Current state and prospects of R-packages for the design of experiments

Emi Tanaka *
Department of Econometrics and Business Statistics
Monash University
Clayton, VIC 3800
emi.tanaka@monash.edu

Dewi Amaliah
Department of Econometrics and Business Statistics
Monash University
Clayton, VIC 3800

December 14, 2022

Abstract

Re-running an experiment is generally costly and, in some cases, impossible due to limited resources; therefore, the design of an experiment plays a critical role in increasing the quality of experimental data. In this paper, we describe the current state of R-packages for the design of experiments through an exploratory data analysis of package downloads, package metadata, and a comparison of characteristics with other topics. We observed that experimental designs in practice appear to be sufficiently manufactured by a small number of packages, and the development of experimental designs often occurs in silos. We also discuss the interface designs of widely utilized R packages in the field of experimental design and discuss their future prospects for advancing the field in practice.

1 Introduction

The critical role of data collection is well captured in the expression "garbage in, garbage out" – in other words, if the collected data are rubbish, then no analysis, however complex it may be, can make something out of it. Therefore, a carefully crafted data-collection scheme is critical for optimizing the information from the data. The field of experimental design is specifically devoted to planning the collection of experimental data, largely based on the founding principles of Fisher (1935) or an optimization framework like those described in Pukelsheim (2006). These experimental designs are often constructed with the aid of statistical software such as R (R Core Team 2021), Python (Rossum 1995), and SAS (SAS Institute 1985); thus the use of experimental design software can inform us about some aspects of experimental designs in practice.

Methods for data collection can be dichotomized by the type of data collected – namely, experimental or observational – or alternatively, categorized as experimental design (including quasi-experimental design) or survey design. This dichotomization, to a great extent, is seen in the Comprehensive R Archive

---

*Corresponding author

Network (CRAN) task views (a volunteer maintained list of R-packages by topic) where R-packages for experimental design are in *ExperimentalDesign* task view and R-packages for survey designs are in *OfficialStatistics* task view. A full list of available topics is provided in Table S1 in the Supplementary Materials. A subset of experimental designs is segregated into the *ClinicalTrials* task view, where the focus is on clinical trials with primary interest in sample size calculations. This paper focuses on packages in *ExperimentalDesign* task view, henceforth referred to as "DoE packages".

From the *ExperimentalDesign* task view, there are 105 R packages for the experimental design and analysis of data from experiments. The sheer quantity and variation of experimental designs in the R-packages are arguably unmatched with any other programming languages, for example, in Python, only a handful of packages that generate design of experiment exist (namely `pyDOE`, `pyDOE2`, `dexpy`, `experimenter`, and `GPdoemd`) with a limited type of design. Thus, the study of DoE packages, based on quantitative and qualitative data, can provide an objective view of the state of current experimental designs in practice.

The utility of the software can also be described by its design to facilitate the clear expression and interpretation of the desired experimental design. Certain programming language designs can hinder or discourage the development of reliable programs (Wasserman 1975). The immense popularity of `tidyverse` (a collection of R-packages for various stages of data analysis that places enormous emphasis on the interface design by Wickham et al. 2019) is a testament to the impact that an interface design can have in practice. The practice of experimental design can be advanced by adopting similar interface design principles across the DoE packages.

The remainder of this paper is organized as follows. Section 2 briefly describes the data source used for the analysis; Section 3 presents some insights into the state of the current DoE packages by the exploratory data analysis of package download data, text descriptions and comparisons with other CRAN task views; Section 4 discusses the interface designs of widely used DoE packages, and we conclude with a discussion in Section 5 of future prospects in the software development of experimental designs.

2 Data

To study the DoE packages, we analyse data using three sources of data as described below.

2.1 RStudio CRAN download logs

The Comprehensive R Archive Network (CRAN) is a network of servers located across the world that stores mirrored versions of the R and R packages. The most popular network is the RStudio mirror (the default server for those that use the RStudio IDE). The RStudio mirror is also the only server that provides a comprehensive daily download logs of R and R packages since October 2012. The summary data can be easily accessed using the `cranlogs` package (Csárdi 2019). This paper uses the data from the beginning of 2013 to the end of 2021 (a total of nine years) for the packages in the CRAN task views.

2.2 Package descriptions

All CRAN packages have a title, description, package connections (suggests, depends, and imports of other packages), and other meta-information in the DESCRIPTION file. We use text data from the title and description (accessed in 2022-12-12).

2.3 CRAN task views

CRAN task views are volunteer-maintained lists of R-packages on CRAN relevant to the corresponding topic. There were 39 CRAN task views in total. Table S1 in the Supplementary Materials list the available topics from the `ctv` package (Zeileis 2005). The list of packages in each CRAN task view (as of 2022-12-12) is used to contrast the characteristics of the DoE packages.

3 Exploratory data analysis

In this section, we derive some conjectures based on an analysis of the data described in Section 2. All results presented are from exploratory data analysis of observational data, consequently, all interpretations are somewhat speculative and may not be indicative of the true state of the field of experimental design. In particular, any analysis over time is confounded by the fact that the nature of users and package management has changed over the years. It should be noted that some DoE packages may have been archived or removed from the task view over the years; therefore, any cross-sectional analysis presented may not reflect the set of all DoE packages at that particular time period (although we assume such incidences are low).

A subset of DoE packages is not primarily about the design of experiments but about the analysis of experimental data. A complete delineation of these packages is difficult, as there is almost always at least one function that can aid decisions or constructions of experimental designs (and any categorization is prone to our subjective bias); therefore we opted not to remove any DoE packages in the analysis.

