed the opinion that time or effort spent on making other points on the spectrum (sometimes described as *"replication"* rather than *"reproduction"*) easier is either wasted, or worse, counterproductive.
This point of view tends to suppose that "mere" replication is simple, which is generally *not* the case, and ignores:

- the case where there are discrepancies between what is stated in the Material and Methods and what is in the original code;
- the case where there is not enough information in the article to reproduce the experiment;
- the considerable benefits of code reuse.

- What makes it hard to replicate your own research?

- "*I thought I used the same parameters but I'm getting different results*"
- "*I can't remember which version of the code I used to generate figure 6*"
- "*The new student wants to reuse that model I published three years ago but he can't reproduce the figures*"
- "*It worked yesterday*"
- "*Why did I do that?*"
- None of these are real quotes, but they distill a lot of similar laments I've heard from myself and colleagues. When I give talks about reproducible research I often show these quotes, and they always elicit laughter and rueful smiles of recognition, irrespective of the scientific specialty of the audience members.
- It is a common experience among computational scientists that replicating our own results, especially with the passage of time (but sometimes only hours or days later), is not always easy, and sometimes impossible.
- *Why is that?*

- I think we can identify three general classes of reasons:
- **Complexity**
- As we tackle more and more challenging problems (due to the progress of science and due to Moore's Law, which allows us to buy more and more powerful hardware), the complexity of our code and computing environment tends to increase. This can lead to an excessive dependence of our results on small details and a situation where small changes have big effects.
- **Entropy**
- both our computing environment (hardware, operating system, compilers, ...) and the versions of libraries that we use in our code change over time. Every time you upgrade your operating system or buy a new computer, the likelihood that you can replicate your old results goes down.
- **Memory limitations**
- By this I mean human, not computer memory limitations. Conscientious scientists write down the details of what they do in their lab notebooks, but often there are so many details that it's not possible to note them all, or things which seem implicit and obvious at the time are not written down and are later forgetten.

Tools for reproducible research

Git.Make and knit r

Reproducibility in R

Reproducibility in R. The statistical programming language **R** can be used to make large tasks more manageable and semi-automated and create reusable code for repeated tasks. ... This is a super desirable trait of **reproducible** code.

Steps for Reproducibility in R

How to write a reproducible example

You are most likely to get good help with your R problem if you provide a reproducible example. A reproducible example allows someone else to recreate your problem by just copying and pasting R code.

There are four things you need to include to make your example reproducible: required packages, data, code, and a description of your R environment.

- **Packages** should be loaded at the top of the script, so it's easy to see which ones the example needs.
- The easiest way to include **data** in an email is to use dput() to generate the R code to recreate it. For example, to recreate the mtcars dataset in R, I'd perform the following steps:

1. Run dput(mtcars) in R
2. Copy the output
3. In my reproducible script, type mtcars <- then paste.

- Spend a little bit of time ensuring that your **code** is easy for others to read:

- make sure you've used spaces and your variable names are concise, but informative
- use comments to indicate where your problem lies
- do your best to remove everything that is not related to the problem. The shorter your code is, the easier it is to understand.
- Include the output of sessionInfo() as a comment. This summarises your **R environment** and makes it easy to check if you're using an out-of-date package.

To write a document using R mark down
What is R mark down in R?
R Markdown. **R Markdown** is a **file** format for making dynamic documents with **R**. An **R Markdown** document is written in **markdown** (an easy-to-write plain text format) and contains chunks of embedded **R** code, like the document below. ...**Markdown** is a simple formatting syntax for authoring HTML, PDF, and MS Word documents.
What is knit r?
knitr is an engine for dynamic report generation with **R**. It is a package in the statistical programming language **R** that enables integration of **R** code into LaTeX, LyX, HTML, Markdown, AsciiDoc, and reStructuredText documents.
What are MD files?
An **MD** file is a text file created using one of several possible dialects of the Markdown language. It is saved in plain text format but includes inline text symbols that define how to format the text (e.g., bold, indentations, headers, table formatting). NOTE: Markdown files also used the .MARKDOWN **extension**.
What is sweave in R?
Sweave is a function in the statistical programming language **R** that enables integration of **R** code into LaTeX or LyX documents. The purpose is "to create dynamic reports, which can be updated automatically if data or analysis change".

Why R is important?
R is very **important** in data science because of its versatility in the field of statistics. **R** is usually used in the field of data science when the task requires special analysis of data for standalone or distributed computing
What is c() function in R?
C() function in R is used in R to create a sequence of values.
What is a vector in r?
Vector. A **vector** is a sequence of data elements of the same basic type. Members in a **vector** are officially called components. Nevertheless, we will just call them members in this site. Here is a **vector** containing three numeric values 2, 3 and 5

Latex in r
-Write a thesis or paper using mark down
-Writing academic paper with mark down

1. When wrting, we should only focus on the content, not worrying about the typesetting, which we will take care later. Word, on the other hand, allows you to see what you get when you write. This makes people (me at least) hard to ignore the typesetting when writing.
2. It is hard to update the figures and pictures inserted into the manuscript in Word. You need delete old ones and insert new ones whenever your figures are updated. Of course, you can say that do not insert figures until the submission. But wouldn't it be easier to revise the manuscript when figures are included in the main text? Using LaTex, I can just put the path of figures there and not worry about replace them in the main text.
3. Literature programming: LaTex allows us to mix code with text in the same file, which increases the reproducibility and decreases potential errors.
4. Cross-references is easy in LaTex (just \label and \ref). With Word, it is painful to get the same thing.

