SoftwareX 22 (2023) 101347

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

An interactive R-Shiny app for quickly visualizing a tidy, long dataset with multiple dimensions with an application in clinical trial simulations for platform trials

Elias Laurin Meyer [a], Constantin Kumaus [a], Michal Majka [b], Franz Koenig [a],*

[a] Center for Medical Statistics, Informatics, and Intelligent Systems, Medical University of Vienna, Austria
[b] Erste Group Bank AG, 1100 Vienna, Austria

A R T I C L E   I N F O

Article history:
Received 31 August 2022
Received in revised form 7 December 2022
Accepted 25 February 2023

Keywords:
R-Shiny
Data visualization
Tidy data
Clinical trial simulation

A B S T R A C T

Visualizing tidy, long data with multiple dimensions is a task often encountered when performing scientific computer simulations. In general, in such simulations the impact of many input variables on many outcome metrics is investigated. One area where such complex data structures containing the simulated data are encountered is clinical trial simulation. When visualizing the complex data, we are usually interested in the marginal effect of (one or several) input variables, which mostly renders existing visualization software useless or tedious to use. We have developed an R-Shiny app designed specifically for this task, which significantly speeds up the workflow. In this article, the app is described alongside two illustrative examples, one of which illustrates the use of the app for visualizing actual platform trial simulation data.

Code metadata

Current code version	v1.0.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-22-00267
Code Ocean compute capsule	Not applicable
Legal Code License	GPL3
Code versioning system used	git
Software code languages, tools, and services used	R, R-Shiny
Compilation requirements, operating environments & dependencies	R no compilation needed; Dependencies: shiny, DT, shinybusy, plotly, tidyverse, shinyBS, colourpicker, shinyWidgets, bslib, shinydashboard, scales, Cairo, shinyAce
Link to developer manual	https://github.com/el-meyer/sim_vis_shiny
Support email for questions	elias.meyer@meduniwien.ac.at

1. Motivation and significance

Kaizer et al. [1] define a *Scientific Computer Simulation* as *"the imitation of a behavior of a system, entity, phenomenon or process in the physical universe using limited mathematical concepts, symbols, and relations through the exercise or use of a scientific computer model"*. As such, simulations are used in many different fields to better understand systems and the interaction of assumptions, randomness and outcomes. One such area where simulations are used extensively is the planning and design of clinical trials [2–4]. Among the most important tasks when planning a clinical trial is the determination of the required sample size to achieve a certain a priori confidence that the trial will be successful. In the past, this could be done analytically (i.e. using closed formulae), e.g. when designing a randomized controlled clinical trial (RCT) comparing a control treatment to a new treatment using a binary endpoint and a χ^2 test, the power of the statistical test is a function of the design parameters sample sizes and significance level, but also the assumed effect sizes. However, in recent years particularly complex and flexible study designs have become increasingly popular [5], with the result that simulations have become necessary to fully understand the study design's behavior. The same is true when multiple endpoints are analyzed and different relationships between the endpoints should be taken into account. In particular, *platform trials* [6]

* Corresponding author.
E-mail address: franz.koenig@meduniwien.ac.at (Franz Koenig).

have gained a lot of momentum recently and many simulation studies have been published to investigate the impact of *assumptions* and *design parameters* on the *operating characteristics (OC)* of the study design [7–10]. For example, in platform trials, contrary to regular RCTs, *design parameters* might now include how many treatments will be investigated, how many patients will be recruited, whether or not control data will be shared, etc. and *assumptions* such as recruitment rate, availability of new treatments over time and treatment effects. In terms of *operating characteristics*, while for classical RCTs power and type 1 error might suffice, in platform trials we investigate further metrics such as per-treatment power, per-platform power, time until first treatment is successful, interim stopping probabilities and many more. For a more detailed definition of *assumptions*, *design parameters* and *operating characteristics*, see *Meyer et al.* [11] — for the purposes of this article it is sufficient to understand that the former two are input variables into the simulation program and the latter are output variables of the simulation program.

When presenting results it is crucial to choose a comprehensive, yet concise visual representation of the data, which can be manually extensive especially when many dimensions are investigated. We searched for available apps, programs and packages which could help achieve this task. We found a plethora of available software aimed at visualizing the results of clinical trials and unstructured data in general, unfortunately none of them was suited to address our particular task of visualizing the marginal effect of (one or several) input variables. In particular, we investigated EasyPlot [12], ggplotgui [13] and INTEREST [14], which all offer extensive features to apply different visualization dimensions to the data, but do not facilitate the visualization of marginal effects of (one or several) input variables. In some scenarios our datasets also exceeded the upload file size limit of the Shiny apps. We therefore created a new R-Shiny App designed to address the aforementioned shortcoming.

