SOFTWARE TOOL ARTICLE

REVISED ExploreModelMatrix: Interactive exploration for improved understanding of design matrices and linear models in R [version 2; peer review: 3 approved]

Charlotte Soneson [1,2], Federico Marini [3,4], Florian Geier [2,5], Michael I. Love [6,7], Michael B. Stadler [1,2]

[1]Friedrich Miescher Institute for Biomedical Research, Basel, Switzerland
[2]SIB Swiss Institute of Bioinformatics, Basel, Switzerland
[3]Center for Thrombosis and Hemostasis, Mainz, Germany
[4]Institute of Medical Biostatistics, Epidemiology and Informatics, Mainz, Germany
[5]Department of Biomedicine, University of Basel, University Hospital Basel, Basel, Switzerland
[6]Department of Biostatistics, University of North Carolina at Chapel Hill, Chapel Hill, North Carolina, USA
[7]Department of Genetics, University of North Carolina at Chapel Hill, Chapel Hill, North Carolina, USA

v2 **First published:** 04 Jun 2020, **9**:512
https://doi.org/10.12688/f1000research.24187.1
Latest published: 16 Sep 2020, **9**:512
https://doi.org/10.12688/f1000research.24187.2

Abstract

Linear and generalized linear models are used extensively in many scientific fields, to model observed data and as the basis for hypothesis tests. The use of such models requires specification of a design matrix, and subsequent formulation of contrasts representing scientific hypotheses of interest. Proper execution of these steps requires a thorough understanding of the meaning of the individual coefficients, and is a frequent source of uncertainty for end users. Here, we present an R/Bioconductor package, *ExploreModelMatrix*, which enables interactive exploration of design matrices and linear model diagnostics. Given a sample data table and a desired design formula, the package displays how the model coefficients are combined to give the fitted values for each combination of predictor variables, which allows users to both extract the interpretation of each individual coefficient, and formulate desired linear contrasts. In addition, the interactive interface displays informative characteristics for the regular linear model corresponding to the provided design, such as variance inflation factors and the pseudoinverse of the design matrix. We envision the package and the built-in collection of common types of linear model designs to be useful for teaching and self-learning purposes, as well as for assisting more experienced users in the interpretation of complex model designs.

Keywords

Linear Model, Experimental Design, Design Matrix, Shiny, R, Interactivity

Open Peer Review

Reviewer Status ✓ ✓ ✓

	Invited Reviewers		
	1	**2**	**3**
version 2 (revision) 16 Sep 2020	✓ report	✓ report	
version 1 04 Jun 2020	? report	? report	✓ report

1. **Jean Fan**, Johns Hopkins University, Baltimore, USA
Johns Hopkins University, Baltimore, USA

2. **Lucy D'Agostino McGowan**, Wake Forest University, Winston-Salem, USA

3. **Matthew Ritchie**, The Walter and Eliza Hall Institute of Medical Research, Parkville, Australia

Any reports and responses or comments on the article can be found at the end of the article.

This article is included in the Bioconductor gateway.

Corresponding author: Charlotte Soneson (charlottesoneson@gmail.com)

Author roles: Soneson C: Conceptualization, Methodology, Software, Validation, Writing – Original Draft Preparation, Writing – Review & Editing; **Marini F**: Methodology, Software, Validation, Writing – Review & Editing; **Geier F**: Conceptualization, Methodology, Software, Validation, Writing – Review & Editing; **Love MI**: Methodology, Software, Validation, Writing – Review & Editing; **Stadler MB**: Methodology, Validation, Writing – Original Draft Preparation, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

Grant information: The work of FM is supported by the German Federal Ministry of Education and Research (BMBF 01EO1003). MIL is supported by National Institutes of Health R01-HG009937.
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

How to cite this article: Soneson C, Marini F, Geier F *et al.* **ExploreModelMatrix: Interactive exploration for improved understanding of design matrices and linear models in R [version 2; peer review: 3 approved]** F1000Research 2020, **9**:512 https://doi.org/10.12688/f1000research.24187.2

First published: 04 Jun 2020, **9**:512 https://doi.org/10.12688/f1000research.24187.1

REVISED **Amendments from Version 1**

The manuscript has been revised in response to the reviewers' comments, specifically by:

* clarifying the target audience and the purpose of the package

* adding an additional use case (and a corresponding new Figure 2)

* updating Figure 1 to represent the current version of the software

Any further responses from the reviewers can be found at the end of the article

Introduction

Linear and generalized linear models are ubiquitous tools in a wide variety of scientific disciplines, and encompass well-known special cases such as linear and logistic regression, ANOVA and Student's t-test. Of particular interest to us, they are also the basis for many of the most widely used tools for analysis of high-throughput biological data. This includes *limma*[1,2] for linear modeling of gene expression microarray and similar data, as well as *edgeR*[3,4] and *DESeq2*[5] for differential expression analysis of RNA-seq and other count data, *missMethyl*[6], *DMRcate*[7] and *minfi*[8] for differential methylation analysis, *DiffBind*[9] for differential binding analysis, *msmsTests*[10] for mass spectrometry, and many others. Since the linear model is a special case of the generalized linear model, and particularly as the aspects of defining the design matrix are shared between the two, we will generally refer to generalized linear models in the rest of this manuscript.

Fitting a generalized linear model requires observations of a response variable y (e.g., inferred abundance levels of a gene) as well as a set of continuous or categorical predictor variables or sample annotations (e.g. the sample genotype, age, or treatment condition). In addition, in the R statistical programming environment, the user provides a *design formula*, specifying which, and how, provided predictor variables should be used to model the expected value of the response. The design formula in R is a version of a syntax for model specification originally proposed in 1973 by Wilkinson and Rogers[11]. This design formula and a specification of a type of contrast coding define a numeric $N \times J$ design matrix X, where N is the number of observations and J the number of model coefficients. The expected response values are then modeled by

$$E[y] = g^{-1}(X\beta),$$

where $\beta = (\beta_1, \ldots, \beta_J)$ are the regression coefficients for the respective columns of the design matrix, and g is a link function[12,13]. $X\beta$ is typically referred to as the *linear predictor*. After fitting the model, statistical tests can be performed to test the null hypothesis that a given combination of coefficients (referred to as a linear *contrast*) is zero. In this manuscript, we will focus on reference cell coding, or "treatment" coding for contrasts, though in general other schemes may also be considered. For more details on how R's design formula functionality is implemented, we refer to the reference for statistical modeling in S[14].

The way that the model is specified, that is, the definition of the design matrix, naturally determines how the model coefficients should be interpreted. As an example, consider a situation with a linear model and a single categorical predictor with two levels. Defining a model including an intercept (a column of the design matrix with the value 1 for all observations) implies that the second regression coefficient represents the difference between the average response values for the two levels of the predictor, while without the intercept, the two regression coefficients directly represent the average response values for the two factor levels. Given the versatility of generalized linear models, determining the proper contrast to use for testing a specific biological hypothesis of interest requires an understanding of the interpretation of the individual regression coefficients, and can be challenging for users of generalized linear model-based tools.

Here, we present *ExploreModelMatrix*[15], an R package for interactive exploration of generalized linear model designs, coefficients, and contrasts. Given a table of predictor variables, the user can specify the desired design formula and explore the value of the linear predictor for each combination of predictor values, expressed in terms of the model coefficients. From this type of visualization, it is often straightforward to determine the contrast corresponding to a given comparison of interest. We envision that *ExploreModelMatrix* can be useful for both research and teaching purposes. Specifically for the latter, the application contains several built-in example data sets, corresponding to some of the most commonly used experimental design setups. The underlying function in *ExploreModelMatrix* that processes the input data and generates visualizations can also be directly called by the user, enabling the generation of static plots for inclusion in reports and educational material. It is worth noting that *ExploreModelMatrix* is not intended as a self-contained resource on generalized linear models, but rather as a complement to existing books and courses on the topic, and the application contains a list of suggested material for further study.

