Building an ISCAM Package Using Best Programming Practices in R

Noelle Pablo

June 5, 2018

The introductory statistics textbook Investigating Statistical Concepts, Applications, and Methods (ISCAM) written by Beth Chance and Allan Rossman, introduces students to fundamental concepts and various applications of statistics. Accompanying the textbook are a variety of technological tools, including a collection of R functions used to graph probability distributions, perform various hypothesis tests, calculate summary statistics, and much more. Currently the functions are written in base R and require no additional packages for implementation and reside in an R workspace. This senior project involves updating the ISCAM functions to involve the use of functions from different packages, namely ggplot2, to enhance the visual representations and improve the documentation of the existing code. To do this, it is crucial to learn and discuss what are considered to be the best programming practices in R. This project also intends to place all the updated functions into an R package of its own, in hopes of providing more thorough documentation, organization, and ease of reproducibility for the ISCAM functions.

The twenty-six R functions accompanying the ISCAM textbook provide students with technological methods to easily conduct hypothesis tests, create visual representations and calculate summary statistics of their data, graph probability distributions according to specified inputs, determine rejection regions and power, approximate distributions, compute probabilities according to various probability distributions, and calculate quantiles of a desired probability. While I did utilize the ISCAM textbook as a first-year student at Cal Poly, I found it necessary to review some of the statistical topics and methods present demonstrated by the functions, so that I could better understand the existing code. The inputs of a hypergeometric probability mass function, when to use a continuity correction, power and its relation to the rejection region were some of the statistical topics I found myself reviewing during this project.

Currently, the ISCAM functions are stored together in an R workspace. Workspaces allow various types of objects, including functions, to be saved and reloaded into R for future use and can be shared between users. Installing an R workspace requires the user to retrieve it from whoever made it and download it onto his or her own computer, then open it up in R. Currently, students utilizing the ISCAM textbook can retrieve the workspace containing all the ISCAM functions through the RossmanChance website. While a workspace serves as an uncomplicated method of saving and sharing code for functions and other objects, workspaces offer little to no options for function documentation. Documentation involves what the function does, more detailed explanations of the required inputs, the function's output, and some examples using the function. Function documentation is necessary for users who are unsure of the function's purpose or what inputs it requires to get the desired output. It is extremely helpful even for the function creator, when the function is no longer fresh in her memory and she needs a quick reminder of how a certain function is used and what it needs in order to run. To access a brief description about the purpose of the function, the user must type a question mark as the sole input in an ISCAM function, for example, `iscamaddexp("?")`. This command returns an error to the R console, with the function description as its error message. This informal type of documentation lacks organization, readability, and function examples. This issue, along with the desire to update the code to adhere to best programming practices in R, was my leading motive to build a package to hold all the ISCAM functions, rather than in a workspace.

R functions created to solve specific problems tend to be held together in a package, rather than a workspace. An R package is a collection of R functions with accompanying documentation, detailing how each function is used. R packages can contain data sets as well as other user-defined objects. A major benefit of packages is the ease of distribution, allowing others to easily reuse the functions and use them in their own projects. There have been many occurrences when I am faced with a problem and do not know the best way to proceed, only to find that someone has already created a function within a package designed to solve it.

The Comprehensive R Archive Network (CRAN) currently has over 12,000 packages, free for any R user to download. Popular packages available in CRAN are ggplot2, a package of graphing functions which I predominately used in this senior project, dplyr, functions used for data manipulation, and devtools, another package I used in this project to develop my own R package. To install a package from CRAN, ggplot2 for example, the user simply needs to type `install.packages(ggplot2)`.

The documentation of R functions downloaded from packages can be easily accessed by using the help function or typing a question mark in front of the function name, for example `help(mean)` or `?mean` into the R console. Some packages also contain a long-form of documentation called a vignette. Vignettes are supplemental documentation that serve as a guide to a package and offer details about its purpose and the functions it contains. Within the ISCAM vignette is a description of the ISCAM package and its purpose, a list of all the ISCAM functions in their corresponding categories, and various examples. To access the vignette, the following code is used: `vignette(ISCAM)`. Through packages, sharing R functions and the documentation that explains how to use them becomes straightforward and simple.