However, LaTex has its learning curve and quirks. And even though it intends to make people to focus on content, we usually spend lots of time fighting with things like floats. Not to mention the collaboration barrier betwen its users to Word users. When I finished a draft of my paper, I need to convert it to Word using pandoc so my advisor can edit. Doing it this way, however, figures and tables are usually messed up, as well as cross-references. Tables will be just LaTex source codes there; cross-references will be replaced with their labels (e.g. see Table tab-labels instead of see Table 1). So everytime, I need to write something like "please do not care about the typesetting" in the email to my advisor.

- Cross references
- Figures
- Tables
- Citations

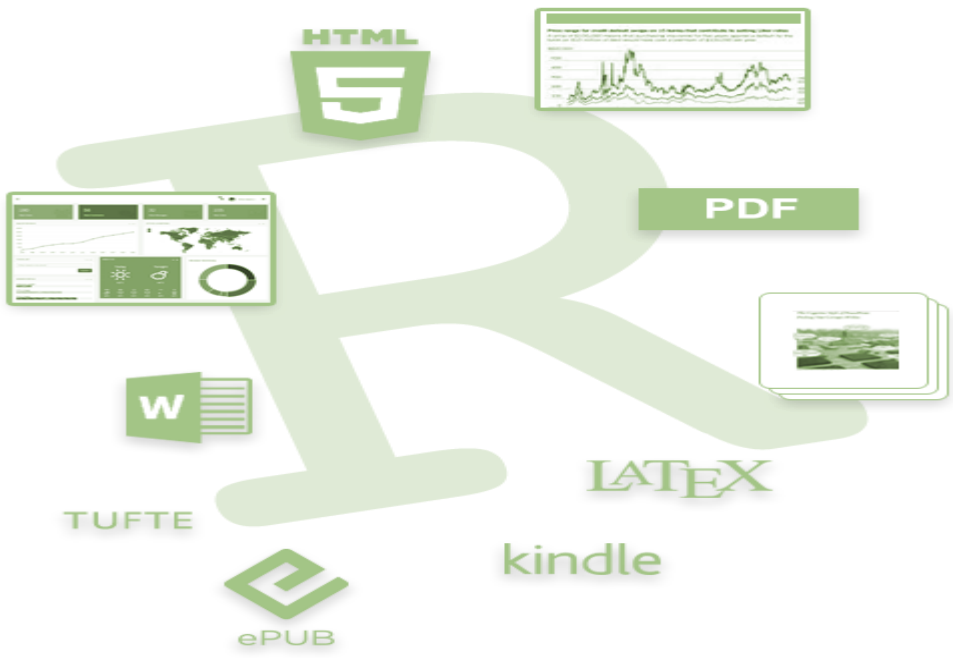
Writing with mark down:An example

```
---
title: Your awesome tile
author: "Author one and Author Two"
date: "`r format(Sys.time(), "%d %B, %Y")`"
output:
  bookdown::tufte_html2:
    number_sections: no
    toc: yes
  bookdown::word_document2: null
  bookdown::pdf_document2:
    includes:
      before_body: doc_prefix.tex
      in_header: preamble.tex
    keep_tex: yes
    latex_engine: xelatex
    number_sections: no
```

```
  toc: no
bibliography: path/to/ref.bib
fontsize: 12pt
link-citations: yes
csl: https://raw.githubusercontent.com/citation-style-language/styles/master/global-ecology-and-biogeography.csl
---
```

R mark down

Our data tells a story about the past. Narrate the story with R mark down. Turn your analyses with quality documents, presentations, and dash boards. R mark down documents are fully reproducible. Use a productive notebook interface to weave together narrative test and code to produce elegantly formatted output. Use multiple languages like R, Python and SQL. R supports plenty of static and dynamic formats including HTML, PDF, LATEX, TUFTE-STYLE handouts, Word, Beamer, books, shiny applications, dash boards, scientific articles and websites and much more.



Keywords

Kable, xtable, latex table, sweave

Creating look and feel tables using Mark down

One of the neat tools available via a variety of packages in **R** is the creation of beautiful tables using data frames stored in **R**. In what follows, I'll discuss these different options using data on departing flights from Seattle and Portland in 2014. (More information and the source code for this **R** package is available at https://github.com/ismayc/pnwflights14.)

We begin by ensuring the needed packages are installed and then load them into our **R** session.

```
# List of packages required for this analysis
pkg <- c("dplyr", "knitr", "devtools", "DT", "xtable")

# Check if packages are not installed and assign the
# names of the packages not installed to the variable new.pkg
new.pkg <- pkg[!(pkg %in% installed.packages())]

# If there are any packages in the list that aren't installed,
# install them
if (length(new.pkg)) {
  install.packages(new.pkg, repos = "http://cran.rstudio.com")
}

# Load the packages into R
library(dplyr)
library(knitr)
library(DT)
library(xtable)

# Install Chester's pnwflights14 package (if not already)
if (!require(pnwflights14)){
  library(devtools)
  devtools::install_github("ismayc/pnwflights14")
  }
library(pnwflights14)

# Load the flights dataset
data("flights", package = "pnwflights14")
```

The dataset provides for the development of a lot of interesting questions. Here I will delve further into some of the questions I addressed in two recent workshops I led in the Fall 2015 Data @ Reed Research Skills Workshop Series. (Slides available at http://rpubs.com/cismay.)

