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

A Preprint

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

June 15, 2022

Abstract

Re-running an experiment is generally costly and in some cases impossible due to limited resources, so 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 the R-packages for the design of experiments through an exploratory data analysis of package downloads, package metadata, and the comparison of characteristics with other topics. We observe that experimental designs in practice appear to be sufficiently manufactured by a small number of packages and the development of experimental designs often occur in silos. We discuss also the interface designs of widely utilised R packages in the field of experimental design and discuss its future prospects to advance 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 is rubbish then no analysis, however complex it may be, can make something out of it. A carefully crafted data collection scheme is therefore critical to optimise the information from data. The field of experimental designs is specifically devoted to planning the collection of experimental data, largely based on the founding principles by Fisher (1935) or an optimisation framework like those described in Pukelsheim (2006). These experimental designs are often constructed with the aid of a statistical software, such as R (R Core Team 2021), Python (Rossum 1995), SAS (SAS Institute 1985), and so on, so the usage of experimental design software can inform us about some aspects of experimental designs in practice.

Methods for data collection can be dichotomised by the type of data collected – namely, experimental or observational – or alternatively, categorised as experimental design (including quasi-experimental design) or survey design. This dichotomisation, to a great extent, is seen in the Comprehensive R Archive Network (CRAN) task views (a volunteer maintained list of R-packages by topic) where R-packages for

*Corresponding author

experimental design are in *ExperimentalDesign* task view and R-packages for survey designs are in *OfficialStatistics* task view. The full list of available topics can be see in Table S1 in the Supplementary Materials. A subset of experimental designs are segregated into *ClinicalTrials* task view, where the focus is on clinical trials with primary interest in sample size calculations. This paper focuses on the packages in *ExperimentalDesign* task view, henceforth referred to as "DoE packages".

In *ExperimentalDesign* task view, there are 114 R packages for 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, e.g. in Python, only a handful of packages that generate design of experiment exist (namely `pyDOE`, `pyDOE2`, `dexpy`, `experimenter` and `GPdoemd`) with limited type of designs. Thus, the study of DoE packages, based on quantitative and qualitative data, can give us an objective view to the state of the current experimental designs in practice.

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

The paper is organised as follows. Section 2 briefly describes the data source used for the analysis in Section 3; 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 store mirrored versions of 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 with the `cranlogs` package (Csárdi 2019). This paper uses the data from the beginning of 2013 to end of 2021 (a total of 9 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 the text data from the title and description (as accessed in 2022-05-23) in Section 3.3.

2.3 CRAN task views

CRAN task views are volunteer maintained list of R-packages on CRAN relevant to the corresponding topic. There are a total 37 CRAN task views. Table S1 in the Supplementary Materials list the full available topics from the `ctv` package (Zeileis 2005). The list of packages in each CRAN task views (as of 2022-05-25) are used to contrast the characteristics of DoE packages in Sections 3.1 and 3.4.

3 Explorative data analysis

In this section, we derive some conjecture based on the analysis of the data described in Section 2. All results presented are from the 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 are confounded by the fact that the nature of users and package management have also 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 so 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 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 categorisation is prone to our subjective bias) so 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 just a handful of packages. For example, Figure 1 shows a Lorenz curve (Lorenz 1905) for the total package downloads in 2021 of 113 DoE packages (first released prior to 2021); we can see from Figure 1 that bottom 90% of DoE packages (in terms of total download count in 2021) only share about 27% of total downloads across all DoE packages – in another words, 73% of the total downloads are due to 11 packages (10% of the DoE packages).