Methods

Operation

ExploreModelMatrix[15] is implemented as an R package[16], using the Shiny framework[17]. The package is available via Bioconductor[18] (from release 3.11 onwards), with the current development version accessible via GitHub. The package has been tested with R version 3.6 and later.

An instance of the interactive application is launched by calling the `ExploreModelMatrix()` function. This function accepts two optional arguments; a `data.frame` with one row per observation and each column corresponding to a measured predictor variable (below referred to as the *sample data table*), and a design formula. If the `ExploreModelMatrix()` function is called without any arguments, the user can either explore one of the built-in designs, or load a sample data table

from a tab-separated text file. The design formula can always be specified or modified interactively in the application. If the user wishes to generate the visualizations independently of the interactive interface, this can be achieved via the `VisualizeDesign()` function, which is also called internally by `ExploreModelMatrix()`.

Implementation

The user interface of *ExploreModelMatrix* consists of a side bar with control widgets and a main window containing a set of fixed, but collapsible, panels, each illustrating a different aspect of the design matrix or the associated standard linear model (Figure 1). A more detailed explanation of each panel is

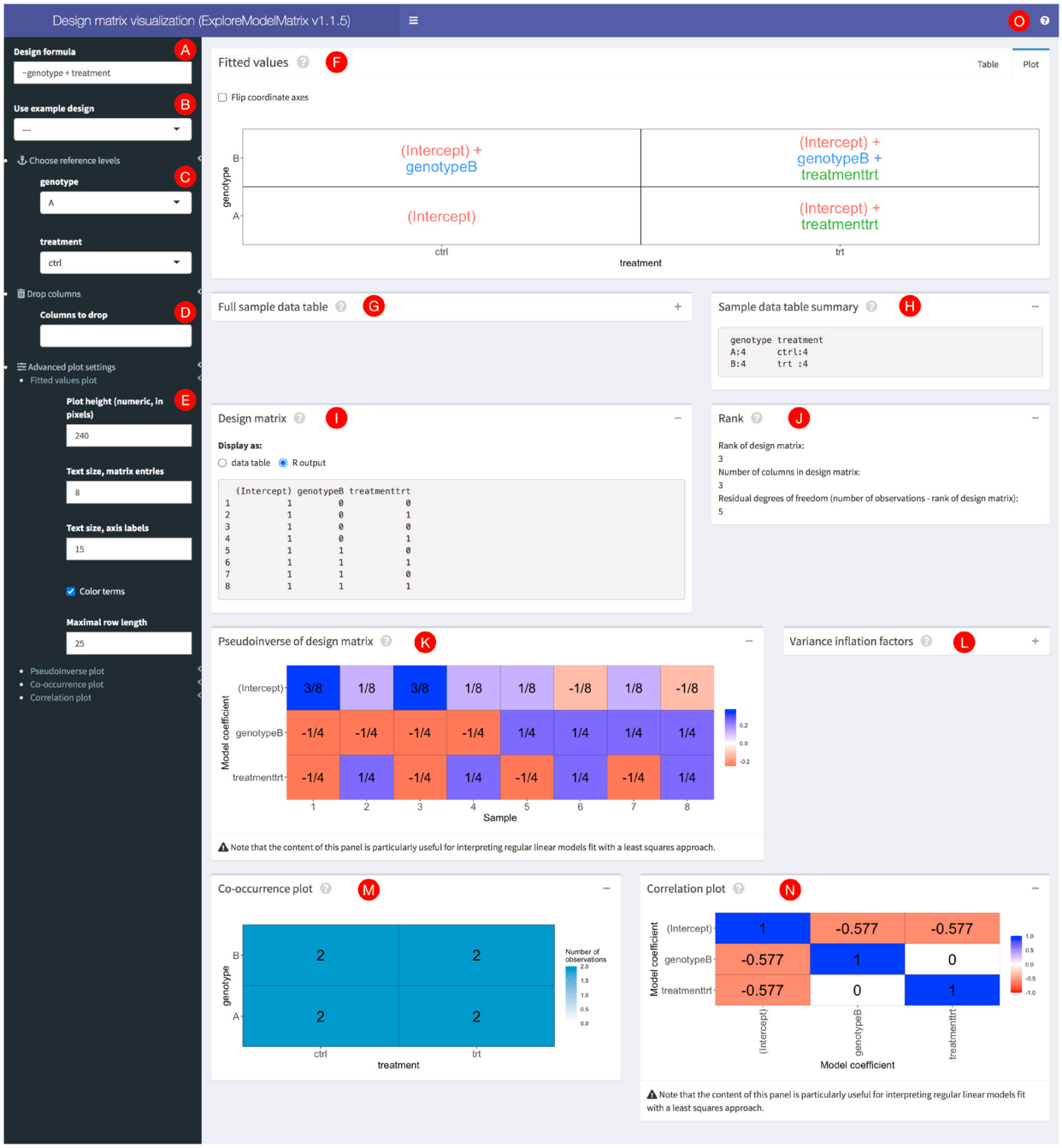

Figure 1. Screenshot of the *ExploreModelMatrix* interface. This example shows a model with two predictors (genotype and treatment), each with two levels, and with the assumption that their effects are additive. Red circles with letters were added to be able to refer to specific parts of the interface in the text.

accessible via the guided tour of the interface, implemented via the *rintrojs* package[19] and accessible by clicking on the question mark icon in the top right corner (represented by the letter `O` in Figure 1). In addition, clicking on the question mark icon within a specific panel opens up the guided tour at the corresponding step.

Given a sample data table and a design formula, either provided by the user or obtained via one of the built-in designs, *ExploreModelMatrix* will first check that the two objects are compatible, i.e., that the terms in the design formula use only variables that are present in the sample data table, and that the design formula is supported by the package. If no problems are detected, *ExploreModelMatrix* will create a design matrix using the `model.matrix()` R function. The full sample data table, a summary of its columns, and the resulting design matrix are all displayed in the application interface for convenience (see `G-I` in Figure 1). In addition, the rank of the design matrix is calculated (`J`). If the design matrix is not full rank, *ExploreModelMatrix* will display a warning, together with an indication of the coefficients that are not estimable (using the `nonEstimable()` function from the *limma* R package[1,2]). In addition, *ExploreModelMatrix* will inform the user if the number of rows (observations) in the design matrix is the same as its rank, in which case there are no residual degrees of freedom, and the variance or dispersion cannot be estimated.

Expressed in terms of the model coefficients, the panel in the first row of the application (`F`) illustrates, in graphical and tabular form, the value of the linear predictor in a generalized linear model, for each combination of levels for the predictors used in the design formula. This provides an intuitive understanding of the interpretation of each of the model coefficients, and can be helpful for specifying appropriate contrasts.

The panels in the lower part of the interface (`K`, `L`, `N`) should largely be interpreted in the context of standard linear models, where coefficient estimates are obtained using least squares fitting. The pseudoinverse $P=(X^T X)^{-1} X^T$[20–22] represents the way each observed response value would contribute to the coefficient estimates. More precisely, in such a linear model represented by

$$y = X\beta + \varepsilon,$$

the estimated regression coefficients are given by

$$\hat{\beta} = \left(X^T X\right)^{-1} X^T y = Py.$$

ExploreModelMatrix also estimates variance inflation factors and correlations among the coefficient estimates. Finally, the co-occurrence plot in the bottom left panel (`M`) shows the number of observations in the data set for each combination of levels of the predictor variables.

The controls in the left-hand sidebar can be used to interactively modify the studied design as well as the display parameters of the panels. The text box in the top (`A`) allows the user to type in a design formula (starting with the ~, or "tilde" character), and the displayed figures will be updated accordingly. The dropdown menu immediately below (`B`) contains the built-in example designs. To use the sample data table provided either as an argument to `ExploreModelMatrix()` or uploaded into the app at run time, select `---` here. The next section of controls (`C`) lets the user control which level should be considered the "baseline" or reference level for each categorical or factor variable in the model. *ExploreModelMatrix* will convert each character variable to a factor when a sample data table is loaded; by default the baseline level will be the first in alphabetical order.

