

tidyTouch: An interactive visualization tool for data science education

Jonah DeVaney<sup>1</sup>

<sup>1</sup> East Tennessee State University

## Abstract

abstract is going here. title page being rewritten in LaTeX to adhere to thesis submission format

tidyTouch: An interactive visualization tool for data science education

## Introduction

Technology is an absolute necessity in professional and academic spaces, where engineers, researchers, programmers, and more utilize an ever-growing collection of digital tools for organizing and analyzing the information vital to their work. Free and open-source software (FOSS) provides opportunities for unconditional access to useful programs and their source code for the sake of modification, improvement, and further sharing (Open Source Initiative, 2020). In cases where software is used for statistical analysis, many find R, a programming language for statistical analyses, to be a universally applicable tool to which many dedicated maintainers and community members contribute (R Core Team, 2020). While accessibility and extensive documentation make R available to individuals with limited knowledge or experience with programming, it is a more technically advanced tool, where a user writes code to read data, perform analyses, and create reports. This barrier gives reason to consider software options that may have limited capability but provide a more intuitive interface.

Combinations of spreadsheet editing programs like Microsoft Excel (Microsoft, 2019) and statistical analysis software like Minitab (Minitab, 2020) and IBM SPSS (IBM, 2017) allow less experienced analysts, like students, to visualize the possible structures and operations available for use with their data. These are typically marketed with intentions of the majority of users taking advantage of the graphical user interfaces (GUI), which are designed to give a point-and-click interaction method that engages the underlying code. These have the disadvantages of limited automation and accessibility, where users must manually perform steps of their analyses, often multiple times, on systems granted permission through paid subscriptions for software usage **citation needed?**.

The R community and immensely popular integrated development environment (IDE), RStudio, encourage the same transparency and information-sharing reflected in the mentality

of FOSS distribution (RStudio Team, 2020). Analyses in R can be performed in the console, where commands are given in the R language to be interpreted by the system. These analyses can just as easily be written in the form of a script that can be run as a combination of all operations intentionally recorded. Providing a powerful set of methods with infinite complexity, R programming is useful for anyone that works with data. As the practice of using large amounts of data to inform processes in various fields becomes more common through the expansion of data science as a field, education has and will continue to experience significant impacts (Picciano, 2012). This can be observed on multiple fronts, where data science practices can be utilized by educational institutions in operation, as well as be implemented more as instructional content (Williamson, 2017).

Data science education has the potential to bring those that would otherwise use proprietary, GUI-based programs, like those previously mentioned, the small amount of technical training required to begin developing proficiency with tools like R. The language and its additional packages, supplemental files that allow expanded capabilities defined by their authors **citation needed?**, have extensive documentation that can be easily viewed within the RStudio IDE (RStudio Team, 2020); however, these are written as technical manuals that read in the format of R. The need to assist new students of R in gaining a literacy level in which they can solve problems on their own has motivated projects like the development of RStudio's Primers, online tutorials that teach basic example scenarios to utilize the RStudio suite of packages known as the **Tidyverse** (RStudio Team, 2020; Wickham, 2017). These kinds of resources for data science education are crucial for training new academics and professionals as they work to embrace the ever-growing importance of data.

## Data Visualization

Working with complex data is not a part of every position in an organization, but the information drawn from it can be meaningful to anyone. The importance of communication is highlighted by data visualization, arguably one of the most familiar aspects of data science to anyone outside of the field. Data visualization is a broad term describing any case in which data is visually represented in a form that allows interpretation of relationships and attribution of meaning to the information (Murray, 2017). Popular proprietary software makes generating plots a quick and simple task, providing ranges of options for visualization types in menus for easy selection; this is a commonly used feature of Microsoft Excel, where inserting “charts” requires few clicks to choose a visualization and designate portions of the available data to be used (Microsoft, 2019). This process is similar for even more specialized software for statistical analysis, like SPSS.

The approach to generating the same type of graphical product in R requires a knowledge of the programming language and the preferred **tidyverse** package for visualization, **ggplot2**. Rather than choosing from menus, a user must create the code for a plot. This is the challenge in teaching R and **ggplot2**, that other options are easier to use. As mentioned, a wide range of careers can be supported by an individual becoming a student of R. A tool that uses basic R and **ggplot2** language in an intuitive point-and-click interface would allow students and others in the early stages of their data science education to build their knowledge base without sacrificing convenience of ease.

