City University London

MSc Data Science
Project Report
February 2017

Predictive Descriptive Statistics

Author: Suzanne Fox
Supervisor: Catagay Turkay
Submitted: 28th February 2017

Declaration

By submitting this work, I declare that this work is entirely my own except those parts duly identified and referenced in my submission. It complies with any specified word limits and the requirements and regulations detailed in the assessment instructions and any other relevant programme and module documentation. In submitting this work, I acknowledge that I have read and understood the regulations and code regarding academic misconduct, including that relating to plagiarism, as specified in the Programme Handbook. I also acknowledge that this work will be subject to a variety of checks for academic misconduct.

Signed:

Suzanne Fox

Abstract

Analysts and data scientists traditionally use Exploratory Data Analysis (EDA) when starting work with a new dataset to understand its shape and characteristics. Data type of each variable determines appropriate statistics and visualisations. Each dataset being different, custom scripts must be written or ad hoc commands entered interactively to explore the data. The process is time consuming, repetitive and error prone.

This project proposes a new approach which reduces human scripting work, freeing the analyst for more intellectual tasks. The report demonstrates a method of creating meta data for a structured dataset by extracting features about the data and constituent variables. A deterministic model uses the meta data to classify data type of variables, and the classification is used to prescribe descriptive statistics for each variable. Statistics are automatically computed and added to the meta data. Finally, a data report customized on the fly for the source dataset is generated. The analyst has a structured entry point for their exploration without the need for script writing.

Irrespective of source dataset, the meta data has a standard structure which is easy to learn and manipulate, and the report gives examples of how this approach offers a powerful addition to the analytical toolbox.

Keywords : Exploratory Data Analysis, Descriptive Statistics, Reproducible Research, Automation, Meta Data

Table of Contents

Chapter 1 - Introduction

1.1 Background

The starting point for any data analysis is to understand the shape and characteristics of the dataset being analysed. Summary statistics and visualisations described by Tukey (1977) in his book "Exploratory Data Analysis" are the foundation of those still used today in the early stages of the data science process. Evidence for the importance of exploratory data analysis (EDA) as defined by Tukey can be found in the large number of citations recorded by Google scholar (15279 as of 17/2/17), and the continuing relevance of his work by the number of new citations (970) added since 2016.

Tukey emphasises the importance of visual displays of the data as part of the EDA process, he states *"there cannot be too much emphasis on our need to see behavior"*. Tukey's recommended tools for creating visualisations were graph paper, tracing paper and sharp pencils; today an analyst or data scientist can create visualisations and perform EDA using software such as Python, R, MATLAB, SPSS or similar depending on their background and skills. Even with modern processing power, this stage is a time-consuming one that needs to be repeated anew for each analysis. As all datasets are different, custom scripts must be written or interactive commands issued to report statistics and create visual outputs that Tukey might find comfortably familiar. As levels of skill and experience vary analysts might miss important steps or not understand the range of statistics and visualisations that could be used.

This research proposes that if the data structure of variables in a dataset could be determined programmatically, then a dataset could be considered in the abstract. It would then be possible to write software to generate a report of appropriate descriptive statistics and visualisations prescribed by the contents of the dataset, without the need for human input.

Framed as a research question -

> *Can the data type of variables in a structured dataset be auto detected well enough to allow meaningful descriptive statistics to be prescribed?*

1.2 Purpose, Product and Beneficiaries

The purpose of the project is to test the hypothesis by designing and building a software prototype which takes as input a dataset and produces as output a customised report specific to the dataset.

Products of the work will be –

- a taxonomy of data types for variables
- the software functions to extract features from the data and classify the variables
- the meta data of features, classifications and statistics
- the software to take the dataset, apply the functions and create the output report

Beneficiaries of the project are data scientists and analysts at any level of expertise. Inexperienced analysts would be able to develop their knowledge from working with a more sophisticated report than they might be able to create with a script, and for the experienced it would offer savings in time and effort. The potentially large user base is the reason the project was chosen.

Specific benefits will be -

- human efficiencies from reduced scripting
- standardisation of initial EDA across datasets
- a reproducible report
- standardised meta data for any dataset

1.3 Objectives

The project can be broken into a series of logical steps, each of which informs the next. This leads to the following series of objectives for the project -

- create a taxonomy of data types
- design and develop a software prototype which extracts features and classifies dataset variables according to the taxonomy
- save the output as meta data for consumption by other processes
- determine descriptive statistics and visualisations appropriate to data types
- create a tool which produces an EDA report, using meta data and source data as input parameters

1.4 Methods

A literature search was performed to build the taxonomy of data types. Sources included books and papers but also data formatting options in software such as Excel.

A throw-away prototyping software engineering paradigm was followed. A sandbox environment was used to test various methods of extracting features about variables from datasets. Methods of classification modelling were explored. Output file formats for the meta data were experimented with, and the functions and objects that would be needed in the final software were planned. Choice of software platform in which to develop the working system was made.

The outcome of this stage led to choosing R as the development environment. Following the data science tradition of animal-themed software products, a name was chosen for the R library to be developed. The software seeks to dig into the information about a dataset much like a mole digs into the ground, exposing soil behind it. Moles are blind and follow their noses for direction; when first presented with a dataset no information is visible and an iterative process must be followed to see where it leads. Hence, the software development was called "***Burrow***", the meta data produced by it will be referred to as ***"Burrow data"***, and the automatically generated report as a ***"Burrow report"***.

Regex pattern matching was used to extract feature information about data variables, and a rules-based classification model was developed.

An R markdown document was created which uses Burrow data to create a custom report for the input dataset. The R ggplot2 library was used to develop visualisations.

1.5 Metrics for Success

Identifying the data type of variables in a dataset was framed as a standard data science classification problem, and the standard technique of measuring classification accuracy when judged against ground truth was used to judge success. Testing datasets drawn from a variety of domains and sources were hand classified and then shown to the classification model.

Speed of processing for various sizes of dataset was considered. To be viable for further development the Burrow function needed to run in a practical amount of time.

To test the ease of use of the software implementation and the usefulness of the Burrow markdown report a small user survey was planned.

1.6 Work plan

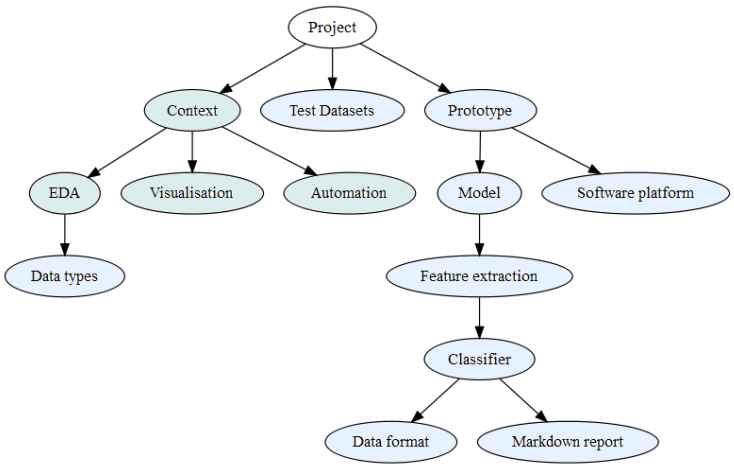


Figure 1. Work plan

The work was divided as shown in Figure 1 into research (shown in green) and software development (shown in blue). In the initial stages research and programming could be done concurrently. Researching the taxonomy was the starting point for the academic research, and software engineering decisions around feature extraction methods, classification methods and file formats were made at the same time.

The project was interrupted midway by unforeseen personal circumstances which halted work almost entirely for 3 months, and a deadline extension was applied for. This led to the effective time span of the project being shortened slightly. Although enough work was completed to establish proof of concept some later steps were not completed as fully as had been planned, and the usability survey was removed from the scope of work.

1.7 Structure of the report

Chapter 2 gives the context for the work, and addresses the areas shown in green on the work plan in Figure 1. The background of EDA is explored more fully, and it is placed in context of the data science pipeline.

Chapter 3 describes the detail of the methods that were used, shown in blue on Figure 1, including those which were assessed and then rejected.

Chapter 4 gives the results of the work both in quantitative terms as measures of classification accuracy and running efficiency, and qualitative terms showing visualisations created. This chapter also explores some unexpected results of the development.

Chapter 5 discusses the results and compares them to the initial objectives.

Chapter 6 evaluates the project, reflecting on what has been learned and looks at future work and ways in which the techniques developed could be practically implemented.

Finally supporting references referred to in the report are provided.

A separate document submitted with the report gives Appendixes -

Appendix A – Original proposal

Appendix B – Taxonomy of data types

Appendix C – Synthetic dataset used for development

Appendix D – Burrow output format documentation

Appendix E – Test datasets

Appendix F – Markdown output

Appendix G – Code listings

Appendix H – User Instructions

Two copies of the code with example data are submitted on USB.

Chapter 2 – Context

2.1 Introduction

EDA is well described by Schutt and O'Neil (2013) in the following paragraph –

> *" .. as much as EDA is a set of tools, it's also a mindset. And that mindset is about your relationship with the data. You want to understand the data – gain intuition, understand the shape of it, and try to connect your understanding of the process that generated the data to the data itself."*

This chapter documents the historical context of EDA, and places it in the wider analytical framework. The impact of fast computing on the techniques and methods used to perform EDA since the days of Tukey's classical text are discussed, and the Schutt and O'Neil quote is explored in this context. This leads to a consideration of current times and the rise of the term *citizen data scientist* to describe those who are using the same software tools as professional data scientists but who have far less training or are entirely self-taught. The most widely used data science software platforms are mentioned, showing how open source software has contributed to citizen data science, and as background to the choice of software platform for the project. Best practice in terms of analytical reproducibility is briefly discussed as the project hopes to contribute in this area. Finally, the downstream process which are informed by EDA are considered and the research being done in the field of automating data science summarised.

2.2 CRISP-DM

In 1996 a consortium of employees from NCR Systems, DaimlerChrysler AG, and SPSS was formed to address the lack of formal process models available for the (then fledgling) industry of Data Mining. The result of their work - Chapman et al. (2000) - was a non-proprietary methodology which they named the Cross Industry Standard Process for Data Mining (CRISP-DM).

The Society of Data Miners is still an active one, and Khabaza (2010) states "*Data mining is the creation of new knowledge in natural or artificial form, by using business knowledge to discover and interpret patterns in data*". Clearly there is much overlap with the more modern field of data science. An online article by Microsoft MVP and practising data scientist Steph Locke - Locke (2017) – argues that CRISP-DM is still a valuable and relevant methodology for the data science process.

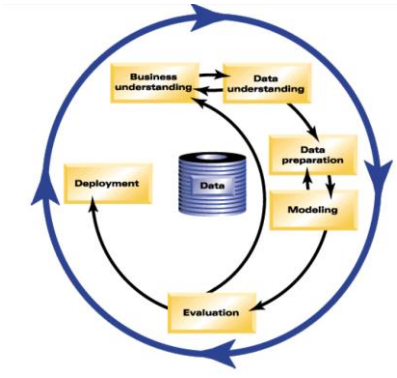


Figure 2. The CRSIP-DM process model
Source : Chapman et al. (2000)

EDA fits into the second step of the CRISP-DM model, headed "Data Understanding", and Figure 2 shows how CRISP-DM places EDA within the context of a wider analytical framework.

In the diagram, the two-way arrows between the terms "Business understanding" and "Data understanding" illustrate the point made by Schutt and O'Neil about the importance of understanding the relationship of the data to the process that generated it.

The later stages of the CRISP-DM model – Modelling and Evaluation – may also be referred to as confirmatory analysis. As the model shows there are iterative cycles of

understanding -> modelling -> evaluation

at all stages of the process. In a paper published in "The American Statistician", Tukey (1980) argued strongly for both exploratory and confirmatory analysis, and differentiated the two, saying "*Important questions can demand the most careful planning for confirmatory analysis. Broad general enquiries are also important. Finding the question is often more important than finding the answer*." This statement sums up the role of EDA quite succinctly – the purpose is to understand the data so that the right questions can be posed and the right models explored.

