5-2022

Randomizer Utilizing R Shiny

Cali R. Savage
University of Nebraska Medical Center

Randomizer Utilizing R Shiny

Cali Savage, Biostatistics

Committee Information

Lynette M. Smith, PhD

Christopher Wichman, PhD

Maneesh Jain, PhD

Abstract

Currently, many published animal experiments fail to include concise information regarding randomization (Festing, 2014). By not including randomization efforts in published research, the ability for one to accurately reproduce a study's results is greatly diminished. This can be seen in randomization efforts that may be difficult to reproduce, such as poor randomization techniques and lack of documentation. To circumvent this, we created a user-friendly Randomizer utilizing software R to enable University of Nebraska Medical Center (UNMC) researchers, among others, for generating randomization schemes for research involving mice, other animal, or human subjects. The program operates in R, utilizing R Shiny, and has the ability for the user to define randomization aspects to tailor their randomization scheme while also creating a transparent, documentation-friendly paper trail.

Project Description

Specific Aim

The Randomizer aims to create a mechanism that will allow the researchers to randomize experimental subjects in a consistent, and methodical way within a web app for ease of use, while at the same time allowing for reproducible results and documentation.

Significance

The use of randomization tools is a common practice in research. The use of such tools diminishes selection and accidental bias (Suresh, 2011). Proper randomization is necessary to minimize the variability of the results. Utilizing a coded randomizer also creates a transparent method allowing for reproducibility and further analysis if needed. This project serves to fill a need at the University of Nebraska Medical Center (UNMC) by providing a method to randomize animals or human subjects used in research projects.

Background

Data shows that only approximately 10-40% (depending on sources) of published research is reproducible (Baker, 2016). However, this problem is not often brought to one's attention. This can be seen in how one survey conducted by Baker found that less than 20% of those with published research had ever been contacted by researchers attempting to recreate their methods due to un-reproducible work. However, this statistic may not be representative of the true amount of research that is un-reproducible due to researchers assuming their inability to reproduce methods was due to user error (Baker, 2016).

Another survey of 271 animal experiments showed that 87% did not disclose randomization measures within their published research (Festing, 2014). Although this does not directly indicate that those researchers had not randomized their subjects, rather, the information

was not easily available. This data shows a pain point within the reproducibility of published research. Although the context and methods of the experiment may be listed, without the randomization schema, bias cannot be ruled out in the data analysis. Therefore, including randomization efforts within published research will allow for a better understanding of the true nature in which the study was conducted.

These data points bring up the reality that much of the published research is not readily reproducible (Festing, 2014). Although there are many variables within animal research across research facilities, one thing that can be common is the documentation of critical parameters such as randomization (Festing, 2014).

Using a coded randomizer would create a consistently reproducible mechanism for documentation between users. If an experiment was having trouble being reproduced and randomization seemed at fault, the coded randomizer could be analyzed and dissected. Another advantage of the coded randomizer is that the results can be documented to show exactly how animal or human subjects were randomized in that study. Therefore, the Randomizer would create a transparent and unbiased method of randomization that can be reviewed and analyzed across researchers.

Previously UNMC, students would randomize their subjects in a plethora of ways depending on the researcher's choice. One example is a student who would place all their mice in one spot, place their hand in the middle of the pile, and then separate the group into two. Another example of current randomization efforts at UNMC is where tumor-bearing mice were randomized in such a way that at baseline each group will have a similar combination of small, medium, and large size tumors. The focus of this method is to eliminate the confounding effects of the burden of tumors on the efficacy of an intervention. Although random, this could be

difficult to reproduce in a truly unbiased manner and to document exactly how this was unbiased. The Randomizer would create a consistent, trackable method of randomization that all researchers within UNMC, and those more widely, could use. Thusly ensuring that experimental research contains documented randomization.

Methods

The Randomizer utilizes R along with the following R packages: shiny, shinythemes, blockrand, tidyr, and plyr. The usage of these packages creates a user-friendly experience where the number of subjects, number of randomization blocks, and other details can be entered to create a personalized randomization scheme. The packages used within the Randomizer have been assessed for long-term usability. It is important to note that if an R package is not maintained, R's future updates may create functionality issues that cause the Randomizer malfunction.