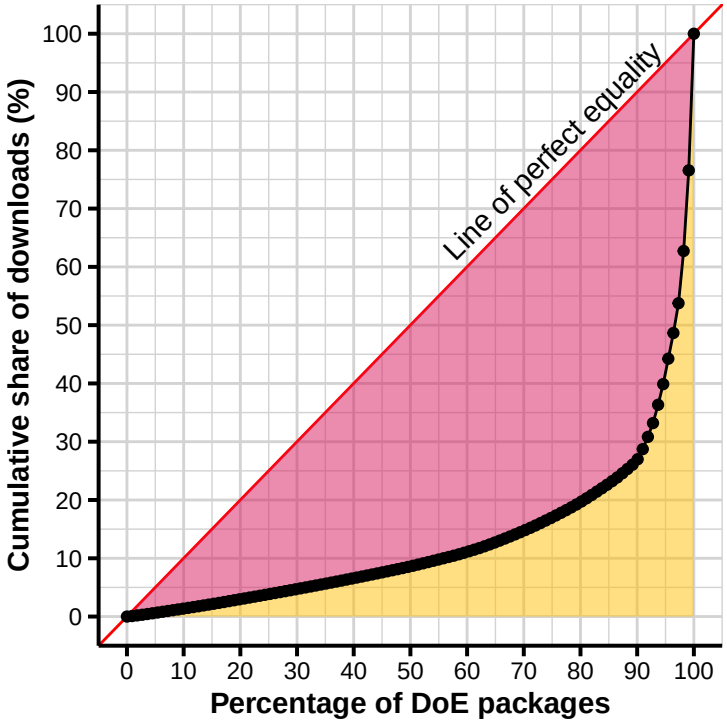

Figure 1: Lorenz curve of the total download count for DoE packages in 2021. The red line corresponds to the line of perfect equality. The yellow region shows the area under the Lorenz curve and the red region shows the area of gap in equality.

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 over 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 while 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 42.6% to 74.1% across the years 2013 to

2021, indicating that there is a high level of inequality of package downloads, particularly with more pronounced inequality in the last 6 years.

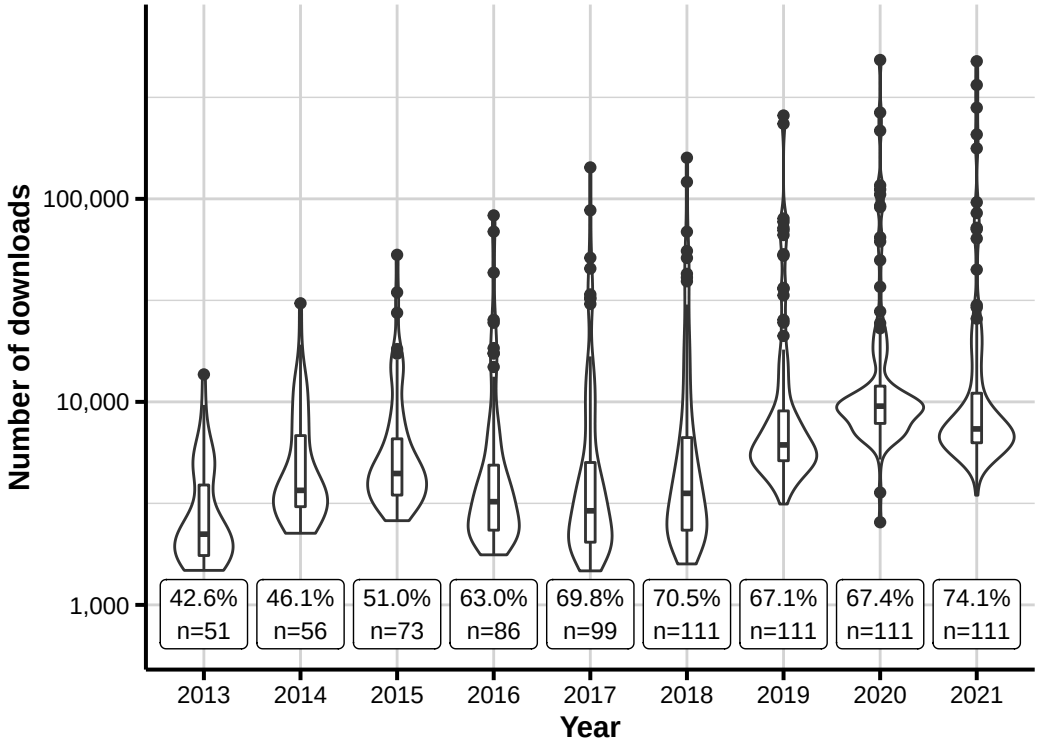

Figure 2: The distribution of the number of downloads for DoE packages by year. Packages were removed in any year if it was released in that year or later so that each download count is for the full year. The label on the bottom of the plot shows the Gini index for downloads and the number of packages with full year of download count in the corresponding year. In the last 6 years, the Gini index is consistent above 60% indicating that most downloads are due to a relatively small number of packages.

The 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 a smaller number of 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 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 the packages, packages in CRAN task views are selected by expert maintainers, thus we can reasonably assume that any package listed in CRAN task view are of a decent utility and quality.