2.3 History of EDA

As seen in the introduction, Tukey wrote a classic reference on EDA in 1977, and it has inspired many books and articles subsequently, clearly there is much to be learned from it. Reading the book today there is much that is familiar alongside much that seems quite dated, for instance the discussion of alternative methods to 5 bar gates for counting and graph paper as a tool for visualisation. The methods in Tukey's book are designed for the reader who is mainly exploring data by hand. While this may seem a somewhat laughable idea these days, the advantages of working so closely with the data are worth exploring. In the introductory quote from Schutt and O'Neil (2013) we are urged to develop "intuition" about the data. Intuition is born from experience, and what better way to experience a dataset than to analyse it manually.

Given the volume of data that requires analysis these days, manual analysis is obviously not a viable approach, and this was beginning to be the case in Tukey's time. When he wrote "Exploratory Data Analysis" he was working at AT&T Bell Labs where the statistical software S was being developed, the forerunner of the R software platform that is popular today. Tukey (1980) observes "*we have to take seriously the need for steady progress toward teaching routine procedures to computers rather than to people. That will leave the teachers of people with only things hard to teach, but this is our proper fate.*"

The move away from connecting with the data is noted by Berthold and Hand (2003) "*As data sets have grown in size and complexity so there has been as inevitable shift away from direct hands-on data analysis towards indirect data analysis in which the analyst works via more complex and sophisticated tools.*"

These days the speed at which analysts are distanced from the source data is only increasing. Cheap, powerful computing and open source software such as R has rendered the barriers of entry to data analysis almost non-existent, creating a climate where it is easy for analysts to never develop that connection with the data and its source that Schutt and O'Neil advocate in the opening quote. In the following section the impact of open source EDA tools which can be used without any analytical experience or formal training is discussed. Tukey talks about "teachers of people", but we live in a world where increasingly we can be our own teachers, randomly picking and choosing what we learn in an unstructured fashion. Later in the report examples of how curated learning sources can be built into the proposed software tool are given, so naïve users are directed to appropriate resources.

2.4 Modern EDA

In the American Statistician article Tukey (1980) says – "*Exploratory data analysis is an attitude, a flexibility, and a reliance on display, NOT a bundle of techniques, and should be so taught*", however searching a selection of the online sources and printed literature, EDA is variously described as "A set of techniques" - Harvard Biomedical Data Science (2017), "An approach/philosophy … a variety of techniques" – National Institute of Standards and Technology (2017), "a set of tools" - Schutt and O'Neil (2013).

The literature reviewed above gave detailed explanation of the tools and techniques of EDA, and frequently the literature referenced Tukey's work. Examples of histograms, boxplots, qqplots, tabulations and so forth were often encountered, but It was rare to find any practical case studies which might walk the reader along an exploratory path, illustrating the attitude and flexibility Tukey describes, or the mindset that Schutt and O'Neil advocate. What was generally observed was that each technique and visualisation was specific to a class of variable as defined by data type, and the workflow is

determine variable data type -> apply technique

Analysts determine the data type of the variable by eye or by interactive inspection, and create a bespoke script or interactive session for each dataset they work with. DeLine and Fisher (2015) describe a use-case study observing data analysts involved in EDA using the command line in a Read-Evaluate-Print Loop (REPL) fashion. The paper evaluates this REPL approach with the same task performed in a live programming environment, and finds the REPL environment to be slower and more error prone.

A script based approach is better, but still has its drawbacks, not least that it is time-consuming and error prone. The initial questions about any dataset are repetitive – how many records are there, how many variables, what type of variables, are there missing values and so on.

In a qualitative paper exploring working practices in data analysis, Kandel et al. (2012) find scripts are not widely shared by analysts. They give various reasons for this for instance lack of documentation by the script authors, difference in background knowledge of various coding languages. One consequence of this approach is that analysts may develop their own approach to EDA, and fall into bad habits, perhaps using a limited repertoire of descriptive statistics, failing to save results or make them available as part of the published research. The project proposal cited the example of Kanchan and Krishan (2013) who are forensic anthropologists researching deceased populations. Descriptive statistics on skeletal teeth are omitted from a paper they reference, although these numbers must have been available to the original researchers, and would have been useful to their work as teeth are an indicator of diet, gender and age distribution in populations.

A "walkthrough" EDA example was found in "R for Marketing Research and Analytics". Chapman and Feit (2015) devote a chapter to EDA, which they introduce as *"the simplest analysis one can do yet also the most crucial"*. A useful roadmap to starting an EDA session is given in the section "Recommended Approach to Inspecting Data", and eight R commands are documented and explained. What can be observed is that the commands are generic to any structured dataset. The book moves on to give other analyses one might perform generalised by data type of variables, with the usual selection of boxplots, histograms and so on as found in the wider literature, giving R code examples.

Amar et al. (2005) decompose the analytic task into the following taxonomy : Retrieve Value, Filter, Compute Derived Value, Find Extremum, Sort, Determine Range, Characterize Distribution, Find Anomalies, Cluster, Correlate. The Chapman and Feit (2015) methods cover about 70% of this this taxonomy. Schutt and O'Neil (2013) describe EDA as "*a method of systematically going through the data*", and this systematic approach is clearly demonstrated by Chapman and Feit (2015).

A systematic approach is not necessarily inconsistent with Tukey's warning that EDA is more than just a set of techniques. Findings at each iteration of EDA determine the questions to be asked at the next step, the work of the analyst is to judge what results are meaningful and which lines of enquiry will prove fruitful.

However, if a process is systematic then it can be abstracted, and there seems to be evidence that the generic, repetitive early stages of EDA could, to a point, be more formally organised into a structure which can be "taught to a computer" to paraphrase Tukey.

Examples of this begin to emerge in more recent literature.

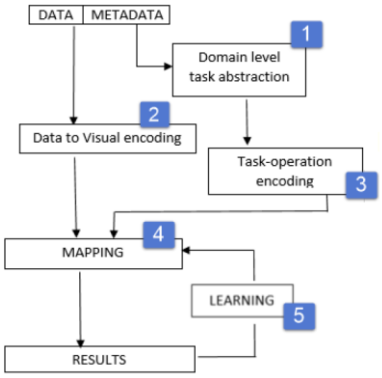


Figure 3. Mapping workflow for visualisation models
Source : Kaur et al. (2015)

In a paper on visualisation design Munzner (2009) notes that "*an explicit discussion of the choices made in abstracting from domain-specific tasks and data to generic operations and data types is not very common in papers covering the design of actual systems*".

Kaur et al. (2015) address exactly this point in their research developing automated visualisation systems. They extract metadata about the data, use it to abstract the task at hand and then map specifics of the data within the framework of the abstracted task as shown in Figure 3.

In summary, many learned sources acknowledge the importance of EDA, and it sits comfortably in a long-established framework of the data analytics process which is still currently relevant. The theory of EDA is that it is an intuitive process requiring human expertise, yet in practice much of EDA consists of a repetitive, uninspiring set of manual tasks.

The last word belongs to Schutt and O'Neil (2013) who state that EDA is "*a critical part of data analysis*", while at the same time acknowledging "*No wonder no one thinks much of it*".

2.5 Visualisation

Tukey frequently emphasized the importance of visualisation as a method of exploring model fit, understanding patterns and connecting with the data, but there was significant manual work involved producing graphs in his day. Forty years later there is huge flexibility offered by modern software, and Information Visualisation has become an independent area of study. Gelman and Unwin (2013) draw the distinction between information visualisation and statistical graphics, stating "*Within statistics, exploratory and graphical methods represent a minor subfield and are not well integrated with larger themes of modeling and inference. Outside of statistics, infographics (also called information visualization or Infovis) is huge, but their purveyors and enthusiasts appear largely to be uninterested in statistical principles*."

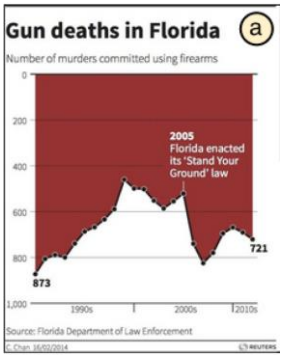


Figure 4. Example of misleading visuals, source Florida Dept of Law Enforcement via Pandey et al. (2015)

This seems a slightly sweeping statement, but it is fair to say that many examples of misleading visuals can be found. Pandey et al. (2015) give several examples, one of which is illustrated in Figure 4.

Gelman and Unwin (2013) acknowledge the importance of information visualisation (InfoViz) in telling important stories and grabbing public attention, but they argue that the goal of statistical graphing is different from that of InfoViz and that this distinction must be observed when designing visual output.

Although a detailed discussion is beyond the scope of this project, as the software design will describe a method for maintaining a library of statistical graphs to which data is passed as appropriate, this will offer a technique for maintaining reporting consistency between datasets and analysts. Consistent presentation facilitates good comparison and, once the visuals become familiar, for human intuition about their interpretation to be developed.

2.6 Citizen Data Science

Professional quality open source software such as R and Python, open data sources such as the UCI repository, free online training from websites such as Coursera and edX, and availability of low cost computing have created an environment where anyone can learn about and practice data science without any formal training or application of scientific rigour, and citizen data scientists have started to emerge. A report from CITO Research (2017) references a Gartner prediction that "*the number of citizen data scientists will grow five times faster through 2017 than the number of data scientists*". Despite the term "citizen" which would suggest social interest, the report places this rise within a business context due to the scarcity of trained data scientists. The report conjectures that there will be a rise in popularity of software tools that this group of people can use. These software tools will not necessarily be high-end systems for performing deep learning, but address more mundane tasks such as data investigation and preparation.

Any advance in data and statistical literacy must be a good thing, but given the ease with which data science software can be accessed it will be increasingly important that products offering rigorous methods and quality of output for fundamental processes are available. EDA seems an overlooked area which is ripe for research.

2.7 Data Science platforms

The 2015 O'Reilly Data Science Salary Survey – O'Reilly (2015) - states that "SQL, Excel, R, and Python" have been the top 4 data science tools since 2013, and a 2016 online poll – KDNuggets (2016) - of data science software reports the same. KDnuggets finds that R and Python to be the two most used software tools (49% vs 45.8%). Figure 5 below compares these platforms.

	SQL	Excel	R	Python
Read from various datafile types	Very hard	Very hard	Tools available	Tools available
Statistical testing routines	No	Some, others need to be programmed	Wide variety of tools	Wide variety of tools
Accessible toy datasets	Northwind	No	Plentiful	Plentiful
Array manipulation	Achievable with custom SQL statements	Limited with use of array formulas	Yes, easily for dataframes	Yes, easily for dataframes
Object oriented	No	No	Yes, not rigorously	Yes
Good tools for creating visualisations	No	Limited	Yes	Yes
Open Source	Free version with good functionality available	Low cost, generally available	Yes	Yes

Figure 5. Data Science Software platforms

2.8 Data wrangling

Reproducible Research is widely accepted as a best practice standard, and it is important for data analysis that both the source data and the analysed version are fully documented. Kandel et al. (2011) state "*Some of the most common insights people gain using visualization are about data errors and outliers … Typical research papers tend to showcase the result of visualizing previously cleaned data, but more often than not neglect to mention how data errors were found and fixed"*. EDA frequently identifies shortcoming of the data and leads to processes such as data cleaning, data wrangling or data reformatting. This steps are necessary for analysis, but they move the data further away from its original source. The project proposes a method for easily collecting and saving detailed meta data (Burrow data) about the source dataset. Burrow data can be collected for each iteration of the cleaning process, implicitly documenting the effect of applied changes, which adds more information than simply a script of actions that were applied.

2.9 Downstream processes

If EDA can be productionised the obvious question is to what extent. Can EDA software be developed to the point where it can offer guidance with downstream analytical tasks? As can be seen in the CRISP-DM diagram (Figure 2) EDA leads to the crucial stage of model selection so this would seem an area worth exploring.