### ***ggplot2*** and The Tidyverse

The popular package **ggplot2** was developed based on a concept known as the grammar of graphics (Wickham, 2016). The grammar of graphics breaks down components of a visual representation of data, what many would call a chart, and give specificity to its components in such a way that combinations of these characteristics can be used to generate

more unique and meaningful “graphics” (Wilkinson, 2005). This design creates a systematic structure for the `ggplot2` package’s own language of functions, the words used to specify modifications to what would otherwise be basic plots.

Every graphic is composed of the same basic components that are each addressed by a dedicated section of the code that creates a `ggplot2` graph. The *data* to be used is the first component specified, with a typical second addition being the plot’s *layers*. A *layer*, in its simplest form, is the geometric object to be used in representing chosen data, what could be considered the type of graph (Wickham, 2016). The *data* and a *layer* are the only components that need to be specified by a user to create a graphic with `ggplot2`, as the other components are set to defaults. These are *scales*, specifications of size dimensions for the plot axes, the related *coordinate system*, *themes* that alter the plot’s overall appearance, and *facets* that can divide information among multiple plots in a combined graphic (Wickham, 2016). Two basic plots showing differences between *themes* and *faceting* can be seen in Figure 1.

```
mtcars %>% ggplot(aes(x = hp, y = mpg)) + geom_point()
mtcars %>% ggplot(aes(x = hp, y = mpg)) + geom_point() +
  theme_bw() + facet_wrap(~cyl)
```

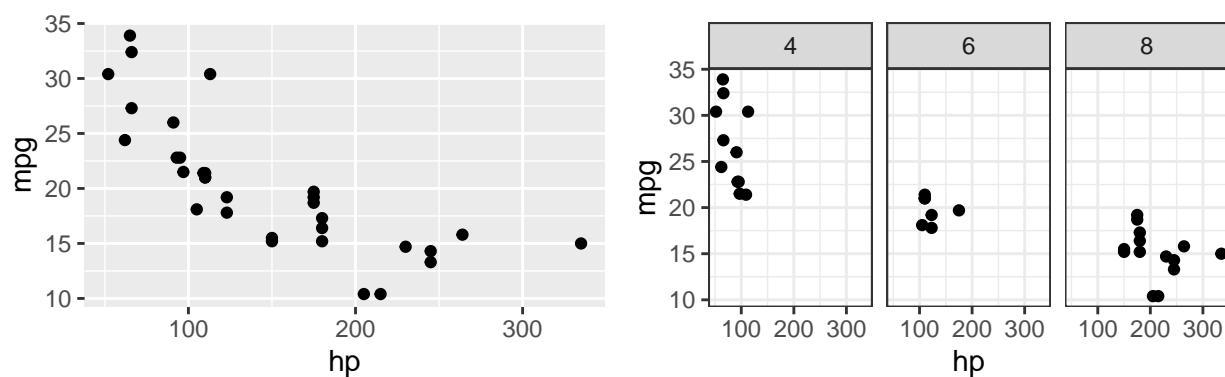


Figure 1. The two graphics are both comparing horsepower with gas mileage of cars in a sample R dataset. The (right) shows faceting by the number of cylinders a car has.

The code displayed for these graphs subtly demonstrates a concept at the core of

`ggplot2` and other **Tidyverse** packages. These tools are designed to be utilized in the infinitely complex workflow of a data scientist, where a person is performing multiple operations on an object in a series of steps (Wickham, 2017). This object is often a dataset that is being read, summarized, or transformed to a “tidy” format, hence the name of the suite. Packages like `dplyr` and `ggplot2`, that use datasets saved in R, require this tidy format, which can be simply described as every variable having a column to represent it and each data point or observation for that variable makes a row (Wickham, 2016). Working with data in this format and gaining experience with **Tidyverse** packages can demonstrate the importance of workflow organization and exposure to proper practice during data science education.