3.1 Small, but diverse, set of packages are sufficient for most experimental designs in practice

There are at least 50 DoE packages since 2013 but most of the downloads are concentrated in only a handful of packages. For example, Figure 1 shows a Lorenz curve (Lorenz 1905) for the total package downloads in 2021 for 102 DoE packages (first released prior to 2021). We can see from Figure 1 that the bottom 90% of DoE packages (in terms of total download count in 2021) only share approximately 32% of total downloads across all DoE packages; in other words, 68% of the total downloads are due to 10 packages (10% of the DoE packages).
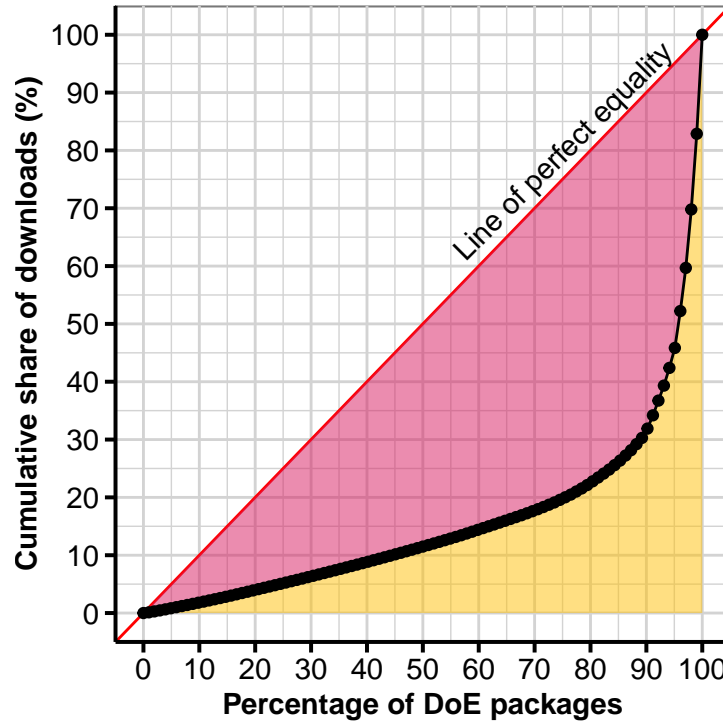

Figure 1: Lorenz curve of the total download count for DoE packages in 2021. The red line corresponds to the line of perfect equality. The yellow region shows the area under the Lorenz curve and the red region shows the area of the gap in equality.

If we consider package downloads as a measure of "wealth", then we can consider using the Gini index (Gini 1921) as a measure of download inequality across packages. The ratio of the red region to the total colored regions in Figure 1 corresponds to the Gini index for 2021. A Gini index of 0% indicates equality

in downloads across packages whereas a value of 100% indicates maximal inequality (all downloads are due to one package). In Figure 2, we see that the distributions of the package downloads each year have a heavy right tail with the Gini index ranging from 32.7% to 69.1% across the years 2013 to 2021, indicating that there is a high level of inequality in package downloads, particularly with more pronounced inequality in the last six years.
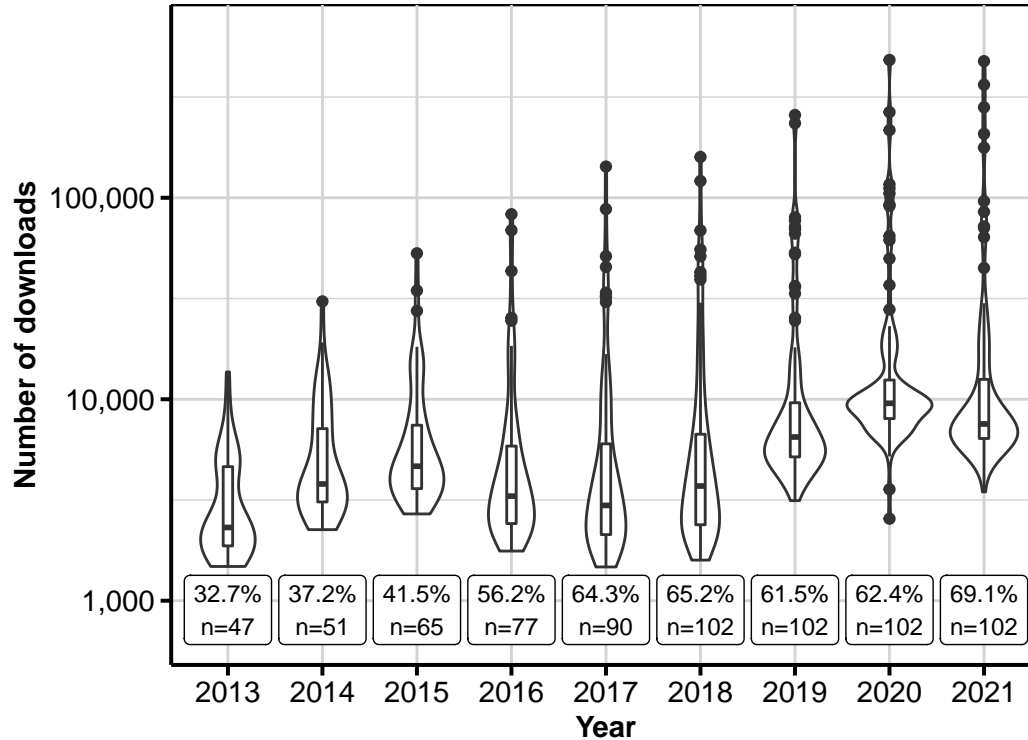

Figure 2: Distribution of number of downloads for DoE packages by year. Packages were removed in any year if they were released in that year or later so that each download count was for the full year. The label at the bottom of the plot shows the Gini index for downloads and the number of packages with a full download count in the corresponding year. In the last six years, the Gini index has consistently exceeded 60%, indicating that most downloads are due to a relatively small number of packages.

An increase in the number of packages that are not highly downloaded may mean that **there are more packages to construct niche experimental designs**. Some examples of these packages include `qtlDesign`, `PwrGSD` and `Crossover` made for QTL experiments, group sequential designs and crossover trials, respectively. These packages would naturally have fewer potential users. Counterfactual to this, the increase could be due to other external factors, such as an increase in the number of skilled developers (and thus more package contributions), a change in CRAN policy or management to add packages (either to CRAN and/or task view), and/or the fact that new packages are still yet to amass users. While there is an argument that low download counts are due to the low utility and/or quality of packages, packages in CRAN task views are selected by expert maintainers. We can reasonably assume that any package listed in the CRAN task view has an acceptable utility and quality.

If the downloads are reflective of the experimental designs used in practice, **a small set of packages appears to be sufficient for most users to construct the full set of designs of experiments they need in practice**. Packages of course evolve, and the top downloaded packages have had regular updates that may have broadened their scope from their previous releases.