In cases where the design matrix is not of full rank, it may be desirable to exclude a subset of the columns in the design matrix (for example, columns with all zero values or columns that are linear combinations of other columns). This can be done in the "Drop columns" section (`D`). As mentioned above, in the case of a non-full rank design matrix, *ExploreModelMatrix* will indicate which coefficients are not estimable and thus candidates for being dropped. The final group of controls (`E`) provide the ability to change the way the panels are displayed, e.g. by setting the height of the plot panels and changing the size and display mode of the text.

Use case

Figure 1 illustrates the *ExploreModelMatrix* output for a factorial design with two predictors (genotype and treatment). We consider the effects of the two predictors to be additive, which is indicated by the design formula (~ genotype + treatment). From the graphical representation of the fitted values (also shown in Figure 2A), we can, for example, conclude that the intercept in the model directly represents the fitted value for the 'genotype A, control' group of samples. Similarly, the fitted value for the 'genotype A, treated' group is given by the sum of the intercept and the `treatmenttrt` coefficient. If we are interested in performing a hypothesis test to compare the treated and control groups for the samples with genotype A, we need to formulate a suitable linear contrast. Using the *ExploreModelMatrix* representation, the estimated effect size can be obtained by subtracting the fitted value for the 'genotype A, control' group from that of the 'genotype A, treated' group. The result is simply `treatmenttrt`, which indicates that a significance test for the difference between the two treatment groups in genotype A samples can be obtained by testing whether the coefficient `treatmenttrt` is zero. Interestingly, performing the same exercise in the genotype B samples yields the same result, indicating that the `treatmenttrt` coefficient represents the treatment effect in each of the two genotypes. This is a result of using an additive model. Changing the provided design formula to include an interaction between the two predictors (~ genotype + treatment + genotype:treatment; Figure 2B) changes the interpretation. Now, while the treatment effect in the genotype A samples is still represented by the `treatmenttrt` coefficient, the treatment effect in the genotype B samples is represented by the sum of the `treatmenttrt` and `genotypeB:treatmenttrt` coefficients. The interaction effect, that

is, the difference between the treatment effects in the two genotype groups, is represented by the `genotypeB:treatmenttrt` coefficient. This example illustrates how the *ExploreModelMatrix* interface can be used to interpret coefficients in generalized linear models and create contrasts of interest.

To further illustrate how *ExploreModelMatrix*[15] can be used to interpret the coefficients in a complex experimental design, we consider the example of differential allele-specific expression analysis with RNA-seq data. Generalized linear models for count data often use the log link function, and we assume this to be the case in some of the interpretations below. This type of experiment contains different groups of subjects (e.g., from different experimental conditions), where each subject contributes two columns in the read count matrix: one representing the read counts for the reference allele, and one representing those for the alternative allele, for each considered gene. Typical scientific questions of interest are whether there are differences

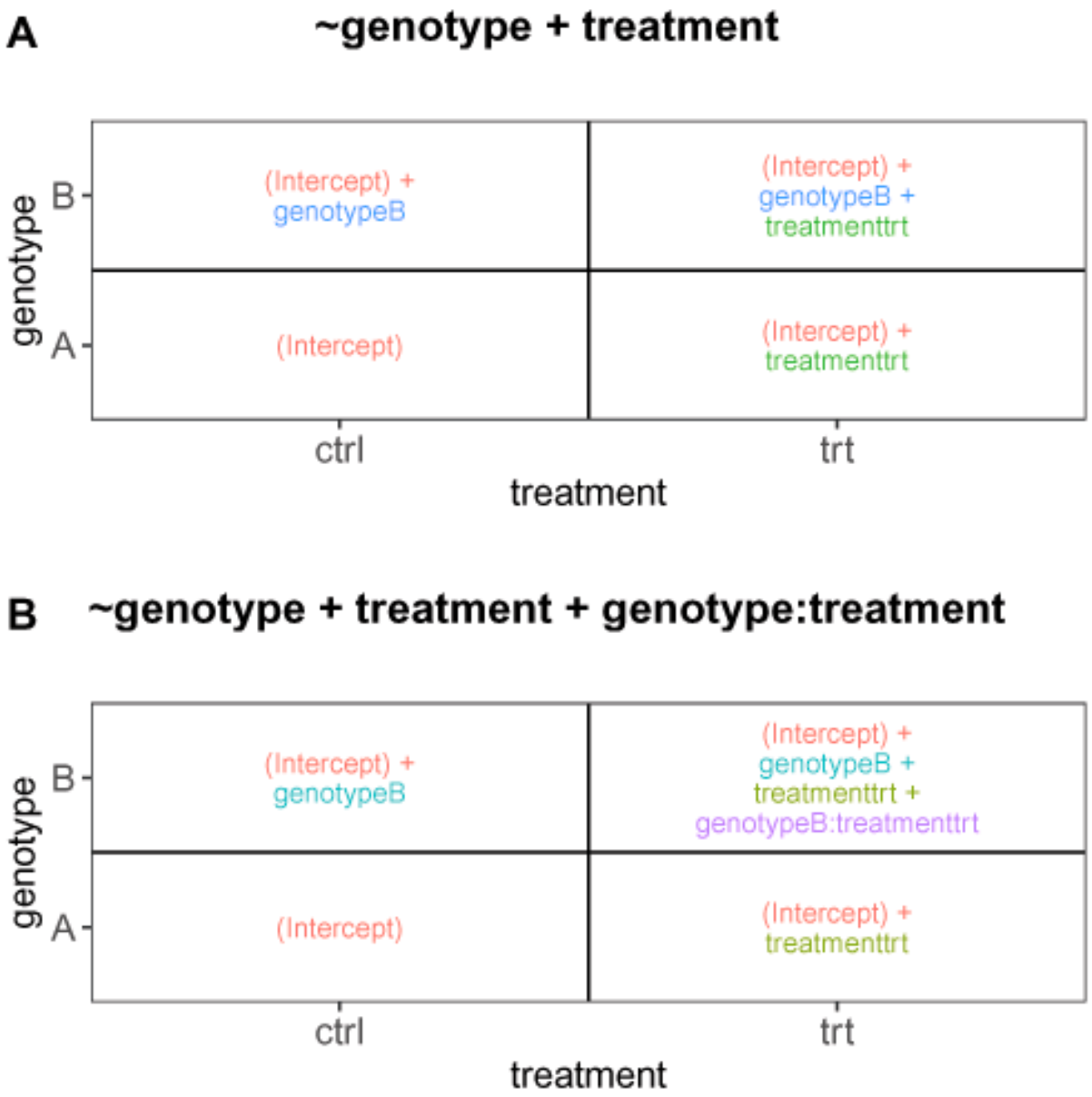

Figure 2. Values of the linear predictor, in terms of the model parameters, for the two-factor example design, with and without an interaction term.

between the expression of the two alleles within each condition, and whether there are differences in the allele-specific expression patterns between the conditions. Similar setups can be observed, for example, in differential methylation experiments (where the two columns for each sample would correspond to methylated and unmethylated read counts for a feature), or in situations where individuals from different groups are each given the same set of treatments.

The sample data table considered here is provided in Table 1. In addition to the columns containing the subject identifier, the condition and the count type (reference or alternative allele), we include a column corresponding to a within-condition relabeling, or dummy encoding, of the subject identifier. Note that this dummy subject identifier has only three levels, compared to six for the original subject identifier. This design setup is available among the example designs provided within *ExploreModelMatrix*, denoted "Two crossed, one nested factor (manuscript example)". We will illustrate two equivalent ways of setting up the design formula, and show how *ExploreModelMatrix* can help in the interpretation of the model coefficients.