There is much ongoing research into automation of data science processes, with perhaps the best-known example being IBM Watson. Not all projects are so successful, the University of Cambridge (2014) announced $750,000 of funding from Google for a project entitled "*The Automatic Statistician*". A year later Simonite (2015) reported that this software would be available on the project website later that year, although there seems to be no significant news of progress later than this. The Automatic Statistician website - Lloyd (2017) - still states "*The project is at an early stage*". It seems this well-funded, publicly announced project has faltered somewhere along the line.

Examples of less ambitious automation projects can be found, for instance Kuhn and Johnson (2013) introduce the well-regarded R package called caret, which allows users to investigate performance of a dataset in many algorithms with many different parameters and thus empirically evaluate the best performing. The accompanying textbook states *"Readers should have knowledge of basic statistical ideas, such as correlation and linear regression analysis. While the text is biased against complex equations, a mathematical background is needed for advanced topics.*". The obvious point is that users of caret should have some data science awareness and, as caret requires data to be in an all numeric format, typically some pre-processing and EDA steps will need to have been performed.

Feurer et al. (2015) introduce the auto-sklearn library as a development motivated by the *"ever-growing demand for machine learning systems that can be used off the shelf by non-experts".* This sentence strikes a worrying note. It makes an implicit assumption that a dataset has valid information within it that can be learnt by an algorithm, and that it takes no human skill or judgement to extract useful meaning; no intuition required. A computer science adage is "Garbage in, Garbage Out". No matter how sophisticated the algorithm or how sleek the software, if the data does not have meaning within it then the machine cannot learn anything. Schutt and O'Neil (2013) express this beautifully in the quotation introducing this section of the report – "*try to connect your understanding of the process that generated the data to the data itself".* It could be argued that if the auto-sklearn library becomes popular then a formal EDA step is even more crucial.

The literature shows that automating the data science process is a huge and complicated field of study, far beyond the scope of this project. However, it follows that if EDA leads to model selection and if EDA can be structured then this may lead to avenues of exploration in the search for automation.

2.10 Conclusion

It can be seen from the large number of references to Tukey's 1977 book that it is still an important work. To put the time at which it was written into context, tools for visualising data were tracing paper and pencils, yet many of the statistical techniques, data considerations and visual outputs he describes are still currently relevant.

The CRISP-DM methodology although first proposed seventeen years ago, is still currently used and written about by respected, practising data scientists.

So, in many ways EDA has not dramatically evolved over the years, and is a well-defined discipline. However, it is still presented as a ragbag collection of assorted techniques, approaches, statistics and visuals. There is plenty of scope to develop a systematic framework that removes the repetitive human input currently required, and evidence that this is achievable. This would leave analysts free to concentrate on the message of EDA, which will allow the opening message from Schutt and O'Neil (2013) that "*it's also a mindset"* to be realised.

Chapter 3 – Methods

3.1 Introduction

The research followed the deductive process, using an experimental method in which a software design was developed and prototyped.

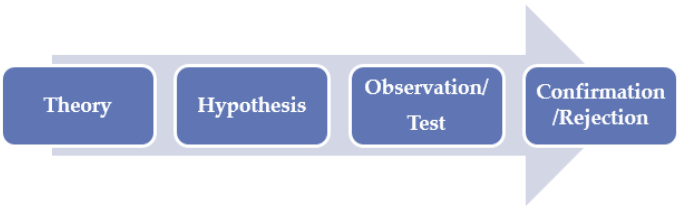


Figure 6. The deductive process, source Dudovskiy (2016)

Gulati (2009) states that "*deductive means reasoning from the particular to the general*". In this research the deductive argument is –

> *If the data type of a variable is x, then the descriptive statistics to produce are y*

The hypothesis on which this theory rests is that it is possible to build a model which can detect the data type of a variable within a dataset accurately. Accordingly, the experiment was split into two independently testable parts.

- Part A – testing the hypothesis. The goal was to develop software where input is a source dataset, and output is a file of the derived features and learned classification of data type for each variable.

- Part B – testing the theory. The goal was to develop software where inputs are a source dataset and the learned features and classifications from Part A, and output is a report of descriptive statistics suitable to data type.

A subsidiary goal was to consider how the software might be built into a live release should it prove successful, and to develop it with that in mind.

3.2 Structure of the work

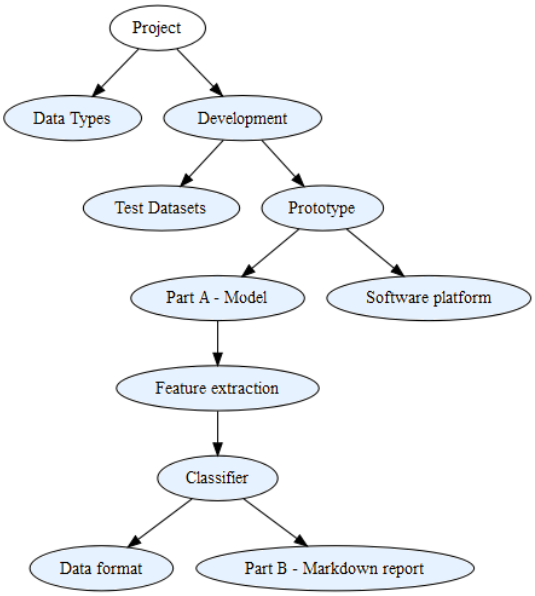


Figure 7. Work plan for the development

The work was structured as shown in Figure 7, and in this section methods for each part of the work are explained. Firstly, as part of the literature search different classifications of data types were identified, and considered in the context of what the project aims to achieve. Determining the data type of variables is one of the very early steps a human analyst performs in EDA, usually by visual inspection or by ad hoc data interrogation. It is this attempt to programmatically categorise variables as a human analyst would that is part of the new learning of the project.

Various programming approaches were trialled in a throwaway sandbox environment before software methods were chosen, and these are described. The reasons for choosing R as a software development tool are explained.

Pseudocode for the development is given, the methods used in Part A for extracting features is discussed and the classifier model explained. Part B uses different tools and the reasons for choosing are given. Finally, the test datasets are documented.

3.3 Taxonomy of data types

In the computer science domain '*data type'* refers to structures used by programming languages which have a direct relationship to their storage structure in the machine. Primitive types are the simplest types of data which cannot be decomposed into smaller units. There are subtleties between programming languages, but generally the definition given by Cardelli and Wegner (1985) is a good one - "*Primitive modes include int, real, char, bool, string, bits, bytes, format, file*".

In analytical terms '*variable'* refers to a set of data items classified by a statistical model of data type, such as that proposed by Stevens (1946) as "*Nominal, Ordinal, Interval, Ratio*".

This project borrows from both frameworks, attempting to classify both meaning and representation, as suggested by Hand (1993) "*Data consist of more than merely the numbers representing the magnitudes of attributes of the objects being investigated. The other component of data is the meaning of the numbers – the context in which they arose "*.

This is perhaps best explained by example. Boolean data is primitive data type which has a dichotomous state, generalised logically as "either True or False". There are many ways Boolean data can be encoded in a variable -

Variable contents	Machine representation
True/False	String
1/0	Numeric
Yes/No	String
1/0 (byte)	Bit
T/F	String

Figure 8. Data type examples

Depending on the source of the data, in the real world this might extend to 3 states, True, False or Don't Know. Survey data, for instance, might take this form. From the perspective of this project, all the examples in Figure 8 have data type of Boolean, with a sub-classification of encoding type, and this is the basis of the taxonomy presented in Appendix B.

The results of literature searches in this area were not sufficient to given the depth of information required, and other less traditional sources of information were considered.

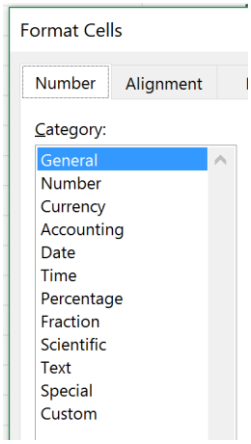


Figure 9. Data type options for column formatting in Excel

For instance, the "Format Cells" option in Excel was used as a source of information of how data types can be usefully classified for analysis purposes.

Given the analysis in Section 2.7 showing Excel is an often-used software tool for data science, it is reasonable to assume that the classification Excel uses is highly relevant.

The Teach-ICT (2017) website aimed at Year 7 pupils was a pleasantly surprising source of good practical examples.

From these searches a taxonomy of data types was created, which can be found in Appendix B. This taxonomy provided a framework for Part A of the development, and a list of features which could be used to identify and distinguish each data type was drawn up. For example, considering decimal variables: the characters in a valid field will be all numbers, there will be 1 decimal point, and there may be a plus or a minus leading the field. If there is a plus or a minus it may be separated by a space from the first number. Decimal fields which are very small or very large might be held in scientific notation expressed as digits and a multiplier.

The taxonomy was extremely useful in determining how to map features to classifications, and considerations of features was useful in deciding the levels of the taxonomy.

3.4 Prototyping

3.4.1 Software Engineering paradigm

A standard software engineering paradigm of the throwaway prototype model was followed in a sandbox environment. Initially no consideration was given to error trapping, ease of use, documentation, code commenting or visual quality of output. Microsoft's .NET platform was used to rapidly implement a prototype in a mixture of C++ and VB. Choice of development platform was based purely on author familiarity with .NET, and the wide range of object libraries available which meant many avenues could be explored with minimum code development.

This initial step allowed the structure and development considerations for the final work to be identified and scoped in a formal manner. Code and analytical results from this step are not presented here, but learning outcomes and pseudocode developed during this stage are given.

3.4.2 Lessons learned in the sandbox

An initial approach quickly discarded during prototyping was that the final software could be developed as an Excel add-in. As shown in 2.7 Excel is widely used in the data science community, and is often a tempting tool of choice to quickly open and explore CSV files. An add-in would allow further exploration to be conducted without having to leave this environment. However, several serious drawbacks were quickly identified. Worth mentioning are –

- On opening Excel tries to fit CSV variables into its internal data structures, and variables such as numeric values with leading zeros can be silently distorted and their meaning lost

- Excel has limits to the size of files that can be read. Currently, Microsoft (2017) these limits are 1,048,576 rows by 16,384 columns, assuming the most current version of Excel

- Limited range of built in statistical tests

- To correctly classify variables, it is important to look at the entire content of the dataset and the internal data structures of .NET or VBA do not lend themselves as efficiently to this as does the dataframe implemented in R and Python. Even if file size limits were not a problem operationally, lack of efficient data structures for the computations required would be.

During the sandbox stage input data was restricted to CSV files, but the potential set of data file formats valid for input was considered. It was identified that a software platform with built in tools for reading from different formats would offer significant advantages over one which only read CSV file types. A platform which read different file types into an internal object-based structure such as a dataframe would give operational advantages as code development could assume file handling considerations would not be required within the scope of work.

Feature extraction was initially prototyped for a variable as a function looping through each record in the data considered all variables as strings, and counting occurrences of target characters. In 3.3 an example is given of the features to consider when classifying a decimal data type, some sample tests considered for this were –

- proportion of characters that are numeric
- proportion of decimal points/full stops
- proportion of plus signs
- proportion of minus signs
- proportion of whitespace
- incidence of the string “EX” or similar, as evidence of scientific notation

Output from this exercise was a wide feature vector containing numeric values for each variable in each data record. The variables were classified by hand and target outputs added to the feature vector. This gave a training dataset which was put into a neural network. The results of this was total failure and detail from it is of no material interest. The form of the feature vector was reconsidered and reworked several ways, but no improvement was gained.

This work was useful, however, as it prompted consideration of how such an approach would not be easily extensible if the final software were to be used on a wider scale. Even if the problems of representation and classification model could be addressed. The feature vector created was large even when considering only a few data types. Many domains have variable data types that are very specific, for instance a survey dataset might contain rating scales held categorically in the data as “Very Likely” … “Very Unlikely” and the system would want to classify all types of rating scale. Trying to extend a single feature vector to cover all the scenarios that might arise would quickly lead to high dimensional considerations. Even if these could be addressed by implementing dimensionality reduction techniques such as Principal Component Analysis, any extension over time to the target data types would require significant model retraining which would limit its potential use and ease of distribution.

