

Number 920



**UNIVERSITY OF
CAMBRIDGE**

Computer Laboratory

Interactive analytical modelling

Advait Sarkar

May 2018

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500
<http://www.cl.cam.ac.uk/>

© 2018 Advait Sarkar

This technical report is based on a dissertation submitted December 2016 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Emmanuel College.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

<http://www.cl.cam.ac.uk/techreports/>

ISSN 1476-2986

ABSTRACT

This dissertation addresses the problem of designing tools for non-expert end-users performing *analytical modelling*, the activity of data analytics through machine learning. The research questions are, firstly, *can tools be built for analytical modelling?* Secondly, *can these tools be made useful for non-experts?*

Two case studies are taken: that of building and applying machine learning models to analyse time series data, and that of building and applying machine learning models to analyse data in spreadsheets. Respectively, two prototypes are presented, Gatherminer and BrainCel. It is shown how it is possible to visualise general tasks (e.g., prediction, validation, explanation) in these contexts, illustrated how these prototypes embody the research hypotheses, and confirmed through experimental evaluation that the prototypes do indeed have the desirable properties of facilitating analytical modelling and being accessible to non-experts.

These prototypes and their evaluations exemplify three theoretical contributions: (a) visual analytics can be viewed as an instance of end-user programming, (b) interactive machine learning can be viewed as dialogue, and (c) analytical modelling can be viewed as constructivist learning. Four principles for the design of analytical modelling systems are derived: begin the abstraction gradient at zero, abstract complex processes through heuristic automation, build expertise through iteration on multiple representations, and support dialogue through metamodels.

ACKNOWLEDGEMENTS

I'd like to thank:

- My sponsors: EPSRC, British Telecommunications plc, and Robert Sansom, for generously supporting my research for the last three or so years.
- Lise Gough, for her help and support in innumerable and immeasurable ways.
- Members of the Rainbow group who made working in the SS corridor so engaging and enjoyable: Alistair, Zhen, Isak, Maria, Mariana, Marwa, Flora, Tadas, Andra, Sam, and many others.
- Co-conspirators from the Happy Hour crew past and present: Erroll, Oli, Guo, and others, for making Fridays awesome.
- Steve, Tom, and Becky, for some awesome Apprentice/Masterchef evenings.
- Collaborators and mentors at Microsoft Research Cambridge and Novartis, including Cecily Morrison, Jonas Dorn, Abi Sellen, Simon Peyton Jones, and Andy Gordon, for two incredibly stimulating and enjoyable summer research internships.
- All of my hardworking experimental participants, from BT Research, the Computer Lab, and the Land Economy department.
- Dyuti kaku and Suresh uncle, for generously hosting and taking good care of me in Melbourne and Palo Alto respectively, when I attended VL/HCC '14 and CHI '16.
- Guy Cuthbert and Atheon, for uncovering the data scientist in me.
- The Vista crowd (families Ramkumar, Shekhawat, Ravikumar, and others) for being a wonderful, joy-filled part of my Bangalore life.
- The Edingale crowd (Wells', Keeys, and others), but especially the wonderful Mason family, for taking care of me on countless occasions.
- Dr. Matthew Sullivan, for teaching me to never settle for any less than the very best.
- My supervisors: Alan Blackwell, Mateja Jamnik, and Martin Spott, thanks to whom I have grown beyond my wildest imagination. Martin: thank you for your guidance, support, and many enjoyable car journeys to Ipswich. Mateja: thank you for your compassion, clarity of vision, and immense generosity with your time. Alan: thank you for your generosity with your amazing repertoire of knowledge, and for freely and exponentially multiplying and elevating my meagre ideas.
- My beloved John, for his steadfast and unconditional love and support.
- My grandparents: dadu, dida, dada, and thakuma.
- My mother and father, Nivedita and Subhanjan, for their endless, selfless love.

CONTENTS

1	Introduction	15
1.1	Research objectives	17
1.2	Dissertation overview	18
1.3	Relevant publications	19
2	Humans in the analytics loop	21
2.1	Introduction	22
2.2	Current approaches to mixed-initiative analytics	22
2.2.1	Intelligent discovery assistants	22
2.2.2	Visual analytics	24
2.2.3	Interactive machine learning	25
2.3	Differences in approaches to interactive analytics	26
2.3.1	Technology	26
2.3.2	End-user task formalisation	27
2.3.3	Approach to knowledge generation	27
2.3.4	Problem domain	28
2.4	Interactive analytical modelling	29
2.4.1	The two cultures of statistical modelling	29
2.4.2	Novel research focus	30
3	Theoretical perspectives	33
3.1	Research methodology	34

3.2	Visual analytics is end-user programming	36
3.3	Interactive machine learning is dialogue	43
3.3.1	Other approaches to the indirect channel	44
3.3.2	Supporting dialogue through metamodels	46
3.3.2.1	Example metamodel applications	51
3.4	Analytical modelling is constructivist learning	54
3.4.1	Implicit and explicit learning outcomes in analytical modelling	55
3.4.2	Critical constructivist issues for analytical modelling	55
3.5	Design principles for analytical modelling systems	58
3.6	Conclusion	60
4	Gatherminer	61
4.1	Introduction	62
4.2	Related work	63
4.2.1	Visualisations for bottom-up time series analysis	64
4.2.2	Explaining behaviour in time series datasets	64
4.3	Design	65
4.3.1	Core colour-mapped matrix visualisation	65
4.3.2	Gathering: automated layout	72
4.3.3	Selection to annotate interesting clusters	78
4.3.4	Alternative interactive representations for decision trees	82
4.4	Comparative study	87
4.4.1	Experimental results	88
4.5	Discussion	89
4.6	Conclusions	94
5	BrainCel	97
5.1	Introduction	98
5.2	Selection as annotation	99
5.2.1	Experimental evaluation of selection-as-annotation	102

5.2.2	Selection-as-annotation experiment results	102
5.3	Supporting dialogue and critical model evaluation	105
5.3.1	Design of the BrainCel interface	106
5.3.2	Design discussion	114
5.4	Exploratory user study	114
5.4.1	Analysis method	116
5.4.2	Exploratory study results	116
5.4.3	Activity flows	119
5.5	Limitations and future work	120
5.6	Conclusions	121
6	Conclusion	123
6.1	Theory exemplified in Gatherminer and BrainCel	123
6.2	Contributions	125
6.3	Future work	126
6.4	Conclusion	128
	Bibliography	129

In loving memory of dadu.



*Dr. Jayant Mukerji
(1933 – 2011)*

INTRODUCTION

Data analytics is useful, that much is certain. However, the tools used to conduct sophisticated analytics, involving statistics and machine learning, are still primitive. The interfaces of standard professional tools are programming languages. We do not burden professional photographers, musicians, doctors, lawyers, teachers, etc., with needing to learn how to program in order to do their job, yet programming remains the dominant activity of modern statisticians and data scientists. Consequently, data analytics is the preserve of a small group of highly-skilled professionals, despite the fact that a much larger population could benefit from access to sophisticated analytics.

“Limited access to Big Data creates new digital divides”, assert Boyd and Crawford (2012). Access to data consists not only of the ability to physically read the data, but also the availability of tools for processing that data, and the ability to acquire sufficient knowledge and expertise to profitably operate those tools.

As the amount of data we produce and consume grows, so does its potential utility. It is not only the professional data analyst working within a corporate setting who might benefit from this process, but anyone who manages information systematically. This is a diverse group with heterogeneous needs and abilities, ranging from education administrators, to journalists, from small business owners, to homemakers, from students, to doctors. Our collective ability to use data is not growing proportionally to its potential. Instead, the domain knowledge and tool expertise required to put data to use is increasingly concentrated in a small number of professional data analysts, sometimes referred to as data scientists. Data scientists typically have a skill set that includes but is not limited to statistics, machine learning, programming, database management, data visualisation, and communication.

While small businesses and individuals may benefit demonstrably from analysis of their data, it is infeasible for many to engage the services of data scientists. Better-designed tools may ameliorate this unfulfilled need. Software tools cannot replace the data scientist entirely, but the usability-driven design of such tools might drastically lower the barrier to entry, making some aspects of data science accessible to users not in possession of extensive domain knowledge or tool expertise.

The space between analytics and model-building

Statistics and machine learning have two uses:

1. **Analytics:** interpreting data. The aim of analytics is to understand something about the nature of the real world by studying observed data in relation to its context. Descriptive summary statistics such as the mean, variance, and correlation coefficient, are simple instances of analytical instruments.
2. **Model-building:** capturing the structure in observed data in a model which enables inference on incomplete or as-yet-unobserved data. For instance, modelling the relationship between two variables using the equation of a line with linear regression is a simple instance of model-building, as the fitted coefficients can be used to estimate the response variable from the explanatory variable if for a particular case the former is unknown but the latter is known.

Advances in machine learning and computational power have stimulated both uses. Complex models with many parameters, such as neural networks and ensembles of decision trees, are popular for their superior performance in prediction tasks. The overlap between analytics and model-building is growing, as analysts seek to interpret the learnt parameters and coefficients of these models in the way they would simple descriptive summary statistics. However, tools available for analytics and model-building remain separate, as there is a perceived tradeoff between accuracy and intelligibility, detailed in the next chapter. This dissertation addresses *analytical modelling*: the hybrid activity of analytics through model-building.

Expertise requirements for analytical modelling

Analytical modelling requires much expertise. Many algorithms and tools are usable only by experts in both programming and machine learning. This dissertation identifies three types of expertise required for analytics and model-building:

1. **Representational expertise:** an understanding of how interface elements map to underlying abstract information structures and information manipulation procedures. Representational expertise generalises *notational expertise* (Stead and Blackwell, 2014), which refers to expertise in programming language syntax. To use a programming language, one must understand its syntactic features: identifiers, delimiters, and so on. Similarly, to use a graphical interface, one must understand how buttons, menus, and other interface elements each modify the information structure being created.

In the common scenario where a programming language is the sole interface to an analytical procedure, representational expertise requirements are immense; not only must the user be an expert in the abstract symbolic manipulation of the programming notation, but also in the idiosyncrasies of particular software libraries. For instance, to perform a linear regression in R, one must know the syntax for creating a data matrix in R, the command for fitting a linear regression including the structure of its arguments, and how to display its results.

2. **Process expertise:** an understanding of a statistical process to be invoked in terms of its assumptions, limitations, algorithmic properties, and the interpretation of its outcomes. For instance, in order to understand the process of linear regression, one must understand the concept of explanatory and response variables, the concept of the equation of a line, the concept of maximum likelihood estimation through a training error minimisation process such as the method of least-squares, and so on.
3. **Domain expertise:** an understanding of the data being processed, how it was generated, its sources of noise, and how to interpret it in the context of broader real-world processes. For instance, analysing time series data of faults on a telecommunications network, a problem addressed in Chapter 5, requires an understanding of the real-world processes and devices that this data describes.

Each type of required expertise poses a barrier to analytical modelling. The term *non-expert end-users* is used throughout this dissertation to refer to the target users of the tools presented. In general, this means that the users lack sufficient representational expertise to use a textual programming language such as R, and also lack sufficient process expertise to be able to thoroughly understand the mathematical theory underlying statistical modelling. It is in these first two types of expertise that there is clearest opportunity for interaction design solutions; a principal concern of interaction design is the study of representation and metaphor to enable humans to do things and have experiences that they otherwise could not, by identifying and reducing the incidental complexities of tasks.

In some parts of this dissertation the term *non-expert* additionally implies a lack of domain expertise; in these cases the departure from the standard interpretation is explicitly demarcated. Domain expertise can improve as well as impair analysis. The relationship between interaction design and domain expertise is the least clear, and well-designed interfaces may not ever be able to compensate for a lack of domain expertise, since the analyst requires it to interpret the data in context. However, sometimes domain expertise is used not to contextualise, but as a (potentially problematic) search heuristic. This causes issues for controlled usability studies. It will be shown how interaction design can intervene here, principally by reframing the task as one other than search. The issues surrounding domain expertise are addressed in detail in Chapter 4.

1.1 Research objectives

This dissertation investigates the following research questions:

1. Can tools be built for analytical modelling, that is, analytics through model-building?
2. Can they be made useful for non-experts?

To answer these questions, an iterative, design-led approach has been taken, motivated by real-world scenarios where an analytical modelling tool would be useful for non-experts. In building prototypes to explicitly facilitate analytics through model building, this dissertation exposes the visual and interaction design challenges inherent in attempting to effectively combine them. In particular, the two major challenges are reducing expertise requirements, and facilitating critical engagement with a model. Novel visual representations, theoretical constructs, and design principles are proposed to address these challenges. These prototypes are demonstrated to be useful for non-experts through studies conducted with non-expert participants.

1.2 Dissertation overview

The remainder of this dissertation is organised as follows.

Chapter 2: Humans in the analytics loop

This chapter summarises the research addressing the question: *what is the best way to involve humans in data analytics and machine learning?* A spectrum of answers emerges from the literature. For instance: ‘intelligent discovery assistants’ (IDAs) chart out the ‘space’ of valid and possible analyses (composed of a sequence of atomic operations) that can be explored automatically, manually, or by a mixture of both. Visual analytics (VA) systems integrate analytical processes and visualisation, on the premise that human visual perception can act as a heuristic to guide the search through this space of analyses. Finally, interactive machine learning (IML) is concerned with solving real-world problems where models are trained by non-experts to fit their particular requirements. In each of IDAs, VA, and IML, we observe to different degrees a ‘mixed-initiative’ interaction, where the user and system work together in a non-trivial way to achieve some goal. This chapter summarises each of these fields, their respective research foci, and makes explicit some of the differences between their approaches. Finally, this chapter addresses in greater depth the concept of *analytical modelling* as a novel research focus.

Chapter 3: Theoretical perspectives

This chapter describes the approach of this dissertation to theory and research in interaction design. It begins by outlining *research through design*, the primary research methodology of this dissertation. It discusses the difficulties of producing general prescriptive theory with implications for design, and clarifies the position of the theory presented as intermediate-level knowledge which annotates the design portfolio of this dissertation. The first theoretical perspective, that visual analytics is end-user programming, is introduced. The next theoretical perspective, that analytical modelling is dialogue, refines the view of interactive machine learning as end-user programming, and highlights a difficulty with that interpretation which is problematic for analytical modelling. As a conceptual solution, interaction is proposed to be more like *dialogue*, the key aspect of which is that neither party, human nor computer, is designated arbiter of truth. As a technical solution, metamodels of machine-learned models are proposed. The third theoretical perspective is that interactive analytical modelling facilitates constructivist learning, as the outcomes of interactive analytical modelling can be characterised as learning outcomes, and the iterative interaction loop at the centre of interactive analytical modelling creates an excellent opportunity for interaction between the user’s ideas and their experiences. Four design principles derived from these perspectives are then introduced.

Chapter 4: Gatherminer

This chapter describes the design and implementation of *Gatherminer*, a visual analytics tool for analysing time series data by building decision trees to explain observed patterns. The problem is first introduced, grounded in a real-world scenario of network fault analysts at BT Research & Technology. The related work on time series analysis is described. The design and implementation of *Gatherminer* is described. Next, a comparative study demonstrating the improvement in analytic insight using *Gatherminer* when compared to an industry standard tool is reported. Finally, the chapter closes with a discussion of the results and reflections on the difficulties of conducting controlled experiments when domain knowledge can be a confounding factor.

Chapter 5: BrainCel

This chapter describes the design and implementation of *BrainCel*, a tool for building and applying machine learning models in spreadsheets. The chapter first introduces the problem, before detailing how selection-as-annotation is used in BrainCel. An experiment is reported evaluating the usability of one particular implementation of selection-as-annotation, and demonstrating its learnability by non-experts. The interface is extended with multiple visualisations to facilitate critical engagement with the model, and evaluated in a study analysing the learning barriers and information needs encountered in its use. The chapter concludes by outlining the limitations in the current work and opportunities for future work.

Chapter 6: Conclusion

This chapter concludes the dissertation. Presented first is a brief discussion of the design principles described in Chapter 3: their application in Gatherminer and BrainCel, and their implications for future research. The research questions and motivation are recapitulated, the contributions of the dissertation are summarised, and the most promising avenues for future work are outlined.

1.3 Relevant publications

This dissertation presents research described in the following papers:

- **Visual discovery and model-driven explanation of time series patterns.** Advait Sarkar, Martin Spott, Alan F. Blackwell, Mateja Jamnik. *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (pp. 78–86). Highly commended paper award (honourable mention).
- **Constructivist Design for Interactive Machine Learning.** Advait Sarkar. *Proceedings of the 34th Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA 2016)* (pp. 1467–1475).
- **Interactive visual machine learning in spreadsheets.** Advait Sarkar, Mateja Jamnik, Alan F. Blackwell, Martin Spott. *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (pp. 159–163).
- **Spreadsheet interfaces for usable machine learning.** Advait Sarkar. *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (pp. 283–284).
- **Confidence, command, complexity: metamodels for structured interaction with machine intelligence.** Advait Sarkar. *Proceedings of the 26th Annual Conference of the Psychology of Programming Interest Group (PPIG 2015)* (pp. 23–36).
- **Visual Analytics as End-User Programming.** Advait Sarkar, Alan F. Blackwell, Mateja Jamnik, Martin Spott. *Psychology of Programming Interest Group Work-In-Progress Workshop 2015 (PPIG-WIP)*.
- **Teach and Try: A simple interaction technique for exploratory data modelling by end users.** Advait Sarkar, Alan F. Blackwell, Mateja Jamnik, Martin Spott. *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (pp. 53–56).

- **Hunches and Sketches: rapid interactive exploration of large datasets through approximate visualisations.** Advait Sarkar, Alan F. Blackwell, Mateja Jamnik, Martin Spott. *Proceedings of the 8th International Conference on the Theory and Application of Diagrams, Graduate Symposium, 2014 (DIAGRAMS 2014)*.

Further papers that address issues surrounding those in the dissertation:

- **Setwise Comparison: Consistent, Scalable, Continuum Labels for Machine Learning.** Advait Sarkar, Cecily Morrison, Jonas F. Dorn, Rishi Bedi, Saskia Steinheimer, Jacques Boisvert, Jessica Burggraaff, Marcus D'Souza, Peter Kontschieder, Samuel Rota Bulò, Lorcan Walsh, Christian P. Kamm, Yordan Zaykov, Abigail Sellen, Siân E. Lindley. *Proceedings of the 34th Annual ACM Conference on Human Factors in Computing Systems (CHI 2016) (pp. 261–271)*.
- **A Live, Multiple-Representation Probabilistic Programming Environment for Novices.** Maria Gorinova, Advait Sarkar, Alan F. Blackwell, Don Syme. *Proceedings of the 34th Annual ACM Conference on Human Factors in Computing Systems (CHI 2016) (pp. 2533–2537)*.
- **Clarifying hypotheses by sketching data.** Mariana Mărășoiu, Alan F. Blackwell, Advait Sarkar, Martin Spott. *18th Eurographics/IEEE VGTC Conference on Visualization (EuroVis), 2016*. Honourable mention.
- **Interaction with uncertainty in visualisations.** Advait Sarkar, Alan F. Blackwell, Mateja Jamnik, Martin Spott. *17th Eurographics/IEEE VGTC Conference on Visualization (EuroVis), 2015*.

Some of the work in this dissertation was presented at the following workshops:

- **Exploratory modelling tools for visual analytics.** Advait Sarkar. *Workshop on Visual and Cognitive Analytics. The 35th Annual International Conference of the British Computer Society's Specialist Group on Artificial Intelligence (SGAI), December 2015 (AI-2015)*.
- **Usability Issues in Mixed-Initiative Visual Analytics.** Alan F. Blackwell, Advait Sarkar. *Workshop on Visual Analytics – Present and Future of Human-Computer Interaction in Data Analytics. The 33rd Annual International Conference of the British Computer Society's Specialist Group on Artificial Intelligence (SGAI), 2013 (AI-2013)*.

CHAPTER 2

HUMANS IN THE ANALYTICS LOOP

This chapter presents the relevant background. Identified within it are three research communities pursuing a nontrivial combination of human and machine capabilities in data analytics. These are: intelligent discovery assistants, visual analytics, and interactive machine learning. Each approach is analysed and distinguished in terms of its foundations and methods. Finally, the dichotomy between analytics and model-building, which motivates this dissertation, is explained.

2.1 Introduction

Data analytics, at its simplest, is the interpretation of data. The technique applied need not be sophisticated. For instance, looking at prices on a menu to assess whether a restaurant is ‘expensive’ is an act of analysis, as it involves interpretation and contextualisation; the assignment of meaning over and above the quantities literally denoted. The phrase ‘data analytics’, however, conventionally connotes the use of statistical methods to model and interpret quantitative data; this is the sense in which it is used in this dissertation.

This dissertation is concerned specifically with *interactive* data analytics, that is, the practice of data analytics conducted with the aid of interactive computer software. This is in contrast to purely-manual or purely-automated analytics. Analytics can of course be conducted manually, without computers. The complete automation of analytics is also the subject of study (Livingston et al., 2001; Lloyd et al., 2014), a line of research which may be traced back to Lenat’s automated mathematician (Lenat, 1976).

It is further possible for analytics to be conducted using software in much the same way as it would be manually. For instance, replication of data tables for simple row and column additions in a spreadsheet. In such cases digitisation and automation may have greatly improved the speed and accuracy of the process, but have not directly added significant value to the result of the analysis.

This dissertation is concerned neither with purely-manual nor with purely-automated approaches, because a variety of applications require a non-trivial mix of automation and human judgment. The focus is *mixed-initiative* (Horvitz, 1999) interfaces: those that add significant value through automation and intelligent inference. Horvitz’ original formulation of ‘mixed-initiative’ is broadened here to mean any system that adds significant value through inferential, heuristic automation. In the context of data analytics, analytic insights that could not have been derived without the system are significantly valuable.

2.2 Current approaches to mixed-initiative analytics

2.2.1 Intelligent discovery assistants

Intelligent discovery assistants (IDAs) approach mixed-initiative data analytics from the perspective of Knowledge Discovery in Databases (KDD). An excellent survey is given by Serban et al. (2013). The problem being solved is the multiplicity of potential operations which can be performed on raw data in order to arrive at a particular insight. Each step of the analytics pipeline entails several decisions. The pipeline includes: identifying relevant subsets of raw data, cleaning the data by handling erroneous, noisy, missing values, etc., transforming the data into suitable input for a data mining process (for example, by representing images as a set of RGB histograms), feature selection, sampling, building statistical models with appropriate parameters and hyperparameters, and finally, model evaluation and interpretation.

Early IDAs, such as Consultant-2 (Craw et al., 1992) were expert systems that included sets of hard-coded data mining workflows, suggested based on heuristic characterisation of the data. The increasing diversity of statistical approaches necessitated a more modular approach, which gave rise to the modern form of IDA. At the core of modern IDAs is an explicit ontology of operations with pre- and post-conditions. The raw data and meta-data represent initial conditions, and the user expresses goal conditions (i.e., the output of the desired analysis, such as a fitted linear model, or hierarchical clustering), known as *desiderata*. A core research concern in IDAs is the automated construction of so-called *workflows*: sequences of operations that, starting from the initial conditions, achieve the desired postconditions. The reader may have noticed the isomorphism of this problem to those of AI planning or program synthesis, and these are indeed approaches investigated by the community to tackle the issue of generating valid workflows. However, more interesting challenges are those of ranking these workflows according to some measure of quality, such as training error (in the case of classification/regression problems), or execution time. In many such cases it is impossible to heuristically evaluate a candidate workflow without actually executing it. Some illustrative examples of IDAs are now presented.

AIDE, an early intelligent discovery assistant, is a ‘knowledge-based planning system’ (St Amant and Cohen, 1997). It works similarly to a partial hierarchical planner. AIDE maintains a library of around 100 general and specific hierarchical (mixed-detail) plans. Exploration is constructive and in the form of a dialogue, with the system recommending a set of operations to perform next, giving the user the option to review and override these suggestions. The result of the analysis is not simply a graph or summary but also the history of applied operations, hierarchically structured as goals and subgoals, which give the context for interpretation. When the plan hits on multiple alternatives, the decision is deferred to the user – this is called a ‘focus point’. The network of focus points is remembered. The authors evaluated the within-subject improvement AIDE gives on a data analysis task and found a significant improvement in correct answers.

IDEA is a system that, given a specification consisting of data and goals, is capable of synthesising/enumerating valid data mining workflows (Bernstein et al., 2005). These are composed of constituent techniques which are stored in an ontology with their input/output constraints as in AI planning. It can rank valid processes by speed, accuracy/error (through auto-experimentation), and comprehensibility. The authors conclude that the straightforward application of tools is insufficient for high performance, and application-specific knowledge related to data mining processes is also required. This knowledge can be added to the ontology by specialists.

Serban et al. (2013) organise IDAs along two dimensions: types of support (e.g. automatic single-step/multi-step KDD workflow generation/checking/repair, explanations for users, workflow execution) and types of background knowledge (e.g., available operators and operator metadata, input data metadata, predictive models for workflow generation, cached workflows). The authors identify 5 distinct categories of IDAs: expert systems, meta-learning systems, case-based reasoning systems, planning systems and workflow composition environments (WCEs). The authors conclude that future IDAs are best built upon WCEs to deal with an explosion of operators and deteriorating usability. Finally, the authors provide an idealised IDA specification and architecture. They lay emphasis on a collaborative approach to workflow design, where data mining workflows are easily shareable and reproducible.

This brief discussion of IDAs has been included for completeness, since they are a mature research approach to mixed-initiative data analytics. However, although the motivation for IDA research is human-centric, the research focus is ultimately algorithmic, the main issues being the automated generation and evaluation of data mining workflows. Usability by non-expert statisticians is not a core priority, and although the development of IDA-like tools with a focus on use by non-experts is an alluring avenue, it has not been explored in this dissertation for reasons which will shortly become apparent.

2.2.2 Visual analytics

Visual analytics (Keim et al., 2008) is the mixed-initiative approach to data analytics emerging from the Information Visualisation (InfoVis) research community. The problem being solved is the difficulty in interpreting the parameters, processes, and output of data workflows. The solution proposed by visual analytics is to exploit the powerful perceptual processing power of the human visual system. There are three interesting challenges here. The first is that of *perceptual* mapping; the design challenge of how marks and visual variables are most effectively (with considerations to aesthetics, speed, error-proneness) mapped to data variables, which proceeds by way of perceptual studies in the manner of Cleveland and McGill (1984). The second interesting challenge is that of *semantic* mapping; the design challenge of how these variables can create a representation which most effectively facilitates reasoning about the data workflow process being used and its outputs, in the tradition of Bertin (1983). The final challenge is that of *interactional* mapping; the challenge of designing appropriate interactions to apply and manipulate data workflow programs through these visualisations. For instance, a large amount of attention is devoted to how important classes of data manipulation such as overview, filter, and detail-on-demand (Shneiderman, 1996) are borne out in a particular graphical representation. Visual analytics systems are typically designed for users with deep domain expertise. Commonly, these users also have some statistical expertise. However, the visual nature of these systems commonly reduces the expertise barriers to applying data workflow procedures, which is the core property of interest in this dissertation.

Visual analytics is an extremely broad and longstanding field of research. Consequently it is difficult to highlight specific examples as part of this general overview. However, certain examples can illustrate the complexity of this field. For instance, the systems presented by Adrienko and Adrienko (2011) and MacEachren et al. (2011) clearly illustrate the visual analytics research process. A highly specific problem is identified. Algorithms are proposed, which are tightly coupled with an intended visual representation. The system is evaluated through a series of case studies, validated by domain experts.

Tableau¹ is a commercially successful modern visual analytics system. It is different from the typical visual analytics systems considered by the VA research community in the fact that it is completely generic; it is not built to facilitate analytics in any specific data domain. It is more like a highly advanced version of the charting tools provided by Microsoft Excel². Unlike the other examples, the research innovation in Tableau is in reducing the expertise required to create complex graphical mappings through what can be considered a visual programming language.

Despite the differences, the common theme running through these examples is that the aim of these systems is to facilitate some type of analytical *insight* in the user's mind, and the core design problems are those of perceptual, semantic, and interactional mapping.

¹<https://www.tableau.com/> (last accessed: April 29, 2018)

²<https://products.office.com/en-us/excel> (last accessed: April 29, 2018)

2.2.3 Interactive machine learning

Interactive machine learning (IML) (Amershi et al., 2011a) is the mixed-initiative approach to data analytics emerging from the HCI and end-user programming (EUP) research communities. It is a slight mischaracterisation to refer to IML as an approach to data analytics. More specifically, IML is of interest because it solves the same problem as visual analytics, namely, difficulty in interpreting the parameters, processes, and output of statistical procedures. However, the aims of IML are very different. The core aim of an IML system is to enable an end-user to build (i.e., program) a statistical model such as a classifier, for reuse in a specific scenario. Similar to the target end-users in this dissertation, the users of IML systems typically have no expertise in statistics or programming.

Two examples are now presented to illustrate the scope and nature of IML systems. The first example is *Crayons* (Fails and Olsen Jr, 2003). *Crayons* enables end-users to build image segmentation classifiers, that is, pixel-level binary classifiers which segment portions of an image as falling into one of two classes. For example, a ‘human detector’ classifier would take a 2D image of size $w \times h$ as input, and as output, produce $w \cdot h$ binary labels, one for each pixel, corresponding to whether or not the pixel is part of a human in the image. To build such a classifier in *Crayons*, users paint labels on an image as they would using a brush tool in a graphics application such as Microsoft Paint or Adobe Photoshop, being able to toggle between two ‘brushes’ for the two classes. As the user paints, a model is trained, and the output of the model is rendered onto the same image, through a translucent overlay. This allows the user to focus further annotation on misclassified areas. A similar interaction has recently also been applied in a 3D context (Valentin et al., 2015).

Another example of an IML system is *EluciDebug* (Kulesza et al., 2015). *EluciDebug* allows end-users to build multi-class classifiers for organising short to medium-length pieces of text, such as email. The user performs manual annotation by moving emails to folders, where each folder represents a class. As the user organises their email, a model is trained, and the output of the model is presented as suggestions for classification within the email client itself, which the user may accept or overrule. The key commonality is that both systems involve a training loop, where the user provides annotations either in the form of training examples or potentially by manually adjusting model parameters (as can be done in *EluciDebug*), a model is trained, and the model output is somehow presented back to the user for further action in such a way as to directly suggest which further action would be useful.

The core design problems in IML are those of facilitating critical end-user evaluation of the model so that model training is effective. There is considerable overlap with the design problems of visual analytics, namely, perceptual, semantic, and interactional mappings. However, a layer of complexity is added because an IML interface is heavily dependent on the output of the model instance being trained, and as with any inferential interface, great care needs to be taken to account for user expectations, and conflicts between user expectations and model output, as exemplified by the design challenges of the *Dasher* text entry system (Ward et al., 2000). This is a problem which does not occur in visual analytics systems because the output of any model being built is only one of many factors for the analyst to consider in choosing a next interaction step. In IML systems, the model’s state and output is not only a first-class decision support feedback mechanism to help the end-user choose what to do next, it is the *only* decision support mechanism.

IML as end-user programming

IML has a close relationship with the end-user programming research community (Ko et al., 2011). End-user programming, strictly speaking, refers to any programming activity where the program is written for the programmer’s own use. Scientists and data analysts writing statistical and data processing scripts fall into this category. However, something as simple as configuring an alarm clock can also be considered an instance of end-user programming. Users of spreadsheet formulae are end-user programmers; it is common for some users to become so proficient at programming spreadsheets that they are able to build large business applications in them. Such users are sometimes humorously referred to as ‘spreadsheet wizards’. It is because end-user programming so commonly occurs when users are not formally trained in computing, that a core research focus in the EUP community has been to design interfaces which can support programming without advanced or formal knowledge of computing.

As will be discussed in greater detail in Chapter 3, there are clear benefits to viewing IML as an instance of EUP. In particular, just as the activity of end-user programming revolves around correcting errors in an existing implementation (i.e., *debugging*), so too does the activity of interactive machine learning revolve around correcting errors in a model’s predictions. Consequently, powerful end-user debugging techniques such as *What You See Is What You Test* (Rothermel et al., 1998) and the *Whyline* (Ko and Myers, 2004) can be appropriated for use in the IML context. The Whyline is an “interrogative debugging” interface for CMU’s Alice (Dann et al., 2011). It takes the form of a “why/why didn’t” hierarchical dropdown menu, which then lists all objects, and then all their behaviours. This allows the user to ask questions such as *Why didn’t Pac-Man resize 0.5?* (i.e., why didn’t Pac-Man shrink?) The Whyline then looks at the execution trace and provides an answer in the form of a visualisation of the runtime actions that prevent Pac-Man from resizing. This execution “history” can be scrubbed to provide rapid feedback. In a study of experienced programmers debugging Alice code, those with the Whyline available to them were significantly faster at fixing bugs than those without. Kulesza et al. explicitly reinterpret Whyline-style interaction in the context of EluciDebug. Recall that email classification is the motivating example. Whyline-style answers are presented as visualisations, including interactive bar charts of the weights assigned to certain keywords by the algorithm (naïve Bayes), which the user can easily edit by dragging the bars.

2.3 Differences in approaches to interactive analytics

2.3.1 Technology

IDAs, VA, and IML are all shaped by slightly different technological concerns and limitations. IDAs are concerned with the automatic generation and evaluation of KDD workflows. Consequently, technical issues of construction and representation of appropriate workflow ontologies (e.g., Diamantini et al. (2009); Kietz et al. (2009); Panov et al. (2008)) are central. A variety of approaches can be used to generate workflows, such as program synthesis (Buntine et al., 1999) and hierarchical planning (St Amant and Cohen, 1997). A common strategy for evaluating workflows is to execute the workflow while gathering diagnostic information, such as execution times and prediction accuracy. In order to scale this approach, another technical focus of IDAs is in designing appropriate distributed system architectures (Wirth et al., 1997).

Visual Analytics systems are concerned with algorithms and system resources required to achieve specific perceptual properties. For instance, a lot of attention is paid to layout algorithms such as in Sugiyama et al. (1981), as well as specific aspects of visual perception, such as in Szafr (2015) and Mittelstädt (2015), which both tackle issues of colour and colour compensation. These must be computed with low latency in order not to inhibit data exploration (Liu and Heer, 2014).

Interactive machine learning systems are shaped by the concerns of the machine learning algorithms being employed. In order to provide understandable machine learning models at interactive speeds, IML systems often use speed-optimised versions of standard machine learning algorithms, or simplified versions with fewer parameters so that they are easier to illustrate with completeness.

2.3.2 End-user task formalisation

IDAs, VA, and IML all differ in the extent to which they formalise the data analytics being performed. IDAs are explicit in formulating formal ontologies. The space of data transformation operations is neatly circumscribed, and each data transformation operation has a concrete mathematical formulation. VA systems do not treat ontologies as explicit software constructs. However, the design and evaluation process for VA systems typically requires a careful formulation of the task in terms of an analytical framework, such as given by Amar et al. (2005) and Brehmer and Munzner (2013). For example, the taxonomy presented by Amar et al. (2005) identifies 10 categories of low-level analytic activities, such as ‘retrieve value’, ‘sort’, ‘cluster’, and ‘correlate’. Designers of VA systems can use such categories to reason about specific aspects of their current design. IML systems are least explicit; however, this is to be expected. The design practice of IML is not sufficiently mature for the IML workflow to be formally characterised in terms of its constituent parts. Instead, IML has intermediate-level taxonomies, such as *intelligibility types* (Lim et al., 2009) and *principles of explanatory debugging* (Kulesza et al., 2015), explained in greater detail in the next chapter.

2.3.3 Approach to knowledge generation

IDAs, VA and IML all differ in their approach to knowledge generation. A core assumption of IDAs is that it is appropriate to characterise exploratory analytics using a formal ontology. This avoids several potential issues by making the assumptions of an analytical process specific. For example, the assumptions of a Pearson correlation: continuous measurements, related pairs, absence of outliers, normality of variables, linearity, and homoscedasticity, can be encoded explicitly as preconditions for application of the operation.

However, analytics is a form of expert behaviour which involves information foraging and sensemaking. This was demonstrated by Pirolli and Card (2005) through their influential cognitive task analysis of intelligence analysts. Analysts’ expert behaviour is driven by *exploratory* processes, which generate theory from data, as well as *hypothesis testing* processes, which drive analyses from theory. Pirolli and Card identify the concept of schemas: patterns around the important elements of tasks / workflows, as built up from experience by experts. ‘Experts don’t just automatically extract patterns and retrieve their response directly from memory. Instead, they select the relevant information and encode it in special representations [...] that allow planning, evaluation and reasoning about alternative courses of actions.’ (Ericsson and Lehmann, 1996).

Importantly, Pirolli and Card note that ‘Information processing [...] can be driven by bottom-up processes (from data to theory) [i.e., *exploratory*] or top-down (from theory to data) [i.e., *hypothesis testing*]. Our analysis suggested that top-down and bottom-up processes are invoked in an opportunistic mix.’ The scope for this to occur in IDAs is limited, as the requirement of formal ontologies precludes intermediate steps which involve approximate human reasoning, discussed in greater detail in Chapter 3. This limitation is the principal reason for not pursuing the IDA approach to interactive analytics in this dissertation.

The core epistemological assumption of visual analytics is that knowledge of data can be constructed through the interpretation of a graphical system representing that data. Consequently, accurate representation of the data is important. Perceptual pattern matching is an approximate process, and judgements of the data which do not have a formal statistical definition (e.g., ‘it looks like sales are going up’, ‘this street has high footfall’, etc.) are also valued as types of knowledge.

When using IML systems, users form mental representations (Norman, 1983) of the statistical model they’re building. Consequently, the research focus is on how to help the user soundly and completely construct these representations (Kulesza et al., 2013). Here again the nascency of IML means that a clear epistemological foundation has yet to emerge. Chapter 3 contributes two novel perspectives towards this: namely, that the usage of IML systems can be viewed as dialogue, and this dialogue facilitates constructivist learning as the user’s mental model is iteratively perturbed and refined through interaction with the IML system.

2.3.4 Problem domain

IDAs, VA and IML are each shaped by the typical problems they are designed to tackle. IDAs are built for professional data analysts and statisticians who perform a wide variety of analyses on different data domains. These analysts face issues such as slow recomputation of results, tedious data wrangling, and suboptimal solutions due to lack of expertise in particular techniques. With the exception of a few visual analytics tools such as Tableau, the vast majority of VA research is aimed towards supporting analysts in highly specific domains. These are either professional analysts deeply embedded in a particular scientific or industrial context, or ‘end-user’ analysts who are not formally statistically trained but are highly expert in a specific data domain. These users have specific analyses which need to be supported by the visual analytics tool, and the analysis is typically an important part of a decision support system which drives business actions. IML research is also aimed towards highly specific domains, but more typically is aimed towards end-users who are not performing an analytics task, but rather tasks which would benefit significantly from partial or complete automation made possible through the strategic deployment of supervised learning. The *CueT* system for network alarm triage (Amershi et al., 2011b), as well as Crayons and EluciDebug are all excellent examples of this.

2.4 Interactive analytical modelling

2.4.1 The two cultures of statistical modelling

In a highly influential article at the turn of the millennium, Leo Breiman (Breiman et al., 2001) characterises two distinct approaches to statistical modelling. The first is the ‘data modelling’ culture. In this culture, a generative model such as linear or logistic regression model is assumed as a candidate to be the ‘true’ underlying natural model. The parameters of this model are estimated from the data. The model is validated using goodness-of-fit tests and examination of the residuals, and the resultant model is used both to make analytical inferences about the data as well as for prediction. Breiman estimated that at the time, 98% of all statisticians fell into this camp. The second is the ‘algorithmic modelling’ culture. In this culture, no intrinsic model is assumed. However, a model such as neural networks or decision trees is chosen and fitted to the data. The focus is on prediction accuracy. The model is evaluated using prediction error.

Breiman’s argument is that there is a disproportionate and deleterious emphasis on data modelling, and not enough on algorithmic modelling. The techniques of algorithmic modelling, such as neural networks, decision trees, and support vector machines, provide higher predictive accuracy in practice. Breiman insists that posing the difference between the two cultures as a tradeoff between accuracy and interpretability is an incorrect view, as a model with higher prediction accuracy is more *informative* by definition, and information is a more important goal for statistical modelling than interpretation. This makes sense, as interpreting a model with low accuracy because it is easy to interpret is akin to dropping your keys at night on the street and looking for them under the next street lamp because there is light there.

I do not disagree with the idea that viewing the relationship between accuracy and interpretability as a zero-sum tradeoff is incorrect. However, the core practical difference between these two cultures, as outlined by Breiman, is in their choice of model. This is not a true dichotomy, as ‘algorithmic modelling’ techniques can be shown to have ‘data modelling’ interpretations. For instance, some neural networks can be shown to have equivalent interpretations as Gaussian processes (MacKay, 1997; Neal, 1996) and Bayesian learning (MacKay, 1992).

Breiman begins by acknowledging two separate goals in statistical modelling: *Prediction*, to predict responses for previously unseen response variables, and *Information*, to understand something about the natural process which generated the observed data. The two cultures are presented as two separate approaches to these goals. While the two cultures continue colliding, with methods and motives intermingling, the tools and techniques tailored to each goal still remain very different, and the tradeoff between these two goals often reflects and draws upon the accuracy/interpretability tradeoff of different models. This dissertation uses the term *model building* to talk about the prediction goal, and *analytics* to talk about the information goal. Consider IML and visual analytics. As discussed in previous sections, IML is heavily focused on the model building goal; the primary information artefact being created in an IML system is a reusable model with high predictive accuracy. In contrast, visual analytics is heavily focused on the analytics goal; no primary information artefact is created in a visual analytics system. Rather, the aim is to create some insight or understanding of the data.

2.4.2 Novel research focus

Bridging analytics and modelling is an important aim because these are rarely separate activities in the practice of data science. High-accuracy models used in practice (e.g, for credit card fraud detection, recommender systems, algorithmic trading, etc.) need to have analytic value, not only to help ensure the absence of bugs, but also to clarify the beliefs and assumptions implicit in the model, to improve the accountability of the system. For instance, a credit card company needs to be able to explain in a court of law what activity their system considers to be fraudulent and why. In practice, these models have large numbers of parameters and are extremely difficult to interpret or debug. Analytics is benefited by statistical modelling, as models can efficiently isolate relevant variables and confirm/refute hypotheses. However, analytics is often required, and consequently conducted, by end-users with little formal training in statistics or machine learning.