First, we specify the design formula

~condition + condition:subjectdummy + condition:count,

including an overall condition effect, a term to account for sample-specific effects, and an interaction between the condition and count type columns to capture allele-specific expression within each condition. In R's design formula syntax, a ":" between two variable names indicates the addition of an interaction term between these two variables, which may have a different effect on columns of X depending on whether these are numeric or factor variables, and what other terms are in the design. Given this design formula together with the sample data table from Table 1 as the input arguments, the *ExploreModelMatrix* functions determine the composition of the linear predictor for each combination of predictor variables shown in Figure 3A (corresponds to panel (`F`) in Figure 1, shown here separately for increased readability). The Rank panel in the application further indicates that the design matrix is of full rank and that the residual degrees of freedom is non-zero, allowing also estimation of variances or dispersions for use in statistical hypothesis tests involving the estimated coefficients. The illustration in Figure 3A can be used to extract appropriate contrasts for statistical testing. For example, comparing the values of the linear predictor for each sample in the control group, we can see that the `conditioncontrol:countalt` coefficient represents the allele-specific expression effect (alt/ref expression log-ratio) in this group. Similarly, the `conditiontreated:countalt` coefficient represents the allele-specific expression in the treated group. As a consequence, the condition-dependent allele-specific expression effect is obtained as the difference between the allele-specific effects within the respective conditions, that is, by `conditiontreated:countalt - conditioncontrol:countalt`.

Table 1. Sample data table for the allele-specific differential expression use case.

subject	count	condition	subjectdummy
S1	ref	control	D1
S1	alt	control	D1
S2	ref	control	D2
S2	alt	control	D2
S3	ref	control	D3
S3	alt	control	D3
S4	ref	treated	D1
S4	alt	treated	D1
S5	ref	treated	D2
S5	alt	treated	D2
S6	ref	treated	D3
S6	alt	treated	D3

Next, we illustrate an alternative way of setting up the design matrix, by specifying the design formula as

~ condition*count + subject.

Here, we use the original subject ID (not the dummy encoded), and include main effects for condition and count type as well as an interaction between the condition and the count type. In R's design formula syntax, a "*" between two variable names indicates the addition of both main effects and an interaction term between these two variables. Upon changing the design formula in *ExploreModelMatrix*, we are notified that the design matrix is no longer full rank, as a consequence of having different subjects in the different conditions. Dropping the `subjectS4` column results in a full-rank design matrix, and the composition of the linear predictor is shown in Figure 3B. The rank of the design matrix, as well as the residual degrees of freedom, are the same as with the previous formulation. However, the composition of the linear predictor for each combination of input variables is different. Comparing the alternative and reference allele groups for the control condition shows that with this formulation, the allele specificity in the control group is encoded by the `countalt` coefficient. Similarly, the allele specificity in the treated group is represented by the sum of the `countalt` and `conditiontreated:countalt` coefficients. Consequently, the difference in allele specificity between the treated and control group is now directly encoded in the `conditiontreated:countalt` coefficient.

Both model formulations can be used to analyze this type of data, and the purpose of *ExploreModelMatrix* is not to select the 'best' among a set of plausible models, but rather to assist the user in the interpretation of a chosen model. The example above stresses that knowing how to interpret a given

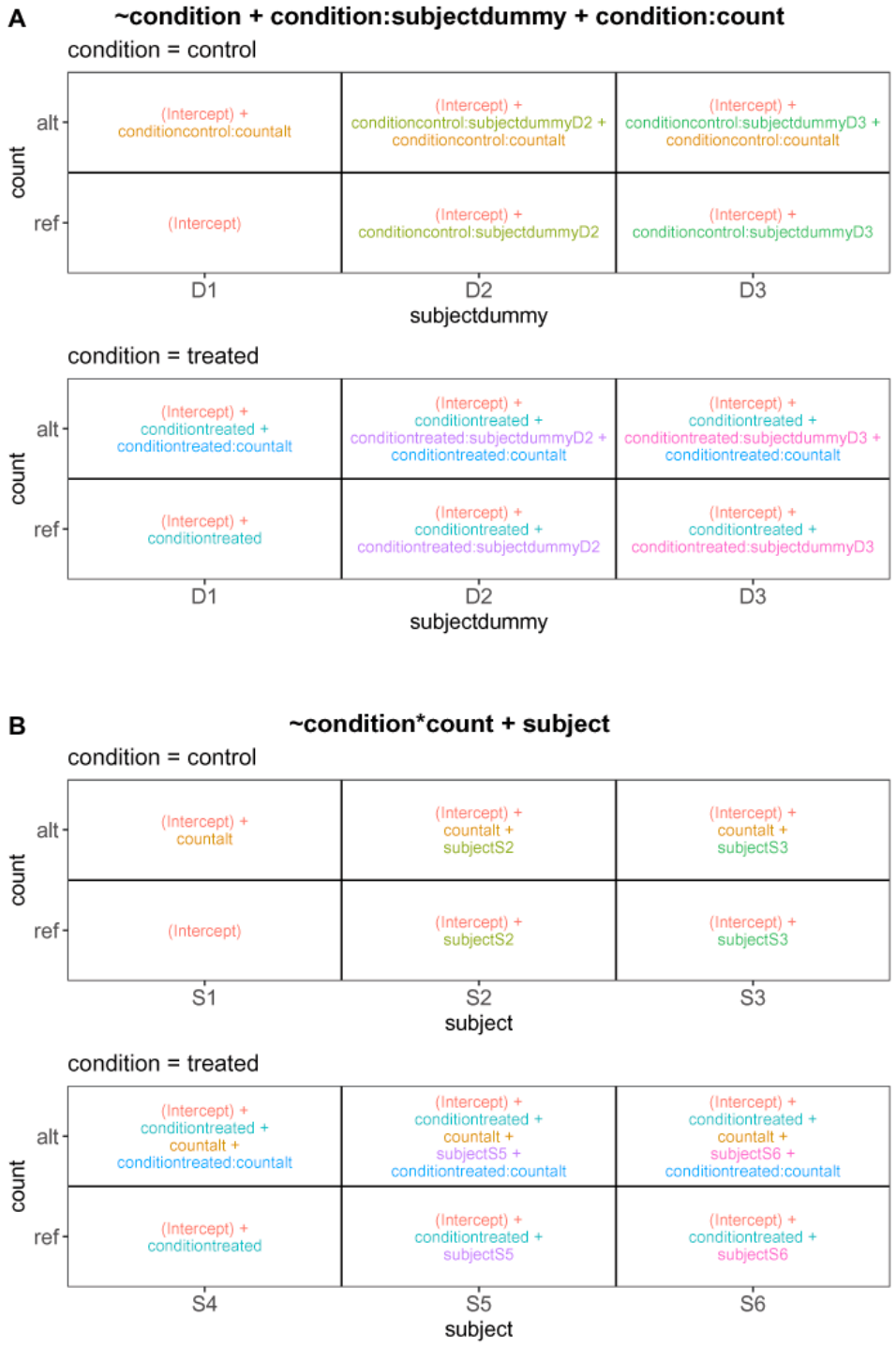

Figure 3. Values of the linear predictor, in terms of the model parameters, for the allele-specific expression use case. A. Using the design formula ~ condition + condition:subjectdummy + condition:count. **B**. Using the design formula ~ condition*count + subject.

coefficient in a generalized linear model is critical, that identically labelled coefficients can have different meanings depending on the chosen design formula, and that *ExploreModelMatrix* can help the user interpret the resulting coefficients for a given choice of design formula and set up an appropriate contrast.

Summary

We have described the *ExploreModelMatrix* R/Bioconductor package[15], which enables interactive exploration for increased understanding of model coefficients in linear and generalized linear models. To the best of our knowledge, this is the first package of its kind, and we envision applications for both research and educational purposes. The application requires minimal input and can be launched from a local R session, as well as be deployed on a Shiny server. An example instance of the latter is available at http://shiny.imbei.uni-mainz.de:3838/ExploreModelMatrix/, and the process for deploying an instance of the application on a Shiny server is documented in one of the vignettes accompanying the software.

Data availability

All data underlying the results are available as part of the article and no additional source data are required.

Software availability

***ExploreModelMatrix* is available at:** http://www.bioconductor.org/packages/ExploreModelMatrix/

Source code available at: https://github.com/csoneson/ExploreModelMatrix.