If the downloads are reflective of the experimental designs used in practice, ***small set of packages appear to be sufficient for most 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 its scope from its previous releases.

While in absolute terms the Gini index is high for *ExperimentalDesign* task view (42.6% to 74.1%), the inequality is not as severe as other CRAN task views as shown in Figure 3. We can see in Figure 3 that the Gini index is generally increasing 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. This observation however doesn't take into account other

approaches to generate experimental designs, like say the proprietary software, CycDesignN (Whittaker, Williams, and John 2022), which may be widely used.

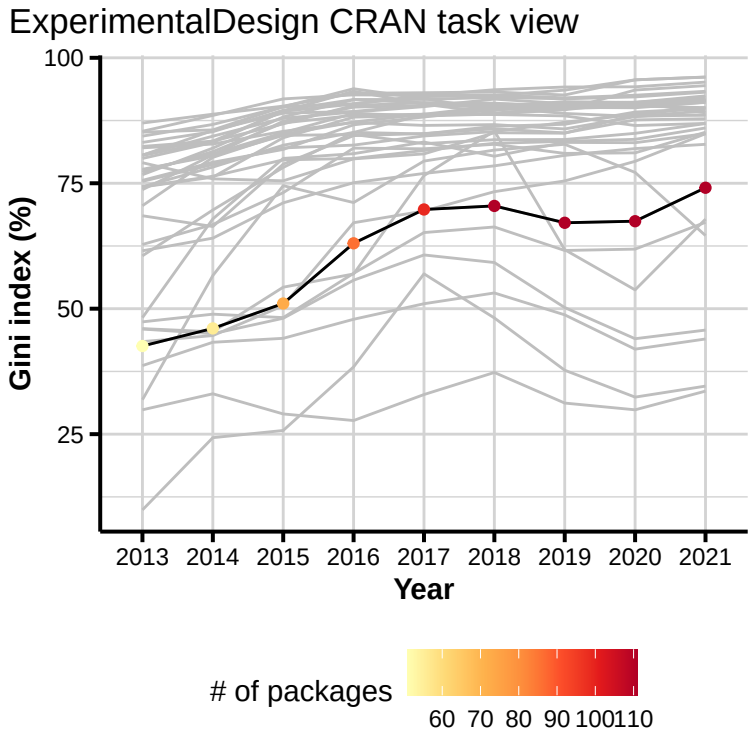

Figure 3: The points show the Gini index of the download counts by year for ExperimentalDesign task view with the color showing the number of packages. The grey lines show the line plots of the Gini index across years for all other CRAN task views. See Figure S1 in the Supplementary Material for the line graph for the Gini index across years by 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 9 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 a 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 packages connections within the DoE packages is shown later 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 in any given year from 2013 to 2021. This suggest that in general, a package 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 the most downloaded packages were generally released in 2004 to 2010.

The consistency in the top 10 ranking packages (Figure 4) and the fact that most downloaded DoE packages are first released more than 10 years ago (Figure 5) indicates that either the existing packages fulfilled the needs of mass in practice or no new packages were compelling for many to switch their practice. We do however also see that the the top downloaded packages generally have more updates (see Figure 5) so it is possible that the packages have improved or broadened the scope of its usage.

3.3 Optimal designs are of interest

Figure 6 shows some 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. In counting the bigrams, we processed the text data as follows:

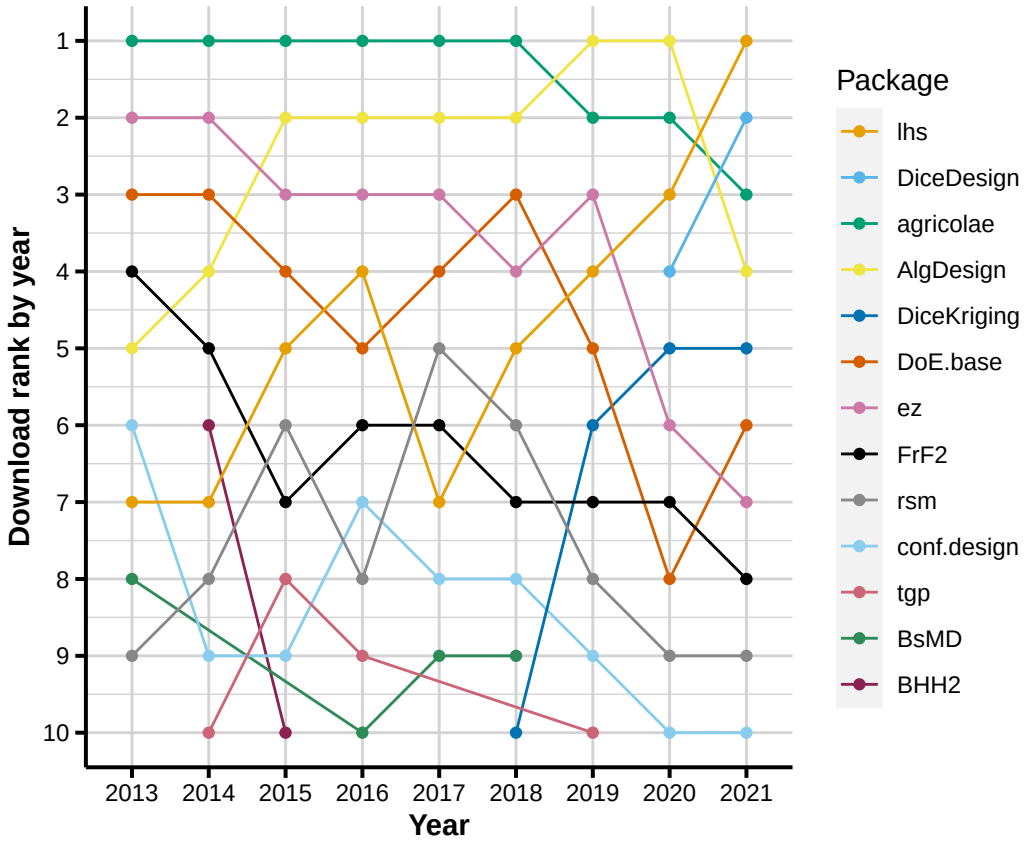

Figure 4: The plot shows the rank of top 10 downloaded packages by year. Packages that do not appear in top 10 for at least two periods are omitted from the plot. Most packages are consistently in the top 10 for the period shown.

1. We standardised the words to lower case and removed pluralization.
2. Multiple mentions of the same bigram within a package was counted as one (e.g. `AlgDesign` mentions "experimental design" four times in the title and the description but this is counted as one).
3. Bigrams that consist the stop words are 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.

Not surprisingly, 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 like Latin hypercube sampling) are designs that generally operate by optimising a user-selected criterion and can be classified as optimal designs.

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 37 CRAN task views. In addition, we can also see in Figure 7 that the *ExperimentalDesign*