While in absolute terms the Gini index is high for the *ExperimentalDesign* task view (32.7% to 69.1%), the inequality is not as severe as in other CRAN task views as shown in Figure 3. We can see in Figure 3 that the Gini index generally increases over time for DoE packages (as is generally the case for other CRAN task views as shown in Figure S1 in the Supplementary Materials), but most other CRAN task views have a Gini index of over 75%. This suggests that other CRAN task views may have dominant standards, and

in comparison to other topics, there are more **diverse approaches to designing experiments**, and thus, no single DoE package is dominant. However, this observation doe not consider other approaches to generate experimental designs, such as the proprietary software, CycDesignN (Whittaker, Williams, and John 2022), which may be widely used.
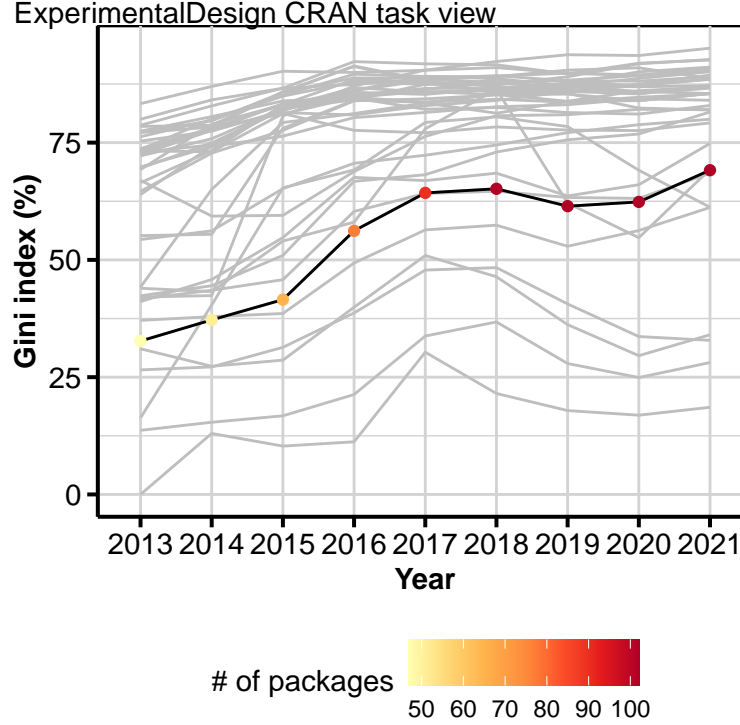

Figure 3: The points show the Gini index of the download counts by year for the ExperimentalDesign task view with the color showing the number of packages. The gray lines show the line plots of the Gini index across years for all other CRAN task views. See Figure S1 in the Supplementary Material for the line graph of the Gini index across years for each CRAN task view.

3.2 The field of experimental design is slow-changing

We can see in Figure 4 that most of the top 10 ranking packages have been in the top 10 for the last nine years with `lhs` steadily climbing up the ranks in the last few years. It should be noted that the download of one package can prompt the download of another package; the most notable package connection is `AlgDesign` and `agricolae`, where the former is an import for the latter. The full network of package connections within the DoE packages is shown in Figure 8.

Figure 5 shows a moderate negative correlation between the first release date and the (log of) total download counts of DoE packages for any given year from 2013 to 2021. This suggests that packages released earlier are more likely to be used today (possibly for legacy reasons or the general inertia to adopt new packages). We can also see in Figure 5 that most downloaded packages were released from 2004 to 2010.

The consistency in the top 10 ranking packages (Figure 4) and the fact that most downloaded DoE packages were first released more than 10 years ago (Figure 5) indicate that either the existing packages fulfilled the needs of mass in practice or no new packages were compelling for many to switch their practice. However, we also see that the top downloaded packages generally have more updates (see Figure 5); therefore, it is possible that the packages have improved or broadened the scope of their usage.
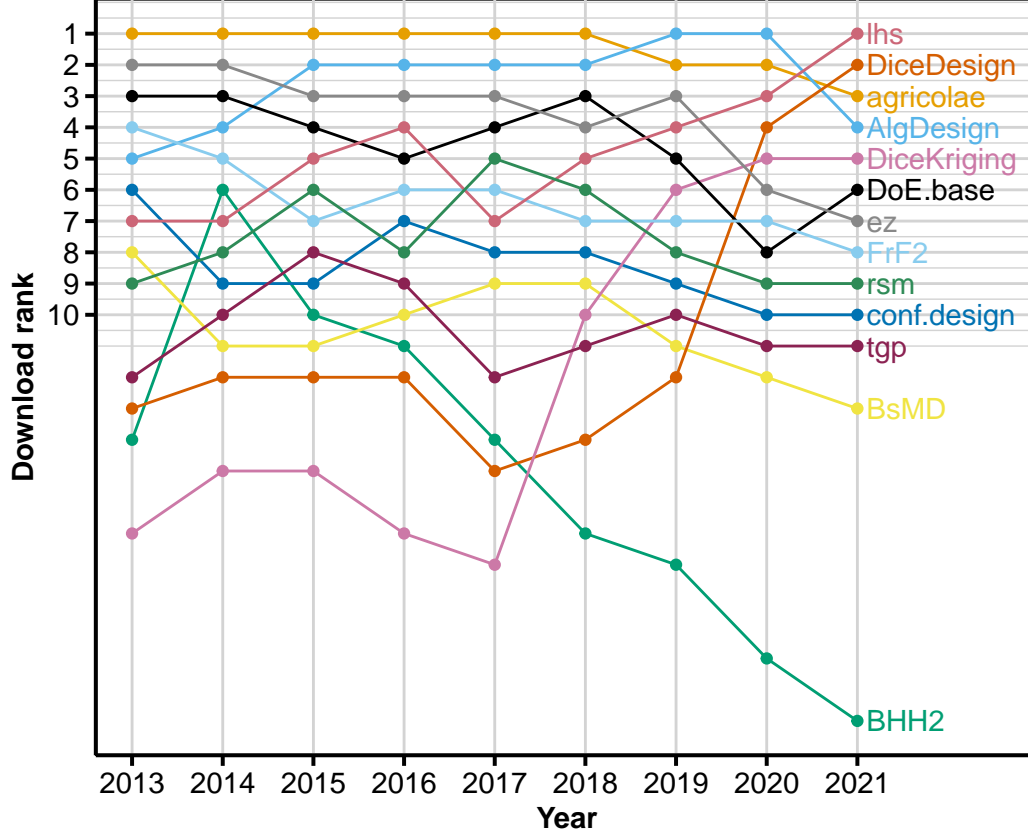
Figure 4: The plot shows the rank of the top 10 packages downloaded by year. Packages that did not appear in the top 10 for at least two periods were omitted from the plot. Most packages are consistently in the top 10 for the period shown.

3.3 Optimal designs are of interest

Figure 6 shows some of the common purposes of DoE packages, based on bigrams in the package title and description. We only show the bigrams as unigrams were not insightful, and there were not many trigrams common across packages. To count the bigrams, we processed the text data as follows:

1. We standardized the words to the lower case and removed pluralization.

2. Multiple mentions of the same bigram within a package were counted as one (for example, `AlgDesign` mentions "experimental design" four times in the title and the description, but this is counted as one).