Source code at time of publication: https://doi.org/10.5281/zenodo.4008415[15].

License: MIT License.

References

1. Smyth GK: **Linear models and empirical Bayes methods for assessing differential expression in microarray experiments.** *Stat Appl Genet Mol Biol.* 2004; **3**(1): Article 3.
PubMed Abstract | Publisher Full Text
2. Ritchie ME, Phipson B, Wu D, *et al.*: ***limma* powers differential expression analyses for RNA-sequencing and microarray studies.** *Nucleic Acids Res.* 2015; **43**(7): e47.
PubMed Abstract | Publisher Full Text | Free Full Text
3. Robinson MD, McCarthy DJ, Smyth GK: **edgeR: a Bioconductor package for differential expression analysis of digital gene expression data.** *Bioinformatics.* 2010; **26**(1): 139–140.
PubMed Abstract | Publisher Full Text | Free Full Text
4. McCarthy DJ, Chen Y, Smyth GK: **Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation.** *Nucleic Acids Res.* 2012; **40**(10): 4288–4297.
PubMed Abstract | Publisher Full Text | Free Full Text
5. Love MI, Huber W, Anders S: **Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2.** *Genome Biol.* 2014; **15**(12): 550.
PubMed Abstract | Publisher Full Text | Free Full Text
6. Phipson B, Maksimovic J, Oshlack A: **missMethyl: an R package for analyzing data from Illumina's HumanMethylation450 platform.** *Bioinformatics.* 2016; **32**(2): 286–288.
PubMed Abstract | Publisher Full Text
7. Peters TJ, Buckley MJ, Statham AL, *et al.*: ***De novo* identification of differentially methylated regions in the human genome.** *Epigenet Chromatin.* 2015; **8**: 6.
PubMed Abstract | Publisher Full Text | Free Full Text
8. Aryee MJ, Jaffe AE, Corrada-Bravo H, *et al.*: **Minfi: a flexible and comprehensive Bioconductor package for the analysis of Infinium DNA methylation microarrays.** *Bioinformatics.* 2014; **30**(10): 1363–1369.
PubMed Abstract | Publisher Full Text | Free Full Text
9. Stark R, Brown G: **DiffBind: differential binding analysis of ChIP-Seq peak data.** R package version 2.15.2. 2011.
Publisher Full Text
10. Gregori J, Sanchez A, Villanueva J: **msmsTests: LC-MS/MS Differential Expression Tests.** R package version 1.25.0 . 2019.
Publisher Full Text
11. Wilkinson GN, Rogers CE: **Symbolic description of factorial models for analysis of variance.** *J R Stat Soc Ser C (Applied Statistics).* 1973; **22**(3): 392–399.
Publisher Full Text
12. Nelder JA, Wedderburn RWM: **Generalized linear models.** *J R Stat Soc Ser A.* 1972; **135**(3): 370–384.
Publisher Full Text
13. McCullagh P, Nelder JA: **Generalized Linear Models**. Second Edition. *Chapman and Hall/CRC Monographs on Statistics and Applied Probability Series.* Chapman & Hall, 1989. ISBN 9780412317606.
Reference Source
14. Chambers JM, Hastie T: **Statistical Models in S**. *Wadsworth & Brooks/Cole computer science series*. Wadsworth & Brooks/Cole Advanced Books & Software, 1992. ISBN 9780534167646.
Reference Source
15. Soneson C, Marini F, Geier F, *et al.*: **ExploreModelMatrix**. August 2020.
http://www.doi.org/10.5281/zenodo.4008415
16. R Core Team: **R: A language and environment for statistical computing**. 2020.
Reference Source
17. Chang W, Cheng J, Allaire JJ, *et al.*: **shiny: Web Application Framework for R**. R package version 1.4.0.2. 2020.
Reference Source
18. Huber W, Carey VJ, Gentleman R, *et al.*: **Orchestrating high-throughput genomic analysis with Bioconductor.** *Nat Methods.* 2015; **12**(2): 115–121.
PubMed Abstract | Publisher Full Text | Free Full Text
19. Ganz C: **rintrojs: A wrapper for the intro.js library.** *J Open Source Softw.* 2016; **1**.
Publisher Full Text
20. Moore EH: **On the reciprocal of the general algebraic matrix.** *B Am Math Soc.* 1920; **26**: 394–395.
21. Bjerhammar A: **Application of calculus of matrices to method of least squares; with special references to geodetic calculations.** *Trans Roy Inst Tech Stockholm.* 1951; **49**.
22. Penrose R: **A generalized inverse for matrices.** *Math Proc Cambridge.* 1955; **51**(3): 406–413.
Publisher Full Text

Open Peer Review

Current Peer Review Status: ✔ ✔ ✔

Version 2

Reviewer Report 02 November 2020

https://doi.org/10.5256/f1000research.29479.r71434

✔ **Jean Fan**
[1] Department of Biomedical Engineering, Johns Hopkins University, Baltimore, MD, USA
[2] Department of Computer Science, Johns Hopkins University, Baltimore, MD, USA

The authors have addressed my main concerns and the manuscript is now suitable for publication in F1000Research.

However, in order to ensure the utility of ExploreModelMatrix both as a guided and self-learning teaching tool, I would strongly encourage the authors to eventually conduct a user study, perhaps in a classroom setting, to assess whether their tool and the features of their tool achieve their stated goal of "helping users interpret a given design and to understand how changing the design influences the way they would perform a given comparison of interest."

For example, I still find the "Choose reference levels" dropdown panels with the options ordered alphabetically rather by the given level ordering to be confusing from a user standpoint, particularly given that the fitted values plot does use the level ordering. While the authors suggest that the alphabetical ordering "was intentionally designed in this way" as they "reasoned that it's typically easier to find the desired level in a list of the options are listed alphabetically", which ordering is more intuitive for users is best assessed through an actual user study (rather than based on an N=1 feedback from myself).

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Bioinformatics, Applied Statistics

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Reviewer Report 12 October 2020

https://doi.org/10.5256/f1000research.29479.r71435

Lucy D'Agostino McGowan
Department of Mathematics and Statistics, Wake Forest University, Winston-Salem, NC, USA

Thank you for addressing the points of my review, this paper and package will provide a nice teaching tool.

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Biostatistics

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Version 1

Reviewer Report 13 July 2020

https://doi.org/10.5256/f1000research.26680.r64395

Matthew Ritchie
The Walter and Eliza Hall Institute of Medical Research, Parkville, Vic, Australia

Understanding how to set up a design matrix is a significant challenge in genomic data science, especially for new analysts. The manuscript by Soneson *et al.* presents the ExploreModelMatrix R package which aims to give the user better intuition on this for any arbitrary design.

This is great contribution and would be really useful in a teaching setting (e.g. when running an Intro to RNA-seq analysis course) to help participants understand how a linear model is parameterised and how this can be changed to suit different biological questions. At the other extreme it is also helpful in the interpretation of coefficients in more complicated designs that include interactions.

The ExploreModelMatrix output is provided via a shiny app which allows the user to interactively change either their own design based on the data.frame supplied, or choose from a series of standard examples, which was easy to navigate. I particularly liked the interactive tour of the different elements of the interface provided by rintrojs.

Overall, I really enjoyed using this package and list a few optional suggestions below to help further improve the work.

1. It might be useful to add a sentence about the intended audience of the package to the abstract.

2. Can a window be given to show what the line of code looks like to make the design matrix? Just thinking about beginners, who could start the app with ExploreModelMatrix(), choose an example design and then immediately see what code they would need to run at the R command prompt to create the design they're interested in (again helpful for teaching). This would provide a concrete output that the user could take forward in their analysis with a simple copy and paste. Likewise, if they provided their own data frame they will know what to do with it in terms of specifying model.matrix().

3. Would it be possible to create a dummy x vs y plot (perhaps by simulating data) to show what a fit might look like for theoretical y? This might be rather complicated to implement in practice given the wide array of models one could envisage, however, such a display would provide an intuitive view of what the coefficients represent graphically.