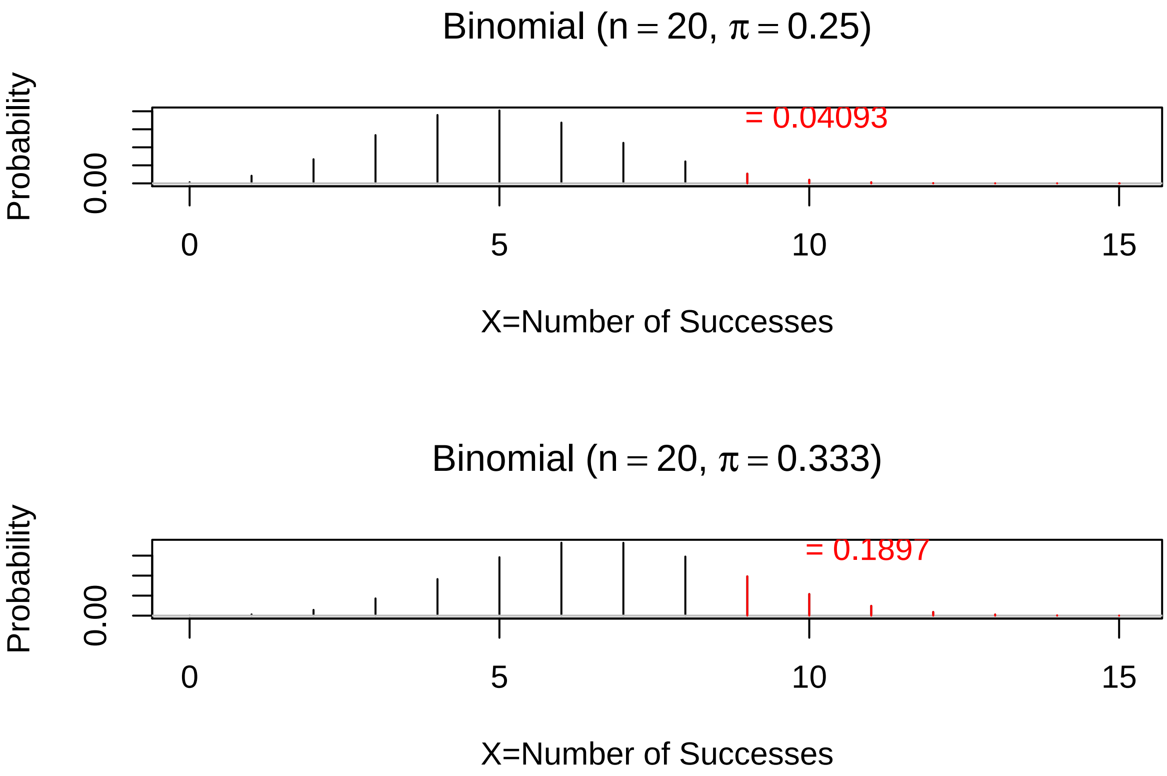
Aside from wanting to making the ISCAM functions easier to distribute and understand, a package so that the ISCAM functions are easier to distribute, an additional reason that I wanted to pursue this project is to bring attention the lack of female programmers making R packages. As a female STEM major, I have grown accustomed to being part of the minority in my programming courses. In my very first computer science course at Cal Poly, less than 10 of the 40 students were female. This glaring gender gap is not unique to my experiences; in 2016, approximately 15% of package authors in CRAN were female (Turner). To combat the notable absence of women in the package creator community and in the field of statistical computing as a whole, more effort needs to be made into changing the way women view their ability to contribute to creating technology, and this change needs to be made at a young age. When more women are involved in creating R packages, the entire R community is bound to benefit from the innovations, solutions, and creations that women can bring to the table.

Before transferring all the existing ISCAM functions into an R package, I wanted to update and potentially improve the code to adhere good programming practices in R. It is highly important to adhere to programming practices that enable R code to be readable, reproducible, and easily understood by others. To learn more about what transforms acceptable, working code into clean, consistent, and reusable code, I turned to Hadley Wickam, Chief Scientist at RStudio, for guidance. Wickam's style guide offers recommendations regarding the naming of files and objects, syntax, and organization (reference). Some notable recommendations that I had not considered before include using `<-`, instead of `=`, for assignment, using verbs when naming functions and nouns when naming variables, placing spaces around infix operators, using two spaces instead of the tab button when indenting code, and limiting a line of code to be approximately 80 characters long. I kept these guidelines in mind when rewriting and updating the ISCAM functions. I also made use of the "Writing Functions in R" course on the DataCamp website. This course features guidelines on writing functions in R, including what constitutes a "good" function. The course describes a good function as being "obviously correct" meaning that it is serves its intended purpose and is understandable not only by computers, but by humans as well.