3. Bigrams consisting of stop words were removed. The stop words are sourced from the lexicons in `tidytext::stop_words` in addition to other words we deemed irrelevant, e.g. "provide", "e.g.", "calculate" and so on – full list is shown in the code provided in the link under Section 6.

Unsurprisingly, the bigram "experimental design" was the most common. More interestingly, "optimal design" and "sequential design" appeared across different packages (indicated by the size of the word in Figure 6), and the bigrams "latin hypercube" and "computer experiment" are used across a few packages that are downloaded frequently (indicated by the color of the word in Figure 6). Sequential design, Latin hypercube sampling and computer experiments (which generally include space-filling designs such as Latin hypercube sampling) generally operate by optimizing a user-selected criterion and can be classified as optimal designs.
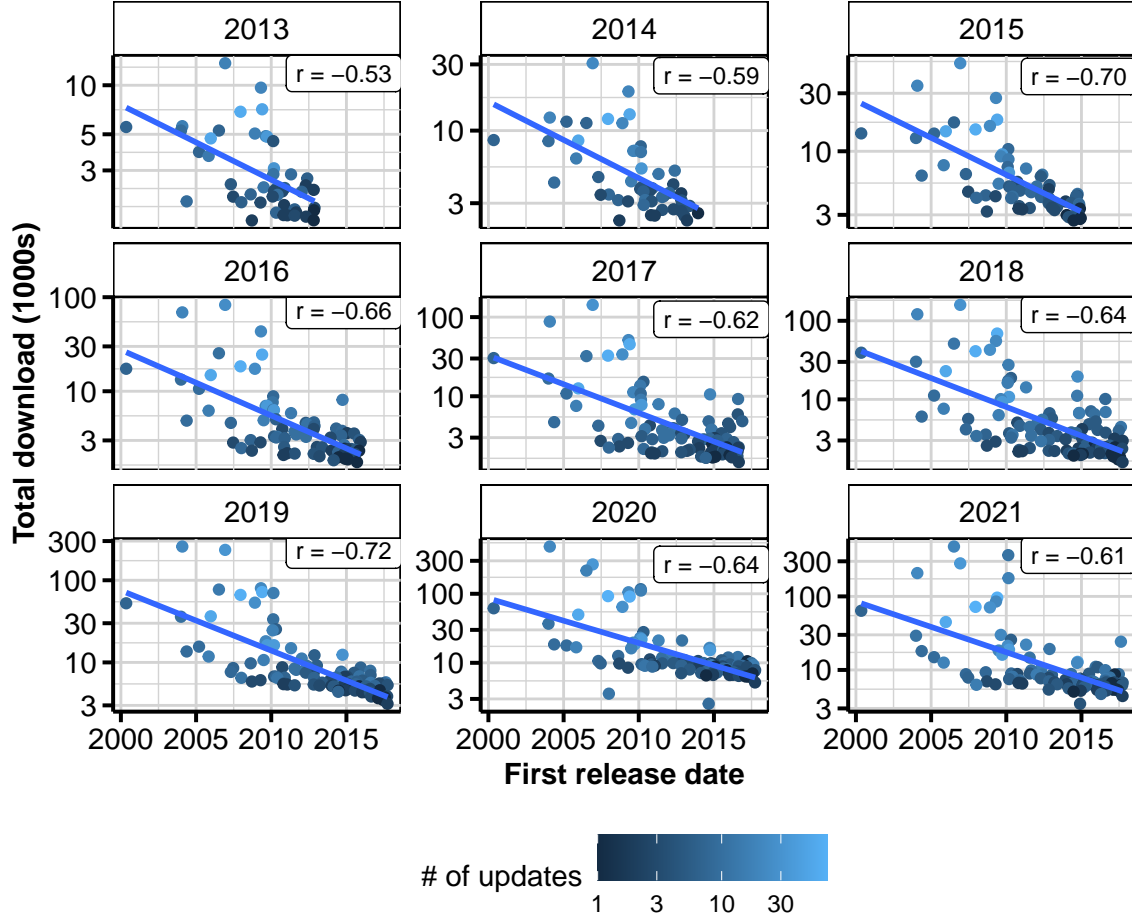
Figure 5: The above figure shows the total download (in log scale) of a package in the corresponding year against the first release date of the package. The blue line corresponds to the least-squares fit of the simple linear regression model. The label in the upper right-hand corner shows the sample correlation coefficient between the first release date and log (with base 10) of the total download count. The high leverage point on the far left belongs to 'conf.design', authored by one of the earlier contributors to R.

Although there exists a separate *ClinicalTrials* task view, the DoE packages clearly include some packages that are of interest to clinical trials as shown by the size of the bigram "clinical trial" (and related bigrams like "dose finding" and "phase ii") in Figure 6.

3.4 Development of experimental designs occur in silos

Figure 7 shows that the *ExperimentalDesign* task view has the lowest average number of contributors among all the 39 CRAN task views. In addition, we can also see in Figure 7 that the *ExperimentalDesign* task view has one of the least intra-connectivity (the percentage of packages that make use of other packages within the same task view). The full connection between DoE packages is shown in Figure 8. These observations suggest that ***experimental design is one of the least collaborative fields*** and package development generally occurs in silos.