Using R can involve much more than importing a dataset and performing the desired analysis. Once a file is imported, manipulating the data is often necessary. This is referred to as transformation, which involves rearranging, filtering, condensing, or calculating new components of datasets. All of these operations are done with simple functions, verbs that represent complex operations in R and take on the form of `f(x, y, ...)` in code (Wickham et al., 2020). The “x” represents the object being acted on with “y, ...” representing any specification of options for the operation. As an analyst transforms data, the series of steps used can be simplified by combining the verbs into a single string of multiple operations using a pipe (`%>%`). In the code below, a set of examples partially represented in `dplyr` documentation (Wickham et al., 2020) is given showing the same process with and without pipes.

```
a <- mutate(mtcars, gpm = 1/mpg)
b <- arrange(a, gpm)
#with pipes, the above becomes:
c <- mtcars %>% mutate(gpm = 1/mpg) %>% arrange(gpm)
```

In both cases, the original dataset was used to create a new variable, and the rows of

the expanded dataset were arranged by increasing order for the new variable. The first example requires saving multiple individual functions and changing the object being referenced. In a short series, this makes little difference compared to the second example; however the format with pipes will remain consistent with additional steps. This process could be expanded to include significantly more verbs, and with pipes, the end state of the process would be easily carried forward. The design of `tidyTouch` incorporates many `Tidyverse` package functions, as well as the use of pipes, to familiarize students of R with both the operations and format that they will find consistent in work across the data science field.

### ***Shiny: Web Applications in R***

Web applications (apps) are designed to make useful programs accessible online, able to be used freely without the hassle of installation processes. R can be used to create web apps with the help of the `Shiny` package (Chang, Cheng, Allaire, Xie, & McPherson, 2019), which converts R instructions to the type of code needed to design and format web pages. By using `Shiny` functions, an R programmer can create a web app that responds to user interaction by providing that user's input to a pre-built R program. This is useful for providing data and intended manipulations to non-programmers through educational tools (Chang et al., 2019; RStudio Team, 2020). With the main challenge in choosing R as an analysis tool being its ease of use, using web apps can serve as a transitional tool by introducing the concepts and terms involved in the analysis without students needing full understanding of the underlying code. The discussed target problems around learning and using programs for working with data, namely accessibility, convenience and simplicity of use, and appropriate contributions to an individual's experience in developing and understanding of R, have attempted to be addressed by this project, `tidyTouch`, a web application written entirely in R that provides a GUI for simple data transformation and



visualization using `ggplot2` and other Tidyverse packages.

## Design and Development

The `tidyTouch` app was developed using `Shiny` to give an intuitive platform capturing the steps necessary for working with the R language for data visualization. The design will be discussed in the order of a standard workflow, showing the relevance of individual packages in the steps necessary for data visualization. These steps include importing data, visually evaluating the data structure, transforming the data as needed, and generating a graphic representing chosen data points. It is important to mention the status of this project being a work in progress, which is discussed in detail under the “Status and Future Development” subheading.

### Importing and Viewing Data

Users of `tidyTouch` are met with an interface including three tabs: Data, Plot, and Code. The process starts on the Data tab’s panel, where a source of data is selected from a drop-down menu. Selected datasets, all of which accessible by any user of base R, were included as options to allow exploration of `tidyTouch` functionality without needing to provide an original source. The menu also provides the “Import Dataset” option, which displays a menu for choosing between CSV, TSV, Excel, SAS, SPSS, and Stata file types. With the file type selected, a “browse” button is used to open a window for selecting data from the user’s personal system. A checkbox option for “Header” is automatically selected, but can be unchecked should a dataset not have variable names as the first row of entries. Once a source has been selected, the data can be viewed in a table with that provides abilities to search for and sort values. On the panel, a user can view the `head` - the first few lines of the dataset - or the full file. The final viewing feature is a window that prints the

structure (`str`) of the data, which contains information about the types of values in the table.

