The AAAI 2017 Spring Symposium on
Artificial Intelligence for the Social Good
Technical Report SS-17-01

Machine Representation of Data Analyses: Towards a Platform for Collaborative Data Science

Evan Patterson
Department of Statistics
Stanford University

Ioana Baldini
Aleksandra Mojsilović
Kush R. Varshney
IBM T. J. Watson Research Center

Abstract

Artificial intelligence and data science play an increasingly important role in solving today's scientific and social challenges. To be successful, the data-driven approach to social good requires effective collaboration between data scientists, subject-matter experts, policymakers, and other stakeholders. We envision a cloud platform for data science that would facilitate collaboration between stakeholders and possess AI capabilities for discovering, benchmarking, and organizing data analyses. Here we present a foundational technology motivated by this vision. Our system automatically extracts a high-level dataflow graph from a data analysis. The graph describes how data flows through an analysis pipeline, including which statistical methods are used and how they fit together. The system requires no special annotations from the data analyst and consumes analyses written in Python using standard tools, such as Scikit-learn and StatsModels. In this paper, we explain how our system works and how it fits into our larger vision for a collaborative data science platform.

Introduction

Artificial intelligence (AI) and data science play an increasingly important role in solving today's scientific and social challenges. The application of AI techniques to social good is promising and relatively unexplored, but the sociotechnical nature of this domain presents special challenges. Research and development cannot be conducted within a closed, highly technical universe of AI specialists and data scientists. Collaboration with subject-matter experts, policymakers, scientists, engineers, and other stakeholders is necessary to make progress on pressing social problems such as global poverty, hunger, disease, and climate change (Kapoor et al. 2015). Within the collaborative process of problem identification, research, discovery, and implementation, the natural and social sciences provide the core body of facts and theory upon which to base decisions. It is therefore essential that stakeholders exchange knowledge with the relevant scientific communities.

The search for a cure to multiple sclerosis (MS) illustrates the challenges of data-driven social good under the status quo. Despite many decades of research, our understanding of the etiology (causes) of MS remains extremely limited. The evidence suggests that MS is an etiologically heterogeneous and multifactorial disease, i.e., MS is not caused by any single factor, and the set of factors that interact to cause it varies between individuals. To understand these complex interactions, researchers must analyze patient data from diverse sources. To that end, the Accelerated Cure Project for Multiple Sclerosis (ACP) has curated a large-scale repository of MS patient data, including clinical records, patient self-reports, and biomarker data. ACP makes its physical biosamples and analytical data available to MS researchers on an open-access basis to generate new insights about MS.

The efficiency of this collaborative process is limited by our existing collaboration and knowledge sharing tools. It is difficult for ACP to organize and synthesize all the analyses conducted on its data resources. Individual MS researchers consuming the ACP data must understand how their work fits into the existing body of data analyses. Any new insights arising from these analyses must be communicated to clinicians, policymakers, donors, and other stakeholders. At present, the tools to facilitate these activities are either nonexistent or highly fragmented.

Of course, the problem is not confined to MS research; it is relevant to any social good enterprise involving complex, data-driven questions about the natural or social sciences. Many of the Sustainable Development Goals (United Nations 2015), which set a global agenda for social good, have this character. Examples include sustaining high economic growth (Goal 8) and combating climate change (Goal 13).

As a step towards fostering knowledge sharing and collaboration, we envision a cloud platform for collaborative data science: a single space for domain experts, data scientists, and other stakeholders to share datasets and data analyses. It should be equipped with AI capabilities for discovering, benchmarking, and organizing data analyses, as well as suggesting relevant datasets or potential collaborators. To enable such features, the platform must possess a high-quality, machine-interpretable representation of its contents, particularly its data analyses. In this work, we propose and implement a foundational technology for automatically extracting a machine representation of a data analysis. The representation is a dataflow graph capturing the most important steps of the analysis.

