Lessons Learned Registering Simulation Studies

Lindsay Alley and Mairead Shaw
McGill University
Study Planning and Justification

• RR vs traditional simulation:
  • Stage 1 reviews!
  • Can’t make choices based on results
  • Injects rigour
• Justify your study conditions
• Describe your expected results
Justify Study Conditions

• What’s different?
  • Can’t use results
  • Locked in (mostly)
  • Reviewers can request changes at Stage 1, but not Stage 2!

• Justify using:
  • Your research questions
  • A previous simulation
  • A review of:
    • Common practice
    • Simulations
Describe Expected Results

<table>
<thead>
<tr>
<th>Research Question</th>
<th>Hypotheses</th>
<th>Analysis Plan</th>
<th>Interpretations</th>
</tr>
</thead>
<tbody>
<tr>
<td>How does sample size affect type 1 error rates?</td>
<td>Lower type 1 error rates with increasing sample size</td>
<td>ANOVA comparing Type 1 Error rates between sample size conditions</td>
<td>If we find support... If we don’t...</td>
</tr>
</tbody>
</table>
Describe Expected Results

• More results than you know what to do with!
• Which results will you present?
  • Consider decision rules in advance
• How will you organize results?
  • Example tables and plots
Reproducibility

• Is it reproducible?
• Is it easy to check reproducibility?
In a simulation study, you’re simulating your data, but don’t want to know actual results

Option 1: small number of fake datasets
  - Saves computing time and keeps results unstable

```r
  c_options <- c(10, 10, 10)
  n_options <- c(10, 10, 10)
  effect_size_options <- c("null", "small", "medium")
  ndat_options <- 5
  # ndat_options <- 1000
```

Option 2: simulate data that has the structure you want, but fake starting values

```r
  # TEST EFFECTS
  if (effect_size == "null") {
    effect <- 0.1
  } else if (effect_size == "small") {
    effect <- 0.1
  } else if (effect_size == "medium") {
    effect <- 0.1
  } else {
    stop("Error in effect size condition")
  }
```
Reproducibility: Package and Software

• List software and package versions in your paper
• `renv` automatically tracks your package versions and installs them on a reproducer’s machine with the call `renv::restore()`

https://rstudio.github.io/renv/articles/renv.html
Reproducibility: Control File

• Many manual changes = recipe for disaster
• Implementing some kind of “control file” can help manage files: one file that manages all conditions and runs the entire simulation, calling separate function-specific files
• More effort up front, way easier in the long run
# control.R

# Load dependencies
library(lme4)

# Load functions written in other files
source("R/generate_data.R")
source("R/run_models.R")
source("R/calculate_results.R")

# Specify variables you manipulate
c_options <- c(10, 20, 30)
n_options <- c(10, 20, 30)
effect_size_options <- c("null", "small", "medium")
ndat_options <- 1000

# Create population data
datasets <- generate_data(c_options, n_options, effect_size_options, ndat_options)

# Run models of interest
raw_values <- run_models(datasets)

# Calculate final results
results <- calculate_results(raw_values)
Reproducibility: Make Reproducers’ Lives Easier

• Implement ways for reviewers to check that they’re on the right track without running the entire simulation
• We’re often simulating hundreds or thousands of datasets; manually exporting some allows reviewers to check theirs match ours early on
• If something doesn’t reproduce, can help narrow down why
Reproducibility: Make Reproducers’ Lives Easier

# create datasets
datasets <- simulate_data(ndat = 1000)

# save out two datasets
write.csv(datasets[[1]], "tests/df1.csv")
write.csv(datasets[[2]], "tests/df2.csv")

---

Step 1.1. Check the generated "df1" dataset for reproducibility. The first five rows of data should be:

<table>
<thead>
<tr>
<th>X_id</th>
<th>y</th>
<th>x1</th>
<th>x2</th>
<th>z1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13.005108</td>
<td>0.005631125</td>
<td>0.6302638</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>6.831263</td>
<td>-0.638537285</td>
<td>0.6302638</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>18.820961</td>
<td>0.263297907</td>
<td>0.6302638</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>16.852893</td>
<td>-1.44011439</td>
<td>0.6302638</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>10.133860</td>
<td>0.230134755</td>
<td>0.6302638</td>
<td></td>
</tr>
</tbody>
</table>

Step 1.2. Check the second generated "df2" dataset. The first five rows of data should be:

<table>
<thead>
<tr>
<th>X_id</th>
<th>y</th>
<th>x1</th>
<th>z1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9.956949</td>
<td>1.2972483</td>
<td>0.09844718</td>
</tr>
<tr>
<td>2</td>
<td>12.188077</td>
<td>1.3864179</td>
<td>0.09844718</td>
</tr>
<tr>
<td>3</td>
<td>9.965660</td>
<td>-0.9633913</td>
<td>0.09844718</td>
</tr>
<tr>
<td>4</td>
<td>12.261863</td>
<td>-1.3337259</td>
<td>0.09844718</td>
</tr>
<tr>
<td>5</td>
<td>9.805620</td>
<td>-0.5786500</td>
<td>0.09844718</td>
</tr>
</tbody>
</table>
Reproducibility: Make Reproducers’ Lives Easier

- Clearly labelled files
- Create numbered sections or use line numbers
  - “Simulated data were generated from a standard normal distribution (data_generation.R, 2.2)”

```r
68  ```
```r
69  ```
```r
70  ```
```r
71  ```
```r
72  ```
Reproducibility: Odds and Ends

• Set seed

• Store your code publicly
  • OSF, GitHub
  • If you’re worried about releasing the code before running the study, you can make a read-only option
  • GitHub read-only mirror of repository: https://gitfront.io
Reproducibility: Odds and Ends

Multilevel Modelling Multiverse Simulation
Closing Thoughts

- A lot of these things apply generally as good practices. Preregistration just makes it unavoidable
- Reviewers are with you along the way, which can help catch your oversights
- Preregistration frontloads the work – you do more reading and planning up front, and once results are in you have identified contingent paths
- Future you is the one following the map, so make it as detailed as possible