The questions I will analyze by creating tables are

1. Which destinations had the worst arrival delays (on average) from the two PNW airports?
2. How does the maximum departure delay vary by month for each of the two airports?
3. How many flights departed for each airline from each of the airports?

The kable function in the knitr package

To address the first question, we will use the dplyr package written by Hadley Wickham as below. We'll use the top_n function to isolate the 5 worst mean arrival delays.

```
worst_arr_delays <- flights %>% group_by(dest) %>%
  summarize(mean_arr_delay = mean(arr_delay, na.rm = TRUE)) %>%
  arrange(desc(mean_arr_delay)) %>%
  top_n(n = 5, wt = mean_arr_delay)
```

This information is helpful but you may not necessarily know to which airport each of these FAA airport codes refers. One of the other data sets included in the pnwflights14 package is airports that lists the names. Here we will do a match to identify the names of these airports using the inner_join function in dplyr.

```
data("airports", package = "pnwflights14")
joined_worst <- inner_join(worst_arr_delays, airports, by = c("dest" = "faa")) %>%
  select(name, dest, mean_arr_delay) %>%
  rename("Airport Name" = name, "Airport Code" = dest, "Mean Arrival Delay" = mean_arr_delay)
```

Lastly we output this table cleanly using the kable function.

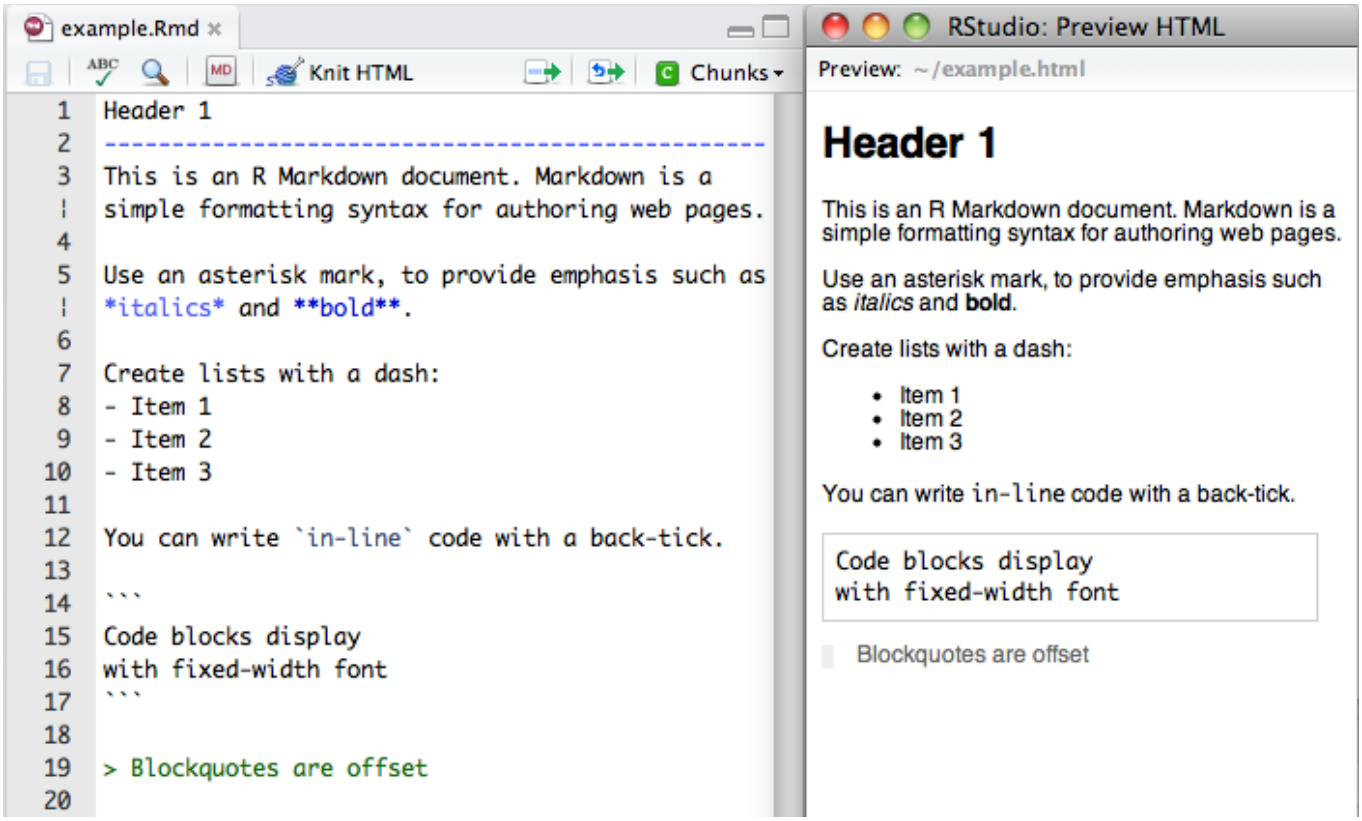
```
kable(joined_worst)
```

Writing reproducible scientific reports using knit r and pandoc tools

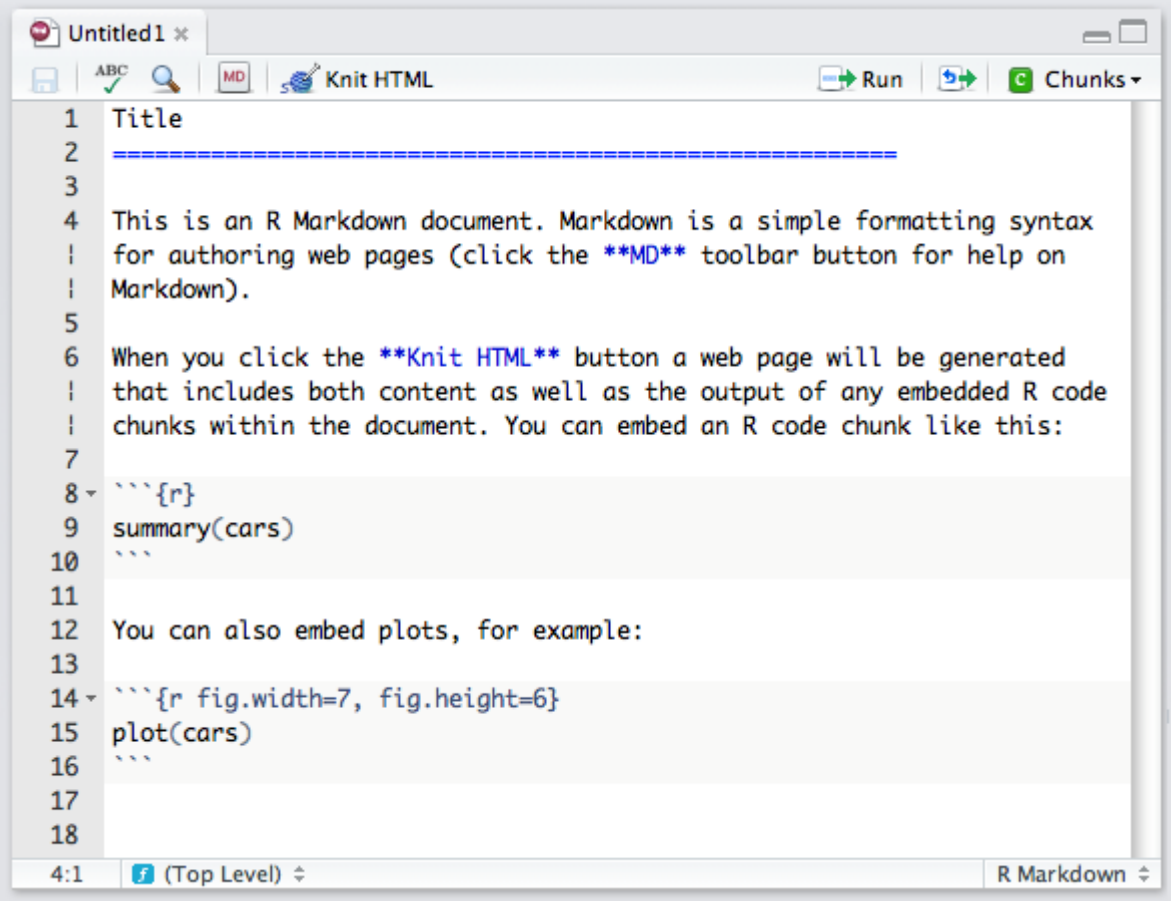
Report with knit r
The knitr package was written to combine elements of RMarkdown and R code within a single document. The best way to realise the strength of knitr is to start with an example.
