Flipbooks: Displaying Statistical Analysis Code Step-By-Step with Output

Journal:	*Journal of the American Statistical Association*
Manuscript ID	Draft
Manuscript Type:	Article – Theory & Methods
Keywords:	Exploratory Data Analysis, Graphical Methods, Statistical Computing

Flipbooks: Displaying Statistical Analysis Code Step-By-Step with Output

Evangeline Reynolds 1 *
Dean Staff, United States Military Academy at West Point

April 11, 2021

Abstract

Writing code allows statisticians, analysts, and students to precisely document their work. Piped workflows are increasingly popular because syntax more closely represents the incremental and sequential way people think through analytic work including data manipulation and visualization. Dynamic documents like Jupyter notebooks and Rmarkdown documents allow analysts to present output alongside pipelines. Showing output helps communicate the overall intent of sequences of steps, but the analytic decisions made *within* pipelines may be unclear. New code flipbooks, by contrast, present pipelines step-by-step and side-by-side with output in a movie-like way, which gives insight into what each step in a code pipeline does. This paper describes code flipbooks and the R package flipbookr that helps creators easily and reliably build them.

Keywords: Graphical Methods, Statistical Computing, Exploratory Data Analysis

*The authors gratefully acknowledge the assistance of Emi Tanaka and Garrick Aden-Buie in foundational contributions to the flipbookr project and to the R Consortium for financial support.

1 Introduction

Displaying output alongside analysis code helps communicate intent and how-to knowledge. Tools like Jupytr notebooks and Rmarkdown documents let authors easily combine code and output (Xie et al. 2018, Kluyver et al. 2016). However, output is often the final product of sequences of steps. Newcomers to specific functions, to the programming language used, or to coding itself might struggle to guess what each step in a code sequence accomplishes when shown only final output. Practitioners may clarify the steps of another person's code by working with that code interactively — partially executing code to observe intermediate output — but this is time consuming and may be prohibitive without appropriate infrastructure in place (input data, packages, software).

Flipbooks (code movies) are a communication tool that help address this problem by presenting code pipelines step-by-step and side-by-side with intermediate output, providing insight on what each function in a sequence accomplishes. Movie-like presentation helps readers effortlessly identify how new code triggers change in output. The animation effect is achieved by presenting each new code addition and output as a frame in an unfolding narrative; each step is presented on a slide in a slide show.

Movie-like presentation is key to the efficiency of code flipbooks as a communication method. Movement is preattentively processed, so when new code and output states are presented back-to-back like frames in a movie, the viewer does not need to exert effort and concentrate to identify differences between stages (Treisman 1985). This contrasts with the case where material is presented laterally, i.e. side-by-side. With lateral presentation viewers need to *search* for visual differences (Wolfe 2010). Eliminating the task of visual search — with its reliance on working memory (Sweller et al. 1998, Ma et al. 2014) — should avail more cognitive resources to identifying data analytic strategies and understanding code behavior. Facilitating visual comparison via superimposition — either by presenting images at once or toggling between them — has been used to great effect in a variety of fields and applications. In astronomy, the blink comparator was used to compare images of the night sky (Tombaugh 1946); in literary studies Hinman collator is used to compare literary editions (Smith 2000) ; in optometry the phoropter is used to prescribe appropriate lens strength to patients (Campbell 2008).

Code flipbooks are built on slide show platforms which allows users to progress through steps at their own pace. They may move backward in the sequence to review a stretch along the pipeline relevant to their work, or they may toggle back and forth between two states to clarify what unfamiliar functions accomplish.

Code flipbooks are particularly effective for displaying visual outputs such as data visualization pipelines, like those of the R `ggplot2` package, or table formatting, like those of the R `gt` and `flextable` packages (Wickham 2016, Iannone et al. 2020, Gohel 2020). Flipbooks are also effective in communicating data cleaning and manipulation strategies.