The original ISCAM functions are all written in base R, meaning they do not include any functions from other packages. I used the ggplot2 package, also written by Hadley Wickam, to create more elaborate, and at times nicer-looking graphs than their counterparts made with R's default graphing functions. One benefit of ggplot is that it enables users to save their plots as objects, which is useful when creating multiple versions of a plot while avoiding the repetition of a lot of code. This allowed for easier-to-read and more organized code, which complied with Wickam's recommended programming practices. Figure 1 represents the "before" version of a graph made using the original function "iscambinompower", and Figure 2 represents the "after" version made using the updated function. In this updated version, I added a new argument, "explain", that color codes the Type I error, Type II error, and Power regions when set to TRUE. Overall, I updated and added documentation to all 26 of the ISCAM functions.

```r
iscambinompower(.05, 20, 0.25, alternative = "greater", 0.333)
```

```
## Probability 9 and above = 0.04092517
```



```
## Probability 9 and above = 0.1896621
```

Figure 1. iscambinompower plot, created using original function.

```
iscam_binompower(.05, 20, 0.25, alternative = "greater", 0.333, explain = T)
```

```
## Probability 9 and above = 0.04093
## Probability 9 and above = 0.1897
```

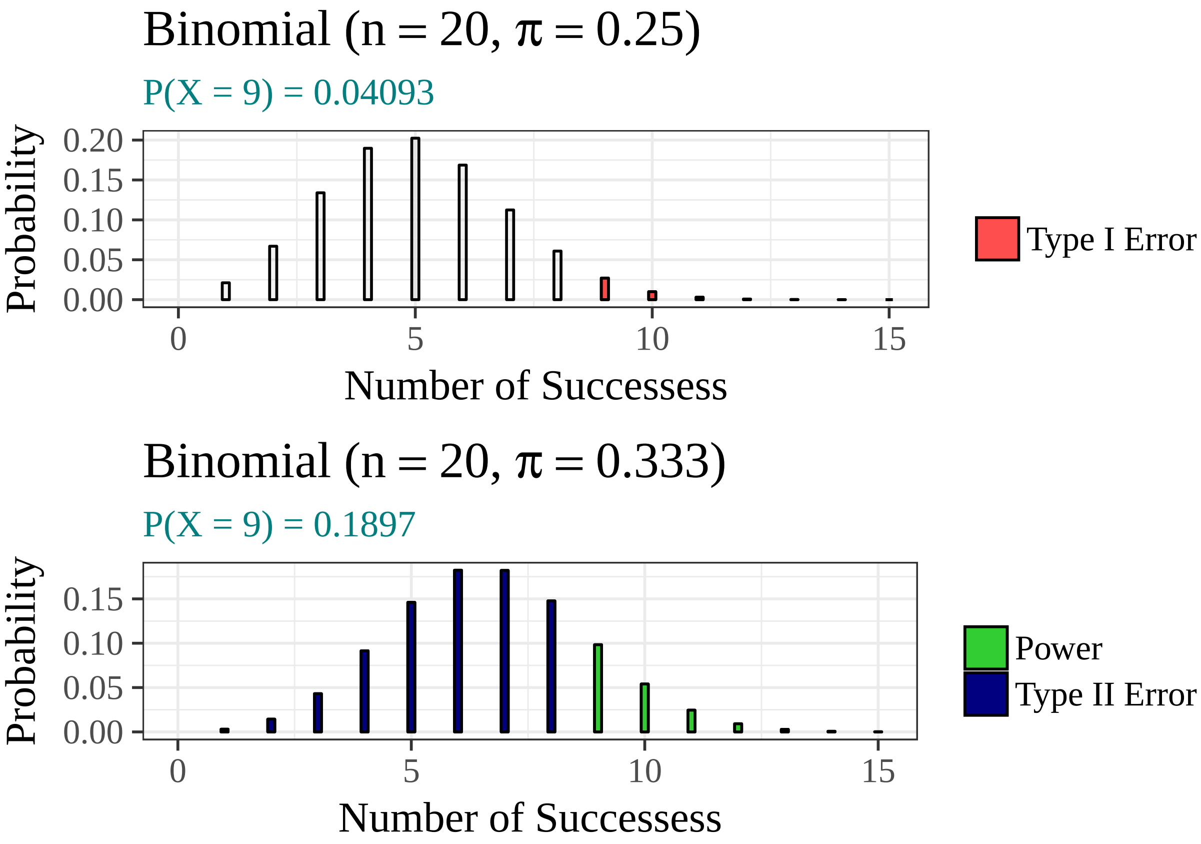


Figure 2. iscam_binompower plot, created using updated function.

