Mr. Clean: a Tool for Tracking and Comparing the Lineage of Scientific Visualization Code

Giacomo Tartari*, Lars Tiede*, Einar Holsbø*, Kenneth Knudsen*, Inge Alexander Raknes‡
Bjørn Fjukstad*, Nicolle Mode†, John Markus Bjørndalen*, Eiliv Lund†, Lars Ailo Bongo*§

*Department of Computer Science, University of Tromsø, Norway
§Center for Bioinformatics, University of Tromsø, Norway
†Department of Community Medicine, University of Tromsø, Norway
‡Norstruct, Deptartment of Chemistry, University of Tromsø, Norway

{giacomo.tartari, lars.tiede, kkn011, inge.a.raknes, nicolle.mode, eiliv.lund}@uit.no
{einar, bjorn, jmb, larsab}@cs.uit.no

***Abstract*—Visualization is a key step in scientific analysis and understanding in many fields. Scientific studies often require development of software that produces visualizations. However, as a study proceeds, the software evolves, and both developers and expert users have to periodically ascertain how code modifications affect visualization output and hence the results of the study. To our knowledge, no current visualization framework enables tracking and comparison of the lineage of scientific visualizations. We describe an approach for comparing and maintaining the code for scientific analysis and modeling through interactive comparison of visualization output. We have realized this approach in a tool called Mr. Clean. This tool provides a framework for combining different visualization tools, interaction devices, and display middleware for visual comparisons on high-resolution displays. Mr. Clean also provides user-configurable interactions supported by many devices. We provide use cases and a requirement analysis for our approach, and we describe the design and implementation of Mr. Clean. Source code is available at: https://github.com/UniversityofTromso/mrclean.**

I. Introduction

Software for processing and managing large amounts of data is vital to the advancement of knowledge in many scientific disciplines. Such data-intensive science is often referred to as the fourth paradigm of scientific research where theory, experiment, and simulation are unified for data exploration [1]. One of the most important tools for data exploration is visualization [2], [3].

Scientific visualizations are often the end result of complex statistical analysis. They are often problem-specific, and to provide novel insights, it is often necessary to develop novel analysis methods. Scientists face the challenges not only of understanding and interpreting one visualization, but also of comparing the visualization with other visualizations resulting from alternative statistical methods or models. During the development of new statistical models, code changes may affect the model's output. It is important to understand how a given change in code relates to a change in output. Due to the complexity of these models and their dependence on the input data, such understanding is typically obtained by comparing the visualizations produced by the different models.

No existing code revision tool enables model developers to interactively compare the differences between visualizations resulting from statistical and mathematical models. Instead such comparison is typically done with ad hoc implementation of multi-graph visualizations either in scripting languages such as R, or by combining images in, e.g., PDF documents. This severely limits research capabilities by forcing scientists to perform boring, time-consuming manual studies to understand how changes to code affect analysis results. An interactive approach for tracking and comparing the lineage of scientific visualizations will allow scientists to better evaluate, understand, and improve the analytical performance of their methods.

We propose an approach for maintaining and comparing scientific data analysis and modeling code through interactive comparison of the code's visual outputs. We have implemented our approach in a system called Mr. Clean, which we have deployed for comparative analysis of genomics data cleaning and image processing methods. In the next section we describe two case studies that we use to provide a requirement analysis for our approach. We then outline this approach, and the design and implementation of the Mr. Clean system.

II. Case Studies and Requirements Analysis

We built Mr. Clean to solve problems we and our collaborators met when developing genomics data cleaning and image processing software. We present these two tasks as case studies and use them to formulate requirements for an approach for tracking and comparing the lineage of scientific visualization code.