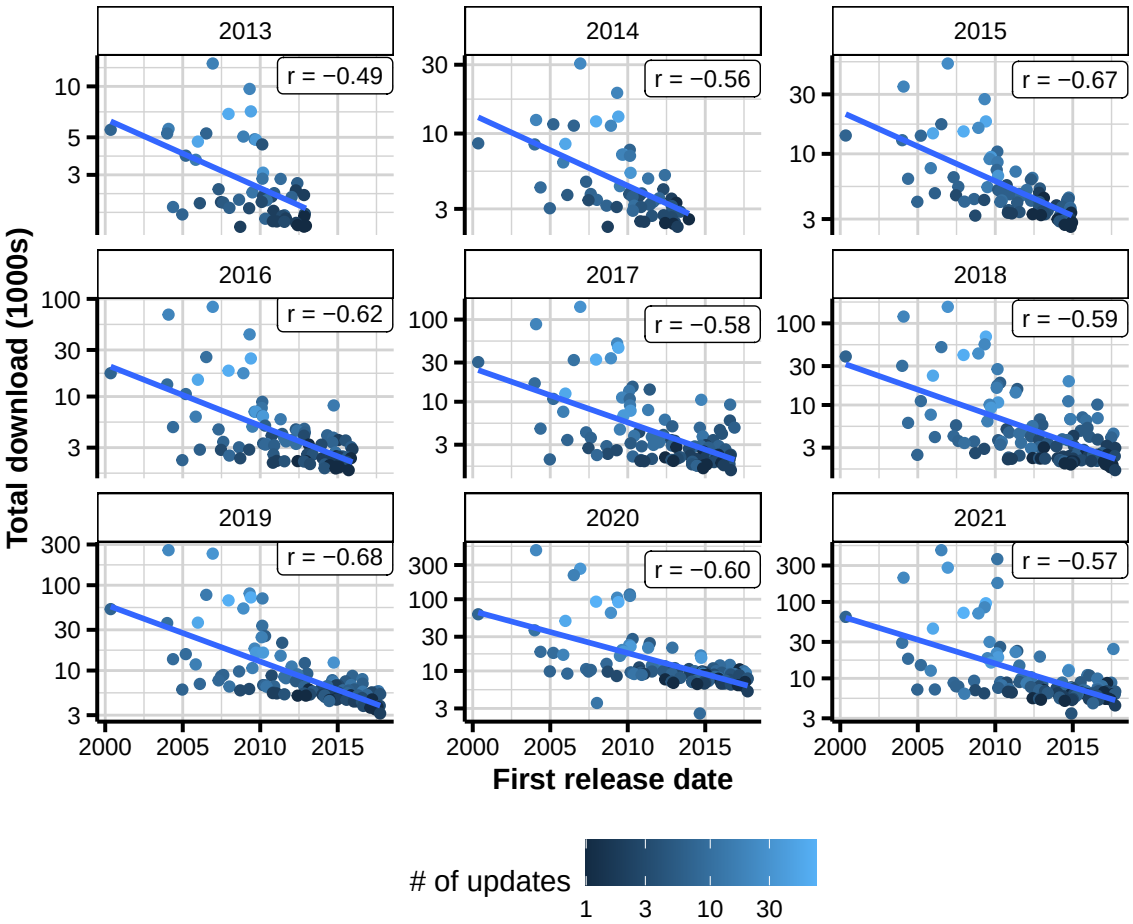

Figure 5: The above figure shows the total download (in log scale) of a package in the corresponding year against the first release date of the package. The blue line corresponds to the least squares fit of a simple linear regression model. The label in the right-hand upper corner shows the sample correlation coefficient between the first release date and the 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.

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 of DoE packages with other DoE packages is shown in Figure 8. These observations suggest that the ***field of experimental design is one of the least collaborative field*** and package development generally occur in silos.

4 Interface design

In software design, there are two interface designs to consider: user interface (UI) and application programming interface (API). 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 the interest to the developer. The UI design is an abstraction to specifying the desired experimental design and its choices enable how a user expresses the specification of the experimental design. The API design aids in other developers to leverage existing systems.

In this section, we discuss the interface designs of functions that output an experimental design based on three broad areas: factorial designs, recipe designs 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 visualisation of experimental data and the latter is about the analysis of