A vastly more successful approach was given using a combination of two techniques –

- regular expression pattern matching (regex) to extract features in the form of density of matched patterns
- writing the results to a tidy data format - Wickham (2014) – structured as one record per feature extracted with tags written to identify the test type

This proved to be a solution which was easy to implement as a code pattern as the only variant per test was the regex expression. The tidy format data is endlessly and easily extensible over time. As the project progressed the flexibility of this construction became apparent and illustrations will be given in the following sections. Appendix D gives further details of the Burrow data and regex expressions.

The conclusion of the sandboxing stage was that R or Python would be an appropriate development platform as both contain the following essential tools -

- reads a variety of data file formats
- flexible internal data structures
- good range of statistic tests
- good regex support

As the taught part of the MSc course used Python, R was chosen for the project development for additional learning and to give the opportunity to compare the two from experience.

3.5 Developing Part A

Part A covers feature extraction and classification modelling. In this stage of the project the pseudocode developed in the sandbox was implemented using the chosen methods.

3.5.1 Pseudocode

```
Read source data
Convert to standard internal format (dataframe)
Collect meta information about the run
Collect meta information about the dataset
For each variable (v)
        For each feature (f)
                Count proportion of feature matches (c)
                Write (v, f, c) data

        Classify each variable into data type (t)

Write (v, t) data
```

Figure 10 – Pseudocode for Part A

3.5.2 Meta information about the run and dataset

Header information about the run itself was collected. The name of the dataset being analysed, the description, run time and date, and the final processing time. The Burrow data produced from each run would be uniquely identifiable if it were being shared between analysts or if an analysis were being returned to after some time. This structure could easily be expanded to automatically include e.g. system environment information picked up by R as shown in Figure 11.

```
> Sys.info()
      sysname        release        version       nodename        machine
    "Windows"    ">= 8 x64"   "build 9200"          "P37"      "x86-64"
        login           user effective_user
    "Suzanne"     "Suzanne"      "Suzanne"
>
```

Figure 11. R System information

Meta information about the dataset was collected using commands in R partly sourced from Chapman and Feit (2015) to find information such as number of records, number of variables, variable names and the order they appear in the data. Hints about the data type of each variable could be collected from the internal classification R makes when it reads a dataset. R data types broadly correspond to the top level of the taxonomy presented in Appendix B, and some type dependant features such as numeric max, min, mean, median could therefore be collected at this early stage.

3.5.3 Extracting features

Features were extracted using several methods depending on the information required. For features determined by regex pattern matching, an expression was developed and tested. Regex expressions have their own grammar, are very terse and can have unforeseen outcomes if they are not very logically constructed with all scenarios considered and catered for. Developing them for the project involved a steep learning curve, and it was important to check them properly. This could be done outside of R as regex is a standalone language, implemented in many programming environments.

Using regex was almost abandoned because constructing sufficiently rigorous expressions proved difficult, and much credit must be given to Toarca (2017) for his website www.debuggex.com shown in Figure 12. This site parses regex expressions and presents a visual workflow representation of the logical branches and entry conditions; the user can move through the expression examining the flow. Several online tools were found, but this one was by far the most helpful due to the visual nature of the analysis and the ability to examine each branch, considering the whole as a distinct set of parts. This tool made the robust development of quite complicated expressions possible, and after using it for a while the syntax of regex became far easier to understand, leading to less errors constructing it. A demonstration of Tukey's insistence that visual methods are best for learning patterns.

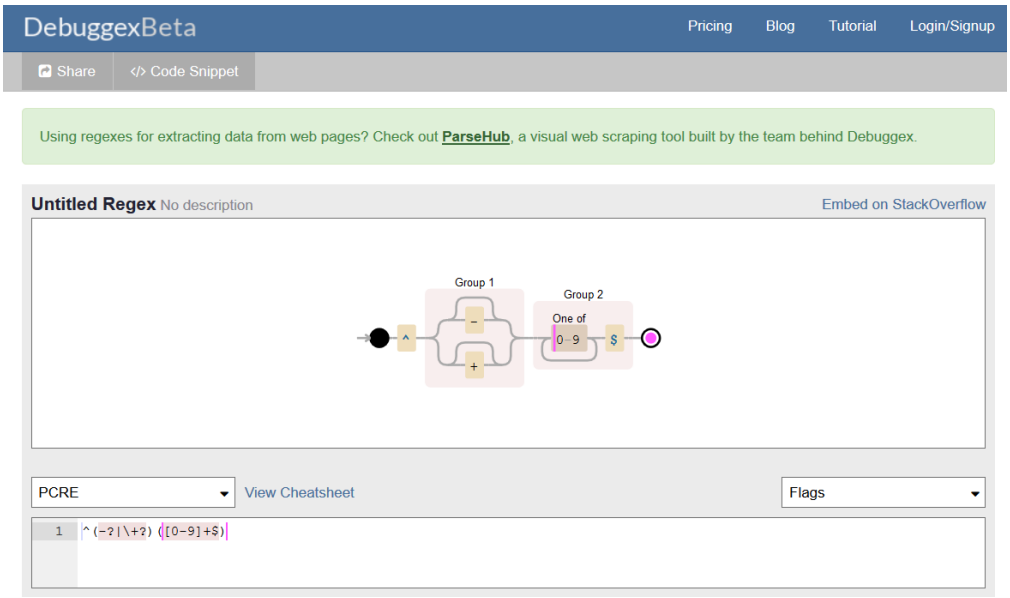


Figure 12. www.debuggex.com

For each regex expression developed the variable data was tested to find the number of records for which it was true. This was divided by the number of valid records (i.e. those without a missing value) to give an index. An example can be seen in Figure 13 which shows partial Burrow data for the Sepal Length variable from the iris dataset, 88.66% of this variable is decimal values and 11.33% is purely numbers.

FIELD	FEATURE	WIDTH_MEAN	UNIVARIATE	Sepal.Length	2.77333333333333
FIELD	FEATURE	REGEX_NUMBER	UNIVARIATE	Sepal.Length	0.113333333333333
FIELD	FEATURE	REGEX_DECIMALS	UNIVARIATE	Sepal.Length	0.886666666666667
FIELD	FEATURE	REGEX_PERCENT	UNIVARIATE	Sepal.Length	0

Figure 13. Example Burrow data for the iris dataset #1

Other features can be obtained without using regex. In Figure 14 we can see that there are no missing values for Sepal Length, and there are 35 unique values which is 23% of the data.

FIELD	FEATURE	COUNT_NA	UNIVARIATE	Sepal.Length	0
FIELD	FEATURE	DENSITY_NA	UNIVARIATE	Sepal.Length	0
FIELD	FEATURE	COUNT_UNIQUES	UNIVARIATE	Sepal.Length	35
FIELD	FEATURE	DENSITY_UNIQUES	UNIVARIATE	Sepal.Length	0.233333333333333
FIELD	FEATURE	DENSITY_UNIQUES_EXNA	UNIVARIATE	Sepal.Length	0.233333333333333
FIELD	FEATURE	MAX	UNIVARIATE	Sepal.Length	7.9
FIELD	FEATURE	MIN	UNIVARIATE	Sepal.Length	4.3
FIELD	FEATURE	MEAN	UNIVARIATE	Sepal.Length	5.84333333333333

Figure 14. Example Burrow data for the iris dataset #2

The DENSITY_UNIQUES feature is useful in text variables for indicating that the data type may be categorical rather than just text. Figure 15 shows Burrow data for the Species variable. From the DENSITY_ROW_ALPHAS feature we can tell that the data is 100% text. As there are less than 11 unique values, this field is classified as categorical and the classifier writes a BEST_GUESS feature with information on what rule was the deciding one for the classification.

FIELD	FEATURE	DENSITY_ROW_ALPHAS	UNIVARIATE	Species	1
FIELD	FEATURE	DENSITY_DATASET_ALPHAS	UNIVARIATE	Species	1
FIELD	FEATURE	DENSITY_ROW_PUNCTUATION	UNIVARIATE	Species	0
FIELD	FEATURE	DENSITY_DATASET_PUNCTUATION	UNIVARIATE	Species	0
FIELD	BEST GUESS	REGEX ALPHAS 95% MATCH AND < 11 UNIQUES	UNIVARIATE	Species	CATEGORICAL

Figure 15. Example Burrow data for the iris dataset #3

3.5.4 Classification model

A deterministic, rules based classification model was created to make a “best guess” at variable data type from the feature data. This terminology is important to the mindset with which the Burrow data is intended to be used, i.e. that this is a method of facilitating EDA, not a replacement for it.

A hierarchy of guessing rules was implemented. Firstly, at step 1, features identifying highly discriminative regex patterns were tested.

```
# put these in order of precedence and test
# the first one that is true  is the choice
regex_precedence <- c(
  "REGEX_DATE_DDMMYYYY_DASHES",
  "REGEX_DATE_DDMMYYYY_SLASHES",
  "REGEX_CURRENCY_UK",
  "REGEX_CURRENCY_USA",
  "REGEX_BIT_ONEZERO",
  "REGEX_BIT_TRUEFALSE",
  "REGEX_BIT_TF",
  "REGEX_BIT_YN",
  "REGEX_BIT_YESNO",
  "REGEX_GEO_POSTCODE",
  "REGEX_SCALE_210120",
  "REGEX_PERCENT",
  "REGEX_SCIENTIFIC",
  "REGEX_TIMES10",
  "REGEX_WEB_URL",
  "REGEX_WEB_EMAIL",
  "REGEX_DECIMALS",
  "REGEX_GEO_LATITUDE",
  "REGEX_GEO_LONGITUDE",
  "REGEX_NUMBER"
)
```

Figure 16. Feature precedence

The features were listed in order of precedence, with highly pattern specific features at the top of the list as shown in Figure 16.

The classifier examines each feature in turn, a value of 0.95 or greater causes the classifier to take that as the best guess and processing moves to the next variable.

At the end of this section, if no valid guess has been made other rules are applied.

Because of the construction this is highly customisable, in fact several models could be implemented concurrently to compare their behaviour.

The classifier discriminates between variables which are pure number or decimal fields, which will have been classified at step 1 above, and variables with mixed numbers and decimals.

```
# ============================================================
# Step 2
# more complicated tests are needed
# ============================================================
# test for decimals
if (test.bestguess == "DK") {
  if (diagnostics == TRUE)
    print(paste("........................... Decimals test :", test.variable))

  test.if <-
    subset(test.features,
           InfoDetail == "REGEX_DECIMALS" | InfoDetail == "REGEX_NUMBER")
  if (nrow(test.if) == 2) {
    test.regextotal <- sum(as.numeric(test.if[, c("myData1")]))
    if (test.regextotal > 0.95)
      test.bestguess <- "DECIMALS"
    test.guesssource <- "MULTIPLE CONDITIONS REGEX MATCH"
    if (diagnostics == TRUE)
      print(paste("BEST GUESS IS", test.bestguess))
  } # end of test.if has 2 records
} # end of decimals test
```

Figure 17. Testing for mixed numeric/decimal variables

Much more sophistication and granularity would be easily possible. Variables could be classified as all positive decimals, numbers including negative values etc.

Further steps of increasing complexity continue to classify variables which have not satisfied any combination of rules. Finally, variables left unclassified are assigned a Don't Know status.

Although this classifier was surprisingly simple compared to the neural network implementation originally proposed, the classification accuracy of this model was good (91%) when tested on unseen data, and this was deemed sufficiently successful to provide proof of concept.

3.5.5 Writing the Burrow data

As can be seen in Figures 13 and 14 the Burrow data is formatted as one data item per record, tagged by various features. This exemplifies the tidy format that lends itself to infinite extension and flexibility of analysis, this will be explored further in the next section. For developmental purposes reaching this stage was a significant milestone because it proved that a classifier could be built, and provided meta data for input to the next stage of development.