2 Building code flipbooks easily and reliably

Without an automated way to build flipbooks, their creation involves copying and pasting code. This method may be painstaking and error prone especially for longer sequences of steps. The `flipbookr` package in R allows creators to quickly and reliably build flipbooks from full sequences of R analysis code pipelines.[1]

The package performs the following steps. First, it parses source code sequences, identifying appropriate breakpoints in the pipelines. Then it creates the partial code pipelines based on these breakpoints. Finally, the package delivers the partial code pipelines to a presentation platform and presents it alongside the associated output. Content (code and output) is aligned between frames yielding an animation effect.

The default setting automatically creates breakpoints within code pipelines where parentheses are balanced at the end of a line. Pipe operators and other connectors — magrittr's pipe operator, '%>%', (Bache & Wickham 2014) and ggplot's component addition operator, '+', (Wickham 2016) for example — are removed at the end of partial sequences.

The current implementation makes use of `xaringan` (Xie 2020), an Rmarkdown package for building html slideshows using the remark.js platform. The simple slide transition default (i.e. no wipes, fades) for remark.js makes change detection easy; users will not suffer from 'change blindness' that can occur when there is a visual disruption between the presentation of superimposed images (Rensink et al. 1997, Simons & Levin 1997, Pashler 1988, Luck & Vogel 2013).

[1]Limited functionality for python data analytic code pipelines also exists in flipbookr.

flipbookr and xaringan deliver code as text, so users can copy and paste full pipelines or partial stretches along a pipeline to use in other contexts. Also, new code in a pipeline is highlighted as one advances through the flipbook. This allows flipbook readers to pay greater attention to changes in the *output*. The new, highlighted code need only be consulted once the change in output has been understood.

The flipbookr package supports several modes of code revelation. In the default mode, the package automatically finds appropriate break points and reveals code sequentially. In addition, users can also manually define their own break points, indicate non-sequential order for code revelation, cycle through different code inputs, or make a series of replacements. Creators can also show multiple realizations of the same code that contains randomization. This mode may be useful in statistical education and communicating uncertainty.

Furthermore, flipbookr allows creators to choose what is displayed from the code walk-throughs. Up to three panels of content may be presented in parallel: code, output, markdown text sequences, lagged output, target output, and start output.

2.1 Usage

The flipbookr package will be fairly intuitive to use for Rmarkdown users.

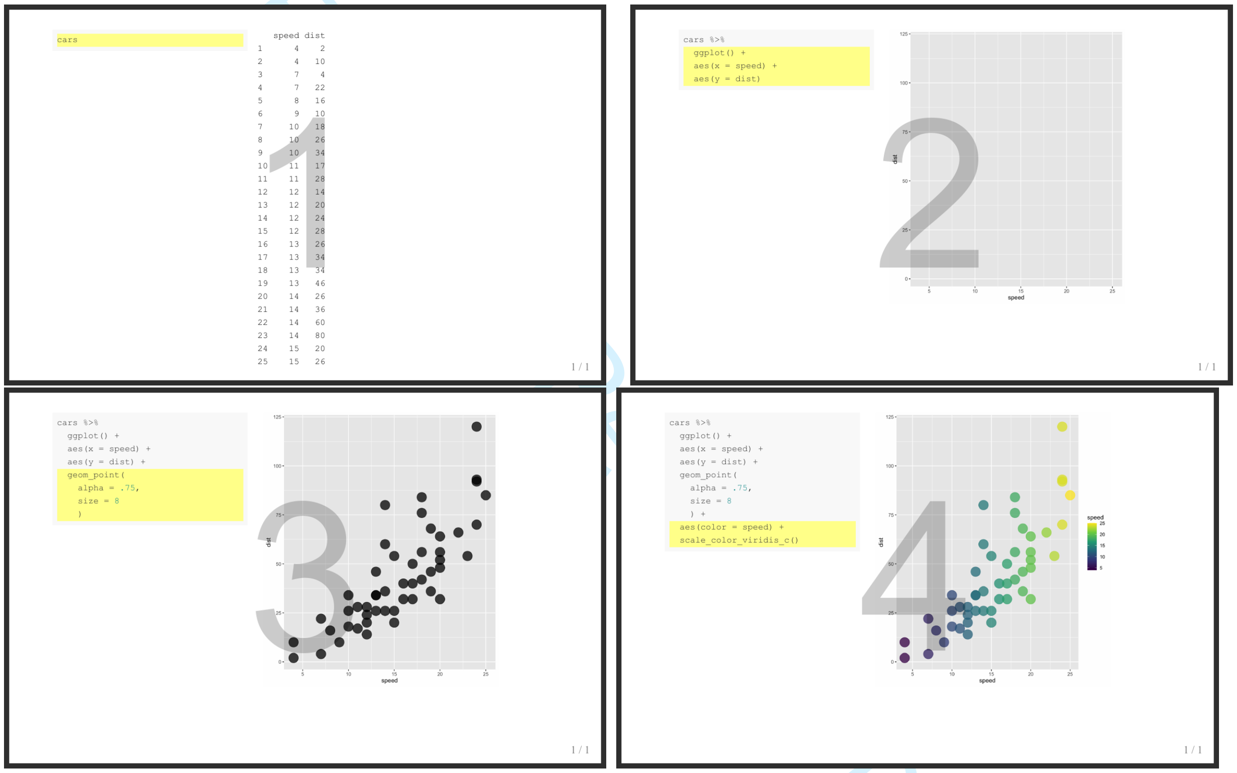
The following code might be the source code for a flipbook in a Xaringan slideshow source document.

```
"```{r cars_chunk, include = F}
cars %>% #BREAK
  ggplot() +
  aes(x = speed) +
  aes(y = dist) + #BREAK
  geom_point(
    alpha = .3,
    size = 8,
    ) + #BREAK
  aes(color = speed) +
```

```
  scale_color_viridis_c() #BREAK
```
```

The function `chunk_reveal()` , which follows, parses the referenced code chunk and delivers partial builds of the code to separate slides alongside accompanying output:

```
`r chunk_reveal(chunk_name = "cars_chunk", break_type = "user")`
```



```
cars
```

```
   speed dist
1      4    2
2      4   10
3      7    4
4      7   22
5      8   16
6      9   10
7     10   18
8     10   26
9     10   34
10    11   17
11    11   28
12    12   14
13    12   20
14    12   24
15    12   28
16    13   26
17    13   34
18    13   34
19    13   46
20    14   26
21    14   36
22    14   60
23    14   80
24    15   20
25    15   26
```

```
cars %>%
  ggplot() +
  aes(x = speed) +
  aes(y = dist)
```

```
cars %>%
  ggplot() +
  aes(x = speed) +
  aes(y = dist) +
  geom_point(
    alpha = .75,
    size = 8
    )
```

```
cars %>%
  ggplot() +
  aes(x = speed) +
  aes(y = dist) +
  geom_point(
    alpha = .75,
    size = 8
    ) +
  aes(color = speed) +
  scale_color_viridis_c()
```

Figure 1: Four frames of a flipbook with frame number overlayed
```

3 Applications

Code flipbooks have a number of practical applications. First, there is great potential for their use in statistical package documentation and reference material. Flipbooks may be used to efficiently communicate functionality that might feel overwhelming to display side-by-side as independent examples. The replacement mode of use, for example, may be use to cycle through the complete set of allowed argument inputs for a function, instead of focusing on one or two argument options in documentation vignettes.

Additionally, flipbooks can be used in the output only mode to aid in presenting complex figures. Figures that are built up step-by-step, such as those created with ggplot2, can also be *presented* step-by-step. This can help presenters systematically discuss each graphical component and visual encoding with audiences.

In educational settings, flipbooks can help introduce students to theoretical concepts while exposing them to the "how-to" knowledge that they to experientially deepen understanding. Moreover, Flipbooks may help teachers create 'cognitive apprenticeship' style reference materials, which are "designed . . . to bring . . . tacit processes into the open, where students can observe, enact, and practice them with help from the teacher. . . " (Brown et al. 1989). Instructors sometimes gloss over steps they consider obvious and unimportant (Hinds 1999); but these details are often necessary to accomplishing goals and newcomers might find them non-obvious. Creating flipbooks can help instructors illuminate the complete decision set constituting a analysis.