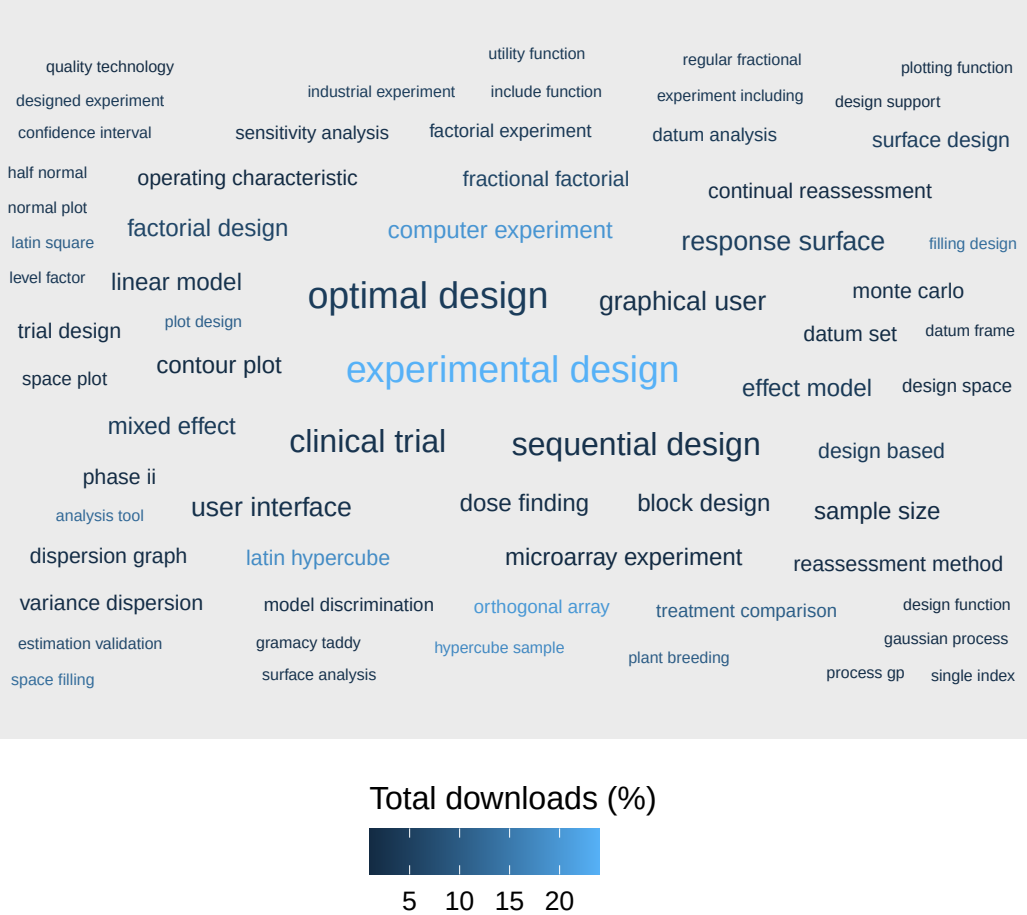

Figure 6: The above figure shows the word cloud of bigrams from the title and descriptions of the DoE packages. The size shows how often the bigram appears across the DoE packages and the color are relative to the total download count in 2021 of the packages that contain the bigram.

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 offers 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 some grouping structure. The effort to address this challenge is reflected by 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 designs via `conf.design()`, and `BHH2` package (Barrios 2016) generates 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