Appendix D documents the meta data format and the tagging structure used.

3.6 Developing Part B

The sandbox prototype did not extend to consideration of how an EDA report would be produced, so some time was spent investigating options. As shown above, the Burrow data can be written as a standalone CSV, so at this point there was no constraint requiring the development to continue in R.

R has several useful libraries for producing visualisations. Libraries such as ggplot2 provide "*a powerful model of graphics that makes it easy to produce complex multi-layered graphics*" Wickham (2013). Because ggplot2 is a grammar for creating graphical plots, not a library of pre-defined plots, parameterizable functions can be created to which data can be passed. This was a very useful construct for the project.

Markdown is a text to HTML conversion tool developed in 2004 by Gruber and Swartz, Gruber (2017). The design goal of markdown is that the input document is human-readable and various software platforms have taken this framework and extended it, writing Word documents, HTML or PDF from an easily readable source document. R implements a version of markdown allowing many formats of output to be created from a source containing a mixture of text and R code RStudio (2016).

The combination of ggplot2 and R-markdown offered much flexibility for developing the report and so the project proceeded in R.

3.6.1 Developing the Markdown document

In Part A, a generalisation was created of a dataset. Its variable structure was decomposed and documented as a Burrow file, and top-level statistics were collected based on data type of variables. Passing this data and the source data to a generalised markdown document allowed a report of appropriate (or prescribed) descriptive statistics to be automatically created for a dataset. In fact, as the markdown document mixes R code and text, Burrow data was created as an integral part of the report execution. The examples provided on the accompanying USB drives are set up like this.

Initially a markdown document showing high level information was developed. For instance, the level of missing values per variable was available from Burrow data, and those figures gave a quick visual summary of data quality. If there were no missing values for a dataset this could also be identified from the Burrow data, and the output varied accordingly. Examples are shown in Figure 18.

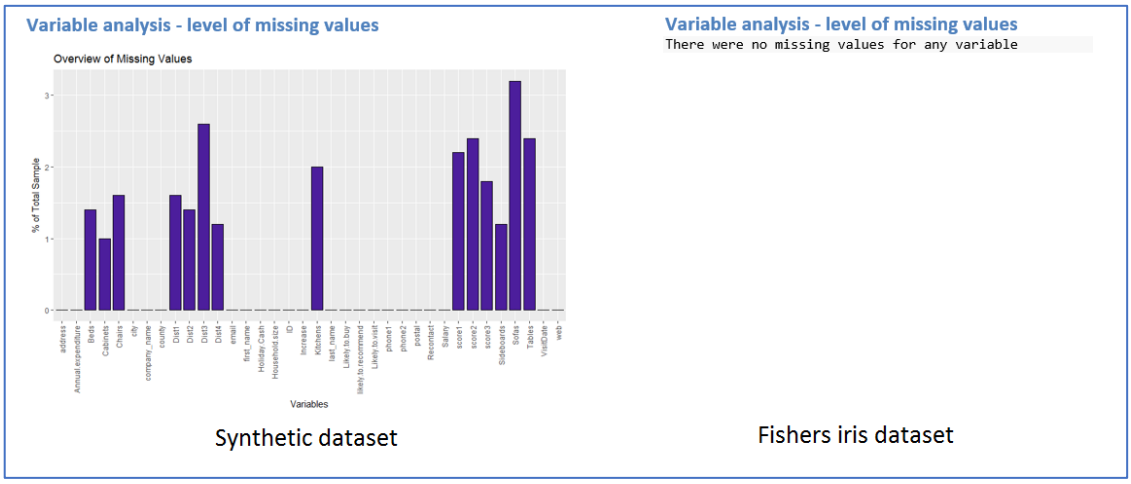


Figure 18. Alternative views of the same report depending on the input data are programmatically obtained.

Pseudocode for this step is given by -

Create dataset of (variable name, missing value count) from the Burrow data
Sum the missing values across all variables
If number of missing values found > 0
Pass dataset to graphing function

The chart used by the report is generic, with data in it extracted from the Burrow file by creating a subset of the DENSITY_NA feature rows. It can be seen from this example how the entire process comes together at this point. A dataset of missing values can be generically constructed for any source dataset irrespective of its structure from the Burrow data and passed to the function creating the chart in Figure 18.

The flexibility of using parameterised functions based on ggplot2 can be illustrated here. To create a visualisation for level of unique values, a dataset of DENSITY_UNIQUES Burrow features is created and passed to the same charting function that created Figure 18. Chart titles are passed as arguments to the function to label the statistics being displayed.

Visualisations can be adapted to any house style and saved in a function library. Visualisation libraries can be shared between analysts, and are reproducible.

Figure 19 shows an example of a custom visualisation built as a demonstration for the project in ggplot2 and again coded as a generalised function to which variable data is passed at runtime. This part of the markdown code picks up all variables whose classification was numeric, and explores their distribution. Figure 19 extends a qqplot by adding lines for mean, median, 1st and 3rd quarter median.

Note particularly how an R function is used at runtime to detect five outliers which are coloured in red on the chart. All this with no human intervention. An example of how the EDA process proposed by the project could be implemented is that specific details about outliers could be included in the Burrow data creation, and a separate outlier markdown report constructed which the analyst could access if necessary. This does not automate EDA, rather it offers the analyst tools to explore the data easily as part of a structured, intuitive approach.

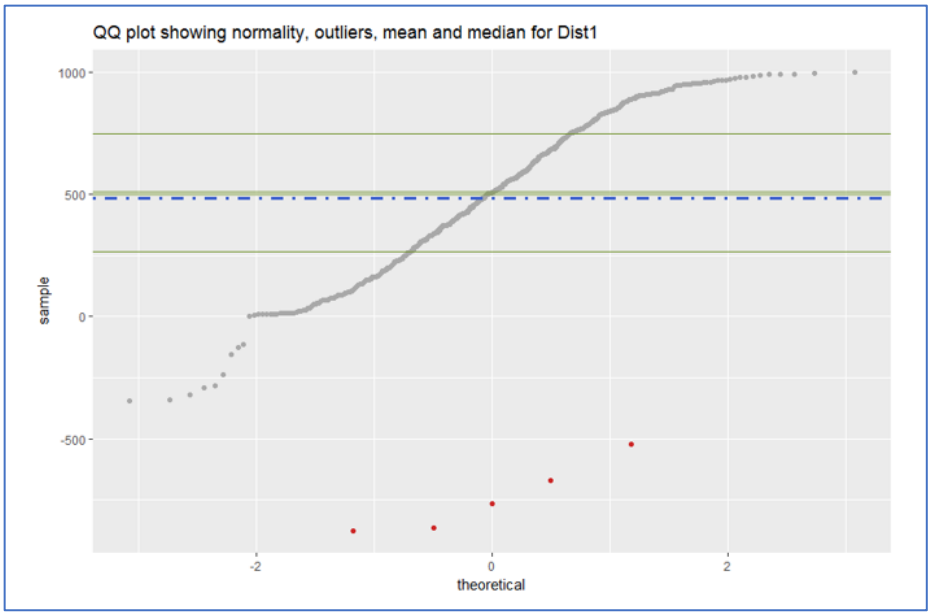


Figure 19. Custom visualisation built in ggplot2.

Pseudocode for Figure 19 is given by -

```
Search the Burrow data for numeric or decimal variables
If number of variables found > 0
    for each variable
        save variable name as a title
        extract named variable from the data source
        save as a generic single-column data structure
        pass single-column data and name to the plot function
```

3.6.2 Statistical tests

At this point the skeleton for the development was in place and worked well. Features could be extracted, accurately classified and visually presented as per the original objectives. The level of ease of use desired had been achieved and structure of the work meant it was easily extensible. The final part of the development was to extend the statistical tests and visualisations beyond the initial implementation, moving further down the taxonomy hierarchy.

Unfortunately, this was the point at which the project stalled as mentioned in the introduction, and implementation of this stage was far less robust than had been planned. As this stage simply builds on what has gone before it would have been illustrative of what can be achieved rather than crucial to proving the concept, nonetheless it was disappointing not to do more at this point.

The plan had been to return to academic research and extend it by identifying a wider range of univariate and bivariate descriptive statistics and visualisations and adding them into the Burrow function. As an example, the Shapiro test for normality was collected for all variables classified as numeric and summary visualisations were added into the markdown document showing how the Burrow data could be used to count the number of variables of numeric data type and adjust the formatting accordingly.

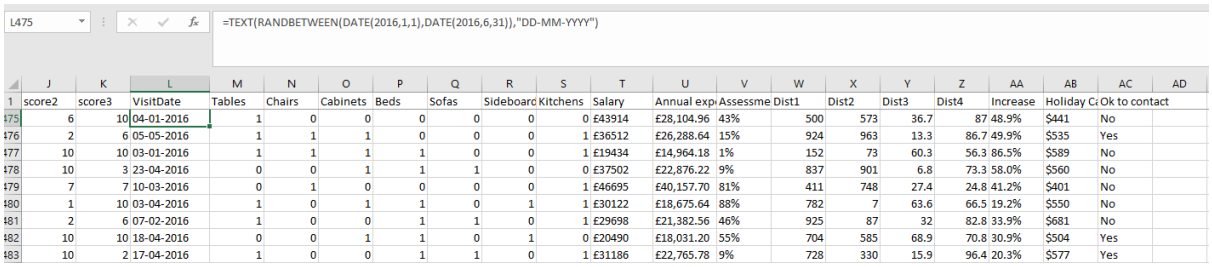
The Shapiro test returns a p-value indicating whether the result is significant. In the same way as demonstrated in the previous section the markdown report could pick up just the significant results and bring them to the analyst's attention. There are several normality tests available in R, the Burrow data could collect the results of all of them and report variables where the test results differ in some way. As computing power is fast and cheap and the Burrow format endlessly extensible there is no limit to what can be collected. Smaller subsets of features can be easily extracted for human readability if required.

3.7 Datasets

3.7.1 Development datasets

Two datasets were used while developing functions and objects. Fishers iris dataset is a widely used dataset, available from within R. It has 150 records, 4 numeric variables and 1 categorical. This was an ideal size to start work with as run times were quick and the results easy to examine.

No single dataset was found with examples of each data type from the taxonomy, and so a synthetic dataset was created for development purposes. Full details can be found in Appendix C. This dataset was based on publicly available data Dunning (2017). Starting with a file of 500 records of syntactically correct but fictitious names, addresses, postcodes and phone numbers, the random generator and distribution functions fields in Excel were used to add variables for data such as UK and US currency fields, boolean fields with various encodings and so on as shown in Figure 20.



L475 =TEXT(RANDBETWEEN(DATE(2016,1,1),DATE(2016,6,31)),"DD-MM-YYYY")

	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD
1	score2	score3	VisitDate	Tables	Chairs	Cabinets	Beds	Sofas	Sideboard	Kitchens	Salary	Annual exp	Assessme	Dist1	Dist2	Dist3	Dist4	Increase	Holiday C	Ok to contact	
475	6	10	04-01-2016	1	0	0	0	0	0	0	£43914	£28,104.96	43%	500	573	36.7	87	48.9%	$441	No	
476	2	6	05-05-2016	1	1	1	0	0	0	1	£36512	£26,288.64	15%	924	963	13.3	86.7	49.9%	$535	Yes	
477	10	10	03-01-2016	1	1	1	1	0	0	1	£19434	£14,964.18	1%	152	73	60.3	56.3	86.5%	$589	No	
478	10	3	23-04-2016	0	0	1	1	1	0	0	£37502	£22,876.22	9%	837	901	6.8	73.3	58.0%	$560	No	
479	7	7	10-03-2016	0	1	0	0	0	0	1	£46695	£40,157.70	81%	411	748	27.4	24.8	41.2%	$401	No	
480	1	10	03-04-2016	1	0	1	1	0	1	1	£30122	£18,675.64	88%	782	7	63.6	66.5	19.2%	$550	No	
481	2	6	07-02-2016	1	0	0	1	1	0	1	£29698	£21,382.56	46%	925	87	32	82.8	33.9%	$681	No	
482	10	10	18-04-2016	0	0	1	1	0	1	0	£20490	£18,031.20	55%	704	585	68.9	70.8	30.9%	$504	Yes	
483	10	2	17-04-2016	1	0	0	1	1	0	1	£31186	£22,765.78	9%	728	330	15.9	96.4	20.3%	$577	Yes	

Figure 20. Synthetic dataset

3.7.2 Datasets used for testing

The program was tested with datasets from a variety of sources and domains. Some of the datasets built into R were used, some data held in the SQL Northwind database, some CSV files sourced from the UCI repository, and some data from the STATS19 Road Traffic Accidents dataset. Full details of datasets used and reasons for their choice are given in Appendix E, and the results section gives the confusion matrix for the classification accuracy testing.