4 Interface design

In software design, there are two interface designs to consider: user interface (UI) and application programming interface (API). The UI is concerned with the interaction of the software by the user, while the API is concerned with how different programs interact and is predominately of interest to
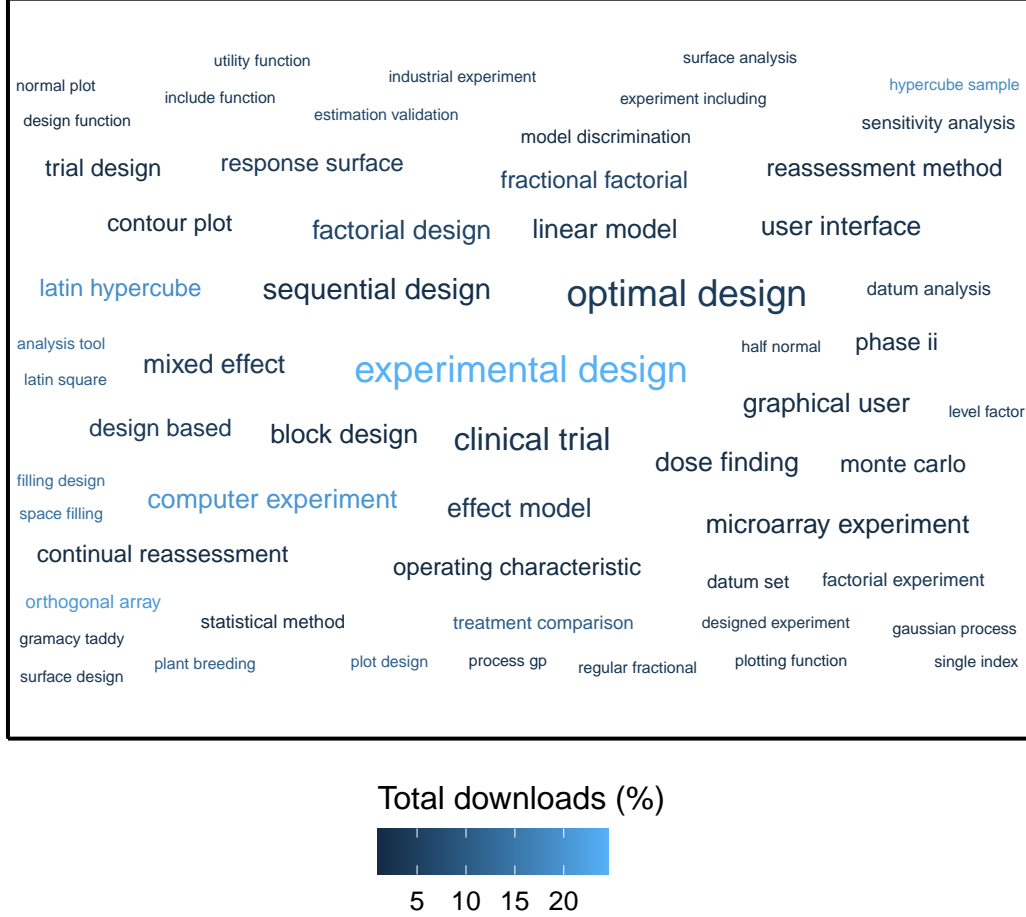
Figure 6: The above figure shows the word cloud of bigrams from the title and descriptions of the DoE packages. The size shows how often the bigram appears across the DoE packages and the color is relative to the total download count in 2021 for the packages that contain the bigram.

the developer. The UI design is an abstraction that specifies the desired experimental design, and its choices enable how a user expresses the specification of an experimental design. The API design aids other developers in leveraging existing systems.

In this section, we discuss the interface designs of functions that output an experimental design based on three broad areas: factorial, recipe and augmenting designs. The discussion is exclusive to the top downloaded packages (shown in Figure 4), with the exception of ez and DiceKriging, as the former is predominately visualization of experimental data and the latter is about the analysis of computer experiments in addition to belonging to the same suite of packages as DiceDesign (Dupuy, Helbert, and Franco 2015).

4.1 The case of factorial designs

Factorial experiments offer a challenge in allocating the treatment factors to experimental units where the full set of factorial treatments cannot be administered (and replicated) and/or the experimental units have a grouping structure. The effort to address this challenge is reflected in the number of packages that focus on the construction of factorial designs as described next.