This dissertation pursues a stronger relationship between analytics and modelling. It is not an unprecedented idea to bridge these goals – this has been the subject of much research and meta-discourse. For instance, Caruana et al. (2015) have been investigating a line of statistical research that seeks to develop models which balance intelligibility and high prediction accuracy. Another example is the recent surge of interest in deep neural networks, which has also spurred a lot of discourse regarding their large quantities of parameters, and how to visualise and interpret them (Zeiler and Fergus, 2014). These solutions focus heavily on the structure of the underlying model, betraying the assumption that the interpretability of a model is dependent solely on its structure.

Interactive analytical modelling

This dissertation takes a different approach to bridging the activities of analytics and modelling by posing it as an interaction design problem. Well-designed interaction in an interface creates a dynamic story between user and system that unfolds over time. To describe the hypothetical middle ground between analytics and modelling, the straightforward term *analytical modelling* shall be used. This dissertation investigates approaches to analytical modelling using interactive computer programs, that is, *interactive* analytical modelling. For brevity, the word ‘interactive’ is henceforth assumed to be implicit, and ‘analytical modelling’ is used instead.

At this point it is worth reiterating our two research questions:

1. Can tools be built for analytical modelling, that is, analytics through model-building?
2. Can they be made useful for non-experts?

These are addressed as follows. To begin with, three novel theoretical perspectives which have emerged over the course of this research are presented. Next, two concrete systems facilitating analysis and model-building are presented and evaluated. The first is a visual analytics tool that incorporates model-building as a core part of the analytical process. The second is a model-building tool that facilitates analytical outcomes. Both tools are designed to be operated without types of expertise which are usually crucial to analysis and model building.

How the questions are answered in the following chapters is now summarised.

Theoretical perspectives on interactive analytical modelling

Three perspectives have helped build effective interactions that bridge the space between visual analytics and interactive machine learning. The first idea is that visual analytics can be viewed as end-user programming. The second idea is that interactive machine learning can be viewed as dialogue. The third idea is that interactive analytical modelling can be viewed as constructivist learning. These, and consequent design principles, are detailed in Chapter 3.

An analytics tool that facilitates model-building

Gatherminer is a visual analytics tool designed for detecting and explaining patterns in time series databases. Users interactively annotate a time series visualisation to indicate aspects of the visualisation which are interesting to them. A model is trained on these annotations (similar to how Crayons (Fails and Olsen Jr, 2003) builds a classifier from annotations on an image) and this model in turn is visualised to present a statistically-grounded explanation for the interesting observations. This is detailed in Chapter 4.

A model-building tool that facilitates analytics

BrainCel is a model-building tool that allows non-expert end users to build and apply machine learning models within spreadsheets. Users interactively annotate data in the spreadsheet to indicate training examples. A model is trained on these examples and visualised, illustrating several types of analytic insight, such as the structure of the dataset in high-dimensional space, value distributions, and outliers. The model can then be used to validate existing data or to guess values for missing data. This is detailed in Chapter 5.

THEORETICAL PERSPECTIVES

This chapter discusses how this dissertation relates to the theory of interaction in the context of analytical modelling. Applying the method of research-through-design, three views have crystallised. Firstly, this dissertation presents the view of visual analytics as end-user programming. Secondly, this dissertation presents the view of interactive machine learning as dialogue, outlining the opportunities and shortcomings of this interpretation. Thirdly, this dissertation links the exploratory knowledge-generation process of visual analytics to learning in a constructivist setting, which provides a novel, first-principles explanation for the supposed benefits of interaction in analytics and machine learning. In closing, four design principles derived from these views are presented.

This chapter presents research described in the following papers:

- **Constructivist Design for Interactive Machine Learning.** Advait Sarkar. *Proceedings of the 34th Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA 2016)* (pp. 1467–1475).
- **Confidence, command, complexity: metamodels for structured interaction with machine intelligence.** Advait Sarkar. *Proceedings of the 26th Annual Conference of the Psychology of Programming Interest Group (PPIG 2015)* (pp. 23–36).
- **Visual Analytics as End-User Programming.** Advait Sarkar, Alan F. Blackwell, Mateja Jamnik, Martin Spott. *Psychology of Programming Interest Group Work-In-Progress Workshop 2015 (PPIG-WIP)*.
- **Hunches and Sketches: rapid interactive exploration of large datasets through approximate visualisations.** Advait Sarkar, Alan F. Blackwell, Mateja Jamnik, Martin Spott. *Proceedings of the 8th International Conference on the Theory and Application of Diagrams, Graduate Symposium, 2014 (DIAGRAMS 2014)*.

3.1 Research methodology

This dissertation has outlined the need for simple end-user tools that facilitate “analytical modelling”, the hybrid activity of analytics through model-building. In subsequent chapters it will be shown through two examples, Gatherminer and BrainCel, how model-building can be incorporated into visual analytics systems, and vice versa. Moreover, Gatherminer and BrainCel will be shown to reduce the representational, process, and domain expertise typically required for analytical modelling.

Design shifts the world from its current state into a “preferred” state through the production of a designed artefact. This dissertation describes the design of Gatherminer and BrainCel with a focus on documenting and theorising those aspects of the design that (a) facilitate analytical modelling and (b) reduce expertise requirements for end-users. Thus, the approach to knowledge production here is *research through design* (Frayling, 1993). The distinction between research through design, and merely design, is one of intent. In the former, design is practiced with the primary intent of producing knowledge for the community of academics and practitioners. Consequently, the design artefact cannot stand in isolation – it must be accompanied by some form of discourse intended to communicate the embodied knowledge result to the community. Moreover, this discourse must make explicit how the artefact is sufficiently novel to contribute to knowledge. In a non-research design activity, neither annotative discourse nor novelty is necessary for success.

Zimmerman et al. (2007) propose four general criteria for evaluating research through design contributions: process, invention, relevance, and extensibility. *Process* refers to the rigour and rationale of the methods applied to produce the design artefact, and *Invention* refers to the degree of academic novelty. These two criteria are addressed *in situ* in the following two chapters, which detail the implementation and design decisions in the two analytical modelling systems. *Relevance* refers to the ability of the contribution to have a wider impact, which has been addressed by the motivation in previous chapters. This chapter is concerned with *extensibility*, the ability of the knowledge as documented to be built upon by future research.

An attractive proposition is to seek a theory of design for analytical modelling systems that not only characterises specifically the nature of these systems and how their important properties may be measured, but also prescribes a straightforward, deterministic strategy for the design of such systems. I initially anticipated that such a prescriptive theory would be elusive for multiple reasons, including the nascency of interactive machine learning, the incomplete characterisation of potential applications, and a wariness of the challenges surrounding “implications for design” (Stolterman, 2008).

My current position is that a *complete* design theory is not only elusive, but impossible – not just for analytical modelling, but *any* design domain. This is because theory underspecifies design, and design underspecifies theory (Gaver, 2012). Theory underspecifies design because a successful design activity must culminate as an *ultimate particular* (Stolterman, 2008): an instantiated, designed artefact, subject to innumerable decisions, situated in a particular context, and limited by time and resource constraints. Design problems are inherently *wicked* problems (Buchanan, 1992); they can never be formulated to a level of precision which affords ‘solving’ through formal methods, and no theory for design can profess to provide a recommendation for every design decision. Conversely, design underspecifies theory, in the sense that an ultimate particular will fail to exemplify some, or even many, of the nuances captured in an articulated theory.

This is not to say that we should do away with theory altogether and focus solely on artefacts themselves. Gaver’s view, to which I am sympathetic, is that design theory is “provisional, contingent, and aspirational”. The aim of design theory is to capture and communicate knowledge generated during the design process, in the belief that it may sometimes, but not always, lead to successful designs in the future.

During the course of this work, three such novel theoretical perspectives have been developed. They do not make many prescriptions for the design *practice* of analytical modelling systems, and the few explicit prescriptions in this chapter are better viewed as example interpretations of the viewpoint posed, to better illustrate their design knowledge content. These perspectives provide a foundational discussion which situates the roots of the new design practice of interactive analytical modelling at the frontiers of information visualisation, end-user programming, and constructivist learning.

In particular, the perspectives presented in this chapter are:

- Visual analytics is end-user programming.
- Interactive machine learning is dialogue.
- Analytical modelling is constructivist learning.

None of these views is without limitations. Each view presents opportunities to draw upon the existing literature to inform design. In particular, viewing visual analytics as end-user programming informs how representational expertise requirements can be reduced, for instance by facilitating explanatory debugging (Kulesza et al., 2015), and developing representational formality (Stead and Blackwell, 2014). Viewing interactive machine learning as dialogue addresses a core shortcoming of the end-user programming interpretation, namely that the analytical sensemaking process is an ill-defined optimisation problem. Additionally, this view formalises the relationship between analytical modelling and mixed-initiative interaction. Finally, viewing analytical modelling as facilitating learning in a constructivist setting provides a novel theoretical basis for *why* interaction in visual analytics and machine learning works: because it provides opportunities for the user’s ideas and experiences to interact.

From these views, four concrete design principles are derived:

- Begin the abstraction gradient at zero.
- Abstract complex processes through heuristic automation.
- Build expertise through iteration on multiple representations.
- Support dialogue through metamodels.

The aim is to show that the systems presented in Chapters 4 and 5 are not merely ad-hoc constructions, but instead constitute an annotated portfolio (Gaver and Bowers, 2012) of design-led research, which has led to the production of what Löwgren (2013) term “intermediate-level knowledge” – although I am not in agreement with their position that intermediate-level knowledge is not theory. This chapter positions the systems presented not as isolated design artefacts, but the first instances of a new and unique form of interaction design with empirically grounded and theoretically defensible principles.

3.2 Visual analytics is end-user programming

Multiple stages of the analytics pipeline can involve programming. For instance,

1. **Capture:** Raw data is captured by programmatic instrumentation of some real-world process. In web traffic analytics, code for logging usage must be injected into the website. In retail analytics, point-of-sales terminals or warehouse management systems are likewise instrumented.
2. **Management:** Data is managed through programmatic manipulation of databases, or through “extract, transform and load” tools, typically thin graphical layers over proprietary domain-specific languages for data transformation.
3. **Analysis:** Data is analysed programmatically. A simple approach is to directly query the database. A more complex approach is to extract projections (‘views’) of the database, which can then be used with statistical packages such as SPSS; for complex data processing operations in a general-purpose programming language that are difficult, slow, or impossible to define in terms of database queries; or for ad-hoc statistics and machine learning with languages such as R and Python.
4. **Presentation:** Analytical results are presented using charting libraries such as D3.js, or through desktop productivity applications which offer charting capabilities.

Stage #3 (analysis) is the focus of this dissertation. Data analytics tools such as Excel and Tableau are designed, amongst other things, to aid this step of the process by standing in as replacements for database queries and statistical languages, doing so in a way that reduces the representational expertise requirements of those languages. The utility of visualisation for data analysis cannot be overstated. Human perception paired with modern visualisation software enables rapid detection of trends, outliers, and comparisons of quantities – even, and perhaps especially so, by those without statistical expertise.

The space of analytical questions one can ask of a particular dataset is infinite, but only some questions yield interesting answers. Consequently, exploratory data analysis is divided between two activities. The “bottom-up”, hypothesis-generation activity identifies interesting questions, for example: *should I investigate the relationship between variables X and Y?* The “top-down”, hypothesis-testing activity applies a formal method to answer a well-defined statistical question, for example: *is there a significant difference between these groups?*

Cognitive task analysis shows that experienced analysts often invoke top-down and bottom-up processes in an opportunistic mix (Pirolli and Card, 2005). Visualisations can help rapidly prune the space of interesting hypotheses, and they can help verify many of these hypotheses (Keim, 2001). This spares the analyst the effort of conducting a more elaborate statistical investigation of a question that in hindsight turns out to be uninteresting, or the wrong question to ask. Thus, visualisations often do not support exact inference, but instead facilitate rapid informal reasoning and the formation of “hunches” – approximate hypotheses and heuristics for exploring the hypothesis space. Hunch-driven reasoning yields informal answers to open-ended questions an analyst might have (e.g., *Does this look like signal or noise? Does there appear to be cluster structure in the data? What is the general shape of the distribution? Is there a point of inflexion in the time series?*) before formulating specific statistical questions.

Visual analytics is a form of end-user programming precisely because visualisations can be used powerfully and generally to informally express and test the same kinds of hypotheses which can be expressed and tested formally through statistical programming languages.

Informal visual reasoning appears to be a form of pattern recognition; the analyst possesses a repertoire of “interesting” visual patterns and the hunches they correspond to. This repertoire is built through experience and domain expertise. One might conjecture that these are mentally represented as archetypes against which the visualisation is compared. One might further conjecture that these archetypes can be hierarchically organised, such that more complex hypotheses are compositions of simpler hypotheses.

As an example, consider the scatterplot. The scatterplot places marks representing data points described by two interval variables, positioning each data point’s mark on a Cartesian manifold defined by two orthogonal axes, with the first coordinate proportional to the value of the first variable, and the second coordinate proportional to the value of the second variable.¹

The very act of viewing a scatterplot can generate, test, and eliminate several families of hypotheses. This process is much faster and more exhaustive, albeit less rigorous, than writing, running and interpreting the output of each corresponding program. It can lead to superior analysis even in the absence of any formal statistical follow-up, as seen in Anscombe’s quartet (Figure 3.1). The four sets of points depicted in this figure have some identical statistical properties: in each case, the x coordinates have identical mean and variance; so do the y coordinates. In each set the correlation between x and y values is the same (0.816), and the slope and intercept of the linear regression line in each case are identical. However, from the scatterplots it is obvious that these four sets are very different. An informal visual reasoner could be more informed about this dataset than a formal reasoner who ‘views’ the data through only a handful of statistical measures.

Some archetypes for scatterplots, along with the hypotheses they correspond to, and their interpretations as programs, are given in Table 3.1. In each case, the formal statistical interpretation provided is merely one of a number of valid interpretations, and the code provided is merely one instantiation of that interpretation.

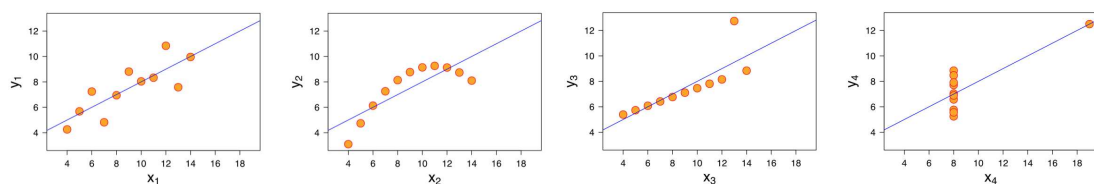


Figure 3.1: Anscombe’s quartet (Anscombe, 1973).

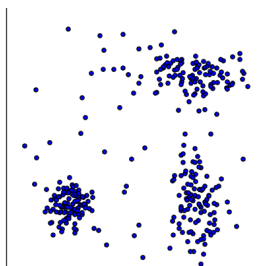
¹This is a basic definition corresponding to the commonest case. Scatterplots can also be made with categorical and ordinal values, but mapping these onto spatial axes is a matter of interpretation. Scatterplots can also be made with multiple axes, non-orthogonal axes, etc. The basic scatterplot considered here is in one sense the most perfect graphical representation, in that it takes our strongest perceptual channel for quantitative comparison: position, and encodes two values by using orthogonal axes, which is the only way of encoding two values independently using position on a Cartesian manifold. In fact, using orthogonal axes lying on the plane of human vision is the only way of perceptually encoding two values independently using position.

Scatterplot archetype

Informal hypothesis

Formal statistical interpretation

There is cluster structure

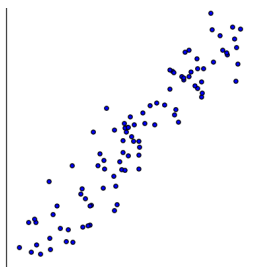


Clustering using k -means, hierarchical clustering, or other method, results in high-quality clusters as measured by indices such as those given by Davies and Bouldin (1979) or Dunn (1974).

Example program in R:

```
# K-Means clustering with 5 clusters
fit <- kmeans(myData, 5)
# evaluating through multiple indices
library(fpc)
cluster.stats(d, fit$cluster)
```

A linear relationship exists between the two variables

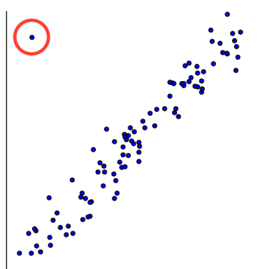


The two variables have a high Pearson correlation. Or, the linear regression of one variable against the other produces a model with low error as measured by residual sum of squares or other metric.

Example program in R:

```
# Correlations with significance
library(Hmisc)
rcorr(myData, type="pearson")
```

There are outliers



Some data points have an unusually low local density (are unusually unlikely) when compared to their neighbours (Breunig et al., 2000).

Example program in R:

```
library(DMwR)
# Use density of 5 neighbours
outlierScores <- lofactor(myData,
k=5)
# pick and show top 5 outliers
outliers <- order(outlierScores,
decreasing=T)[1:5]
print(outliers)
```

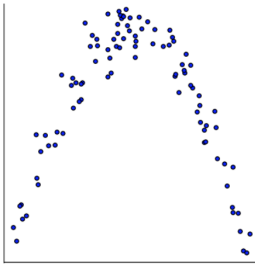
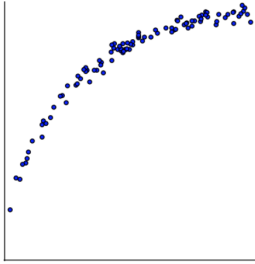
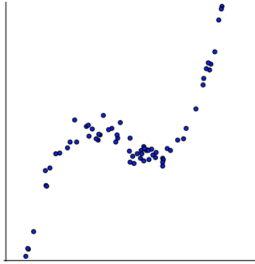
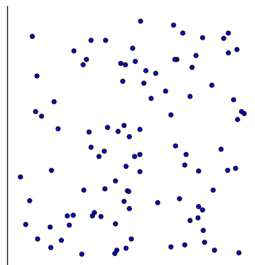
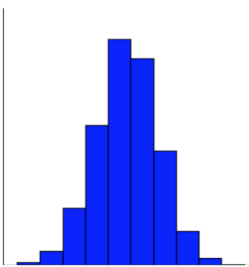
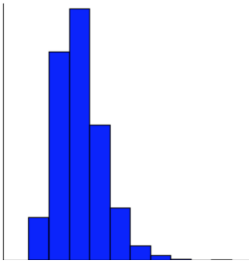
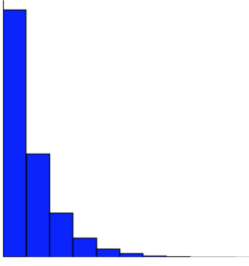
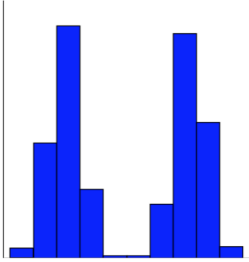
	<p><i>A nonlinear relationship exists between the two variables</i></p>	<p>The regression of one variable against a nonlinear function of the other (belonging to a family such as the polynomial, logarithmic, exponential, logistic, etc.) produces a model with low error as measured by residual sum of squares or other error metric.</p>
		
		
	<p><i>The two variables are unrelated</i></p>	<p>The mutual information (Cover and Thomas, 2012) of the two variables is low. Or, a highly nonlinear function is required to produce a model with low training error.</p>
		<p>Example program in R:</p> <pre>require(entropy) mutualInfo=mi.empirical(myData)</pre>

Table 3.1: Some scatterplot archetypes, corresponding informal hypotheses, and formal statistical interpretations.

More examples of how graphical representations can be viewed as archetypal patterns with corresponding statistical interpretations are in Table 3.2, in this case using histograms and line charts. Code snippets are excluded for brevity.

Tables 3.1 and 3.2 are not presented as rigorous taxonomies, although the creation thereof would be very interesting future work. Some groundwork has already been laid through perceptual similarity studies conducted by Pandey et al. (2016). The taxonomy could be immediately applied as a perceptually-grounded set of pictograms to assist visual analysis (Lehmann et al., 2015).

Similarly, the ‘archetype’ images are not precise or empirically grounded. They are simply meant to illustrate the many ways in which the informal visual analytics reasoning process can serve the same purpose as writing statistical programs. It is not possible, for example, to have a canonical scatterplot of cluster structure, precisely because that is an approximate, fuzzy, visual concept.

Visualisation archetype	Informal hypothesis	Formal statistical interpretation
	<p><i>The data has a normal, log-normal, power law, bimodal, etc. distribution.</i></p>	<p>Fitting a normal, log-normal, power law, bimodal, etc. distribution to the data (e.g., by the maximum likelihood method) results in a good fit as evaluated by, for example, the Kolmogorov-Smirnov test (Massey Jr, 1951).</p>
		
		
		

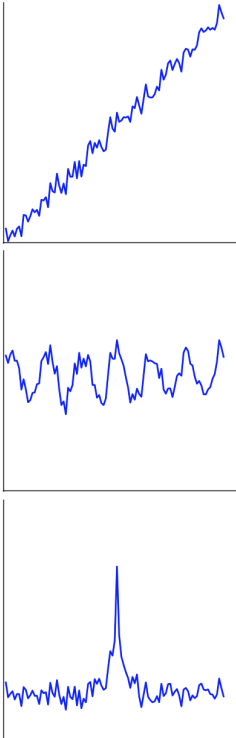
	<p><i>There is a trend, a periodic function, a spike, etc.</i></p>	<p>These features are detected using a robust framework such as compositional kernel search (Duvenaud et al., 2013).</p>
--	--	--

Table 3.2: More example archetypes, corresponding informal hypotheses, and formal statistical interpretations.

This section has presented the argument that visual analytics is like end-user programming because reasoning supported by visualisations can achieve the same outcomes as reasoning supported by statistical programming. One may object, fairly, that this constitutes a fallacy: simply because informal visualisation-driven reasoning is a means to an end (analytic insight), and statistical programming is another means to that same end, it does not follow that visual analytics is indeed statistical programming.

A more conventional argument would be to show how typical programming constructs, such as conditionals, loops, and abstraction, manifest in visual analytics systems. It is difficult to make this type of argument for all visual analytics systems as they vary greatly in such explicit programming affordances, and at least in this sense, the usage of many visual analytics systems cannot be considered ‘programming.’ However, at least two programming constructs appear in nearly every general-purpose visual analytics system. The first is *filtering*, or the conditional inclusion or exclusion of data in the visualisation. This expresses the same sort of computation as, for example, `HAVING` or `WHERE` clauses in SQL queries. The second is the *abstraction of the reusable visualisation*. This means that the system represents any given visualisation, for example, a set of charts, their value mappings, data filters, and aesthetic settings, as an abstract object *separate* from the underlying data. Consequently, in making a chart, the end-user in fact creates a reusable algorithm for graphical presentation. In Excel, this manifests as the ability to simply paste in new data into the cells containing the data for the chart. In Tableau, the same visualisation can be ‘connected’ to different underlying data sources. An Excel chart or Tableau visualisation, then, is a ‘program’ for producing a certain type of analytic insight.

The abstraction of the reusable visualisation disappears from specialised visual analytics systems. For instance, the end-user customisability of Gatherminer and BrainCel (presented in subsequent chapters) is sufficiently low that the charting aspects of those interfaces cannot be considered programming. However, as explained in the next section, the explicit end-user programming component of those interfaces comes from their implementations of interactive machine learning. That the visualisations of Gatherminer and BrainCel serve the same informal hypothesis generation and testing purpose as discussed in this section is, additionally, an implicit end-user programming outcome.

Implications of visual analytics as end-user programming

What is gained by viewing visual analytics as end-user programming? The aspiration is a theory that can inform design practice. Many results from end-user programming are just as applicable to visual analytics as they are in interactive machine learning. For example:

- *Multiple representations*: redundant mappings of data variables to visual variables are generally not considered good design practice in visual analytics. However, redundant mappings by way of multiple representation has been found to be useful in end-user programming, as they help the end-user scaffold their understanding of representational abstractions, as exemplified in systems such as DrawBridge (Stead and Blackwell, 2014) and Infer.IDE (Gorinova et al., 2016).
- *Liveness*: instantaneous feedback during program editing, or the ability to modify a program mid-execution, is known as liveness (Tanimoto, 1990). Visual analytics systems typically exhibit some liveness without acknowledging the fact. Studies of progressive visualisation (Fisher et al., 2012; Sarkar et al., 2015; Stolper et al., 2014) and interactive latency (Liu and Heer, 2014) are predicated on liveness being desirable for visual analytics. The value of liveness can be expressed in terms of cognitive dimensions Green and Petre (1996) or patterns of user experience Blackwell (2015a).
- *Explicit programming constructs*: visual analytics systems vary greatly in explicit programming affordances. However, the view of them providing an abstraction paves the way for incorporating more advanced programming constructs. One potential application would be an analytics system with the same visual programming principles as Palimpsest (Blackwell, 2014), with multiple programming constructs available, such as shared state and referencing (e.g., variables), repeated behaviour (e.g., loops), conditional behaviour, and abstraction (e.g., encapsulation, subroutines).
- *Debugging*: end-user programming has a number of mature debugging strategies, such as WYSIWYT (Rothermel et al., 1998) and the Whyline (Ko and Myers, 2004). Visual analytics does consider usability issues, but does not acknowledge two fundamental usability barriers are classes of bugs: syntactic bugs, where the user is unable to construct the visualisation they desire, and semantic bugs, where the user has constructed a visualisation that results in an incorrect analysis.
- *Evaluation*: the idea that function dictates form is implicit in visual analytics design practice. Specifically, if the intended function (analytics problem) is sufficiently well-characterised so as to be optimisable, studies can investigate which forms are appropriate for those functions, and one naturally arrives at a well-designed solution to the problem. When approached from this perspective, it would appear that a well-designed solution to a visual analytics problem implicitly entails a ‘well-designed’ user experience. End-user programming considers a wider variety of outcomes, including the learnability of the interface, the soundness and completeness of mental models, etc.

- *New foci*: visual analytics typically draws motivating scenarios from scientific research and industry. End-user programming concerns a broad class of users, and could supply new foci for visual analytics, such as education and creative practices.

3.3 Interactive machine learning is dialogue

The previous section presents a novel view of visual analytics as end-user programming that has clear benefits for the design of such systems. These benefits are well-known to the interactive machine learning (IML) research community, which at the time of writing is closely connected to and aware of end-user programming research. Consequently, it would certainly not be a novel observation that IML is like end-user programming. That interpretation has served the design of IML systems well.

In pushing the boundaries of IML systems towards visual analytics, this research has uncovered a limitation of interpreting IML as an end-user programming activity, namely, that IML represents a type of programming which is a form of dialogue with machines. It is unclear how this dialogue should be structured, as the notion of ‘correctness’ for these programs is unknown or ill-defined. The nature of this dialogue will now be described, and a potential design approach presented: metamodels of machine-learned models.

Why is IML different from traditional programming? In a simplified caricature of the traditional programming paradigm, the programmer has a mental model of the ‘goal’ information structure (program) to be built. Through a direct channel, such as inspection of the source code, its output, and execution traces, the programmer can build a mental model of the information structure as it currently is. Thus, the programmer can compare these two models against each other and decide whether the program matches the goal, or whether it is incomplete, or contains errors. This is a simplification; programmers seldom consider entire programs at once, but rather focus on smaller units at a time.

The utility of the direct channel characterises the traditional programming paradigm. The expected output is sufficiently well-defined, that should it depart from the programmer’s expectations (i.e., an error), inspecting the program and its output suffices to resolve the situation (i.e., debugging). There has been much study, detailed shortly, on enriching the debugging experience with information through an ‘indirect’ channel, for example, through descriptions of the program, its time and memory requirements, and through visualisations of its operation. Nonetheless, it is still possible, and mostly sufficient, to conduct debugging through direct inspection of the program source code, output, and traces.

End-user machine learning is an emergent form of programming

The needs of end-user machine learning are broader than the applications of current IML systems. Users increasingly interact with an inferred world (Blackwell, 2015b), and program behaviour is becoming predominantly probabilistic and data-dependent, rather than deterministic. Training statistical models is an act of programming. Users of systems such as recommendation systems (e.g., Amazon’s product recommendations, Pandora’s music recommendations), intelligent personal assistants (e.g., Apple’s Siri, Microsoft’s Cortana, Google Now), and intelligent consumer tools (e.g., Excel’s Flash Fill) etc., increasingly find themselves programming their environment, implicitly or explicitly.

The decision making processes of these systems, which often involve considerable uncertainty, are largely opaque to end-users. When the output departs from their expectations, neither are their expectations well-defined (can, for instance, product or music recommendations be ‘right’ or ‘wrong?’), nor would inspecting the source code resolve the situation.

The problem of ill-defined expectations is especially pertinent to analytical modelling. Analytical models seldom reveal conclusive answers; the primary questions of interest, such as “*Is the appropriate model being used?*” and “*Is this prediction good?*”, etc., are more nuanced and much less well-defined than the primary question of interest in traditional programming: “*Is this a bug?*” This is where a dialogue is necessary.

This new programming paradigm is characterised by the following three properties:

1. “Programs” are stored as model parameters (often massively many), unintelligible to humans through direct inspection.
2. The programmer is likely an end-user programmer who is not necessarily skilled at computing.
3. The goal state of the program is unknown or ill-defined.

These make the direct channel less useful. Consequently, greater emphasis must be placed upon the use of the indirect channel. This shifts the emphasis from facilitating the user’s understanding *of* the program, to their understanding *about* the program. Previous end-user debugging research has by no means ignored this channel, and neither has interactive machine learning. A few examples of this are now described.

3.3.1 Other approaches to the indirect channel

The indirect channel in IML

Fails and Olsen Jr (2003) motivate their work by emphasising the ease of generating a classifier in an interactive visual manner. Similarly, the systems in Fogarty et al. (2008), Brown et al. (2012b) and Hao et al. (2007c) are presented primarily for ease-of-use. These systems achieve ease of use by selectively abstracting implementation details, a useful strategy as long as the behaviour of the program corresponds to user expectations. But what happens when the system gets it wrong, and not in a way that is easily apparent (Nguyen et al., 2015; Szegedy et al., 2013)? To better involve the user in the process, the repeated use of the word “explain” throughout the interactive machine learning literature (Herlocker et al., 2000; McSherry, 2005; Pu and Chen, 2006; Tintarev and Masthoff, 2007) does not appear to be coincidental; the underlying aim is clearly to give our interaction with programs a more dialogue-like quality.

Amershi et al. (2011a) identify a few questions for end-user interaction with machine learning: *What examples should a person provide to effectively train the system?*, *How should the system illustrate its current understanding?*, and *How can a person evaluate the quality of the system’s current understanding in order to better guide it towards the desired behavior?*

Lim and Dey (2009) have directly addressed the problem of what types of information about intelligent applications should be given to end-users. They call these “intelligibility types,” and some examples are as follows: *Input & output*: what information does the system use to make its decision, and what types of decision can the system produce? *Why, why not, & how*: why did the system produce the output that it did, why did it not produce a different output, and how did it do so? *What if*: what would the system produce under given inputs? *Model*: how does the system work in general? *Certainty*: how certain is the system of this report? *Control*: how can I modify various aspects of this decision making process? Kulesza et al. (2013) show that these information types are critical for the formation of users’ mental models.

Kulesza et al. (2011) also proposed a set of information types which would benefit end-users who were debugging a machine-learned program, including: *Debugging strategy*: which of many potential ways of improving the model should be picked? *Model capabilities*: what types of reasoning can the model do? *User interface features*: what is the specific function of a certain interface element? *Model's current logic*: why did the model make certain decisions? *User action history*: how did the user's actions cause the model to improve/worsen?

The indirect channel in end-user programming

The producers of these machine learning models are also their users. As such, they are end-user software engineers (Ko et al., 2011), and in particular they engage in end-user debugging. End-user debugging research has been explicit in framing the interaction as dialogue. For instance, Wilson et al. (2003) present a strategy to incentivise users to write more assertions in spreadsheets, a difficult and tedious activity. The strategy – surprise, explain, reward – is much like dialogue. The software generates a potentially surprising assertion that nonetheless fits a cell's formula. It changes the value of the cell to be valid under this assertion, and explains this decision and how to change the assertion through a tooltip. The user is rewarded by virtue of having a more correct spreadsheet.

The *WhyLine* (Ko and Myers, 2004) is a debugging tool which operates literally as dialogue. By scanning the function call structure of a program, the tool can create hierarchical menus which allow the user to formulate pseudo-grammatical “why” questions about the execution of a program. Kulesza et al. (2009, 2011) modify this approach to facilitate end-user debugging of the underlying naïve Bayes model of an email spam classifier.

As with IML, allusions to “explanations” also appear throughout the end-user debugging literature, such as in the “surprise-explain-reward” approach. However, an important distinction exists between the type of dialogue one engages in when debugging, and the type of dialogue one has with a machine learning model. The activity of “debugging” principally occupies the direct channel, as in the traditional programming paradigm. In debugging it is assumed that the user's mental information structure is the correct version, which the computer's internal information structure must aim to reproduce. That is, it is assumed that the human knows the right answer. This is not to say that it is always straightforward for the programmer to concretely express the required information structure in code. Perhaps assistance is received from the system, as in WYSIWYT. Nonetheless, in the traditional paradigm, the final arbiter of what is, and is not a “bug,” is the programmer.

In the new paradigm of end-user machine learning, the goal is unknown or ill-defined. It follows that under these circumstances, “debugging”, or even a “bug”, cannot definitively exist. This is particularly the case for analytical modelling. In Chapter 5, an analytical modelling system called “Teach and Try” is described. In this system, users press a button labelled “Teach” to train a supervised learning model, and a button labelled “Try” to use the model to predict missing data. The system was found to successfully generate some understanding of the limitations of statistical procedures in non-experts. This can be partially attributed to the deliberate selection of the word “Try”, which implies fallibility and evokes empathy, unlike more conventional labels such as “Fill” or “Apply model”.

3.3.2 Supporting dialogue through metamodels

All previous approaches to the indirect channel share one property: each addresses the lack of a *decision support system* to help the user choose their next action. This must necessarily be framed as a dialogue since the system cannot tell the programmer exactly what to do (if so, it should be automated), but merely equip them with the tools to make better decisions.

There are a limited number of important user actions in interactive analytical modelling. Consequently, at each step, there is a limited number of decisions to make:

- What action should be taken next?
- Should the training data be altered? What training data should be added/removed/edited?
- Should the model be deployed? On which data should it be deployed?
- Should any charts/graphs/UI elements be inspected?

A system participates in this dialogue by providing support for these questions.

Since IML models are large quantities of parameters which are unintelligible through direct inspection, a *metamodel* layer is required to present information about the model. Three metamodels are proposed here:

1. **Confidence:** how sure is the model that a given output is correct?
2. **Prediction path:** how did the model arrive at a given output?
3. **Command:** how well does the model know the domain?

These have emerged from consideration of the engineering requirements of analytical modelling systems over the course of the development of the systems in Chapters 4 and 5. They are not exhaustive. Aspects of these metamodels have previously been introduced in the intelligibility types proposed by (Lim and Dey, 2009) and information needs proposed by (Kulesza et al., 2011). This relationship is summarised in Table 3.3. This table excludes some intelligibility types and information needs that are not directly related to the metamodels.

The intelligibility types and information needs frameworks prescribe types of information which would be beneficial to an end-user programmer of machine learning models, but do not prescribe *how* such information might be generated. While metamodels are a conceptual solution at the same level as intelligibility types, that is, they prescribe things which should be shown to the user, they are also an engineering solution at a technical level, that is, they prescribe how this information can be generated. Approaching the problem as one of metamodeling may alleviate the need to recreate methods for providing intelligibility for each new interface and machine learning system on an ad-hoc basis.

Metamodels	Lim and Dey (2009)	Kulesza et al. (2011)
<i>Confidence</i> : how sure is the model that a given output is correct?	<i>Certainty</i> : how certain is the system of this report?	-
<i>Prediction path</i> : how did the model arrive at a given output?	<i>Input</i> : what data does the system use? <i>Why / Why Not</i> : why did/didn't the system make a specific prediction? <i>How</i> : how does the system produce its output?	<i>ML Program's Current Logic</i> : why did/didn't the system make a specific prediction? Why did its prediction not change in response to my action?
<i>Command</i> : how well does the model know the domain?	-	-

Table 3.3: Aspects of the metamodels which have been previously described in terms of intelligibility types (Lim and Dey, 2009) and information needs (Kulesza et al., 2011).

Confidence

Confidence is well-studied in statistics and machine learning. Methods for estimating the error or confidence for a given prediction have been developed for many models. For instance, linear regression is accompanied by a procedure for computing the 95% confidence intervals for its learnt parameters. A similar procedure computes 95% ‘prediction’ intervals per prediction, which can be interpreted as confidence in the prediction. The ability to estimate this confidence is not always incentivised in benchmarks of machine learning performance, which are primarily concerned with the correctness of the output.

Table 3.4 presents suggestions for computing confidence for popular machine learning techniques. A general technique is to generate Bayes confidence intervals from bootstrap samples (Laird and Louis, 1987). Measures of confidence can be used to prioritise human supervision of machine output. When there are large quantities of output to evaluate, the user’s attention can be focused on low-confidence outputs that may be problematic. González-Rubio et al. (2010) use this approach to improve interactive machine translation, and Kulesza et al. (2015) use this approach to improve interactive email classification. Behrisch et al. (2014) apply a confidence metamodel in their software, where the user interactively builds a decision tree by annotating examples as “relevant” or “irrelevant”. The user is able to decide when the exploration has converged using a live visualisation that shows how much of the data passes a certain threshold for classification confidence.

Confidence can also be deceiving. Recent work (Nguyen et al., 2015; Szegedy et al., 2013) has demonstrated how carefully injected noise can trick a state-of-the-art image classifier into labelling apparently straightforward images, and images completely unrecognisable to humans, as arbitrary objects with high confidence. Confidence is necessary but not sufficient for evaluating a model.

Model	Example calculations of confidence
<i>k</i> -NN	For a given prediction, confidence can be measured as the mean distance of the output label from its <i>k</i> nearest neighbours as a fraction of the mean pairwise distance between all pairs of training examples. A similar metric is proposed in Smith et al. (1994).
Neural Network	For a multi-class classification, where each output node emits the probability of the input belonging to a certain class, confidence can be measured simply as the probability reported. More sophisticated confidence interval calculations can be obtained by considering the domain being modelled, as in Chryssolouris et al. (1996); Weintraub et al. (1997); Zhang and Luh (2005).
Decision Tree	The confidence of a decision tree in a given output can be measured as the cumulative information gain from the root to the outputted leaf node. Alternatively, Kalkanis (1993) provides a more traditional approach.
Naïve Bayes	The confidence of a Naïve Bayes classifier in a given prediction can be measured as the probability of the maximally probable class. More sophisticated treatment of the problem is given by Carlin and Gelfand (1990); Laird and Louis (1987).
Hidden Markov Model	The primary tasks associated with HMMs (filtering, prediction, smoothing, and sequence fitting) all involve maximising a probability; the confidence can simply be measured as the probability of the maximally probable output. More fine-grained confidences can be measured by marginalising over the relevant variables (Eddy, 2004).

Table 3.4: Concrete confidence metamodel suggestions

Prediction path

How has the model arrived at a given output label for particular input data? This is a difficult question even for engineers of practical machine learning systems, especially when attempting to generalise from small datasets.

For example, in building a rule-based learning system to predict the likelihood of a patient’s death from pneumonia, Cooper et al. (1997) discovered that the system was exploiting an artefact in the training data to make its predictions. Namely, if the patient had a history of asthma, the model actually predicted a higher likelihood of survival! This directly contradicted established medical knowledge that asthmatics are at a higher risk of death from pneumonia. The oddity was traced to the fact that the model had been trained on treatment records where asthmatics were given much more aggressive treatment in order to compensate for their increased risk. As a consequence, they had a better survival rate than non-asthmatics, a fact that the model had imbibed. It was impossible to guarantee that there were no such other inconsistencies in the rule-based learning system being used, so the model was dropped, and a much simpler class of “intelligible” models were adopted (Caruana et al., 2015; Lou et al., 2012, 2013) despite having lower predictive power. A wise decision, as the model was subsequently found to be exploiting other similar false correlations. Similarly, researchers building a computer vision system for quantifying multiple sclerosis progression based on depth videos (Kontschieder et al., 2014) found that the system was exploiting patients’ facial features in order to “remember” their training labels, so as to cheat the leave-one-out cross validation being used.

The prediction path metamodel attempts to address the question: how much can be explained about a specific prediction without having to expose the internals of the ML model?

One aspect of the prediction path may be captured in the notion of the “complexity” of a prediction. Consider a neural network. It can be argued that when an input highly activates many nodes in the network, the decision making process is more complex than one which involves fewer nodes. This is the case despite the fact that the model structure is identical, with identical edge weights. It can be likened to the difference between mentally computing $199+101$ and $364+487$. One can follow the same arithmetic “algorithm”² and be equally confident in both answers, but one of these instances appears to be more complex than the other.

Command

The “command” metamodel expresses how well the model understands the input domain, that is, whether it has a good command of the input space.

The idea of command has been previously expressed in various forms. The notion of a self-regulated, autonomous agent is long-lived in GOFAI³ research and modern machine learning, motivated by such issues as the “exploration-versus-exploitation” tradeoff; that is, should the agent do something which has been known to provide a certain reward, or should the agent explore the wider world in search of potentially better rewards, at the risk of wasting resources on less-rewarding world states?

A famous problem that benefits from this form of metacognition is the multi-armed bandit (Gittins et al., 2011). A gambler at a row of slot machines has to decide which machines to play, how many times to play each machine, and in which order to play them, in order to maximise the cumulative reward earned. Each machine provides a random reward from a distribution specific to that machine. Thus, the tradeoff is between *exploration*, that is, playing machines in order to learn about their reward distributions, and *exploitation*, that is, playing machines in order to gain the reward. A solution to this problem must necessarily involve a model of command, that is, how much is known about the reward distribution of each machine, in order to effectively navigate this tradeoff.

Similarly, the concept of reinforcement learning (Watkins, 1989) involves a “reward function”, which records the reward an intelligent agent might hope to receive upon transitioning to any given world state; the agent can then probabilistically transition to world states that will either fulfil its information need by updating the reward function, or alternatively will pay off by way of actually receiving the reward. A related concept is *active learning* (Cohn et al., 1996; Settles, 2010), where the algorithm selects examples it believes to be most useful for its learning, and presents these to a human oracle (or other information source) for labelling. The motivation behind active learning is similar to exploration-vs-exploitation: that the algorithm may achieve greater accuracy with fewer training examples should it choose the data from which it learns. Savitha et al. (2012) show a “metacognitive” neural network which can decide for itself whether and when to learn from each training datum it is given. These techniques all rely on a model of the input domain to distinguish between what is known and what remains to be known. In reinforcement learning, this takes the form of the *state space*. Any practical definition of “command” has to be constructed in relation to the domain being modelled.