The idea that artificial intelligence will help create and organize scientific knowledge is increasingly entering the

popular consciousness and the research agendas of scientists and AI specialists. In a recent book, Michael Nielsen dreams of a "data web" enabling "a world where all scientific knowledge has been made available online, and is expressed in a way that can be understood by computers" (Nielsen 2012). The extent to which this vision has been realized varies greatly across disciplines. Perhaps the greatest strides have been made in biology, where knowledge bases like the Gene Ontology (Ashburner et al. 2000) and BioPortal (Noy et al. 2009) enjoy great success. By contrast, the machine representation of data analysis—the focus of our work—has received relatively little attention, despite its importance for science and data-driven social good.

This paper is an overview of our technical work in its broader context. Future publications will single out particular components of our system for more rigorous treatment. The paper is organized as follows. In the next section, we further motivate our vision for a collaborative data science platform, contrasting it with existing platforms. The two subsequent sections describe the basic functionality and architecture of our system. In the penultimate section, we survey existing work related to the dataflow representation of data analyses. To conclude, we situate our work within some broader trends in science and social good.

A new kind of data science platform?

Our thesis is that a cloud platform for collaborative data science could lead to dramatic efficiency gains for the Accelerated Cure Project and for the larger scientific and social good communities. A skeptical reader may object that such platforms already exist, with perhaps the further complaint that they have failed to deliver on their promise. In response, we propose two desiderata for a collaboration platform, which we believe to be necessary for significant efficiency gains but which are not, to our knowledge, satisfied by any existing offering. First, the content hosted by the platform should not be restricted to datasets or error metrics, but should include what is probably of greatest scientific value, the data analyses themselves. Second, the content should be represented in a machine-interpretable form.

Since these desiderata are not universally recognized, we offer a few points in their defense. Why is it essential to host complete data analyses, rather than summary statistics like the prediction error on held-out data? Under the *challenge* model of data science, organizers define a prediction problem and teams throughout the world compete to achieve the best error rate. The paradigmatic platform in this class is Kaggle, with variations offered by Driven Data (Bull, Slavitt, and Lipstein 2016), Dream Challenges (Stolovitzky, Monroe, and Califano 2007; Marbach et al. 2012), and OpenML (Vanschoren et al. 2014). Naturally, the challenge model is well-suited to questions that can be crisply formulated as supervised learning problems. Such questions occur infrequently in social good and in science. More often, research questions are open-ended, multifaceted, and irreducible to a single real number captured by a loss function. On this view, data analysis is an open-ended process that may involve (informal) exploratory data analysis and (formal) statistical inference beyond prediction. A successful platform for collaborative data science will capture these activities. Consequently it must publish complete data analyses in addition to any error metrics deemed appropriate.

Machine interpretability is important because it allows computers to interact with the platform's content. In the data science domain, computer interactions could range from mechanical processing, such as querying the corpus of analyses or benchmarking statistical models, to sophisticated AI capabilities, such as predicting fruitful collaborations or summarizing a corpus of data analyses. There are several strategies for creating machine-interpretable content. Traditionally, human users provide it directly to the system in a highly structured format for knowledge representation—often a tedious chore. Our strategy is to automatically extract a useful representation from the computer code implementing a data analysis. A complementary approach, not pursued here, would use natural language processing (NLP) to extract meaning from the human text in the data analysis. (In the increasingly popular notebook formats, such as Jupyter Notebook and R Markdown, the human text and computer code are placed side-by-side, suggesting a possible synergy between the two approaches.)

Among existing platforms, challenge platforms like Kaggle store error metrics for the uploaded models. These metrics are obviously machine interpretable and are utilized to create public leaderboards of the best models. But as we have seen, the challenge platforms do not meet the first desideratum. There is a different category of cloud platforms that treat data analyses as first-class citizens. These platforms are designed to make existing distributed computing frameworks, such as Hadoop and Spark, and data analysis environments, such as Jupyter Notebook and R Studio, conveniently available on the cloud. Some examplars of this rapidly growing space are Domino Data Lab, IBM Data Science Experience, and Microsoft Azure Machine Learning. These platforms do not form useful machine representations of their content. We are not aware of any existing platform that satisfies both our desiderata.