The DoE.base package (Grömping 2018) can construct full factorial and (regular and irregular) orthogonal array designs via fac.design() and oa.design(), respectively. The FrF2 package (Grömping 2014) constructs regular fractional 2-level factorial designs using FrF2() and FrF2Large(), with the latter for large designs. The conf.design package (Venables 2013) constructs symmetric confounded factorial
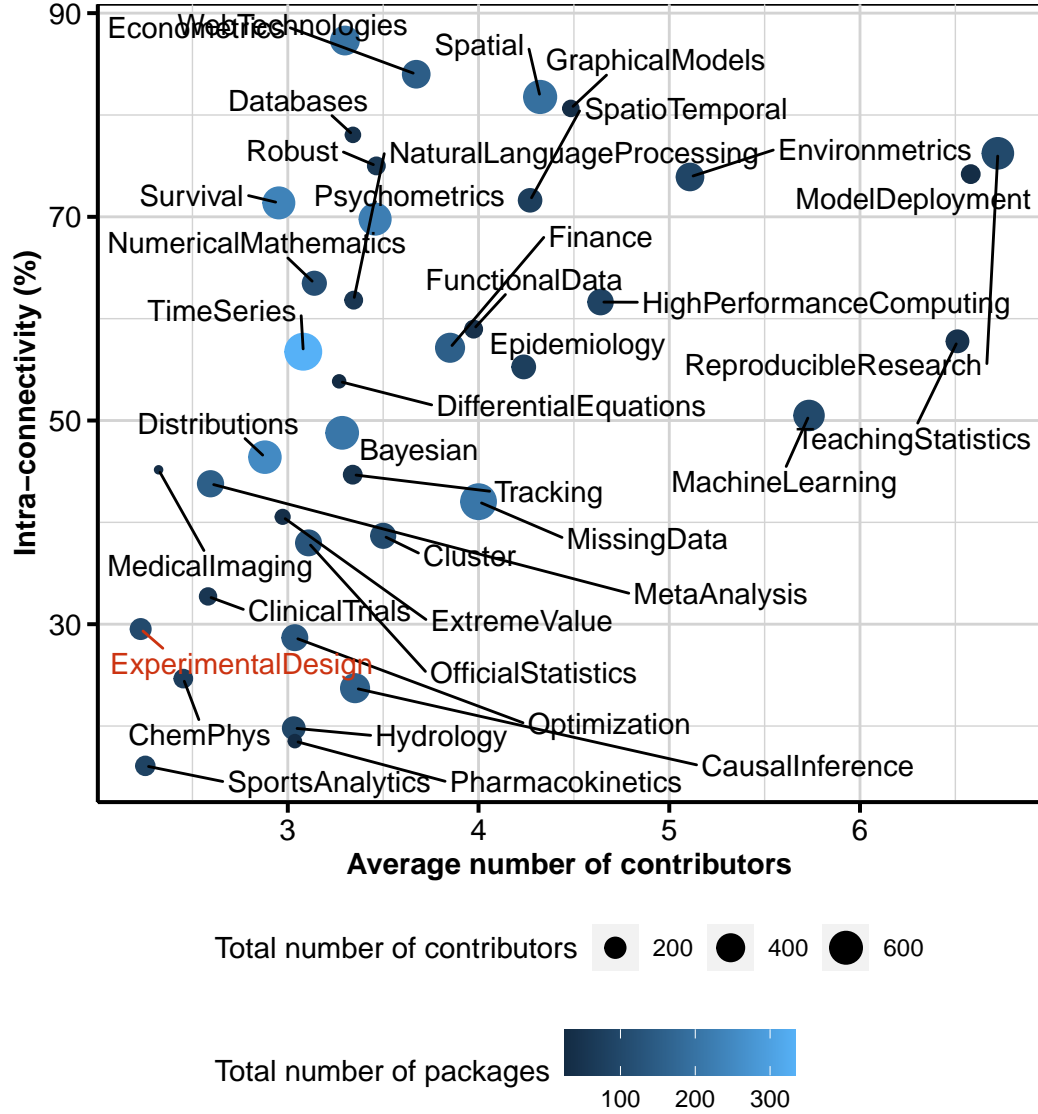
Figure 7: The above figure is a scatterplot of intra-connectivity (the percentage of packages that depends, suggests or imports at least one other package within the same task view) and the average number of contributors for each CRAN task view. Low intra-connectivity suggests that development within the topic mostly occurs in silos, while high intra-connectivity suggests that there are more interactions within the topic. The color shows the number of packages, the size of the point corresponds to the total number of contributors, and the text labels show the CRAN task view names. The label of ExperimentalDesign task view is colored in red. The task views in the bottom-left corner are topics that are more indicative of contributors working in silos. The actual numerical values are listed in Table S1 in the Supplementary Material.
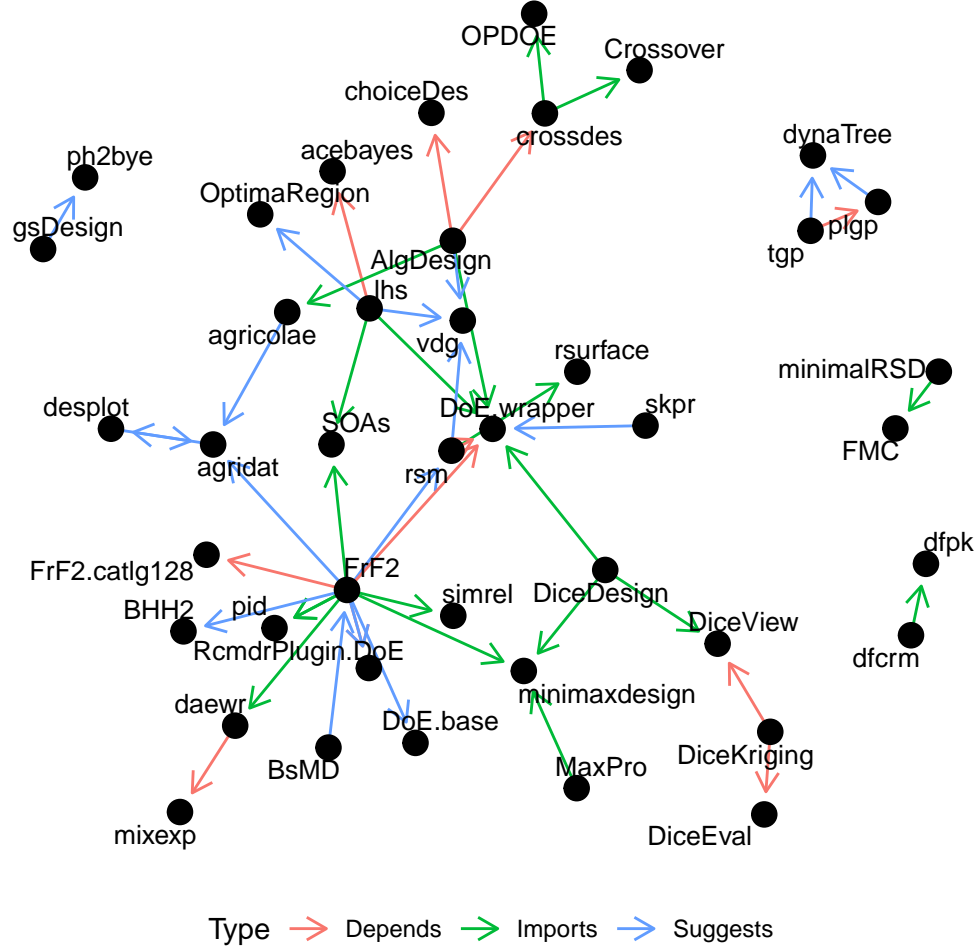
Figure 8: Package connections (depends, suggests and imports) within DoE packages. The direction of the arrow shows the connection of packags where the package on the tail of the arrow is a dependency, suggestion or import for the package on the head of the arrow. DoE packages that do not depend, suggest or import another DoE package are not shown.

designs via `conf.design()` and the BHH2 package (Barrios 2016) generates a full or fractional 2-level factorial design matrix via `ffDesMatrix()`.

While the argument names and underlying algorithms of these functions to generate factorial designs differ, it generally requires the users to input the number of:

- treatment factors,
- experimental runs (or replications), and
- levels for each factor if the design was allowed to vary in the number of levels.

The output of the design is either a special class of list (e.g. the `design` class for `DoE.base` and `FrF2` or a `data.frame` for `conf.design`) or a matrix for `BHH2` such that an element or column corresponds to a treatment factor with each value corresponding to one experimental run. The treatment factors are generally assigned pseudo names (e.g. letters of the alphabet) or an argument exists for users to input treatment names as a character vector.

Some forms of factorial design are known by other names, for example, *response surface designs* are factorial designs where the treatment factors are discrete levels of continuous variables. Two types of response surface designs can be constructed using the `rsm` package (Lenth 2009): Box-Behnken design (Box and Behnken 1960) and central-composite designs (Box and Wilson 1951) via functions

bbd() and ccd(), respectively, where the minimum required input is the number of factors. Another form of factorial design is the *saturated designs* where higher-order interaction effects of treatment factors are typically confounded with the main effects. Plankett-Burnman designs (Plackett and Burman 1946) are a type of saturated design that can be generated by the function pb() in the FrF2 package, where the user provides the number of experimental runs and the number of treatment factors.

4.2 The case of recipe designs

The agricolae package (de Mendiburu 2021) is the prime example of constructing designs based on a set of so-called "recipe functions", where each function corresponds to a single class of experimental design. For example, design.crd(), design.rcbd() and design.split() construct completely randomized, randomized complete block and split-plot designs, respectively. Users typically supply treatment labels (or the number of treatments in the case of design.split()) and the number of replications as arguments for these functions. The output is a list with one element corresponding to a data.frame that contains the design in a table such that the row corresponds to the experimental run and the columns correspond to the experimental variables (we refer to this format simply as "table format" henceforth).