4. If a model is parameterised without an intercept, it will generally be necessary to define contrasts between coefficients. Is there a way to point this out to the user in the app, or can a module be added to help set-up contrasts to show how this is done and again provide code that the analyst could then use in their analysis?

5. On page 3, column 2, paragraph 4, line 2 there appears to be a formatting problem with th'e' in 'the ExploreModelMatrix()'

6. In the 'Use cases' section, I wondered whether the simple example that features in Figure 1 could be stepped through (again to appeal to beginners) in addition to the complex ASE analysis example of Table 1 and Figure 2 which is likely to be more niche.

7. In the interactive tour of the interface, the rendering of some equations hasn't occurred properly in steps 19 and 22.

8. When you 'Flip coordinates' in the Fitted values / Co-occurrence plot, the y-axis labels aren't preserved in the flip (not sure if this is intentional)

Is the rationale for developing the new software tool clearly explained?

Yes

Is the description of the software tool technically sound?

Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Yes

Is sufficient information provided to allow interpretation of the expected output datasets

and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Gene expression analysis

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Author Response 07 Sep 2020

Charlotte Soneson, Friedrich Miescher Institute for Biomedical Research, Basel, Switzerland

Understanding how to set up a design matrix is a significant challenge in genomic data science, especially for new analysts. The manuscript by Soneson et al. presents the ExploreModelMatrix R package which aims to give the user better intuition on this for any arbitrary design. This is great contribution and would be really useful in a teaching setting (e.g. when running an Intro to RNA-seq analysis course) to help participants understand how a linear model is parameterised and how this can be changed to suit different biological questions. At the other extreme it is also helpful in the interpretation of coefficients in more complicated designs that include interactions.
The ExploreModelMatrix output is provided via a shiny app which allows the user to interactively change either their own design based on the data.frame supplied, or choose from a series of standard examples, which was easy to navigate. I particularly liked the interactive tour of the different elements of the interface provided by rintrojs.
Overall, I really enjoyed using this package and list a few optional suggestions below to help further improve the work.

Thank you for your constructive comments and careful evaluation of the software. Point-by-point responses are provided below.

It might be useful to add a sentence about the intended audience of the package to the abstract.

We have added a sentence about this in the abstract.

Can a window be given to show what the line of code looks like to make the design matrix? Just thinking about beginners, who could start the app with ExploreModelMatrix(), choose an example design and then immediately see what code they would need to run at the R command prompt to create the design they're interested in (again helpful for teaching). This would provide a concrete output that the user could take forward in their analysis with a simple copy and paste. Likewise, if they provided their own data frame they will know what to do with it in terms of specifying model.matrix().

This is an interesting point and something that we may consider in a future version. One challenge is that different analysis packages require the design to be specified in different ways (e.g., as a design formula, or as a design matrix), and in some cases will automatically deal with e.g. non-estimable parameters. Thus, it is not obvious how to provide this code to make it as useful as possible.

Would it be possible to create a dummy x vs y plot (perhaps by simulating data) to show what a fit might look like for theoretical y? This might be rather complicated to implement in practice given the wide array of models one could envisage, however, such a display would provide an intuitive view of what the coefficients represent graphically.

We agree that such plots are often instructive for the user, and some examples are included e.g. in the Data Analysis for the Life Sciences book, which we now also refer to in the application as an additional resource. Implementing them in a very general context, on the other hand, is a non-trivial task, and furthermore there is a risk that a user is misled if the coefficients estimated from the simulated response are different (perhaps with opposite signs) compared to those obtained from their actual data. For these reasons, we currently consider this out of scope for ExploreModelMatrix.

If a model is parameterised without an intercept, it will generally be necessary to define contrasts between coefficients. Is there a way to point this out to the user in the app, or can a module be added to help set-up contrasts to show how this is done and again provide code that the analyst could then use in their analysis?

We have expanded on the use case section in the manuscript to provide additional guidance on how the ExploreModelMatrix interface can be used to generate contrasts.

On page 3, column 2, paragraph 4, line 2 there appears to be a formatting problem with th'e' in 'the ExploreModelMatrix()'

Thanks for pointing this out. We hope that this has been resolved in typesetting of the revised version.

In the 'Use cases' section, I wondered whether the simple example that features in Figure 1 could be stepped through (again to appeal to beginners) in addition to the complex ASE analysis example of Table 1 and Figure 2 which is likely to be more niche.

In the revised manuscript, we have added a paragraph walking through the design in Figure 1 and showing how ExploreModelMatrix can be used to interpret coefficients and generate contrasts of interest.

In the interactive tour of the interface, the rendering of some equations hasn't occurred properly in steps 19 and 22.

This is correct - unfortunately the rintrojs package currently doesn't support proper rendering of equations via e.g. MathJax. We have initiated a discussion with the developer and hope that a

solution can be engineered - in the meantime, we hope that the non-rendered equations still provide some help with the interpretation.

When you 'Flip coordinates' in the Fitted values / Co-occurrence plot, the y-axis labels aren't preserved in the flip (not sure if this is intentional)

We were not able to reproduce this behaviour - when the coordinates are flipped, the labels are flipped as well. If you would be willing to open an issue in the GitHub repository with a reproducible example, that would be super helpful!

Competing Interests: No competing interests were disclosed.

Reviewer Report 07 July 2020

https://doi.org/10.5256/f1000research.26680.r64393

?

Lucy D'Agostino McGowan
Department of Mathematics and Statistics, Wake Forest University, Winston-Salem, NC, USA

Summary

This article describes an R package and corresponding shiny application, ExploreModelMatrix, that can be used to explore design matrices for linear models. I can see this having a lot of utility for helping partitioners understand their design for simple research questions or for pedagogical use in the classroom. While I see this being extremely useful, the current design falls somewhere in the middle, a bit basic for complex research questions and a bit too complex for those with little statistical background. My suggestions assume that the primary user will be in the latter group.

Major Comments

1. The main take away seems to be from the "fitted values" boxes (both the plot and the table at the top of the application display this information). These seem to show the exact same content, so it feels repetitive to have both. The table is much more legible, but I can also see the utility of visualizing this. Perhaps these should be a *single* box with two tabs, one with the table and one with the plot. I find the content in these depictions of the fitted values confusing, the variable names here are referring to the beta coefficients from the model, not the variable values themselves. To someone familiar with R / model output this may be obvious, but I'm not sure that is the case for the target user. Additionally, the use of : to indicate an interaction is not something I would expect novice users to know. If this is meant to help users interpret/calculate fitted values after they fit their models, perhaps the "pseudo" model output could be printed above the "fitted values" plot/table. This would explain where the values that are being plugged into each of these variable names come

from. Additionally, this fitted values plot becomes quickly illegible, it would be great to have some simple defaults built in to expand the plot space if there are several inputs (see #5).

2. There are several terms used in the application that may not be familiar to those without statistical/mathematical backgrounds. It would be great to have definitions provided to explain the plots/terms. For example, you could have a question mark next to each term that will pop up an explanation when the user hovers. Terms that need defining in the application: Design Matrix, Pseudoinverse of the design matrix, Variance inflation factors, Co-occurrence plot.

3. Several of the boxes output `code` type texts (for example "Design matrix" and "Sample table summary") I am not sure what the utility of this being in this format is. If this was something the user could copy and paste into R, for example, this design choice would make sense, but I don't believe that is the purpose.

4. The first argument of the R function is `sampleData`, however everywhere else in the application/documentation this is referred to as a `sample table`. This should be consistent.

5. There are several pieces for the user to control that could have some better defaults based on user inputs (for example the height of the plots, the size of the text, etc). Especially for a novice user, it would be great if the defaults for these values were reactive (updated based on the user input) and were *mostly* correct, allowing for tweaking only if absolutely necessary. In that case, you could hide these options in an "Advanced plot settings" tab in the sidebar, rather than having them visible when the user first opens the application. I think this would greatly improve the user experience.

Minor Comments

1. The "Rank" box outputs text using the textOuput() function, this means that you end up with a [1] prepending the text (which may be confusing to a novice user). Using renderUI() and uiOutput() instead would remove this [1].

