Appendix C: Power Calculation for Hybrid Experimental Designs

A User Guide for Employing Monte Carlo Simulations

R files are available to perform Monte Carlo simulations to estimate power for various hypotheses, under different scenarios. This code is suitable for the type of hybrid experimental design (HED) discussed in the paper (Figure 6) and for the models discussed in Appendix A. The files are available at https://github.com/dziakj1/Hybrid_Designs_Simulation. The R software is available at cran.r-project.org.

Monte Carlo Approach to Power Calculation

The main function is `simulate_power_hybrid()`, found in the file `simulate_power_hybrid.R`. This function contains a loop which repeatedly calls `generate_random_data_hybrid()` to generate random data under specified true parameters and sample size, and then calls `analyze_data_hybrid()` to estimate the parameters and test their statistical significance against the null hypothesis that they are zero. The proportion of times in which the null hypothesis is rejected becomes the Monte Carlo estimate of power given those parameters and sample size. These files are also accompanied by `run_the_simulations.R`, a wrapper which uses the three basic functions described above to construct the table of results presented in the paper.

Inputs to Main Function

The `simulate_power_hybrid` function accepts several arguments (input parameters), which are listed below. All of them are optional except for the first two (`n_sim` and `n_sub`). If not specified, the optional arguments are set by default to the values used in the simulation example described in this paper. The defaults are shown in parentheses in the list below.

- `n_sim`: number of datasets to simulate. The higher n_sim is, the longer the simulation will take to finish, but the more accurate the power estimates will be (assuming that the hypothesized

parameters are correct). A value of 2000 or 5000 seems reasonable for n_sim, but using 100 instead will give a quick approximation.

- `n_sub`: number of study participants per simulated dataset; this reflects the proposed sample size for which a power estimate is desired.
- `n_obs (112)`: number of JITAI decision points in which participants are randomized
- `K (28)`: ADI decision point in which non-responders get re-randomized
- `p_responder (0.5)`: proportion of responders
- `b0 (0.25)`: coefficient representing an intercept constant
- `b1 (-0.03)`: coefficient representing half the main effect of first-stage ADI options Z_1
- `b2 (-0.03)`: coefficient representing half the effect of second-stage ADI options Z_2 among non-responders
- `b3 (-0.03)`: coefficient representing the interaction between first-stage and second-stage ADI options
- `gamma0 (-0.02)`: coefficient representing half the proximal main effect of randomized JITAI options A
- `gamma1 (-0.02)`: coefficient representing interaction between A and Z_1
- `gamma2 (-0.02)`: coefficient representing interaction between A and Z_2
- `gamma3 (-0.02)`: coefficient representing interaction between A, Z_1 and Z_2
- `delta (-0.08)`: non-causal effect of response status on proximal outcome
- `sigma_sqd (0.20)`: error variance of proximal outcome
- `rho (0.50)`: autocorrelation between consecutive error terms in proximal outcomes
- `make_all_effects_null (FALSE)`: included for convenience to override defaults and set all beta and gamma parameters to zero. If `make_all_effects_null` is set to TRUE then the

function will estimate Type One error instead of power, and in that case any inputs for the beta or gamma parameters will be ignored.

- `print_results (TRUE)`: whether to print results on screen (vs. simply record them to a data object)

Use Examples of Main Function

For example, the function can be used in R as follows:

```
results <- simulate_power_hybrid(n_sim=200,n_sub=120)
```

Note that before running the function above, the required files (`generate_random_data_hybrid.R`, `analyze_data_hybrid.R`, and `simulate_power_hybrid.R`) need to be loaded using R's built-in `source()` function.

Output from Main Function

The output from the code above should look similar to the following:

```
               proximal distal

Z1                0.925  0.945

Z2stage2          0.450  0.440

Z1:Z2stage2       0.480  0.495

A                 0.960  0.100

Z1:A              0.865  0.085

Z2stage2:A        0.605  0.095

Z1:Z2stage2:A     0.570  0.095
```

The power estimates will not be exactly identical because of different random seeds. The output is also saved to an object called `results` in the workplace, which can be viewed using the code `print(results)`.

Assumptions for the Included Example

As a caveat, note that the code included makes some restrictive assumptions for the sake of simplicity. The input parameter `p_responder,` which reflects the proportion of responders in each arm, is assumed to be the same regardless of initial treatment. This is likely to be unrealistic but can be easily relaxed by changing the code to allow different `p_responder` values for each value of Z_1. The randomizations to JITAI options A are assumed to have 0.5 probability per occasion. The code can be changed to allow other set probabilities or to allow the probabilities to depend on Z_1 and/or Z_2. The proximal outcomes Y_1, Y_2, etc., are assumed to be normally distributed and homoskedastic, and the distal outcome is assumed to be the sum of the proximal outcomes. Also, re-randomization to second-stage ADI options is assumed to occur at a fixed time for all participants, and there are assumed to be no missing data. These assumptions would not necessarily be appropriate for planning an empirical study, but make for relatively simple and easily readable code for demonstrating the ideas introduced in this paper.