A tool to help lay out Multiple View Visualisations guided by view analysis

H. M. Al-maneea[1,2] and J. C. Roberts[1]

[1]Bangor University, United Kingdom
[2]University of Basrah, Iraq

Abstract

In this paper, we present 'Layouts for Multiple View' (LMV), a tool that helps users build, control and save multiple view visualisations simply and easily using a bespoke grammar. The tool incorporates template multiple view layout strategies as quantified from prior research on view analysis, and the user can build different layouts by defining the grammar, or through the linked visual interface. LMV saves the multiple view layout as a JSON file, including all the details of the layout and attributes of the visualisation, which can be subsequently loaded and adapted or used to create dashboards. LMV guides the user to (i) design and control the multiple view layout, (ii) add data and allocate a specific visualisation technique for each view, and (iii) to adapt specific appearance properties of the layout.

CCS Concepts

• ***Human-centered computing*** → ***Visualization systems and tools;***

1. Introduction

Visualization tools, libraries and systems all help users create visualisations. While there are many ways to create a visualisation, controlling the layout of multiple view systems is still difficult. This is not the case for websites, where there are many design tools to help users lay out their websites. Moreover, Java programmers can use methods such as GridBagLayout to easily organise different components, or a BorderPane layout to place nodes: top, bottom, left, right, and center. Why can we not have the same idea in visualisation? Visualisation developers would likewise benefit from a similar system. Templates to help users follow 'typical' layout strategies, and methods to graphically design different layouts. At the start of our research, we asked several questions: How can we develop a system to help visualisation users lay out their views? How can we allow users to quickly lay out their views, and then easily change their design? How do we map data to a view and easily change the appearance of the layout viewer? Our motivation is to give the users the ability to create juxtaposed view layouts in a simple and easy way. We also wanted to allow users to create 'typical' layouts, and consequently we drew on recent work by Almaneea and Roberts [AR18, AR19], on quantifying and identifying typical and frequent layout strategies.

We present LMV (Layouts for Multiple views), a tool to help users create and re-configure different multiple view visualisations. We designed the tool by starting with design sketches, and incrementally developed a final prototype in JavaScript and HTML. Users can create different multiple view layouts, associate their choice of visualisation libraries to a view, and change the appearance of the final multiple view layout. They can control this through either a bespoke shorthand grammar or longhand JSON code, choose default layouts, design their own layout, and go back-and-forth between grammar and visual interface.

2. Background and Related work

Research in multiple-view visualisation is broad [Rob07]. For example, there are papers that focus on the theoretical aspects of multiple views, such as rules and principles for the use of multiple views [WBWK00], investigating juxtaposition, superposition and explicit designs [GAW*11, Gle18], the phraseology of multiple views [RAMB*19] and recently the structure of view layouts [AR18, AR19]. Other researchers have focused on developing coordinated multiple view systems, such as ComVis [MFGH08], Snap-together [NS00], Waltz [Rob98], Improvise [Wea04], Jigsaw [SGL08]). Or investigated techniques such as linked brushing [BC87] and rudiments of coordination [BRR03]. While other researchers have directed their attention to keep multiple views consistent [QH18], or work across large displays [LKD19].

There has been some research on multiple view layout. For instance, Spotfire and IVEE could re-group widgets [AS94] and later Spotfire tools allowed users to drag and drop views. Likweise Snap-Together [NS00] snapped views together to create custom combinations. And Improvise [Wea04] packs many visualisations in a tight space, while Keshif allows users to create visualisations in different parts in the Web page [YEB17]. Layout is also controlled in dashboard visualisations, and there are many ways to create visualisation dashboards, including D3.js [BOH11], highcharts (highcharts.com), and in tools such as Tableau, SAS or Power BI. Tools like datahero.com can be used to create dashboards using a drag-

DOI: 10.2312/eurp.20201114

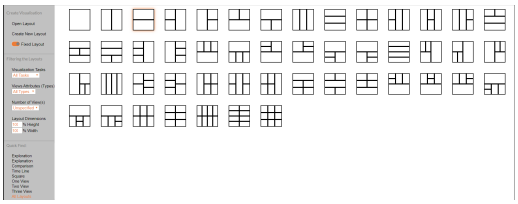

Figure 1: *Template viewer. Users can choose a starting layout, search for a specific view quantity. They can later edit the template.*