²This example is purely illustrative. Naturally the mental computation which actually takes place in each case is as yet indeterminable.

³Good Old Fashioned AI, referring to a philosophy of symbolic reasoning and associated techniques. See: Haugeland, John. *Artificial intelligence: The very idea*. MIT press, 1989.

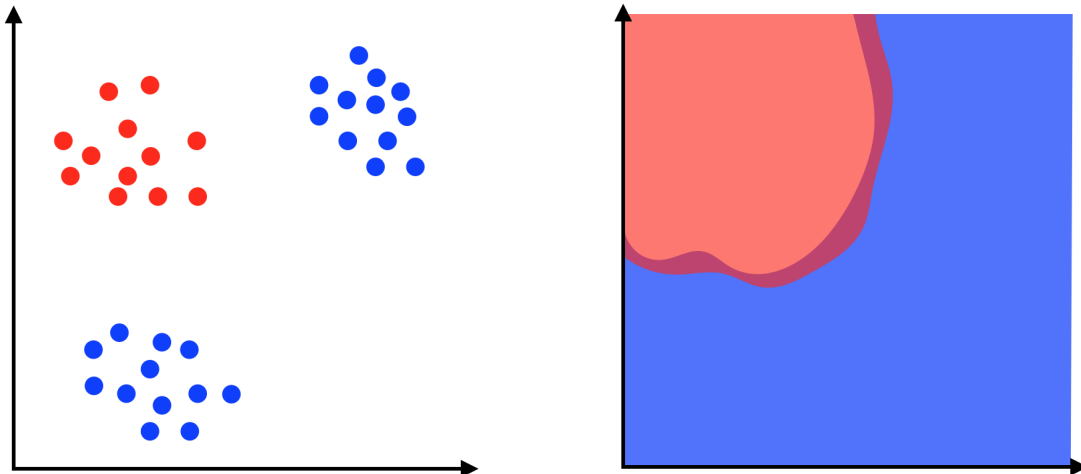


Figure 3.2: Two alternative views of command. The task is to classify examples in this space as ‘red’ or ‘blue’. Left: labelled training examples in the input space. Right: the learned decision boundaries (e.g., through logistic regression), showing an area of reduced certainty. These are radically different interpretations. Consider the bottom right-hand corner. The classifier predicts ‘blue’ with high confidence, but since it has not seen any examples from that area, should it really be confident?

Confidence and prediction path measures are both computed only with respect to a particular prediction. That is, whenever the model is used to make a prediction or classification, there is an associated confidence and prediction path unique to that output. In contrast, the “command” metamodel refers to the current state of the algorithm’s knowledge, independent of any single output.

Two simple methods of illustrating the command of a model over a domain are: to position all training examples so far received in the input domain, and to consider the classifier’s confidence at all points in the domain. These are illustrated in Figure 3.2. The two images present very different pictures of “command” over a domain. If command is viewed as some integral of confidence, then a model with high levels of confidence in the majority of the domain can be considered to have a good command of the domain. If we view command as some integral of the occurrences of training examples encountered, then a model trained on a representative set of training examples may be considered to have a good command.

The command metamodel is related to the problem, in interactive machine learning, of seeking relevant examples for efficient training. When Amershi et al. (2009) discuss how one might seek examples providing greatest information gain for the classifier, they are describing building a partial command metamodel; a full metamodel would allow generative dialogue – software would not only be able to identify examples from the existing corpus but also generate examples which satisfy perfectly the classifier’s information need, provided that the human or other oracle can actually provide ‘good’ labels (Baum and Lang, 1992). Examples that satisfy information requirements perfectly can also be the most challenging for human labellers, and a mixture of ‘easy’ and ‘hard’ examples may elicit higher quality labels (Sarkar et al., 2016). Groce et al. (2014) approach this from the perspective of end-user classifier testing, and show various strategies for selecting evaluation examples. Any such method of eliciting examples includes a command metamodel in order to define and identify deficiencies in the machine’s training.

Metamodels as explanatory metaphors

It may be possible to relax the constraint that the visual representation of these metamodels is strictly driven by the underlying model. That is, it may be possible to provide plausible representations of a model's confidence, command, and prediction path, by substituting representations developed for another model. For instance, the decisions of a deep neural network are difficult to explain to a non-expert, but if explanations are presented as though the system is performing case-based reasoning (shown to be an intuitive approach in the study in Chapter 4), then that may suffice to produce the necessary mental model soundness. One model could be used as a metaphor for another.

Concretely, the representation developed for one type of model can be substituted for another, if that representation depends solely on the quantities produced by the metamodels. For instance, in Chapter 4, a visualisation for the k -nearest neighbours model is shown, which depends only on a notion of model confidence and proximity between data points. This visualisation could be used to support interaction with a model such as a deep neural network, for which a direct visual representation is difficult to design, by simply using the computed proximity and confidence values from the neural network, and 'pretending' that the system was using k -nearest neighbours. This of course is dependent on the availability of a suitable substitution, which does not diverge too far from the true behaviour of the underlying model, and appropriate interface considerations for cases where a divergence has occurred. This idea is not pursued further within the scope of this dissertation, but is an interesting new frontier for metaphor in HCI.

3.3.2.1 Example metamodel applications

Some interactive machine learning systems are already benefiting from metamodel implementations, and can be usefully augmented by considering additional metacognitive models, or by newly considering their existing implementations.

Image segmentation: The *Crayons* application (Fails and Olsen Jr, 2003) is a classic example of interactive machine learning. By "painting" positive and negative examples onto an image, the user builds a classifier which can segment areas of an image into two classes, for example, a classifier which can classify human skin from non-skin objects in an image. *Crayons* provides direct visual feedback on the image itself, by respectively darkening and lightening the negatively and positively classified areas. With a confidence metamodel, instead of a fixed hue, the image could be overlaid with a colour corresponding to the confidence with which pixels were classified, as in Figure 3.3. This would further help the user refine their classifier. For example, it would be possible to identify regions that, while correctly classified, only just cross the decision boundary and thus have low confidence.

Email classification: Kulesza et al. have investigated assisting users in debugging rules learnt by a naïve Bayes model for classifying emails into user-generated categories (Kulesza et al., 2009, 2011, 2015). They present *EluciDebug*, a visual tool for providing explanations of the naïve Bayes classifier's classification decision with respect to a given email. They present an explicit confidence metamodel, which can be used to sort emails and focus user attention on emails which may have been misclassified. They present an explicit prediction path metamodel, wherein the entire set of weights used to make the decision can be inspected through a series of bars, and thus it is apparent whether the classification was straightforward, dominated by a few extreme weights, or complex with a wide distribution of potentially conflicting weights.



Figure 3.3: Left: a simplification of the original *Crayons* interface, which shows pixel classification based on a binary threshold. Right: confidence-based colouring exposes how the classifier may still be uncertain about the fingers, and additional annotation may be beneficial.

EluciDebug also approaches a command metamodel. Sizes of different folders, which represent different classes, explain the models’s priors for an unknown message belonging to any given class. This approaches a command metamodel since it alludes to the distribution of training examples the machine has thus encountered. It does not situate these examples in the input domain. One might envision a plot of all training examples, along with their text, projected from the high-dimensional space in which they reside onto a 2D manifold, such that deficiencies in the algorithm’s experience can be identified (a notable caveat being that such a projection may not be available). This is precisely the approach taken by Amershi et al. (2009) for interactive image classification. A full command metamodel, defined with respect to the input space (e.g., word vectors, or a latent semantic document embedding), would also enable appropriate training examples to be elicited. This could take the form of either identifying emails which would greatly improve the overall confidence of the classifier if a label was obtained for them, or could extend to the artificial synthesis of emails whose labels would satisfy the classifier’s information needs.

Concept evolution in images: The *CueFlik* application (Fogarty et al., 2008) presented a visual, programming-by-example method for designing rules to sort images into categories. Kulesza et al. (2014) refine this by acknowledging that users may not initially, or ever, have well-defined mental concept models – a key characteristic of the new programming paradigm discussed earlier. In their system the user is walked through a sequence of images which can be selected as belonging to a suggested class, or not. Automatic summaries of categories are generated to help the user remember what was distinctive about a particular category. Similar images from the corpus are displayed to assist the user in deciding whether creating a new category is warranted. Thus, the user refines their understanding, as well as the machine’s understanding, of the categories they are creating. Categories are suggested through an algorithm similar to content-based filtering. Currently, the only visual indication is a yellow star icon next to the suggested category. A confidence metamodel would enable an interface where different category labels are ranked, sized, or coloured according to the machine’s confidence. This would help users identify categories which are potentially only weakly described by the training data.

Representations of prediction paths, linked to the input space, could alert users to trivial simplifications being exploited by the algorithm, as in the pneumonia example. An example of how this might occur in image classification is as follows. Consider a classifier being trained to categorise images of dogs and cats. Most pictures of dogs are taken outdoors on green lawns, and most pictures of cats are taken indoors. With a carelessly selected dataset, one may produce a classifier which merely detects the colour green. A prediction path metamodel could highlight how many, or which of the input image features are being used to make a decision, enabling the user to decide when to enrich the dataset or when to prune the feature space to prevent oversimplifications of the domain.

Recommender systems: a common problem with music recommender systems, such as the engines underlying Pandora or iTunes radio, is that for an accurate model of your preferences to be built, the system needs to observe many examples of your listening history. As a consequence, users of such systems typically abandon the service before an accurate model is built. Researchers have investigated fast-converging estimates for recommendation systems. For example, Herbrich et al. (2007) tackle the issue of effectively recommending opponents in multiplayer games. It is important that opponents are well-matched, otherwise the game is not fun to play for either party. It is also important that these recommendations converge quickly, and that it is not necessary for a player to play several mismatched games before the system is able to correctly estimate their skill.

These recommendation systems do not expose the underlying uncertainty associated with each prediction. Through visible indicators of progress and improvement, showing how the confidence of the system improves over time, and how its command of the domain of music preferences improves with exposure to new examples, the user may be made more sympathetic to the amount of time and data required to properly train such systems.

Intelligent home devices: devices in our homes are getting increasingly intelligent. For instance, the Nest thermostat⁴ learns daily usage patterns and begins to adjust itself. Similarly, ‘smart’ refrigerators⁵ can detect when a particular item is running low and place an online order. These devices may ostensibly be programmed through general-purpose Internet-of-Things languages, such as IFTTT.⁶ However, the primary programming interfaces of many devices is direct interaction, so that these devices can learn over time. In these cases, it can be quite helpful for the system to express parts of its model to the user.

⁴<https://nest.com/thermostat/meet-nest-thermostat/> (last accessed April 29, 2018)

⁵For example, <http://bit.ly/evning-standard-smart-fridge> (last accessed April 29, 2018)

⁶<https://ifttt.com/wtf> (last accessed April 29, 2018)

3.4 Analytical modelling is constructivist learning

Analytical modelling systems allow end-users, often non-experts, to build and apply statistical models for their own uses. Constructivism is the view that human learning occurs when ideas and experiences interact (Fosnot, 2013). In this section, it is argued that the objectives of analytical modelling can be interpreted as constructivist. By so characterising them, constructivist learning environments are shown to have relationships with analytical modelling systems, with potential implications for design.

This dissertation has previously identified two tasks: model-building and analytics. The model-building activity is not principally concerned with the *structure* of the model, but rather its ability to robustly predict real-world outcomes, and so it is usually acceptable for the model to have large numbers of undecipherable parameters, and convoluted structure. The desired outcome is a reusable model. In contrast, the analytics activity is concerned with the interpretation of data through the model, and typically involves models with few, readily-interpretable parameters. The *analytical modelling* systems proposed in this dissertation incorporate elements of both tasks.

Constructivism is a theory describing the human learning process; the manner in which human knowledge is generated. It posits the view that human knowledge is constructed as a result of the interaction between a person's mental models and their perceptual experience. This stands in opposition to the naïve psychology made explicit by Heider (Baldwin, 1967) that knowledge is information, and consequently, learning is analogous to information delivery; in this instructionist view, one can optimise learning purely by training teachers to transfer information more effectively. Constructivism has had many influences but is largely attributed to Piaget (Wadsworth, 1996). Although evidence of the benefits of constructivism as a pedagogical tool has been mixed (Kirschner et al., 2006), the theory has been highly influential on learning theory and educational reform.

Computer science education has been the source of some notable applications of constructivist theory, such as Papert's Logo programming system (Abelson and DiSessa, 1986). Logo's emphasis on direct visual representation and feedback through turtle graphics gave novice programmers a direct perceptual experience of the effects of their code, increasing the surface area for interaction between their ideas and experiences, facilitating the construction of new or better mental models. The idea has been refined and combined with powerful notions such as blocks programming and multiple representations, to create sophisticated constructivist environments such as Alice (Cooper et al., 2000), Scratch (Resnick et al., 2009), and DrawBridge (Stead and Blackwell, 2014).

3.4.1 Implicit and explicit learning outcomes in analytical modelling

As noted, end-user machine learning systems have multiple objectives. Some, such as Crayons, build a reusable model, whereas others, such as Behrisch et al.'s, analyse a dataset with the aid of a statistical model. Analytical modelling tools, such as those described in the following chapters, aim to bridge the two, and additionally expose the end-user to statistical concepts.

Each of these objectives can be restated as *learning outcomes* for the user:

1. Model-building: learning about the model instance, its strengths, weaknesses, coverage of training data, fitted parameters, etc.
2. Analysis: learning about the structure, statistics, features, etc. of the data.
3. Exposure to statistical concepts: learning about a particular algorithm, or learning about generic statistical modelling concepts such as training and testing, class representation, noisy data, outliers, etc.

Systems such as Logo aim to maximise constructivist knowledge generation by maximising the opportunities for interaction between experience and ideas. Conveniently, the feedback loop between system and user which drives interactive machine learning is also a rich source of experience generation. As the user adds training examples or manipulates other parameters of the model, the system illustrates its current understanding through externalisations such as the translucent overlays in Crayons, an experience which causes the user to update their mental model of the system's intelligence.

Thus, analytical modelling systems are constructivist learning environments just like Logo and Scratch, but here the 'programs' being written are not graphics or animations, but models, and the programming language is not a blocks language, but the stream of interactions, including annotation, labelling, and parameter adjustment. However, unlike Logo and Scratch, 'learning' in the sense of acquiring concepts and gaining real-world skills is not an explicit outcome in analytical modelling systems. Rather, in analytical modelling systems, production of the desired result, whether reusable model or analytic insight, is incumbent on a series of implicit intermediate learning outcomes.

3.4.2 Critical constructivist issues for analytical modelling

Many interesting challenges and opportunities arise from the view that analytical modelling systems are really constructivist programming systems that generate certain types of knowledge through an experiential interaction loop. Virtual learning environments grounded in constructivism, called constructivist learning environments (CLEs), have been developed for domains outside programming. For instance, the *Lab Design Project* was a hypermedia system designed for researchers to practice sociological research skills and to learn about how lab design shapes scientific practice (Honebein et al., 1993). The *Jasper* series (Pellegrino et al., 1992) developed at Vanderbilt immersed students in vivid stories to encourage situated response to mathematics problems. Consequently, much research has focused on the design of such CLEs.

The following paragraphs highlight some established principles of CLEs that ask meaningful critical questions for the design of analytical modelling systems, based on a review of a few core texts on the design of CLEs (Cunningham and Duffy, 1996; Hannafin et al., 1997; Honebein, 1996; Jonassen, 1999; Jonassen and Rohrer-Murphy, 1999; Lebow, 1993; Taylor et al., 1997).

Task ownership

Users learn by working towards a problem which they see to be relevant and reflective of real-world situations. Such a stimulus for authentic activity causes users to be goal-oriented and intrinsically motivated to complete the task. It is typical for analytical modelling systems to satisfy this criterion, as both model-building and analytics applications facilitate an end-user task; users only engage with these systems precisely because they need to solve a real-world problem of which they have already taken ownership.

Ill-defined problem

Users better engage with problems that are ill-defined. Firstly, because users can invest in the problem by creating defensible interpretations and judgements. Secondly, because users are less likely to be deterred by the prospect of providing an “incorrect” solution as none exists. Analytical modelling is inherently ill-defined. As discussed earlier, with traditional software engineering problems the definition of ‘bug’ is typically uncontroversial, and the programmer can decide whether any given behaviour of their program is desirable or undesirable. However, in model-building and analytics, the user cannot know the ‘right’ answer. Instead, these activities are dominated by ill-defined questions, such as: does the model accurately capture the domain? Is it overfitting? Are these variables interrelated? Is this analysis leading to sound conclusions?

The design of early IML systems exhibit what may be termed ‘techno-pragmatism’, where the interaction is designed around satisfying the technical needs of statistical models. Indeed, this has led to an odd juxtaposition between the objectivist requirements of machine learning algorithms, and the open-ended nature of the tasks being facilitated. To build a robust machine learning model requires high-quality training data with clearly discriminable classes, representing well-defined concepts. The user’s task is to recognise and label objects, organise them coherently, and integrate them with existing knowledge – a decidedly instructionist approach. By contrast, in the constructivist view, objects do not have absolute meaning; meaning is constructed by the individual as a result of experiences and situated beliefs. This type of activity requires a rich context where meaning can be negotiated and understanding can emerge and evolve.

When IML systems hit the limits of the objectivist approach, new interaction design techniques coupled with powerful inference algorithms have been shown to provide a unique middle ground. Some examples of this include *structured labelling* (Kulesza et al., 2014) and *setwise comparison* (Sarkar et al., 2016), where training labels are assumed to be ill-defined, and the user and system are together responsible for gradually co-forming stable notions of labelled concepts, fit for use in machine learning systems.

Perturbation

Perturbation (or disequilibrium, in Piagetian terms) drives the learning process. It refers to a stimulus which does not conform to, or gently subverts, the mental model of the user, forcing them to construct new knowledge in order to ‘accommodate’ this experience. The introduction of perturbations, and encouraging the strategic exploration of errors, is already a central issue in IML systems, since building effective and interpretable models revolves around the activity of addressing errors made by the machine-learned model.

CLEs acknowledge that errors have deeply embedded negative connotations in our socio-cultural environment. As the intended learning outcomes of analytical modelling systems become more explicit, designers need to be sensitive to the impact of errors on learners’ motivations, and the potential for the misattribution of poor instructional outcomes.

This section began by introducing three matters (*task ownership*, *ill-defined problem*, and *perturbation*) with which analytical modelling systems already engage to some extent, but for which a new theoretical grounding based in constructivist design has been elaborated. Next, four further issues core to CLEs that shed new light on analytical modelling systems will be discussed.

Reflexivity

A critical self-awareness of one’s learning, beliefs and knowledge is central to constructivist environments. Reflective users take control over and responsibility for their thoughts, and create a defensible catalogue of provenance for their knowledge. Analytical modelling systems currently do not promote critical reflection, but a promising solution is to capture the user’s interaction history in detail, and facilitate simple querying and browsing. Creating such ‘graphical histories’ (Kurlander and Feiner, 1988) is not technically straightforward, but design solutions exist in visual analytics (Heer et al., 2008), as well as sketching (Zhao et al., 2015) and source code change history (Yoon et al., 2013), which demonstrate how such affordances support reasoning about knowledge provenance through direct manipulation of the knowledge construction history.

Collaboration

Learning takes place in a social context. The construction of meaning, like so many other activities, seldom occurs individually. The ability to perform in the professional world is predicated on group contexts, unlike the artificially individualistic school and classroom settings. To this end, CLEs often incorporate collaborative activities intended to promote dialogue and encourage the social exposure of ideas. IML systems are typically not designed with collaboration in mind, but may import lessons from collaborative analytics (Heer and Agrawala, 2008) in order to do so.

Task in context

Knowledge construction is context-dependent. Within a particular setting, it is historically developed, evolved over time in conjunction with culture. Moreover, beliefs and opinions are constantly adjusted by socially-mediated expectations. This process tends towards increasing common ground (Clark et al., 1991), resulting in an increased robustness of mental models. Professional customs, skills, workflows, and institutional expectations all filter and sculpt learning. The design *process* for analytical modelling systems is sensitive towards these issues, but what this might mean for their *design* is a critical question.

Tool mediation

The process of creating knowledge is mediated by tools and symbols. Just as carpentry is not merely teleology for the hammer but is also actively shaped by its invention, so do many aspects of modern technology influence the practice from which they emerge. For example, email hasn't merely made us more efficient communicators – the nature of that technology has radically changed our paradigms for communication. The invention of tools may be a cultural necessity, but the tools, in turn, transform culture.

Analytical modelling systems, especially those with an emphasis on analytical outcomes, need to be aware of their role as cultural mediators. Boyd and Crawford (2012) and Blackwell (2015b) have highlighted as a concern the fact that knowledge is shaped by the constraints, assumptions, and contexts of statistical algorithms and the data on which they operate. For instance, are Twitter posts representative of global sentiment, simply because they constitute a large sample? Analytical modelling systems embody epistemological and ontological assertions, which are currently implicit. This is made further problematic because statistical inference is the subject of much dispute. We are perhaps not ready to settle into ideologies when there is ambiguity in several areas. For instance, consider the debate between frequentist and Bayesian approaches, which differ on such fundamental axioms as the interpretation of 'truth' (Bayarri and Berger, 2004; Efron, 2005). Since these analytical systems constitute the indispensable *umwelt* of our collective experience of data, it is well worth these assumptions being made explicit.

3.5 Design principles for analytical modelling systems

This chapter has presented three new theoretical perspectives, summarised below.

- **Visual analytics is end-user programming**

Visual analytics can be considered a form of end-user programming, because informal reasoning driven by visualisations helps generate, test, and eliminate hypotheses, in much the same way as does writing statistical programs to formally encode statistical hypotheses in a language such as R. Moreover, many visual analytics systems present the visualisation as an abstract object, so constructing a reusable visualisation to facilitate a certain type of insight, separate from the underlying dataset, is a form of programming.

- **Interactive machine learning is dialogue**

End-user machine learning represents a paradigm shift in programming, where the dominant mode of programming has moved from one with well-definable mental models to one without. This is accompanied by a movement from a direct, explicit information channel (*the program itself*) to an indirect, meta-information channel (*about the program*). Previous work in explanatory debugging and interactive machine learning has shown several different items which may be present in this meta-information channel, elevating our interaction with programs to a status resembling dialogue. An addition to this channel has been proposed: metamodels of machine learning. This framework is grounded in the engineering requirements for providing such types of information for intelligent systems.

- **Analytical modelling is constructivist learning**

The interaction loop of analytical modelling systems facilitates constructivist learning, as it maximises the interaction between the end-user's experience of the model, and their ideas regarding the model status. Parallels have been drawn between interactive machine learning systems and constructivist learning environments. While interactive machine learning systems have so far had a certain set of pragmatic design influences, this constructivist interpretation opens up new avenues and implications for design.

From these observations, four principles for the design of analytical modelling systems are derived, with a focus on non-expert end users. They are: begin the abstraction gradient at zero, abstract complex processes through heuristic automation, build expertise through iteration on multiple representations, and support dialogue through metamodels.

Principle 1: Begin the abstraction gradient at zero.

Reduce *representational expertise* requirements by always including at least one representation with zero abstraction. This begins the abstraction gradient (Green and Petre, 1996), which refers to the range of levels of abstraction exposed by a notation, at zero.

Concretely, in the case of analytical modelling tools, this corresponds to having at least one direct visual representation of the data being operated upon. In contrast, statistical programming languages require at least one level of abstraction higher than data, such as arrays or matrices, in order to apply useful operations. Spreadsheets are a notable exception; direct data manipulation is *the* central element of their success at reducing representational expertise.

Principle 2: Abstract complex processes through heuristic automation.

Reduce *process expertise* requirements in order to deploy statistical and machine learning techniques by exposing as few (hyper)parameters as possible and relying on heuristic inference. For every parameter θ of a particular computation, consider whether it is absolutely necessary to have the user provide θ , or whether an appropriate value can be heuristically determined.

For instance, when designing a system that allows users to build k -nearest neighbour models, should the user be allowed to set k , the number of neighbours used? In this case, the user requires sufficient process expertise in the k -NN algorithm to understand the impact of editing that parameter. A simple heuristic might be to use the value of k which minimises training error. The design decision to apply heuristic inference is a tradeoff between the reduction in process expertise required to provide the parameter, and the quality of the heuristic.

Principle 3: Build expertise through iteration on multiple representations.

Analytical modelling comprises of an interaction loop involving training and deploying a model. Accompany this with multiple representations of the model and its output at varying levels of complexity. A rapid, incremental interaction loop with multiple representations provides constructivist scaffolding through which users can be exposed to increasingly complex concepts, building both *representational* as well as *process* expertise.

For example, when designing a system for building decision tree models, non-experts may not be familiar with navigating the features of their data in a hierarchical structure. If a flattened representation of the tree were to be presented at the same time as a more advanced hierarchical representation, the user would gradually become familiar with the correspondence between representations.

Principle 4: Support dialogue through metamodels.

Support the user by providing information *about* the model, such as the model's confidence, command, and prediction path. When building a complex model, it is difficult to gain insight into what action should be taken next by inspecting the workings (parameters) of the model directly. This must necessarily be framed as a dialogue since the goal is ill-defined and both user as well as system have some degree of agency. Concretely, indicating model confidence and prediction path helps the user evaluate the quality of each prediction. Illustrating model command helps evaluate the quality of the training dataset.

3.6 Conclusion

This chapter has presented three new theoretical perspectives: that visual analytics is end-user programming, interactive machine learning is dialogue, and analytical modelling is constructivist learning. Four design principles informed by these perspectives have been briefly outlined: begin the abstraction gradient at zero, abstract complex processes through heuristics, build expertise through iteration on multiple representations, and support dialogue through metamodels. The subsequent two chapters present concrete examples of design artefacts which influenced, and were in turn influenced by, these principles.

GATHERMINER

This chapter presents the design and evaluation of Gatherminer, an analytical modelling tool for detecting and explaining patterns in time series datasets. Gatherminer applies all four design principles. Furthermore, it exemplifies selection-as-annotation, the idea that labels for a supervised learning algorithm can be obtained from user selections on an interface. A reorderable, colour-mapped matrix is the substrate upon which such selections are made. A decision tree is trained on the user selection and is presented as an interactive treemap through which an analyst can critically evaluate multiple explanations. A user study showed that with Gatherminer, analysts without domain expertise discovered more interesting patterns, explained them more correctly in terms of the time series' attributes, and were more confident with their analysis, as compared to Tableau, their standard tool.

This chapter presents research described in the following papers:

- **Visual discovery and model-driven explanation of time series patterns.** Advait Sarkar, Martin Spott, Alan F. Blackwell, Mateja Jamnik. *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (pp. 78–86). Highly commended paper award (honourable mention).
- **Visual Analytics as End-User Programming.** Advait Sarkar, Alan F. Blackwell, Mateja Jamnik, Martin Spott. *Psychology of Programming Interest Group Work-In-Progress Workshop 2015 (PPIG-WIP)*.

4.1 Introduction

Previous chapters have motivated the questions of whether tools can be built for analytical modelling, and whether they can be made useful for non-experts. In this chapter, Gatherminer, an analytics tool which incorporates model building, is described. Gatherminer reduces requirements for many types of representational, process, and domain expertise. The system enables non-expert users to identify interesting patterns in a database of time series without knowing the shape of the patterns beforehand, and to find a concise explanation of these patterns in terms of the relevant subset of time series attributes.

Visual analytics tools, as used for exploratory statistical analysis, can be regarded as a class of visual programming language. Like spreadsheets, these tools focus on presenting data, rather than the control structure of algorithms operating on the data. They also share some properties of direct manipulation interfaces, in that the user focuses on visual output from the system rather than the input (the numeric data being analysed). The “programs” being written are statistical models and tests of statistical hypotheses, which could alternatively be implemented in R or Python. The human-centric research goal is to improve on the usability of those classical languages, enabling a wider range of end-users to engage in statistical analysis, and also making professional analysts more effective.

The analysis of time series data is a ubiquitous exercise. For instance, analysts in a retail setting might study series of sales and inventory, hoping to understand patterns which help them optimise various aspects of their retail system. The operators of a large communications infrastructure may study series of network faults and device operation, to diagnose issues and improve customer service. Financial analysts study series of stock exchange rates to spot emerging trends and exploit opportunities. The work presented in this chapter is grounded in a specific application domain – analysing patterns of network faults in the BT communications network.

The BT problem

At BT Research and Technology, analysts study time series of faults in the various devices on BT’s telecommunications network. An international network infrastructure is comprised of the order of 10^6 devices, including hubs, routers, cables, etc. in the core network, in exchanges, and in customers’ homes. Each device is characterised by hundreds of metadata attributes, such as their geographic location, the customer type served, etc. Every day, faults on the network are reported by customers, through devices raising alarms, and through field engineers performing maintenance. These fault reports are registered in a central database. Thus, a time series of daily fault counts is created for each *type* of device (i.e., all devices which share metadata properties).

The role of an analyst is to identify patterns of faults in subsets of this large database of time series, and perform analyses of the potential causes of these patterns. Analysts might look for interesting behaviour, such as “devices with unusually low fault rates”, or “devices with an inflexion in the time series.” Once this behaviour is found, a corresponding explanation is sought, such as “do any device types consistently seem to have inflexions?”, or “what attribute values are predictive of devices with unusually low fault rates?” These explanations are then used to drive business decisions, such as investment allocation and special investigations.

However, the analysts face two important challenges. The first is that the shape of interesting patterns is not known beforehand, so the system cannot be querying-oriented; “interesting” time series cannot simply be retrieved. Even if an interesting pattern is known, it may not always be straightforward to express using the standard tools available to them, such as their relational database system. The second important challenge is that of finding a concise explanation for these patterns in terms of the metadata attributes. Having hundreds of attributes, each with several values, leads to an explosion of attribute-value combinations which cannot simply be each inspected in turn. A further complication is that these analysts are experts in the domain of the network data but have limited statistical expertise.

In observing this process at BT, we noticed that the task of detecting and explaining interesting patterns is typically performed using opportunistic approaches with notable drawbacks: (1) they rely heavily on the domain expertise of the analyst to guide the exploration of the large space of attribute-value combinations; (2) they may result in interesting features being overlooked; (3) they may result in spuriously correlated attribute explanations being ‘discovered’; (4) they rely on extensive manual attribute inspection. Consequently, current methods result in analyses which are *incomplete*, *inaccurate*, *slow*, and leave analysts feeling *unconfident*.

Each of the aforementioned drawbacks is directly addressed by Gatherminer’s design. The evaluation section of this chapter presents a study comparing Gatherminer against Tableau,¹ the tool of choice for the expert analysts at BT, in which it was found that analyses using Gatherminer were more complete, more often correct, faster, and improve analyst confidence.

4.2 Related work

The analysts’ objectives can be expressed through the analytical framework due to Amar et al. (2005) as follows:

- *Identifying interesting features*: detecting groups of similar time series (Cluster) by identifying trends, peaks, inflexions and their overall shapes (Extrema, Range, Distribution, Anomalies).
- *Explaining features in terms of attributes*: detecting potential links between time series attributes and their behaviour (Correlation).

¹<http://www.tableau.com/> (last accessed: April 29, 2018)

4.2.1 Visualisations for bottom-up time series analysis

Since the nature of interesting time series is unknown *a priori* (e.g., it is not possible to say whether we wish to retrieve series with peaks, troughs, inflexion points, or some other behaviour), we must enable the analyst to conduct *bottom-up* analyses, where hypotheses about interesting behaviour are first generated by inspecting the data (Pirolli and Card, 2005). For this process to be robust, a simultaneous overview is required of all the time series in this large collection. This problem has been tackled in many ways before (Muller and Schumann, 2003); a synthesis of multi-resolution techniques is given by Hao et al (Hao et al., 2007a). Pixel-matrix displays are used for representing time series in several scientific disciplines, such as gene expression data (Lex et al., 2010) and machine hearing (Haitsma and Kalker, 2002).

To facilitate better discovery of collective trends from such overviews, the technique of reshuffling data series was proposed by Bertin (1981) with his ‘reorderable matrix’. Bertin proposed a visual procedure where pieces of paper representing rows of a matrix were cut and manually reordered on a flat surface. We now have the computing power and advanced clustering techniques to adapt this method for large datasets. A survey of time series clustering techniques is given by Liao (2005). Elmqvist et al. (2008) incorporated a significant reordering step in their “Zoomable Adjacency Matrix Explorer”. Mansmann et al. (2012) explored the use of correlation-based arrangements of time series for movement analysis in behavioural ecology. The “Bertifier” is a general-purpose tool for applying reordering operations to tabular data (Perin et al., 2014). Previous work has also been done on exposing motifs in time series (Chiu et al., 2003; Hao et al., 2012). These systems do not, however, subsequently capitalise on the rearranged visualisation as an interface for conducting automated analyses of the metadata attributes.

4.2.2 Explaining behaviour in time series datasets

Bernard et al.’s system (Bernard et al., 2012) visually guides the discovery of interesting metadata properties of time series clusters, which is closely related to our goals. However, while their work was primarily focused on automated notions of “interestingness,” due to the open-ended nature of BT analyses, we must necessarily take a mixed-initiative approach, with interestingness defined by user selections. A number of interfaces have been proposed for performing information retrieval tasks on time series databases, such as sketch editors and visual catalogues (Bernard et al., 2014a; Buono et al., 2005; Lin et al., 2005), but these systems are query-driven (i.e., assume the nature of the interesting pattern is known beforehand).

The Line Graph Explorer (Kincaid and Lam, 2006) compactly represents line graphs as rows of colour-mapped values, that is, a colour-mapped matrix. This allows the user to get a full overview of the time series data in a compact space. Line Graph Explorer provides a focus+context view using a lens-like tool, and has an elaborate metadata panel which draws on the table lens (Rao and Card, 1994). The metadata panel enables analysts to explain any observed patterns in terms of the time series attributes, but relies on manual inspection and so does not scale to large attribute spaces.

Keim et al. identify “advanced visual analytics interfaces” (Keim et al., 2010), which showcase an advanced synergy between visualisation and analytics. Hao et al. describe “intelligent visual analytics queries” (Hao et al., 2007b), the process of selecting a focus area, analysing the selection, and presenting results of the analysis as appropriate. This is precisely the technique we employ.

A number of investigations have been made into improving visual interaction with various statistical procedures. These procedures include exploratory approximate computation (Fisher et al., 2012; Sarkar et al., 2015), distance function learning (Bernard et al., 2014b; Brown et al., 2012a), and advanced feature space manipulations (Endert et al., 2011; Mamani et al., 2013; Sedlmair et al., 2014). Fails and Olsen Jr (2003), Wu and Madden (2013), and Behrisch et al. (2014) present systems for interactive machine learning. These address a range of classification problems for different types of data; however, visual mining for explanations amongst a large set of attributes has not been addressed.

4.3 Design

Data is represented as a colour-mapped matrix (Figure 4.1(a)), which is rearranged to expose patterns and clusters (we refer to this as “gathering” (Pirolli et al., 1996)). To navigate this visualisation, which can become very large for massive databases of time series, we provide an overview+detail mechanism, where the overview is facilitated by a thumbnail scrollbar (Figure 4.1(b)), and detail is given through a scanning display (Figure 4.1(c)). For analysis, we use selection on the core visualisation as annotation to deploy explanation procedures such as summary bar graphs and decision tree learning (Figure 4.1(d)). An overview of the architecture is shown in Figure 4.2. The individual components are elaborated in the following sections.

4.3.1 Core colour-mapped matrix visualisation

Our primary visualisation is a colour-mapped matrix where each row is an individual time series and each column is an individual time point. Thus, a cell is a single data point within a single time series, coloured according to its value. This representation has useful properties (Hao et al., 2007a), most importantly compactness: each datum can be shrunk to a single pixel in size² before the visualisation ceases to be lossless. As each datum is directly represented at all times, this visualisation satisfies design principle 1: begin the abstraction gradient at zero.

Prior to colour-mapping, values must be normalised; by default Gatherminer uses cumulative distribution (CD) normalisation, with the option to switch to range normalisation and truncated Z -score normalisation, but a user-supplied JavaScript normalisation function may also be used.

Importantly, even though pixel matrices are perceptually inferior to line charts for comparative quantitative analysis, since they rely on colour rather than height to convey magnitude, it is the *identification* of patterns which we consider to be more important than *characterisation*. That is, it is more important for analysts to be able to spot that interesting behaviour exists rather than to immediately understand the nature of that behaviour (spike, trough, etc.). The colour-mapped matrix greatly facilitates identification. The exact behaviour can easily be further characterised by inspecting aggregate line graphs generated by selecting the time series.

²or to the resolution limit of the human eye, whichever comes first

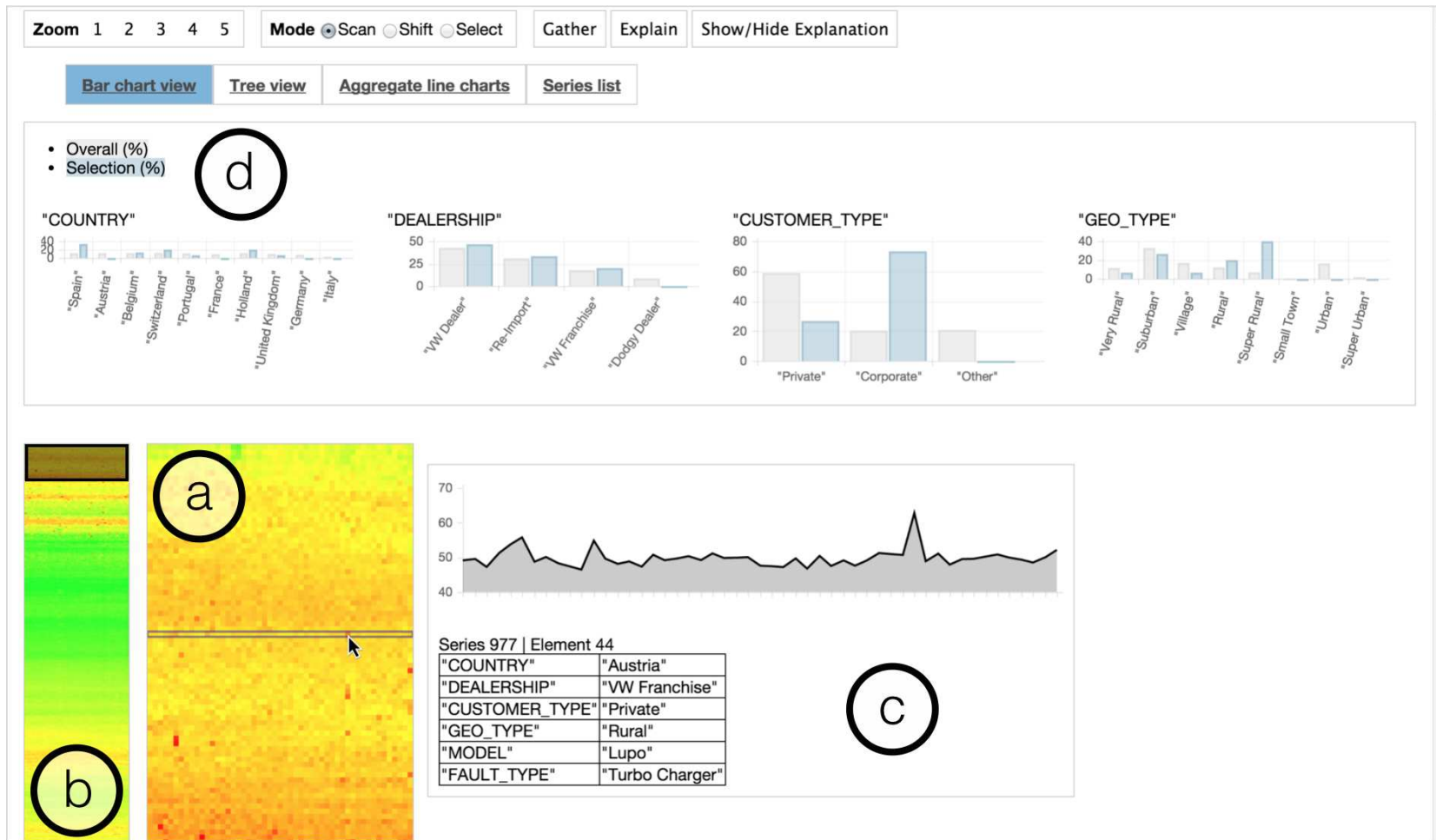


Figure 4.1: The Gatherminer software showing (a) its primary colour-mapped matrix visualisation, (b) the thumbnail overview scrollbar, (c) scanning for detail, (d) attribute charts generated during analysis.

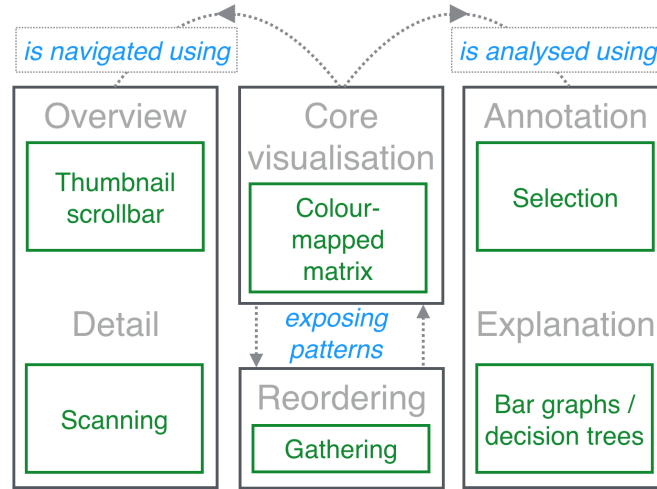


Figure 4.2: Gatherminer’s architecture. System objectives are presented in grey; their respective implementations are in green. Relationships between components are in blue.

Cumulative distribution normalisation as default

Given that we wish to prioritise pattern identification, CD normalisation presents itself as the natural way to produce a default colour map, for reasons explained in this section. A colour map is a function mapping values in the data domain to values in the colour domain (e.g., RGB triples). To allow for modularity, and to decouple colour maps from the parameters of specific datasets, we assume that colour maps take values which have been normalised to the continuous range $[0, 1]$. An appropriate normalisation function must first be applied to bring an arbitrary dataset into this range. CD normalisation works as follows. For a given multiset of real values D , a value d is normalised to yield d' by setting:

$$d' = \frac{|\{x \in D | x \leq d\}|}{|D|}$$

Essentially, this sets $d' = P(x \leq d)$ where the probability distribution is measured directly from the dataset (the ‘empirical’ distribution function). This is *almost* equivalent to setting d' to the percentile of d , the difference being that percentile refers to the probability of the strict inequality. The choice of cumulative distribution over percentile is not important as long as the colourmap function is approximately perceptually uniform (Robertson, 1988). That is, for a given difference Δ , such that $0 \leq \Delta \leq 1$, the colours corresponding to the values x and $x + \Delta$ are equally perceptually different for all $x \in [0, 1 - \Delta]$.