A. Microarray data cleaning

Our genomics collaborators have developed several data cleaning methods for the Norwegian Women and Cancer (NOWAC) postgenome biobank [4]. Data cleaning is an important first step in genomics data analysis that may severely bias the statistical analysis at later stages. For microarray data analysis, this includes selection of a normalization method and

978-0-7695-5305-4/14 $31.00 © 2014 IEEE
DOI 10.1109/VISSOFT.2014.21

methods for identification and removal of outliers. A human expert uses these methods to identify and visualize outliers. Oldham et al. [5] describe one such outlier removal process in detail. The statistical methods are often implemented in a framework for statistical computing such as R, which provides a big ecosystem of packages for implementing different methods and parsers for genomics data.

The expert uses visualizations to determine which outliers should be removed and which should be kept. An expert typically wants to remove outliers resulting from instrumentation or experimental error and keep outliers resulting from natural biological variation. The expert removes unwanted outliers from the input data and repeats the process as many times as necessary. Although we focus on microarray data, the approach is similar to data cleaning for other data types [6]. There is no one-size-fits-all data cleaning solution, so the analyst must typically explore several statistical methods to find the best approach for a particular dataset. Such a comparative approach is often a collaboration between developers, statisticians, and domain experts.

B. KEGG pathway image processing

We developed a system, Amdex [7], that extracts metadata such as reactions and entities from KEGG pathway images [8] using computer vision algorithms from the OpenCV library [9].

Computer vision problems are notoriously difficult to tackle. Results can vary greatly when different algorithms are used in different orders, but also due to the input images having slightly different features. The common approach is to use an iterative process to figure out which algorithms, parameters, and order of algorithms to apply. To compare two approaches, the developer typically inspects the resulting images. Although we focus on pattern recognition on KEGG images, development of computer vision software in other domains faces similar challenges.

C. Requirements

Each of the above use cases produces many images. For an expert to make meaningful comparisons, she must organize and layout the images in a way that makes it easy to understand the relationships between the images and their association with the model code that produced them. The images can be grouped and sorted based on their underlying approach, dataset, iteration, or statistical method. By showing multiple integrated images simultaneously, users can compare how different data cleaning approaches perform. However, there is no predefined hierarchy among the groups, so the user must explore several groupings. For example grouping by method and ordering by iteration, or grouping by iteration and ordering by method. It may also be necessary to compare visualizations produced by latter-stage analysis tools to truly understand the effects of the data cleaning on the final analysis results.

Based on the case studies, we believe a tool for tracking and comparing the lineage of scientific visualizations should satisfy the following requirements:

1) User-defined visualization scripts. It is not realistic to change the many different tools used to generate visualizations, nor to provide libraries for the many different frameworks used to generate visualizations.
2) Interactive grouping. To compare the lineage of visualizations, it is necessary to interactively group these based on methods, iterations, versions, etc. It is difficult to automatically create the best grouping and ordering of the visualizations, so the tool should provide an interactive user interface to change these.
3) Multiple visualizations. There are often a large number of visualizations. Displaying multiple visualizations simultaneously may make the comparison easier.
4) Automated provenance management. The tool should automate provenance management so that the user can experiment with alternative methods and parameters, and revert to the best methods and parameters.
5) Collaboration. The approaches are often compared in a collaborative setting. The tools should enable multiple people to view and interact with the visualizations.

III. Mr. Clean

We have implemented the Mr. Clean tool to fulfill the above requirements. Our approach is as follows:

1) Domain experts supply visualization scripts that produce image files (requirement 1) to a directory tree managed by Mr. Clean. The path in this tree is user-defined and represents the metadata of the image files.
2) Mr. Clean records new images and changes to the scripts to track the visualizations provenance (requirement 4). We use a revision control system to manage data provenance for both the modeling code and the output visualizations.
3) Mr. Clean reads the visualization files to populate an internal data structure with files and their corresponding metadata. It uses this metadata to group and sort the images. The images are then displayed spatially organized according to group and sort order. The user interacts with the groups and sorted images to explore data by, e.g., regrouping the images or rearranging the spatial display of the images (requirement 2). We use a flexible user input event system that enables the use of the interaction devices most suitable for a given display platform, and the adaptation of these to application-specific interaction patterns.
4) We use large high-resolution displays that show many images simultaneously and enable collaborative image comparisons (requirements 3 and 5). Mr. Clean is deployed and in use on the Tromsø display wall [10].