Having motivated our hypothetical platform in the abstract, we now consider some specific features that could be enabled by a high-quality, machine-interpretable representation of a data analysis. An obvious first application is a sophisticated query engine. On a platform using our system, users could ask queries like "Find all analyses that take dataset D as input, perform clustering analysis, and output three clusters" or "Find all plots of variable X against variable Y (drawn from dataset D)". It is impossible to make such precise queries using existing indexes like Google Scholar.

More ambitiously, the machine representations could serve as input to AI algorithms that operate on data analyses. For instance, we envision a recommender system that automatically identifies relevant data analyses or potential collaborators on the basis of shared datasets, methodology, and social connections. Likewise, we could try to identify analyses with novel (but fruitful) methodology, a form of outlier detection. One might even try to develop an automated, personalized system for organizing and summarizing a body of

scientific work. At present, such surveys must be conducted at great cost by human subject-matter experts.

All these features operate on existing data analyses created by human scientists. In another frontier of artificial intelligence, called *computational creativity* (Colton, López de Mantaras, and Stock 2009; Colton and Wiggins 2012), machines play an active role in the creation of new content. The platform could surgically modify existing analyses, replacing certain steps with semantically compatible ones (e.g., replacing k-means clustering with hierarchical clustering), or even generate entirely original analyses. The new analyses would be evaluated by some combination of novelty and quality metrics, where novelty is measured against existing analyses on the platform. We note that creative applications require a *bidirectional* representation: the high-level dataflow graph must be converted back into executable computer code. This capability, interesting in its own right, is not currently supported by our system.

Machine representation of data analyses

Briefly, our system automatically extracts a data-flow representation of a data analysis, which is interpretable by both humans and machines.

In this section, we will explain what that sentence means, but first let's consider an example. Figure 1 shows the dataflow graph extracted from an exploratory data analysis of the Accelerated Cure Project's survey data on MS patients. The objective of the analysis is to understand how the symptoms of MS are distributed across the population of patients and how they are related to the four clinically recognized MS disease types. To provide context for Figure 1 we summarize the steps of the analysis. The table of symptom indicator variables is loaded and studied using multiple correspondence analysis (MCA), a variant of principal components analysis (PCA) suitable for categorical data. K-means clustering with four clusters is then applied to the top four MCA factor scores. The resulting clusters are visually compared with the four disease types (loaded from a separate file) in a scatter plot of the top two factor scores. All this information is contained in the dataflow graph of Figure 1.

We should clarify our terminology. For our purposes, a *data analysis* is a computer program, in the form of either a source file or an interactive Jupyter notebook (Pérez and Granger 2007; Ragan-Kelley et al. 2014), that executes a sequence of data analysis tasks. For example, the program might perform a clustering analysis, fit a sparse linear regression model, or test a statistical null hypothesis. At present we require that the program be written in Python, but this limitation is not fundamental; we hope to add support for R and Julia in the near future.

The representation that we extract is a *dataflow graph* (or simply *flow graph*) summarizing the execution of the analysis. The flow graph is a directed acyclic graph (DAG). Ideally, it will capture the most important steps of the analysis, such as: reading a data file, fitting a statistical model, making predictions, computing error metrics and p-values, saving transformed data to a new file, and so on. In any realistic analysis, there will also be some steps to which the system

