Monte Carlo Simulation

Ben Kite and Terrance Jorgensen

KU CRMDA

2017 Stats Camp

Agenda

- Introduction to concept of Monte Carlo
- Examples in R
- Tips and tricks
- Break for lunch (12:00-1:30)
- Discussion of power analysis
- Examples in R
- Questions

About me

- Ben Kite
- CRMDA Assistant Researcher
- Quantitative Psychology
- Four years of simulation research experience

What Is a Monte Carlo Simulation?

- Anything that involves generating random data in a parameter space
- Simulating *data* from a known *parameter* space that we specify
- Read Johnson's (2013) "Monte Carlo Analysis in Academic Research" for history and applications
  - doi:10.1093/oxfordhb/9780199934874.013.0022

What Is a Monte Carlo Simulation?

- Consider a statistical procedure (e.g., a *t* test) that receives data and returns a result
  - i.e., parameter estimates, sample statistics
- Presumably there is a "true" population value that the estimate is supposed to represent
  - Does the procedure yield good estimates of the true parameters?
  - Is the sampling distribution of the estimates normal, symmetric, consistent, etc.?

What Is a Monte Carlo Simulation?

- Specify a population (i.e., a set of parameters), then draw random samples from it
  - A population is a data-generating process
- Apply the procedure to each sample
  - Save estimates, tests, *p*-values, etc.
- Evaluate the procedure
  - Compare stats to parameters, check distributions

Goals of a Monte Carlo Simulation

- Check that a procedure behaves as expected
  - Nominal Type I error rates, unbiased estimates…
- See how a procedure behaves when assumptions are violated
  - Inflated Type I error rates? Robust if minor? Effects of missing data? Effect of sample size?
- Compare 2 procedures
  - OLS v. WLS; LGCM v. MLM
- Power analysis

Replicating a Random Process

- We must be able to regenerate results exactly without saving each data set
- Pseudorandom number generator (PRNG)
  - Algorithm that generates seemingly random streams of integers
  - The “random” numbers you get depend on a random “state” characterized by a “seed”
    - Seed can typically be specified using a single integer
    - Setting the seed allows you to replicate results

Let’s Generate Random Numbers in R!

Let's open our R syntax and get started. Here is an outline of today's topics/tasks:

- Generate some simple (pseudo)random numbers
- Generate random samples of data using population parameters
- Design a small-scale Monte Carlo study
  - How are Type I errors affected by between-group differences in *N* and *SD*?

Advice for Monte Carlo Designs

DO NOT:

- Think of the Monte Carlo experiment as “One Giant Sequential Script” of commands
- Generate a massive block of data that needs to be saved and re-loaded every time you run a procedure on it

Rather, create a function for every action you need to take

- Data-generation and -manipulation functions
  - Generates data for one “run” of the simulation
  - Perturbs the data (impose missingness, etc.)
- A function that accepts 1 data set and analyzes it
- A function that combines the above steps
  - Runs one complete replication
- Functions to harvest estimates, summarize/plot results

Monte Carlo Designs

- Think of a random sample as a person/case
  - Multiple samples in each condition
- Between-subjects factors change the data-generating process
  - Parameters, distributions, missing data, scales
- Within-subjects factors analyze the same data using different methods
  - Estimation method, with/out covariates, *N*?

Monte Carlo Outcomes

- Sampling distributions of… anything!
- Consistency, efficiency, normality
- Bias in point and *SE* estimates
- (Root) mean-squared error
- Confidence Interval coverage rates
- Rejection rates (alpha, power)
- Convergence rates
- Model fit

Design your study to test hypotheses

Exploratory simulations can get out of hand

- Are all conditions necessary to test your hypotheses?
  - Consider how factors are expected to affect outcomes of interest, including interactions
- If exploring potential effects, try 2 levels of each variable (2^k) for pilot study
  - Detect interactions (use η^2/R^2, not p value)
  - Confound higher-order ones to reduce conditions
  - Add levels to detect nonlinear effects

Write down a recipe for your code

Writing syntax can be daunting, so start with plain language

- Ingredients
  - Characteristics of your population(s)
  - Manipulated factors, outcomes of interest
- Write down steps from beginning to end
  - Can start broad, move to specific
  - Ultimately, easier to translate to R, C, Fortran

Variance Reduction Techniques

Save time and computing power, as well as reduce the amount of noise in your results

- When is it necessary to draw new samples?
  - NOT for factors like sample size, different estimators, prior variance, competing models
  - Typically, ONLY when the population differs (e.g., normal/nonnormal data), or the factor reflects an aspect of design that changes characteristics of the data (e.g., number of response categories)

Variance Reduction Techniques

- Consider sample size, etc., to be within-sample (or within-replication) factors
  - Recycle same seeds, or better yet, perform all analyses/conditions on the data the one time is generated
  - Generate largest N, then take first N_j from sample
- Repeat this for # of replications, within each cell of between-replication design

Analysis Plan

- Carefully consider outcomes of interest
  - Have testable hypotheses/predictions
  - In each replication, save the output you intend to investigate, in a way that makes it easy to analyze
- Picture your analysis of results ahead of time
  - Perhaps make up data in a spreadsheet that mimics the format of your results
  - Could help your design

Useful Tools

- In R, the package `portableParallelSeeds` allows you to exercise great control of replicability using random seed-states
  - Developed by Paul Johnson
  - Run this syntax to install and find help files:

```
install.packages("portableParallelSeeds", repos =
   "http://rweb.quant.ku.edu/kran", type = "source")
```