2. Software description

The app is divided into a navigation sidebar (see Fig. 1) and an output field. Within the navigation sidebar, several tabs can be found that guide the user step-by-step through the individual workflow of the app. The tabs 'Data Settings' and 'Default values' are designed to let the user set technically necessary settings, while the tabs 'Plot' and 'Scatterplot' deliver data plots along with visual settings. In the following, we will describe the functionality and assumptions of the app in more detail. For a more detailed guide on how to use the app, see Section 3.

2.1. Data structure

The app expects a tidy (i.e. each variable is a column, each observation is a row) [15], long (also sometimes referred to as "tall" or "narrow") dataset. By "long", we understand that the data will usually contain more observations than input/output variables and different sets (values) of input variables should be reflected in additional rows and not additional columns. As the program is designed particularly to help visualize simulation results, the app assumes the dataset contains columns with input parameters followed by columns with output parameters. If the assumption concerning the order of variables is violated, the app will not function properly. To be more precise, two different types of data structures are allowed, representing scenarios, in which the data "a" contains all individual simulation runs (see Fig. 2; could be referred to as "replication-level structure") or "b" contains only the summary measures across all individual simulation run of a particular set of simulation input parameters (see Fig. 3, could be referred to as "summary-level structure"). In

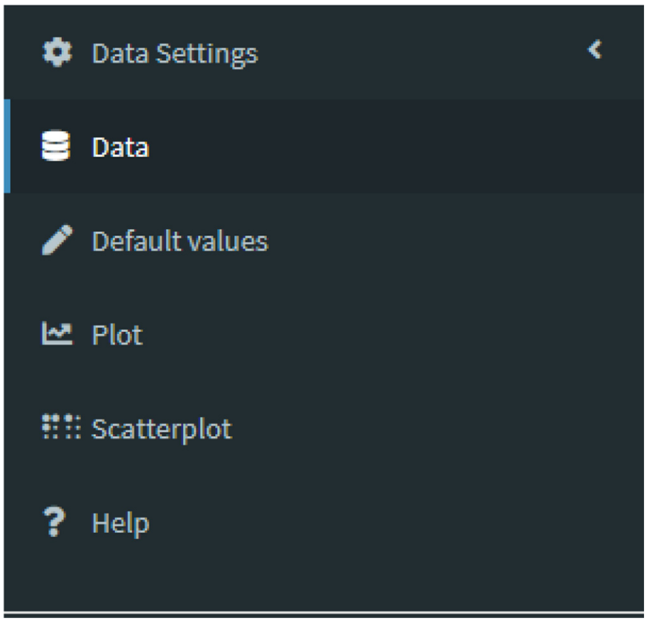


Fig. 1. Navigation sidebar of the R-Shiny app.

scenario "a", there is an additional column indicating the index of the replication for each set of simulation input variables. In this case, the app allows the user to summarize the data for every set of simulation input variables. The app features a built-in dataset of structure "a" for interested users to play around with without having to upload any dataset themselves. When uploading data, by default and in the server version of the app the size limit is set to 50MB, however this can be changed in the header of the app R script when run locally. Thus, when downloading the app and running it locally, there should be no practical limitation on the size of the dataset besides available RAM memory and ever increasing waiting times due to data processing.

2.2. Input parameter definition

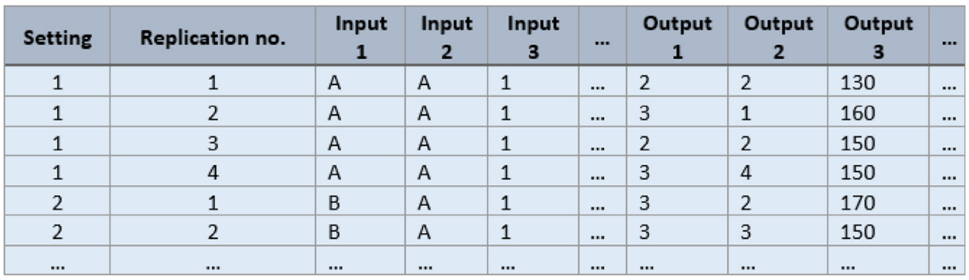
After the data upload, the user needs to state which of the columns contain the input parameters and which columns contain the output parameters. This is done via specifying the respective column which contains the last input variable such that following this column, all columns contain either only outcome variables or the replication index (see Section 2.3). This approach was chosen in favor of specifying for every column whether it represents an input or output variable (infeasible with large datasets).

2.3. Replications

In data structure "a" (as mentioned in Section 2.1), one of the columns contains a replication index corresponding to different sets of simulation input parameters. In this case, the user may tick a checkbox in the "data settings" tab, allowing them to chose mean or median as aggregation method.