The intentions for these features were to provide options that any user may be familiar with or will be necessary to learn in R. In many tutorials or guides, such as the RStudio Primers or *Cookbook for R*, preloaded datasets are used to demonstrate modifications to code (Chang, 2013; RStudio Team, 2020). A variety of these common datasets were included, in addition to allowing exploration of the app, to maintain consistency with examples student of R are likely to encounter. The file types chosen are likely to be used by those working with other analysis tools, and their selection in the menu utilizes functions from the `haven`, `readr`, and `readxl` packages (Wickham & Bryan, 2019; Wickham, Hester, & Francois, 2018; Wickham & Miller, 2019). The “Header” checkbox includes an important argument for many of these functions that is necessary for variables to be recognized correctly. Code from the `reactable` package (Lin, 2019) was used to support the point-and-click interactivity for viewing data, limiting the amount of transformation necessary for a new user to examine their data. The `head` option is set as a default to prevent loading large datasets to interfere with the usability of the app. Using the `head()` function is more useful in a development environment, where printing a dataset in its entirety serves little purpose for a programmer. Inclusion of the `head` option serves to improve the performance of the application, while familiarizing students with terminology involved in typical practice of more advanced R users.

## Transformations

Being able to alter a dataset for sorting or creating variables is necessary for any work with data. These features are implemented in any spreadsheet editing program, like Microsoft Excel’s use of “formulas” (Microsoft, 2019). Even with a significantly more complex range of possible alterations available through `dplyr` and `tidyr` (Wickham et al.,

2020; Wickham & Henry, 2020), only a few options are made available to keep the app’s functionality intuitive for those with limited experience generating or manipulating data. Variable types can be changed to indicate their status as numeric, categorical, or strings of characters (such as names). The dataset can also be filtered by selecting only certain values for a given variable, condensing data to a target range for analysis. Grouping information with a specified variable does not condense the data, but allows future operations to be applied within groups. This feature can be used in conjunction with other transformation functions that are discussed under “Future Status and Development.” Functions from `dplyr` and base R are used for these features (R Core Team, 2020; Wickham et al., 2020).

## Visualization

Creating graphics using `tidyTouch` involves selecting attributes from a series of menus on the Plot tab. Users can choose to create visualizations for one or two variables, with all variables from the chosen dataset being listed in drop-down menus. The type of geometric object for the plot is chosen, with the options changing based on the number of variables being observed. Theme changes and faceting are also controlled in this panel, with each change manifesting in a graphic on the main section of the display. As mentioned, only the data represented by the plot and a layer are necessary to create plot; the theme and faceting options are not set to be included unless otherwise specified. Below the graphic, additional components are nested within tabs, where a user can add labels, such as a title or axis names, or modify the plot’s aesthetics, such as color, size and transparency.

The options available are limited to commonly used graphs and customizations in `ggplot2`. All terms used for layers, themes, faceting types, and aesthetic specifications match the package’s code that would be used in a development environment to create identical graphics. This familiarizes a user with the terminology used in `ggplot2` to enhance the utility of `tidyTouch` as a *transitional* learning tool, so that practice from the

environment of the app is transferrable to writing R code. Under aesthetic specifications, the options available are dependent on the chosen geometric object to only allow user to make changes appropriate for the type of graph being created. Conditional displays are used with faceting, as well as the tabs underneath the graphic to minimize clutter from these expanding components when not in use.

### ***Shiny* Reactivity and Code Printing**

Reactivity refers to the dynamic changes demonstrated by `tidyTouch`, with interaction altering the interface. This is apparent in cases of some menu choices providing additional options, or changing the contents of other menus - how different geometric objects and aesthetic specifications are made available. **Shiny** reactivity also allows input by a user to generate components, displaying information such as variable names in menus and input modules. This aspect of **Shiny** allows the selections of the app user to determine which elements of a `ggplot2` graphic's code to use when creating the visualization; extra information or missing values would both prevent any graphic from being generated. By determining which code to use for the visualization, the app can print the most simple form of code that a student would use to recreate their visualization in the RStudio IDE.

Giving students of R an opportunity to make graphics with a point-and-click interface, while providing the code necessary for the process, `tidyTouch` aims to provide a platform for learning about data visualization with R to those without programming experience. Relying on reactive **Shiny** components, however, the operations discussed require a significantly more complex design than one would need in writing R scripts to accomplish the same results. The example below shows how reactive **Shiny** programming differs from the equivalent R code.

```
# standard R
data <- airquality
data %>% ggplot(aes(x = Day, y = Solar.R)) + geom_line()
```

```
# reactive Shiny (condensed)
reactive_data <- reactive({...else if(input$data_source == "airquality"){
  airquality}...})           # "airquality" selected in data source menu
reactive_geom <- reactive ({...else if(input$geom == "Line"){
  geom_line()}...})          # "Line" selected in geom menu
reactive_x <- reactive({input$x_variable}) # "Day" variable selected
reactive_y <- reactive({input$y_variable}) # "Solar.R" variable selected
reactive_data() %>% ggplot(aes(x = reactive_x(), y = reactive_y()))+
  reactive_geom()             #reactive Shiny graphic code
```