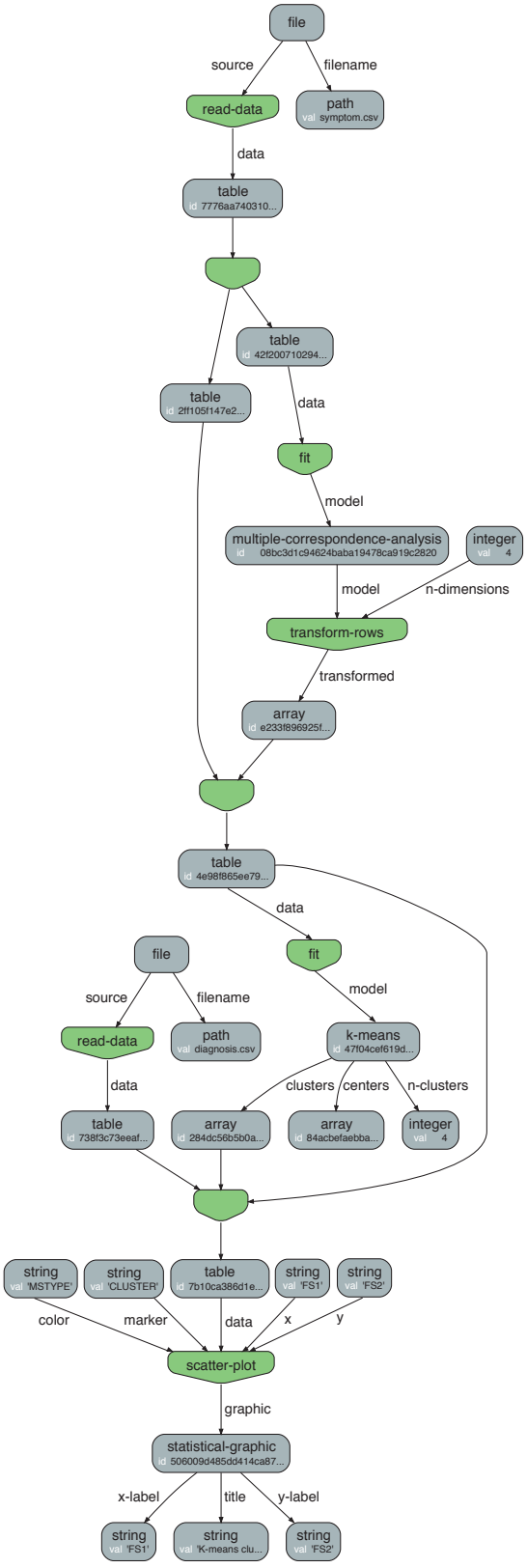

Figure 1: Example dataflow graph: exploratory data analysis for the Accelerated Cure Project

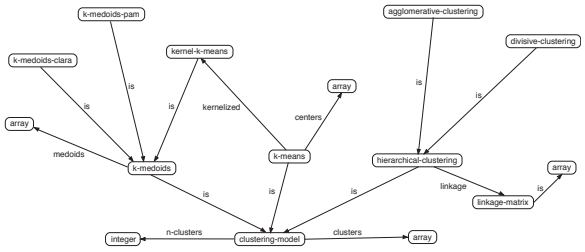

Figure 2: Excerpt of clustering methods in the knowledge base

cannot assign a label. For example, data munging is generally uninterpretable by our system. The flow graph therefore consists of a mixture of labeled and unlabeled steps.

The construction of the dataflow graph is *automatic* in the sense that it requires no additional annotations by the data analyst. Put differently, the only input to the system is the computer program constituting the data analysis. However, the system *does* rely on human annotations of another kind, namely for the underlying analysis tools. For example, if the analyst fits a support vector machine (SVM) using the Python library Scikit-learn (Pedregosa et al. 2011), then the system's annotation database must include entries for the SVM library functions. This approach to functional annotation leads to a large gain in human efficiency, since library annotations written by a single individual can be leveraged by the whole community. It also supports one of our most important design goals: to be minimally intrusive to the workflows of experienced data analysts.