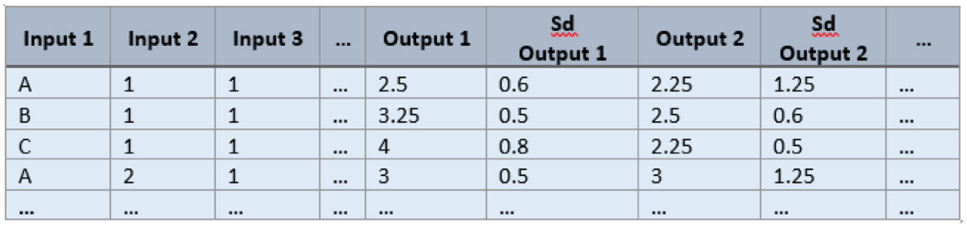
2.4. Default values

As mentioned in the introduction, the main issue when visualizing simulation results is to distill the marginal effect of one (or more) simulation input parameter(s), which requires filtering the dataset by setting a single value for all simulation input parameters not investigated in a particular plot. This is manually



Setting	Replication no.	Input 1	Input 2	Input 3	...	Output 1	Output 2	Output 3	...
1	1	A	A	1	...	2	2	130	...
1	2	A	A	1	...	3	1	160	...
1	3	A	A	1	...	2	2	150	...
1	4	A	A	1	...	3	4	150	...
2	1	B	A	1	...	3	2	170	...
2	2	B	A	1	...	3	3	150	...
...	...	...	...	...	...	...	...	...	...

Fig. 2. Data structure "a", containing all individual simulation runs.



Input 1	Input 2	Input 3	...	Output 1	Sd Output 1	Output 2	Sd Output 2	...
A	1	1	...	2.5	0.6	2.25	1.25	...
B	1	1	...	3.25	0.5	2.5	0.6	...
C	1	1	...	4	0.8	2.25	0.5	...
A	2	1	...	3	0.5	3	1.25	...
...	...	...	...	...	...	...	...	...

Fig. 3. Data structure "b", containing only aggregated data from every set of simulation input variables.

extensive, especially with a large dimension of simulation input parameters. A more convenient option is implemented in our app — the user sets default values for all simulation input parameters and if this dimension is not investigated in a plot produced later interactively, the dataset is automatically filtered according to the chosen default values. If a dimension is chosen to be shown in the plot (e.g. on the x-axis), no filtering is done. This allows fast, efficient exploration of the data. To reduce setup time, there is a button which – for every simulation input parameter – sets the first value to appear in the dataset to be the default value. So either the default values are set manually in the app (which can be burdensome in scenarios with many input parameters) or the dataset is ordered in a corresponding way before uploading it into the R-Shiny app.

2.5. Plot creation

Finally, the data can be plotted using the 'Plot' and 'Scatterplot' tabs. In data structure "a" (as mentioned in Section 2.1), the distribution of outcomes across replications can be visualized in the "Distribution" tab, which only appears if the respective checkbox in the "Data Settings" tab was ticked. The cornerstone of the app is the "Plot" tab, which in principle is a line ("spaghetti") plot, but can be extended to show the impact of many different simulation input dimensions, adjust labels, font sizes, colors, etc. To allow the involvement of several input parameters, one parameter is selected for the x-axis. Others can be selected to group different lines within a figure, but it is also possible to have a panel of figures where the rows and columns correspond to the values of selected input parameters ("small multiple"). In principle, an unlimited number of input variables can be visualized simultaneously (via facet wrap), but the maximum recommended number of input dimensions to be displayed simultaneously is four (x-axis, linetype, grid columns and rows) with the colors being allocated to as many output dimensions as sensible (depending also on their scale). Input variables containing characters are automatically ordered alphabetically and numerical input variables are ordered by size. If a specific ordering is required, the categories of the variables could be labeled e.g. "(a) Drug Y" and "(b) Drug X" in the uploaded dataset. Finally, the plot can be saved locally either via right-click or via a download button.

3. Illustrative examples

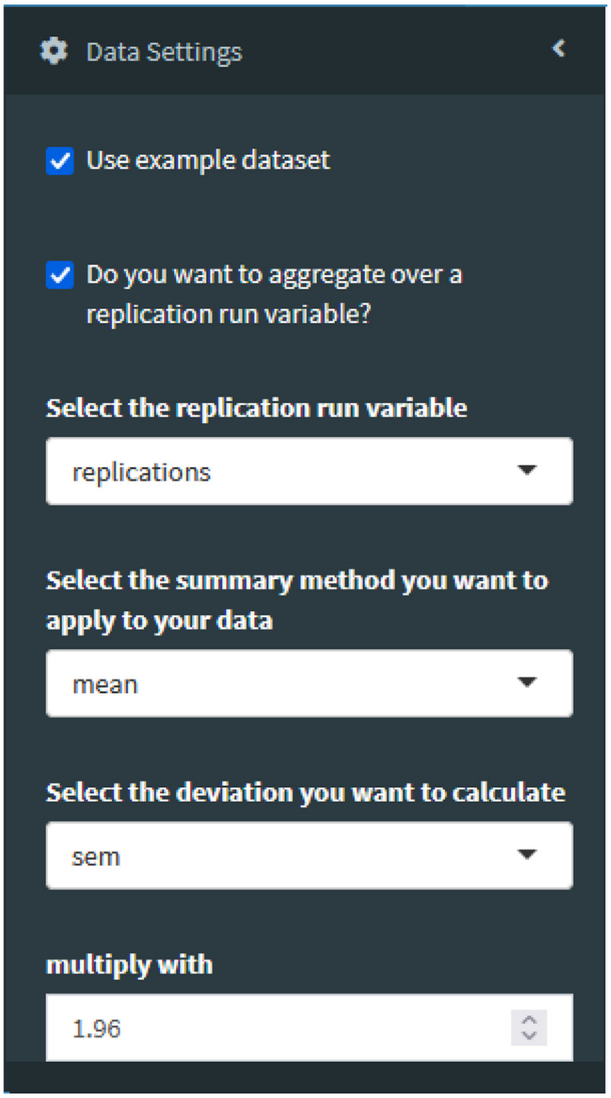
We present two illustrative examples to detail the usage of the app. The first example uses the built-in example dataset. The second example uses data from a clinical trial simulation for a complex platform trial investigating several drugs from a recent publication [8].