This complexity poses a significant challenge to a Shiny web app's contribution to data science education. The developers of Shiny (Chang et al., 2019) recognize the potential utility of these web applications as teaching tools for the R language itself, and a package currently under development attempts to address this problem. The `shinymeta` package allows a programmer to translate code from a Shiny application to the format that would be used directly in R (Sievert, 2019). As tidyTouch displays code for a plot that has been created, `shinymeta` makes readable code that is useful to a beginner's learning process.

still working with code to see where to include section on `shinymeta` with demonstration; will likely be placed here

## Discussion

### Status and Future Development

### Applications and Audiences

### R Packages and Session Info

To recognize those that contribute to R, tools used by members of the R community, and the continually developing field of data science, the software used in creating the tidyTouch app is listed: R (Version 3.6.3; R Core Team, 2020) and the R-packages \*  
cowplot\* [ @ R-cowplot ], *dplyr* (Version 0.8.5; Wickham et al., 2020), *ggplot2* (Version 3.2.1; Wickham, 2016), *haven* (Version 2.1.1; Wickham & Miller, 2019), *papaja* (Version 0.1.0.9942; Aust & Barth, 2020), *reactable* (Version 0.1.0; Lin, 2019), *readr* (Version 1.3.1; Wickham et al., 2018), *readxl* (Version 1.3.1; Wickham & Bryan, 2019), *rmarkdown* (Version 2.1; Xie, Allaire, & Grolemund, 2018), *shiny* (Version 1.4.0.9000; Chang et al., 2019; Chang, 2018; Sievert, 2019), *shinymeta* (Version 0.2.0; Sievert, 2019), *shinythemes* (Version 1.1.2; Chang, 2018), and *tidyr* (Version 1.0.2; Wickham & Henry, 2020). This document was created using *papaja* and *rmarkdown* (Aust & Barth, 2020; Xie et al., 2018). Documentation for individual packages may include links to additional educational resources, like the Tidyverse cheat sheets from RStudio.

The session info for this project in its current state - the R version and additional loaded packages used during the development of this app, as well as the generation of this document - is printed below.

```
## R version 3.6.3 (2020-02-29)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 18.04.4 LTS
```

```
##

## Matrix products: default

## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.7.1

## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.7.1

##

## locale:

##  [1] LC_CTYPE=en_US.UTF-8      LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8      LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8  LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8     LC_NAME=C
##  [9] LC_ADDRESS=C             LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

##

## attached base packages:

## [1] stats      graphics  grDevices  utils      datasets  methods    base

##

## other attached packages:

##  [1] cowplot_1.0.0      rmarkdown_2.1      reactable_0.1.0    haven_2.1.1
##  [5] tidyr_1.0.2        readxl_1.3.1       readr_1.3.1        shinythemes_1.1.2
##  [9] shinymeta_0.2.0    shiny_1.4.0.9000   dplyr_0.8.5        ggplot2_3.2.1
## [13] papaja_0.1.0.9942

##

## loaded via a namespace (and not attached):

##  [1] styler_1.2.0      tidyselect_1.0.0   xfun_0.13          purrr_0.3.4
##  [5] colorspace_1.4-1  vctrs_0.2.4        sourcetools_0.1.7  htmltools_0.4.0
##  [9] yaml_2.2.1        rlang_0.4.5        pillar_1.4.3       later_1.0.0
## [13] glue_1.4.0        withr_2.1.2        lifecycle_0.2.0    stringr_1.4.0
```

```
## [17] munsell_0.5.0      gtable_0.3.0      cellranger_1.1.0  htmlwidgets_1.5.1
## [21] evaluate_0.14      labeling_0.3       forcats_0.4.0     knitr_1.28
## [25] fastmap_1.0.1      httpuv_1.5.2      fansi_0.4.1       highr_0.8
## [29] Rcpp_1.0.4.6       xtable_1.8-4      scales_1.0.0      promises_1.1.0
## [33] backports_1.1.6    mime_0.9          hms_0.5.1         digest_0.6.25
## [37] stringi_1.4.6      bookdown_0.18     grid_3.6.3        cli_2.0.2
## [41] tools_3.6.3        magrittr_1.5      lazyeval_0.2.2    tibble_3.0.0
## [45] crayon_1.3.4       pkgconfig_2.0.3   ellipsis_0.3.0    assertthat_0.2.1
## [49] R6_2.4.1           compiler_3.6.3
```