Finally, we clarify the sense in which our dataflow graph is *interpretable*. To achieve our goal of producing a machine-interpretable representation of a data analysis, it is not enough to attach labels to steps in the analysis; the system must also understand how these labels are related to each other. For instance, suppose an analysis involves a logistic regression model, to which the system assigns the label "logistic regression". To put this step in context, the system must understand that logistic regression is a type of classification model, which is in turn a type of predictive model. Or, to take another example, suppose two analysts each perform a clustering analysis of the same dataset, one using k-means clustering and the other using hierarchical clustering. The system should be able to recognize two instances of clustering and compare them accordingly. To enable these features, our system aligns the dataflow graph with a *knowledge base* (or *ontology*) of data analysis concepts (Brachman and Levesque 2004; Spivak and Kent 2012). Figure 2 shows a selection of the clustering methods in our knowledge base. Notice that some concepts, such as `k-means` and `clusters`, also appear in the dataflow graph of Figure 1.

System architecture

In this section, we briefly explain the organization and operation of our system. An important methodological point is that our system uses dynamic analysis, not static analysis; that is, we *execute* the data analysis code rather than simply inspecting it. Our system is designed to extract information that is *only* available at runtime, such as the column names of data tables and the parameters of statistical models. At present our system makes no use of static analysis.

So far we have spoken of "the" dataflow graph, but there are actually *three* different dataflow graphs in our system, which we call the *raw flow graph*, the *annotated flow graph*, and the *semantic flow graph*. The semantic flow graph is the final output of our system and is exemplified in Figure 1 above. The other two graphs are intermediate forms.

The architecture of our system, including the relations between the three dataflow graphs, is shown in Figure 3. A *runtime environment* for data analysis, such as the Python interpreter, supplies a stream of trace events to the system. Trace events are emitted for each function call and matching function return. The system processes these events in an online fashion, building up the raw dataflow graph as the analysis executes. The raw graph is a directed graph whose vertices represent function calls and whose edges represent objects passed between them. The "raw" graph is so called because it is low-level, language-dependent, and generally uninterpretable. For a typical data analysis encountered in practice, which involves not just statistical modeling but also data cleaning and preparation, the raw graph will contain hundreds or thousands of nodes. This representation is too large and detailed to be readily interpretable by humans, yet too unstructured to be interpretable by machines.

The subsequent stages of the pipeline transform the raw graph into more useful representations. These stages use auxiliary information from an *annotation database* for statistical software and a *knowledge base* of universal statistical concepts. First, the annotated graph is constructed from the raw graph by attaching annotations to functions and objects that have entries in the annotation database. In addition, subgraphs of unannotated vertices are collapsed into single vertices. The latter transformation tends to greatly reduce the size of the graph, as most function calls are unannotated. In the final stage, the semantic graph is created from the annotated graph by using information in the annotations to identify specific language and library constructs with universal concepts from the knowledge base. Unlike the raw and annotated graphs, the semantic flow graph is independent of the particular language (such as Python or R) and libraries (such as Scikit-learn or StatsModels) used to implement the data analysis.

From an architectural point of view, we emphasize that our system is *modular*. It is possible for the research com-

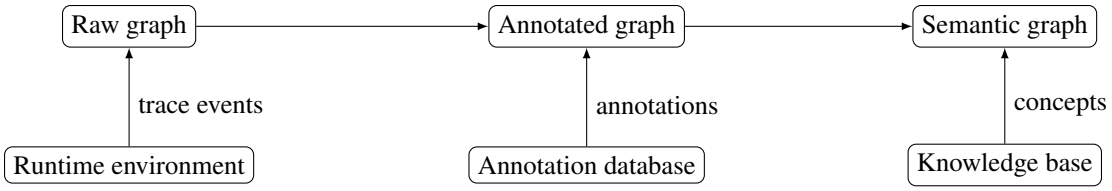

Figure 3: Schematic of dataflow graph pipeline

munity to build on some components of our system while ignoring or replacing others. For example, a researcher could retain our generic system for functional annotation without adopting our knowledge representation formalism. Consequently, although we have designated the raw and annotated graphs as "intermediate forms," we regard them as having independent interest.