3.1. Example 1

This example uses the built-in example dataset, which is an artificially simulated dataset with four simulation input variables ("input1", "input2", "input3" and "input4"), whereby depending on the chosen input variables, we draw a sample from a bivariate normal distribution. The two output variables ("output1" and "output2") correspond to the x and y coordinate of the drawn data point. Furthermore, the dataset contains a replication variable, therefore represents data structure "a" (as mentioned in Section 2.1).

We load this dataset by clicking on the "Data Settings" tab and ticking the checkbox "Use example dataset". Next, as this dataset contains individual replications and is not yet aggregated, we tick the respective checkbox allowing us to choose an aggregation methods. By selecting the replication index variable the app lets the user choose methods for calculating the average estimates as well as a variability measure for all the different sets of simulation input variables. In order to illustrate some data manipulation features of the app, we decide among other settings in Fig. 4 to choose the Standard Error of the Mean (SEM) as the measure of variability, which furthermore allows for a multiplier to be passed to calculate the confidence intervals for any desired significance level.

Next, we switch to the "Default values" tab and need to set a reference value for every of the simulation input variables. We do this by simply clicking "Take first row as default values".


Data Settings

✓ Use example dataset

✓ Do you want to aggregate over a replication run variable?

Select the replication run variable: replications

Select the summary method you want to apply to your data: mean

Select the deviation you want to calculate: sem

multiply with: 1.96

Fig. 4. How to correctly specify the data settings for example 1.

Finally, we choose the “Plot” tab from the navigation bar and specify our plot according to the options in Fig. 5. As now for every point estimate there is a variable indicating a measure of variability we choose to present errorbars in the plot. Apart from showing both of the outcome variables with respect to “input2” on the x-axis we would also like to see the effect of “input1”. Ticking the checkbox “Do you want to add a Linetype dimension?”, allows specification of “input1” as a grouping variable. The resulting plot can be found in Fig. 6.

3.2. Example 2

This example uses a dataset published and described in a recent simulation study [8] (the dataset can also be found on Github). The structure of the dataset corresponds to data structure “b” (as mentioned in Section 2.1). To illustrate the functionality of the R Shiny app, we aim to recreate figure 4 of *Meyer et al.* [8] which shows the impact of several simulation input parameters simultaneously. In short, we investigate the impact of the sample sizes, type of control data sharing, maximum number of treatments investigated in the trial and speed of inclusion of new treatments into the trial on the per-treatment and per-platform power. For a detailed description of all simulation input and output variables and the underlying simulation mechanism, we refer to *Meyer et al.* [8].

After the initial data upload, a table appears giving a brief overview of the uploaded data (see Fig. 7). The first step is to set the last input variable as explained earlier. In this particular case, the last input variable is ‘Treatment.Efficacy.Setting’ and should be selected in the corresponding dropdown menu. No replication variable needs to be chosen.

Next, for every simulation input variable we have to set a default value. We choose the same default values as in the paper [8] (see Fig. 8).