2. If you remove the value from some of the numeric inputs, you end up with an error that is hard to parse ("An error has occurred. Check your logs or contact the app author for clarification."). You can check for whether the values needed for each value/plot are input using the validate() function and provide better error messages if not. (It looks like this is done for the formula input, just needs to be done for other inputs, for example, plot height, text size, etc.)

Is the rationale for developing the new software tool clearly explained?

Yes

Is the description of the software tool technically sound?

Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Partly

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Partly

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Biostatistics

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Author Response 07 Sep 2020

Charlotte Soneson, Friedrich Miescher Institute for Biomedical Research, Basel, Switzerland

Summary

This article describes an R package and corresponding shiny application, ExploreModelMatrix, that can be used to explore design matrices for linear models. I can see this having a lot of utility for helping partitioners understand their design for simple research questions or for pedagogical use in the classroom. While I see this being extremely useful, the current design falls somewhere in the middle, a bit basic for complex research questions and a bit too complex for those with little statistical background. My suggestions assume that the primary user will be in the latter group.

Thank you for your constructive comments. Please find point-by-point responses to each comment below.

Major Comments

The main take away seems to be from the "fitted values" boxes (both the plot and the table at the top of the application display this information). These seem to show the exact same content, so it feels repetitive to have both. The table is much more legible, but I can also see the utility of visualizing this. Perhaps these should be a single box with two tabs, one with the table and one with the plot.

This is a good idea, and also frees up more of the horizontal screen space for the plot. In the revised version of the package (v1.1.5, available on GitHub and in the devel branch of Bioconductor), these two representations are now shown in a single box with two tabs.

I find the content in these depictions of the fitted values confusing, the variable names here are referring to the beta coefficients from the model, not the variable values themselves. To

someone familiar with R / model output this may be obvious, but I'm not sure that is the case for the target user. Additionally, the use of : to indicate an interaction is not something I would expect novice users to know. If this is meant to help users interpret/calculate fitted values after they fit their models, perhaps the "pseudo" model output could be printed above the "fitted values" plot/table. This would explain where the values that are being plugged into each of these variable names come from. Additionally, this fitted values plot becomes quickly illegible, it would be great to have some simple defaults built in to expand the plot space if there are several inputs (see #5).

The main motivation for this representation is to simplify the interpretation of the beta coefficients and the generation of appropriate contrasts. The actual values of the predictor variables are given as row and column names, and when combined provide a "label" for each box in the plot. For example, in the example shown in Figure 1, to find the estimated genotype effect in the control group, one would take the fitted value for the "genotype B" control samples ((Intercept) + genotypeB) and subtract the fitted value for the "genotype A" control samples ((Intercept)). The result (genotypeB) is the coefficient (column) in the design matrix to test for significance. Thus, it is essential that what is shown in the figure/table are the coefficient names (as generated by R's model.matrix()), since they are directly referred back to the column names of the design matrix. It is, in a way, less important to understand exactly why the coefficients are named as they are - the important thing is to be able to extract the proper combination of them for testing. We have addressed the automatic space allocation for the panel depending on the number of predictors and levels - see comment below.

There are several terms used in the application that may not be familiar to those without statistical/mathematical backgrounds. It would be great to have definitions provided to explain the plots/terms. For example, you could have a question mark next to each term that will pop up an explanation when the user hovers. Terms that need defining in the application: Design Matrix, Pseudoinverse of the design matrix, Variance inflation factors, Co-occurrence plot.

The tour provided with the application (accessible via the question mark icon in the top-right corner) provides some of these descriptions and additional help for interpretation. In the revised version, we have expanded on these and also added links to external documentation. In addition, we have added a question mark to each panel. Clicking on this will open the corresponding step of the built-in tour.

Several of the boxes output `code` type texts (for example "Design matrix" and "Sample table summary") I am not sure what the utility of this being in this format is. If this was something the user could copy and paste into R, for example, this design choice would make sense, but I don't believe that is the purpose.

The original purpose of this formatting was to display the output as it would be shown in R, to make it easier for users to make the connection. For the sample table summary, it is also important to display the class of each column (to see, for example, if a column that displays as numbers is really a factor). In the revised version of the package, the user has the choice of whether to display the design matrix as a data table or as 'regular' R output.

The first argument of the R function is `sampleData`, however everywhere else in the application/documentation this is referred to as a `sample table`. This should be consistent.

We have updated the application to be more consistent by replacing 'sample table' by 'sample data table'. We have also updated the text in the manuscript for consistency.

There are several pieces for the user to control that could have some better defaults based on user inputs (for example the height of the plots, the size of the text, etc). Especially for a novice user, it would be great if the defaults for these values were reactive (updated based on the user input) and were mostly correct, allowing for tweaking only if absolutely necessary. In that case, you could hide these options in an "Advanced plot settings" tab in the sidebar, rather than having them visible when the user first opens the application. I think this would greatly improve the user experience.

In the revised package, we have implemented a reactive panel size for the fitted values plot, which changes with the total number of displayed 'rows' (number of panels x number of levels of the predictor shown on the y-axis). It is still possible to tweak if desired. We now also collapse all plot settings in the sidebar by default.

Minor Comments

The "Rank" box outputs text using the textOutput() function, this means that you end up with a [1] prepending the text (which may be confusing to a novice user). Using renderUI() and uiOutput() instead would remove this [1].

We have changed the renderPrint() in these statements to renderText(), which removes the [1].

If you remove the value from some of the numeric inputs, you end up with an error that is hard to parse ("An error has occurred. Check your logs or contact the app author for clarification."). You can check for whether the values needed for each value/plot are input using the validate() function and provide better error messages if not. (It looks like this is done for the formula input, just needs to be done for other inputs, for example, plot height, text size, etc.)

This has been fixed, and the app now displays more informative messages if numeric inputs are not properly specified.

Competing Interests: No competing interests were disclosed.

Reviewer Report 18 June 2020

https://doi.org/10.5256/f1000research.26680.r64392

Jean Fan
[1] Department of Biomedical Engineering, Johns Hopkins University, Baltimore, MD, USA
[2] Department of Computer Science, Johns Hopkins University, Baltimore, MD, USA

Executive Summary

Choosing between different model designs is a common first step in hypothesis testing. Here, Soneson *et al.* created an R/Bioconductor package called ExploreModelMatrix that provides an interactive Shiny interface for exploring such model designs. While the potential utility of such a package in a guided teaching setting is evident, its utility in a research setting is currently limited by its inability to accommodate larger, more complex designs common to real biological research. Most pressingly, it remains unclear how each of the explorer's modules can be used to guide a non-mathematical user to choose the appropriate model design.

Major comments:

1. Overall, it is unclear to me who is intended to be the target user for this package. The authors suggest that ExploreModelMatrix can be use in biological research or teaching, where many users will not be formal mathematicians.

   However, if I try to model using redundant variables, I get a warning "The design matrix is not full rank," which is a difficult message to interpret, particularly for non-mathematicians.

   Likewise, if the number of observations in the design matrix is the same as its rank, I get a warning "The residual degrees of freedom is 0. Values such as variances or dispersions can not be estimated from data with this design," which again is a difficult message to interpret, particularly for non-mathematicians.

   A more actionable jargon-free set of recommendations would be important for users.

2. In biological research, it is not uncommon to have dozens of patients and dozens of cell-types for multiple treatments across multiple time points for example. For designs with more than a few options per predictor, the current interface becomes quickly unusable. For example:

   ```
   celltype <- factor(sapply(1:10, function(x) rep(paste0('celltype', x), 30)))
   patient <- as.factor(sample(1:10, 300, replace=TRUE))
   levels(patient) <- paste0('patient', 1:10)
   names(celltype) <- names(patient) <- paste0('cell', 1:300)

   sampleData <- data.frame(patient, celltype)
   head(sampleData)
   ExploreModelMatrix(sampleData = sampleData,
                      designFormula = ~ patient + celltype)
   ```

The 'Fitted values', 'Pseudoinverse of design matrix', and 'Correlation plot', all become overlapping and illegible, rendering the interface unusable.