Related work

Our system automatically constructs a dataflow representation of a data analysis by tracing the flow of objects at runtime. Thus, there are two essential elements in our work: (i) describing a data analysis as a flow graph, and (ii) tracing data flow in a computer program. There is considerable prior art in both of these areas. However, to our knowledge, we are the first to combine these elements to produce a fully automated system for summarizing a data analysis. In this subsection, we review related work in the two areas.

Regarding the first element, we should acknowledge the obvious fact that many data analysis environments *explicitly* represent analyses as dataflow graphs. Almost always, these environments are software applications with graphical interfaces, not programming languages; examples include open source software like Knime (Berthold et al. 2009) and Orange (Demšar and Zupan 2013) and commercial offerings like SPSS Modeler and RapidMiner. Graphical environments for scientific workflow management, such as Galaxy (Goecks et al. 2010), Kepler (Altintas et al. 2004), and Taverna (Oinn et al. 2004), belong to a related category, as they often include data analysis functionality. In this paper, we focus exclusively on the programming language model of data analysis, in particular the Python language.

To some extent, our goal of recording the steps of a data analysis is shared by the field of *data provenance*. The provenance of a data resource includes its origin and the process of transformation by which it was derived. Insofar as the survey (Simmhan, Plale, and Gannon 2005) is representative, we would argue that the main difference between our system and the typical data provenance system is *granularity*. In data provenance, the finest granularity of data resource is usually files or database records, whereas our system operates on a single file and traces arbitrary programming language objects. For example, the StarFlow system (Angelino, Yamins, and Seltzer 2010) is, like our's, aimed at data analysts and built around Python, but operates at the level of scripts. There are some graphical environments for data provenance that offer finer granularity, such as VisTrails (Callahan et al. 2006), but again our work targets text-based programming languages.

Turning to the second element, the analysis and instrumentation of computer programs is a large field of research unto itself. In the computer science community, dataflow analysis is usually a means to the end of building better compilers, debuggers, and verification tools. A dataflow analysis can involve static analysis or dynamic analysis (or both), but the literature emphasizes static analysis because of its relevance to optimizing compilers (Aho et al. 2006; Seidl, Wilhelm, and Hack 2012). Static dataflow analyses are classified as *intraprocedural* (within a single procedure) or *interprocedural* (between procedures across the entire program). The latter class includes methods for extracting function call graphs. However, as we have already noted, our system does not use static analysis.

To our knowledge, the most relevant work in the area of dynamic analysis is Adrian Lienhard's *dynamic object flow analysis*, described in a dissertation (Lienhard 2008) and several papers (Lienhard, Ducasse, and Gîrba 2009; Lienhard, Gîrba, and Nierstrasz 2008). Like us, he proposes to track how objects are passed between functions at runtime. However, his approach differs in both purpose and implementation. His primary goal is to measure object aliasing in large object-oriented systems, as an aid to debugging and reverse engineering. His work does not explicitly address our main technical problem, namely to map the (imperative) object-oriented programming model onto the (functional) dataflow programming model. In our work, we offer only a partial solution to that problem, which is in general quite challenging.

Conclusion

In this work, we proposed and implemented a novel method to automatically extract a machine-interpretable representation of a data analysis. We argued that the scientific and social good communities could profit greatly by a new kind of collaborative data science platform, emphasizing open-ended data analysis and rich, machine-interpretable content. Our system could serve as a foundational technology for such a platform.

We see our work as continuous with the quiet but far-reaching transformation of the scientific process that is currently underway. The general trend is towards greater openness and interconnectivity, with special emphasis on open-access research, reproducibility (Munafò et al. 2017), collaboration (Sonnenwald 2007), and machine-interpretable knowledge (Renear and Palmer 2009; Evans and Rzhetsky 2010). In a possible culmination of this trend, human and ar-

tificial agents will collaborate on a body of scientific knowledge that is fully open, online, and ontologically integrated. It is difficult to predict the impact of such a development. One author suggests that it could be comparable to the birth of modern science in the Enlightenment era (Nielsen 2012).