Both the shiny and shinythemes packages make up the user interface for the Randomizer. The R Shiny (also called Shiny) package within R is used to build interactive web applications (Shiny, 2020). Shiny allows users to create interactive web pages that can be coded to perform a multitude of tasks utilizing the R language. With this randomizer, R Shiny holds the web interface while also running the background code based on the selection the user makes in the front-end. The shinythemes package creates the layout and the functionality of the Randomizer. The default bootstrap theme is created via the shinythemes packages (Chang et al., 2021). This layout includes a floating sidebar, a main page, and several tabs at the top of the page.

The blockrand, tidyr, and plyr packages make up the actual randomization method within the Randomizer. The blockrand package makes up the randomization function for both the stratified and block randomization methods. This package is meant to use the blockrand function

on each stratum and then to combine the data frames using rbind while pursuing a stratified randomization scheme (Snow, 2020). When using blocked randomization, the blockrand function is used once for the entire schema. The tidyr package functions to tidy messy data and pivot as needed (Wickham and Girlich, 2022). The plyr package is responsible for breaking down the data frames using the count function (Wickham, 2020). This allows for the creation of the final count tables allowing for better user readability.

The R code used for the Randomizer will be passed on to the Center for Collaboration on Research Design and Analysis (CCORDA) and maintained. Once completed, the Randomizer was tested by various researchers to ensure functionality and user experience. A Google form has been included in the "Information" tab of the Randomizer allowing users to leave feedback.

The Randomizer covers three methods of randomization: simple, block, and stratified. Simple randomization pertains to the use of a single sequence for randomization (such as flipping a coin or a random number table) (Kang, Ragan, and Park, 2008). This method is preferred for larger sample sizes as this will result in similar numbers in each group with smaller sample sizes being avoided as unequal group sizes may result. The method of simple randomization is important as it creates true randomization between a subject and their assigned group (Kang, Ragan, and Park, 2008).

The simple randomization method is programmed in the Randomizer to allow the user to name their study, enter in the number of subjects to randomize, enter in a seed (a user set pin), specify their probability from 0.01-1, and name their groups. Users can also download the information needed to recreate their randomization scheme along with their randomization table.

The simple randomization method is programmed using the following structure:

```
Function_name <- function(seed, ncases, pvalue, ID_prefix){
set.seed(seed)
p <- pvalue_simp
```

```
n <- ncases_simp
data <- data.frame(id = paste(input$ID_prefix, 1:input$ncases,
sep = ""), trt = c("og"))
test <- head(data)
u <- runif(input$ncases)
one <- (data$trt[u<input$pvalue] = input$group1_name)
two <- (data$trt[u>input$pvalue] = input$group2_name
idea <- data[]
tot <- count(idea, vars = "trt")
```

Here, the function is created where the seed, number of subjects, p-value, and the specified ID prefix is brought in. The p-value and number of subjects are set as variables within the function. Then, a data frame is created with the subjects specified. This data frame is then randomized against the p-value and put into the two separate groups then brought back into one data frame. From here, the randomization table is created. Lastly, the count function is optional and is used to create the overall total table of the number of subjects within each group.

Block randomization pertains to the use of groups during randomization to ensure equal numbers within all groups (Kang, Ragan, and Park, 2008). Block sizes are predetermined and are routinely kept small to ensure equal numbers. Block sizes can be either fixed or random. Fixed block sizes will result in equal numbers among groups but maybe predictable. Random block sizes may create extra security among predictability but may have negative effects on the study (Dixon, 2016). This method is helpful as it is more likely to create equal numbers within groups but does not control for covariates which may introduce bias into the study.
Kang, Ragan, and Park, 2008).

The block randomization method is programmed in the Randomizer to allow the user to name their study, enter in the number of subjects to randomize, enter in a seed, specify the number of subjects, specify their number of groups (up to ten), name all groups, specify a subject ID prefix, and chose the block size to use (both fixed and random options). Users can also

download the information needed to recreate their randomization scheme along with their randomization table.

The blocked randomization method is programmed using the following structure:

```
Function_name <- function(seed, ncases, levels, blocksize){
set.seed(seed)
lev <-levels
blockrandmeth <- blockrand(n=(input$ncases), num.levels=2,
levels = c(input$lev1_name, input$lev2_name), block.sizes = (2),
id.prefix=input$lev2_prefix, block.prefix='Block')
count(blockrandmeth, vars = "treatment")
```

Here, a function is created bringing in the information specified by the user within the R Shiny app. The seed and number of levels are set as variables within this function. Then, using the blockrand function from the Blockrand package, the subjects are randomized across different levels. This creates the final randomization table. The count function is optional and is used to create the overall count table where users can see how many subjects have been randomized into the different groups.