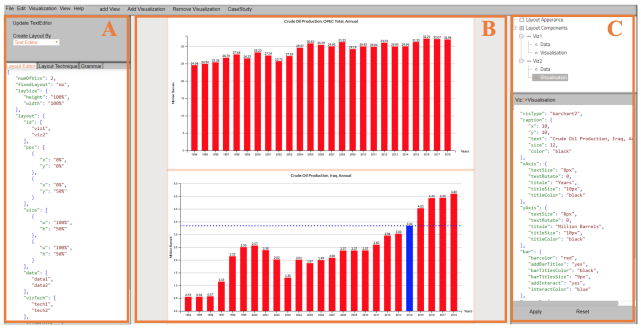

Figure 2: *LMV tool, with three linked views. (A) grammar panel to edit the grammar (either shorthand e.g., v(50,50) or full JSON, shown), (B) Visualisation panel of either the wireframe editor or visualisation editor (shown) and (C) property panel.*

and-drop interface, and small-multiples can be created by using the grammar of graphics [Wil05] in ggplot2 [Wic15]. Shiny and shinydashboard or Highcharter (an R wrapper for Highcharts in javascript) can be used to create bespoke multiple view dashboards (cran.r-project.org). But there has been no systematic study investigating view-layout strategies, and no tool that focuses on multiple view layout using a specialised grammar.

3. Design of Layouts for Multiple View (LMV)

To create a multiple view system, the views must be placed on the screen in a deliberate way. We can consider who (or what) makes this decision, it could be data-driven (like lattice or ggplot2) where the view position is governed by the data; or it could be controlled by the designer [AR19]. We can think of the challenge of view layout as one of 'geometric object packing' [LMM02]; placing the objects (views in our instance) side-by-side such that they fit and don't leave any gaps; therefore this challenge also draws upon space filling ideas, e.g., used in small-multiples [MXH*03], view bracketing [Rob04], treemaps [SW01] and spreadsheet visualisations [CKBR97]. Consequently, and similar to packing problems, we realised that we needed to solve two **requirements**: first, a way to define the space (the view layout), and second to define the objects in that space – the visualisation types used, how data is applied to a view, and the appearance of all the views (such as view spacing, background colour, etc.).

Our **design methodology** was to start by sketching designs, then build several prototypes, evaluating each major prototype in turn. We chose the Five Design-Sheet method [RHR16, RHR17] because we had used it in several projects. And we followed Wilson's method of heuristic evaluation [Wil14] to examine each prototype. We used three developers (separate to this project) two with visualisation expertise and one a generalist software engineer to provide think-aloud comments, we developed three major prototypes, and are now working on a full evaluation of the tool. Our first prototype (a two-view system) used a slice-and-dice methodology, to draw the view layouts. On the editor view the user can cut a view in half, and subsequently cut each other view in two. The feedback was positive, but the experts found some configurations tedious to create (such as 5 by 5 view). Sometimes it is easier to craft view layouts by hand, and other times through commands. Subsequently, we defined a grammar based on the vertical and horizontal cuts, and developed the system further. Our final prototype has four main view panels, each addressing a different task: template viewer, grammar panel, visualisation panel, and property panel. **The template viewer** (shown in Figure 1) is the default first view, and allows a pre-built design to be selected. Users can also search and filter designs. After selecting a starting template (which could be blank) the three-part viewer opens.

The second screen has three main parts (Figure 2), the grammar panel, visualisation editor, and property panel. **The grammar panel** (Figure 2A), allows users to edit a language description of the layout. The basic idea is based on hierarchical cuts. Cut one view horizontally (h) or vertically (v) to produce two views, and so on. E.g., *h(50,50)* creates an equal sized side-by-side view ◫. The value 50 is representative, $h(20,20)$ would provide the same result. Complex cuts can be easily created, e.g., *h(50**v(75,25)**,50)* creates layout with three views, a long bottom view, with the top split 75% across ⊟. We can control quantities, make variables, define prototype layouts in the grammar. E.g., *a:50; b:(a,a); Expr:h(30vb, 40hb, 30vb)* creates a layout with six views. We can quickly define a nine-grid view (*"laytech":"Grid3*3"*) or place six views in a golden ratio with a centre in the fourth quarter (*"laytech":"GoldenRatioV6Q4"*). A full description of the grammar is not possible here, due to space constraints. The JSON can be saved, and reloaded, and is checked for errors on load. **The visualisation editor** shows either a wireframe layout editor (without visualisations), or the visualisation view. The wireframe editor allows users to add, delete and change the size and the position of the views, and snap wireframe views together. The grammar description is dynamically updated on the left panel, allowing users to jump between grammar or layout descriptions. **The property panel** allows users to change the appearance of the layout (border width, background colour, etc.) and how the visualisations and data are mapped to panels. Developers can fill in any visualisation technology they require, for instance, a Google Map view, D3 view, highcharts view, etc. E.g., the example in Figure 2 uses D3.