The use of recipe functions is not limited to classical experimental designs; the AlgDesign (Wheeler 2022) package offers three primary functions for generating optimal designs: optBlock(), optFederov() and optMonteCarlo(). In general, these functions require data and formulas in terms of the supplied data variables with the choice of the criterion (e.g. the D-criterion), with the output as a list with one element corresponding to the design in a table format. The difference between these functions lies in the underlying search strategy for optimal designs, and the name of the function is a surrogate for the search algorithm.

Computer experiments, which generally involve space-filling designs, are implemented in packages such as lhs (Carnell 2022) and DiceDesign (Dupuy, Helbert, and Franco 2015). For the lhs package, functions such as randomLHS(), optimumLHS(), and maximinLHS(), require users to specify the sample size (n) and the number of variables (p), then it generates a Latin hypercube sample (McKay, Beckman, and Conover 1979), based on different optimization schemes (in this case, random, S optimal and maxmin criteria, respectively; see package documentation for more details). Similarly for DiceDesign, there is a comprehensive list of space-filling designs such as dmaxDesign(), lhsDesign(), and wspDesign() with input of n and p as before (among additional parameters for some) that implement algorithms that either maximize the entropy (Shewry and Wynn 1987), produce a random Latin hypercube sample, or use the WSP algorithm (Santiago, Claeys-Bruno, and Sergent 2012), respectively. These designs output an $n \times p$ matrix with values between 0 and 1. Again, these functions employ a recipe style, in which each function has a name that corresponds to a certain search strategy to generate the experimental design.

4.3 The case of augmenting designs

Some functions in the DoE packages require the input of existing experimental designs to produce new designs. For example, the DoE.base package contains some experimental functions, cross.design() and param.design(), to combine designs with the former taking a Cartesian product of the input designs while the latter uses a Taguchi style (Taguchi 1986) to aggregate the designs with the inner and outer arrays. The lhs package contains a function that augmentLHS() to add additional samples to the existing Latin hypercube sample.

Another class of augmenting design is *sequential design* (also called *adaptive sampling*), which is best represented by tgp (Gramacy and Taddy 2010). This requires prior information that is used to inform the next experimental design using tgp.design() and dopt.gp(). The user is required to supply candidate samples to subsample from and a model or a prior experimental design. Follow-up experiments, which can also be classified as sequential designs, is implemented by BsMD (Barrios 2020) using a model-discriminant approach using MD().

5 Discussion

Through the exploratory data analysis of the three data sources (package download logs, package metadata and CRAN task views) outlined in Section 2, we observed in Section 3 that the the total download of DoE packages is concentrated only on a handful of R-packages although these represent a diverse set in comparison with other CRAN task views. Furthermore, the data suggest that experimental design is the least collaborative field.

There are a number of limitations and shortcomings to our exploratory data analysis. First, CRAN task views are volunteer maintained so some experimental design packages may not be included in the DoE packages. Second, we used only the RStudio CRAN mirror download, which may have biased our observations. Third, our analysis was limited to R-packages alone and many practitioners may use other methods to construct experimental designs. Finally, all our statements should be treated as speculative rather than conclusive; the data are all observational so no conclusive, generalizable statement is possible. Regardless, the data-driven nature of our analysis provides objective insight into the field of experimental design.

The interface design (discussed in Section 4) reveals that the most widely used DoE packages generally have functions that 1) focus on certain aspects of experimental design (e.g. factorial structure or augmenting design), 2) are a recipe format (i.e. the name of the function is a surrogate to a single class of the design or optimal search algorithm) and 3) context is often a second thought – many inputs a single integer corresponding to the number of factors, levels, or experimental runs (or sample size). The function will often assign pseudo-factor names or there is an optional argument to input a character vector that corresponds to the factor names. These interface designs require users to have processed the experiment in statistical terms (often stripping the experimental context away) and simultaneously, users must choose the generation mechanism by selecting an appropriate function. Arguably, the current dominant interface designs are not aligned with the way practitioners cognitively design their experiments. Often, the critical part of designing an experiment is to understand the experimental structure, and the experimental context can govern or guide the choice of algorithm to allocate treatments. In addition, the nature of recipe functions can obscure the understanding and relation of designs (e.g., how do you go from an unstructured factorial design to split plot design?). Each new method for generating an experimental design appears to correspond to a completely new function, and intermediate results are often not easily accessible. These factors may contribute to why developers often work in silos in the field of experimental design.

A consistent cognitive interface design that leverages existing developments and can be easily extended to new methods will conceivably be of great help to practitioners. Some efforts to this end are seen in `DoE.wrapper` (Grömping 2020), which contains wrapper recipe functions for other DoE packages such as `lhs`, `AlgDesign` and `FrF2` (see Figure 8), and is also the subject of the developmental package `edibble` (Tanaka 2021). Undoubtedly, no single developer or package can cater to all experimental designs; therefore, any unifying interface should consider how other developers can contribute or add their methods. Future research could benefit from further exploratory data analysis, expanding the study to beyond R-packages, and discussing other aspects of interface designs.

6 Acknowledgement

Barrios, Ernesto. 2016. *BHH2: Useful Functions for Box, Hunter and Hunter II*. `https://CRAN.R-project.org/package=BHH2`.

———. 2020. *BsMD: Bayes Screening and Model Discrimination*. `https://CRAN.R-project.org/package=BsMD`.

Box, G E P, and D W Behnken. 1960. "Some New Three Level Designs for the Study of Quantitative Variables." *Technometrics: A Journal of Statistics for the Physical, Chemical, and Engineering Sciences* 2 (4): 455–75. `https://doi.org/10.2307/1266454`.

Box, G E P, and K B Wilson. 1951. "On the Experimental Attainment of Optimum Conditions." *Journal of the Royal Statistical Society. Series B, Statistical Methodology* 13 (1): 1–45.

Carnell, Rob. 2022. *Lhs: Latin Hypercube Samples*. `https://CRAN.R-project.org/package=lhs`.

Csárdi, Gábor. 2019. *Cranlogs: Download Logs from the 'RStudio' 'CRAN' Mirror*. `https://CRAN.R-project.org/package=cranlogs`.

de Mendiburu, Felipe. 2021. *Agricolae: Statistical Procedures for Agricultural Research*. `https://CRAN.R-project.org/package=agricolae`.