Open up Rstudio and install the knitr package

```
install.packages(knitr)
```

Then open this demo file and click the knit HTML button



This file is written in RMarkdown and includes bits of text and code. The code bits are the "chunks" surrounded tick marks.



Clicking on knit HTML does several things

1. It runs all the bits of code in the file
2. It generates a markdown file, including bits of the original document and it's output.
3. It converts the markdown document into html.

You can also make the document from the console, with the following set of commands:

```
library(knitr)
library(markdown)
knit("example.Rmd")  # produces the md file

markdownToHTML("example.md", "example.html")  # converts an md file to html
```

Note, for this code to work, the example file needs to be in your working directory, or you need to provide the path to the RMD file:

```
knit("myPath/example.Rmd")  # produces the md file
```

OK, so you have a document (html file), where you can document your analyses. Now just replace the example code with some real material and away you go.

Some benefits of this approach include:

- no copying and pasting
- your report can be easily updated, once you have more data, new ideas etc
- because they are just like any other code, you can track your knitr scripts under version control.
- if it is important, you can show bits of the code used to generate the results.
- your analysis is fully transparent and reproducible.
  People now use knitr for all sorts of things, e.g.

- writing reports of their data (here's one by Rich as Rmd and html)
- preparing tutorials
- writing blog posts.

Create dynamic R statistical report using mark down
In this document, we will introduce how to create dynamic R statistical reports in HTML.