A. Architecture

Mr. Clean is a component-based distributed system (Figure 1). The Core component is responsible for coordinating the other components, and for providing these components with communication interfaces. It holds the state of the whole system, and it processes and propagates events received from

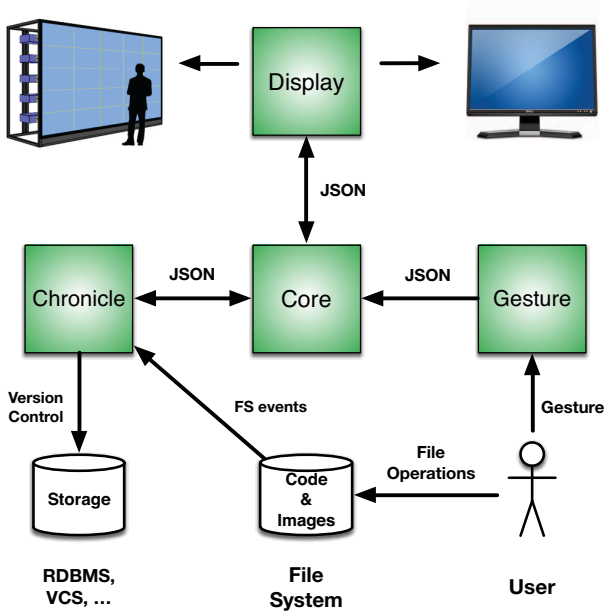

Fig. 1. Mr. Clean architecture.

the other components. For example, it sends a newly-created visualization (a *visual*) to the Display component, or it communicates a new arrangement of visuals to the Display in response to a user gesture from a Gesture component.

The Chronicle component handles file system events caused by file operations, such as the visualization script writing a new image, or the user modifying the visualization script. The Chronicle component uses an external process to keep track of these changes. The current implementation uses *git*.

The Gesture component captures user gestures and translates them into Mr. Clean messages. Multiple gesture components can be used simultaneously to provide interaction with multiple devices. The Display component shows the visuals provided by the Core component to the user on the available display. It is possible to use display walls, which is Mr. Clean's preferred display method.

B. Visuals entities and relationships

The metadata associated with each visual is the path from the base directory where Mr. Clean initializes the *git* repository. Mr. Clean assumes a multi-level hierarchy where each level is associated with a parameter changed in the source code or input data (such as task, approach, method, or iteration). The model developer provides a configuration file that Mr. Clean parses to find names for each level. For example, the configuration file may contain the string *task/approach/method/iteration/name*. Using this configuration file, the metadata for the file genotype/transformation/filtering/3/scatterplot.png is: task=genotype, approach=transformation, method=filtering, iteration=3, name=scatterplot.png. This lack of a predefined structure for a visual's relationships gives Mr. Clean more flexibility in grouping and sorting visuals.

C. Provenance management

Although experiment reproducibility is a cornerstone in the scientific method, it is often difficult to reproduce data analysis results. Automating data provenance is therefore a recognized best practice [11]. Mr. Clean uses the *git* version control system to maintain versioning information for both the visualization code and the output images. When either the code is changed or a new image is created, a new version is committed to the git repository. The git logs provide the necessary data to connect each image to a specific code version, satisfying requirement 4. Mr. Clean takes no other action on the user provided code to not disrupt any established work-flows, satisfying requirement 1.

D. Display System

We use the Display Cloud system [12] to show images on a large, high-resolution display wall. In a display cloud, clients can freely compose their own views on a set of displays. Such a view is called a cloud display. On a cloud display, the client can place and move around visuals that can, for example, be images or a desktop. The client communicates with its cloud display through a combination of JSON RPC and websockets. Display cloud clients include browser-based end-user interfaces and programs such as Mr. Clean.

