Illustrated example

With the GSDesigner app you can add interim analyses to your statistical design in 3 short steps. In this document we show how this can be done for the example from the main article.

Step 1: basic design

The first step only asks for the information that remains the same as for the fixed sample design, i.e. the standard design without any interim analyses.

In our example we wish to test if a treatment group performs better than the control group, which will be tested through a one-sided t-test. The significance level is the usual 95% and the desired power is 80%, so the type I and type II errors (a and b) are 5% and 20% respectively.

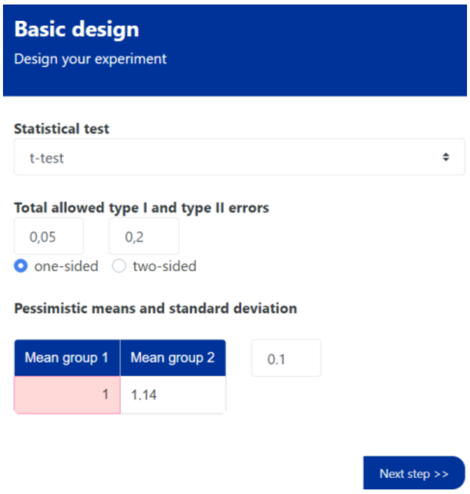


Figure 1: Basic design input for the toy example.

The control group is known to have a mean and standard deviation equal to 1 and 0.1 respectively. The standard deviations are assumed to be the same in both experimental groups. From similar experiments, we think it is likely that the treatment group can outperform the control group with a mean that is 20% higher. However, the minimal difference we wish to be able to detect is 14%, as this is the minimal relevant difference worth publishing. Since this is the minimal value we wish to have sufficient power for, this is the mean we enter at this stage.

Step 2: Interim analyses

Based on the information from the previous step, the app tells us that the minimum required sample size for the fixed sample design is 8 mice per group. For the group sequential design (GSD) we need at least as many, so that is what we will start with.

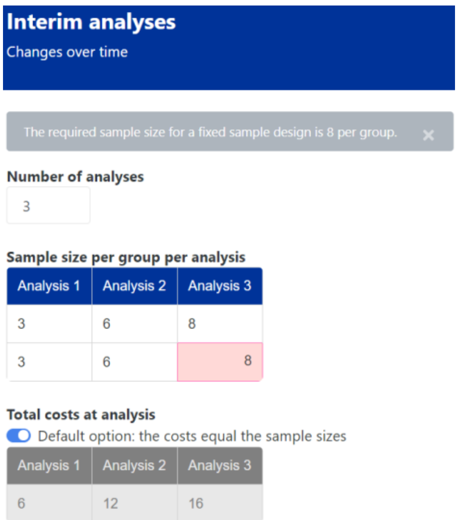


Figure 2: Interim analyses input for the toy example

The data collection process is very labor intensive and as a result only 6 mice can be processed per day, i.e. 3 mice per group. This means that the total data collection process will take 3 days. We choose to perform an interim analysis at the end of each day, since this is the most practical for us.

Since we are performing an in vivo experiment with live mice, we wish to minimize the number of animals used, i.e. the expected sample size. Hence the default option of using the sample sizes as the cost makes sense. However, if we were performing e.g. an in vitro experiment, we might be more concerned with the expected duration of the experiment, or the expected financial costs. In that case we could uncheck the default option and enter the total number of days or amount of money spent at each analysis.

Step 3: Error spending

In our last input step we define how the allowed errors are distributed over the interim analyses. In our case, the only thing we did when entering this page was check the boxes for the spending functions at the end of this page, since the default options are quite sensible for our design.