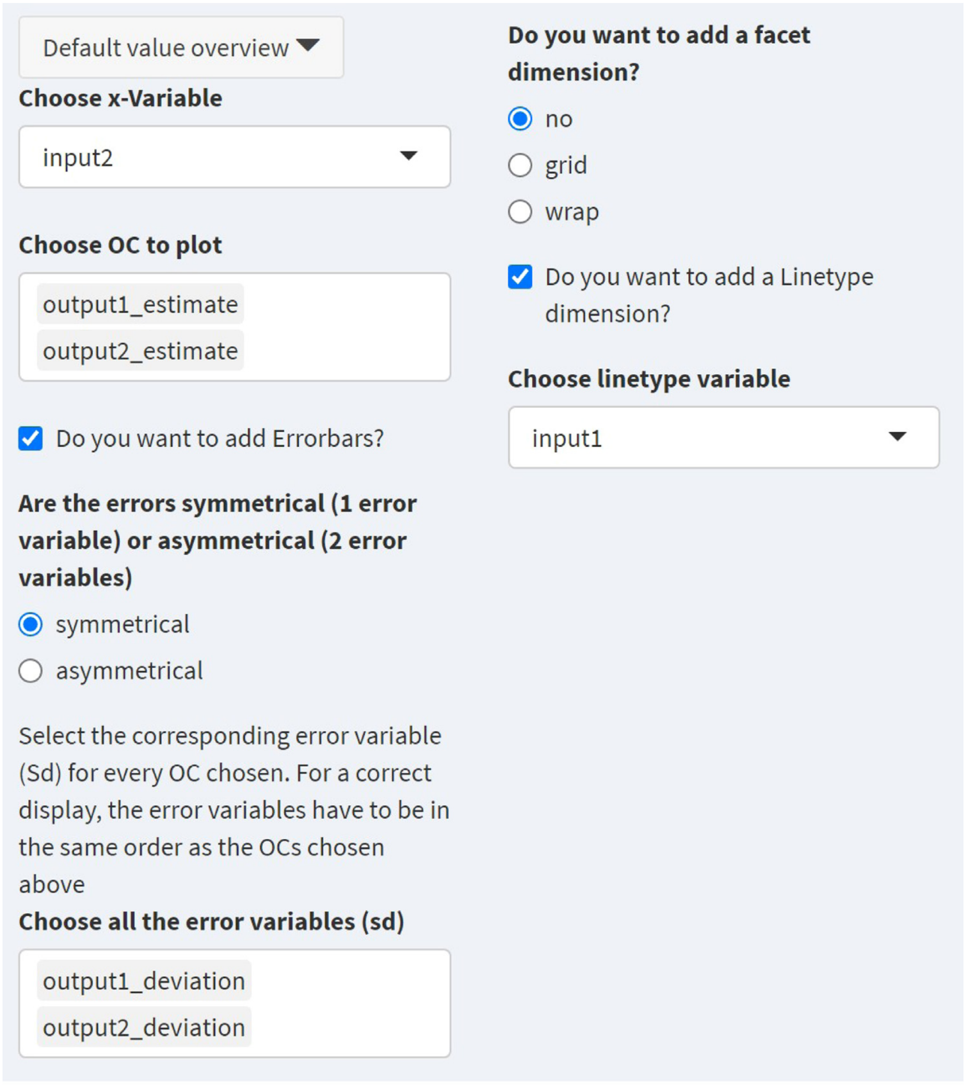
After clicking on the “Plot” tab, we set the visualization options as detailed in Fig. 9. Apart from the color, we have successfully recreated figure 4 from *Meyer et al.* [8]. To set the color, we use the draggable panel in the top right corner of the app. A color can be assigned to every output variable using either hex-codes or R-specific color names. Upon checking the box above the panel, the color choices are applied and the desired plot is achieved (see Fig. 10).

4. Impact

Our R Shiny app fills the gap of ready-to-use software with a graphical user interface that allows for visualization of longitudinal data with the special restriction that only marginal effects of certain columns should be shown and therefore the data needs to be reactively filtered (which is the case when repeated measurements are existent). The app is particularly useful when discussing different simulation results, e.g. evaluating different design options in clinical trial simulations with a clinical team in an interactive session. This will significantly speed up the workflow of investigating and summarizing data from e.g. simulation studies and preparing figures for scientific or internal use. While the app can be used for summarizing clinical trials, it can also be used by graduate students for summarizing data for master and doctoral theses, as well as in any scenario where simulations are required to better understand complex problems. More often than not, results of simulation studies are presented in tables, despite the advantages that visual presentations offer, such as being able to show more data points and relationships between different simulation dimensions simultaneously, which is a tremendous help in investigating and communicating the results. Our R Shiny app provides an easy-to-use, straightforward option to help summarize simulation results more efficiently.

5. Conclusions

In this paper, we presented an R Shiny app for the visualization of (repeated measures) longitudinal data and demonstrated that it can be used to create publication-ready figures. One area where such data structures arise is when simulation studies investigate the impact of different simulation input parameters. In such cases, it is usually desired to show the relative impact of one or a handful of particular simulation input variables on several outcome measures. We are not aware of any other free, open-source software solution based on R with this functionality. The development of the app is ongoing, with features such as different figure options, adding horizontal and vertical lines, animated plots, different supported data file systems and automatic creation of R code to be included in the future.


Default value overview

Choose x-Variable

input2

Choose OC to plot

output1_estimate
output2_estimate

☑ Do you want to add Errorbars?