Mr. Clean composes a cloud display out of the displays of a large, high-resolution display wall, satisfying requirement 3.

E. User Interaction

A core feature of Mr. Clean is its flexible interface for different input systems. This interface accepts input events and maps these to operations that manipulate the visuals' data structure and their layout on the display. We currently use the gesture system described by [13], which we have extended to support different devices.

Our current implementation allows the users to associate a particular gesture with a sorting criterion for the visuals. The sorting criterion is a user-defined list of parameters that must

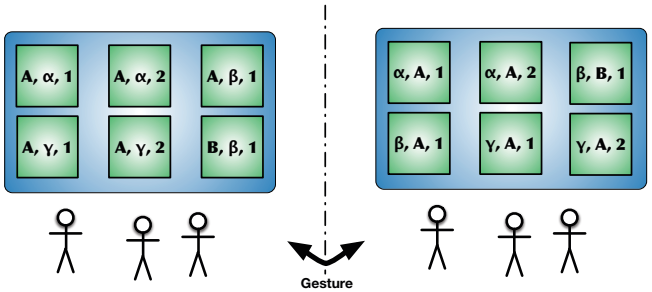

Fig. 2. Schematic example of Mr. Clean in action. The users can interactively change the sorting criterion of the images, according to the metadata, to better compare them. In this case the metadata for each image is: method (latin capital letter), approach (greek letter), iteration (number). This example shows a gesture changing the sorting order from method-approach-iteration to approach-method-iteration and back.

match the directory structure from Section III-B. This allows the users to cycle through different visual arrangements and compare the different results of the scripts executions (Figure 2), satisfying requirement 2.

F. Informal Evaluation

We are currently evaluating Mr. Clean with our domain-expert collaborators. In this paper we report our initial experiences developing and using Mr. Clean.

We found it easy to modify the data cleaning scripts to output files with the necessary metadata. The modifications had the added benefit of better structuring the code, and making it easier to keep track of code changes.

The high resolution and large size of the display wall enables multiple users to compare images by walking along the wall. In addition, the gesture interface allows the users to interact with the images while standing in front of the display wall without having to walk back to a workstation. We believe that a more refined and intuitive gesture interface can improve and possibly speed up the data cleaning process.

IV. Related Work

Display walls are often used for scientific visualization [14]. Hibbs et. al. [15] demonstrate how integrated views enable novel biological discoveries. WindowScape [16] is a window manager that uses flexible implicit grouping of windows, but it relies on mouse and keyboard for interaction. It is common to use different interaction approaches depending on the interaction space [14]. Mr. Clean is, to our knowledge, the first display wall system that provides a specialized display and interaction approach for interacting with scientific visualization code lineages.

Systems for lineage management of scientific code and data include Galaxy [11], which is popular in Bioinformatics. However, the data lineage is usually managed manually. DEVis [17] is an example of a visualization tool that focuses on the evolution of non-code artifacts in a software development setting, namely technical documentation. VisTrails [18] is a system for work-flow and data provenance management with support for data exploration, visualization and simulations.

Recent advances in human–computer interfaces include perceptual input systems [19], and ubiquitous gesture recogntion systems [20].

V. Conclusion

Mr. Clean is a tool for maintaining and comparing scientific data analysis and modeling code through interactive comparisons of their produced output.

Our approach allows users to understand how changes to their scientific code affect analytical results. By automatically organizing the visualization output of different versions, datasets, iterations, and methods, the user can overcome the limitations of ad hoc comparisons.

We believe that our approach is general enough to be applicable to other fields of scientific visualization code than only those of our two case studies. Understanding the effects of output data from changes to scientific data analysis and modeling code is vital for the development of robust analysis and modeling software. Mr. Clean enables such understanding.

Although Mr. Clean is already useful we plan to improve it. The first improvement planned is to provide a *code diff* by selecting two or more images with a gesture. The diffs provide the user information about the cause of the changes in the images by looking at the changes in the visualization code.