We wonder whether a parallel development in the methodology of social good is required to achieve the most ambitious Sustainable Development Goals by 2030. Consider the subgoal within Goal 2: "By 2030, end hunger and ensure access by all people, in particular the poor and people in vulnerable situations, including infants, to safe, nutritious and sufficient food all year round" (United Nations 2015). Or within Goal 3: "By 2030, reduce by one third premature mortality from non-communicable diseases through prevention and treatment and promote mental health and well-being" (ibid). To achieve these goals, advances in basic science must be coupled with deep research on social policy. Policymakers and philanthropists must absorb this diverse knowledge and translate it into effective action. The same values driving the transformation of basic science—openness and interconnectivity—could initiate a paradigm shift for social good, fundamentally changing the way that stakeholders collaborate and share knowledge. We believe that the AI community has a crucial role to play in this development. We hope that the community will share our enthusiasm and work together to realize the vision of machine-assisted social good.

Acknowledgments

We thank Robert McBurney and Hollie Schmidt for teaching us about how the Accelerated Cure Project for Multiple Sclerosis operates. They provided valuable perspective on the opportunities and challenges of collaborative, data-driven social good.

References

Aho, A. V.; Lam, M. S.; Sethi, R.; and Ullman, J. D. 2006. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley, 2 edition.

Altintas, I.; Berkley, C.; Jaeger, E.; Jones, M.; Ludascher, B.; and Mock, S. 2004. Kepler: an extensible system for design and execution of scientific workflows. In *Proceedings of the 16th International Conference on Scientific and Statistical Database Management*, 423–424. IEEE.

Angelino, E.; Yamins, D.; and Seltzer, M. 2010. StarFlow: A script-centric data analysis environment. In *International Provenance and Annotation Workshop*, 236–250.

Ashburner, M.; Ball, C. A.; Blake, J. A.; Botstein, D.; Butler, H.; Cherry, J. M.; Davis, A. P.; Dolinski, K.; Dwight, S. S.; Eppig, J. T.; Harris, M. A.; Hill, D. P.; Issel-Tarver, L.; Kasarskis, A.; Lewis, S.; Matese, J. C.; Richardson, J. E.; Ringwald, M.; Rubin, G. M.; and Sherlock, G. 2000. Gene Ontology: tool for the unification of biology. *Nature Genetics* 25(1):25–29.

Berthold, M. R.; Cebron, N.; Dill, F.; Gabriel, T. R.; Kötter, T.; Meinl, T.; Ohl, P.; Thiel, K.; and Wiswedel, B. 2009. KNIME - the Konstanz information miner: version 2.0 and beyond. *ACM SIGKDD Explorations Newsletter* 11(1):26–31.

Brachman, R., and Levesque, H. 2004. *Knowledge Representation and Reasoning*. Elsevier.

Bull, P.; Slavitt, I.; and Lipstein, G. 2016. Harnessing the power of the crowd to increase capacity for data science in the social sector. In *Proceedings of the ICML Workshop on #Data4Good: Machine Learning in Social Good Applications*, 31–35.

Callahan, S.; Freire, J.; Santos, E.; Scheidegger, C.; Silva, C.; and Vo, H. 2006. VisTrails: visualization meets data management. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, 745–747. ACM.

Colton, S., and Wiggins, G. A. 2012. Computational creativity: The final frontier? In *ECAI 2012: 20th European Conference on Artificial Intelligence*, volume 12, 21–26.

Colton, S.; López de Mantaras, R.; and Stock, O. 2009. Computational creativity: Coming of age. *A.I. Magazine* 30(3):11–14.

Demšar, J., and Zupan, B. 2013. Orange: Data mining fruitful and fun - a historical perspective. *Informatica* 37:55–60.

Evans, J., and Rzhetsky, A. 2010. Machine science. *Science* 329(5990):399–400.

Goecks, J.; Nekrutenko, A.; Taylor, J.; and Team, T. G. 2010. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology* 11(8):1.