Chapter 4 – Results

4.1 Sandbox

Results from the sandbox phase were selection of methods for –

- development environment
- development workflow diagram
- method for extracting features
- method for classifying data type from feature
- file formats for burrow data output

4.2 Workflow diagrams

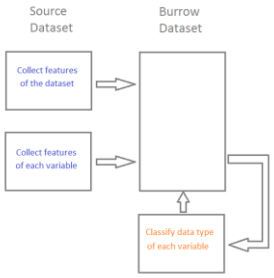


Figure 21. Workflow diagram classification

Collecting features and classifying the data type for each variable, was abstracted to the workflow shown in Figure 21, which formed Part A of the development.

At the end of this process the Burrow data contained features about the dataset, features about the variables, and the data type classification for each variable.

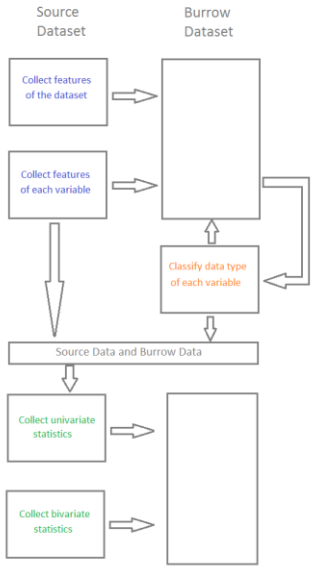


Figure 22. Full Workflow diagram

The workflow is then extended as shown in Figure 22, to use what has been determined to collect univariate and multivariate statistics as prescribed by the data type of the variable.

The burrow file extends as each calculation is added. This ability to accommodate endless expansion is a result of the consideration given to output formats during the sandbox stage.

4.3 Burrow data extensible format

Features, classifications and statistics were written to an output burrow data file in a structured "tidy" format - Wickham (2014).

InfoLevel	InfoType	InfoDetail	VarLevel	Variable1	myData1	Variable2	myData2	myNotes
HEADER	RUN DATE	Sun Feb 19 2017, 16:45:07	UNIVARIATE					
DATASET	SOURCEDATA	iris	UNIVARIATE					
DATASET	SOURCE	Fishers Iris dataset	UNIVARIATE					
DATASET	FEATURE	COUNT_COLUMNS	UNIVARIATE	NUMCOLS	5			ncol(myData)
DATASET	FEATURE	COUNT_ROWS	UNIVARIATE	NUMROWS	150			nrow(myData)
DATASET	DETAIL	ORDINAL_POSITION	UNIVARIATE	Sepal.Length	1			
DATASET	FEATURE	COLUMN_NAME	UNIVARIATE	COLNAME	Sepal.Length			names(myData)
DATASET	FEATURE	DATA_CLASS_R	UNIVARIATE	Sepal.Length	numeric			class(myData[,Field])
DATASET	FEATURE	DATA_TYPE_R	UNIVARIATE	Sepal.Length	double			sapply(myData,typeof)
DATASET	DETAIL	ORDINAL_POSITION	UNIVARIATE	Sepal.Width	2			
DATASET	FEATURE	COLUMN_NAME	UNIVARIATE	COLNAME	Sepal.Width			names(myData)
DATASET	FEATURE	DATA_CLASS_R	UNIVARIATE	Sepal.Width	numeric			class(myData[,Field])
DATASET	FEATURE	DATA_TYPE_R	UNIVARIATE	Sepal.Width	double			sapply(myData,typeof)
DATASET	DETAIL	ORDINAL_POSITION	UNIVARIATE	Petal.Length	3			
DATASET	FEATURE	COLUMN_NAME	UNIVARIATE	COLNAME	Petal.Length			names(myData)
DATASET	FEATURE	DATA_CLASS_R	UNIVARIATE	Petal.Length	numeric			class(myData[,Field])
DATASET	FEATURE	DATA_TYPE_R	UNIVARIATE	Petal.Length	double			sapply(myData,typeof)

Figure 23. Burrow output for the iris dataset

Figure 23 shows Burrow data from the iris dataset, it is standard CSV so easily consumable by most software programs. Levels of meaning are encoded for each record. Full documentation is given in Appendix D, but an exploration of the versatility of this method is worth describing.

The InfoLevel variable is encoded with the level at which the data is collected, values are Header, Dataset, Field. The file can be quickly searched by this term. Extending the search term to - InfoLevel contains Field and Variable1 contains Sepal.Length - gives all the feature and classification information for that variable.

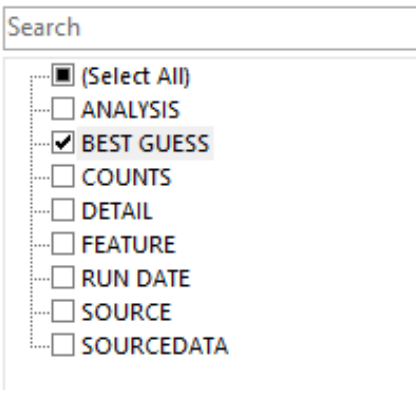


Figure 24. Burrow data filtered by InfoType in Excel

The Burrow data could be opened in Excel and explored interactively using filters. Figure 24 shows the values for the InfoType column. To quickly look at the classification for each variable, filter by the "BEST GUESS" value.

Results are shown in Figure 25.

InfoLev	InfoType	InfoDetail	VarLevel	Variable1	myData1	Variabl	myData	myNotes
FIELD	BEST GUESS	MULTIPLE CONDITIONS REGEX MATCH	UNIVARIATE	Sepal.Length	DECIMALS			Guess by Burrow function, revision 0.0.9000
FIELD	BEST GUESS	MULTIPLE CONDITIONS REGEX MATCH	UNIVARIATE	Sepal.Width	DECIMALS			Guess by Burrow function, revision 0.0.9000
FIELD	BEST GUESS	MULTIPLE CONDITIONS REGEX MATCH	UNIVARIATE	Petal.Length	DECIMALS			Guess by Burrow function, revision 0.0.9000
FIELD	BEST GUESS	MULTIPLE CONDITIONS REGEX MATCH	UNIVARIATE	Petal.Width	DECIMALS			Guess by Burrow function, revision 0.0.9000
FIELD	BEST GUESS	REGEX ALPHAS 95% MATCH AND < 11 UNIQUES	UNIVARIATE	Species	CATEGORICAL			Guess by Burrow function, revision 0.0.9000

Figure 25. Burrow output of classification results

The Burrow data reports which rule of the model was the winning one, and which version of the Burrow function was used for traceability. As the model is improved over time it might give different results for the same dataset at a later date, so versioning is important.

The result of using this format is that the project is endlessly extensible. Any number of features can be extracted, and several concurrent classification models could be run.

4.4 Feature extraction

The result of using regex as a feature extraction mechanism was very satisfying. It was easy to test, quick to run and code could be developed in a modular, easy to read fashion. Figure 26 shows the code fragment that defines, extracts and writes a feature.

```
# Regex : anything that looks like an integer (whole number)
myArgs.InfoDetail <- "REGEX_NUMBER"
density.regex <- "^(-?|\\+?)([0-9]+$)"
density.grepl <- grepl(density.regex, FieldAsChar.NoSpaces, perl=TRUE, ignore.case = TRUE)
myArgs.Data1 <- sum(density.grepl > 0) / nvalids
df.burrow = myContentLine(df.burrow, myArgs.InfoLevel, myArgs.InfoType, myArgs.InfoDetail,
                          myArgs.Variable1, myArgs.Data1, myArgs.Variable2, myArgs.Data2,
                          myArgs.Notes)
```

Figure 26. Code fragment for the "REGEX_NUMBER" feature, measuring density of records which optionally start with + or -, and then only contain numeric characters.

All the features extracted using regex require this 5 line pattern, and only 2 of those lines require editing. This results in easily readable, robust code that is quick to develop. It was considered that the regex pattern should be written as a note to the burrow output, but as some of them are very long this did not give a good result and the idea was abandoned.

4.5 Classification accuracy

The accuracy of the classifying function was tested on 18 datasets previously unseen by the classifier. The datasets were chosen to test the system in a variety of ways, by using datasets from different domains, different source systems, very large datafiles, datasets with different variable types and structures etc. Justifications for each dataset are given in Appendix E. In total 174 Variables were used as testing data from these datasets.

Variables in the dataset were classified by hand using the taxonomy framework to give a ground truth. 160 (91%) of the variables were correctly classified by the burrow function. Figure 27 gives the confusion matrix for the accuracy determination.

			Ground Truth									
	TOTAL	CLASSIFIED CORRECTLY TOTAL	BIT_ ONEZERO	BIT_ YESNO	CATEGORICAL	DATE	DECIMALS	LATITUDE	LONGITUDE	MAINLY NUMERIC	NUMBER	TEXT
TOTAL	174	160	2	16	26	1	17	1	1	9	89	12
BIT_ONEZERO	6	2	2								4	
BIT_TRUEFALSE	5	0			4					1		
BIT_YESNO	16	16		16								
CATEGORICAL	22	22			22							
DECIMALS	18	17					17	1				
DK	1	0										1
MAINLY NUMERIC	8	7				1				7		
NUMBER	85	85									85	
SCIENTIFIC	1	0							1			
TEXT	11	11										11
TIMES10	1	0								1		

Figure 27. Confusion matrix for test data

Of the 14 mis-classifications, 9 were classified as one of the binary (BIT_*) categories despite having quite different ground truth classifications, suggesting the deciding algorithm could be improved to be more discriminating in future work. For the purposes of proving that the goal of the project is achievable however, 91% accuracy was felt to be sufficient.

		R Data type		
	TOTAL	character	double	integer
TOTAL	160	57	40	63
BIT_ONEZERO	2	0	2	0
BIT_YESNO	16	16	0	0
CATEGORICAL	22	22	0	0
DECIMALS	17	0	17	0
MAINLY NUMERIC	7	7	0	0
NUMBER	85	1	21	63
TEXT	11	11	0	0

Figure 28. Comparison of R data types and burrow classification

When data is read into an R object, R makes a determination of native data type for each of the variables in the dataset. Examining the R data type of the 160 correctly classified variables in the testing data shows how much information has been added by the Burrow function.

4.6 Runtimes

Figure 29 provides the testing dataset metrics. The effective size for each file is the number of data Items, found by multiplying row count by variable count. The tests were run repeated times and the average run time per dataset calculated. There was little variation between each test run.