3. Testing two different design formula (one with and one without intercepts):

```
ExploreModelMatrix(sampleData = sampleData,
        designFormula = ~ genotype + treatment)
ExploreModelMatrix(sampleData = sampleData,
        designFormula = ~ 0 + genotype + treatment)
```

I was unable to achieve the author's stated goals of "extract[ing] the interpretation of each individual coefficient, and formulat[ing] desired linear contrasts" and ultimately deciding which design was best suited for my data. It is not clear how the 'variance inflation factors', 'Pseudoinverse of design matrix', 'Co-occurrence plot', and 'Correlation plot' should be used to inform my decision.

A video tutorial or walkthrough showing how this could be done would be useful.

Minor comments:

1. I was unable to install the package using the instructions provided:

```
> BiocManager::install("ExploreModelMatrix")
Bioconductor version 3.9 (BiocManager 1.30.10), R 3.6.0 (2019-04-26)
Installing package(s) 'ExploreModelMatrix'
Warning message:
package ‘ExploreModelMatrix’ is not available (for R version 3.6.0)
```

Instead, I had to use:

```
devtools::install_github('csoneson/ExploreModelMatrix')
```

2. The "Choose reference levels" dropdown panels order the options alphabetically rather by the given level ordering. The fitted values plot does use the correct order though.

3. All the functions and utilities that are available are not clear. For example, the "Flip coordinates" toggle was very useful and it took me awhile to find it. Again, video tutorial or walkthrough highlighting these features will be very helpful.

4. If the annotation names ("subjectdummy", "condition", "count", etc) is too long, the Fitted values visualization becomes illegible as the text overlaps each other without wrapping.

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
No

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
No

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Bioinformatics, Applied Statistics

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Author Response 07 Sep 2020

Charlotte Soneson, Friedrich Miescher Institute for Biomedical Research, Basel, Switzerland

Executive Summary

Choosing between different model designs is a common first step in hypothesis testing. Here, Soneson et al. created an R/Bioconductor package called ExploreModelMatrix that provides an interactive Shiny interface for exploring such model designs. While the potential utility of such a package in a guided teaching setting is evident, its utility in a research setting is currently limited by its inability to accommodate larger, more complex designs common to real biological research. Most pressingly, it remains unclear how each of the explorer's modules can be used to guide a non-mathematical user to choose the appropriate model design.

Thank you for your constructive comments. We provide point-by-point responses below. Generally, we would like to emphasize that the main purpose of the application is to help users interpret a given design, and to understand how changing the design influences the way you would perform a given comparison of interest, rather than choosing a design for a specific experiment (the latter needs to be guided by insights and expectations about the system at hand,

and possibly in consultation with a statistical collaborator).

Major comments:
Overall, it is unclear to me who is intended to be the target user for this package. The authors suggest that ExploreModelMatrix can be use in biological research or teaching, where many users will not be formal mathematicians.
However, if I try to model using redundant variables, I get a warning "The design matrix is not full rank," which is a difficult message to interpret, particularly for non-mathematicians.
Likewise, if the number of observations in the design matrix is the same as its rank, I get a warning "The residual degrees of freedom is 0. Values such as variances or dispersions can not be estimated from data with this design," which again is a difficult message to interpret, particularly for non-mathematicians.
A more actionable jargon-free set of recommendations would be important for users.

In the revised version, we have expanded on these messages and added links to external documentation. We have also expanded the built-in tour, which provides additional interpretation assistance for the different concepts.

In biological research, it is not uncommon to have dozens of patients and dozens of cell-types for multiple treatments across multiple time points for example. For designs with more than a few options per predictor, the current interface becomes quickly unusable. For example:
```
celltype <- factor(sapply(1:10, function(x) rep(paste0('celltype', x), 30)))
patient <- as.factor(sample(1:10, 300, replace=TRUE))
levels(patient) <- paste0('patient', 1:10)
names(celltype) <- names(patient) <- paste0('cell', 1:300)
sampleData <- data.frame(patient, celltype)
head(sampleData)
ExploreModelMatrix(sampleData = sampleData,
                   designFormula = ~ patient + celltype)
```
The 'Fitted values', 'Pseudoinverse of design matrix', and 'Correlation plot', all become overlapping and illegible, rendering the interface unusable.

It is true that with a large number of levels for each factor, the available screen space can be too small. There are several ways around this. First, the size of the displayed text as well as of the panels can be modified. In the revised package we have implemented a reactive panel height for the fitted values plot, which should avoid the need for the user to change it manually. Furthermore, by combining the plot and table representations of the fitted values in one panel, the full width of the application is used for the plot. Second, if the number of levels becomes very large, it may be better to use the non-interactive interface where the output can be written to a file of arbitrary dimensions. Third, in order to understand a given type of model, it is often possible to work with a reduced version, with fewer levels per predictor but the same underlying structure.

Testing two different design formula (one with and one without intercepts):
```
ExploreModelMatrix(sampleData = sampleData,
                   designFormula = ~ genotype + treatment)
ExploreModelMatrix(sampleData = sampleData,
                   designFormula = ~ 0 + genotype + treatment)
```
I was unable to achieve the author's stated goals of "extract[ing] the interpretation of each individual coefficient, and formulat[ing] desired linear contrasts" and ultimately deciding which design was best suited for my data. It is not clear how the 'variance inflation factors', 'Pseudoinverse of design matrix', 'Co-occurrence plot', and 'Correlation plot' should be used to inform my decision.
A video tutorial or walkthrough showing how this could be done would be useful.

The main purpose of the application is to help users interpret a given design, and to understand how changing the design influences the way you would perform a given comparison of interest, rather than choosing a design for a specific experiment (the latter needs to be guided by insights and expectations about the system at hand). In the example above, the fitted values panel will inform the user about, e.g., the contrast required to compare two of the treatment groups. Neither design is more 'suitable' for the data, but depending on which one is selected, the contrast needs to be adapted.

Minor comments:
I was unable to install the package using the instructions provided:
```
> BiocManager::install("ExploreModelMatrix")
Bioconductor version 3.9 (BiocManager 1.30.10), R 3.6.0 (2019-04-26)
Installing package(s) 'ExploreModelMatrix'
Warning message:
package ‘ExploreModelMatrix’ is not available (for R version 3.6.0)
```
Instead, I had to use:
```
devtools::install_github('csoneson/ExploreModelMatrix')
```

ExploreModelMatrix was added to Bioconductor in release 3.11 (the current release, since April 2020). Thus, in order to install it via BiocManager::install() you need to have the most recent Bioconductor release. We now mention this in the manuscript, and we have expanded a bit on the installation instructions in the README to clarify this point.

The "Choose reference levels" dropdown panels order the options alphabetically rather by the given level ordering. The fitted values plot does use the correct order though.

This was intentionally designed in this way - if the user wants to choose a new reference level, the current level ordering doesn't matter, and we reasoned that it's typically easier to find the desired

level in a list if the options are listed alphabetically.

All the functions and utilities that are available are not clear. For example, the "Flip coordinates" toggle was very useful and it took me awhile to find it. Again, video tutorial or walkthrough highlighting these features will be very helpful.

The application contains a walkthrough (accessible via the question mark in the top-right corner), pointing out useful features. Moreover, we have added a question mark button to each panel, which will open up the corresponding step in the tour. We have also moved the "Flip coordinates" toggle inside the panel with the fitted values plot for easier access.

If the annotation names ("subjectdummy", "condition", "count", etc) is too long, the Fitted values visualization becomes illegible as the text overlaps each other without wrapping.

This is difficult to manage automatically, since the actual width of the panel depends on the size of the browser window (as opposed to the height, which is set in pixels). It is best remedied by changing the font size of the displayed text in the side bar. We have intentionally kept the full coefficient name on a single line, to avoid misinterpretations and make it easier to match it back to the column names of the design matrix.

Competing Interests: No competing interests were disclosed.