While updating the ISCAM functions to adhere to better programming practices, I learned more about computational reproducibility. Computational reproducibility is defined as providing detailed information regarding code, software, hardware, and implementation ("Reproducibility Guide"). Its goal, which goes hand in hand with good programming practices, is to enable future researchers to utilize the same methods and recreate the results to eventually advance research while saving time and effort. The importance of reproducibility in programming and in the statistics discipline as a whole cannot be overstated. R facilitates reproducibility through its large collection of packages. A different form of reproducibility to which I was introduced during this project is version control. Version control allows programmers to save changes to a file over time, so that past versions may be easily obtained later. A widely-used version control tool is Google Drive. During the updating of the ISCAM functions and the creation of the ISCAM package, I made use of a common version control tool, Github. Github is one of the largest coding communities online that enables collaborators to track edits to code over time and just as easily undo any unwanted changes. Because I had never used Github before, I took the "Introduction to Git for Data Science" course on the DataCamp website, where I learned the power of Github as a version control tool and its ability to share changes within and between repositories. With the use of my own Github repository, I was able to easily collaborate with my senior project supervisor in the writing and editing of code. My Github repository contains all of the components of my resulting R package, and it can be accessed by anyone interested in viewing my code and potentially building upon and improving it.

After learning more about best programming practices in R, developing functions, and version control, it was finally time to put it all together into a package. Not knowing where to begin, I first read over data scientist Hilary Parker's blog post about creating an R package. With the use of Parker's guide, I was able to create my own package, add functions and documentation, and install it on my own computer in about five steps. My first inclination was to move all the original, existing functions inside the ISCAM workspace into an R package. After doing this, I created a second package to contain all of the updated ISCAM functions. To write a package from scratch, I needed to download two packages– devtools to create the package and roxygen2 to develop the documentation. After installing those two packages, I created a package directory then saved all the ISCAM functions to it and added documentation. The documentation contained the original description of the function, along with explanations of the inputs, output, and examples. Lastly I created the package's vignette to describe its purpose and a listing of all the ISCAM functions. I also noted the packages that are necessary in order to properly run the ISCAM package– ggplot2, skimr, and gridExtra. By adding the names of these packages into my package's description file, the user will not have to manually download them, rather they will be downloaded automatically with the installation of the ISCAM package. To download the updated ISCAM package, the following line of code is used: `devtools::install_github("shannonpileggi/SP--Pablo--RProgramming/ISCAM")`.

While I was able to meet this project's main objective of updating every ISCAM function in some way to adhere to the best programming practices in R and compiling them all into a package, there are further steps to consider in the future. As mentioned earlier, packages in CRAN, R's main repository for packages, can be easily installed using `install.packages("packagename")`. However, the code to install my package looks quite different. The reason is that my package is not currently available in CRAN. The process of getting a package on CRAN is lengthy and can be quite daunting for first-time package creators. Some requirements include the package containing no errors, warnings, or significant notes, examples should run for no more than a few seconds each, and the package's license must give the right for CRAN to distribute the package in perpetuity. An exhaustive list of policies required to submit a package for consideration can be found on the CRAN website.

With the completion of this project, I hope to encourage more students to write functions that make their programming endeavors easier and less time-consuming. Adhering to good programming practices while writing these functions will make code easier to read, understand, and reproduce. Including clear documentation to accompany the functions further facilitates the ease of discerning the purpose of a function and exactly how to use it. Saving these functions by storing them into a package allows for easier accessibility and distribution between R users. I strongly believe that this project helped me to grow as a programmer, develop my coding skills in R, and boost my confidence in my ability to create a valuable R package

References

"Reproducibility Guide." ROpenSci Project, ropensci.github.io/reproducibility-guide/sections/introduction/.

Turner, Heather. "Addressing the Gender Gap in the R Project." RPubs, 13 Oct. 2016, rpubs.com/hturner/eRum2016.