The advantage of CD normalisation is that it facilitates pattern identification regardless of the distribution of the underlying data. This satisfies design principle 2: abstract complex processes through heuristic automation. Higher-density areas of the underlying distribution are allocated larger proportions of the normalised range. By contrast, consider range normalisation, a popular normalisation method:

$$d' = \frac{d - \min_x\{x \in D\}}{\max_x\{x \in D\} - \min_x\{x \in D\}}$$

If a dataset following, for example, a power law distribution (as does the BT dataset) were to be range normalised, a lot of the normalised range would be allocated to values which occur disproportionately few times. More perniciously, patterns which involve the most frequently occurring elements are obscured due to the fact that these elements are ‘compressed’ into a small portion of the normalised range.

This is made clearer in Figure 4.3, which shows four different distributions as normalised by range normalisation in the left column, and CD normalisation in the right column. Range normalisation works well for the uniform distribution, mapping equally dense regions of data with equal areas of the normalised range (corresponding to an equal space of colour). However, range normalisation breaks down for the Gaussian, power law and non-standard distributions, where highly dense areas are still given equal areas of the normalised range. Note how, in contrast, the CD normalisation maps according to density; the grey lines represent equally-spaced samples in the normalised range, and the green lines project them back onto the original data range. Denser areas are accordingly allocated more of the normalised spectrum. The impact this can have on the visualisation is illustrated in Figure 4.4.

Like the normalisation method, the colour map chosen should depend on the domain. Analysts in various settings may have adopted semantic colour conventions which could inform the hues chosen in the colour map. Sometimes the analytical value from such semantic conventions will supersede limitations such as poor support for colour blindness and perceptual non-uniformity. The red-green colour map is an excellent example of this; the immediate positive and negative connotations of ‘green’ and ‘red’ in Western cultures is a strong argument for its use to represent data in several domains, such as finance and performance monitoring. However, the red-green colour map has poor perceptual uniformity and is not safe for colour blindness. Ultimately, this is a decision for the analyst to make. Gatherminer supports several common colour maps, and new colour maps can be installed in a modular fashion.

To illustrate an example workflow in Gatherminer, a real BT dataset has been extracted, with commercially sensitive material disguised so as to resemble data about faults in cars. The colour-mapped matrix representation of this dataset when initially loaded can be seen to the left of Figure 4.5. The remainder of this figure will be explained in Section 4.3.2.

Overview+detail

The warping lens in Line Graph Explorer provides focus+context (Card et al., 1999), allowing individual time series to be inspected in detail whilst still being aware of the series’ location with respect to the overall dataset. This interaction mechanism dynamically distorts the underlying visualisation, which we require to be static for purposes of selection (explained shortly), so it is not applicable to our use. Instead, we allow the user to scrub over the visualisation, and display a detailed line graph and attributes table for the series being hovered over (item (c) in Figure 4.1). This is complemented with an overview which never exceeds the height of the screen. The overview acts as a scrollbar. It contains a translucent window indicating which section of the time series is currently being magnified, which can be dragged like a scroll handle (Figure 4.6). Thus, the scrollbar and main visualisation together create an overview+detail view (Cockburn et al., 2009). Moreover, the main visualisation itself can be seen as an ‘overview’, which can be inspected in ‘detail’ through the scrubbing mechanism. This multiple-layered overview+detail has an additional advantage over the lens approach: the time series dataset can be large, containing thousands of time series, but a shrunken representation is always visible and available for use as a navigational aid.

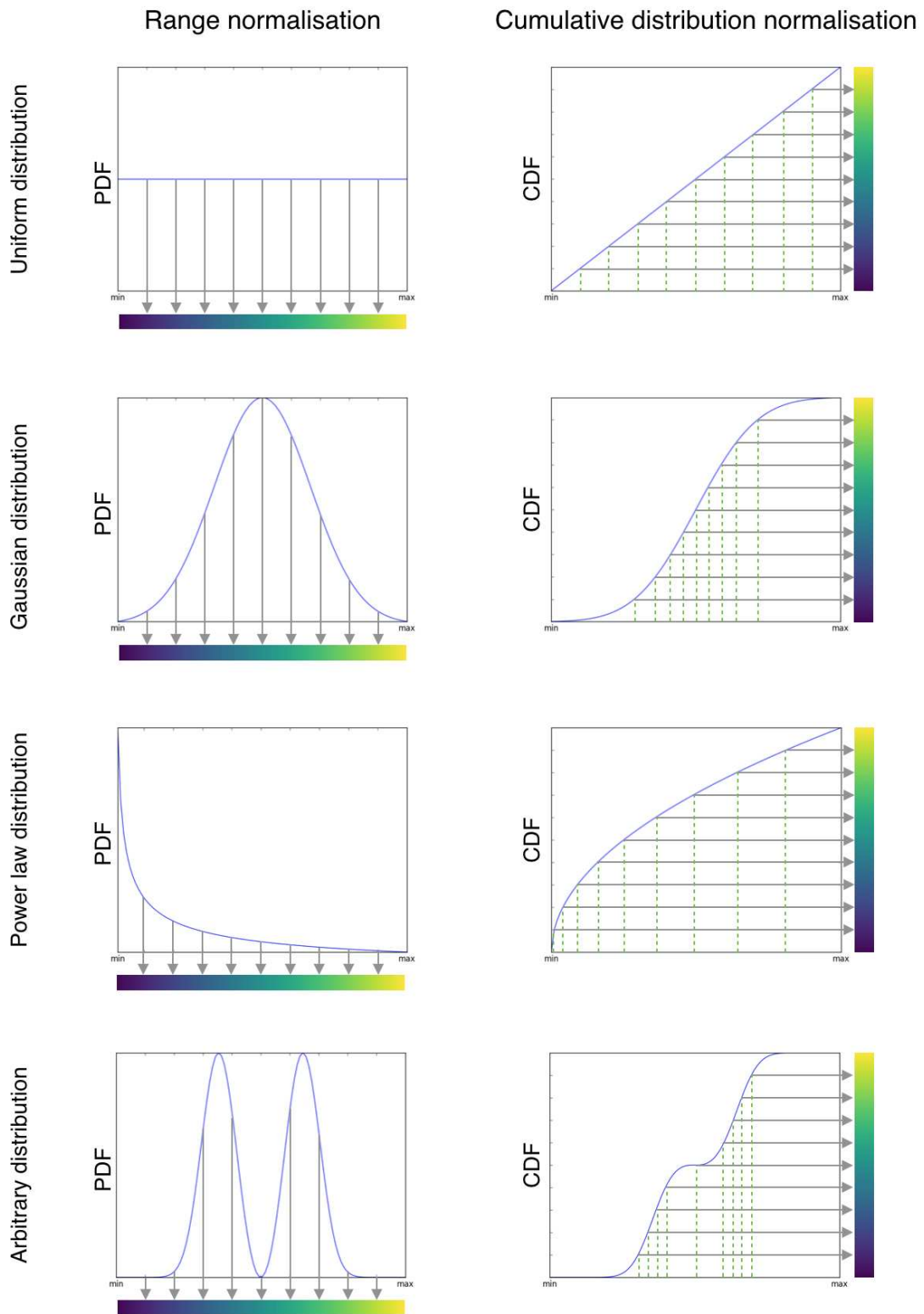


Figure 4.3: Illustration of how range and cumulative distribution (CD) normalisation map different distributions onto a colour scale. Left column: range normalisation, plotting probability density function (PDF). Right column: CD normalisation, plotting cumulative distribution function (CDF).

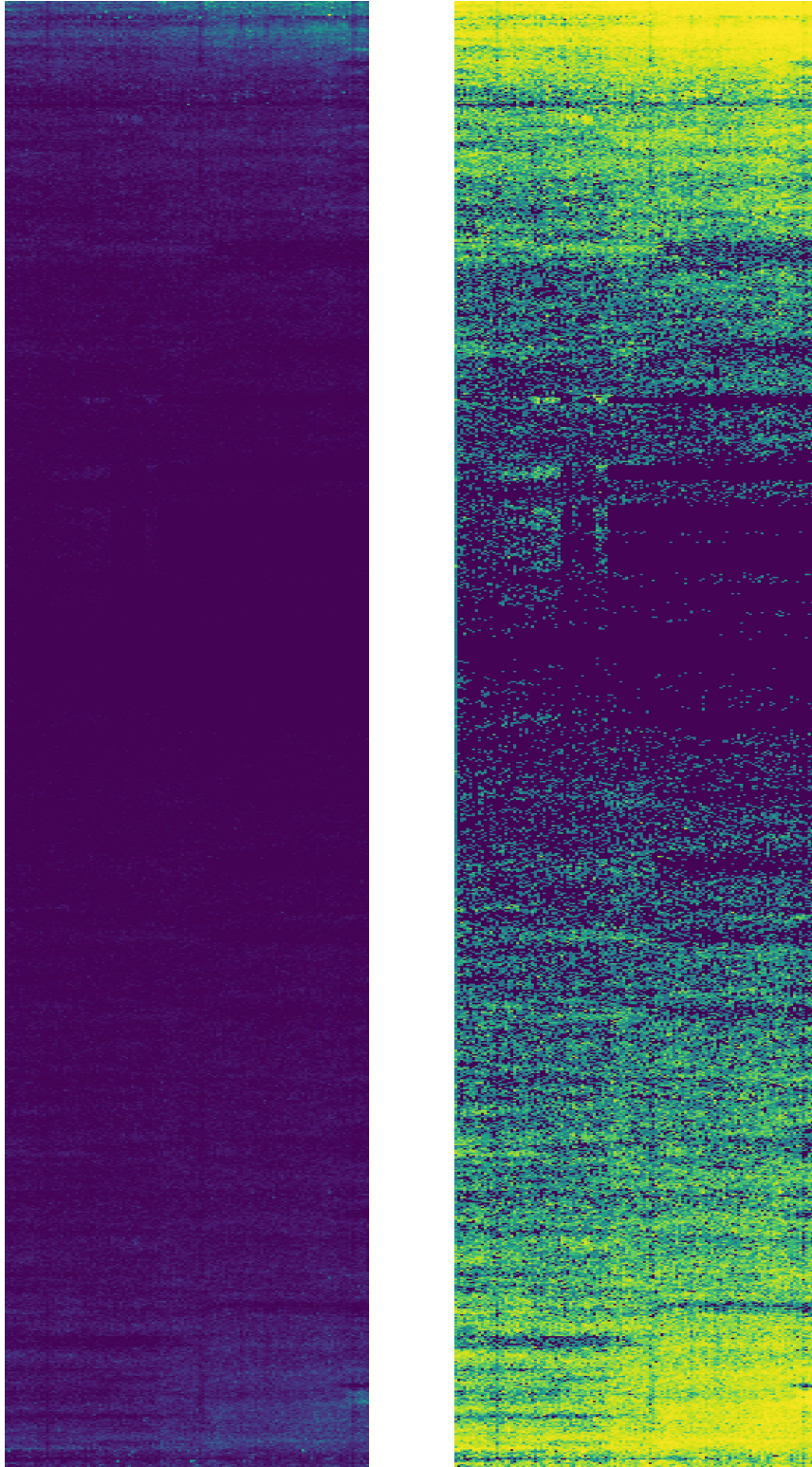


Figure 4.4: Effect of different normalisation methods on a power-law distributed dataset of 1000 series. Left: range normalised, right: cumulative distribution normalised.

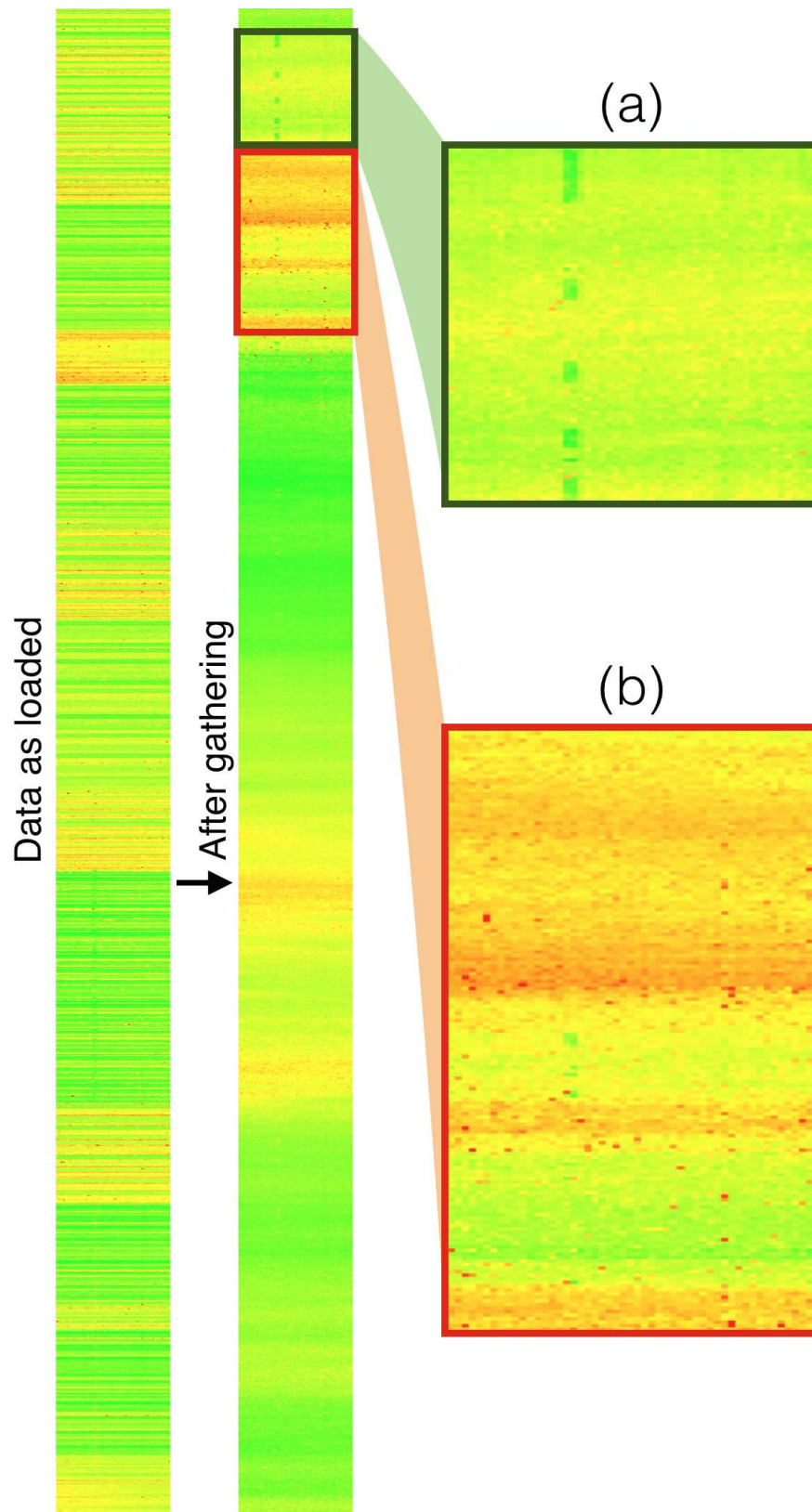


Figure 4.5: Two patterns exposed by gathering a BT dataset of 1,335 rows. (a) Time series with a trough in the middle. (b) Clusters of high-valued time series.

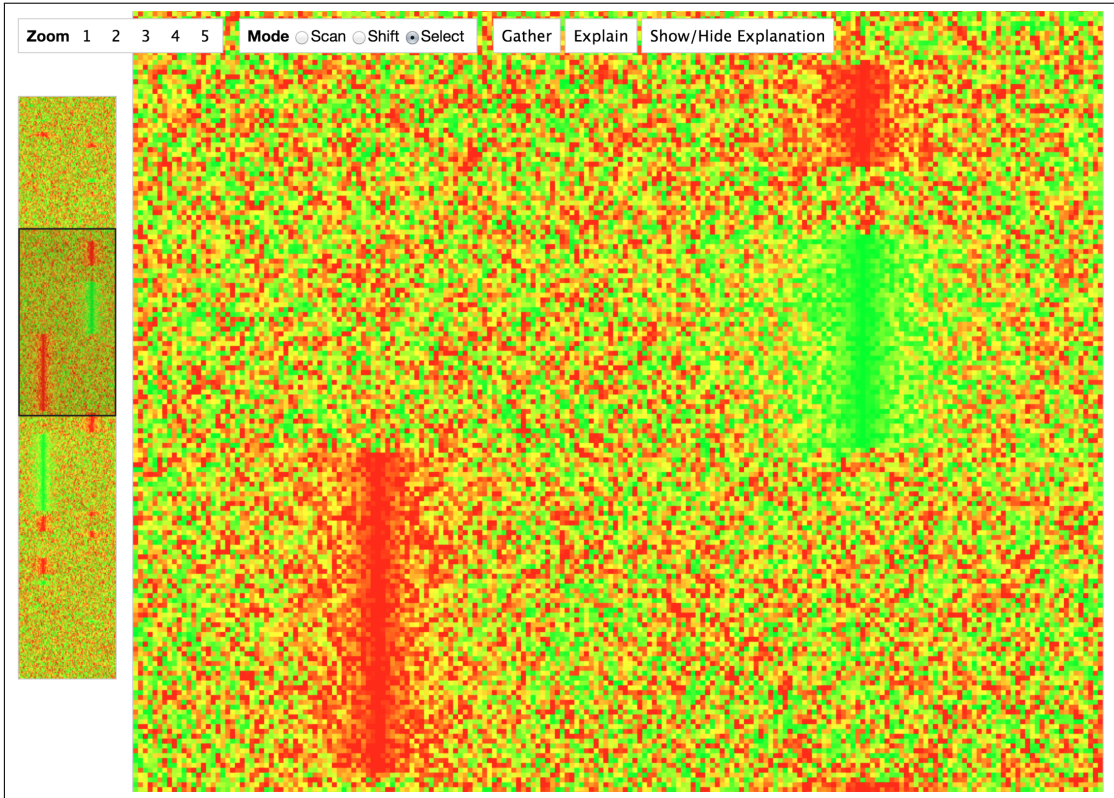


Figure 4.6: The thumbnail scrollbar appears as a shrunk overview on the left, with a translucent overlay demarcating the section of the overall visualisation currently being viewed in the main area of the window. The translucent overlay functions as a draggable scroll handle.

4.3.2 Gathering: automated layout

The visualisation format alone facilitates some analysis, but for more efficient pattern detection a layout algorithm must now be applied. A number of clustering, sorting, or optimisation methods may be appropriate here; by default, Gatherminer reorders the time series such that those which are most similar are placed close together, hence “gather.” Specifically, the final layout minimises the sum of pairwise distances between neighbouring time series. The visual perceptual principle behind this ordering is that by minimising the sum of pairwise distances between neighbouring series, we bring together series which have similar visual colour profiles. When ‘stacked up’, rows of similar colours create larger areas which are easily spotted. This rearranged visualisation may be called a ‘gatherplot’.³ The default choice to reorder based on perceptual principles also satisfies design principle 2: abstract complex processes through heuristic automation.

In the following, we use T to denote a univariate time series. The subscript T_i denotes the i^{th} element of T . In a collection of many time series, the superscript notation $T^{(k)}$ denotes the k^{th} series. Thus, $T_i^{(k)}$ denotes the i^{th} element of the k^{th} series.

³A light-hearted analogy to the scatterplot, referencing Cutting et al. (1992).

Pairwise distance between time series

By default, the distance between any two time series is a pairwise weighted metric:

$$distance(T^{(a)}, T^{(b)}) = \sum_i \sum_j c(T_i^{(a)}, T_j^{(b)}) \cdot w(i, j) \quad (4.1)$$

where

- c is a cost function which specifies how the distance between two individual elements of time series should be calculated, for example, $c(T_i^{(a)}, T_j^{(b)}) = |T_i^{(a)} - T_j^{(b)}|$, or alternatively, $c(T_i^{(a)}, T_j^{(b)}) = (T_i^{(a)} - T_j^{(b)})^2$.
- w is a decay function which specifies how the neighbourhood of each element is weighted, for example, $w(i, j) = e^{-|i-j|}$, or alternatively, $w(i, j) = |i - j|^{-1}$, or alternatively, $w(i, j) = 2^{-|i-j|}$.

These weighting functions express the idea that distances between elements in the same position in the two series are most heavily penalised, and distances between elements in different positions are not completely ignored, but are given successively lower weightings as the respective positions grow further apart.

Faster distance computation by bounded approximation

For time series of length l , computing a distance of this form between any two time series costs $O(l^2)$. However, a bounded approximation to the weighting function can reduce this to $O(l)$. Observe, for example, that for the weighting function $w(i, j) = 2^{-|i-j|}$, when $|i - j| = 7$, $w(i, j) < 0.01$. For several datasets, including BT's time series dataset, distances are no longer meaningfully affected by adding element-wise costs which have been multiplied by weights this small. That is, if all pairwise distances were computed, and ordered from least to greatest, ignoring costs added by small weights would not change this ordering. Thus, through empirical, manual tuning, a bound b can be selected to create a function w_b , a b -bounded version of any weighting function w , as follows:

$$w_b(i, j) = \begin{cases} w(i, j) & \text{if } w(i, j) \geq b, \\ 0 & \text{otherwise.} \end{cases} \quad (4.2)$$

Conveniently, in the case of w being defined in terms of the absolute difference between i and j , this bound can be translated into a bound on the summation of the j terms. Concretely, consider our example of $w(i, j) = 2^{-|i-j|}$, which becomes negligible (< 0.01) for $|i - j| \geq 6$. To calculate the distance function in equation 4.1 with a bound of $b = 0.01$, since $w_b(i, j) = 0$ for $|i - j| > 6$, we can simply calculate the distance as:

$$distance(T^{(a)}, T^{(b)}) = \sum_{i=1}^l \sum_{j=\min(1, i-6)}^{\max(l, i+6)} c(T_i^{(a)}, T_j^{(b)}) \cdot w(i, j)$$

In general, each bound b corresponds to a window λ where $w_b(i, j) = 0$ for $|i - j| > \lambda$. Then,

$$distance(T^{(a)}, T^{(b)}) = \sum_{i=1}^l \sum_{j=\min(1, i-\lambda)}^{\max(l, i+\lambda)} c(T_i^{(a)}, T_j^{(b)}) \cdot w(i, j)$$

This computation is now $O(\lambda l)$, which reduces to $O(l)$ since λ is constant.

When time series are loaded into the system, they are assigned labels from 1 to n in the order in which they appear in the data file. That is, $T^{(1)}$ is the first time series in the file, $T^{(2)}$ is the second, and so on. An ordering of the time series can be specified through a function $p : \{1, 2, \dots, n\} \rightarrow \{1, 2, \dots, n\}$, where $p(k)$ specifies the position of $T^{(k)}$ in this ordering. The (identity) ordering $p_I(k) = k$ is the default ordering just described.

Reordering

For n series a minimal pairwise distance ordering p_{min} is desired such that

$$p_{min} = \underset{p}{\operatorname{argmin}} \sum_{k=1}^{n-1} distance(T^{(p(k))}, T^{(p(k+1))})$$

Finding such an ordering is not straightforward, as the problem is equivalent to that of finding a minimal Hamiltonian path (similar to the problem of finding a minimal Hamiltonian cycle, more commonly known as the Travelling Salesman Problem), known to be NP-complete. To see why our problem is equivalent, imagine that each time series is a vertex in a fully-connected graph, and each edge has weight equal to the distance between its two vertices. A complete minimal ordering is equivalent to a minimal-weight path which touches each vertex exactly once, that is, a minimal Hamiltonian path.

Computing the distance matrix is an unavoidable $O(l^2 n^2)$, or $O(ln^2)$ if a bounded approximation is applicable. Thereafter, to compute the ordering, Gatherminer implements two alternative solvers. The first is a greedy nearest-neighbour search ($O(n^2)$), described in Algorithm 1. This algorithm is a slight variant of the standard nearest-neighbour search as it proceeds greedily from both ends of the current path, rather than from any one end. Note also that for the sake of convenience, the distance function is assumed to be symmetric, that is for any i and j , $distance(T^{(i)}, T^{(j)}) = distance(T^{(j)}, T^{(i)})$. For the asymmetric case, a little more care is required, but it is nonetheless a straightforward extension of the algorithm presented here.

The greedy search can sometimes produce poor, or even maximal (worst-case) orderings (Gutin et al., 2002). Although naturally occurring datasets rarely exhibit this pathological property, it is useful to have an alternative in such an event. Consequently, the second solver is a genetic optimisation algorithm, which is robust to such datasets. It is also $O(n^2)$, but is slower in practice due to large constant factors. The detailed pseudocode of the genetic algorithm used shall not be presented, but a high-level overview follows. A detailed example of how a genetic approach can be used to solve the travelling salesman problem is given by Mühlenbein et al. (1988). Kernel PCA is another potential technique for initialising the ordering (Schölkopf et al., 1997).

Data: L : list of labels $\{1, 2, \dots, l\}$
Result: p_{nn} : the approximately-minimal nearest-neighbour ordering

Initialisation:

p_{nn} is a new empty double-ended queue;
 $i, j = \operatorname{argmin}_{i,j \in L} \operatorname{distance}(T^{(i)}, T^{(j)})$;
push i to head of p_{nn} , append j to tail of p_{nn} ;
top = i , bottom = j ;
delete i and j from L ;

while L is not empty **do**

$i = \operatorname{argmin}_{i \in L} \operatorname{distance}(T^{(\text{top})}, T^{(i)})$;
 $j = \operatorname{argmin}_{j \in L} \operatorname{distance}(T^{(\text{bottom})}, T^{(j)})$;
if $\operatorname{distance}(T^{(\text{top})}, T^{(i)}) \leq \operatorname{distance}(T^{(\text{bottom})}, T^{(j)})$ **then**
 push i to head of p_{nn} ;
 top = i ;
 delete i from L ;
else
 append j to tail of p_{nn} ;
 bottom = j ;
 delete j from L ;

end

end

return p_{nn} ;

Algorithm 1: Greedy nearest-neighbour search for p_{min} .

The genetic solver produces an initial random population of orderings by selecting random permutations of the list $[1, 2, \dots, n]$, each such permutation being interpreted as an ordering of labels according to the list indices. For a fixed number of iterations, new ‘generations’ of individuals are produced. To produce the next generation, three steps are followed: selection, crossover, and mutation. In the selection phase, the population is sampled (with replacement) with probability proportional to individual fitness, to yield a new population of the same size as the original. The fitness of an ordering is the inverse of the total distance of the Hamiltonian path it represents. In the crossover phase, randomly selected ‘parent’ pairs of orderings are ‘interbred’ to produce a pair of children, which replace the parents in the population. The interbreeding process randomly inspects individual steps in both parents’ paths, selecting the least costly step for inclusion in the child. Finally, with some probability, each individual is ‘mutated’ through one of two types of mutations: either a randomly chosen subpath within the larger path is reversed, or two randomly chosen disjoint subpaths are interchanged. On BT datasets ($\sim 10^3$ series of length $\sim 10^2$), the genetic algorithm typically converges to a near-optimal path after around 100 generations.

While neither greedy search nor genetic algorithm is guaranteed to produce an optimal ordering, any ordering which is sufficiently close to optimal has the desired visual property when rendered as a colour-mapped matrix, namely, that similar rows are brought close together, increasing the area of repeated patterns, making them more visible.

The “Gather” button triggers reordering. The resulting visualisation exposes groups of series bearing interesting analytical features such as peaks and trends (e.g., Figure 4.7). With careful selection of distance metrics, one might also be able to expose motifs (Chiu et al., 2003; Hao et al., 2012). The colour-mapped matrix representation of our faults dataset after gathering can be seen in Figure 4.5. One limitation of this process is that each time series can only have two neighbours in the colour-mapped matrix, and so clusters are sometimes “flattened” counterintuitively, with similar rows being placed further apart than expected. This does not usually impair “pattern spotting” as it is an approximate visual process, but it does require the user to make multiple selections.

The distance metric can and should be changed for the task at hand, as data in various domains typically have very different notions of what constitutes similarity. For instance, Dynamic Time Warping (Berndt and Clifford, 1994) is a common metric for comparing time series which vary in speed or time. Currently, any user-supplied JavaScript distance function can be used. However, since our target end-users are not experts in programming or statistics, this is not a satisfactory solution. In future work, it would be useful to investigate interactive visual methods of specifying distance metrics (Bernard et al., 2014b; Brown et al., 2012a; Mamani et al., 2013). For instance, the user could specify a distance metric by dragging together rows which they feel should be placed close together, which would create a system of soft constraints that could be solved as an optimisation problem to yield a distance function.

Each time series in the collection is associated with attributes describing various properties of the series (i.e., “metadata” as used by Kincaid and Lam (2006) and Bernard et al. (2012)). For instance, the BT fault data has “Device Type”, “Location”, etc. as attributes. We denote these attributes A_j , meaning the set of values they are allowed to have. Attributes may also be continuous (e.g., *Product_Weight*). Thus, each time series is characterised by an n -tuple of attribute values (a_1, a_2, \dots, a_n) where $\forall j. a_j \in A_j$.

The two primary activities of BT analysts which are facilitated by our system can now be framed as follows:

- “Identifying interesting features” corresponds to discovering the sets of time series *Interesting* such that for $k \in \text{Interesting}$, $T^{(k)}$ contains interesting behaviour, for example, “devices with unusually low fault rate.”
- “Explaining features in terms of attributes” corresponds to discovering attribute-value tuples (a_1, a_2, \dots) which discriminate well between $T^{(k)} \in \text{Interesting}$ and $T^{(k')} \notin \text{Interesting}$. An example question is “what attribute values are predictive of devices with unusually low fault rates?”

It is worth reiterating that analysts do not necessarily know what types of patterns they are looking for. Gatherminer is not, and cannot be, a query-driven interface. Rather, it supports the *generation* of relevant queries.

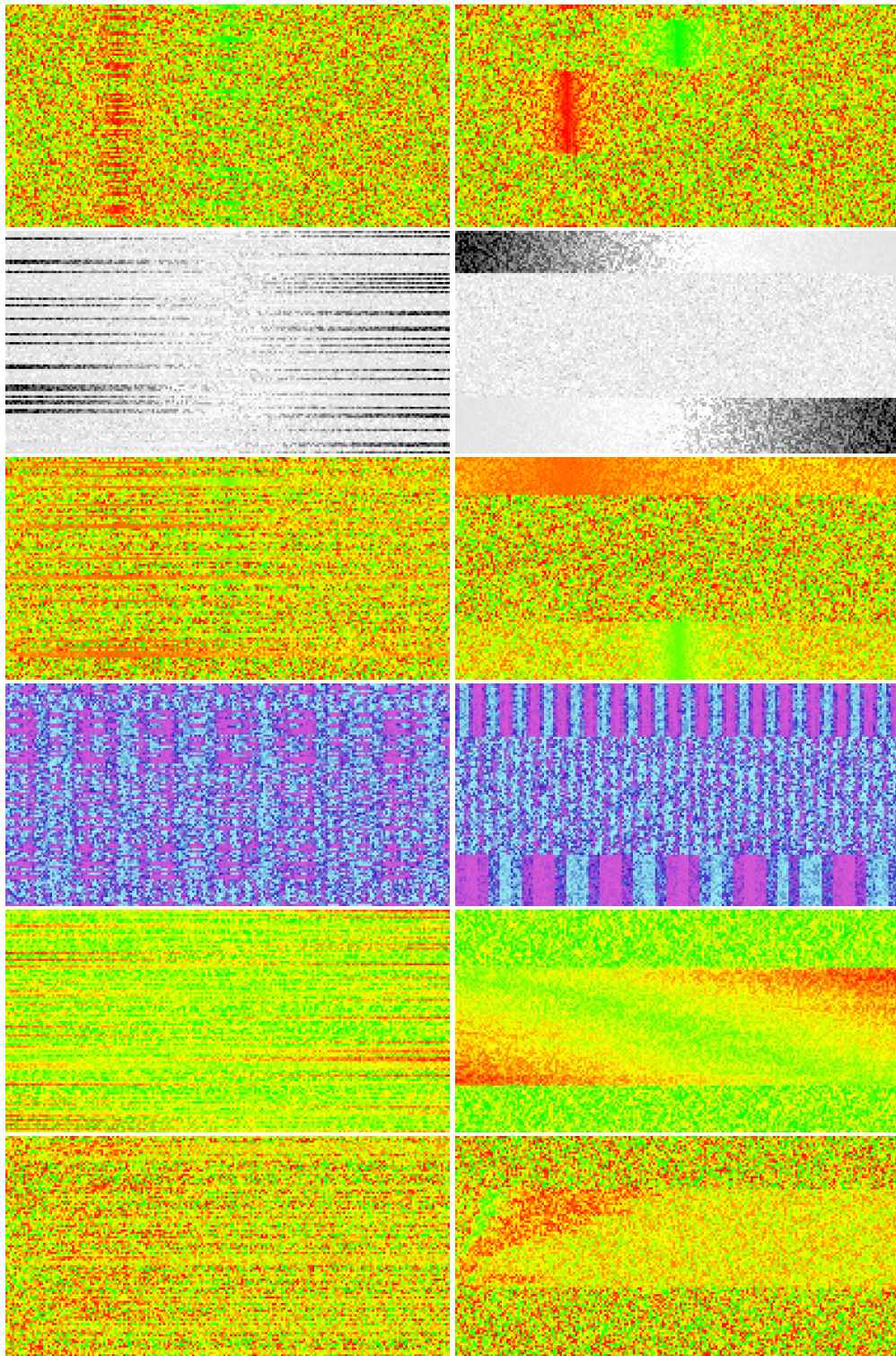


Figure 4.7: Examples of the gathering process, with the dataset as loaded on the left, and after gathering on the right, demonstrating the detection and separation of different types of patterns from noise; from top to bottom: peaks and troughs, linear trends, features of different widths, functions of different periodicities, complex cross-series cascades. Multiple colour mappings are supported.

4.3.3 Selection to annotate interesting clusters

The Gathering process exposes interesting patterns as visual artefacts such as coloured blobs and streaks. The next step is to query for explanations of these patterns directly using the reordered visualisation.

Gatherminer allows users to select regions of interest in order to mark them as ‘interesting.’ This constitutes manual annotation of a subset of data points, similar to the interactive applications presented by Fails and Olsen Jr (2003), and Wu and Madden (2013). In Fails and Olsen’s *Crayons* application, the user drew on an image to interactively build a classifier to segregate the image (e.g., a classifier that detects a human hand against a background). Similarly, in Gatherminer, the user directly annotates the visualisation to build a classifier. While *Crayons* facilitated image classification on image data, Gatherminer extends that style of interaction to time series data visualised as a colour-mapped matrix; it provides an “intelligent visual analytics query” (Hao et al., 2007b).

The gathering step is essential for this annotation to be effective. In the underlying dataset the time series may appear in any ordering, for instance the order of generation of the data entries, or sorted by attribute-values. Once gathered, however, the resultant ordering $\{T^{(1)}, T^{(2)}, \dots, T^{(n)}\}$ is such that neighbouring time series have similar behaviours, regardless of their attributes or date of entry into the original file. Thus, “interesting” time series appear in contiguous regions, allowing the user to use their selection to specify an interval $[a, b]$, or k intervals $[a_i, b_i]$ for $i = 1$ to k , such that $\bigcup_{i=1}^k \{T^{(a_i)}, T^{(a_i+1)}, \dots, T^{(b_i)}\}$ constitute the *interesting* set, and the remaining time series constitute a *not-interesting* set. See Figure 4.8 for an example of selection.

Once the selection is made, clicking the “Explain” button deploys multiple strategies (e.g., decision tree learning) to discover which attributes of the time series best discriminate the interesting (selected) regions from the non-interesting ones. Thus, the user asks the software to “explain” regions of interest by querying for explanatory attributes.

Mining for explanatory attributes

Gatherminer currently supports three explanatory visualisations, illustrating the variety of interesting possibilities for selection as annotation. Moreover, each visualisation provides a different representation, thus exemplifying design principle 3: build expertise through multiple representations. The first explanation method is a set of bar charts which compare the distribution of the attribute values in the selection against the distribution of the attribute values in the overall dataset. This builds upon the system presented by Bernard et al. (2012) for analysing relationships between time series clusters and attributes. These charts do not require statistical expertise for interpretation. “Explanations” are read off by comparing the heights of the bars. A large discrepancy between an attribute’s values in the selection and its values in the overall dataset indicates that the presence or absence of that value is highly correlated with the time series marked “interesting.” An example can be seen in Figure 4.9 (recall that the attributes are disguised to resemble a dataset about faults in cars).

The second explanatory visualisation demonstrates how the selection-based interface can be used to implement more sophisticated machine learning algorithms. Supervised learning can be formalised as the process of discovering a hypothesis $h : X^n \rightarrow Y$, given a sequence of training examples (\vec{x}_i, y_i) . The intention is that the learnt hypothesis achieves a level of generality that renders it useful for modelling and prediction purposes. Here, each \vec{x}_i is known as the *feature vector* and each y_i is known as the *label* or *class*.

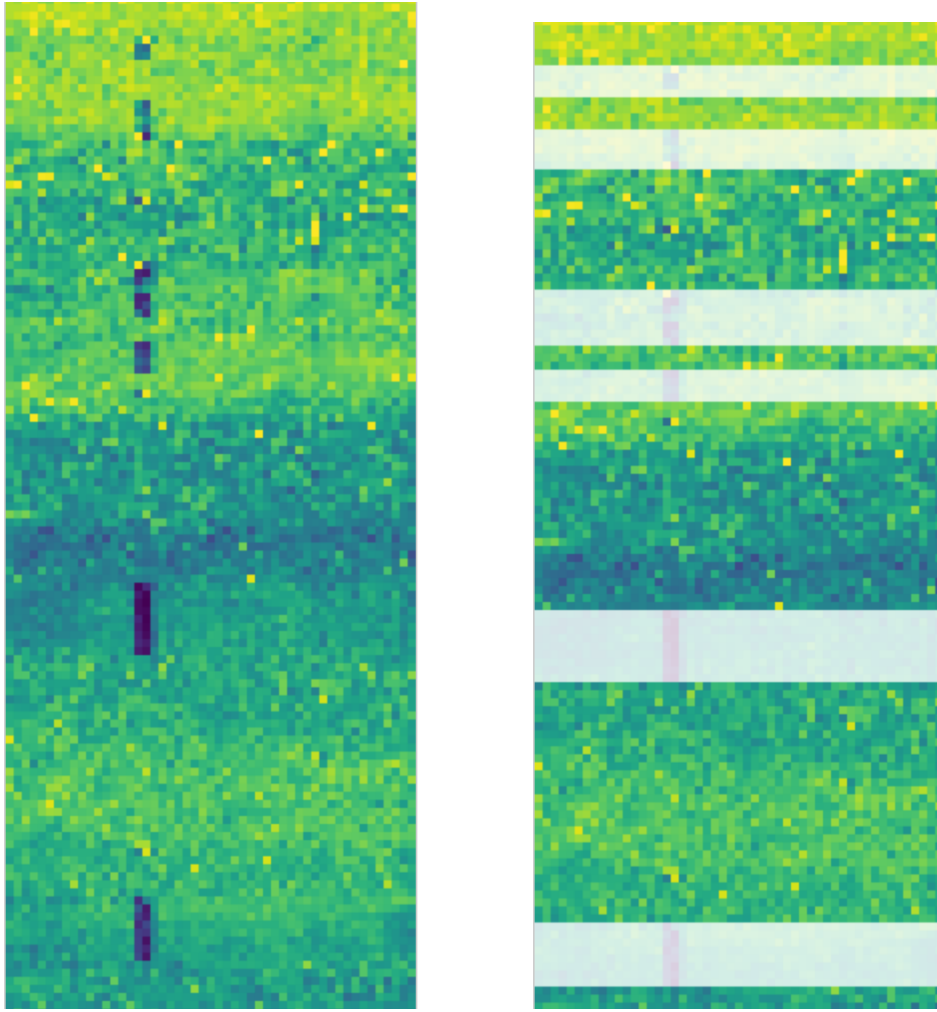


Figure 4.8: Left: a gathered dataset has exposed some series with an interesting feature. Right: these series have been selected by specifying intervals using a brushing tool.

The ID3 decision tree algorithm (Quinlan, 1986) is a natural choice for demonstrating selection-as-annotation, as it produces human-interpretable models in the form of rules. For each time series, our feature vector is the attribute vector of the series: (a_1, a_2, \dots, a_n) , and our label is a binary value indicating whether the time series was part of the selection, that is, was marked as “interesting”:

$$label(T^{(k)}) = \begin{cases} \textit{Interesting}, & T^{(k)} \in \textit{Selection} \\ \textit{Not Interesting}, & T^{(k)} \notin \textit{Selection} \end{cases}$$

The training dataset D consists of $(n + 1)$ -tuples of the form $(a_1, \dots, a_n, label(T^{(k)}))$. The ID3 algorithm can now be called on D , specifying $(\forall j. A_j)$ as the attributes and $label$ as the target attribute (class). More sophisticated tree algorithms such as C4.5 (Quinlan, 2014), which has better support for continuous attributes could be used, and the same interaction design principles apply. The ID3 algorithm has been chosen for simplicity and clarity.



Figure 4.9: Some explanatory charts for the high-valued clusters in Figure 4.5. Around 60% of the marked series have `CUSTOMER_TYPE=Corporate`, but that value only occurs in 20% of series overall. Thus, the rule `CUSTOMER_TYPE=Corporate` could partly explain the behaviour of the selected time series. Similarly, `MODEL≠Polo` and `MODEL≠Passat` and `GEO_TYPE=Super Rural` are also potential explanations. Hovering on bars reports exact percentages in tooltips.

We map the data structure resulting from the call to ID3 directly onto a tree visualisation. Explanations are read as a conjunction of nodes from root to leaf. The tree is interactive, featuring collapsible nodes, panning and zooming. Figure 4.10 shows an example.

A key advantage of deploying the ID3 algorithm in this manner is that it scales to large attribute-value spaces. The tree visualisation displays combinations of explanatory attributes using exactly the tree-depth (i.e., number of relevant attribute values) required. For instance, if a single attribute value contains complete discriminatory information about the user selection, tree expansion stops at that attribute. An example tree can be seen in Figure 4.10. The figure shows only one path, but when completely uncollapsed (i.e., showing all paths), the full tree has only 100 nodes. For just the 8 attributes in our example dataset, there are over 10^5 attribute-value combinations. The tree, constructed on an information-theoretic basis, represents only the most relevant ones.