Finally, in scholarship analyses could be presented using flipbooks to make work more transparent to wider sets of audiences. Even if a consumer of an analysis is unfamiliar with specific functions use in data analysis pipelines, presenting work in the form of a flipbook could illuminate analytic moves. Using flipbooks in scholarship could also mean that more how-to knowledge is concurrently transferred with the substantive findings of studies (Brown et al. 1989).

4 Conclusion

Flipbooks communicate how additions to code pipelines translate to change in output by presenting change as frames in a slide show yielding a movie-like effect. Changes between slides are perceived as movement, which is preattentively processed and makes the change discovery very efficient. The new flipbookr package in R combined with the xaringan slideshow package aids in creating flipbooks by parsing code pipelines, creating partial builds, and delivering the partial sequence builds to slides with their associated output. Flipbooks shows promise in contributing to statistical and analytics education, presentation, and documentation.

References

Bache, S. M. & Wickham, H. (2014), *magrittr: A Forward-Pipe Operator for R*. R package version 1.5.
URL: *https://CRAN.R-project.org/package=magrittr*

Brown, J. S., Collins, A. & Duguid, P. (1989), 'Situated cognition and the culture of learning', *Educational researcher* **18**(1), 32–42.

Campbell, G. L. (2008), *Phoroptors: Early American Instruments of Refraction and Those who Used Them*, Gary L. Campbell.

Gohel, D. (2020), *flextable: Functions for Tabular Reporting*. R package version 0.5.11.
URL: *https://CRAN.R-project.org/package=flextable*

Hinds, P. J. (1999), 'The curse of expertise: The effects of expertise and debiasing methods on prediction of novice performance.', *Journal of experimental psychology: applied* **5**(2), 205.

Iannone, R., Cheng, J. & Schloerke, B. (2020), *gt: Easily Create Presentation-Ready Display Tables*. R package version 0.2.2.
URL: *https://CRAN.R-project.org/package=gt*

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J. B., Grout, J., Corlay, S. et al. (2016), Jupyter notebooks-a publishing format for reproducible computational workflows., *in* 'ELPUB', pp. 87–90.

Luck, S. J. & Vogel, E. K. (2013), 'Visual working memory capacity: from psychophysics and neurobiology to individual differences', *Trends in cognitive sciences* **17**(8), 391–400.

Ma, W. J., Husain, M. & Bays, P. M. (2014), 'Changing concepts of working memory', *Nature neuroscience* **17**(3), 347.

Pashler, H. (1988), 'Familiarity and visual change detection', *Perception & psychophysics* **44**(4), 369–378.

Rensink, R. A., O'Regan, J. K. & Clark, J. J. (1997), 'To see or not to see: The need for attention to perceive changes in scenes', *Psychological science* **8**(5), 368–373.

Simons, D. J. & Levin, D. T. (1997), 'Change blindness', *Trends in cognitive sciences* **1**(7), 261–267.

Smith, S. E. (2000), '"the eternal verities verified": Charlton hinman and the roots of mechanical collation', *Studies in Bibliography* **53**, 129–161.

Sweller, J., Van Merrienboer, J. J. & Paas, F. G. (1998), 'Cognitive architecture and instructional design', *Educational psychology review* **10**(3), 251–296.

Tombaugh, C. W. (1946), 'The search for the ninth planet, pluto', *Leaflet of the Astronomical Society of the Pacific* **5**, 73.

Treisman, A. (1985), 'Preattentive processing in vision', *Computer vision, graphics, and image processing* **31**(2), 156–177.

Wickham, H. (2016), *ggplot2: Elegant Graphics for Data Analysis*, Springer-Verlag New York.
URL: *https://ggplot2.tidyverse.org*

Wolfe, J. M. (2010), 'Visual search', *Current biology* **20**(8), R346–R349.

Xie, Y. (2020), *xaringan: Presentation Ninja*. R package version 0.16.
URL: *https://CRAN.R-project.org/package=xaringan*

Xie, Y., Allaire, J. & Grolemund, G. (2018), *R Markdown: The Definitive Guide*, Chapman and Hall/CRC, Boca Raton, Florida. ISBN 9781138359338.
URL: *https://bookdown.org/yihui/rmarkdown*