- What do you mean by *dynamic*? The traditional way to write a report is save your analysis from R first, i.e. saving the ANOVA table or using pdf() to save the graphs. A report written in this way could be problematic. For instance, imagine your client telling you that they want to use a sub-sample instead of the entire sample. You would have to redo all of your work!! Therefore, in this way dynamic also means reproducible, in the sense that people who get the file from you can reproduce the entire work in the report.
- How does R Markdown work out to be .html file? R Markdown is the file with the file extension .Rmd, the *knitr* package will then transform the file into a Markdown file with the extension .md. Then Rstudio will load another package called *markdown* to transform the file into .html.
- Is this a popular method for creating reports? Check out Rpubs. This website shares lots of documents written in the way we will introduce below.
  Let's take this in four steps:

1. Getting the Software Ready
2. Creating a Simple Report
3. Customizing your CSS to Make Your Report Pretty
4. Become a Ninja!

Using Sweave & Knit r

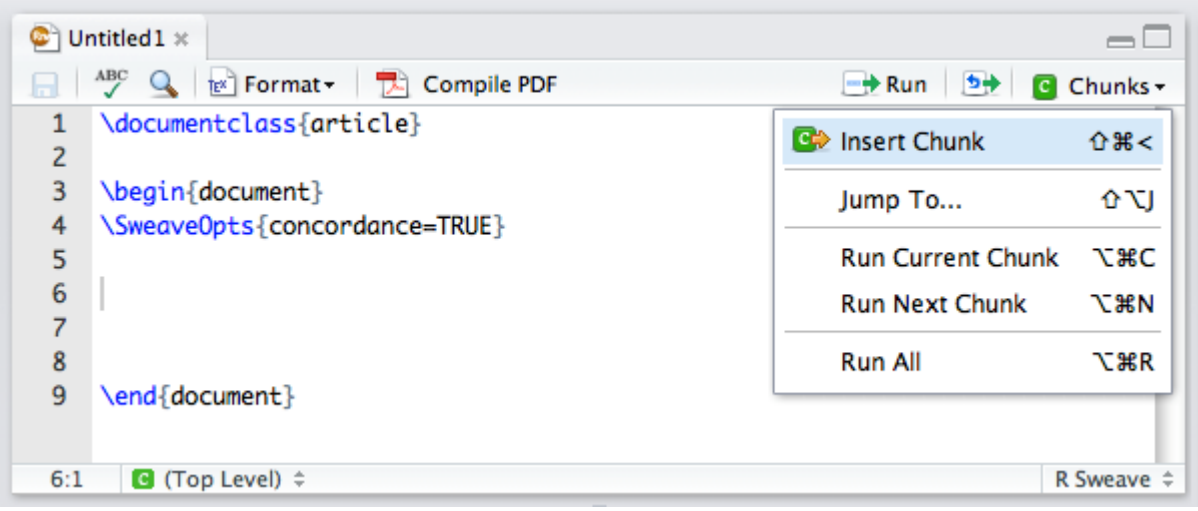
R includes a powerful and flexible system (Sweave) for creating dynamic reports and reproducible research using LaTeX. Sweave enables the embedding of R code within LaTeX documents to generate a PDF file that includes narrative and analysis, graphics, code, and the results of computations.

knitr is an R package that adds many new capabilities to Sweave and is also fully supported by RStudio.

To use Sweave and knitr to create PDF reports, you will need to have LaTeX installed on your system. LaTeX can be installed following the directions on the LaTeX project page.

Working with sweave and knit r

Creating a New Document



To start a new Sweave document, go to **File | New** and select "R Sweave". This will provide a basic Sweave template. From here, you can enter text and LaTeX commands. R chunks can also be inserted to interweave R commands and output into your document. To insert an R chunk, use the **Chunks** menu at the top right of the source editor. You can compile the Sweave document into a PDF using the **Compile PDF** button on the toolbar. Note that for reproducibility purposes, this compile runs in a separate process and environment (rather than using the current workspace). This is to ensure the script will always produce the same result and not be effected by (or pollute) your regular R environment.

Using knitr

You can use knitr rather than classic Sweave for weaving Rnw files. See the Weaving Rnw Files documentation for additional details on how to enable the use of knitr on a global, per-project, or per-file basis.

References:

Video:https://www.coursera.org/lecture/data-scientists-tools/reproducible-research-overview-NAudh

https://www.coursera.org/learn/reproducible-research

rrcns.readthedocs.io/en/latest/reproducible_research.html

https://www.practicereproducibleresearch.org/core-chapters/4-casestudies.html

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University Established Under Section 3 of UGC Act 1956)
Pollachi Main Road, Coimbatore-21

Class: III B.Sc(CT)
Subject Code:16CTU503A Introduction to Data Science Batch(2016-2019)

Register_____

(16CTU503A)

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University Established Under Section 3 of UGC Act)

Coimbatore – 641 021

B.Sc DEGREE EXAMINATION

FIRST INTERNAL EXAMINATION – JULY 2018

FIFTH SEMESTER

COMPUTER TECHNOLOGY

INTRODUCTION TO DATA SCIENCE

Class : III B.Sc(CT)

Subject Code: 16CTU503A Duration: 2 Hours

Date & Session : Max.Marks: 50

PART A (20 × 1 = 20 Marks)