4. Discussion and Conclusions

We have designed and built a tool 'Layouts for Multiple View' (LMV) to help users lay out multiple view systems. Users can select a template-layout, or edit the layout visually or control the layout through a grammar. The grammar is used to save/load view layouts, and each view is linked, such when the user controls the wireframe editor the grammar updates. We have developed and improved the prototypes with heuristic feedback. We have demonstrated its use with D3, and it has potential to be used with other visualisation languages and tools. We are still developing and improving LMV, and are planning a more in-depth user evaluation.

References

[AR18] AL-MANEEA H. M., ROBERTS J. C.: Study of Multiple View Layout Strategies in Visualisation. In *Posters presented at the IEEE Conference on Visualization (IEEE VIS 2018), Berlin, Germany* (Oct. 2018). URL: http://ieeevis.org/year/2018/welcome. 1

[AR19] AL-MANEEA H. M., ROBERTS J. C.: Towards quantifying multiple view layouts in visualisation as seen from research publications. In *2019 IEEE Visualization Conference (VIS)* (Oct 2019), pp. 121–121. doi:10.1109/VISUAL.2019.8933655. 1, 2

[AS94] AHLBERG C., SHNEIDERMAN B.: Visual information seeking using the filmfinder. In *CHI '94: Conference companion on Human factors in computing systems* (1994), ACM Press, pp. 433–434. doi:10.1145/259963.260431. 1

[BC87] BECKER R. A., CLEVELAND W. S.: Brushing Scatterplots. *Technometrics 29*, 2 (1987), 127–142. 1

[BOH11] BOSTOCK M., OGIEVETSKY V., HEER J.: D3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2301–2309. doi:10.1109/TVCG.2011.185. 1

[BRR03] BOUKHELIFA N., ROBERTS J. C., RODGERS P. J.: A coordination model for exploratory multiview visualization. In *Proceedings International Conference on Coordinated and Multiple Views in Exploratory Visualization - CMV 2003* (July 2003), pp. 76–85. doi:10.1109/CMV.2003.1215005. 1

[CKBR97] CHI E. H.-H., KONSTAN J., BARRY P., RIEDL J.: A spreadsheet approach to information visualization. In *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 1997), UIST '97, ACM, pp. 79–80. doi:10.1145/263407.263513. 2

[GAW*11] GLEICHER M., ALBERS D., WALKER R., JUSUFI I., HANSEN C. D., ROBERTS J. C.: Visual comparison for information visualization. *Information Visualization 10*, 4 (2011), 289–309. doi:10.1177/1473871611416549. 1

[Gle18] GLEICHER M.: Considerations for visualizing comparison. *IEEE Transactions on Visualization and Computer Graphics 24*, 1 (Jan 2018), 413–423. doi:10.1109/TVCG.2017.2744199. 1

[LKD19] LANGNER R., KISTER U., DACHSELT R.: Multiple coordinated views at large displays for multiple users: Empirical findings on user behavior, movements, and distances. *IEEE Transactions on Visualization and Computer Graphics 25* (1 2019), 608–618. doi:10.1109/TVCG.2018.2865235. 1

[LMM02] LODI A., MARTELLO S., MONACI M.: Two-dimensional packing problems: A survey. *European Journal of Operational Research 141*, 2 (2002), 241 – 252. doi:10.1016/S0377-2217(02)00123-6. 2

[MFGH08] MATKOVIC K., FREILER W., GRACANIN D., HAUSER H.: Comvis: A coordinated multiple views system for prototyping new visualization technology. In *2008 12th International Conference Information Visualisation* (July 2008), pp. 215–220. doi:10.1109/IV.2008.87. 1