An index of unit run time per dataset was found using the formula

$$\frac{\text{Mean Run Time}}{\text{Data items}} \times 1000$$

Dataset		Source	Record count	Variable count	Data items	Mean run time (secs)	Unit run time (Index)
mtcars	mtcars dataset	Integral to R	32	11	352	0.81	2.29
HairEyeColor	Person chracteristics dataset	Integral to R	4	4	16	0.26	15.94
WorldPhones	World phones	Integral to R	7	7	49	0.46	9.29
airmiles	airmiles	Integral to R	24	1	24	0.05	2.08
attenu	attenuation data	Integral to R	182	5	910	0.32	0.35
esoph	Smoking Alcohol and (O)esophageal Cancer	Integral to R	88	5	440	0.37	0.83
warpbreaks	The Number of Breaks in Yarn during Weaving	Integral to R	54	3	162	0.20	1.20
swiss	Swiss Fertility and Socioeconomic Indicators (1888)	Integral to R	47	6	282	0.46	1.63
quakes	Quakes off Fiji	Integral to R	1,000	5	5,000	0.41	0.08
ratings.csv	Movie Ratings	MovieLens 20M Dataset	20,000,263	4	80,001,052	884.46	0.01
movies.csv	Movies	MovieLens 20M Dataset	27,278	3	81,834	1.94	0.02
tags.csv	Movie Tags	MovieLens 20M Dataset	465,564	4	1,862,256	18.91	0.01
accidents0514.csv	STATS19 Accident data	Data.gov.uk	1,640,597	32	52,499,104	341.74	0.01
student-por.csv	UCI Student performance dataset - Portuguese	UCI repository	649	33	21,417	2.76	0.13
student-mat.csv	UCI Student performance dataset - Maths	UCI repository	395	33	13,035	2.46	0.19
SQL Server	Northwind database Information Schema	Microsoft	29	4	116	0.24	2.07
SQL Server	Northwind database orders table	Microsoft	830	14	11,620	1.14	0.10
TOTAL			22,137,043	174	134,497,669		

Figure 29. Analysis of runtime statistics

The HairEyeColor and WorldPhones datasets have very high unit run times. This can be explained by looking at their native data structures in R which are not the dataframe structure the function uses. Hence, there is an overhead involved in converting these datasets before processing.

Larger files have a much lower unit run time as read/write overheads become proportionately smaller elements of the processing task.

The function is efficient enough to be viable for production work, as can be seen from the time taken to process the large movie ratings datasets.

4.7 Implementation Quality

Rosenberg and Hyatt (1997) state that attributes on which software is judged to be of good quality are –

- Efficiency - Are the constructs efficiently designed?
- Complexity - Could the constructs be used more effectively to decrease the architectural complexity?
- Understandability - Does the design increase the psychological complexity?
- Reusability - Does the design quality support possible reuse?
- Testability/Maintainability - Does the structure support ease of testing and changes?

Considering the software presented in this project by these attributes –

Efficiency	The software has efficient run times as can be seen from run time analysis in Figure 29.
Complexity	The internal code of the Burrow function is not complex. The pattern of repeating a code fragment which calls a regex statement as seen in Figure 26 forms the main body of the code.
Understandability	The code is understandable. Regex expressions are somewhat terse and can take some learning, but they are widely used and very well documented.
Reusability	The code is reusable in the sense that it could be easily ported to other software environments that have regex support such as Python or .NET, there are no proprietary elements. Because the output from the Burrow function is easily writable to a CSV file it is easily consumable by other programs in addition to the R markdown document presented in this project.
Testability	As it has a modular design the software is amenable to testing. Each feature extracted is independently testable by confirming the validity of the regex expression that is at its core
Maintainability	The software is easy to both change and extend. Additional feature detection can be easily added by developing the desired regex expression and adding a few lines of code following the pattern as shown in Figure 26. As the output of the Burrow function is structured as a row per feature test it is infinitely extensible.

As the software satisfies these engineering principals, the resulting conclusion is that it is of good quality both in terms of design and implementation efficiency.

4.8 Visualising the dataset

The complete result of the markdown report for the synthetic development data and the iris dataset are given in Appendix F. The methods section gave examples of outputs and showed the level of customisation that could be obtained using the markdown format. The results obtained from using markdown were very satisfactory, but it would also be interesting to see what could be achieved using other visualisation systems.

4.9 Incidental Results

Outside the scope of the original project, but an interesting by-product from it is the potential for EDA given the paradigm offered by the project where many statistics are automatically collected prior to analysis rather than a few statistics generated by it. A limitation of what has been presented so far is that the only input parameter is the source data. This was part of the project design as the purpose was to test the fundamental hypothesis.

However, often in analysis what is required is to compare groups of data on the same basis to identify differences. In the iterative nature of EDA this leads to questions about why these differences should be.

As the project progressed the realisation dawned that the Burrow data can provide a useful tool to easily compare different groups of data on many dimensions, far more than would be realistic in a traditional EDA analysis. Two examples are presented below.

Figure 30 was created by running the Burrow function for the three different Species of the iris dataset. The outputs were merged to give a dataset of the same information for all three groups, and then a scatterplot created comparing numeric variables on dimensions of means, max, min and median. From this visualisation we can observe such things as the range of values for Sepal Width is greater for setosa than for versicolor, for Petal Length the range of values is greater for versicolor than setosa. Mean and Median overlap slightly differently for Petal Length in both species.

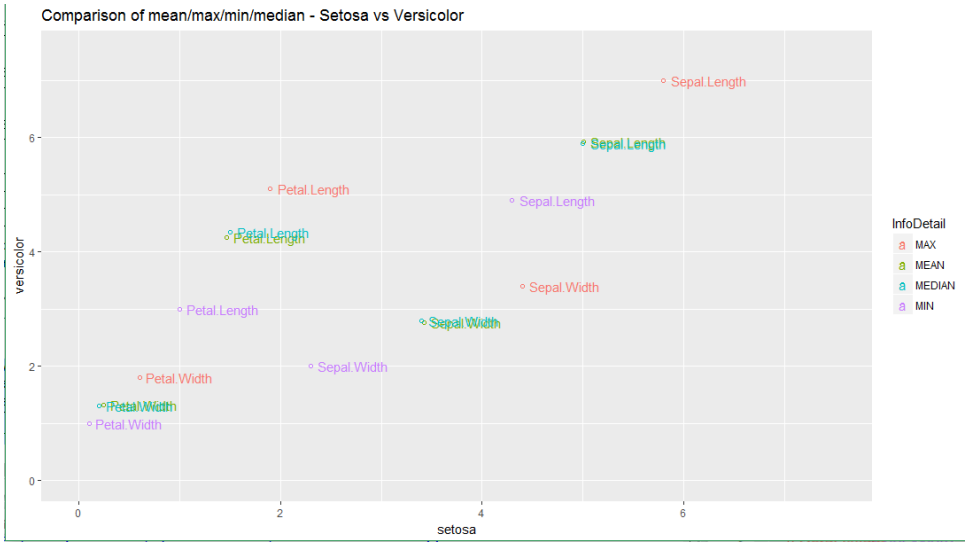


Figure 30. Visualising the difference of variables between categories

The second example is taken from the two UCI School data files used as part of the test datasets. Data is from a social survey in which each school returned questionnaires about pupil performance. This data was used during the taught part of the course for the Machine Learning coursework. EDA was conducted on these datasets as part of the coursework, and the takeaway experience at the time was, to paraphrase Schutt and O'Neil (2013), that it didn't offer much. At the end of the coursework it was possible to look back with machine learned hindsight and see patterns in the data that might have been identified earlier had a more thorough EDA process been followed, but it would have been a laborious task to look across all the possible statistics and identify the areas of difference. The desire to reduce scripting work was one of the motivations for the project, but an incidental result is that a technique of "collect everything and scan for differences" has advantages.

Figure 31 shows the difference between the means of numeric variables for each the schools calculated from the Burrow data. Intuition from this visual suggests the datasets are unbalanced for some reason, and the Burrow data highlights the absences and failures variables as areas for further consideration.

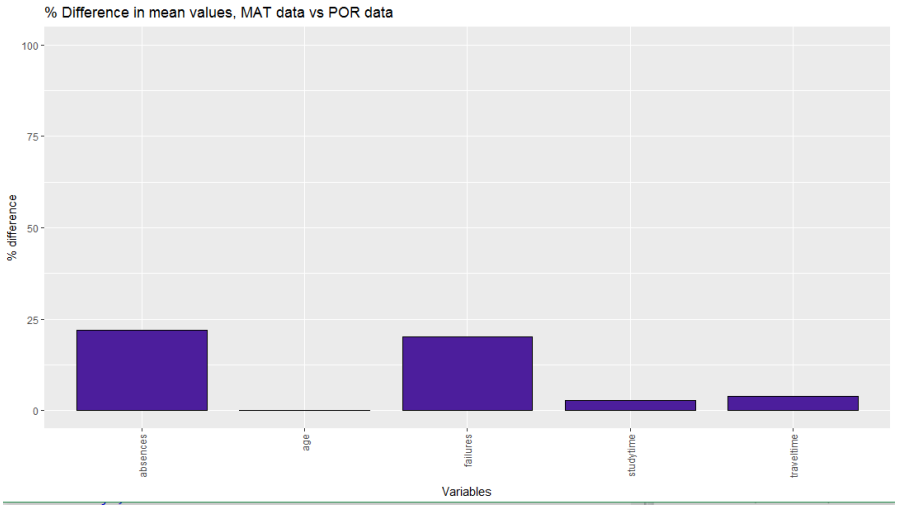


Figure 31. Comparing the mean values of numeric variables for the Schools survey datasets

4.10 Conclusion

Using the Burrow function to create generalised, structured information about a dataset allows a determination to be made about the data type with a good degree of accuracy. This allows a logical visualisation framework to be implemented as a single R markdown document which will customise itself to the input dataset.

The techniques used are extensible, easy to understand, robust and portable to other software platforms.

The meta data has potential uses beyond the production of a report as a tool to compare groups of data, highlighting differences can be identified for further study.

Chapter 5 – Discussion

5.1 Evaluation of objectives

5.1.1 Taxonomy of data types

The Taxonomy of data types created was sufficiently detailed to provide an effective framework for the prototype development, and it gives a good indication of the direction future work could take. If the project were to progress beyond the prototype stage, the best strategy to grow the taxonomy would be from feedback by users from different domains working on live projects.

5.1.2 Prototype to extract features, classify data type

This objective was very well met, and the result is very satisfying.

There was an initial steep learning curve developing and testing regex expressions and this approach was almost abandoned, but once a mechanism for building reliable pattern expressions had been found, the strategy of using regex to obtain features could be tested and was found to be extremely quick, robust and easy to implement. In turn, once feature extraction became quick and robust, developing the classification model became quite easy. Obtaining an accuracy of 91% on unseen testing data from development using only the iris and the synthetic dataset was a very satisfying result.

The debuggex.com website was a big factor in both the success of the project and in fulfilling a long time personal goal of being able to confidently use regex. There are many situations in which pattern matching expressions can be very useful. Once developed patterns can be reused in any software that supports regex. For instance the regex used by the Burrow function to recognise UK postcode, was originally sourced from www.gov.uk although the weblink had disappeared by the time the report was written so cannot be given as an attributable source.

More work could have been done at this stage to refine the classifier, including investigating re-engineering it, possibly as a machine learning decision tree. However, it was judged that 91% accuracy was more than sufficient to prove the concept, and that the project should move on to investigate its other objectives.

5.1.3 Saving the Meta data

The Tidy data format used to save the Burrow data worked really well. During the sandbox prototyping phase, quite some time was spent working out a system of tagging each data item so that the Burrow file would be easily subset-able by building search terms based on tags. This gave a lot of flexibility at later stages of the project and, as explained in the results section, offers potential for comparison of classes of data during EDA, although this was beyond the scope of the project.

Having a notes tags as part of the Burrow data is another way this data extends its usefulness beyond the original intention. Literature or urls offering explanations of the calculations behind the statistics collected can be made integral to the meta data. This offers the potential for analysts to learn about the statistics collected and makes the Burrow data self-referencing.

Of all the outcomes of the project this is the area which is the most successful as it has numerous applications, and the technique itself is applicable to other projects.

5.1.4 Descriptive statistics and visualisations appropriate to data types

This was a disappointing area of the project in terms of the range of examples produced, however sufficient work was done to reliably demonstrate the concept. A more complete result could have been obtained had a wider range of post-classified statistics and visualisations been researched and included. No technical obstacles to doing this were identified, the issue was balancing the time required for research and to then generalise the code for adding each statistic and visualisation, with the overall work required to complete the other parts of the project.