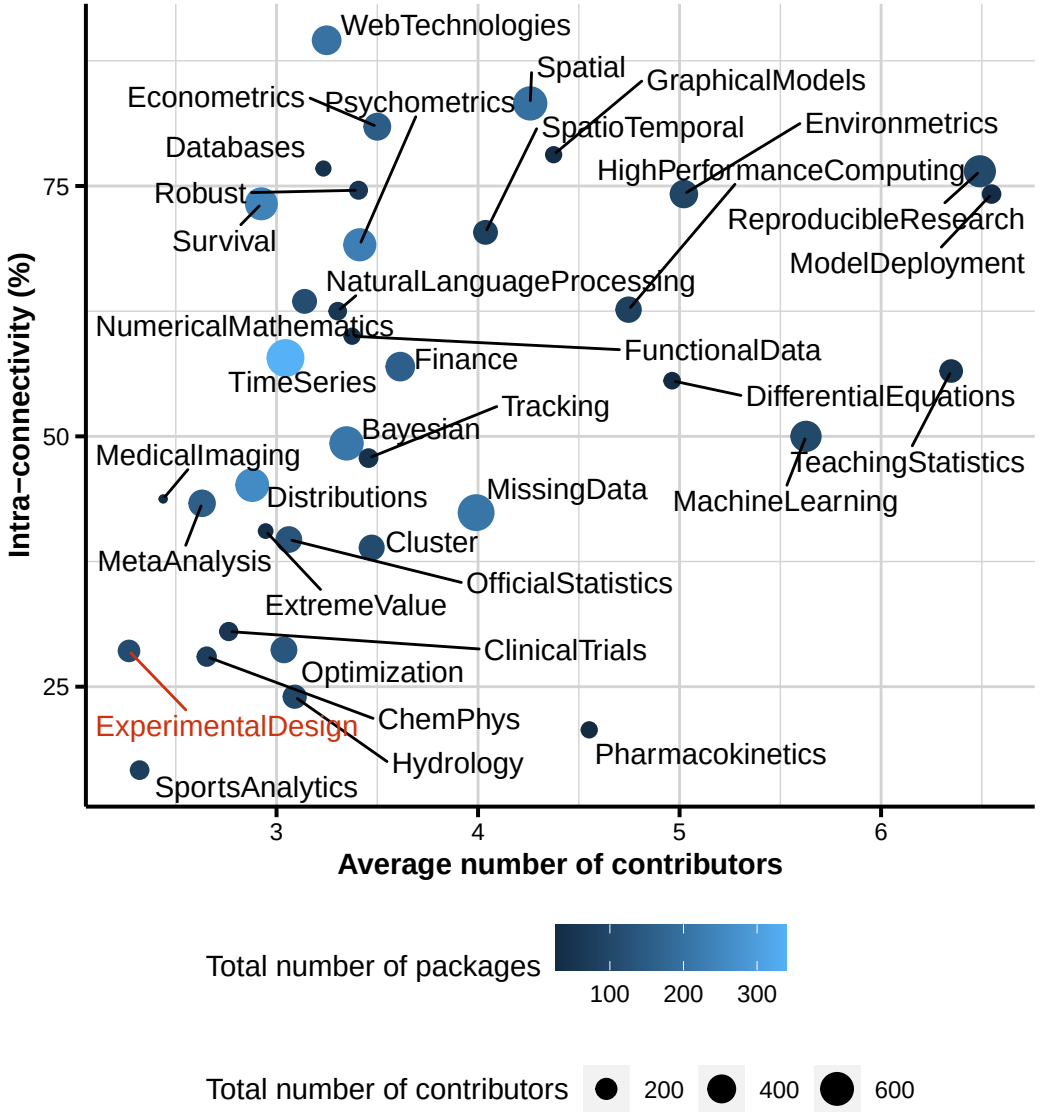

Figure 7: The above figure is a scatterplot of the intra-connectivity (the percentage of packages that depends, suggest or imports at least one other package within the same task view) and the average number of contributors for each CRAN task view. A low intra-connectivity suggests that development within the topic mostly occur in silos whilst 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 name. The label of ExperimentalDesign task view is colored 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 show in Table S1 in the Supplementary Material.

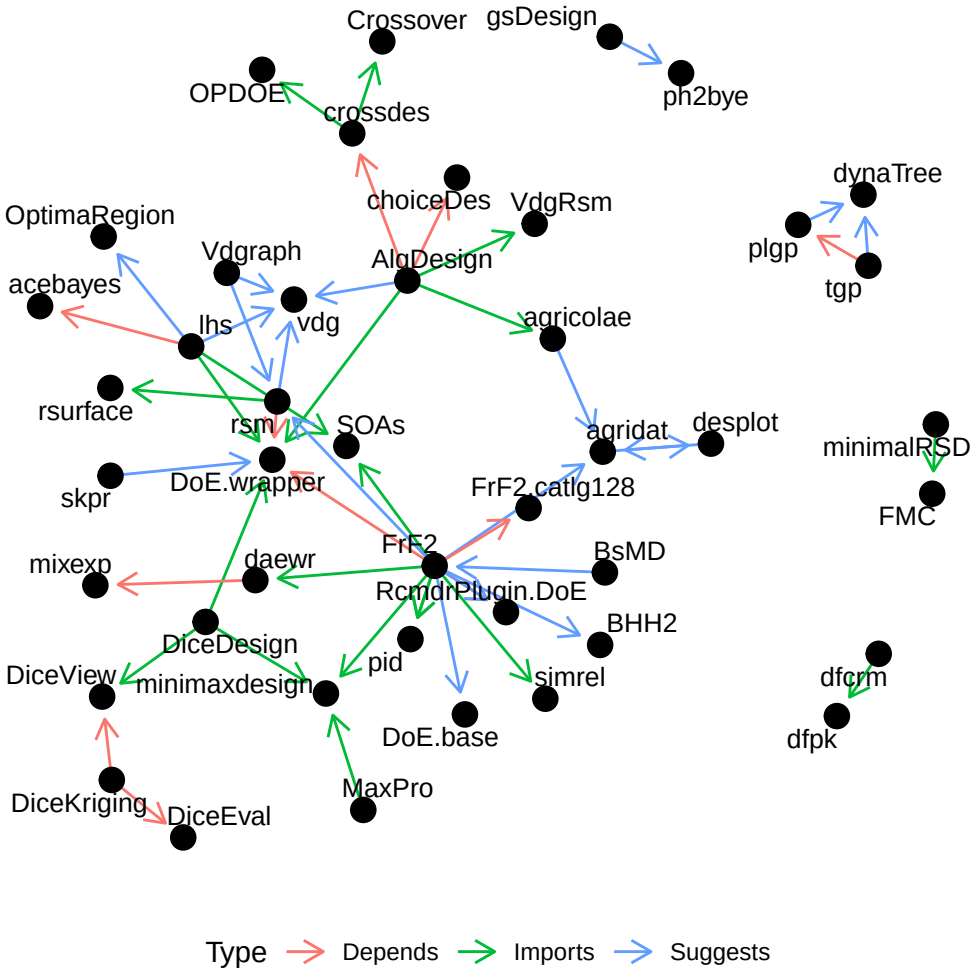

Figure 8: Package connections (depends, suggests and imports) within the DoE packages. The direction of the arrow shows the connection of packags where the package on the tail of the arrow is a depedency, suggestion or import for the package on the head of the arrow. DoE packages that do not depend, suggest or import another DoE package is not shown.