Are the errors symmetrical (1 error variable) or asymmetrical (2 error variables)

[✓ symmetrical] asymmetrical

Select the corresponding error variable (Sd) for every OC chosen. For a correct display, the error variables have to be in the same order as the OCs chosen above

Choose all the error variables (sd)

output1_deviation
output2_deviation

Do you want to add a facet dimension?

[✓ no] grid wrap

☑ Do you want to add a Linetype dimension?

Choose linetype variable

input1

Fig. 5. Plot specification for example 1.

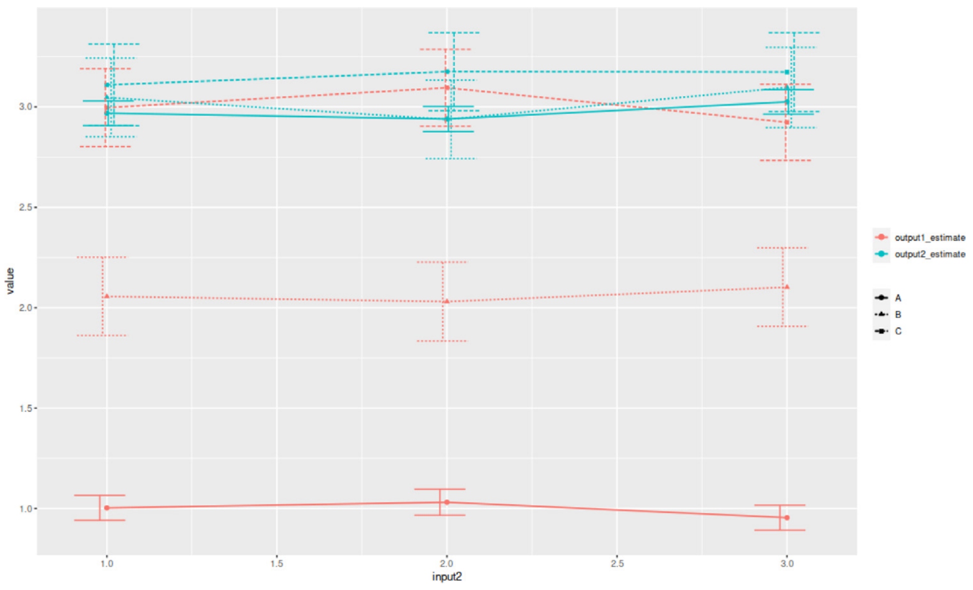
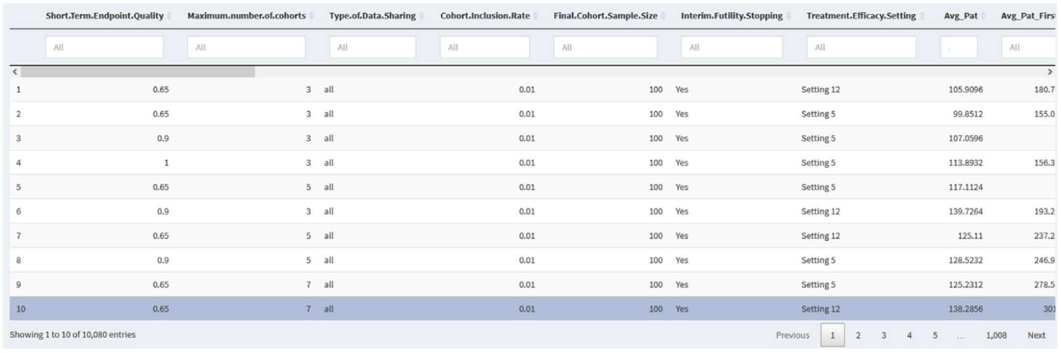


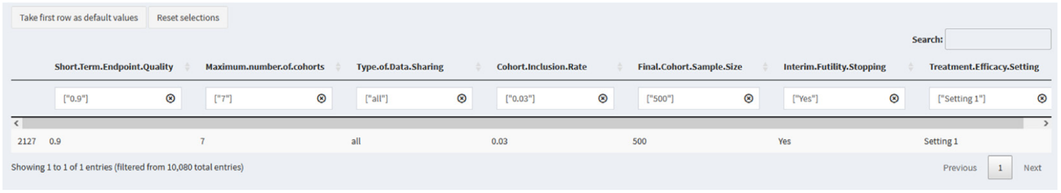
Fig. 6. Final plot example 1.