5.1.5 Automated generation of descriptive statistics report

The creation of the report turned out to be the least taxing part of the work, due to the powerful resources available in R meaning that minimal development was required. The highly versatile ggplot2 library was a great tool for creating visualisations, and working with it was another good learning opportunity. A powerful aspect was the ability to create a function containing code for a generalised visualisation, then to pass the specific data and titles to the function.

This project objective was met, although the sample report presented in Appendix F should be considered as proof-of-concept standard. If the software were to be released as a live package, user feedback on this section would be required. Confidence that the software could be developed to a live standard can be found from the wide usage of markdown and ggplot2 as reliable robust software tools.

5.2 System design and coding

The development was well managed, due in large part to the adoption of the sandbox prototyping model which allowed a well-constructed work plan to be scoped. Although it can seem counter-intuitive to work on a piece of software with the knowledge that none of it will be used, in practice it can be quite liberating. There are several overheads involved when writing production code, for instance commenting, strict adherence to a system of naming conventions and identifying supporting functions to name but a few. Sandboxing removes these constraints.

Designing the implementation to have two separable parts which passed data to each other not only make checking of each stage easier, but gave an open structure which can be built upon.

The choice of R as a development platform was good one, although doubtless Python would have been equally successful. Leveraging data i/o routines, internal structures of R, ggplot2 and markdown saved development work that would have been required in a less feature rich platform.

5.3 New learning

The technique of extracting features about a variable and classifying data type was not seen anywhere in the literature search. This technique could be useful for other applications in the data science community.

In a talk given by Max Kuhn, author of the caret package, on 3rd October 2016; a side topic discussed was "Why analyse, why not just put everything in a decision tree?". The iterative process that EDA follows could be regarded as a manual decision tree, with the most informative statistics at each stage leading to the next. As a manual process, it is difficult to always find the informative paths through the data. A new idea generated by the project is that EDA might be inverted with many statistics collected up front prior to any human activity and then examined for analytical direction.

Personal learning was skill development in R, regex, ggplot2 and markdown all of which have wide applications.

5.4 Answering the Research Question

The research question was answered positively. It is, indeed, possible to automatically determine the data type of a variable, and the determination is good enough to allow prediction of appropriate descriptive statistics.

Chapter 6 – Evaluation, Reflections and Conclusions

6.1 Choice of objectives

The objectives of the project proved to be somewhat over-ambitious. As the main limitation was time available rather than any technical stumbling blocks, the project provides a viable proof-of-concept for future work.

The central objective supporting the project is the development of the Burrow classification model, which was successful. The success stems more from the technique used to extract feature information than the classification function which was not implemented in a very sophisticated fashion.

The Taxonomy of data types created was slightly disappointing, as the literature search proved to be unsatisfying. Reflecting on why this should be leads to a consideration of the term "data type", and suggests it is too simplistic for what the project seeks to accomplish. The taxonomy in its current form might be better considered as a proposed system of "type encoding for dataset variables". A somewhat longwinded term, but even slightly different phrases such as "variable type encoding" introduce ambiguities.

The reporting system demonstrates how the results of the project can be put into practical use and demonstrates how much human time could be saved by the methods presented here. Building up standard chart libraries for defined subsets of Burrow features which could be shared between analysts would also offer efficiencies.

6.2 Limitations of the project

As the project aims to be of practical use to analysts and data scientists, a usability survey is required to confirm that the work presented here would receive a favourable reaction. If this proved to be the case, then observing user's behaviour with the system would provide an interesting perspective on the direction further development could take. The paper by Kandel et al. (2012) referred to in the literature search would provide a good basis for study design.

6.3 Future work

Increasing the parameters supplied to the Burrow function to include target variable(s) would make it infinitely more useful as demonstrated in the results section. As previously noted, the range of statistical tests reported could be expanded.

R has libraries for reading semi-structured data such as JSON, XML, and these file types could be included in the scope of the project once read into a structured dataframe.

Features such as variable name could be added to the classifier, adding semantic refinements to the features. For instance, a text variable called First name with a short character length is classifiable far more reliably than a text variable called A12 with a short character length.

The markdown report could be developed, or even replaced by some other system. As the stages of the project are standalone there is no requirement to develop it all in R. A user survey and wider involvement in development of a standard reporting framework would be useful.

6.4 Reflection on the project

The literature survey reviewed the work done in automating data science processes, and questioned whether this project could contribute anything in this area. Tukey and others warn against regarding EDA as a mechanistic process, and hopefully it is clear that this is not the intention of the project. However, there is potential to automate the discovery and presentation of information to the user that might guide their analysis. By delivering appropriate sources of learning with the results the user has the context to evaluate the results and develop their analytical intuition as time goes by. If the rise of citizen data scientists is as sharp as predicted, then more structured EDA with built in directed learning will be helpful in raising quality and integrity of analysis.

A final thought is that the Burrow function could best be implemented as an API, offering a service to analysts and data scientists regardless of software platform.

References

Amar, R., Eagan, J. and Stasko, J., 2005, October. Low-level components of analytic activity in information visualization. In *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on* (pp. 111-117). IEEE.

Berthold, M. and Hand, D.J., 2003. *Intelligent data analysis: an introduction*. Springer Science & Business Media.

Cardelli, L. and Wegner, P., 1985. On understanding types, data abstraction, and polymorphism. *ACM Computing Surveys (CSUR)*, *17*(4), pp.471-523.

Chapman, C. and Feit, E.M., 2015. *R for marketing research and analytics*. Springer.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C. and Wirth, R., 2000. *CRISP-DM 1.0 Step-by-step data mining guide*. CRISP-DM consortium

CITO Research, 2017. *Why Only Fully Equipped Citizen Data Scientists Will Succeed*. [ONLINE] [Accessed 18 February 2017], Available at : http://www.citoresearch.com/data-science/why-only-fully-equipped-citizen-data-scientists-will-succeed

DeLine, R. and Fisher, D., 2015, October. Supporting exploratory data analysis with live programming. In *Visual Languages and Human-Centric Computing (VL/HCC), 2015 IEEE Symposium on* (pp. 111-119). IEEE.

Dudovskiy, J., 2016. *Research Methodology*, [ONLINE], [Accessed 15 February 2017], Available at *:* http://research-methodology.net

Dunning, B., 2017. *Free sample data for testing,* [ONLINE], [Accessed 11 November 2016], Available at : https://www.briandunning.com/sample-data/

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M. and Hutter, F., 2015. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems* (pp. 2962-2970).

Gelman, A. and Unwin, A., 2013. Infovis and statistical graphics: different goals, different looks. *Journal of Computational and Graphical Statistics*, *22*(1), pp.2-28.

Gruber, J., 2017. *Daring Fireball*, [ONLINE], [Accessed 17th February 2017], Available at *:* http://daringfireball.net/projects/markdown/

Gulati, P. M., 2009. *Research Management: Fundamental & Applied Research*. Busca Inc.

Hand, D.J., 1993. Data, metadata and information. *Statistical Journal of the United Nations Economic Commission for Europe*, *10*(2), pp.143-151.

Harvard Biomedical Data Science – Github pages, 2017. *Genomicsclass*. [ONLINE], [Accessed 17 February 2017], Available at : http://genomicsclass.github.io/book/pages/exploratory_data_analysis.html

Kanchan, T., Krishan, K., 2013. Significance of descriptive statistics in forensic anthropology research, *Journal of Forensic and Legal Medicine*, Vol 20, page 1151.

Kandel, S., Heer, J., Plaisant, C., Kennedy, J., van Ham, F., Riche, N.H., Weaver, C., Lee, B., Brodbeck, D. and Buono, P., 2011. Research directions in data wrangling: Visualizations and transformations for usable and credible data. *Information Visualization*, *10*(4), pp.271-288.

Kandel, S., Paepcke, A., Hellerstein, J.M. and Heer, J., 2012. Enterprise data analysis and visualization: An interview study. *IEEE Transactions on Visualization and Computer Graphics*, *18*(12), pp.2917-2926.

Kaur, P., Owonibi, M. and Koenig-Ries, B., 2015. Towards Visualization Recommendation-A Semi-Automated Domain-Specific Learning Approach. In *GvD* (pp. 30-35).

KDnuggets, 2016. *Data Science Analytics Software Poll*, [ONLINE], [Accessed 17 February 2017], Available at: http://www.kdnuggets.com/2016/06/r-python-top-analytics-data-mining-data-science-software.html.

Khabaza, T., 2010. *Nine Laws of Data Mining.* [ONLINE], [Accessed 17 February 2017], Available at : http://www.socdm.org/index.php/methodologies/21-methodologies-9-laws.

Kuhn, M. and Johnson, K., 2013. *Applied predictive modeling* (Vol. 26). New York: Springer.

Kuhn, M., 2016. *The caret package*, [ONLINE], [Accessed 21 February 2017], Available at : http://topepo.github.io/caret/index.html

Lloyd, J., 2017. *The Automatic Statistician*, [ONLINE], [Accessed 28 February 2017], Available at : https://www.automaticstatistician.com/index/

Locke, S., 2017. *CRSIP-DM and why you should know about it*, [ONLINE], [Accessed 17 February 2017], Available at : https://www.r-bloggers.com/crisp-dm-and-why-you-should-know-about-it/.

Microsoft, 2017. *Excel specifications and limits*, [ONLINE], [Accessed 20 February 2017], Available at : https://support.office.com/en-gb/article/Excel-specifications-and-limits-1672b34d-7043-467e-8e27-269d656771c3

Munzner, T., 2009. A nested model for visualization design and validation. *IEEE transactions on visualization and computer graphics*, *15*(6).

National Institute of Standards and Technology, 2017. *Engineering Statistics Handbook*, [ONLINE], [Accessed 21 February 2017], Available at : http://www.itl.nist.gov/div898/handbook/index.htm

O'Reilly. 2015. *O'Reilly Data Science Salary Survey*, [ONLINE], [Accessed 21 February 2017], Available at : https://www.oreilly.com/ideas/2015-data-science-salary-survey

Pandey, A.V., Rall, K., Satterthwaite, M.L., Nov, O. and Bertini, E., 2015, April. How deceptive are deceptive visualizations?: An empirical analysis of common distortion techniques. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (pp. 1469-1478). ACM.

Rosenberg, L.H. and Hyatt, L.E., 1997. Software quality metrics for object-oriented environments. *Crosstalk journal*, *10*(4), pp.1-6.

RStudio, 2016. *R Markdown,* [ONLINE], [Accessed 26 February 2017], Available at *:* http://rmarkdown.rstudio.com/index.html

Schutt, R. and O'Neil, C., 2013. *Doing data science: Straight talk from the frontline*. " O'Reilly Media, Inc.".

Simonite, T., 2015. *Automating the Data Scientists*. MIT Technology Review

Stevens, S.S., 1946. On the theory of scales of measurement. Science, New Series, Vol. 103, No. 2684 (Jun. 7, 1946), pp. 677-680 Published by: American Association for the Advancement of Science

Teach-ICT.com, 2017. [ONLINE], [Accessed 26 February 2017], Available at *:* http://www.teach-ict.com/ks3/year7/data_handling/miniweb/pg7.htm

Toarca, S., *Debuggex.com*, 2017. [ONLINE], [Accessed 26 February 2017], Available at *:* https://www.debuggex.com/

Tukey, J.W., 1977. Exploratory data analysis.

Tukey, J.W., 1980. We need both exploratory and confirmatory. *The American Statistician*, *34*(1), pp.23-25.

University of Cambridge, Computational and Biological Learning, 2014. *Google announces $750,000 for "The Automatic Statistician"*. [ONLINE], [Accessed 17 February 2017], Available at : http://mlg.eng.cam.ac.uk/?p=1578

Wickham, H., 2013. *ggplot2,* [ONLINE], [Accessed 17 February 2017], Available at : http://ggplot2.org/

Wickham, H., 2014. Tidy data. *Journal of Statistical Software*, *59*(10), pp.1-23.