## Screenshots

## References

- Aust, F., & Barth, M. (2020). *papaja: Create APA manuscripts with R Markdown*. Retrieved from <https://github.com/crsh/papaja>
- Chang, W. (2013). *R graphics cookbook*. (M. Loukides & C. Nash, Eds.). O'Reilly Media, Inc.
- Chang, W. (2018). *Shinythemes: Themes for shiny*. Retrieved from <https://CRAN.R-project.org/package=shinythemes>
- Chang, W., Cheng, J., Allaire, J., Xie, Y., & McPherson, J. (2019). *Shiny: Web application framework for r*. Retrieved from <http://shiny.rstudio.com>
- IBM. (2017). SPSS, version 25.0. Retrieved from <https://www.ibm.com/analytics/spss-statistics-software>
- Lin, G. (2019). *Reactable: Interactive data tables based on 'react table'*. Retrieved from <https://CRAN.R-project.org/package=reactable>
- Microsoft. (2019). Microsoft excel, version 16.0.12819.37950. Retrieved from <https://products.office.com/en-us/excel>
- Minitab. (2020). Minitab statistical software, version 19.2020.1. Retrieved from <http://www.minitab.com/en-us/products/minitab/>
- Murray, S. (2017). *Interactive data visualization for the web*. (M. Foley, Ed.) (2nd ed.). O'Reilly Media, Inc.
- Open Source Initiative. (2020). OSI: Open source definition. Retrieved from <https://opensource.org/docs/osd>

- Picciano, A. G. (2012). The evolution of big data and learning analytics in american higher education. *Journal of Asynchronous Learning Networks*, 16(3), 9–20.
- R Core Team. (2020). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from <https://www.R-project.org/>
- RStudio Team. (2020). *RStudio: Integrated development environment for r*. Boston, MA: RStudio, Inc. Retrieved from <http://www.rstudio.com/>
- Sievert, C. (2019). *Shinymeta: Record and expose shiny app logic using metaprogramming*.
- Wickham, H. (2016). *Ggplot2: Elegant graphics for data analysis*. Springer-Verlag New York. Retrieved from <https://ggplot2.tidyverse.org>
- Wickham, H. (2017). *Tidyverse: Easily install and load the 'tidyverse'*. Retrieved from <https://CRAN.R-project.org/package=tidyverse>
- Wickham, H., & Bryan, J. (2019). *Readxl: Read excel files*. Retrieved from <https://CRAN.R-project.org/package=readxl>
- Wickham, H., François, R., Henry, L., & Müller, K. (2020). *Dplyr: A grammar of data manipulation*. Retrieved from <https://CRAN.R-project.org/package=dplyr>
- Wickham, H., & Henry, L. (2020). *Tidyr: Tidy messy data*. Retrieved from <https://CRAN.R-project.org/package=tidyr>
- Wickham, H., Hester, J., & Francois, R. (2018). *Readr: Read rectangular text data*. Retrieved from <https://CRAN.R-project.org/package=readr>
- Wickham, H., & Miller, E. (2019). *Haven: Import and export 'spss', 'stata' and 'sas' files*. Retrieved from <https://CRAN.R-project.org/package=haven>

Wilkinson, L. (2005). *The grammar of graphics* (2nd ed.). Springer Science & Business Media.

Williamson, B. (2017). *Big data in education*. (J. Clark, Ed.). SAGE Publications Inc.

Xie, Y., Allaire, J., & Grolemund, G. (2018). *R markdown: The definitive guide*. Boca Raton, Florida: Chapman; Hall/CRC. Retrieved from <https://bookdown.org/yihui/rmarkdown>