[MXH*03] MACEACHREN A., XIPING D., HARDISTY F., GUO D., LENGERICH G.: Exploring high-d spaces with multiform matrices and small multiples. In *IEEE Symposium on Information Visualization 2003* (Oct. 2003), pp. 31–38. doi:10.1109/INFVIS.2003.1249006. 2

[NS00] NORTH C., SHNEIDERMAN B.: Snap-together visualization: A user interface for coordinating visualizations via relational schemata. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (New York, NY, USA, 2000), AVI '00, ACM, pp. 128–135. doi:10.1145/345513.345282. 1

[QH18] QU Z., HULLMAN J.: Keeping multiple views consistent: Constraints, validations, and exceptions in visualization authoring. *IEEE Transactions on Visualization and Computer Graphics 24*, 1 (Jan 2018), 468–477. doi:10.1109/TVCG.2017.2744198. 1

[RAMB*19] ROBERTS J. C., AL-MANEEA H. M. A., BUTCHER P. W. S., LEW R., REES G., SHARMA N., FRANKENBERG-GARCIA A.: Multiple Views: different meanings and collocated words. *Computer Graphics Forum* (3 2019). doi:10.1111/cgf.13673. 1

[RHR16] ROBERTS J. C., HEADLEAND C., RITSOS P. D.: Sketching designs using the five design-sheet methodology. *IEEE Transactions on Visualization and Computer Graphics* (January 2016). doi:10.1109/TVCG.2015.2467271. 2

[RHR17] ROBERTS J. C., HEADLEAND C. J., RITSOS P. D.: *Five Design-Sheets – Creative design and sketching in Computing and Visualization*. SpringerNature, 2017. doi:10.1007/978-3-319-55627-7. 2

[Rob98] ROBERTS J. C.: Waltz - an exploratory visualization tool for volume data, using multiform abstract displays. In *Visual Data Exploration and Analysis V, Proceedings of SPIE* (Bellingham, Washington, USA, 1998), Erbacher R. F., Pang A., (Eds.), vol. 3298, pp. 112–122. 1

[Rob04] ROBERTS J. C.: Exploratory visualization using bracketing. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (2004), AVI '04, ACM, p. 188–192. doi:10.1145/989863.989893. 2

[Rob07] ROBERTS J. C.: State of the art: Coordinated multiple views in exploratory visualization. In *Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV 2007)* (July 2007), pp. 61–71. doi:10.1109/CMV.2007.20. 1

[SGL08] STASKO J., GÖRG C., LIU Z.: Jigsaw: supporting investigative analysis through interactive visualization. *Information visualization 7*, 2 (2008), 118–132. 1

[SW01] SHNEIDERMAN B., WATTENBERG M.: Ordered tree layouts. In *IEEE Symposium on Information Visualization, 2001. INFOVIS 2001.* (2001), pp. 73–78. doi:10.1109/INFVIS.2001.963283. 2

[WBWK00] WANG BALDONADO M. Q., WOODRUFF A., KUCHINSKY A.: Guidelines for using multiple views in information visualization. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (New York, NY, USA, 2000), AVI '00, ACM, pp. 110–119. doi:10.1145/345513.345271. 1

[Wea04] WEAVER C.: Building highly-coordinated visualizations in improvise. In *IEEE Symposium on Information Visualization* (2004), pp. 159–166. doi:10.1109/INFVIS.2004.12. 1

[Wic15] WICKHAM H.: *ggplot2: Elegant Graphics for Data Analysis (2nd ed.)*. Springer, New York, NY, 2015. doi:10.1007/978-3-319-24277-4. 2

[Wil05] WILKINSON L.: *The Grammar of Graphics (Statistics and Computing)*. Springer-Verlag, Berlin, Heidelberg, 2005. doi:10.5555/1088896. 2

[Wil14] WILSON C.: *User Interface Inspection Methods. A User-Centered Design Method*. Elsevier (Morgan Kaufmann), 2014. 2

[YEB17] YALCIN M. A., ELMQVIST N., BEDERSON B. B.: Keshif: Rapid and expressive tabular data exploration for novices. *IEEE Transactions on Visualization & Computer Graphics* (2017). doi:10.1109/TVCG.2017.2723393. 1