Why not just show clusters separately?

One alternative to our neighbour-distance-minimising layout and manual user annotation process would have been to employ a traditional clustering algorithm, such as *k*-means or agglomerative hierarchical clustering. Heuristically selecting a suitable number of clusters, each cluster could then be visualised as its own separate Gatherplot. Instead of allowing the user to make ad-hoc selections, the system could simply offer explanations for each cluster in turn. This is similar to the approach taken by Bernard et al. (2012). However, this would introduce an additional layer of parameterised complexity. Namely, choosing a clustering algorithm, its hyperparameters, and a sensible number of clusters, must either be done heuristically, which may produce a poor clustering with no way for the user to select series across clusters, or explicitly by the user, which raises the expertise required to operate the system. This is another instance of how Gatherminer exemplifies design principle 2: abstract complex processes through heuristic automation.

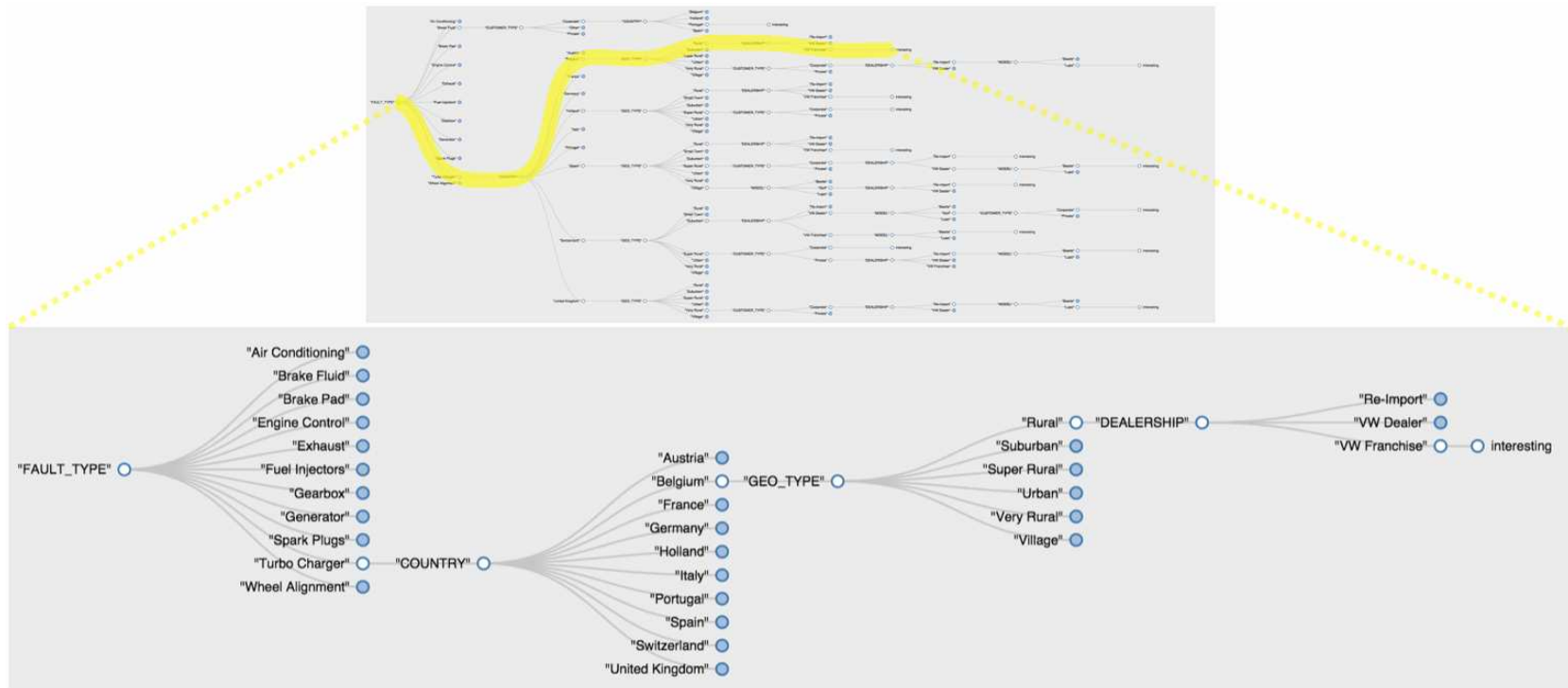


Figure 4.10: Above, the full ID3 tree generated from the same high-valued clusters as Figure 4.9, where “interesting” leaf nodes have been left uncollapsed. Explanations are read as a conjunction of nodes from root to leaf. As an example, one such path (thick yellow line) is magnified below: when $(\text{FAULT_TYPE}=\text{Turbo Charger}) \wedge (\text{COUNTRY}=\text{Belgium}) \wedge (\text{GEO_TYPE}=\text{Rural}) \wedge (\text{DEALERSHIP}=\text{VW Franchise})$, the time series is likely to be interesting (i.e., in the selection).

4.3.4 Alternative interactive representations for decision trees

The tree visualisation presented in previous sections is a straightforward mapping of the underlying ID3 output structure onto a node-link diagram in which hierarchy/depth is mapped onto the horizontal axis. Its advantage is that the topology of the tree is immediately apparent. However, it has limitations: notably, that it is cumbersome to read the rule which a node describes; that the analyst is not aided in their evaluation of nodes as providing analytic value; and that the visualisation does not scale, becoming cluttered with as few as 100 nodes onscreen. To provide an alternative design which better facilitates analytical modelling, it is necessary to first clarify its design objectives:

- **Rules:** allow rules to be easily read off. Paths through the tree from root to leaf are the primary analytic output of the tree, and it should be straightforward and frictionless to read the path from root to any node.
- **Explanation:** answer why, why not, how questions (recall the discussion of intelligibility (Lim et al., 2009) from previous chapters). In particular, ‘why/why not was this datum classified as interesting/not-interesting’, ‘how are predictions made?’, and ‘how is the tree built?’
- **Scenting (Pirolli and Card, 1999):** allow efficient, intentional tree navigation. The representation must assist analysts in deciding the order in which to consider nodes.
- **Assessment:** allow critical assessment of the learned model. In particular, this calls for design principle 4: support dialogue through metamodels. In the case of the ID3 algorithm, this reduces to assessing whether any given learnt ‘rule’ has a good balance between support/coverage (i.e., it provides explanations of a significant amount of the data) and discrimination (i.e., the rule discriminates well between classes; it has high information gain). This can be expressed in terms of the metamodels introduced in the previous chapter: a rule with poor discrimination/support has low confidence, and a rule with low coverage has poor command.
- **Scaling:** scale to trees of arbitrary sizes. An important strength of decision trees as an explanation mechanism is their ability to reduce the combinatorial explosion of attribute-value combinations to only those which are most valuable from an information-theoretic perspective, down to $\sim AV$ nodes from $\sim V^A$, where A is the number of attributes and V the number of values each attribute can have. Nonetheless, firstly: this is merely the *expected* behaviour of the tree, and a pathological dataset may cause the tree to increase to its worst-case size, and secondly: as the attribute-value space grows, the size of even well-learned trees grows accordingly.

The solution presented here is an interactive treemap (Shneiderman and Wattenberg, 2001), shown in Figures 4.11 and 4.12. The layout is squarified (Bruls et al., 2000) for its superior aesthetic properties, as well as the ability to provide easier hover/click targets than, for example, “slice and dice”. At any given time, the treemap represents a single node in the tree as being composed of its children, which are in turn composed of their grandchildren. That is, the treemap displays a node and up to the next two deeper levels of hierarchy. Children are represented as coloured rectangles, and grandchildren in turn are smaller rectangles nested within those. The area of each rectangle is proportional to the number of data points falling under that node. ‘Falling under a node’ means possessing the combination of attribute values represented by that node; data points which if passed through the tree for prediction would be routed through the node.

The colour of each rectangle is computed as follows. A unique colour is assigned to each class in the data. In Gatherminer, the ‘interesting’ class is assigned a particular shade of blue, and the ‘not-interesting’ class is assigned pure white. Finally, the colour of a rectangle is computed as the average colour of data points falling under that node. This visually represents the expected class of data falling under the node. The size and saturation colour of a rectangle corresponds roughly to the strength of the rule; this mapping exemplifies design principle 4: support dialogue with metamodels.

The rule corresponding to the focused node is shown in the grey bar above the treemap. Navigation in the treemap is as follows: clicking on a region in the treemap descends one level, moving the focus to the clicked child of the currently focused node. Clicking on the grey bar moves “up” one level; focus moves to the parent of the currently focused node.

Reinforcing data coverage through ‘isotype’ circles

The space of each rectangle is filled with circles which each represent a single data point falling under that node. The circles are coloured slightly darker than their backgrounds so as to always remain visible. Associating circle count to data coverage is a redundant mapping (rectangle area is already mapped to this quantity), but serves to ground that mapping in a countable number, which is important when moving through the tree, since the absolute size of a node changes as it goes from being a grandchild, to child, to (intermediate) root. This also follows design principle 1: begin the abstraction gradient at zero. Moreover, the circles help judge the support aspect of the assessment criterion, especially when the numbers are very small (<10). The circles may be regarded as a simplistic, generic isotype (Brinton, 1914; Neurath et al., 2010), despite the fact that classical isotypes are pictographic depictions of the real-world objects represented by the data, for two reasons. First, they are a generic representation of the data ‘point’, the abstract entity with which analysts operate, and second, they serve the same purpose, namely grounding in countable numbers, arranged to provide an intuitive representation of the total amount.

The radius of all circles in a node is set according to the position of the node in the tree. In particular the radii are set to be inversely proportional to the *height* of the node, defined as the length of the longest path between the node and any of the leaf nodes you can reach from that node. This is a distinct concept from the depth of the node, which is the distance between the root and that node. Nodes at the same depth may have very different heights. The idea behind this mapping is that the height of a node corresponds to the furthest that the drill-down exploration could proceed from that node. Thus, the leaves of a very high node are ‘further away’ than those of a low node. The mapping of circle size capitalises on the visual perception phenomenon of distant objects appearing smaller. Thus, the size of the circles indicate how far you could go down that node before hitting the leaves (representing complete rules).

These decisions combine to satisfy the design requirements as follows:

- Rules can be easily read from the grey bar at the top.
- Rules themselves provide why and why-not explanations, and ‘how’ explanations are supported by the user composing rules through iterative drill-down.
- The size and colour of rectangles, as well as the count and size of circles provide scenting for an order of consideration, as well as assist users’ spatial reasoning about navigating the tree.

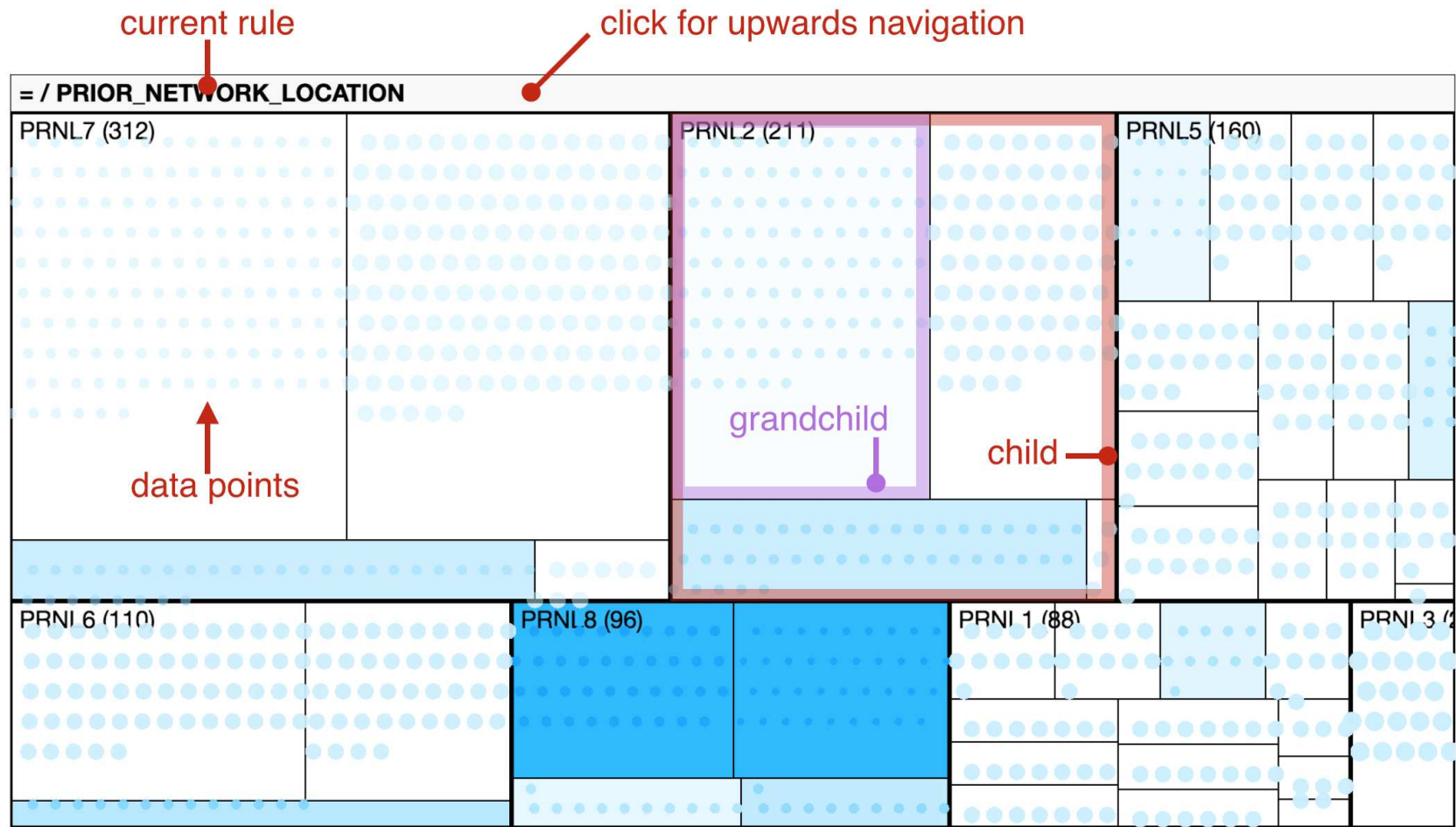


Figure 4.11: Treemap: top level.



Figure 4.12: Descending through successive levels.

- Support/coverage and inter-class discrimination are mapped to primary perceptual variables (size and colour), and reinforced by dot count, exemplifying design principles 1 (begin the abstraction gradient at zero) and 4 (support dialogue through metamodels). This allows users to evaluate the analytical merit of each rule. Note that this is the key step requiring human judgement. Rules can be heuristically evaluated, but no general set of heuristics can assess the analytic value of a particular rule as well as an analyst with extensive expertise in the domain of the data.
- As the same rectangular space is multiplexed to represent a subtree at any given time, this scales spatially to arbitrarily large trees. This does however sacrifice a general overview of the tree structure, which the node link diagram facilitates.

Many alternative tree representations exist. An excellent survey is given by Schulz et al. (2011). The treemap design is now further validated by comparison to some alternatives.

One alternative is simply to provide a textual list of rules. This perfectly satisfies our rule criterion. The rules provide why and why-not explanations. The order in which rules appear could guide navigation (e.g., sort rules in decreasing order of coverage), and a coloured background could indicate class distribution. If placed within a scrollable window, the list could be arbitrarily large, satisfying the scaling criterion. However, there are two issues with this representation. The first is that eliminating the hierarchy of the tree structure reduces the ability to provide ‘how’ explanations of the ID3 algorithm. The second is that as the list of rules grows, analytical exploration of the tree is mechanically biased by the ordering of rules. The interactive treemap has one clear mechanical bias which is explicit to the user: depth. That is, since it requires one click to descend into the next level of the tree, it takes less effort to inspect nodes of lesser depth. Nodes of lesser depth involve fewer attributes in their explanation and are consequently ‘simpler’, so prioritising the consideration of these nodes is justified by Occam’s razor (the parsimony principle). By contrast, the mechanical bias introduced by a list is the scroll order; it requires less effort to inspect nodes whose rules appear earlier in the list. Consequently, producing an ordering of rules based on some computation of support, discrimination, and tree depth has an unsurprisingly significant effect on which rules are considered by the analyst. However, any particular way of ordering rules cannot correspond to their analytic merit for every possible domain and dataset. Moreover, regardless of how nodes are mapped to this linear ordering, our parsimony principle is always violated, because nodes at the same tree-depth will necessarily require different amounts of effort to scroll to. This limitation extends to any such variation on a list of rules, including indented or collapsible tree lists.

This is not to say that we propose tree depth or the parsimony principle as the only or best heuristic for evaluating each rule. As already discussed, no general heuristic can capture the analytic merit of a particular rule as well as an analyst with domain expertise. The bias induced by the mechanical effort of exploration is unavoidable. However, it can be mitigated if the design facilitates the analyst’s awareness of this bias, which can be done if the mechanical effort required corresponds to a very obvious principle. Our design decision to favour tree depth, and depth only, illustrates that point.

Other alternatives are sunburst plots (Schulz, 2011) and icicle plots (Kruskal and Landwehr, 1983). These are closely related to the treemap, and have the favourable property that the mechanical exploration bias is directly mapped to tree depth. However, neither of these are space-filling layouts, which is important because the utility of basic perceptual cues such as relative size and colour is greater when the areas occupied by each datum are larger. Sunburst plots suffer from the further issues that the use of angle is not a good perceptual encoding of magnitude (Cleveland and McGill, 1984), and that the area of segments becomes incorrectly visually conflated with magnitude.

4.4 Comparative study

The first question of this dissertation is whether tools can be built for analytical modelling. Does Gatherminer address this? Trivially, the answer is ‘yes’, as Gatherminer has certainly been designed to occupy that space – it is clearly a tool which incorporates analytics and interactive machine learning. A more demanding question is whether it does so fruitfully. This can be expressed as the following sub-questions:

With respect to a current industry-standard analysis tool, does Gatherminer:

- result in more interesting patterns found?
- result in more correct explanations being found?
- result in features and explanations being found faster?
- improve users’ confidence in their analyses?

The second question of this dissertation is whether such tools can be made usable by non-expert end users. In order to understand this we must unpick once again the notion of ‘expertise’ as has been done in previous chapters, and address its components in turn. Gatherminer reduces *representational expertise* requirements by leveraging basic perceptual principles to provide an interface which improves upon classical statistical languages, that it reduces *process expertise* requirements in allowing users without advanced knowledge of clustering and data mining techniques to apply them correctly, and that it reduces *domain expertise* requirements by using well-chosen generic statistical techniques. In terms of our design principles, aspects of Gatherminer’s design begin the abstraction gradient at zero, abstract complex processes through heuristic automation, build expertise through iteration on multiple representations, and support dialogue through metamodels.

To verify the suitability of Gatherminer as a non-expert tool, it suffices to directly evaluate on users without such expertise. An affirmative answer to any of the four sub-questions presented above is consequently both a validation of Gatherminer’s role as a hybrid system as well as its reduction of expertise requirements. A user study was conducted to investigate the research questions above.

Participants

Six participants were recruited from analytics groups within BT Research and Technology, situated at Adastral Park in Ipswich, UK. Participants were experienced professional analysts who regularly study the BT network data using Tableau. Importantly, they were not statisticians or programmers. We chose Tableau⁴ as the visual analytics tool against which to make comparisons, as this was most representative of our participants’ typical workflows. A generic tool like Tableau is the only viable option in industry, since no tool tailored to this problem is available. Each participant had extensive prior experience of using Tableau, but no participant had any prior exposure to Gatherminer. The experiment was carried out in the participants’ place of work, in their standard office environments.

⁴<http://www.tableau.com/> (last accessed: April 29, 2018)

Experimental tasks

Each participant completed 5 matched pairs of tasks (10 tasks in total). Five of the tasks were conducted using Tableau, and the other five using Gatherminer; this allowed us to make within-subjects comparisons. The order of tasks was randomised between participants to account for order effects. For each task, participants were given a time series dataset of 500 time series, each of length 200. Each time series had 6 attributes: A, B, C, D, E, F . Each attribute had 6 values, $A = \{A1, A2, A3, A4, A5, A6\}$, $B = \{B1, \dots, B6\}$, and so on. For our experimental tasks, an “uninteresting” time series consisted of random samples from the uniform integer distribution between 1 and 100. An “interesting” time series contained a segment where the distribution is heavily weighted towards 1 or 100, that is, an upward or a downward spike. Each interesting feature had a unique corresponding causally-related attribute value (e.g., in one task, all series with $A = A2$ contain an upward spike). The dataset for task pair #1 was synthesised to have 2 interesting features. Task pairs #2 and #3 had 3 interesting features each, and task pairs #4 and #5 had 4 interesting features each.

While the design of our tool was informed by and aimed towards real-world data (as in our examples), for the purposes of the experimental task we deliberately chose to synthesise domain-independent data. This is because the reliance of analysts on their domain expertise is so strong that it acts as a confound and prevents meaningful comparisons of the intrinsic benefits of various visualisation systems. This will shortly be discussed in greater detail.

Using datasets generated in this manner, participants were requested to “find and explain as many interesting features” of the time series as they could. Additionally, participants were requested to rate their confidence about their performance after each task, using a 10-point scale in the format of the validated Computer Self-Efficacy inventory (Compeau and Higgins, 1995). Specifically, they rated themselves on a scale of 1-10 with respect to the following two questions: (1) “How confident are you that you found all the interesting features?”, and (2) “How confident are you that you found plausible explanations for the interesting features you found?” Participants were not made aware beforehand of the nature or number of interesting features in any task. Participants’ remarks were also recorded during the experiment; a discussion of these is reported shortly.

4.4.1 Experimental results

In general, data was paired, not normally distributed, and had equal sample sizes for all conditions, so comparisons were drawn using the Wilcoxon signed rank test (WSRT).

Task completeness

Participants found significantly more interesting features with Gatherminer than with Tableau (WSRT: $V = 171, p = 1.7 \cdot 10^{-4}$); the effect size is a median discovery of an additional 50% of features with Gatherminer. Similarly, with Gatherminer they found significantly more correct explanations for those features (WSRT: $V = 276, p = 2.2 \cdot 10^{-5}$); a median of an additional 66.7% correct explanations were discovered with Gatherminer. This is illustrated in Figure 4.13.

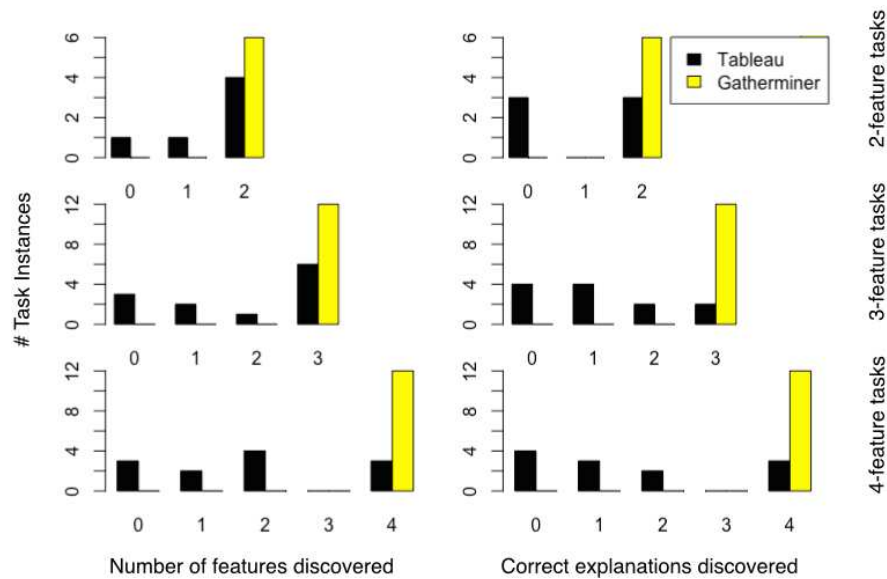


Figure 4.13: Comparative histograms of discovered feature and explanation counts when using Gatherminer and Tableau. Observe the wide spread of completeness with Tableau.

Discovery times

With Gatherminer, participants took significantly less time to discover features (WSRT: $V = 1176, p = 1.7 \cdot 10^{-9}$); the effect size is a median improvement of 110.5s using Gatherminer. They also took significantly less time to discover correct explanations (WSRT: $V = 654, p = 4.8 \cdot 10^{-7}$); a median improvement of 181.5s. This improvement is not altogether surprising, since Tableau is a general-purpose tool, meaning that there is always a certain amount of time invested to create a visualisation in Tableau in the first instance.

Confidence

Post-task, participants were significantly more confident that they had indeed discovered all major interesting features using Gatherminer than using Tableau (WSRT: $V = 465, p = 1.7 \cdot 10^{-6}$); the effect size is a median increase of 6.5. Similarly, they were more confident that they had discovered plausible explanations for all of the discovered features while using Gatherminer (WSRT: $V = 465, p = 1.6 \cdot 10^{-6}$); a strong median increase of 8. This is illustrated in Figure 4.14.

4.5 Discussion

Analysis strategies in Tableau

Gatherminer is a specialised tool that emphasises certain types of analysis over others. Tableau, on the other hand, is a much more general-purpose analysis tool, facilitating many strategies for solving these tasks. Consequently, this may appear to be an unfair or straw man comparison. It is not, because the participants were *expert* users of Tableau, experienced in performing precisely this type of statistical analysis using Tableau. Some observations regarding the analysis strategies our expert participants in Tableau are now reported, and how Gatherminer’s design improves upon these is discussed.

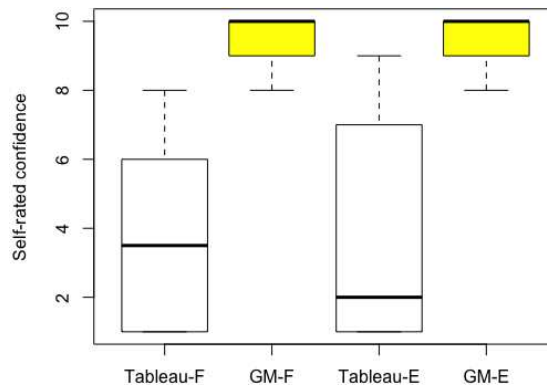


Figure 4.14: Boxplots of self-reported confidence scores for feature discovery (F) and explanation discovery (E), comparing Gatherminer (GM) vs Tableau.

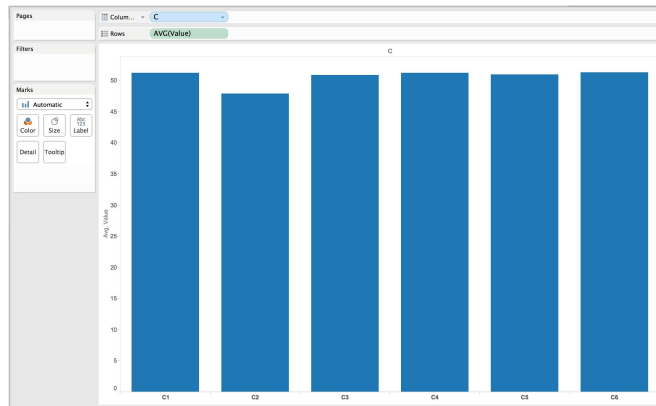
Successful strategies in Tableau

Successful strategies relied on finding levels of aggregation that generate visualisations with a manageable level of complexity, whilst simultaneously revealing interesting features. These can be viewed in Figure 4.15. However, these strategies still fell back onto manually iterating over each attribute in turn, and this resulted in participants feeling less confident about their analysis. One such strategy was to observe aggregate line charts of each individual attribute-value pairing (e.g., one line chart summing all series where $A = A2$). Here, any attribute-value that caused spikes or dips was clearly reflected. Since our tasks consisted only of 6 attributes, each with 6 values, and each feature only involved one attribute at a time, it was possible to apply these strategies effectively. However, in practice, with many more attributes and values, these strategies quickly become intractable. In contrast, since Gatherminer shows the completely disaggregated time series, it is possible for the user to view interesting features across all values of all attributes simultaneously. The analysis of interesting features drives the discovery of correlated attributes, not vice versa.

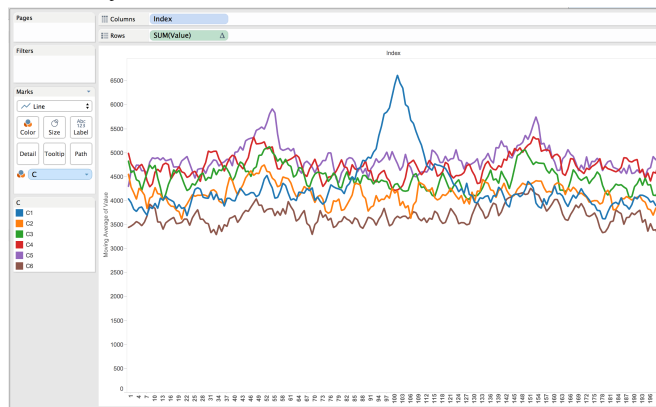
Unsuccessful strategies in Tableau

Unsuccessful strategies generally stemmed either from viewing data in completely disaggregated form (e.g., one line chart for each of the 500 series), which lead to unmanageable complexity in the visualisation, or aggregating the data too much (e.g., one line chart that summed over all 500 series), which resulted in features going completely undetected. A few of these strategies are shown in Figure 4.16. In Gatherminer, the data is also completely disaggregated, but the compactness of the colour-mapped matrix display, combined with automated reordering, makes the complexity of the visualisation manageable.

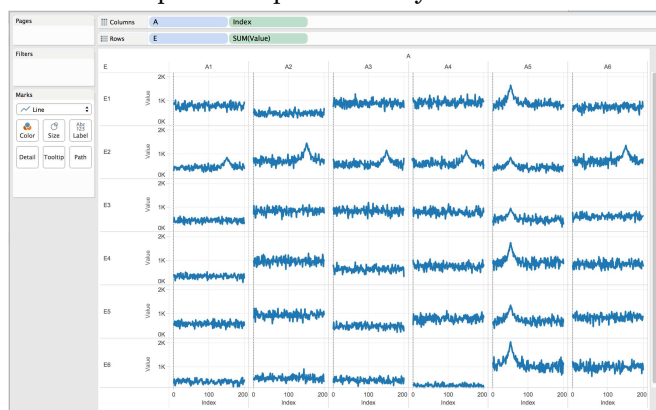
Another frequent issue was the discovery of false correlations. A common strategy was to take a few examples of time series with interesting features and inspect their attributes. If these series had more than one attribute value in common, the analysts were likely to conclude that the conjunction of those values together produced the effect, whereas in reality it may have just been one of the attributes, and the spurious correlation of the other attribute was simply a consequence of the small sample size. In Gatherminer, since series with interesting features are grouped together, it is trivial for the analyst to select large sets of series with shared behaviour to inspect the overall properties of their attributes.



(a) This strategy involved comparing bar charts of each attribute-value pairing, aggregated over the entire span of time. Since the interesting features in our time series consisted of unusual spikes/troughs, this usually reflected in a higher/lower overall sum or average for those series – easily spotted in an unusually tall or short bar.

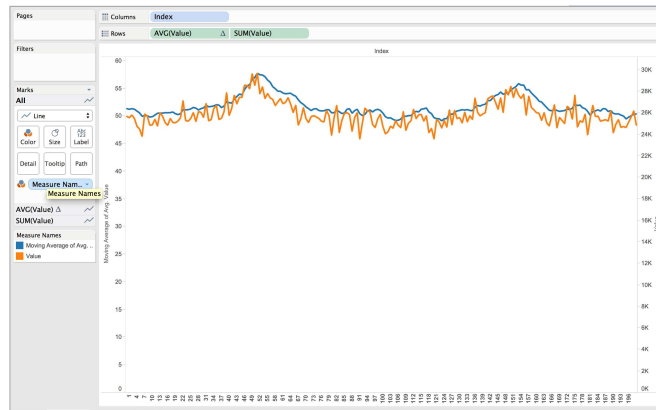


(b) This strategy involved comparing aggregate line charts of each attribute-value pairing. Here, any attribute-value that caused spikes or dips was clearly reflected.

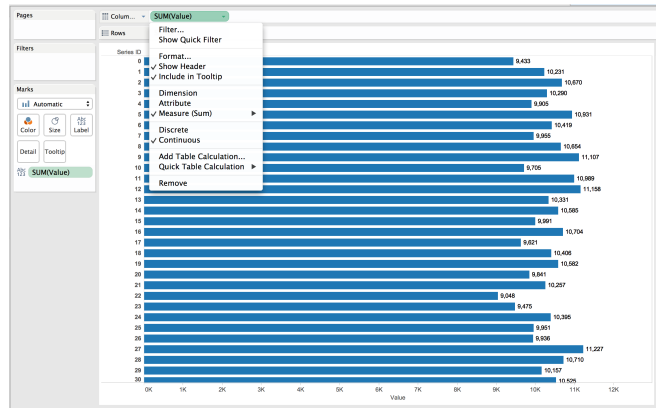


(c) This interesting strategy also compared aggregate line charts of each attribute-value pairing. Here, by creating a 2D matrix of small multiples, the analyst was able to investigate the interaction of any two attributes.

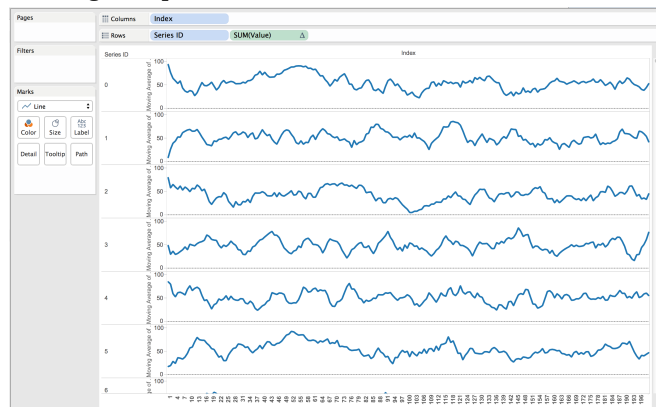
Figure 4.15: Three successful strategies in Tableau.



(a) This strategy involved inspecting a completely aggregated line graph. In this dataset, we prepared a number of time series that had spikes at about 1/3 and 2/3 the duration of the series, which are clearly visible in the aggregate chart. However, there are also a number of series which have an upward spike in the halfway mark, and an equal number which have an equal and opposite downward spike at the same position. The two cancel each other out and become invisible in the aggregate line graph, and so the analyst never discovers them.



(b) This strategy, similar to the first successful strategy, uses summary bar graphs to represent the time series. However, since the series are completely disaggregated (one bar is generated per series), it is impossible to seek out global patterns.



(c) This strategy involved scanning through the entire list of time series, represented as line graphs, and manually noting down the attributes of any which appeared interesting. Needless to say, this is extremely ineffective and led to several false correlations being “discovered”.

Figure 4.16: Three unsuccessful strategies using Tableau.

Confidence

It is important for analytical tools to enable analysts to have confidence in their analyses. In this regard, one major strength of colour-mapped matrices that benefits tools such as Gatherminer and Line Graph Explorer is that they provide an exhaustive overview of the data; this satisfies analysts' desire to "leave no stone unturned." In particular, even the analysts who developed successful strategies in Tableau recognised that the manual nature of their strategy was not scalable, remarking: *"You've got too many dimensions to visualise simultaneously"*; *"Maybe I should just focus on one attribute for a start"*; *"I'm going to scroll through this list, and when I see one..."*; *"I feel like I'm missing a lot if I do it manually"*. Remarks regarding the confidence of their analysis in Gatherminer include: *"I can explore all of it. I don't have to drill down."*; *"Am I confident I have discovered all the features? Yes, of course, I have seen it."*

Value of the gathering process

Gatherminer strongly encourages partially-automated analysis. On almost every occasion, while using Gatherminer, participants first deployed the "Gather" function before doing anything else. While using Tableau, participants often mentioned dissatisfaction with the limitations of the (nonetheless rather sophisticated) built-in sorting functionality: *"If I can some way get a cluster"*; *"What I want is interesting features grouped together"*; *"I want to see groups of lines that are behaving [similarly] because then I can see which of these variables is impacting the series."* Remarks from our participants regarding Gatherminer's reordering function include: *"It's nice to have this hybrid approach where you get [the reordering] automatically and then the analyst can also scan it manually to see what is going on"*; *"At first the [colour-mapped matrix] view itself is helpful because you understand that there is something going on. The clustering then makes it very evident."*

Role of domain expertise

Our tasks deliberately used meaningless codes for attributes and values in order to separate the utility of our tool from the domain expertise of the participant. Had we used network fault data, then the relative experience of the participant in that domain could potentially have impacted their ability to effectively analyse the data, independently of the analytical tool. Domain expertise provides a variety of prior expectations regarding what types of features might be present (e.g., linear trends, peaks, troughs, periodic functions), and what attributes might be of explanatory value – providing an efficient order of consideration for brute force attribute value checking. Note that these prior expectations may not necessarily be beneficial, as they may lead to the discovery of spurious correlations, or overlooking attributes not expected to be related.

Our hypothesis about the latent confounding power of domain expertise was further substantiated by comments made by our expert participants whilst analysing the data in Tableau. One participant said: *"If this was data I knew about, then I'd have some idea of where to start. Here, I'm lost."* Other remarks include: *"Part of that [difficulty experienced with experimental tasks] is that I have no sense of the features"*; *"Doing data analysis when you have no idea of the data is quite unusual"*; *"Given that I have no idea of the attributes, I have to ignore them."*

Validity

A rationale has been provided for the use of synthesised data for the preliminary experimental validation of the Gatherminer analysis interface. Domain expertise plays a significant role in the analyst's heuristic approach to discovering explanations of interesting features. Thus, controlled usability experiments designed with real-world data in order to preserve external validity may have an adverse effect on internal validity; the participants' use of domain expertise may confound any meaningful comparison between visualisation systems.

The requirements of a controlled experiment have precluded the use of real-world data. Our experiment has shown that Gatherminer significantly improves the quality of analyses conducted in the absence of domain expertise. This could be interpreted as Gatherminer effectively reducing the domain expertise barrier for this particular type of task. That is, it improves the ability of analysts without deep expertise in the data to perform these tasks. This provides an affirmative answer to the second primary research question of this dissertation, namely, whether such tools can be built for non-expert end users. Here, 'non-expert' refers to a lack of domain expertise as well as statistical expertise.

This experiment has not answered two complementary questions: does Gatherminer reduce expertise barriers to such an extent, that analysts without domain expertise using Gatherminer become as effective as those with domain expertise using Tableau? What about when both use Gatherminer? Separate controlled studies would be required to investigate these questions. These questions are, however, secondary to the question which our experiment was designed to answer, namely, whether Gatherminer *intrinsically* improves the analytics process, independent of domain expertise confounding.

A more important complementary question is whether Gatherminer significantly improves the quality of analyses compared to Tableau where domain expertise *is* present. This question can be placed on equal footing with the question investigated in our study, as it is most representative of the real requirements of analysts, that is, has the highest external validity. For the reasons previously explained, it would be challenging to design a properly controlled study because the effect of domain expertise is unpredictable. However, a longitudinal industrial deployment of the tool would supply the necessary external validation through qualitative interviews and contextual inquiry. Anecdotal comments from analysts at BT as well as other industrial analysts suggest that even when domain expertise is present, Gatherminer still provides significant benefit.

4.6 Conclusions

An important class of analytical tasks related to the study of time series has been outlined. In these tasks the shape of interesting patterns is not known beforehand, and the space of explanatory attributes is large. Fault data from a national telecommunications network is used as a motivating example. In many industries, these tasks pose huge challenges which cannot easily be solved with commercial analytics software. Previous work has provided compact representations of large time series data in the form of colour-mapped matrix displays; how rearrangement of the visualisation can expose interesting features; and how visualisations and machine learning can be combined for intelligent querying.

Gatherminer is a tool for the analysis of time series data that takes these established ideas and uses them in a novel combination to tackle the class of analytical tasks we have outlined. A user study demonstrated that for these tasks, Gatherminer results in significantly faster and more complete analyses of time series datasets. Moreover, Gatherminer significantly improves analysts' confidence in the quality of their analyses. Field observations of expert strategies for analysing time series data show how our approach overcomes common difficulties. Finally, the benefits of mixed-initiative interaction have been discussed, and the importance of carefully considering the effects of domain expertise when designing evaluations of new analysis tools have been illustrated.

Besides domain expertise, Gatherminer also greatly reduces the process expertise required to apply clustering to time series. Concretely, it is often the case that time series analysts are experts in the domain of their data, but are not highly trained in statistics or machine learning. Consequently, sophisticated unsupervised 'bottom-up' hypothesis generation techniques are out of reach for such analysts. Even excellently designed software libraries for clustering suffer from this high statistical expertise requirement. Gatherminer's default reordering methodology is founded on basic perceptual principles, and the only user-tunable parameter is in optionally overriding the default distance metric.

Concretely, this chapter contributes the design rationale for a reduced-expertise hybrid analytics/model-building tool in a particular time series analysis setting, including

- a colour-mapped matrix which is reordered to reveal interesting patterns, with a rationale for default colour mappings and algorithms for reordering,
- a description of how to use selection-as-annotation to allow the application of arbitrary supervised learning algorithms for analytical modelling,
- a case study of the interactive visualisation of one such algorithm, ID3 decision tree learning, following established principles of interactive machine learning, and
- an evaluation demonstrating the fitness of the design for its intended purpose.

Gatherminer applies four design principles for analytical modelling:

- It begins the abstraction gradient at zero in at least two instances: first, the core data representation is a lossless visual encoding of each datum, and second, the tree representation represents data points with isotypes.
- It abstracts complex process by applying heuristics. These heuristics are derived from perceptual principles. Concretely, it applies a reorderable matrix using a distance metric having visual semantics, and applies a default normalisation scheme for improving visual clarity. This enables Gatherminer to be a useful clustering tool without exposing any parameters of the underlying clustering mechanism.
- It builds expertise through multiple representations. It builds expertise of the colour-mapped matrix by allowing users to re-cast rows of the matrix as line charts on demand, and it builds expertise of the explanation trees through at least two representations of the tree, and a simplified bar chart representation.
- It supports dialogue through metamodels. It reinforces data coverage and class discrimination of explanatory rules through the primary perceptual cues of size and colour in the treemap.