1. Knowledge workers in any organization collectively termed as _____
   a) MIS b)ESS C)OLAP d)**DSS**
2. Organizations are capturing, storing and analyzing data that has high volume, high velocity, and variety of _____ comes from variety of sources like social media, log files, text, image etc.,
   a)information b)knowledge c)**data** d)business value
3. Business value from big data is great. Though, some companies have significant privacy Concerns. What are they?
   a)Hindustan Lever and Proctor and Gamble b) **Google and Face book**
   c) Amazon and Verizone d) none of the above
4. I TB of data is equivalent to
   a) 1000 PB b)1024 MB c) 1024 YB d)**none of the above**
5. 1 million web pages are equivalent to
   a) 1 lakhs b)100 lakhs c)**10 lakhs** d)1 crore
6. _____ is all about diving deep at a granular level to mine and understand complex behaviors, trends, and inferences .
   a)big data b)DSS c)**Data science** d)Business Intelligence
7. _____ identifies major customer segments and unique shopping behaviors within the segment.
   a)Netflix b)**Target** c)Spotify d)spam filter
8. A _____ is called as recommendation engine in data science
   a) data insight b)**data product** c)Google d)none of the above
9. _____ used for self driving cars is also a data product able to sense traffic lights, pedestrians, and other obstacles on road
   a)Netflix b)big data c)**computer vision** d)none of the above
10. _____ is a software that helps software developers to work together and maintain a complete history of their work.
    a) Data insight b)**version control system** c)pentaho d)rapid miner
11. _____ work at the raw database level to derive insights and build data product.
    a)analyst b)**data scientist** c)algorithm d)none of the above
12. Extending current internet and providing connection and communication between physical objects and devices called
    a)WoT b)**IoT** c)knowledge d)none of the above
13. The _____ software library is a big data framework, allows distributed processing of large datasets across clusters of computers.
    a)Hadoop b)**Apache Hadoop** c)weka d)Tableau

14. _____i s a light weight markup language with plain text formatting syntax.
a) knit r b)**Mark down** c)R language d)none of the above

15. _____ is a version control system for tracking changes in computer files and coordinating work on those files among multiple people.
a)GitHub b)**Git** c)mark down d)knit r

16. The _____ is widely used among statisticians and data miners for developing statistical software and data analysis.
a)knit r b)mark down c)**R language** d)none of the above

17. A tool to manage your source code history is called
a)**Git** b)GitHub c)knitr d)none of the above

18. Modulo division in R is computed using the symbol
a) % b)%% c)**%/%** d)%&%

19. Which one is not an assignment operator in R
a)<- b)= c)assign() d)**equalto()**

20. Which command in R used to search for help pages containing word "plotting"
a)**?"plotting"** b)??"plotting" c)??plotting d)?plotting

PART B (3 × 2 =6 Marks)

21. What is data insight?

22. What is big data?

23. Distinguish between R and R studio.

PART C (3 ×8 =24 Marks)

24.a) Explain the history of data science.

(or)

b)Explain categories of analysis in big data analytics.

25. a)Write in detail about control structures in R.

(or)

b)Explain about dates and times functions in R

26. a)Write a R program to sum up numbers from 1 to n. Accept 'n' as input.

(or)

b)Discuss the data types in R each with an example.

Part A

1.DSS
2.DATA
3.GOOGLE AND FACEBOOK
4.NONE OF THE ABOVE
5. 10 LAKHS
6.DATA SCIENCE
7.TARGET
8.DATA PRODUCT
9.COMPUTER VISION
10.VERSION CONTROL SYSTEM
11.DATA SCIENTIST
12.IOT
13.APACHE HADOOP
14.MARK DOWN
15,GIT
16.R language
17.Git
18.%/%
19.equalto
20.?"plotting"

Part B

21. What is an insight?

Insight is the value obtained through the use of analytics. The insights gained through analytics are incredible powerful, and can be used to grow your business while identifying areas of opportunity.

22. What is Big data?

Big Data is defined as high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation.

23. Distinguish between R and R studio.

Both work together. **R** is a programming language for statistical calculation. And **RStudio** is an Integrated Development Environment (IDE) that helps you develop programs in **R**. You can use **R** without using **RStudio**, but you can't use**RStudio** without using **R**, so **R** comes first.

Part C

24. a)Explain the history of Data science.

DSS->OLAP->DATA MINING->MINING OF WEB DATA->BIG DATA ANALYTICS->DATA SCIENCE

Evolution of data science initially started with Decision support system in organizations. With the advent of database technology and client./server implementation, corporate data was well maintained in repositories. To retrieve potential result,SQL queries were used to retrieve relevant information or manipulated result directly from the database. Later, an advanced form of drilling down and drilling down multi-dimensional data were done using online analytical processing[OLAP].

The concept of data mining were widely used in organization data using association rules, correlation analysis on mined data to arrive inference rules for improving business scenario. The widely used association rule mining example is Market Basket Analysis, to promote cross selling in super markets ultimately to generate more revenue using hidden patterns mined on large databases.

The advent of internet has enormous web servers world wide and internet contains adequate information in the form of text, audio and video contents invariably in all disciplines. The mobility of data through smart phones, laptop, palmtop etc., has further increased hosting of new web servers and accessing by individual and entity.