	Short.Term.Endpoint.Quality	Maximum.number.of.cohorts	Type.of.Data.Sharing	Cohort.Inclusion.Rate	Final.Cohort.Sample.Size	Interim.Futility.Stopping	Treatment.Efficacy.Setting	Avg_Pat	Avg_Pat_Firs
	All	All	All	All	All	All	All	.	All
1	0.65	3	all	0.01	100	Yes	Setting 12	105.9096	180.7
2	0.65	3	all	0.01	100	Yes	Setting 5	99.8512	155.0
3	0.9	3	all	0.01	100	Yes	Setting 5	107.0596	
4	1	3	all	0.01	100	Yes	Setting 5	113.8932	156.3
5	0.65	5	all	0.01	100	Yes	Setting 5	117.1124	
6	0.9	3	all	0.01	100	Yes	Setting 12	139.7264	193.2
7	0.65	5	all	0.01	100	Yes	Setting 12	125.11	237.2
8	0.9	5	all	0.01	100	Yes	Setting 5	128.5232	246.9
9	0.65	7	all	0.01	100	Yes	Setting 5	125.2312	278.5
10	0.65	7	all	0.01	100	Yes	Setting 12	138.2856	301

Showing 1 to 10 of 10,080 entries

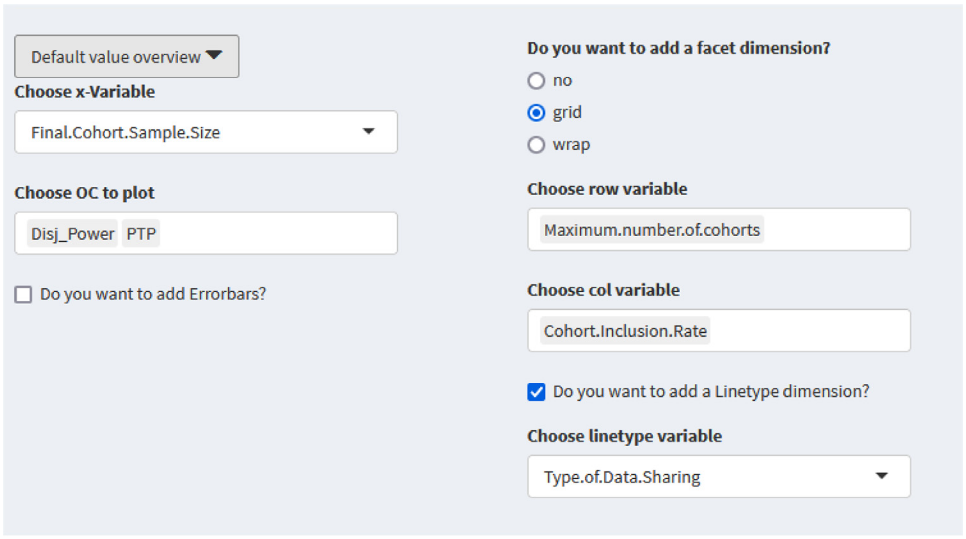
Fig. 7. Overview of user uploaded dataset in example 2.



	Short.Term.Endpoint.Quality	Maximum.number.of.cohorts	Type.of.Data.Sharing	Cohort.Inclusion.Rate	Final.Cohort.Sample.Size	Interim.Futility.Stopping	Treatment.Efficacy.Setting
	["0.9"]	["7"]	["all"]	["0.03"]	["500"]	["Yes"]	["Setting 1"]
2127	0.9	7	all	0.03	500	Yes	Setting 1

Showing 1 to 1 of 1 entries (filtered from 10,080 total entries)

Fig. 8. Default values in example 2.



Choose x-Variable: Final.Cohort.Sample.Size

Choose OC to plot: Disj_Power PTP

☐ Do you want to add Errorbars?

Do you want to add a facet dimension? no [✓ grid] wrap

Choose row variable: Maximum.number.of.cohorts

Choose col variable: Cohort.Inclusion.Rate

☑ Do you want to add a Linetype dimension?

Choose linetype variable: Type.of.Data.Sharing

Fig. 9. Plot options for example 2.

CRediT authorship contribution statement

Elias Laurin Meyer: Conceived the idea for the R Shiny App and the paper, Developed the first version of the R Shiny App, Wrote the final manuscript, Corrected and agreed on the final manuscript. **Constantin Kumaus:** Developed the final version of the app, Wrote the initial draft of the paper, Corrected and agreed on the final manuscript. **Michal Majka:** Supported the development of the app, Corrected and agreed on the final manuscript. **Franz Koenig:** Conceived the idea for the R Shiny App and the paper, Corrected and agreed on the final manuscript.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Funding

EU-PEARL (EU Patient-cEntric clinicAl tRial pLatforms) project has received funding from the Innovative Medicines Initiative (IMI) 2 Joint Undertaking (JU) under grant agreement No 853966. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation program and EFPIA and Children's Tumor Foundation, Global Alliance for TB Drug Development non-profit organization, Springworks Therapeutics Inc. This publication reflects the authors' views. Neither IMI nor the European Union, EFPIA, or any Associated Partners are responsible for any use that may be made of the information contained herein.