BRAINCEL

This chapter presents the design and evaluation of BrainCel, a tool for building and applying machine learning models such as k -nearest neighbours, neural networks, decision trees, etc. to relational data in spreadsheets. BrainCel applies the design principles for analytical modelling, and also demonstrates selection-as-annotation, here showing that desiderata for a supervised learning algorithm can be obtained from user selections. BrainCel features multiple coordinated views of the model being built, explaining the model's current confidence with respect to its predictions, as well as its coverage of the input domain, thus helping the user to correct the model and select new training examples.

An experiment demonstrates that selection-as-annotation can be understood and successfully applied by users having no professional training in statistics or computing, and that the experience of interacting with the system leads them to acquire some understanding of the concepts underlying exploratory statistical modelling. Further, an exploratory study is reported that investigates users' learning barriers and information needs while building models using BrainCel. The study shows that the novel approach of BrainCel exhibits properties observed in similar previous work on interactive machine learning, but within the general purpose spreadsheet paradigm.

This chapter presents research described in the following papers:

- **Interactive visual machine learning in spreadsheets.** Advait Sarkar, Mateja Jamnik, Alan F. Blackwell, Martin Spott. *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (pp. 159–163).
- **Spreadsheet interfaces for usable machine learning.** Advait Sarkar. *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (pp. 283–284).
- **Teach and Try: A simple interaction technique for exploratory data modelling by end users.** Advait Sarkar, Alan F. Blackwell, Mateja Jamnik, Martin Spott. *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (pp. 53–56).

5.1 Introduction

In previous chapters, we have motivated the questions of whether tools can be built for analytical modelling, and whether they can be made useful for non-experts. The example of Gatherminer has shown how interactive machine learning design principles can enhance analytical activity. In this chapter, BrainCel, a model-building tool where the design facilitates visual analytics, is described. Like Gatherminer, BrainCel reduces requirements for many types of representational, process, and domain expertise. BrainCel enables non-expert end-users to build and apply machine learning models on their own datasets in spreadsheets. In doing so, users become more informed about both their dataset as well as about statistical inference more generally.

As stated in the introduction to this dissertation, the ability to build and apply powerful statistical models, such as neural networks, linear regression, and decision trees, is currently only available to users with expertise in both programming as well as statistics and machine learning. Clearly this is an issue; the scarcity of data analytics tools aimed towards non-experts in computing and statistics excludes large groups of end-users. The claim that “everyone should have access to data analytics,” might appear excessively idealistic, as of course, analytics is a complex and skilled activity. The production of a data scientist requires years of education to acquire foundational knowledge in mathematics, computation, and statistical inference, and then further years of experience in the domain of the data they analyse — no amount of clever interface design can overcome these requirements. However, it is not the ambition of this work to replace the entire statistical profession. Rather, it aims, through well-designed tools, to enable a wider range of end-users to address data-related issues which they previously could not.

The suitability of the spreadsheet

The solution proposed in this chapter is to take advantage of the expressive power of spreadsheets to bring these techniques to non-experts. Since end-users are likely to be readily familiar with the manipulation of data in spreadsheets, integrating statistical modelling into the spreadsheet environment appears to be a promising avenue. The problem can be framed as one of end-user programming (EUP) (Ko et al., 2011). Much work has been done on improving the quality of EUP in spreadsheets, with mature debugging approaches such as What-You-See-Is-What-You-Test (Rothermel et al., 1998).

End-users are being exposed to machine learning and statistical inference in a rapidly increasing range of applications, such as email filtering and recommender systems. In these applications, the user implicitly manipulates probability distributions and statistical models by providing examples rather than explicitly programming instructions. While easily adopted by end-users, these applications typically only allow the user to build models specific to limited data domains. The spreadsheet is a domain-independent tool which has allowed us to design a *general-purpose* interface for interactive machine learning. Moreover, the spreadsheet satisfies our first design principle, which is to begin the abstraction gradient at zero, because the data being manipulated is always directly represented.

Prior to this work, close research attention had not been paid to bringing interactive machine learning to the spreadsheet, despite it being arguably the most ubiquitous and important EUP environment. Recently, however, the computing industry has taken a keen interest in augmenting spreadsheets with statistical inference. The incorporation of a ‘flash fill’ feature into Microsoft Excel (Gulwani, 2011) has shown that programming-by-example, albeit for simple string manipulation functions only, is recognised as sufficiently useful for it to be promoted as a headline, consumer-facing feature of the market-leading spreadsheet product. Flash fill operates as a ‘black box’; interaction is limited exclusively to typing into the spreadsheet, and the end-user is not exposed to the model or its parameters. The view of IML as dialogue suggests that this interaction could be extended to make the learnt model more interpretable. The restricted synthesis vocabulary of string manipulation functions also means that flash fill’s modelling capabilities are limited in comparison to those offered by general machine learning algorithms. The interface is therefore inapplicable for complex modelling tasks, nor was it intended for such tasks.

Applications with more explicit support for statistical modelling are available, such as Oracle BI (Campos et al., 2005). Subsequent to our work with Teach and Try (Sarkar et al., 2014b), solutions from several vendors have emerged, including Microsoft’s Azure ML¹ and Google’s ‘smart autofill’ plugin for spreadsheets². However, these are aimed towards professionals with some formal understanding of the concepts underlying statistical modelling, and are not sensitive to the needs of non-expert users. Thus, while they are more accessible than the traditional, programmatic tools for data scientists, such as WEKA and R, they still fall short of the intelligibility and usability requirements of our target population, namely, end-users who do not have formal training in statistics or computing.

5.2 Selection as annotation

In BrainCel, users select a range of cells to mark that those values are correct and can be used as training data. The marked cells are used to train a statistical model. This is similar to the selection mechanism used in Gatherminer, where selected rows are marked as ‘interesting’, which provides training label information for the classifier.

Once the training set has been specified in this manner, the user can select any other range of cells, potentially overlapping with the training set, in order to apply the model to those cells. If cells in the new selection are empty, they are filled using the model prediction. Otherwise, they are coloured according to their deviation from the model prediction.

Teach and Try prototype

To evaluate this particular implementation of selection-as-annotation, a simple prototype was built, named ‘Teach and Try’ after the only two buttons on the interface. The sequence of operations with which a user may train a model and use it to estimate missing data is illustrated in Figure 5.1. The user first makes a selection of a number of rows, and then clicks on the ‘Teach’ button. At this point, the selected rows are added to a training dataset. The cells are visually marked as ‘taught’ by colouring the text and background green, and placing a green check mark icon in the cell.

¹<https://azure.microsoft.com/en-us/services/machine-learning/> (last accessed: April 29, 2018)

²<http://bit.ly/google-smart-autofill> (last accessed: April 29, 2018)

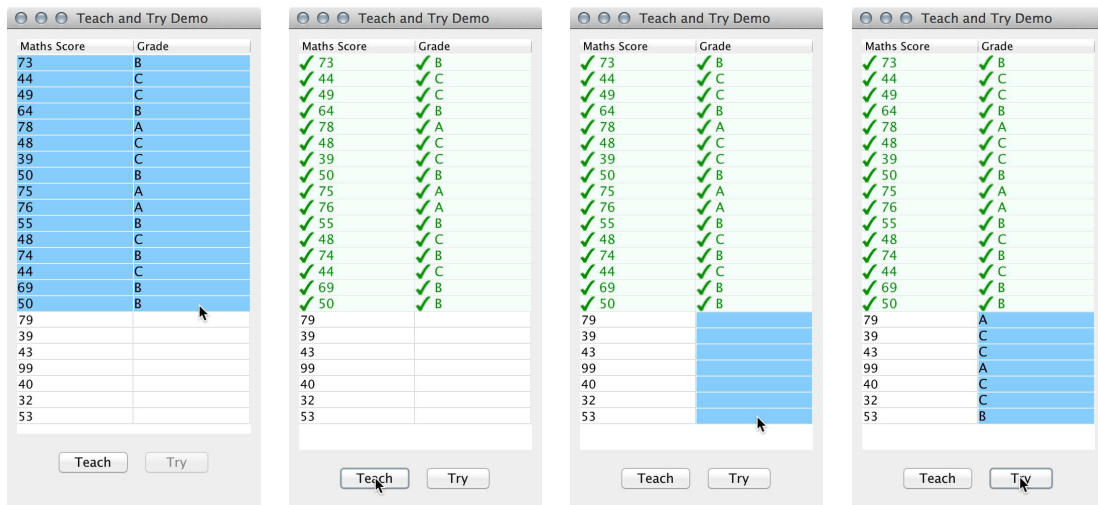


Figure 5.1: The “filling” capability of our initial prototype, depicted as a sequence of actions from left to right. This resembles an actual scenario presented to the participants of our study, in which they were asked to imagine themselves as a maths teacher grading students based on their score on a recent exam. Each row is a student. The first column is their score, and the second column is their grade. A score of 75 or above gets an A, 50 to 74 gets B, and 49 or below gets C. Some of the Grade column is pre-populated, and the participants were asked to use the software to “quickly grade the remaining students”. The model being implicitly built in these figures is a decision tree classifier.

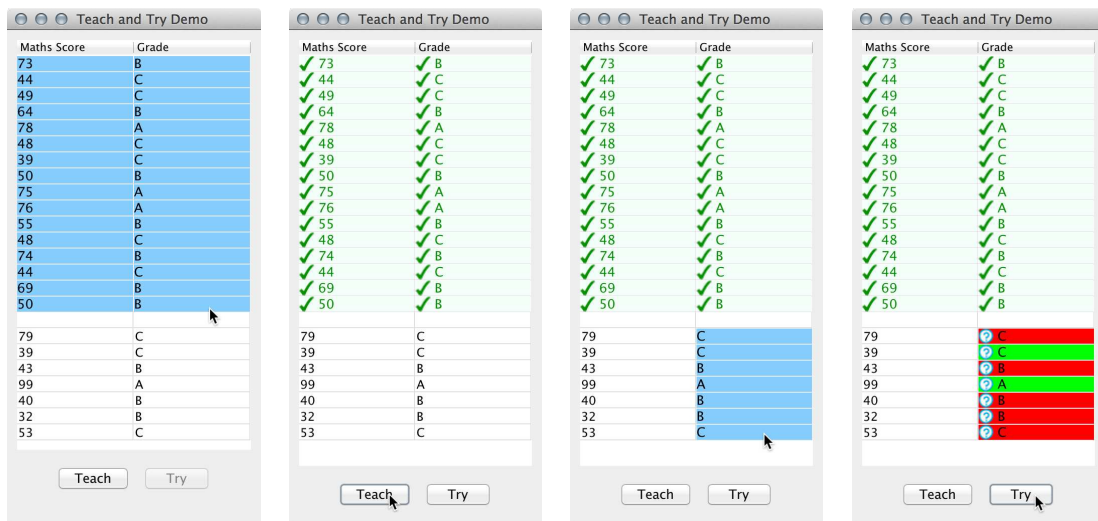


Figure 5.2: The “evaluation” capability of our initial prototype, depicted as a sequence of actions from left to right. This resembles an actual scenario presented to the participants of our study. They were asked to imagine themselves as a maths teacher assessing the competency of a colleague, who had graded the latter section of the students. The first section of the Grade column is pre-populated with “correct” data as per the marking scheme from Fig. 5.1, and the participants were asked to use the software to “check whether the new teacher understands how to grade papers”. Here, the machine learning task is that of classification, so there are only two colours assigned to the cells; green for “correct” and red for “incorrect”. In other scenarios, where the target variable was continuous, cells were coloured according to their percentage error on a linear red-green scale.

Next, the user selects cells from a single column and clicks on the ‘Try’ button, as in the two rightmost panels of Figure 5.1. When ‘Try’ is clicked, the software trains a model on the rows of data previously taught, and applies the model to the rows containing the current selection. The variable in the selected column is interpreted as the *target* or *dependent* variable for the statistical model, and the variables in the unselected columns are interpreted as the *feature vector* or *independent* variables. This interpretation of cell selection bounds allows the user to build and apply a suitable model with a single interaction. This is contingent on the data in the spreadsheet being laid out in a well-defined relational schema. If the selected cells are empty, then the model’s predictions are used to populate the cell contents. If the selected cells contain values, as in Figure 5.2, these values are not altered. Instead, they are coloured on a red-green scale to reflect the deviation of those values from the expectations of the model, and a question mark icon is added.

The initial prototype in Figures 5.1 and 5.2 is implemented in Java using the Swing UI library. The Java API provided by the WEKA Data Mining Software (Hall et al., 2009) is used to implement the algorithms for statistical inference. In the first instance, the prototype supported the use of the standard WEKA implementations of the multilayer perceptron, C4.5 decision tree, support vector machine, and simple linear regression, with the model chosen through a command line flag when launching the application.

The cell annotation aspect is related to WYSIWYT (Rothermel et al., 1998), in which users of a spreadsheet test it by marking ‘correct’ values in individual cells, allowing the system to synthesise boundary conditions. WYSIWYT allowed users without spreadsheet programming knowledge to debug their data. Similarly, Teach and Try allows users without statistical knowledge to build and apply statistical models.

The filling behaviour may appear similar to the string processing algorithm for spreadsheets due to Gulwani (2011); however, the capabilities of the underlying systems are different. While Gulwani’s system synthesises a specific class of string manipulation functions, our interaction technique allows for the application of many kinds of statistical models and inference algorithms. An appropriate model can often be inferred from heuristic characterisation of the selected regions. For instance, if the ‘Try’ selection contains categorical data, a classifier is likely to be appropriate. A more general solution would be to explore the space of models with the intent of minimising training error.

The Oracle Spreadsheet add-in for Predictive Analytics (SPA) (Campos et al., 2005) provides a spreadsheet-based interface for estimating the explanatory power of one variable with respect to another, and for performing SVM-based classification/regression. However, the output of the system operations is displayed in a separate spreadsheet to the original data, which reduces the directness of its manipulation. Furthermore, operations are triggered through a graphical wizard where the dependent variable and feature vector are manually specified, whereas Teach and Try exploits the information present in the user’s selection bounds to achieve the same effect. Finally, being targeted towards Business Intelligence (BI) professionals, it exposes statistical concepts that require domain knowledge to be interpreted, and thus it is unusable by those without such knowledge.

5.2.1 Experimental evaluation of selection-as-annotation

An experiment was conducted to evaluate the usability of this particular implementation of selection-as-annotation for demarcating training and testing data in spreadsheet. The prototype was demonstrated to non-expert end-users, who were then asked to then carry out tasks of a similar nature by themselves, in order to see whether they are able to generalise their conceptual understanding and develop the competence to apply it without further assistance. They were also interviewed to determine whether they understood the conceptual basis of what they had seen without explicit explanation.

Participants

Twenty-one participants were recruited, largely administrative staff, from the University of Cambridge Computer Laboratory and BT Research. Of these, 8 participants had prior exposure to statistical concepts. The remaining 13 had no prior experience with either machine learning or statistical modelling, but had basic familiarity with spreadsheets. This categorisation was enforced through a post-experiment interview about users' prior knowledge. When a participant indicated any awareness/prior exposure to statistical/machine learning concepts, for instance mentioning a model such as 'decision tree', 'neural network', or domain terminology such as 'regression', 'classification', 'least squares', 'hypothesis testing', 'training set' etc., they were excluded from the 'inexperienced' category.

Tasks

Seven brief tasks with corresponding datasets were created. Each task presented the user with a hypothetical scenario in which they used the software to perform a simple statistical procedure, without any explicit acknowledgement that they were doing so. Of the seven tasks, three were so-called 'filling' tasks. These required the user to use existing data to fill in missing information. A further three were 'evaluation' tasks, which required the user to use known high-quality data to assess or evaluate the quality or correctness of certain other data. A final task combined both filling and evaluation into one spreadsheet.

Procedure

The experimenter demonstrated the software for one of the evaluation and filling tasks each. The user was then asked to complete the remaining tasks, and was asked after each to explain what they had done. Task order was counterbalanced across participants. User responses were recorded, transcribed and analysed. For each user we recorded the time taken to complete the task, and observed unprompted references to statistical concepts.

5.2.2 Selection-as-annotation experiment results

Task durations

No significant difference was observed in task durations between the inexperienced and the experienced participants, suggesting that the software is usable by users with knowledge of machine learning or statistics as well as users without any prior exposure to such concepts.³ It was observed anecdotally that users found the 'evaluation' tasks more difficult than the 'filling' tasks, but this did not have a significant effect on task duration. Furthermore, the order of the task groups (whether the participant did the 'filling' or 'evaluation' task first) did not have a significant effect on task duration.

³Noting nonetheless that the absence of statistical significance is not evidence for the null hypothesis.

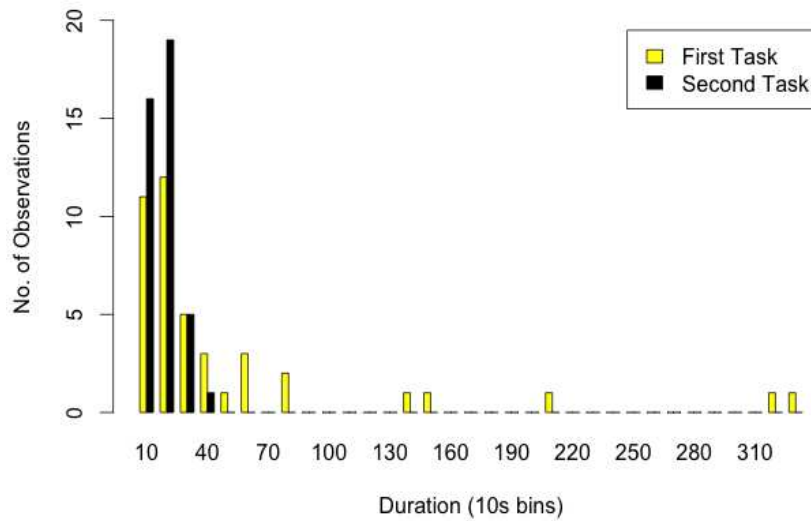


Figure 5.3: Frequency distribution of task duration, broken down by task order. Observe the decreased spread in the second task.

A significant effect of repetition on task duration was observed within task type. Specifically, the second task of any type (‘filling’ or ‘evaluation’) took *less* time than the first task of the same type. The task durations were not normally distributed, so the Wilcoxon rank sum test was applied to yield a highly significant location shift ($p < 0.004$). The size of the effect is a median improvement of 2.9s from the first task to the second (the mean improvement is 33.6s, but as the data is highly skewed this measure is biased by outliers). A plot of the task durations is shown in Figure 5.3. This can be interpreted as evidence of the *learnability* of the interaction mechanism; the experience of interacting with the software leads the user to quickly acquire the competence to apply it in a new context.

Conveyance of statistical concepts

From the post-experiment interview, it was clear that the participants had developed a nontrivial understanding of how the software works. Upon completion of the tasks, the participants were asked two questions in order to establish what kind of understanding of the system the user had acquired; specifically, these were (1) “How might the computer be doing this?”, and (2) “Why might the computer make mistakes?”

Because of the potential bias from previous experience, this section deals exclusively with the utterances of the participants who were classified as *inexperienced* with respect to machine learning and statistical concepts. Note that the following observations are presented not to make strong claims about the taxonomy of users’ beliefs, but rather to demonstrate that users gained some appreciation of the machine learning paradigm, namely, that of building a model using trusted data and applying it for inference. Participant answers to “how might the computer be doing this?” are now presented.

1. *Mathematical*: On 3 occasions, participants used mathematical terminology to describe their understanding of the software, guessing that it was “solving a system of linear equations”, “finding some sort of correlation”, or “plotting a graph”.
2. *Technical / software constructs*: On 3 occasions participants explained the software’s behaviour in terms of technology they were familiar with; in particular, these participants thought that the computer might be constructing complicated spreadsheet formulae, SQL queries or conditional formatting rules.
3. *Case-based reasoning*: On 3 occasions participants informally described the case-based reasoning, or nearest-neighbour prediction algorithms (“If a statement has been shown to be true in the past, then in the future, this statement must also be true” / “It checks to see if there is a precedent”). This suggests that nearest-neighbour matching or case-based reasoning are relatively intelligible algorithms.
4. *Non-technical*: On 6 occasions, participants described the software’s behaviour in abstract, non-technical terms, saying that it “spotted patterns”, “makes different connections between the numbers”, “deduces rules”, “makes assumptions about how things should look and extrapolates”, or “accumulates experience”.

When asked why the computer might make mistakes, participants provided multiple explanations with familiar implications in statistics and machine learning:

1. *Insufficient examples* (7 cases): “There’s not enough information available [...] to exactly predict the pattern” / “What you taught it didn’t cover it” / “The more data you have, the better will be the outcome”.
2. *Noisy training data* (7 cases): “What you taught was wrong” / “It could be getting mixed messages from the data” / “There is a contradiction in the data”.
3. *Insufficient discriminatory power* (between statistical classes) (4 cases): “Maybe the [data] overlap and they’re quite ambiguous”.
4. *Outliers* (3 cases): “It might be exceptional data”.
5. *Incorrect model* (3 cases): “The method isn’t yielding the correct answer”.
6. *Insufficient dimensions* (3 cases): “There might be other factors besides those listed in the data that influence the outcome”.

Usability issues

Some participants initially misunderstood how the selection bounds for the “Try” action were being interpreted. The most common error (3 cases) was to select *all* the cells in the target rows, as they had done for the “Teach” action. The next most common error (2 cases) was to select all cells in the target row *except* the cells in the target column, to instruct the computer to “Try to use *this* data...”. This suggests that while the “Teach” selection is immediately intuitive, a better mode for the “Try” selection might be desirable. One possible alternative is to invert the order in which the actions are taken, so that the “Teach” and “Try” buttons are pressed *before* making the selection. Here, the cursor would allow the user to “paint” regions of the spreadsheet as training or test data, in much the same way as regions were marked “interesting” or “uninteresting” in Gatherminer.

Some participants found the “Teach” and “Try” labels confusing for another reason: “Teach” is an action taken by the user, whereas “Try” is an action taken by the computer. This semantic irregularity caused some initial difficulty which was overcome once the participant had completed the first task; however, this suggested alternative labels such as “Train” and “Test”, or “Learn” and “Apply”. As discussed in Chapter 3, a potential reason for the success of this prototype at generating an understanding of statistical procedures in non-experts is the deliberate selection of the word “Try” as opposed to “Fill” or “Apply model”; it implies fallibility and evokes empathy, in line with the view of IML as dialogue.

Consequently, the labels were changed to “Learn” and “Guess”. Both clearly denote actions taken by the computer, whilst retaining the provisionality of “Teach” and “Try”. Additionally, the word “Guess” alleviates uncertainty around selection modality; it is clear to users that only the target cells are to be selected, as those are the values to be “guessed.”

5.3 Supporting dialogue and critical model evaluation

To recapitulate, this dissertation views interactive machine learning as *dialogue*, where the system must provide decision support to assist the user in tackling an ill-defined problem. This perspective is an evolution of previous work in interactive machine learning and end-user programming. The Whyline (Ko and Myers, 2004) introduced the idea of asking *why* and *why not* questions about program execution. Lim and Dey (2009) expanded upon this, categorising various *intelligibility types*; types of information which an intelligent system or program might give its user or programmer, including *how* a model makes a certain prediction. Recently, these frameworks have applied to end-user debugging of naïve Bayes classifiers in the domain of email categorisation (Kulesza et al., 2011).

The previous sections describe a simple interaction mechanism that allows end-users with no statistical training to build and apply machine learning models within spreadsheets. However, it has not been addressed how the user might conduct a sustained interaction with the system, building a model in the face of noisy data, outliers, and large datasets, where it is not possible to manually annotate many rows of data as being “correct” before needing to use the model to predict/evaluate new data. These complexities raise questions for the interface: how would the user know what rows to pick (example selection)? How would the user know their model has acquired good domain coverage? How would the user know if they needed additional data? How does the user establish whether some of the training data they have chosen might be noisy? These can be summarised as: *how can we implement the fourth design principle, namely, support dialogue through metamodels?*

Concretely, for the task of construction and application of machine learning models in spreadsheets, we wish to enable the end-user to:

1. *Judge the quality of the model*: to address this problem, multiple parallel visualisations are presented of the model’s training set representativeness and confidence – the ‘command’ and ‘confidence’ metamodels.
2. *Understand how their actions modify the model*: to address this problem, summary visualisations of the model’s understanding and progress are presented.
3. *Identify good training examples*: to address this, confidence-based colouring in the interface ‘nudges’ the user into focussing their attention on rows of data where the model has low confidence.

4. *Understand why and how the model makes certain predictions*: to address this, a force-directed layout visualisation of the k -nearest neighbours algorithm is presented, which provides a simple, consistent way of displaying decisions in an arbitrarily high-dimensional space, and also visualises a ‘prediction path’ metamodel.

The focus is on *prediction*, where the model is used to fill in missing information or used for classification/regression. The initial prototype also separately considered *evaluation*, where the model is used to assess the consistency of some data with respect to patterns in the training data. However, in BrainCel’s design, the prediction case subsumes the evaluation case, in the sense that it is clear that the interface can be used for both.

5.3.1 Design of the BrainCel interface

BrainCel is a browser-based prototype implemented using current web technologies. Data can be loaded from comma-separated (CSV) files on the user’s local filesystem, and must conform to a well-defined relational schema. The core of the interface is a standard spreadsheet component (item (a) in Figure 5.4), in which the loaded CSV file appears. Upon loading, columns are heuristically characterised as containing either categorical or continuous data, based on whether the cells in the column can be parsed as numeric.

Overview+detail+peeking

To the left of the spreadsheet (item (b) in Figure 5.4) is a narrow vertical column, which presents an overview of the entire spreadsheet. Hovering on the overview displays a preview of the rows in the proximity of the hover location (item (c) in Figure 5.4). This enables the user to “peek” at various parts of the spreadsheet without losing their current position in the main spreadsheet. Once the user has identified a region of the spreadsheet they would like to examine in greater detail, they can click on the overview to move the main spreadsheet to that position. This creates an augmented overview+detail (Cockburn et al., 2009); that is, *overview+detail+peeking*.

Model training and application

BrainCel uses the two-step interface for training and applying models introduced previously. The user first selects rows of data in which they have high confidence to select them as training data. By pressing the “Learn” button, rows in the selection are added to the training set. These rows are demarcated within the spreadsheet and overview with a blue highlight in the row label. After the model has been trained from rows with complete data, the user can apply the model to predict values for empty cells. They do so by selecting empty cells and using the “Guess” button to apply the model and predict a value for the cell. As discussed, the new button labels “Learn” and “Guess” were selected to address the usability issues of the previous labels “Teach” and “Try”.

“Guessing” uses the k -nearest neighbours algorithm (henceforth k -NN). Using the simple Euclidean distance metric, the algorithm selects k rows in the training set most similar to the row containing the cell for which a value is to be predicted. The distance between categorical values is set to 0 if equal, or 1 if unequal. If the cell is in a numerical column, the values of the neighbours in the same column are averaged to provide a guess. If the cell is in a categorical column, the majority vote (mode) is taken. Column values are not normalised or weighted. Many other algorithms, and even more sophisticated implementations of k -NN are possible, however, we have not considered this within the scope of the current prototype (this limitation is further discussed in Section 5.3.2).

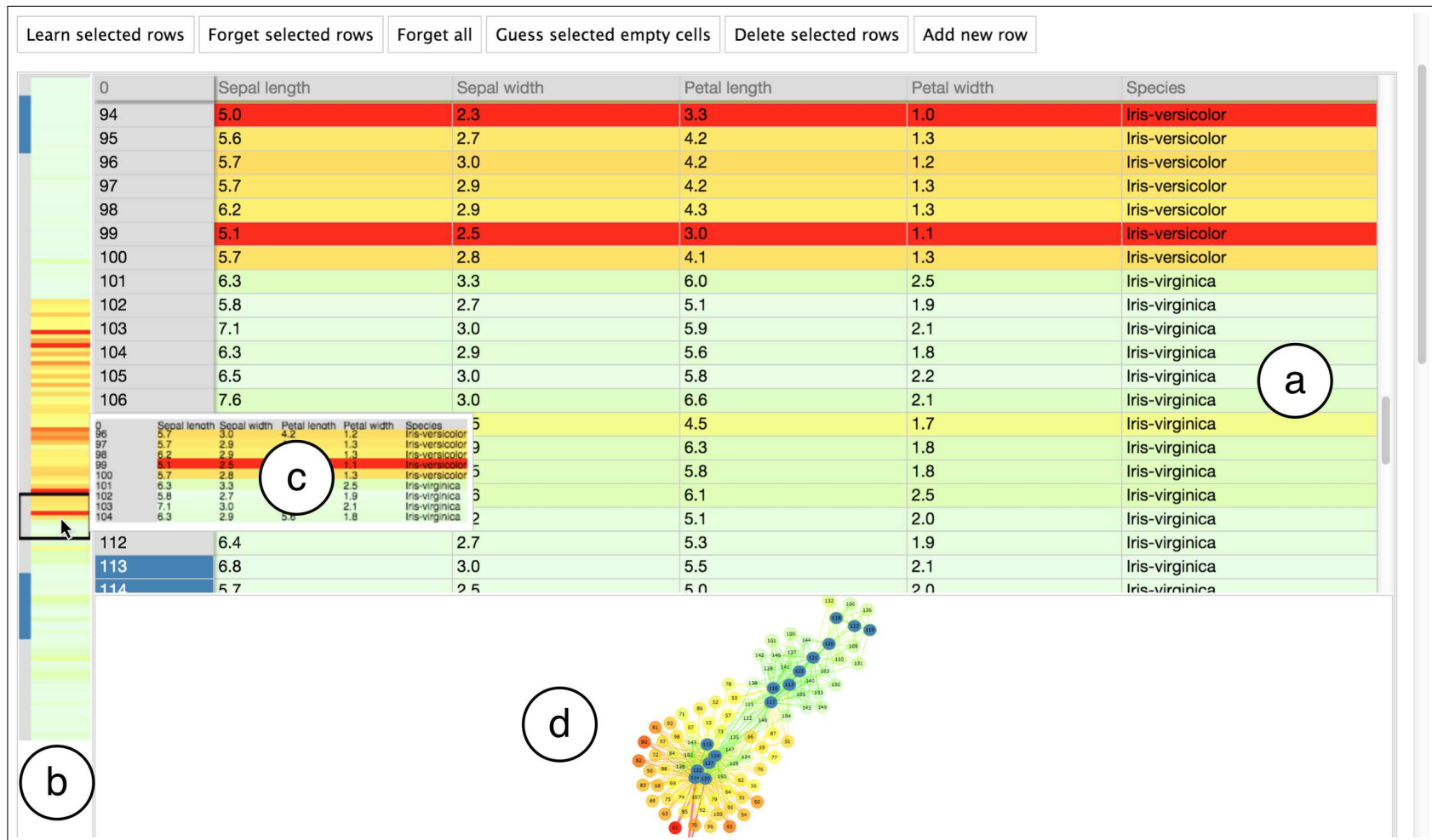


Figure 5.4: Part of the BrainCell interface showing (a) the core spreadsheet, (b) the overview, (c) peeking at the spreadsheet contents, and (d) the explanatory network visualisation. Rows with a blue highlight in their row label have been added by the user to the training set, and all rows have been coloured according to the model's confidence. Other parts of the interface, such as the distributions in Figure 5.7 and the progress graphs in Figure 5.8 appear further down in the interface.

Expressing confidence through colour

Measures of confidence can be used to prioritise human supervision of machine output; when there are large quantities of output to evaluate, the user’s attention can be focused on low-confidence outputs which are likely to be problematic. González-Rubio et al. (2010) use this approach to improve interactive machine translation, Kulesza et al. (2015) use this approach to improve interactive email classification, and Behrisch et al. (2014) use a live visualisation of model confidence to assist users in deciding when decision tree construction has converged. Groce et al. (2014) adopt the perspective of end-user classifier testing, and show various computational strategies for selecting a testbed of evaluation examples, among which they find selecting examples with low confidence to be effective. This work supported our design decisions regarding how to present the confidence of the system to the user, and how to help users select good training examples.

For each row in the spreadsheet, a confidence value is computed based on the mean distance to each of its k neighbours (Smith et al., 1994). This is linearly scaled into the range $[0, 1]$. Specifically, for a row \hat{r} , confidence is computed using the following expression:

$$1 - \frac{\frac{1}{k} \sum_{n \in kNN(\hat{r})} distance(\hat{r}, n)}{\max_{r \in Rows} \left(\frac{1}{k} \sum_{n \in kNN(r)} distance(r, n) \right)}$$

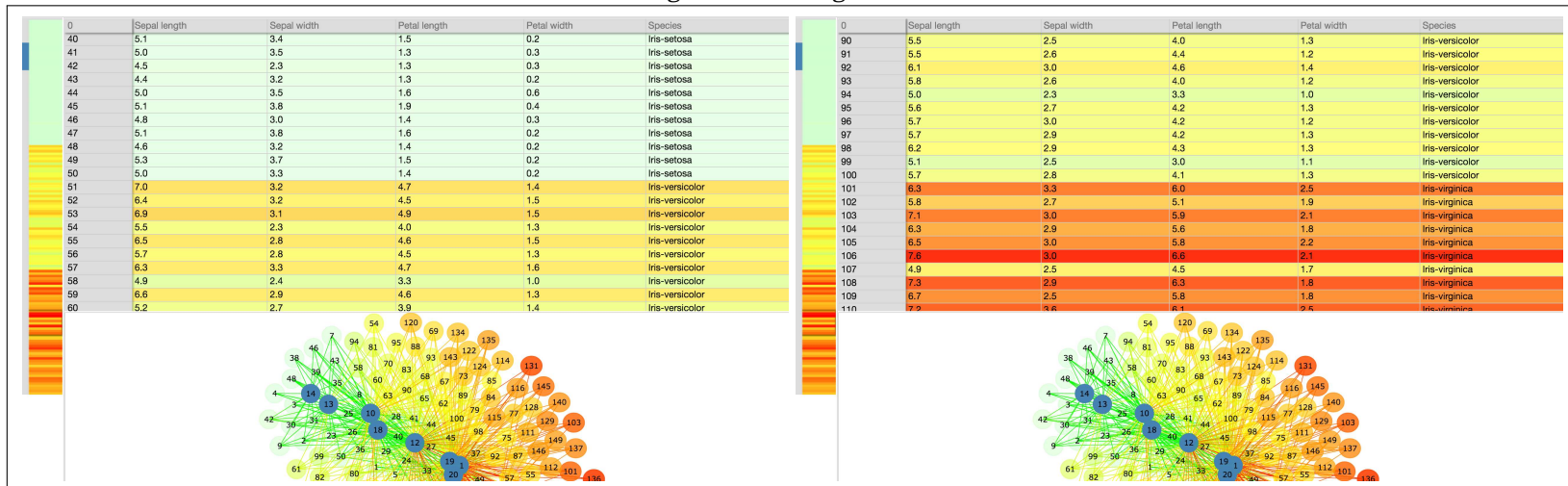
Where $Rows$ denotes the set of all rows, and $kNN(r)$ yields the set of k nearest neighbours for row r from the training set. In practice, averaging through multiplication with $1/k$ can be omitted since it appears in both numerator and denominator, but it appears here for clarity. Moreover, the denominator is constant for a given training set so it is only computed once each time the training set is edited. When the training set has cardinality lower than k , k is reduced to the cardinality of the training set. While more sophisticated notions of confidence exist, this confidence metric is chosen for its simplicity and efficiency.

A high mean distance is interpreted as *low* confidence; conversely, a low mean distance is interpreted as *high* confidence. A confidence value of 0 indicates that a row’s nearest neighbours are very far from it, and thus the model is less confident when it draws upon those neighbours to predict a value in that row. A confidence value of 1 indicates that a row’s nearest neighbours are close to it, and thus the model is more confident when it predicts values in that row.

Using the normalised confidence measure, the rows in the spreadsheet as well as the rows in the overview are coloured on a modified red-green scale, where rows coloured red are the ones where the machine has the least confidence. Lightness is also scaled to de-emphasise the visual salience of the green colour. A higher confidence is given a higher lightness, and capped at a maximal threshold to prevent complete whiteness. The effect of lightness scaling can be seen in Figure 5.5. A beneficial side-effect of lightness scaling is that it makes the scale safe for red-green colour-blindness.

Due to this colouring, the overview shows at a glance the state of the machine’s confidence over the entire spreadsheet. Green areas in the spreadsheet indicate that they are well-modelled by the training data, whereas red areas in the spreadsheet indicate that they are dissimilar to their nearest neighbours in the training set, marking them out either as cases which have not yet been covered in the training set or as outliers.

Lightness scaling enabled



Lightness scaling disabled

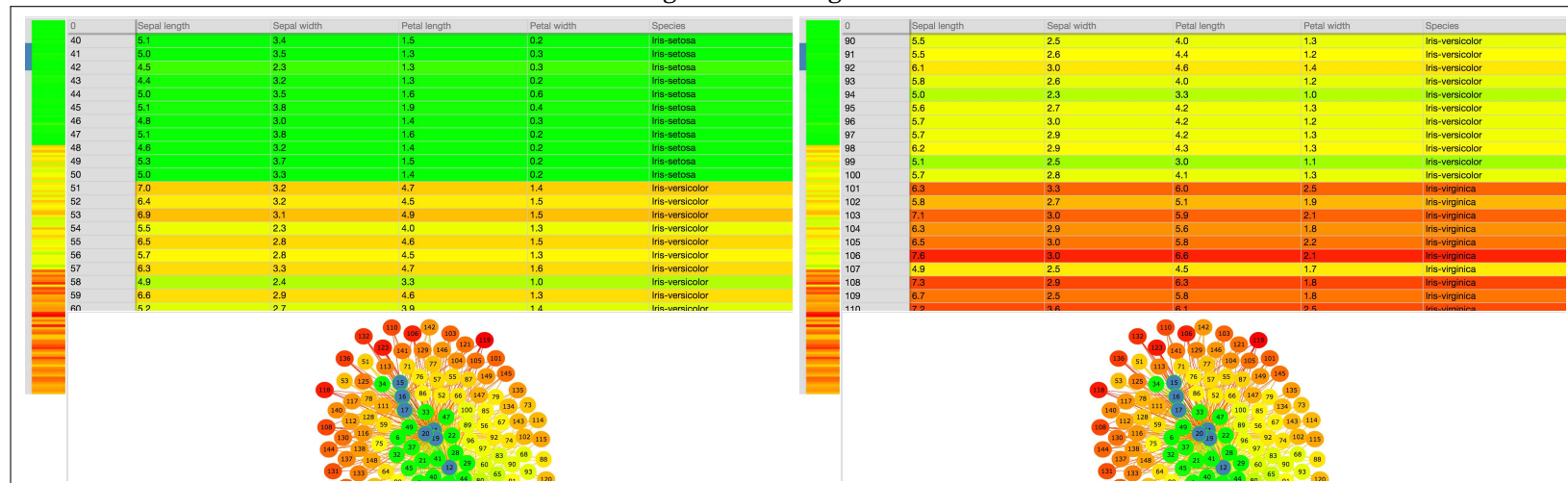


Figure 5.5: The effect of lightness scaling. Without lightness scaling, high-confidence (green) rows command disproportionately greater visual attention (the effect is most apparent onscreen).

Explaining k -NN to end-users

Below the spreadsheet, BrainCel displays a force-directed network visualisation of all the rows in the dataset (item (d) in Figure 5.4). The layout is computed using Barnes-Hut simulation (Barnes and Hut, 1986). Each node represents a row. Each node has k edges going to its k nearest neighbours in the training set. The lengths of these edges are proportional to the distances to the k neighbours. This projects the n -dimensional rows onto a 2D space. In order to explain the k -NN algorithm’s behaviour in the aggregate, it suffices to represent *proximity*, rather than variation along any particular dimension.

In this way, concrete interpretations of the spatial axes are sacrificed in favour of simply expressing the concepts of “nearness” and “farness”, exemplifying design principle 2 (abstract complex processes through heuristic automation). This layout method bears a similarity to t-distributed Stochastic Neighbour Embedding (t-SNE) (Van der Maaten and Hinton, 2008), which has not been investigated, but would be interesting future work.

This representation of k -NN has several desirable properties. First, it directly provides *why* and *why not*-style explanations for any given prediction: the answer to “why was this cell value guessed?” is that it drew upon the rows to which it was directly connected in the graph visualisation. Second, this very literal representation also provides a *how* explanation for the k -NN algorithm; the general answer to “how is a cell value guessed?” is that the model arranges all the rows by their mutual similarity and each guess draws upon the rows which are most relevant. Third, the emergent clusters visually reify the structure of the underlying abstract model. For example in a 3-class classification problem, as in the Iris dataset (popularised by Fisher (1936)), as the user adds more rows to the training data, the network shows how the underlying model’s characterisation of the input space first branches from one cluster into two, and then converges at three (Figure 5.6).

Fourth, a colour consistent with the spreadsheet reinforces the user’s understanding of how confidence works in the model. Nodes in the network visualisation are coloured according to the machine’s confidence in those rows in exactly the same fashion as the corresponding rows in the spreadsheet and overview are coloured (applying design principle 3: build expertise through iteration on multiple representations). Training set rows are coloured blue, but are given accents in the colour of the model’s confidence. While the use of the colour in the overview allows the user to understand the confidence of the model with respect to the structure of data as laid out in the spreadsheet, the colour in the network visualisation expresses confidence with respect to the structure of data as laid out in the high-dimensional similarity space. Since high-confidence rows are positioned very close to rows in the training set, it becomes clear that to help the model become more confident in a certain row, the user must either add training data which is similar to the row, or modify the row so that it is more similar to the training data.