Dupuy, Delphine, Céline Helbert, and Jessica Franco. 2015. "DiceDesign and DiceEval: Two R Packages for Design and Analysis of Computer Experiments." *Journal of Statistical Software* 65 (11): 1–38. `https://www.jstatsoft.org/v65/i11/`.

Fisher, Ronald. 1935. *The Design of Experiments*. Oliver; Boyd.

Gini, Corrado. 1921. "Measurement of Inequality of Incomes." *The Economic Journal* 31 (121): 124–26. `https://doi.org/10.2307/2223319`.

Gramacy, Robert B., and Matthew Taddy. 2010. "Categorical Inputs, Sensitivity Analysis, Optimization and Importance Tempering with tgp Version 2, an R Package for Treed Gaussian Process Models." *Journal of Statistical Software* 33 (6): 1–48. `https://doi.org/10.18637/jss.v033.i06`.

Grömping, Ulrike. 2014. "R Package FrF2 for Creating and Analyzing Fractional Factorial 2-Level Designs." *Journal of Statistical Software* 56 (1): 1–56. `https://www.jstatsoft.org/v56/i01/`.

———. 2018. "R Package DoE.base for Factorial Experiments." *Journal of Statistical Software* 85 (5): 1–41. `https://doi.org/10.18637/jss.v085.i05`.

———. 2020. *DoE.wrapper: Wrapper Package for Design of Experiments Functionality*. `https://CRAN.R-project.org/package=DoE.wrapper`.

Landau, William Michael. 2021. "The Targets r Package: A Dynamic Make-Like Function-Oriented Pipeline Toolkit for Reproducibility and High-Performance Computing." *Journal of Open Source Software* 6 (57): 2959. `https://doi.org/10.21105/joss.02959`.

Le Pennec, Erwan, and Kamil Slowikowski. 2019. *Ggwordcloud: A Word Cloud Geom for 'Ggplot2'*. `https://CRAN.R-project.org/package=ggwordcloud`.

Lenth, Russell V. 2009. "Response-Surface Methods in R, Using rsm." *Journal of Statistical Software* 32 (7): 1–17. `https://doi.org/10.18637/jss.v032.i07`.

Lorenz, M O. 1905. "Methods of Measuring the Concentration of Wealth." *Publications of the American Statistical Association* 9 (70): 209–19. `https://doi.org/10.2307/2276207`.

McKay, M D, R J Beckman, and W J Conover. 1979. "Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code." *Technometrics: A Journal of Statistics for the Physical, Chemical, and Engineering Sciences* 21 (2): 239–45. `https://doi.org/10.1080/00401706.1979.10489755`.

Pedersen, Thomas Lin. 2021. *Ggraph: An Implementation of Grammar of Graphics for Graphs and Networks*. `https://CRAN.R-project.org/package=ggraph`.

Plackett, R L, and J P Burman. 1946. "The Design of Optimum Multifactorial Experiments." *Biometrika* 33 (4): 302–25.

Pukelsheim, Friedrich. 2006. *Optimal Design of Experiments*.

R Core Team. 2021. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. `https://www.R-project.org/`.

Rossum, G. van. 1995. "Python Tutorial." CS-R9526. Amsterdam: Centrum voor Wiskunde en Informatica (CWI).

Rudis, Bob, and Blake Embrey. 2020. *Pluralize: Pluralize and 'Singularize' Any (English) Word*. `https://CRAN.R-project.org/package=pluralize`.

Santiago, J, M Claeys-Bruno, and M Sergent. 2012. "Construction of Space-Filling Designs Using WSP Algorithm for High Dimensional Spaces." *Chemometrics and Intelligent Laboratory Systems* 113 (April): 26–31. `https://doi.org/10.1016/j.chemolab.2011.06.003`.

SAS Institute. 1985. *SAS User's Guide: Statistics*. Vol. 2. Sas Inst.

Shewry, M C, and H P Wynn. 1987. "Maximum Entropy Sampling." *Journal of Applied Statistics* 14 (2): 165–70. https://doi.org/10.1080/02664768700000020.

Silge, Julia, and David Robinson. 2016. "Tidytext: Text Mining and Analysis Using Tidy Data Principles in r." *JOSS* 1 (3). https://doi.org/10.21105/joss.00037.

Taguchi, Genichi. 1986. *Introduction to Quality Engineering: Designing Quality into Products and Processes*. Quality Resources.

Tanaka, Emi. 2021. "edibble R-package." *GitHub Repository*. https://github.com/emitanaka/edibble; GitHub.

Ushey, Kevin. 2022. *Renv: Project Environments*. https://CRAN.R-project.org/package=renv.

Venables, Bill. 2013. *Conf.design: Construction of Factorial Designs*. https://CRAN.R-project.org/package=conf.design.

Wasserman, Anthony I. 1975. "Issues in Programming Language Design - an Overview." In.

Wheeler, Bob. 2022. *AlgDesign: Algorithmic Experimental Design*. https://CRAN.R-project.org/package=AlgDesign.

Whittaker, D., E. R. Williams, and J. A John. 2022. *CycDesign: A Package for the Computer Generation of Experimental Designs,* (version 7.0). https://vsni.co.uk/software/cycdesign.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'agostino McGowan, Romain Francois, Garrett Grolemund, et al. 2019. "Welcome to the Tidyverse." *Journal of Open Source Software* 4 (43): 1686.

Xie, Yihui. 2015. *Dynamic Documents with R and Knitr*. 2nd ed. Boca Raton, Florida: Chapman; Hall/CRC. https://yihui.org/knitr/.

Xie, Yihui, J. J. Allaire, and Garrett Grolemund. 2018. *R Markdown: The Definitive Guide*. Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown.

Zeileis, Achim. 2005. "CRAN Task Views." *R News* 5 (1): 39–40. https://CRAN.R-project.org/doc/Rnews/.

———. 2014. *Ineq: Measuring Inequality, Concentration, and Poverty*. https://CRAN.R-project.org/package=ineq.

Zeileis, Achim, Jason C. Fisher, Kurt Hornik, Ross Ihaka, Claire D. McWhite, Paul Murrell, Reto Stauffer, and Claus O. Wilke. 2020. "colorspace: A Toolbox for Manipulating and Assessing Colors and Palettes." *Journal of Statistical Software* 96 (1): 1–49. https://doi.org/10.18637/jss.v096.i01.

Zhu, Hao. 2021. *kableExtra: Construct Complex Table with 'Kable' and Pipe Syntax*. https://CRAN.R-project.org/package=kableExtra.