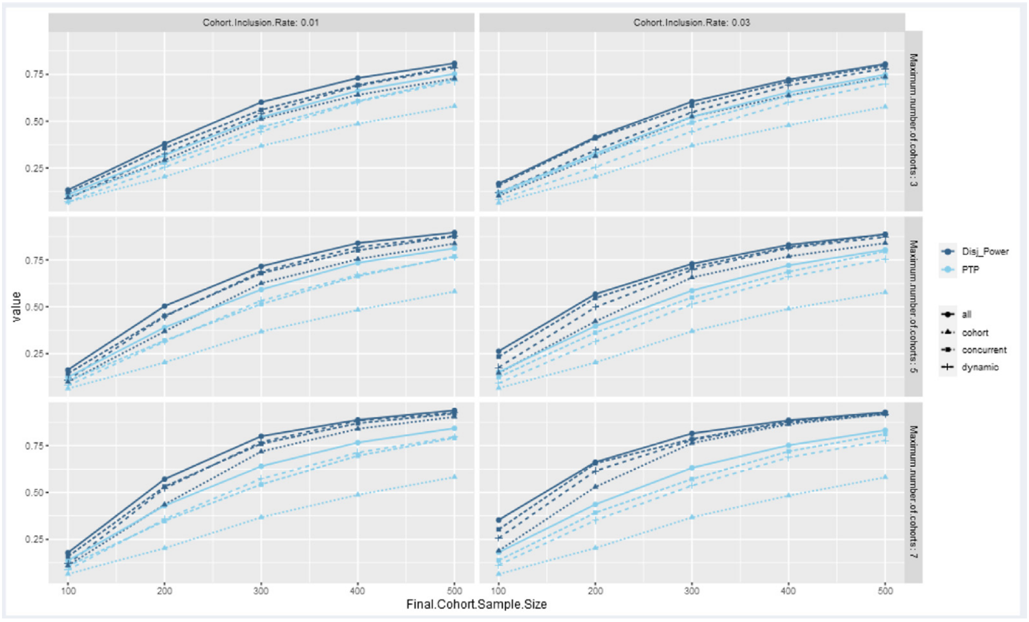


Fig. 10. Final plot example 2. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

References

[1] Kaizer JS, Heller AK, Oberkampf WL. Scientific computer simulation review. Reliab Eng Syst Saf 2015;138:210–8.

[2] Benda N, Branson M, Maurer W, Friede T. Aspects of modernizing drug development using clinical scenario planning and evaluation. Drug Inf J: DIJ/Drug Inf Assoc 2010;44(3):299–315.

[3] Bauer P, Bretz F, Dragalin V, König F, Wassmer G. Twenty-five years of confirmatory adaptive designs: opportunities and pitfalls. Stat Med 2016;35(3):325–47.

[4] Morris TP, White IR, Crowther MJ. Using simulation studies to evaluate statistical methods. Stat Med 2019;38(11):2074–102.

[5] Meyer EL, Mesenbrink P, Dunger-Baldauf C, Fülle H-J, Glimm E, Li Y, Posch M, König F. The evolution of master protocol clinical trial designs: A systematic literature review. Clin Ther 2020;42(7):1330–60. http://dx.doi.org/10.1016/j.clinthera.2020.05.010, URL http://www.sciencedirect.com/science/article/pii/S0149291820302447.

[6] Woodcock J, LaVange LM. Master protocols to study multiple therapies, multiple diseases, or both.. N Engl J Med 2017;377(1):62–70. http://dx.doi.org/10.1056/NEJMra1510062.

[7] Roig MB, Krotka P, Burman C-F, Glimm E, Gold SM, Hees K, Jacko P, Koenig F, Magirr D, Mesenbrink P, et al. On model-based time trend adjustments in platform trials with non-concurrent controls. BMC medical research methodology 2022;22(1):1–16.

[8] Meyer EL, Mesenbrink P, Dunger-Baldauf C, Glimm E, Li Y, König F. Decision rules for identifying combination therapies in open-entry, randomized controlled platform trials. Pharm Statist 2022;21(3):671–90.

[9] Zehetmayer S, Posch M, Koenig F. Online control of the false discovery rate in group-sequential platform trials. Statistical Methods in Medical Research 2022;31(12):2470–85.

[10] Saville BR, Berry SM. Efficiencies of platform clinical trials: A vision of the future. Clin Trials 2016;13(3):358–66. http://dx.doi.org/10.1177/1740774515626362.

[11] Meyer EL, Mesenbrink P, Mielke T, Parke T, Evans D, König F. Systematic review of available software for multi-arm multi-stage and platform clinical trial design. Trials 2021;22(1):1–14.

[12] Majka M. Easyplot. 2019, https://github.com/majkamichal/easyPlot.

[13] Stulp G. Ggplotgui. 2020, https://github.com/gertstulp/ggplotgui.

[14] Gasparini A, Morris TP, Crowther MJ. INTEREST: Interactive tool for exploring results from simulation studies. 2019, arXiv preprint arXiv:1909.03813.

[15] Wickham H. Tidy data. J Stat Softw 2014;59(10). http://dx.doi.org/10.18637/jss.v059.i10.