Kapoor, S.; Mojsilović, A.; Nguyen Strattner, J.; and Varshney, K. R. 2015. From open data ecosystems to systems of innovation: A journey to realize the promise of open data. In *Bloomberg Data for Good Exchange: 2015 Proceedings*.

Lienhard, A.; Ducasse, S.; and Gîrba, T. 2009. Taking an object-centric view on dynamic information with object flow analysis. *Computer Languages, Systems & Structures* 35(1):63–79.

Lienhard, A.; Gîrba, T.; and Nierstrasz, O. 2008. Practical object-oriented back-in-time debugging. In *European Conference on Object-Oriented Programming*, 592–615.

Lienhard, A. 2008. *Dynamic object flow analysis*. Ph.D. Dissertation, University of Bern.

Marbach, D.; Costello, J. C.; Küffner, R.; Vega, N. M.; Prill, R. J.; Camacho, D. M.; Allison, K. R.; The DREAM5 Consortium; Kellis, M.; Collins, J. J.; and Stolovitzky, G. 2012. Wisdom of crowds for robust gene network inference. *Nature Methods* 9(8):796–804.

Munafò, M. R.; Nosek, B. A.; Bishop, D. V.; Button, K. S.; Chambers, C. D.; du Sert, N. P.; Simonsohn, U.; Wagenmakers, E.-J.; Ware, J. J.; and Ioannidis, J. P. 2017. A manifesto for reproducible science. *Nature Human Behaviour* 1(0021).

Nielsen, M. 2012. *Reinventing Discovery: The New Era of Networked Science*. Princeton University Press.

Noy, N. F.; Shah, N. H.; Whetzel, P. L.; Dai, B.; Dorf, M.; Griffith, N.; Jonquet, C.; Rubin, D. L.; Storey, M.-A.; Chute, C. G.; and Musen, M. A. 2009. BioPortal: ontologies and

integrated data resources at the click of a mouse. *Nucleic Acids Research*.

Oinn, T.; Addis, M.; Ferris, J.; Marvin, D.; Senger, M.; Greenwood, M.; Carver, T.; Glover, K.; Pocock, M. R.; Wipat, A.; and Li, P. 2004. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics* 20(17):3045–3054.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Pérez, F., and Granger, B. 2007. IPython: a system for interactive scientific computing. *Computing in Science & Engineering* 9(3):21–29.

Ragan-Kelley, M.; Pérez, F.; Granger, B.; Kluyver, T.; Ivanov, P.; Frederic, J.; and Bussonier, M. 2014. The Jupyter/IPython architecture: a unified view of computational research, from interactive exploration to communication and publication. In *AGU Fall Meeting Abstracts*, volume 1.

Renear, A. H., and Palmer, C. L. 2009. Strategic reading, ontologies, and the future of scientific publishing. *Science* 325(5942):828–832.

Seidl, H.; Wilhelm, R.; and Hack, S. 2012. *Compiler Design: Analysis and Transformation*. Springer.

Simmhan, Y.; Plale, B.; and Gannon, D. 2005. A survey of data provenance in e-science. *ACM Sigmod Record* 34(3):31–36.

Sonnenwald, D. H. 2007. Scientific collaboration. *Annual Review of Information Science and Technology* 41(1):643–681.

Spivak, D., and Kent, R. 2012. Ologs: a categorical framework for knowledge representation. *PLoS One* 7(1).

Stolovitzky, G.; Monroe, D.; and Califano, A. 2007. Dialogue on reverse-engineering assessment and methods. *Annals of the New York Academy of Sciences* 1115(1):1–22.

United Nations. 2015. Transforming our world: the 2030 agenda for sustainable development. A/RES/70/1.

Vanschoren, J.; van Rijn, J. N.; Bischl, B.; and Torgo, L. 2014. OpenML: networked science in machine learning. *ACM SIGKDD Explorations Newsletter* 15(2):49–60.