Web data mining is sub-divided in to web content mining, web usage mining, and web structure mining.Volume of data being increased in terms of terabytes, where this conventional methods was not sufficient to mine useful patterns. Researchers came out with a concept called big data and big data analytics. **Introduction to Bigdata** – Organizations are capturing storing and analyzing data that has high volume, velocity, and variety of data comes from variety of sources including social media, machines, logfiles, video, text, image, RFID and GPS. These sources have strained the capabilities of traditional relational database management systems and spawned a host of new technologies, approaches, and platforms. The potential value of big data analytics is great and is clearly established by a growing number of studies. There are keys to success with big data analytics, including clear business speed, strong committed sponsorship, alignment between the business and IT strategies, a fact-based decision making culture, a strong data infrastructure, the right analytical tools and people skilled in the use of analytics. Because of the paradigm shift in the kind of data being analyzed and how this data is used as a 4^{th} generation decision support system. Though the business value from big data is great, especially for online companies like Google and Facebook, how it is being used is raising significant privacy concerns.

Big data and analytics are intertwined, but analytic techniques such as regression analysis, simulation, and machine learning has been used in DSS. The value in analyzing unstructured data such as email and documents are well understood. The value addition is the advances in technology and software, new sources of data(e.g social media) and business opportunity. It is even spawning a new area of practice and study called “data science” that encompasses the techniques, tools, technologies and processes for making sense out of big data.

Big data is changing existing jobs and creating new ones. For example, market researchers must now be skilled in social media analytics. Data management professionals must be able to store massive amount of data of any structure. The job of data scientist the" high priest" of big data analytics has emerged. Many companies are seeking people with big data skills, many universities, are offering new courses, certificates and degree programs to provide students with the needed skills. Vendors like IBM provide education to faculty and students through their university support programs. At a high level, the requirements for organizational success with bug data analytics are the same as business intelligence [BI], at a deeper level(micro level) there are many nuances to be considered for a cost effective decision making. Governments and companies are able to integrate personal data from numerous sources and learn much of what you do, where you go, who your friends are, and what are your preferences. This leads to better services which in turn making profit, it also raises privacy concerns. There are legal restrictions on what big data companies such as Facebook, and Google can do with the data they collect. From an evolutionary perspective bigdata is not new, A major reason for creating data warehouses in 1990's was to store large amount of data. Later, terabyte of data was considered as bigdata. Teradata a company has more than 35 customers such as Walmart, Verizon with data warehouses over a petabyte in size. eBay captures a terabytes of data per minute and maintains over 40 petabytes, most of any company in the world. Some people consider 10 terabytes to be bigdata, but any numerical definition is likely to change over time as organizations collect, store and analyse more data. Applications need to bring incremental value to businesses. Big Data Analytics is all about processing unstructured information from call logs, mobile-banking transactions, online user generated content such as blog posts and tweets, online searches, and images which can be transformed into valuable business information using computational techniques to unveil trends and patterns between datasets.

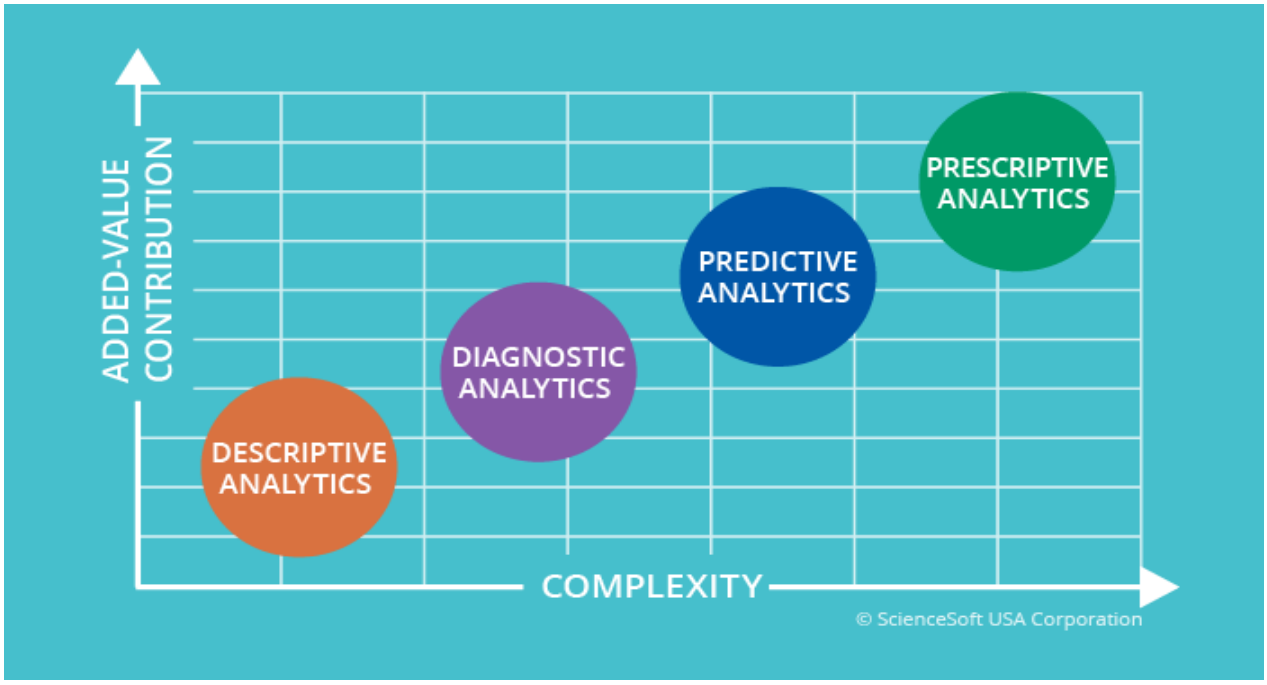
Data Science: Data Science is an extension of big data with an objective to discover potentially useful, previous unknown hidden patterns and correlations to support decision making. Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from data in various forms, both structured and unstructured similar to data mining.