- levels for each factor if the design is 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 factor 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 designs are known by other names, e.g. *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 designs is *saturated designs* where typically higher-order interaction effects of treatment factors are confounded with the main effects. Plankett-Burnman designs (Plackett and Burman 1946) is one type of saturated design which 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. E.g. `design.crd()`, `design.rcbd()` and `design.split()` construct a completely randomised design, randomised complete block design and split-plot design, respectively. Users typically supply the treatment labels (or number of treatments in the case of `design.split()`) and the number of replications as argument to these functions and 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 corresponds 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()`. These functions in general require a data and formula in terms of the supplied data with the choice of criterion (e.g. 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 the optimal design 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 like `lhs` (Carnell 2022) and `DiceDesign` (Dupuy, Helbert, and Franco 2015). For the `lhs` package, functions like `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 optimisation schemes (in this case, random, S optimal and maxmin criteria, respectively; see package documentation for more details). Similarly for the `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 extra parameters for some) that implement algorithms that either maximise the entropy (Shewry and Wynn 1987), produce a random Latin hypercube sample, and use the WSP algorithm (Santiago, Claeys-Bruno, and Sergent 2012), respectively. These designs all output a $n \times p$ matrix with values between 0 and 1. Again, these functions employ a recipe style where each functions have a name that correspond to a certain search strategy in generating the experimental design.

4.3 The case of augmenting designs

Some functions in DoE packages require input of existing experimental designs to produce new designs. For example, the `DoE.base` package contain 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 using a Taguchi style (Taguchi 1986) to aggregate the designs with the inner and outer arrays. The `lhs` package contain the function that `augmentLHS()` to add additional samples to 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) – these require prior information, which are used to inform the next experimental design using `tgp.design()` and `dopt.gp()`. The user is required to supply candidate samples to sub-sample from and a model or a prior experimental design. Follow-up experiments, which can also be classified as a sequential design, is implemented by `BsMD` (Barrios 2020) using a model discriminant approach using `MD()`.

5 Discussion

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

There are a number of limitations and short coming in 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 only use the RStudio CRAN mirror download, which may bias our observations. Third, our analysis are limited to R-packages alone and many practioners may indeed be using 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, generalisable statement are possible. Regardless, the data driven nature of our analysis give an 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) is 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 make a choice of the generation mechanism by selecting an appropriate function. Arguably, the current dominant interface designs are not aligned with the way practitioners cognitively design experiments. Often the critical part of designing an experiment is understanding the experimental structure and the experimental context can govern or guide the choice of the algorithm to allocate treatments. In addition, the nature of recipe functions can obscure the understanding and relation of the designs (e.g. how do you go from an unstructured factorial design to split plot design?). Each new method to generating an experimental design appear to correspond to a totally new function and any intermediate results are often not easily accessible. These may be contributing factors to why developers often work in silos in the field of experimental design.

A consistent cognitive interface design that leverage existing developments and can be easily extensible for new methods will conceivably be of great help to practitioners. Some efforts to this end is seen in `DoE.wrapper` (Grömping 2020), which contain wrapper recipe functions to 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 for all experimental design needs so 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

This paper uses `targets` (Landau 2021) and `renv` (Ushey 2022) for reproducibility, `knitr` (Xie 2015) and `rmarkdown` (Xie, Allaire, and Grolemund 2018) for creating reproducible documents, `ggplot2` (Wickham 2016), `ggraph` (Pedersen 2021), `ggwordcloud` (Le Pennec and Slowikowski 2019) and `colorspace` (Zeileis et al. 2020) for visualisation, `kableExtra` (Zhu 2021) for customising the table in the Supplementary Material and `tidyverse` (Wickham et al. 2019), `tidytext` (Silge and Robinson 2016), `pluralize` (Rudis and Embrey 2020), and `ineq` (Zeileis 2014) for data processing and manipulation, and `cranlogs` (Csárdi 2019) and `ctv` (Zeileis 2005) for extracting data. All code to reproduce this paper is found at `https://github.com/emitanaka/paper-DoE-review`.

References

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.