It is important to note that the blockrand function may result in more randomized subjects than originally specified. This is because depending on the number of subjects, groups, and the block size chosen, there needs to be equal numbers of subjects in all groups. Therefore, to achieve this equal status, more subjects may be randomized. A warning note has been added to the top of the page to draw attention to this so that users can remove extra subjects manually.

Lastly, the stratified method pertains to the creation of strata based on covariates in which participants are placed and then randomized within those strata using simple or blocked randomization (Kang, Ragan, and Park, 2008). Strata of different combinations of the covariates are then assigned to subjects based on the covariates they possess. After subjects have been placed into their strata, each strata goes through block randomization (Kang, Ragan, and Park,

2008). This method allows one to control for the covariates while also creating equal numbers within groups.

The stratified randomization method is programmed in the Randomizer to allow the user to name their study, enter the seed and number of subjects to randomize, specify the number of subjects, specify their number of levels and groups (up to ten each), name all levels and groups, specify a subject ID prefix, and chose the block size to use (both fixed and random options). Users can also download the information needed to recreate their randomization scheme along with their randomization table.

The stratified randomization method is programmed using the following structure:

```
Function_name <- function(seed,  ncases1, ncases2, ncases3,
strata, levels, blocksize){
set.seed(seed)
strat <- strata
lev <-levels
strata1 <- blockrand(n=input$ncases1, num.levels=2, levels =
     c(input$lev1_name, input$lev2_name), block.sizes = (2),
     id.prefix=input$st1n,
     block.prefix='Block',stratum=input$st1)
strata2 <- blockrand(n=input$ncases2, num.levels=2, levels =
     c(input$lev1_name, input$lev2_name), block.sizes = (2),
     id.prefix=input$st2n, block.prefix='Block',
     stratum=input$st2)
treatment <- rbind(strata1,strata2)
tot <- count(treatment, vars = "treatment")
```

Here, a function is created bringing in information specified by the user within the R Shiny app. The seed and number of levels are set as variables within this function. Then, using the blockrand function from the Blockrand package, the subjects are randomized across the different levels for each stratum. The data frames of the different stratum are then combined. This creates the final randomization table. The count function is optional and is used to create the overall count table where users can see how many subjects have been randomized into the different groups. A warning note is located at the top of the page warning users that due to the

nature of the blockrand function, more subjects may be returned than originally requested, in order to create equal numbers across groups.

The Information tab within the Randomizer contains hyperlinks for both a Google Form to collect feedback and an information document. The information document contains most of this paper to provide feedback on how the Randomizer functions. Real-world examples are also included to help guide users to visually understand how their study could involve the Randomizer. Lastly, the layout of each randomization tab is laid out with details as to what each field is collecting. This tab allows users to interact more with the application in that they can both seek guidance and provide feedback for future renditions.

Results

The Randomizer concludes with approximately 6,500 lines of code of the R language. The capabilities of the application are that users can randomize their subject pools using the stratified, block, and simple randomization methods. The number of subjects is unlimited; however, the randomization method is limited to up to ten levels and groups, and the blocked randomization is limited to up to ten groups.

As mentioned before, the blockrand package that is utilized in both the stratified and block randomization methods will possibly return more randomized subjects than previously specified. This is due to the package wanting to ensure there are equal numbers amongst each group. A warning message has been added to the top of both pages to alert users of this stipulation and to suggest the external removal of these extra subjects.

To create the functionality needed for a fully customizable randomization method, the Randomizer's code contains many duplicated functions. This is due to R Shiny needing unique names for all fields on the sidebar. Therefore, every possible choice needed a unique name, and

every possible combination needed a unique function. Also, to create the overall randomization table and the final count table, the code for the randomization table was duplicated and had a count function added to the end.

Discussion

The Randomizer successfully creates an application that users can use to quickly randomize their subjects in three different ways. There are stipulations with each method as mentioned above. However, the inclusion of an information or "help" document allows users to fully understand how the Randomizer should be utilized.

There are many strengths to the Randomizer. One being is its fully customizable programming. The user can create the randomization schema that is needed for their experiment. The Randomizer also allows the user to download the information needed to replicate this randomization method and randomization table which is useful in proper documentation and user experience. Lastly, this application will create a standard and central method of randomization within UNMC. The creation of this documentation will create a basis of transparency to improve reproducibility within research.