Data science is a "concept to unify statistics, data analysis, machine learning and their related methods" in order to "understand and analyze actual phenomena" with data. It employs techniques and theories drawn from many fields within the context of mathematics, statistics, information science, and computer science.

Data science is a multidisciplinary blend of data inference, algorithm development, and technology in order to solve analytically complex problems. At the core is data. Troves of raw information, streaming in and stored in enterprise data warehouses by mining it. Advanced capabilities we can build with it. Data science is ultimately about using this data in creative ways to generate business value.

24 b) Categories of Analysis:

There are many **types of data analysis**. Some of them are more basic in nature, such as descriptive, exploratory, inferential, predictive, and causal. Some, however, are more specific, such as qualitative **analysis**, which looks for things like patterns and colors, and quantitative **analysis**, which focuses on numbers.



Descriptive Analysis - Descriptive analytics answers the question of *what happened*. For instance, a healthcare provider will learn how many patients were hospitalized last month; a retailer – the average weekly sales volume; a manufacturer – a rate of the products returned for a past month, etc. Let us also bring an example from our BI consulting practice: a manufacturer was able to decide on focus product categories based on the analysis of revenue, monthly revenue per product group, income by product group, total quality of metal parts produced per month.
Descriptive analytics juggles raw data from multiple data sources to give valuable insights into the past. However, these findings simply signal that something is wrong or right, without explaining why. For this reason, highly data-driven companies do not content themselves with descriptive analytics only, and prefer combining it with other types of data analytics.

Diagnostic Analysis - At this stage, historical data can be measured against other data to answer the question of *why something happened*. Thanks to diagnostic analytics, there is a possibility to drill down, to find out dependencies and to identify patterns. Companies go for diagnostic analytics, as it gives a deep insight into a particular problem. At the same time, a company should have detailed information at their disposal, otherwise data collection may turn out to be individual for every issue and time-consuming.
Let's take another look at the examples from different industries: a healthcare provider compares patients' response to a promotional campaign in different regions; a retailer drills the sales down to subcategories. Another flashback to our BI projects: in the healthcare industry, customer segmentation coupled with several filters applied (like diagnoses and prescribed medications) allowed measuring the risk of hospitalization.

Predictive Analysis - Predictive analytics tells *what is likely to happen*. It uses the findings of descriptive and diagnostic analytics to detect tendencies, clusters and exceptions, and to predict future trends, which makes it a valuable tool for forecasting. Despite numerous advantages that predictive analytics brings, it is essential to understand that forecasting is just an estimate, the accuracy of which highly depends on data quality and stability of the situation, so it requires a careful treatment and continuous optimization.

Thanks to predictive analytics and the proactive approach it enables, a telecom company, for instance, can identify the subscribers who are most likely to reduce their spend, and trigger targeted marketing activities to remediate; a management team can weigh the risks of investing in their company's expansion based on cash flow analysis and forecasting. One of our case studies describes how advanced data analytics services allowed a leading FMCG company to predict what they could expect after changing brand positioning.

Prescriptive Analysis - The purpose of prescriptive analytics is to literally prescribe *what action to take* to eliminate a future problem or take full advantage of a promising trend. An example of prescriptive analytics from our project portfolio: a multinational company was able to identify opportunities for repeat purchases based on customer analytics and sales history.

This state-of-the-art type of data analytics requires not only historical data, but also external information due to the nature of statistical algorithms. Besides, prescriptive analytics uses sophisticated tools and technologies, like machine learning, business rules and algorithms, which makes it sophisticated to implement and manage. That is why, before deciding to adopt prescriptive analytics, a company should compare required efforts vs. an expected added value.With various types of analytics, companies are free to choose how deep they need to dive in data analysis to satisfy their business needs best. While descriptive and diagnostic analytics offers a reactive approach, predictive and prescriptive analytics makes users proactive approach.

25 a) control structures in R

1) Usage of if-elseif statements with an example
2) For statement with an example
3) While looping statement with an example
4) Repeat … until statement with an example
5) Break and next statement with an example

25 b) Dates and Times functions in R

<<refer R programming by Sudhamathy and Jothi venkateswaran

26 a)

```
#summing numbers from 1 to n
N=as.integer(readline(prompt("Enter n\n"))
Sum=0
For(I in 1:10)
{
   Sum=sum+i
```

}

Cat(“The sum is\n”,sum)

26 b) Data types in R each with an example explanation

Integer -

Numeric – all real data with single and double precision are of numeric datatype

Character – “a”,”b”,”c” same as in c language

Complex - Complex number like 3+4i

Logical – TRUE or FALSE

.