References

[1] A. J. Hey, S. Tansley, K. M. Tolle, and others, *The fourth paradigm: data-intensive scientific discovery*, 2009.

[2] Visit, "Visit." [Online]. Available: https://wci.llnl.gov/codes/visit/

[3] ParaView, "ParaView." [Online]. Available: http://www.paraview.org/

[4] V. Dumeaux, K. S. Olsen, G. Nuel, R. H. Paulssen, A.-L. Børresen-Dale, and E. Lund, "Deciphering Normal Blood Gene Expression Variation—The NOWAC Postgenome Study," *PLoS Genet*, vol. 6, no. 3, p. e1000873, Mar. 2010.

[5] M. C. Oldham, G. Konopka, K. Iwamoto, P. Langfelder, T. Kato, S. Horvath, and D. H. Geschwind, "Functional organization of the transcriptome in human brain," *Nat Neurosci*, vol. 11, no. 11, pp. 1271–1282, Nov. 2008.

[6] J. M. Hellerstein, "Supplment to: Quantitative data cleaning for large databases," *United Nations Economic Commission for Europe*, 2008.

[7] K. Knudsen, "Amdex: automated meta-data extraction from kegg pathways," Bachelor Thesis, Dept. of Computer Science, University of Tromsø, 2014.

[8] KEGG. [Online]. Available: http://www.kegg.jp/

[9] OpenCV, "OpenCV." [Online]. Available: http://opencv.org/

[10] O. Anshus, D. Stødle, T. Hagen, B. Fjukstad, J. Bjørndalen, L. Bongo, Y. Liu, and L. Tiede, "Nine years of the tromsø display wall," in *Proceedings of Powerwall, SIGCHI Workshop.*, 2013.

[11] J. Goecks, A. Nekrutenko, and J. Taylor, "Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences," *Genome Biology*, vol. 11, no. 8, p. R86, Aug. 2010.

[12] L. Tiede, J. M. Bjørndalen, and O. J. Anshus, "Cloud Displays for Mobile Users in a Display Cloud," in *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*. New York, NY, USA: ACM, 2013, p. 12:1–12:6.

[13] G. Tartari, D. Stødle, J. M. Bjørndalen, P. H. Ha, and O. J. Anshus, "Global interaction space for user interaction with a room of computers," in *Human System Interaction (HSI), 2013 The 6th International Conference on*. IEEE, 2013, p. 84–89.

[14] J. Leigh, A. Johnson, L. Renambot, T. Peterka, B. Jeong, D. Sandin, J. Talandis, R. Jagodic, S. Nam, H. Hur, and Y. Sun, "Scalable Resolution Display Walls," *Proceedings of the IEEE*, vol. 101, no. 1, pp. 115–129, Jan. 2013.

[15] M. Hibbs, G. Wallace, M. Dunham, K. Li, and O. Troyanskaya, "Viewing the Larger Context of Genomic Data through Horizontal Integration," in *Information Visualization*, Jul. 2007, pp. 326–334.

[16] C. Tashman and W. K. Edwards, "WindowScape: Lessons Learned from a Task-centric Window Manager," *ACM Trans. Comput.-Hum. Interact.*, vol. 19, no. 1, p. 8:1–8:33, May 2012.

[17] J. Zhi and G. Ruhe, "DEVis: A tool for visualizing software document evolution," in *Proceedings of VISSOFT*, Sep. 2013, pp. 1–4.

[18] VisTrails. [Online]. Available: http://www.vistrails.org/

[19] K. Sabir, C. Stolte, B. Tabor, and S. O'Donoghue, "The Molecular Control Toolkit: Controlling 3D molecular graphics via gesture and voice," in *Proceedings of BioVis*, Oct. 2013, pp. 49–56.

[20] B. Kellogg, V. Talla, and S. Gollakota, "Bringing gesture recognition to all devices," in *Usenix NSDI*, vol. 14, 2014.