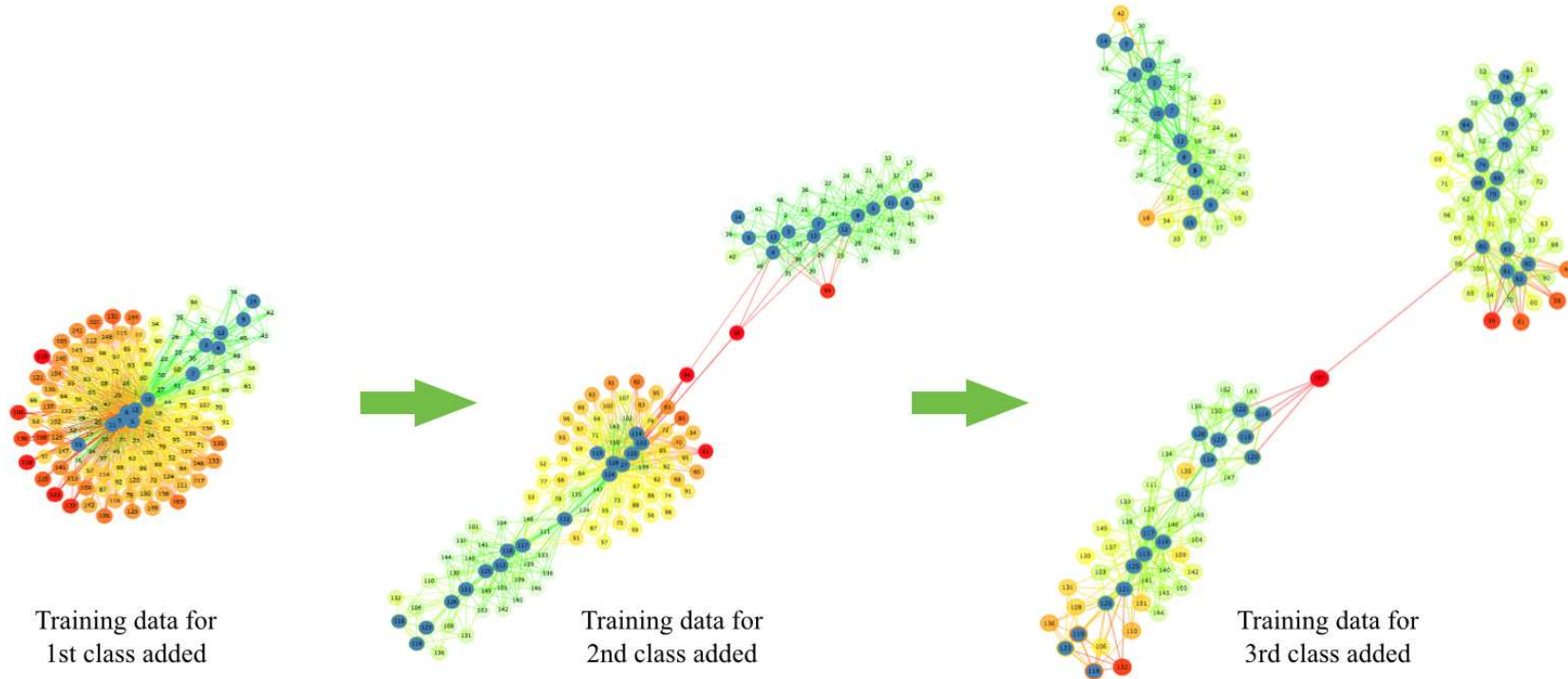


Figure 5.6: Evolution of the model as shown by BrainCel's network visualisation when data from the three classes in the Iris dataset (Fisher, 1936) is incrementally added. Rows in the training set are depicted in dark blue, and all other nodes are coloured according to their mean distance from their k nearest neighbours.

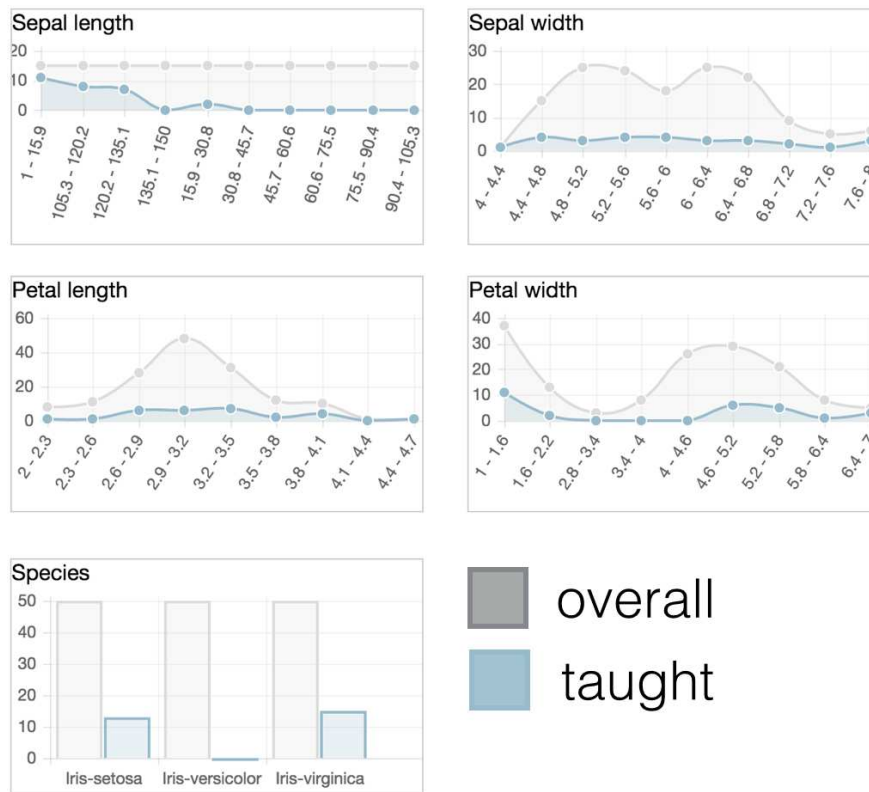


Figure 5.7: Visualisations of the the distributions of taught vs overall data. These express the “command” or “grasp” of the model over the domain; currently it is clear from the “species” distribution graph (bottom left) that the class *Iris-versicolor* is underrepresented in the training data.

Expressing training set representativeness

BrainCel displays the value distributions within the columns in the dataset (Figure 5.7). These distributions are compared against the distributions of the same column in the training set. These charts expose whether certain classes or types of data are under- or overrepresented, either in the underlying dataset, or in the training data. Thus, they provide a “command” metamodel. A distribution of taught data which matches more closely, or covers more extensively, the distribution of the data in the spreadsheet, is more representative of the domain than a distribution of taught data which poorly matches or covers the overall distribution.

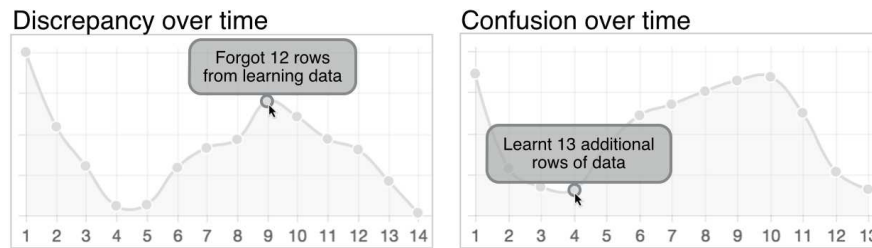


Figure 5.8: Left: total discrepancy between the overall dataset and the training data, that is, “command”. Right: visualisation of the machine’s overall confusion (interpreted as the inverse of confidence). The x -axis is an interaction history axis, with each new data point being created whenever the user edits the spreadsheet. Tooltips describe the action taken.

Visualising the progression of the model

The user needs to understand how the model evolved with their actions, in order to develop a consistent mental model for detecting and addressing prediction errors. BrainCel shows two line graphs which update over time (Figure 5.8).

The first graph represents the “Discrepancy” between taught and overall data. It summarises ‘command’ – how well the training data represents the overall dataset. New points are added whenever the training set is modified. It is measured as the sum of the Hellinger distances (Hellinger, 1909) from the distributions of attribute values in the training set to those of the overall dataset; a standard metric for comparing the divergence of distributions, which handles discrete as well as continuous distributions and can handle improper distributions and non-continuous functions (which Kullback-Liebler divergence does not). The command of the system improves, as the discrepancy between the distribution of its knowledge and the distribution of the overall data reduces.

The second is a graph of “confusion”, the inverse of confidence. The overall confidence is computed as the mean row confidence for all the rows in the spreadsheet. BrainCel records the mean confidence over all rows as a function of interaction history: a new data point is added whenever the spreadsheet is modified. This graph is vertically inverted to match the other graph, and presented to the user as representing the machine’s “confusion.” By inverting the graph in this manner, we unify the interpretation of the two graphs so that lower values are more desirable in both.

In both “confusion” and “discrepancy” graphs, a short description of the action which led to the data point being added is available in a tooltip. Thus, if the user sees a sudden spike or dip in “confusion” or “discrepancy,” they can hover over the relevant point to see descriptions such as “Learnt 6 rows of data”, “Forgot 12 rows”, “Edited cells,” etc. This provides a direct chronological account of how the understanding of the machine evolved in response to their actions, whether the model improved or degraded. These descriptions are also critical since the vertical scale on these graphs has no direct relationship to the user’s problem domain. It is more important that the user forms connections between specific actions they performed and a corresponding increase or decrease in the confidence and information requirements of the model. Accordingly, the y -axis is unlabelled to encourage the user to think of the quantities as merely increasing or decreasing, rather than focus on exact values. In future work, it would also be possible to capture images of the network visualisation and overview, and present these as graphical interaction histories (Kurlander and Feiner, 1988).

5.3.2 Design discussion

BrainCel has been designed to express information types identified as being important by previous research, but adapted to address the context of machine learning in spreadsheets. Primary design influences include Amershi et al. (2011a) and Lim and Dey (2009), who identify what types of information about intelligent applications should be given to end-users, and on the demonstration by Kulesza et al. (2013) that these information types are critical for the formation of users' mental models.

BrainCel's multiple visualisations of progress in the confidence as well as command dimensions enables users to judge when exploration has reached convergence. This directly builds upon Behrisch et al. (2014), where the user is able to decide when the exploration has reached convergence due to a live visualisation of how much of the data passes a certain threshold for classification confidence. Moreover, it exemplifies design principle 3: build expertise through iteration on multiple representations.

The confidence metric is used to highlight areas of the spreadsheet on which the user might wish to focus their attention, based on experimental evidence found by Groce et al. (2014) that model confidence was an effective way of selecting testing examples. However, this is only a suggestion to the user and the user is free to inspect other areas of the spreadsheet for testing if they so desire.

BrainCel incorporates at least 3 types of explanations, also exemplifying design principle 3: build expertise through iteration on multiple representations.. First, the network visualisation is a compact representation that not only shows how the system works, but secondly, it also answers specific "why" and "why not" questions about individual predictions, by exposing which rows were involved in the prediction. Thirdly, the column distributions show how much the computer knows. This is based on Kulesza et al.'s explorations of intelligibility types for music recommender systems (Kulesza et al., 2013), wherein the authors found that users' mental models were best if they were presented with explanations for how the system works. Users discussed more valid features than invalid ones when presented with explanations of what the computer knows. Finally, users most preferred discussing the concrete "why this song" explanations.

The emphasis in BrainCel is on showing how concepts evolve in the "mind" of the *computer*, as training data is added and edited (consider the sequence of networks in Figure 5.6), exemplifying design principle 4: support dialogue through metamodels. This is related to Kulesza et al.'s work in concept evolution (Kulesza et al., 2014), which acknowledges that users may not have well-defined mental concept models, and present an interface to help refine models in the mind of the *user*. The intersection of our two approaches suggests interfaces for facilitating joint concept evolution, whereby both machine as well as human understanding are visualised.

5.4 Exploratory user study

An exploratory study was conducted, modelled after Kulesza et al.'s work in end-user debugging of naïve Bayes email classification (Kulesza et al., 2011). Our motivation, like theirs, was not to conduct a summative evaluation of our prototype. Instead, the aim was to understand how our approach could support end-user machine learning by observing (1) when and where users encounter barriers to the task, and (2) what the users' information needs are. For compactness, the Kulesza et al. (2011) paper shall henceforth be referred to as 'the EC study', with 'EC' standing for 'email classification'.

0	Sepal length	Sepal width	Petal length	Petal width	Species
11	5.4	3.7	1.5	0.2	Iris-setosa
12	4.8	3.4	1.6	0.2	Iris-setosa
13	4.8	3.0	1.4	0.1	Iris-setosa
14	4.3	3.0	1.1	0.1	Iris-setosa
15	5.8	4.0	1.2	0.2	Iris-setosa
16	5.7	4.4	1.5	0.4	Iris-setosa
17	5.4	3.9	1.3	0.4	Iris-setosa
18	5.1	3.5	1.4	0.3	Iris-setosa
19	5.7	3.8	1.7	0.3	Iris-setosa
20	5.1	3.8	1.5	0.3	Iris-setosa
21	5.4	3.4	1.7	0.2	Iris-setosa
22	5.1	3.7	1.5	0.4	Iris-setosa
23	4.6	3.6	1.0	0.2	
24	5.1	3.3	1.7	0.5	
25	4.8	3.4	1.9	0.2	
26	5.0	3.0	1.6	0.2	
27	5.0	3.4	1.6	0.4	
28	5.2	3.5	1.5	0.2	
29	5.2	3.4	1.4	0.2	
30	4.7	3.2	1.6	0.2	
31	4.8	3.1	1.6	0.2	

Figure 5.9: The version of Braincel used in the *without*-visualisation condition.

Procedure

The study used a think-aloud protocol. Participants completed two equally-difficult tasks. They completed one task with a version of BrainCel that presents only the core spreadsheet component without any confidence-based colouring, which we shall refer to as the *without* condition (Figure 5.9), and one with the complete BrainCel interface, which we shall refer to as the *with* condition. In doing so, we drew upon the design of evaluation for the Whyline (Ko and Myers, 2004), which also compared a full-featured version to a version with reduced functionality. This helped us understand how our interface affected users’ exploration of their statistical models. In the tasks users were given a spreadsheet containing empty cells, and were asked to fill in the missing information using the software. The ground truth for this missing information is known to us, as the tasks were created by selectively deleting information from complete spreadsheets, in particular the Iris and Zoo datasets⁴, which were assigned to each condition in a randomised fashion.

Each participant first completed the *without* task before moving on to the *with* task. The interface was briefly demonstrated before each task. Task order was not randomised, despite the potential for learning effects, because learning effects were expected to be much greater if the *with* task was completed first, since it is designed to be an environment promoting learning and understanding of the modelling process. Moreover, this study was exploratory in nature; statistical analysis was not an intended outcome.

Participants

Participants with no prior exposure to statistics or machine learning were recruited from humanities departments at the University of Cambridge. A pre-experiment screening about the participants’ previous knowledge allowed us to enforce the non-expert categorisation, as in the initial study. When a participant indicated any awareness/prior exposure to statistical/machine learning concepts (e.g. mentioning a model such as “decision tree”, “neural network”, or domain terminology such as “regression”, “classification”, “least squares”, “hypothesis testing”, “training set” etc.), they were excluded from the subsequent analysis. We began with 8 participants and excluded 1 because of prior statistical knowledge, leaving 7 in the final analysis. All participants successfully completed both tasks by populating the spreadsheets with correct values in under 45 minutes.

⁴<http://archive.ics.uci.edu/ml> (last accessed April 29, 2018)

5.4.1 Analysis method

Since our task involves end-user programming of a machine learning system, we use the same coding scheme as the EC study, in particular, their modified version of Ko's learning barriers (Ko et al., 2004), which removes the sixth barrier (searching for external validation). In brief, the barrier codes are as follows:

1. *Design barrier*: the user's goals are unclear.
2. *Selection barrier*: the goal is clear but the programming tools required to achieve this goal are unclear.
3. *Use barrier*: the tools required are clear, but the user does not know how to use them properly.
4. *Coordination barrier*: the tools required are clear, but the user does not know how to make them work together.
5. *Understanding barrier*: the user thinks they know what to do, but their actions have surprising results.

We also adopt their code scheme for debugging, which incorporates the previous scheme due to Ko (2008), as follows:

1. *Fault detection*: the participant has detected an incorrect prediction by the system.
2. *Diagnosing*: the participant believes they have found the cause of a detected fault.
3. *Hypothesising*: the participant hypothesises a general (rather than specific) solution to a detected fault.

The codes were applied to sentences. A sentence containing a significant pause was segmented into two. If the same barrier spanned multiple contiguous sentences, it was coded as a single occurrence. Two researchers independently coded a random five-minute transcript excerpt. To maintain consistency with the EC study, we determined similarity by calculating the Jaccard index, dividing the size of the intersection of codes by the size of the union for each sentence, and then averaging over all sentences. Coding rules were iteratively refined and tested. Agreement reached 86% for a five-minute transcript section, and 83% for a complete 45-minute transcript. This level of reliability was considered acceptable, and the remaining transcripts were coded.

5.4.2 Exploratory study results

Learning barriers encountered with and without visualisations

Figure 5.10 shows the distributions of learning barriers encountered by our participants when using BrainCel with and without the additional visualisations. Our observations that Selection and Coordination barriers are most common overall mirror those of the EC study. We mostly observed Selection barriers occur because aspects of the interface were not intuitive (e.g., P1: *Can I select these to learn?*, P3: *They've highlighted, so that must mean they're okay, or does it?*). Our Coordination barriers, similar to ones observed in the EC study, occurred because the model's predictions are dependent on a complex sequence of program logic, which is sensitive to small changes to the training data or spreadsheet in non-obvious ways (e.g., P3: *So let's delete the [classifications] it got wrong and try again... no, they're still wrong.*, P4: *So, what are [the model's] problem areas?*).

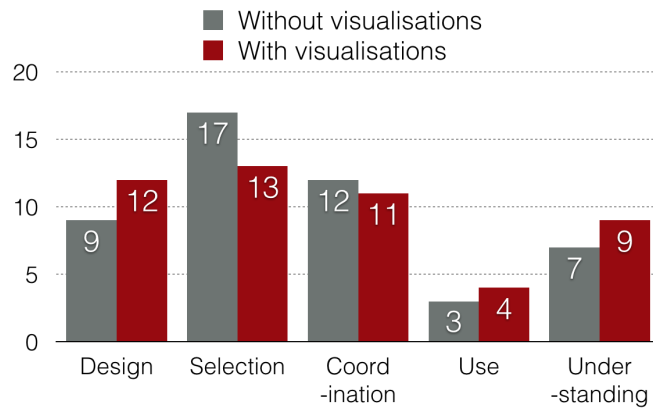


Figure 5.10: Number of learning barriers encountered in all transcripts.

Participants also encountered several Design and Understanding barriers. Design barriers mostly occurred when the model's predictions were wrong, and the participant had to choose a strategy for proceeding, for example, adding training data from elsewhere in the spreadsheet; editing the row to manually correct the prediction; editing the row and then adding the newly edited row itself to the training data.

Small sample notwithstanding, it appears as though the barriers noted in the EC study to be most prevalent (Selection and Coordination) are both less frequent when visualisations are introduced. Design, Use and Understanding barriers were observed more frequently when visualisations were available. This increase can be attributed to the fact that once users had their attention drawn to the structure of the model rather than surface features, the barriers they started to encounter became more interesting. For example, P5 expressed the following relatively mundane understanding barrier when working without the visualisations: *Why is that row [misclassified]?* However, with the visualisations, this participant moved on to describing understanding barriers that were more sophisticated, for example: *It's gotten that correct, but why is it still not confident about it?* The visualisations helped end-users to address more sophisticated conceptual issues, in a manner that can be considered as extending the zone of proximal development (Vygotsky, 1987) past the simpler concepts of the spreadsheet training paradigm to critical assessment of model in line with our view of IML as facilitating constructivist learning.

Figure 5.11 shows the major transitions that our participants made between barriers and debugging activities. Similar to the EC study, we find that this chart is dominated by an iterative loop from Selection barriers. Use barriers did not play as significant a role, and instead, we observed a frequent tendency to detect faults immediately upon encountering a Coordination barrier. This pattern emerges from the fact that making changes to the training set results in a recalculation of the model's confidence in each row, leading to the participant first speculating about how their modification resulted in certain improved or reduced confidences (Coordination), and subsequently spotting an error (Fault detection).

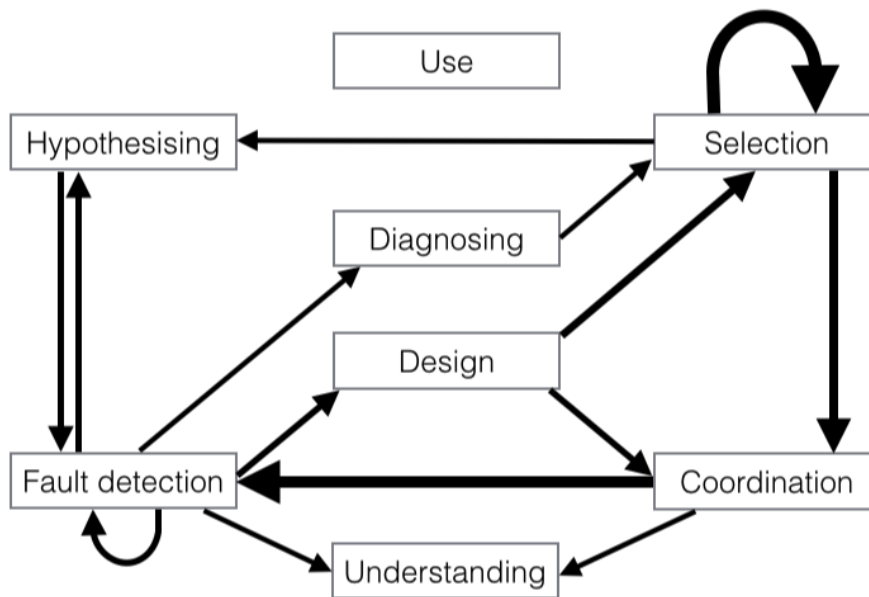


Figure 5.11: All participants' paths through the barriers and debugging activities. The width of the arrows indicate the percentage of transitions: thinnest = 3%, thickest = 7%. Transitions accounting for 2% or less of the total are not shown.

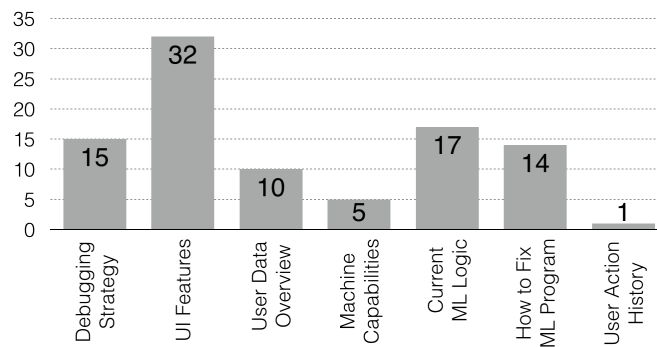


Figure 5.12: Number of times a participant implied a particular type of information would help them overcome their current barrier.

Participants' information needs

Figure 5.12 and Figure 5.13 present information needs according to the classification by Kulesza et al. (2011). These are summarised as follows:

1. *Debugging strategy*: should it learn all rows? Should I make the guesses all at once?
2. *UI features*: which button should I use? Can these rows be selected?
3. *User data overview*: how many classes are there? How many unconfident guesses were just made?
4. *Machine capabilities*: what does the model use to guess? What types of information can it process?
5. *Current ML logic*: why did the model behave in a particular manner?

	Debugging Strategy	UI Features	User Data Overview	Machine Capabilities	Current ML Logic	How to Fix ML Program	User Action History
Design	6	1	1	0	0	4	0
Selection	1	15	7	1	0	1	1
Coordination	2	1	1	0	9	2	0
Use	0	7	0	0	1	1	0
Understanding	0	10	0	1	5	0	0
Hypothesizing	5	0	1	3	2	5	0
Diagnosing	1	0	0	1	0	1	0

Figure 5.13: Participants’ information needs when encountering barriers. Values indicate the number of instances in which participants implied that they needed the information in the column header to overcome each barrier.

6. *How to fix ML program:* concrete steps to take to fix the model’s understanding. In BrainCel, this means which row(s) to add or delete to the training set, and what edits to make to the current row.
7. *User action history:* what did I do? Is what I did helpful? Is what I did wrong?

Similar to their findings, our participants frequently requested information pertaining to the program’s current logic and how to fix it. However, our participants most frequently requested information regarding how the various interface features worked. As shown in Figure 5.13, this correlates with the occurrences of Selection, Understanding, and Use barriers. Participants were sometimes confused about the relationship between the training set and the machine’s guesses (e.g., does the training set need to be re-specified for each guess, should a guessed row itself be in the training set, etc.) and whether they were able to treat the core spreadsheet component as a normal spreadsheet (e.g., P2: *Can I just edit this cell?*, P6: *So I can select these, right? Just click and drag?*). This shows that there is much room for improvement in the usability of our prototype. Nonetheless, all participants were able to overcome these information needs and successfully complete the task. A task was considered to have been successfully completed when the participant was satisfied that they had correctly filled in all the missing values, and these did indeed correspond to the correct ground truth values.

5.4.3 Activity flows

Participants’ transitions between interface activities in the *with*-visualisation tasks were recorded. The most common transitions are presented in Figure 5.14 (transitions accounting for <5% of the total are not shown). “Learn” corresponds to adding training data, “Guess” to invoking the model, and “Edit” to editing cell values. Network refers to the activity of inspecting the network visualisation. Edges are weighted according to how many times the participant switched from one activity to another.

Figure 5.14 is presented as an attempt to capture the patterns of debugging activities. There are several local loops showing repeated occurrences of learning, guessing and editing. This corresponds to an incremental approach commonly adopted by participants where in order to understand the machine’s logic and capabilities, they add data to the training set one row at a time, and apply the model to guess values one cell at a time, to try and inspect the direct consequence of adding or removing rows. Similarly, the pattern Edit→Learn→Guess also appears frequently: incrementally editing incorrect guesses,

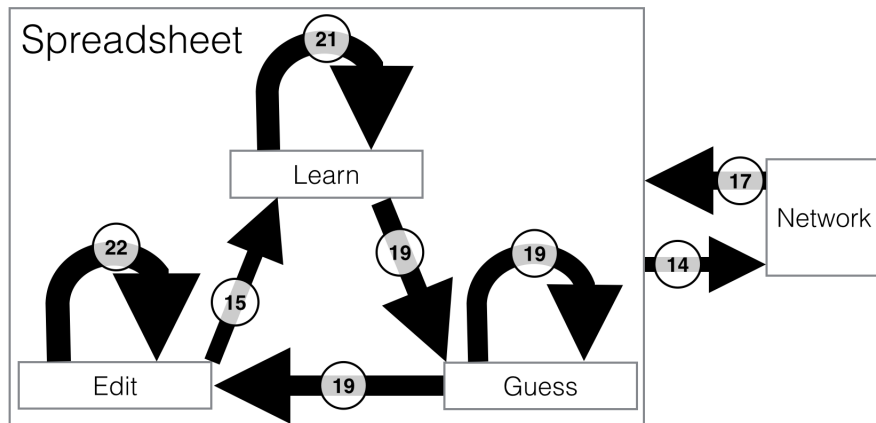


Figure 5.14: Participants’ paths through the interface. Numbers indicate occurrence counts of each transition. Transitions accounting for 2% or less of the total are not shown.

adding the corrected rows to the training set, and seeing if other incorrect guesses were now correct was a common pattern. This suggests that a lot of manual effort may be saved if guessed cells were not static, but *live* (Tanimoto, 1990), like spreadsheet formulae, so that the guess is re-evaluated every time the training set is modified. Interestingly, participants overwhelmingly preferred to incrementally *add* rows to the training data, rather than remove rows, in order to test the model. A plausible explanation, worthy of further study, is that removing knowledge to improve learning is fundamentally counterintuitive. This perhaps shows the limits of a naïve appropriation of ‘learning’ as a metaphor.

Another frequent pattern in Figure 5.14 is alternation between the spreadsheet and the network visualisation. The participants made heavy use of the network visualisation to identify outliers, areas of low confidence, diagnose mispredictions, study the model structure, and to check for why and why-not explanations of certain guesses. They then moved to the spreadsheet to inspect the row(s) implicated by the network, and potentially engaged in the Edit→Learn→Guess pattern.

5.5 Limitations and future work

Our initial work in this space has been an encouraging indicator that interactive visual machine learning in spreadsheets is viable. However, at least four important issues remain to be addressed, which are described now:

1. *Visualising models other than k-NN:* it is not clear how BrainCel’s network visualisation can be adapted for models other than *k-NN*. Our investigation also showed that the network visualisation was repeatedly used for addressing various information needs (Figure 5.14). The question is: how can we “explain” other machine learning algorithms? This would make an interesting subject of future study. One option is to try and explain other algorithms as though they were *k-NN*. In general, it might be possible to use a visualisation developed for one model to explain the behaviour of another while still maintaining consistency in the user’s mental model of the system’s behaviour, as suggested in the third chapter. Another option would be to create new, model-specific visualisations, or adopt visualisations which have been created on an ad-hoc basis for other models such as the bar graph visualisations for naïve Bayes classifiers used in the EC study. The general-purpose and modular nature of BrainCel makes it a flexible testbed for the comparison of different visual “explanations” of the same model.

2. *Varying model parameters and hyperparameters*: Our prototype currently does not support adjusting feature weights in the classifier, which is important as not all features are equally informative. Future work might investigate additional parameter controls, such as the bar-chart based controls for feature weighting in the naive Bayes classifier used in *EluciDebug* (Kulesza et al., 2015). It would also be interesting to introduce a way for users to explore the effects of varying the parameter k , and the distance metric, which is currently not user-configurable in the prototype, potentially through methods suggested by Brown et al. (2012b) to maintain low process expertise requirements.

3. *Additional interaction modalities*: the action of selection to indicate training and test data works well; however, it is not yet clear whether it is optimal. There are alternative approaches available. For instance: selecting only certain columns to learn could be used to indicate that a subset of features should be used for prediction. Another possibility is to invert the order in which the actions are taken, so that a mode is selected before making the selection, allowing the user to ‘paint’ regions of the spreadsheet as training or test data. These alternative modalities may be investigated through experimentation.

4. *New evaluation frameworks*: it is difficult to devise powerful, general evaluation criteria for IML systems due to the complexity of the tasks they facilitate. The learning barriers framework, which categorises barriers users encounter when learning new programming systems, is a mature approach from EUP which can potentially be used to evaluate IML systems. However, in our study, it emerged that the same type of learning barrier can manifest in qualitatively different ways; some barriers are ‘higher’ than others. Thus it may be possible to argue between two interfaces where users encounter a similar number of barriers, that one is more desirable because the barriers encountered were ‘harder’ or more ‘sophisticated.’ Our investigation suggested that analysis of the difference in learning barrier distributions in the *with*-visualisation and *without*-visualisation use of BrainCel was not sufficiently nuanced to show the way in which the visualisations helped. While participants still encountered several barriers with the visualisations, they were barriers at a more sophisticated level than the ones they encountered without the visualisations. This suggests future exploration for an additional dimension to the learning barriers which captures this notion of “hardness” or “sophistication.”

5.6 Conclusions

This chapter proposed that the spreadsheet is an ideal interface for machine learning, satisfying design principle 1 (begin the abstraction gradient at zero). A simple interaction technique for training and applying statistical models was described, satisfying design principle 2 (abstract complex processes through heuristic automation). An experiment demonstrated that the system could be successfully applied by users with no formal knowledge of statistics or computing, and that the experience of interacting with the system led them to acquire some understanding of the concepts underlying statistical modelling.

BrainCel, an interactive system for general-purpose machine learning in spreadsheets, was presented. Multiple coordinated views of the model explain its confidence in its predictions as well as its coverage of the input domain, helping the user evaluate and improve the model, satisfying design principles 3 and 4 (build expertise through multiple representations, and support dialogue through metamodels). An exploratory study confirmed that the novel approach of BrainCel retains the properties observed in previous work, but within a general purpose spreadsheet paradigm as originally proposed in the Teach and Try system. Through interaction design, it is possible to implement tools for users with no expertise in statistics, machine learning, or computing, enabling them to confidently apply sophisticated machine learning techniques for their own use.

CONCLUSION

This work is motivated by advances in machine learning, the proliferation of data, and the increasing socio-digital divides being created by limited access to tools and techniques for non-expert end-user data analytics.

The following research questions have been investigated in this dissertation:

1. Can tools be built for analytical modelling?
2. Can they be made useful for non-experts?

These questions have been pursued through exploratory design studies, where two new applications have been developed to enable non-expert end users to perform analytics and model building. These applications have been evaluated on non-expert end users, demonstrating their fitness-for-purpose.

6.1 Theory exemplified in Gatherminer and BrainCel

Three theoretical perspectives were presented in Chapter 3: visual analytics is end-user programming, interactive machine learning is dialogue, and analytical modelling is constructivist learning. Four emergent design principles were subsequently presented: begin the abstraction gradient at zero, abstract complex processes through heuristic automation, build expertise through multiple representations, support dialogue through metamodels. These are deeply embedded in both Gatherminer and BrainCel and the clearest instances are listed in this section. In accordance with research through design methodology, these theoretical perspectives are neither the foundations nor the consequences of the two systems – artefacts and theory were developed in parallel.

In both Gatherminer and BrainCel, data is always represented directly; the abstraction gradient begins at zero. In Gatherminer, the colour-mapped matrix is a lossless visual encoding of each data point. Gatherminer's treemap employs isotypes to represent data points. In BrainCel, data is represented directly using the standard spreadsheet paradigm. Consequently, users need not possess or learn the concept of a data abstraction (e.g., arrays, matrices, objects) and operations thereupon (e.g., slice, concatenate, map, filter) in order to manipulate data within these tools.

In Gatherminer, explanations are presented with multiple representations. This helps users of various expertise levels engage with the model. In the treemap visualisation, the isotype circles are redundantly mapped to the data count, but this serves to ground the analysis in concrete numbers. In BrainCel, the use of confidence-based colouring throughout the spreadsheet, overview, and network visualisations helps users create sound mental models of how confidence is computed and how to interpret it for their own analysis.

In BrainCel, ‘debugging’ is facilitated by colouring, which draws the user’s attention to low confidence nodes, and by the network visualisation, which provide why, why-not, and how style explanations for any given prediction. In Gatherminer, why/why-not style explanations are provided by the rules represented by paths on the tree, and the interactive process of iterative drill-down provides a how explanation.

The Teach and Try study showed that the experience of interacting with the software led users to gain some appreciation of statistical procedures. In a post-experiment interview, participants were asked questions such as *how might the computer be doing this?*, and *why might the computer make a mistake?* Despite having no formal training in statistics or computing, participants informally articulated several potential algorithms (e.g., nearest-neighbours, case-based reasoning, and linear regression), and well-known issues in statistical modelling (e.g., insufficient data, insufficient dimensions, outliers, noise, etc.).

Building on Teach and Try, BrainCel was motivated by the idea that with multiple meta-models, the system would support dialogue for users to critically evaluate the model and its predictions. A confidence metamodel provides a heuristic with which the user can assess its performance and choose to prioritise examining and correcting low-confidence predictions. A command metamodel shows users how the ‘taught’ rows are spread across the input domain, and highlights areas where receiving a user label would be beneficial. Confidence and command metamodels are present in BrainCel in the form of confidence-based colouring, and the training set representation bar charts. Gatherminer conveys the same information through colour averaging and dot count, presenting the information needed to critically evaluate each node as a rule with analytical merit. An ‘impure’ average colour can be interpreted as lower confidence, since the node has low class purity. Similarly, a low dot count can be interpreted both as low command as well as low confidence, since the node is not supported by many data points. Node depth corresponds to complexity.

Constructivist design features

Ill-defined problem: Both Gatherminer and BrainCel address ill-defined problems. BrainCel does so because what constitutes a ‘good’ model is not well-defined and is subject to interpretation. Gatherminer begins with the premise that the shape of interesting patterns is not known *a priori*, and its design is built for that situation. Moreover, the design of elements such as the treemap representation has explicitly considered the role of human expertise in judging the analytical merit of rules.

Reflexivity: BrainCel’s interactive history visualisation provides a basic level of provenance for the current model. However, in both Gatherminer and BrainCel, there is potential for more sophisticated treatment, such as with interactive graphical histories.

Iterative interaction: the core interaction pattern in BrainCel was the Edit→Learn→Guess loop. Users iteratively added or removed rows, which they believed helped them understand how the model evolved. This rapid iteration provides the ideal setting for the user’s mental model to interact with their experiences. Moreover, the confidence-based colouring, as well as instances of incorrect guesses, provide the necessary mental model *perturbation* which encourages further learning behaviours.

6.2 Contributions

1. Analytics through model building

This dissertation has recognised and characterised the activity combining analytics and model building as *analytical modelling*, placing an emphasis on how it might be carried out with interactive computer tools. Two systems have been built to illustrate the potential for interactive analytical modelling tools. Gatherminer is a visual analytics tool which enables non-expert analysts to interactively train decision tree classifiers to explain patterns in time series data. BrainCel is a general-purpose interactive analytical modelling tool which allows non-expert end-users to build and apply statistical models in spreadsheets. Through user studies, both systems have been evaluated and found to facilitate analytical modelling by non-expert end-users. Evaluation of Gatherminer has also exposed a discussion around the confounding effects of domain expertise on controlled usability experiments, which is not typically addressed by visual analytics research.

2. Theoretical perspectives

Three theoretical perspectives have been developed with implications for design in interactive analytical modelling.

The first is that visual analytics is end-user programming. This view is justified by observing that visual analytics systems facilitate the generation and testing of informal visual hypotheses which have equivalent interpretations as formal statistical programs. This enables us to draw upon the EUP literature to inform the design of visual analytics systems.

The second is that interactive analytical modelling is dialogue. This view stems from the observation that the optimisation goal in interactive machine learning becomes increasingly ill-defined as the lines between analytics and model-building are blurred; consequently the programmer is no longer in the position of being able to define what constitutes a ‘bug’. This necessitates a shift in emphasis from the *direct* inspection channel to the *indirect*, which contains information describing the program, as opposed to the program itself. Ad-hoc exploitation of the indirect channel is common in EUP design research, but for the purposes of interactive analytical modelling, systematic metamodelling is proposed as a fundamental addition to this channel.

The final perspective is that interactive analytical modelling is constructivist learning. This follows from the observation that the main outcomes of interactive analytical modelling can be viewed as learning objectives: learning about the model, learning about the data, and learning about the process. The rapid, iterative, incremental interaction in these systems is the ideal setting for constructivist learning. Consequently, if the objective is to maximise learning, this perspective suggests that designers exploit those aspects of constructivist environments to which interactive analytical modelling is already amenable, namely: task ownership, ill-defined problems, and perturbation. Moreover, it suggests new explicit design considerations: reflexivity, collaboration, and task-in-context.

3. Design principles

Four principles for the design of analytical modelling systems have been derived, with a focus on non-expert end users.

The first is to begin the abstraction gradient at zero. Analytical modelling systems can reduce *representational expertise* requirements by always including at least one representation with zero abstraction. Concretely, this corresponds to having at least one direct visual representation of the data being operated upon.

The second is to abstract complex processes through heuristic automation. Analytical modelling systems can reduce *process expertise* requirements in order to deploy statistical and machine learning techniques by exposing as few (hyper)parameters as possible and relying on heuristic inference.

The third is to build expertise through iteration on multiple representations. Analytical modelling systems can build both *representational* as well as *process* expertise by providing multiple representations of the model and its output at varying levels of complexity. Rapid, incremental interaction with multiple representations provides constructivist scaffolding for grappling with complex concepts.

The fourth is to support dialogue through metamodels. Analytical modelling can support the user by providing meta-information such as the model's confidence, command, and prediction path. When building a complex model, it is difficult to gain insight into what action should be taken next by inspecting the workings (parameters) of the model directly. This can only be framed as a dialogue since the goal is ill-defined and both user as well as system have some degree of agency.

4. Design artefacts

Various aspects of the design of Gatherminer and BrainCel are novel contributions which may be isolated and repurposed for other future systems:

- The augmentation of overview+detail by the introduction of *peeking* is a modification which improves the fluidity of overview+detail interfaces.
- Both BrainCel and Gatherminer make careful use of colour. BrainCel uses a red-green scale but scaled for lightness in order to compensate for the visual saliency of green, and Gatherminer defaults to cumulative distribution normalisation as a robust, distribution-invariant method of mapping colour spectra to data domains.
- A new visual representation for k -NN models is presented.
- A new representation for decision trees is presented, with a first-principles argument for how mechanical bias can affect the exploration of a particular representation.

6.3 Future work

Model substitution

Illustrating the structure and parameters of a statistical model is an ad-hoc design problem. A visualisation for a naïve Bayes model, for instance, requires explicit representation of class priors, empirically observed feature distributions, and resulting class distribution for an example datum. To visualise a decision tree model, the design research presented in this dissertation has had to consider explicit representation of the tree structure in terms of its node hierarchy. To represent a k -NN model, a mapping capable of robustly representing the concept of “neighbours” has had to be designed.

However, there are several classes of models for which designing faithful visual representations for non-experts are not straightforward. Ensembles of decision trees, support vector machines, and neural networks fall into this category. It is entirely possible that a visual representation of these models exists which *could* facilitate dialogue, but is simply yet to be designed. However, until that point, the absence of a well-designed representation should not preclude our use of these powerful models.

In Chapter 3, substituting visual representations and metamodel computations of one model as a *metaphor* for another was suggested as a possible solution to this problem. This has not been investigated, but it would make for very interesting future work. For instance, one might retain the network visualisation in BrainCel, whilst replacing the underlying model with a decision tree ensemble. Some mathematical finesse would be required to identify when the visualisation has diverged from the model, and the interface should modify itself to gracefully accommodate these situations. Emulating one model with another is not completely far-fetched; recent work has shown how decision tree ensembles can be used to approximate nearest-neighbour matching (Konukoglu et al., 2013).

Education

With growing public interest in machine learning and artificial intelligence, a major application for tools such as BrainCel and Gatherminer is to provide a scaffolded learning environment within which students can experiment with machine learning. BrainCel, in particular, provides a graphical layer within a familiar spreadsheet environment for students to build and apply models on arbitrary datasets. In contrast, typical first machine learning courses are introduced using programming languages such as Python, R, or MATLAB. The requirements for representational and notational expertise, in particular with manipulation of data structures in language or package-specific idioms, are very high with these languages. A recent study compared a scaffolded interaction tool for statistical analyses with a traditional lecture for creating understanding of statistical concepts as measured by test scores (Wacharamanotham et al., 2015). BrainCel could be similarly compared with an introductory machine learning lecture for teaching efficacy.

Large datasets

Scalability to large datasets has been investigated to some extent already in this dissertation. For instance, the use of the scrollable thumbnail in Gatherminer, and the spreadsheet overview in BrainCel, has necessitated innovations in the design of overview+detail mechanisms. This has enabled the interfaces to scale to datasets of moderate sizes, where the scaling factor starts becoming unwieldy at the order of 10^5 data rows. However, new technical as well as representational approaches will be required for *truly* large datasets, of the order of 10^{9+} training examples, typical of modern machine learning applications. Recent work (Sarkar et al., 2014a, 2015) has investigated and demonstrated how approaches to approximate computation can be smoothly integrated into existing diagrammatic conventions to create simple interfaces for interaction with such large datasets at interactive speeds. Fusing these approaches to enable interaction and training with large datasets in analytical modelling tools is an interesting future avenue.