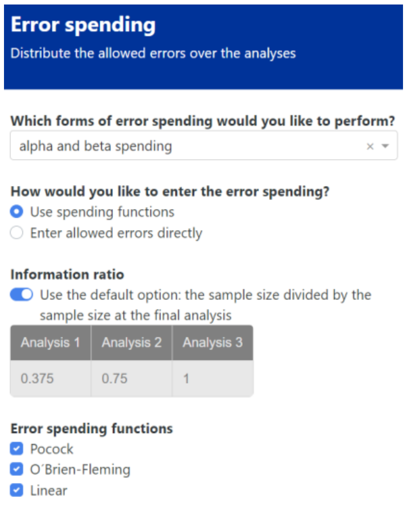


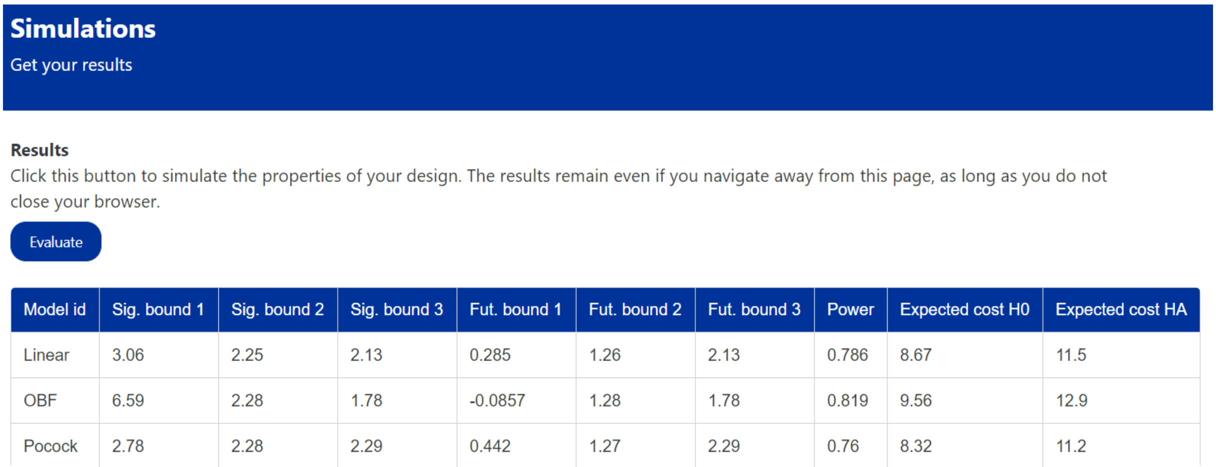
Figure 3: Error spending input for the toy example

We wish to stop if the results are significant (alpha spending), but also if the results are not promising enough (beta spending). The standard spending functions that are currently implemented are good enough for us, since these three functions represent three very different spending strategies. However, should we want to use custom spending functions, this is possible through entering the allowed errors per analysis directly.

Since we do not think it likely to lose a significant amount of data[1], the default definition of the information ratio is most sensible and we can just leave that as is.

Step 4: Results

The only thing left to do at this point, is to go to the last page, press 'Evaluate' and wait for the simulations to finish. If we want to, we can save the results either to a comma-separated values (csv) file or an excel file.



Model id	Sig. bound 1	Sig. bound 2	Sig. bound 3	Fut. bound 1	Fut. bound 2	Fut. bound 3	Power	Expected cost H0	Expected cost HA
Linear	3.06	2.25	2.13	0.285	1.26	2.13	0.786	8.67	11.5
OBF	6.59	2.28	1.78	-0.0857	1.28	1.78	0.819	9.56	12.9
Pocock	2.78	2.28	2.29	0.442	1.27	2.29	0.76	8.32	11.2

Figure 4: Toy example results

In this table we can see that the linear and Pocock spending functions do not have sufficient power. To compensate we return to step 2 and increase the sample size at the last analysis by 1 per group. If you want to make the effort, you can go to step 3 and uncheck the box for the OBF spending function, since we already have an OBF design with sufficient power. The benefit is that the simulations will be slightly faster than if all 3 spending functions are checked, but not really that much faster.

[1] For a document on efficiently handling data loss in group sequential designs see 'S3_Handling data loss' at github.com/ICDS-vubUZ/gsd-dash-app

Model id	Sig. bound 1	Sig. bound 2	Sig. bound 3	Fut. bound 1	Fut. bound 2	Fut. bound 3	Power	Expected cost H0	Expected cost HA
Linear	3.19	2.33	2.03	0.223	1.17	2.03	0.837	9.08	12.4
Pocock	2.87	2.33	2.18	0.389	1.21	2.18	0.818	8.64	12

Figure 5: results for the toy example with a larger sample size

As we can see, adding one extra mouse per group was enough to obtain the desired power level, so we can now compare the different designs and choose one of them. The statistical design

Additional comments

As you may have noticed, the results in the tables are rounded to 3 significant figures. You might want to get a more precise result, e.g. in case the obtained test statistic is close to the critical value. While the unrounded results are available in the csv and excel files, using them is not advised as the rounding is based on the simulation precision. If we want a more precise result, we should lower the relative tolerance level and run more simulations.

Getting one more significant figure means having a standard error that is 10 times smaller, which requires 100 times as many simulations. More simulations require more time, and in this case significantly more time. Currently, we can run our app for free on Heroku, for which we are grateful. The catch is that the amount of memory and CPU we are allowed to use is limited. As a result, it can be faster to run the app locally.