The major weakness of the Randomizer is the risk, as mentioned prior, that the blockrand package could return with more randomized subjects than requested by the user. This possibly can create an extra step for users that are not handled by the application. Also, as it is not handled by the Randomizer, the removal of extra subjects is not included in the downloadable documentation. This could lower the transparency and availability of documentation for a user's study.

The recommended sustainability plan is that once a year the Randomizer is tested by the UNMC Biostatistics department or CCORDA to ensure it is working properly. If a package

needs to be updated due to an update in the R software, this will need to be managed by the biostatistics department.

In conclusion, the Randomizer creates a user experience that allows researchers to randomize subjects visually while using a standard method that also populates documentation. This will create a standard method of randomization for a variety of experiment types. Lastly, the Randomizer creates as user-friendly experience that is easy for all levels of researchers to use.

Literature Cited

Baker, M. (2016). 1,500 scientists lift the lid on reproducibility. *Nature*, *533*(7604), 452-454. https://doi.org/10.1038/533452a

Bello, N. M., & Renter, D. G. (2018). Invited review: Reproducible research from noisy data: Revisiting key statistical principles for the animal sciences. *Journal of Dairy Science*, *101*(7), 5679-5701. https://doi.org/https://doi.org/10.3168/jds.2017-13978

Chang. W. (2021, January 25). Package 'shinythemes'. *CRAN.* https://cran.r-project.org/web/packages/shinythemes/shinythemes.pdf

Dixon, P. M. (2016). Should blocks be fixed or random.

Festing, M. F. W. (2014). Randomized Block Experimental Designs Can Increase the Power and Reproducibility of Laboratory Animal Experiments. *ILAR Journal*, *55*(3), 472-476. https://doi.org/10.1093/ilar/ilu045

Hirst, J. A., Howick, J., Aronson, J. K., Roberts, N., Perera, R., Koshiaris, C., & Heneghan, C. (2014). The Need for Randomization in Animal Trials: An Overview of Systematic Reviews. *PLOS ONE*, *9*(6), e98856. https://doi.org/10.1371/journal.pone.0098856

Kang, M., Ragan, B. G., & Park, J.-H. (2008). Issues in outcomes research: an overview of randomization techniques for clinical trials. *Journal of athletic training*, *43*(2), 215-221. https://doi.org/10.4085/1062-6050-43.2.215

Snow, G. (2020, April 6). Package 'blockrand'. *CRAN*. https://cran.r-project.org/web/packages/blockrand/blockrand.pdf

Suresh, K. (2011). An overview of randomization techniques: An unbiased assessment of outcome in clinical research. *J Hum Reprod Sci*, *4*(1), 8-11. https://doi.org/10.4103/0974-1208.82352

van Eenige, R., Verhave, P. S., Koemans, P. J., Tiebosch, I. A. C. W., Rensen, P. C. N., & Kooijman, S. (2020). RandoMice, a novel, user-friendly randomization tool in animal research. *PLOS ONE*, *15*(8), e0237096. https://doi.org/10.1371/journal.pone.0237096

Wickham, H. (2020, March 3). Package 'plyr'. *CRAN*. https://cran.r-project.org/web/packages/plyr/plyr.pdf

Wickham, H., & Girlich, M. (2022, February 1). Package 'tidyr'. *CRAN.* https://cran.r-project.org/web/packages/tidyr/tidyr.pdf

Application Of Public Health Competencies

The Randomizer embodies the MPH Foundational Competency "Evidence-based Approaches to Public Health." The capstone integrates this competency in that it uses R, a computer-based programming software used in qualitative and quantitative data analysis. This competency is also integrated in that it creates the randomization scheme needed in data collection.

The Biostatistics concentration competency BIOSMPH2 is also focused upon in the capstone project. BIOSMPH2's competency focuses upon applying statistical methods using a software package for statistical analysis. This capstone utilizes BIOSMPH2 by using the R software to prepare randomization schemes for research studies.

The Randomizer also applies the Biostatistics concentration competency BIOSMPH3. BIOMPH3's competency focuses upon applying statistical methods for quality control. The Randomizer applies this competency by creating a program using R to randomize subjects consistently allowing for quality control across studies within UNMC.

Supervision And Facilities

The Randomizer is built using the free R software under supervision of the committee. If problems arise with the maintenance of the program, those within the biostatistics department of UNMC will be able to troubleshoot along with the author of the program. Future applications of the Randomizer could involve expanding the programming to allow for the randomizations of humans.

Human Subjects

The Randomizer does not require human subjects and IRB review is not needed.