Informal visual statistics

In Chapter 3, visual analytics was argued to be a form of end-user programming on the basis that visual reasoning is an approximate process which can generate and test statistical hypotheses in the same way as statistical programs. Several examples were given of how certain visualisations answer questions which could equivalently be asked through code. These examples were meant purely to be illustrative of the idea of visualisations as expressing statistical programs, and not a rigorous taxonomy. However, a thorough exploration of this would be very interesting. For instance, a corpus of scientific charts could be gathered, and through crowdsourced annotations, one could begin to categorise the most common statistical insights and hypotheses expressed by each basic chart type. This could lead to interesting hybrid statistical visualisation systems, where the act of creating a particular chart suggests that the corresponding statistical analyses be automatically computed, or the act of invoking a statistical method suggests that the corresponding visualisation be automatically drawn.

6.4 Conclusion

In this dissertation, the activity of *interactive analytical modelling*, which seeks to create analytic insight from data through building predictive models, has been examined. While this activity is by no means new, this dissertation is the first to recognise it as encountering research problems at the intersection of visual analytics, interactive machine learning, and end-user programming. The problem of interactive analytical modelling is framed explicitly as an interaction design problem, as opposed to a problem of designing new statistical models to navigate the tradeoff between accuracy and intelligibility.

An important new mode of human-computer interaction is emerging: end-user programming of machine-learned models. Moreover, the increasing sophistication of analytical methods in our knowledge economy will expand the divide between those who can use them, and those who cannot.

This dissertation has presented work that empowers non-expert end-users to build and apply machine learning models for analytical purposes. New theoretical contributions have been made, highlighting the complexity of developing general-purpose theory for interaction design, and expanding the cache of intermediate-level knowledge regarding visual analytics, interactive machine learning, end-user programming, and constructivist learning. Together, these contributions illustrate how the design of analytical modelling interfaces for non-expert end users is a challenging new research field, with implications not only for human-computer interaction and machine learning, but also for ensuring a more equitable and accessible knowledge economy of the future.

BIBLIOGRAPHY

- Abelson, Harold and DiSessa, Andrea. *Turtle geometry: The computer as a medium for exploring mathematics*. MIT press, 1986.
- Adrienko, Natalia and Adrienko, Gennady. Spatial generalization and aggregation of massive movement data. *Visualization and Computer Graphics, IEEE Transactions on*, 17(2):205–219, 2011.
- Amar, Robert; Eagan, James, and Stasko, John. Low-level components of analytic activity in information visualization. *IEEE Symposium on Information Visualization*, pages 111–117, 2005. doi: 10.1109/INFVIS.2005.1532136.
- Amershi, Saleema; Fogarty, James; Kapoor, Ashish, and Tan, Desney. Overview based example selection in end user interactive concept learning. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 247–256. ACM, 2009.
- Amershi, Saleema; Fogarty, James; Kapoor, Ashish, and Tan, Desney. Effective end-user interaction with machine learning. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI’11*, pages 1529–1532. AAAI Press, 2011a.
- Amershi, Saleema; Lee, Bongshin; Kapoor, Ashish; Mahajan, Ratul, and Christian, Blaine. CueT: Human-Guided Fast and Accurate Network Alarm Triage. In *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI ’11*, page 157, New York, New York, USA, 2011b. ACM Press. ISBN 9781450302289. doi: 10.1145/1978942.1978966.
- Anscombe, Francis J. Graphs in statistical analysis. *The American Statistician*, 27(1): 17–21, 1973.
- Baldwin, Alfred Lee. *Theories of child development*. Wiley, Oxford, England, 1967.
- Barnes, Josh and Hut, Piet. A hierarchical $O(N \log N)$ force-calculation algorithm. *Nature*, 324(6096):446–449, 1986.
- Baum, Eric B and Lang, Kenneth. Query learning can work poorly when a human oracle is used. In *International Joint Conference on Neural Networks*, volume 8, pages 335–340. IEEE, 1992.
- Bayarri, M Jesús and Berger, James O. The interplay of bayesian and frequentist analysis. *Statistical Science*, pages 58–80, 2004.

- Behrisch, Michael; Korkmaz, Fatih; Shao, Lin, and Schreck, Tobias. Feedback-driven interactive exploration of large multidimensional data supported by visual classifier. In *Visual Analytics Science and Technology (VAST), 2014 IEEE Conference on*, pages 43–52. IEEE, 2014. doi: 10.1109/VAST.2014.7042480.
- Bernard, Jürgen; Ruppert, Tobias; Scherer, Maximilian; Schreck, Tobias, and Kohlhammer, Jörn. Guided discovery of interesting relationships between time series clusters and metadata properties. In *Proceedings of the 12th International Conference on Knowledge Management and Knowledge Technologies*, page 22. ACM, 2012.
- Bernard, Jurgen; Daberkow, Debora; Fellner, Dieter; Fischer, Katrin; Koepler, Oliver; Kohlhammer, Jorn; Runnwerth, Mila; Ruppert, Tobias; Schreck, Tobias, and Sens, Irina. Visinfo: a digital library system for time series research data based on exploratory search—a user-centered design approach. *International Journal on Digital Libraries*, pages 1–23, 2014a. ISSN 1432-5012. doi: 10.1007/s00799-014-0134-y.
- Bernard, Jurgen; Sessler, David; Behrisch, Michael; Hutter, Marco; Schreck, Tobias, and Kohlhammer, Jorn. Towards a user-defined visual-interactive definition of similarity functions for mixed data. In *2014 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 227–228, 2014b. doi: 10.1109/VAST.2014.7042503.
- Berndt, Donald J and Clifford, James. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.
- Bernstein, Abraham; Provost, Foster, and Hill, Shawndra. Toward intelligent assistance for a data mining process: An ontology-based approach for cost-sensitive classification. *Knowledge and Data Engineering, IEEE Transactions on*, 17(4):503–518, 2005.
- Bertin, Jacques. *Graphics and graphic information processing*. Walter de Gruyter, 1981.
- Bertin, Jacques. *Semiology of Graphics: Diagrams, Networks, Maps*. University of Wisconsin Press, 1983.
- Blackwell, Alan. Patterns of User Experience in Performance Programming. In *Proceedings of the First International Conference on Live Coding*, pages 12–22. ICSRiM, University of Leeds, July 2015a. doi: 10.5281/zenodo.19315.
- Blackwell, Alan F. Palimpsest: A layered language for exploratory image processing. *Journal of Visual Languages & Computing*, 25(5):545–571, 2014.
- Blackwell, Alan F. Interacting with an inferred world: The challenge of machine learning for humane computer interaction. In *Proceedings of The Fifth Decennial Aarhus Conference on Critical Alternatives, AA '15*, pages 169–180. Aarhus University Press, 2015b. doi: 10.7146/aahcc.v1i1.21197.
- Boyd, Danah and Crawford, Kate. Critical questions for big data: Provocations for a cultural, technological, and scholarly phenomenon. *Information, communication & society*, 15(5):662–679, 2012.
- Brehmer, Matthew and Munzner, Tamara. A multi-level typology of abstract visualization tasks. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2376–2385, 2013.
- Breiman, Leo and others, . Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical Science*, 16(3):199–231, 2001.

- Breunig, Markus M; Kriegel, Hans-Peter; Ng, Raymond T, and Sander, Jörg. Lof: identifying density-based local outliers. In *ACM SIGMOD Record*, volume 29, pages 93–104. ACM, 2000.
- Brinton, Willard Cope. *Graphic methods for presenting facts*. Engineering magazine company, 1914.
- Brown, Eli T.; Liu, Jingjing; Brodley, Carla E., and Chang, Remco. Dis-function: Learning distance functions interactively. *IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 83–92, 2012a. doi: 10.1109/VAST.2012.6400486.
- Brown, Eli T; Liu, Jingjing; Brodley, Carla E, and Chang, Remco. Dis-function: Learning distance functions interactively. In *Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on*, pages 83–92. IEEE, 2012b.
- Bruls, Mark; Huizing, Kees, and Van Wijk, Jarke J. *Squarified treemaps*. Springer, 2000.
- Buchanan, Richard. Wicked problems in design thinking. *Design issues*, 8(2):5–21, 1992.
- Buntine, Wray; Fischer, Bernd, and Pressburger, Thomas. Towards automated synthesis of data mining programs. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 372–376. ACM, 1999.
- Buono, Paolo; Aris, Aleks; Plaisant, Catherine; Khella, Amir, and Shneiderman, Ben. Interactive pattern search in time series. In *Electronic Imaging 2005*, pages 175–186. International Society for Optics and Photonics, 2005.
- Campos, Marcos M; Stengard, Peter J, and Milenova, Boriana L. Data-centric automated data mining. In *Machine Learning and Applications, 2005. Proceedings. Fourth International Conference on*, pages 8–15. IEEE Computer Society, 2005.
- Card, Stuart K; Mackinlay, Jock D, and Shneiderman, Ben. *Readings in information visualization: using vision to think*. Morgan Kaufmann, 1999.
- Carlin, Bradley P and Gelfand, Alan E. Approaches for empirical bayes confidence intervals. *Journal of the American Statistical Association*, 85(409):105–114, 1990.
- Caruana, Rich; Lou, Yin; Gehrke, Johannes; Koch, Paul; Sturm, Marc, and Elhadad, Noemie. Intelligible Models for HealthCare. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '15*, pages 1721–1730, New York, New York, USA, 2015. ACM Press. doi: 10.1145/2783258.2788613.
- Chiu, Bill; Keogh, Eamonn, and Lonardi, Stefano. Probabilistic discovery of time series motifs. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 493–498, New York, NY, USA, 2003. ACM. ISBN 1-58113-737-0. doi: 10.1145/956750.956808.
- Chryssolouris, George; Lee, Moshin, and Ramsey, Alvin. Confidence interval prediction for neural network models. *Neural Networks, IEEE Transactions on*, 7(1):229–232, 1996.
- Clark, Herbert H; Brennan, Susan E, and others, . Grounding in communication. *Perspectives on socially shared cognition*, 13(1991):127–149, 1991.
- Cleveland, William S and McGill, Robert. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American statistical association*, 79(387):531–554, 1984.

- Cockburn, Andy; Karlson, Amy, and Bederson, Benjamin B. A review of overview+detail, zooming, and focus+context interfaces. *ACM Comput. Surv.*, 41(1):2:1–2:31, January 2009. ISSN 0360-0300. doi: 10.1145/1456650.1456652.
- Cohn, David A; Ghahramani, Zoubin, and Jordan, Michael I. Active learning with statistical models. *Journal of artificial intelligence research*, 1996.
- Compeau, Deborah R and Higgins, Christopher A. Computer self-efficacy: Development of a measure and initial test. *MIS quarterly*, pages 189–211, 1995.
- Cooper, Gregory F; Aliferis, Constantin F; Ambrosino, Richard; Aronis, John; Buchanan, Bruce G; Caruana, Richard; Fine, Michael J; Glymour, Clark; Gordon, Geoffrey; Hanusa, Barbara H, and others, . An evaluation of machine-learning methods for predicting pneumonia mortality. *Artificial intelligence in medicine*, 9(2):107–138, 1997.
- Cooper, Stephen; Dann, Wanda, and Pausch, Randy. Alice: a 3-d tool for introductory programming concepts. In *Journal of Computing Sciences in Colleges*, volume 15, pages 107–116. Consortium for Computing Sciences in Colleges, 2000.
- Cover, Thomas M and Thomas, Joy A. *Elements of information theory*. John Wiley & Sons, 2012.
- Craw, Susan; Sleeman, D; Graner, N; Rissakis, M, and Sharma, S. Consultant: Providing advice for the machine learning toolbox. In *Proceedings of the Research and Development in Expert Systems IX*, pages 5–23. Cambridge University Press, 1992.
- Cunningham, Donald J. and Duffy, Thomas M. Constructivism: Implications for the design and delivery of instruction. *Handbook of research for educational communications and technology*, pages 170–198, 1996.
- Cutting, Douglass R; Karger, David R; Pedersen, Jan O, and Tukey, John W. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 318–329. ACM, 1992.
- Dann, Wanda P; Cooper, Stephen, and Pausch, Randy. *Learning to Program with Alice (w/CD ROM)*. Prentice Hall Press, 2011.
- Davies, David L and Bouldin, Donald W. A cluster separation measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (2):224–227, 1979.
- Diamantini, Claudia; Potena, Domenico, and Storti, Emanuele. KDDONTO: An Ontology for Discovery and Composition of KDD Algorithms. *Proc. 2nd Intl. Wshp. on Third Generation Data Mining: Towards Service-Oriented Knowledge Discovery*, pages 13–25, 2009.
- Dunn, Joseph C. Well-separated clusters and optimal fuzzy partitions. *Journal of cybernetics*, 4(1):95–104, 1974.
- Duvenaud, David; Lloyd, James Robert; Grosse, Roger; Tenenbaum, Joshua B., and Ghahramani, Zoubin. Structure discovery in nonparametric regression through compositional kernel search. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1166–1174, 2013.
- Eddy, Sean R. What is a hidden markov model? *Nature biotechnology*, 22(10):1315–1316, 2004.

- Efron, Bradley. Bayesians, frequentists, and scientists. *Journal of the American Statistical Association*, 100(469):1–5, 2005.
- Elmqvist, Niklas; Do, Thanh-Nghi; Goodell, Howard; Henry, Nathalie, and Fekete, J. Zame: Interactive large-scale graph visualization. In *Visualization Symposium, 2008. PacificVIS'08. IEEE Pacific*, pages 215–222. IEEE, 2008.
- Endert, A.; Han, Chao; Maiti, D.; House, L.; Leman, S., and North, C. Observation-level interaction with statistical models for visual analytics. In *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on*, pages 121–130, Oct 2011. doi: 10.1109/VAST.2011.6102449.
- Ericsson, K Anders and Lehmann, Andreas C. Expert and exceptional performance: Evidence of maximal adaptation to task constraints. *Annual review of psychology*, 47(1): 273–305, 1996.
- Fails, Jerry Alan and Olsen Jr, Dan R. Interactive machine learning. In *Proceedings of the 8th international conference on Intelligent user interfaces (IUI)*, pages 39–45. ACM, 2003. doi: 10.1145/604050.604056.
- Fisher, Danyel; Popov, Igor; Drucker, Steven, and others, . Trust me, i'm partially right: incremental visualization lets analysts explore large datasets faster. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1673–1682. ACM, 2012.
- Fisher, Ronald A. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.
- Fogarty, James; Tan, Desney; Kapoor, Ashish, and Winder, Simon. Cueflik: interactive concept learning in image search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 29–38. ACM, 2008.
- Fosnot, Catherine Twomey. *Constructivism: Theory, perspectives, and practice*. Teachers College Press, 2013.
- Frayling, Christopher. *Research in art and design*. Royal College of Art, London, 1993.
- Gaver, Bill and Bowers, John. Annotated portfolios. In *Interactions*, volume 19, pages 40–49. ACM, 2012.
- Gaver, William. What should we expect from research through design? In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 937–946. ACM, 2012.
- Gittins, John; Glazebrook, Kevin, and Weber, Richard. *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.
- González-Rubio, Jesús; Ortiz-Martínez, Daniel, and Casacuberta, Francisco. Balancing User Effort and Translation Error in Interactive Machine translation via confidence measures. In *Proceedings of the ACL 2010 Conference Short Papers, Uppsala, Sweden*, volume 173, pages 173–177. ACL, 2010.
- Gorinova, Maria I; Sarkar, Advait; Blackwell, Alan F, and Syme, Don. A live, multiple-representation probabilistic programming environment for novices. *Proceedings of the 34th Annual ACM Conference on Human Factors in Computing Systems (CHI 2016)*, 2016.

- Green, Thomas R. G. and Petre, Marian. Usability analysis of visual programming environments: a ‘cognitive dimensions’ framework. *Journal of Visual Languages & Computing*, 7(2):131–174, 1996.
- Groce, Alex; Kulesza, Todd; Zhang, Chaoqiang; Shamasunder, Shalini; Burnett, Margaret; Wong, Weng-Keen; Stumpf, Simone; Das, Shubhomoy; Shinsel, Amber; Bice, Forrest, and McIntosh, Kevin. You Are the Only Possible Oracle: Effective Test Selection for End Users of Interactive Machine Learning Systems. *IEEE Transactions on Software Engineering*, 40(3):307–323, 2014. ISSN 0098-5589. doi: 10.1109/TSE.2013.59.
- Gulwani, Sumit. Automating string processing in spreadsheets using input-output examples. In *ACM SIGPLAN Notices*, volume 46, pages 317–330. ACM, 2011.
- Gutin, Gregory; Yeo, Anders, and Zverovich, Alexey. Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the tsp. *Discrete Applied Mathematics*, 117(1):81–86, 2002.
- Haitsma, Jaap and Kalker, Ton. A highly robust audio fingerprinting system. In *ISMIR*, volume 2002, pages 107–115, 2002.
- Hall, Mark; Frank, Eibe; Holmes, Geoffrey; Pfahringer, Bernhard; Reutemann, Peter, and Witten, Ian H. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.
- Hannafin, Michael J; Hannafin, Kathleen M; Land, Susan M, and Oliver, Kevin. Grounded practice and the design of constructivist learning environments. *Educational Technology Research and Development*, 45(3):101–117, 1997.
- Hao, MC; Dayal, Umeshwar; Keim, Daniel A, and Schreck, Tobias. Multi-resolution techniques for visual exploration of large time-series data. *Eurographics / IEEE VGTC Symposium on Visualization (EuroVis)*, pages 27–34, 2007a.
- Hao, Ming C.; Dayal, Umeshwar; Keim, Daniel A.; Morent, Dominik, and Schneidewind, Joern. Intelligent visual analytics queries. In *IEEE Symposium on Visual Analytics Science and Technology 2007*, pages 91–98. IEEE, 2007b. doi: 10.1109/VAST.2007.4389001.
- Hao, Ming C; Dayal, Umeshwar; Keim, Daniel A; Morent, Dominik, and Schneidewind, Joern. Intelligent visual analytics queries. In *Visual Analytics Science and Technology (VAST), 2007 IEEE Symposium on*, pages 91–98. IEEE, 2007c.
- Hao, Ming C; Marwah, Manish; Janetzko, Halldór; Dayal, Umeshwar; Keim, Daniel A; Patnaik, Debprakash; Ramakrishnan, Naren, and Sharma, Ratnesh K. Visual exploration of frequent patterns in multivariate time series. *Information Visualization*, 11(1):71–83, 2012.
- Heer, Jeffrey and Agrawala, Maneesh. Design considerations for collaborative visual analytics. *Information visualization*, 7(1):49–62, 2008.
- Heer, Jeffrey; Mackinlay, Jock D; Stolte, Chris, and Agrawala, Maneesh. Graphical histories for visualization: Supporting analysis, communication, and evaluation. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1189–1196, 2008.
- Hellinger, Ernst. Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen. *Journal für die reine und angewandte Mathematik*, 136:210–271, 1909.
- Herbrich, Ralf; Minka, Tom, and Graepel, Thore. TrueSkill™: A Bayesian Skill Rating System. In Schölkopf, B.; Platt, J.C., and Hoffman, T., editors, *Advances in Neural Information Processing Systems 19*, pages 569–576. MIT Press, 2007.

- Herlocker, Jonathan L; Konstan, Joseph A, and Riedl, John. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 241–250. ACM, 2000.
- Honebein, Peter C. Seven goals for the design of constructivist learning environments. In *Constructivist learning environments: Case studies in instructional design*, pages 11–24. Educational Technology Publications, Inc., 1996.
- Honebein, Peter C; Duffy, Thomas M, and Fishman, Barry J. Constructivism and the design of learning environments: Context and authentic activities for learning. In *Designing environments for constructive learning*, pages 87–108. Springer, 1993.
- Horvitz, Eric. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems the CHI is the limit - CHI '99*, number May, pages 159–166, New York, New York, USA, 1999. ACM Press. ISBN 0201485591. doi: 10.1145/302979.303030.
- Jonassen, David H. Designing constructivist learning environments. *Instructional design theories and models: A new paradigm of instructional theory*, 2:215–239, 1999.
- Jonassen, David H and Rohrer-Murphy, Lucia. Activity theory as a framework for designing constructivist learning environments. *Educational Technology Research and Development*, 47(1):61–79, 1999.
- Kalkanis, G. The application of confidence interval error analysis to the design of decision tree classifiers. *Pattern Recognition Letters*, 14(5):355–361, 1993.
- Keim, Daniel; Andrienko, Gennady; Fekete, Jean-Daniel; Görg, Carsten; Kohlhammer, Jörn, and Melançon, Guy. *Visual analytics: Definition, process, and challenges*. Springer, 2008.
- Keim, Daniel A. Visual exploration of large data sets. *Communications of the ACM*, 44(8): 38–44, 2001.
- Keim, Daniel A; Bak, Peter; Bertini, Enrico; Oelke, Daniela; Spretke, David, and Ziegler, Hartmut. Advanced visual analytics interfaces. *Proceedings of the International Conference on Advanced Visual Interfaces - AVI '10*, page 3, 2010. doi: 10.1145/1842993.1842995.
- Kietz, Jörg-Uwe; Serban, Floarea; Bernstein, Abraham, and Fischer, Simon. Towards cooperative planning of data mining workflows. In *Proceedings of the Third Generation Data Mining Workshop at the 2009 European Conference on Machine Learning (ECML 2009)*, pages 1–12, 2009.
- Kincaid, Robert and Lam, Heidi. Line graph explorer: scalable display of line graphs using Focus+Context. In *Proceedings of the working conference on Advanced visual interfaces - AVI '06*, page 404, New York, New York, USA, 2006. ACM Press. ISBN 1595933530. doi: 10.1145/1133265.1133348.
- Kirschner, Paul A; Sweller, John, and Clark, Richard E. Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational psychologist*, 41 (2):75–86, 2006.
- Ko, Andrew J. *Asking and answering questions about the causes of software behavior*. PhD thesis, Carnegie Mellon University, 2008.

- Ko, Andrew J and Myers, Brad A. Designing the whyline: a debugging interface for asking questions about program behavior. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 151–158. ACM, 2004.
- Ko, Andrew J.; Myers, Brad A., and Aung, HH. Six Learning Barriers in End-User Programming Systems. In *2004 IEEE Symposium on Visual Languages - Human Centric Computing*, pages 199–206. IEEE, 2004. ISBN 0-7803-8696-5. doi: 10.1109/VLHCC.2004.47.
- Ko, Andrew J; Abraham, Robin; Beckwith, Laura; Blackwell, Alan; Burnett, Margaret; Erwig, Martin; Scaffidi, Chris; Lawrance, Joseph; Lieberman, Henry; Myers, Brad, and others, . The state of the art in end-user software engineering. *ACM Computing Surveys (CSUR)*, 43(3):21, 2011.
- Kontschieder, Peter; Dorn, Jonas F; Morrison, Cecily; Corish, Robert; Zikic, Darko; Sellen, Abigail; D’Souza, Marcus; Kamm, Christian P; Burggraaff, Jessica; Tewarie, Prejaas, and others, . Quantifying progression of multiple sclerosis via classification of depth videos. In *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2014*, pages 429–437. Springer, 2014.
- Konukoglu, Ender; Glocker, Ben; Zikic, Darko, and Criminisi, Antonio. Neighbourhood approximation using randomized forests. *Medical image analysis*, 17(7):790–804, 2013.
- Kruskal, Joseph B and Landwehr, James M. Icicle plots: Better displays for hierarchical clustering. *The American Statistician*, 37(2):162–168, 1983.
- Kulesza, Todd; Wong, Weng-Keen; Stumpf, Simone; Perona, Stephen; White, Rachel; Burnett, Margaret M; Oberst, Ian, and Ko, Andrew J. Fixing the program my computer learned: Barriers for end users, challenges for the machine. In *Proceedings of the 14th international conference on Intelligent user interfaces*, pages 187–196. ACM, 2009.
- Kulesza, Todd; Stumpf, Simone; Wong, Weng-Keen; Burnett, Margaret M; Perona, Stephen; Ko, Andrew, and Oberst, Ian. Why-oriented end-user debugging of naive bayes text classification. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 1(1):2, 2011.
- Kulesza, Todd; Stumpf, Simone; Burnett, Margaret; Yang, Sherry; Kwan, Irwin, and Wong, Weng-Keen. Too much, too little, or just right? Ways explanations impact end users’ mental models. In *Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC*, pages 3–10. IEEE, 2013. doi: 10.1109/VLHCC.2013.6645235.
- Kulesza, Todd; Amershi, Saleema; Caruana, Rich; Fisher, Danyel, and Charles, Denis. Structured labeling for facilitating concept evolution in machine learning. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 3075–3084. ACM, 2014. doi: 10.1145/2556288.2557238.
- Kulesza, Todd; Burnett, Margaret; Wong, Weng-Keen, and Stumpf, Simone. Principles of Explanatory Debugging to Personalize Interactive Machine Learning. In *Proceedings of the 20th International Conference on Intelligent User Interfaces (IUI)*, pages 126–137. ACM, 2015. doi: 10.1145/2678025.2701399.
- Kurlander, David and Feiner, Steven. Editable graphical histories. In *IEEE Workshop on Visual Languages*, pages 127–134. Citeseer, 1988.
- Laird, Nan M and Louis, Thomas A. Empirical bayes confidence intervals based on bootstrap samples. *Journal of the American Statistical Association*, 82(399):739–750, 1987.

- Lebow, David. Constructivist values for instructional systems design: Five principles toward a new mindset. *Educational technology research and development*, 41(3):4–16, 1993.
- Lehmann, Dirk J; Kemmler, Fritz; Zhyhalava, Tatsiana; Kirschke, Marco, and Theisel, Holger. Visualnostics: Visual guidance pictograms for analyzing projections of high-dimensional data. In *Computer Graphics Forum*, volume 34, pages 291–300. Wiley Online Library, 2015.
- Lenat, Douglas B. AM: An artificial intelligence approach to discovery in mathematics as heuristic search. Technical report, DTIC Document, 1976.
- Lex, Alexander; Streit, Mac; Kruijff, Ernst, and Schmalstieg, Dieter. Caleydo: Design and evaluation of a visual analysis framework for gene expression data in its biological context. *Pacific Visualization Symposium (PacificVis), 2010 IEEE*, pages 57–64, 2010. doi: 10.1109/PACIFICVIS.2010.5429609.
- Liao, T Warren. Clustering of time series data – a survey. *Pattern recognition*, 38(11): 1857–1874, 2005.
- Lim, Brian Y. and Dey, Anind K. Assessing Demand for Intelligibility in Context-aware Applications. In *Proceedings of the 11th International Conference on Ubiquitous Computing, UbiComp '09*, pages 195–204, New York, NY, USA, 2009. ACM. doi: 10.1145/1620545.1620576.
- Lim, Brian Y.; Dey, Anind K., and Avrahami, Daniel. Why and why not explanations improve the intelligibility of context-aware intelligent systems. In *Proceedings of the 27th international conference on human factors in computing systems (CHI)*, pages 2119–2129. ACM, 2009. doi: 10.1145/1518701.1519023.
- Lin, Jessica; Keogh, Eamonn, and Lonardi, Stefano. Visualizing and discovering non-trivial patterns in large time series databases. In *Information Visualization*, volume 4, pages 61–82. SAGE Publications, 2005.
- Liu, Zhicheng and Heer, Jeffrey. The effects of interactive latency on exploratory visual analysis. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):2122–2131, 2014.
- Livingston, G.R.; Rosenberg, J.M., and Buchanan, B.G. Closing the loop: heuristics for autonomous discovery. In *Proceedings 2001 IEEE International Conference on Data Mining*, pages 393–400. IEEE, 2001. doi: 10.1109/ICDM.2001.989544.
- Lloyd, James Robert; Duvenaud, David; Grosse, Roger; Tenenbaum, Joshua B., and Ghahramani, Zoubin. Automatic construction and natural-language description of non-parametric regression models. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, AAAI'14*, pages 1242–1250. AAAI Press, 2014.
- Lou, Yin; Caruana, Rich, and Gehrke, Johannes. Intelligible models for classification and regression. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–158. ACM, 2012.
- Lou, Yin; Caruana, Rich; Gehrke, Johannes, and Hooker, Giles. Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–631. ACM, 2013.
- Löwgren, Jonas. Annotated portfolios and other forms of intermediate-level knowledge. *interactions*, 20(1):30–34, 2013.

- MacEachren, Alan M; Jaiswal, Anuj; Robinson, Anthony C; Pezanowski, Scott; Savelyev, Alexander; Mitra, Prasenjit; Zhang, Xiao, and Blanford, Justine. Senseplace2: Geotwitter analytics support for situational awareness. In *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on*, pages 181–190. IEEE, 2011.
- MacKay, David JC. A practical bayesian framework for backpropagation networks. In *Neural computation*, volume 4 (3), pages 448–472. MIT Press, 1992.
- MacKay, David JC. Gaussian processes – a replacement for supervised neural networks? (Unpublished lecture notes) University of Cambridge, 1997.
- Mamani, Gladys MH; Fatore, Francisco M; Nonato, Luis Gustavo, and Paulovich, Fernando Vieira. User-driven feature space transformation. In *Computer Graphics Forum*, volume 32, pages 291–299. Wiley Online Library, 2013.
- Mansmann, Florian; Spretke, David; Janetzko, Halldor; Kranstauber, Bart, and Safi, Kamran. *Correlation-based Arrangement of Time Series for Movement Analysis in Behavioural Ecology*. Bibliothek der Universität Konstanz, 2012.
- Massey Jr, Frank J. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78, 1951.
- McSherry, David. Explanation in recommender systems. *Artificial Intelligence Review*, 24 (2):179–197, 2005.
- Mittelstädt, Sebastian. *Methods for Effective Color Encoding and the Compensation of Contrast Effects*. PhD thesis, Universität Konstanz, Konstanz, Germany, 2015.
- Mühlenbein, Heinz; Gorges-Schleuter, Martina, and Krämer, Ottmar. Evolution algorithms in combinatorial optimization. *Parallel Computing*, 7(1):65–85, 1988.
- Müller, Wolfgang and Schumann, Heidrun. Visualization methods for time-dependent data-an overview. In *Proceedings of the 2003 Winter Simulation Conference.*, volume 1, pages 737–745. IEEE, 2003.
- Neal, Radford M. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 1996.
- Neurath, Otto; Eve, Matthew, and Burke, Christopher. *From hieroglyphics to Isotype: a visual autobiography*. Hyphen Press, 2010.
- Nguyen, Anh; Yosinski, Jason, and Clune, Jeff. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Computer Vision and Pattern Recognition (CVPR '15), IEEE*, pages 427–436. IEEE, 2015.
- Norman, Donald A. Some observations on mental models. In *Mental models*, volume 7, pages 7–14. Lawrence Erlbaum Associates, Inc., 1983.
- Pandey, Anshul Vikram; Krause, Josua; Felix, Cristian; Boy, Jeremy, and Bertini, Enrico. Towards understanding human similarity perception in the analysis of large sets of scatter plots. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 3659–3669. ACM, 2016.
- Panov, Panče; Dzeroski, Sasso, and Soldatova, Larisa N. Ontodm: An ontology of data mining. In *Data Mining Workshops, 2008. ICDMW'08. IEEE International Conference on*, pages 752–760. IEEE, 2008.

- Pellegrino, JW; Hickey, D; Heath, A; Rewey, K; Vye, NJ, and Vanderbilt, CGTV. Assessing the outcomes of an innovative instructional program: The 1990-1991 implementation of the "adventures of jasper woodbury.". *Nashville, TN: Learning Technology Center, Vanderbilt University*, 1992.
- Perin, Charles; Dragicevic, Pierre, and Fekete, Jean-Daniel. Revisiting bertin matrices: New interactions for crafting tabular visualizations. In *Visualization and Computer Graphics, IEEE Transactions on*, volume 20 (12), pages 2082–2091. IEEE, 2014. doi: 10.1109/TVCG.2014.2346279.
- Pirolli, Peter and Card, Stuart. Information foraging. *Psychological review*, 106(4):643, 1999.
- Pirolli, Peter and Card, Stuart K. The Sensemaking Process and Leverage Points for Analyst Technology as Identified Through Cognitive Task Analysis. *Proceedings of International Conference on Intelligence Analysis*, 5:2–4, 2005.
- Pirolli, Peter; Schank, Patricia; Hearst, Marti, and Diehl, Christine. Scatter/gather browsing communicates the topic structure of a very large text collection. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 213–220. ACM, 1996.
- Pu, Pearl and Chen, Li. Trust building with explanation interfaces. In *Proceedings of the 11th international conference on Intelligent user interfaces*, pages 93–100. ACM, 2006.
- Quinlan, J. Ross. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.
- Quinlan, J Ross. *C4.5: programs for machine learning*. Elsevier, 2014.
- Rao, Ramana and Card, Stuart K. The table lens. In *Proceedings of the SIGCHI conference on Human factors in computing systems celebrating interdependence - CHI '94*, pages 318–322, New York, New York, USA, 1994. ACM Press. ISBN 0897916506. doi: 10.1145/191666.191776.
- Resnick, Mitchel; Maloney, John; Monroy-Hernández, Andrés; Rusk, Natalie; Eastmond, Evelyn; Brennan, Karen; Millner, Amon; Rosenbaum, Eric; Silver, Jay; Silverman, Brian, and others, . Scratch: programming for all. *Communications of the ACM*, 52 (11):60–67, 2009.
- Robertson, Philip K. Visualizing color gamuts: A user interface for the effective use of perceptual color spaces in data displays. *Computer Graphics and Applications, IEEE*, 8 (5):50–64, 1988.
- Rothermel, Gregg; Li, Lixin; DuPuis, Christopher, and Burnett, Margaret. What You See is What You Test: A Methodology for Testing Form-based Visual Programs. In *Proceedings of the 20th International Conference on Software Engineering, ICSE '98*, pages 198–207, Washington, DC, USA, 1998. IEEE Computer Society.
- Sarkar, Advait; Blackwell, Alan F; Jamnik, Mateja, and Spott, Martin. Hunches and sketches: rapid interactive exploration of large datasets through approximate visualisations. In *Proceedings of the 8th International Conference on the Theory and Application of Diagrams, Graduate Symposium (DIAGRAMS 2014)*, pages 18–22, July 2014a.
- Sarkar, Advait; Blackwell, Alan F; Jamnik, Mateja, and Spott, Martin. Teach and try: A simple interaction technique for exploratory data modelling by end users. In *Visual Languages and Human-Centric Computing (VL/HCC), 2014 IEEE Symposium on*, pages 53–56. IEEE, 2014b. doi: 10.1109/VLHCC.2014.6883022.

- Sarkar, Advait; Blackwell, Alan F.; Jamnik, Mateja, and Spott, Martin. Interaction with uncertainty in visualisations. In Bertini, E.; Kennedy, J., and Puppo, E., editors, *Eurographics/IEEE VGTC Conference on Visualization (EuroVis 2015)*. The Eurographics Association, 2015. doi: 10.2312/eurovisshort.20151138.
- Sarkar, Advait; Morrison, Cecily; Dorn, Jonas F.; Bedi, Rishi; Steinheimer, Saskia; Boisvert, Jacques; Burggraaff, Jessica; D'Souza, Marcus; Kontschieder, Peter; Bulò, Samuel Rota; Walsh, Lorcan; Kamm, Christian P.; Zaykov, Yordan; Sellen, Abigail, and Lindley, Siân E. Setwise Comparison: Consistent, Scalable, Continuum Labels for Computer Vision. In *Proceedings of the 34th annual ACM conference on Human factors in computing systems - CHI '16*, pages 261–271. ACM Press, 2016. doi: 10.1145/2858036.2858199.
- Savitha, Ramaswamy; Suresh, Sundaram, and Sundararajan, Narasimhan. Metacognitive learning in a fully complex-valued radial basis function neural network. *Neural Computation*, 24(5):1297–1328, 2012.
- Schölkopf, Bernhard; Smola, Alexander, and Müller, Klaus-Robert. Kernel principal component analysis. In *International Conference on Artificial Neural Networks*, pages 583–588. Springer, 1997.
- Schulz, Hans-Jorg. Treevis.net: A tree visualization reference. *IEEE Computer Graphics and Applications*, 31(6):11–15, Nov 2011. doi: 10.1109/MCG.2011.103.
- Schulz, Hans-Jorg; Hadlak, Steffen, and Schumann, Heidrun. The Design Space of Implicit Hierarchy Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics*, 17(4):393–411, 2011. doi: 10.1109/TVCG.2010.79.
- Sedlmair, Michael; Heinzl, Christoph; Bruckner, Stefan; Piringer, Harald, and Möller, Torsten. Visual parameter space analysis: A conceptual framework. In *Visualization and Computer Graphics, IEEE Transactions on*, volume 20 (12), pages 2161–2170. IEEE, 2014. doi: 10.1109/TVCG.2014.2346321.
- Serban, Floarea; Vanschoren, Joaquin; Kietz, Jörg-Uwe, and Bernstein, Abraham. A survey of intelligent assistants for data analysis. *ACM Computing Surveys*, 45(3):1–35, June 2013. doi: 10.1145/2480741.2480748.
- Settles, Burr. Active learning literature survey. *University of Wisconsin, Madison*, 52 (55-66):11, 2010.
- Shneiderman, Ben. The eyes have it: A task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pages 336–343. IEEE, 1996.
- Shneiderman, Ben and Wattenberg, Martin. Ordered treemap layouts. In *IEEE Symposium on Information Visualization*, pages 73–78. IEEE, 2001. doi: 10.1109/INFVIS.2001.963283.
- Smith, Stephen J.; Bourgoïn, Mario O.; Sims, Karl, and Voorhees, Harry L. Handwritten character classification using nearest neighbor in large databases. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(9):915–919, 1994.
- St Amant, Robert and Cohen, Paul R. Interaction with a mixed-initiative system for exploratory data analysis. In *Proceedings of the 2nd international conference on Intelligent user interfaces*, pages 15–22. ACM, 1997.

- Stead, Alistair and Blackwell, Alan F. Learning Syntax as Notational Expertise when using DrawBridge. In *Psychology of Programming Interest Group Annual Conference 2014*, pages 41–52, 2014.
- Stolper, Charles D; Perer, Adam, and Gotz, David. Progressive visual analytics: User-driven visual exploration of in-progress analytics. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):1653–1662, 2014.
- Stolterman, Erik. The nature of design practice and implications for interaction design research. *International Journal of Design*, 2(1), 2008.
- Sugiyama, Kozo; Tagawa, Shojiro, and Toda, Mitsuhiko. Methods for visual understanding of hierarchical system structures. *Systems, Man and Cybernetics, IEEE Transactions on*, 11(2):109–125, 1981.
- Szafir, Danielle Nicole Albers. *Improving Color for Data Visualization*. PhD thesis, University of Wisconsin-Madison, 2015.
- Szegedy, Christian; Zaremba, Wojciech; Sutskever, Ilya; Bruna, Joan; Erhan, Dumitru; Goodfellow, Ian, and Fergus, Rob. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.
- Tanimoto, Steven L. Viva: A visual language for image processing. *Journal of Visual Languages & Computing*, 1(2):127–139, 1990.
- Taylor, Peter C; Fraser, Barry J, and Fisher, Darrell L. Monitoring constructivist classroom learning environments. *International journal of educational research*, 27(4):293–302, 1997.
- Tintarev, Nava and Masthoff, Judith. A survey of explanations in recommender systems. In *Data Engineering Workshop, 2007 IEEE 23rd International Conference on*, pages 801–810. IEEE, 2007.
- Valentin, Julien; Vineet, Vibhav; Cheng, Ming-Ming; Kim, David; Shotton, Jamie; Kohli, Pushmeet; Nießner, Matthias; Criminisi, Antonio; Izadi, Shahram, and Torr, Philip. Semanticpaint: Interactive 3d labeling and learning at your fingertips. *ACM Transactions on Graphics (TOG)*, 34(5):154, 2015.
- Van der Maaten, Laurens and Hinton, Geoffrey. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.
- Vygotsky, Lev S. *Mind in society: The development of higher psychological processes*. Harvard University Press, 1987.
- Wacharamanotham, Chat; Subramanian, Krishna; Völkel, Sarah Theres, and Borchers, Jan. Statsplorer: Guiding novices in statistical analysis. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 2693–2702. ACM, 2015.
- Wadsworth, Barry J. *Piaget's theory of cognitive and affective development: Foundations of constructivism*. Longman Publishing, 1996.
- Ward, David J; Blackwell, Alan F, and MacKay, David JC. Dasher? a data entry interface using continuous gestures and language models. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 129–137. ACM, 2000.
- Watkins, Christopher John Cornish Hellaby. *Learning from delayed rewards*. PhD thesis, University of Cambridge England, 1989.

- Weintraub, Mitch; Beaufays, Françoise; Rivlin, Ze'ev; König, Yochai, and Stolcke, Andreas. Neural-network based measures of confidence for word recognition. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, volume 2, pages 887–887. IEEE Computer Society, 1997.
- Wilson, Aaron; Burnett, Margaret; Beckwith, Laura; Granatir, Orion; Casburn, Leah; Cook, Curtis; Durham, Mike, and Rothermel, Gregg. Harnessing curiosity to increase correctness in end-user programming. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 305–312. ACM, 2003.
- Wirth, Rüdiger; Shearer, Colin; Grimmer, Udo; Reinartz, Thomas; Schlosser, Jorg; Breiter, Christoph; Engels, Robert, and Lindner, Guido. Towards process-oriented tool support for knowledge discovery in databases. *First European Symposium, PKDD '97 Trondheim, Norway, June 24–27, 1997 Proceedings*, pages 243–253, 1997. doi: 10.1007/3-540-63223-9_123.
- Wu, Eugene and Madden, Samuel. Scorpion: Explaining Away Outliers in Aggregate Queries. volume 6 (8), pages 553–564. VLDB Endowment, 2013. doi: 10.14778/2536354.2536356.
- Yoon, YoungSeok; Myers, Brad A, and Koo, Sebon. Visualization of fine-grained code change history. In *Visual Languages and Human-Centric Computing (VL/HCC), 2013 IEEE Symposium on*, pages 119–126. IEEE, 2013.
- Zeiler, Matthew D and Fergus, Rob. Visualizing and understanding convolutional networks. In *Computer vision—ECCV 2014*, pages 818–833. Springer, 2014.
- Zhang, Li and Luh, Peter B. Neural network-based market clearing price prediction and confidence interval estimation with an improved extended kalman filter method. *Power Systems, IEEE Transactions on*, 20(1):59–66, 2005.
- Zhao, Zhenpeng; Benjamin, William; Elmqvist, Niklas, and Ramani, K. Sketcholution: Interaction Histories for Sketching. In *International Journal of Human-Computer Studies*, pages 11–20. Elsevier, 2015.
- Zimmerman, John; Forlizzi, Jodi, and Evenson, Shelley. Research through design as a method for interaction design research in hci. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 493–502. ACM, 2007.