Monte Carlo Simulations

USC JumpStart

Greg Faletto

Ph.D. Candidate, University of Southern California Marshall School of Business

Department of Data Sciences and Operations (Statistics Group)

June 29h, 2023

Outline

1. **What are simulation studies?**
2. How to do a simulation study
    1. Designing a setup
    2. Choosing an evaluation metric
    3. Presenting results
3. Example simulation study using the R simulator package
    1. Specifying our simulation study
    2. Carrying out the simulation study
    3. Presenting results
4. More advanced topics

What is a simulation study?

- Remember: the definition of a *statistic* is anything that we can calculate from a random data set.
- Imagine we have some kind of belief or expectation about a statistic: given a certain kind of data set, we expect a statistic to behave a certain way.
    - **Example 1**: we have n samples $\{Y_1, Y_2, \ldots, Y_n\}$. The true mean of these samples is some number μ. **Expectation:** the sample mean $\overline{Y} = \frac{1}{n}(Y_1 + Y_2 + \ldots + Y_n)$ should be a good estimator for μ, an the larger n gets, the closer $\overline{Y}$ should be to μ.
    - **Example 2**: we observe a random matrix $\boldsymbol{X}$ with n rows and p columns. **Expectation:** the $p \times p$ matrix $\boldsymbol{X}^\top \boldsymbol{X}$ is symmetric and positive semidefinite (all of its eigenvalues are nonnegative).
- (We'll return to these examples throughout the tutorial.)

What is a simulation study?

- In a *Monte Carlo simulation*, we:
  - Generate many random data sets.
  - Calculate a statistic on each of these data sets.
  - Evaluate whether our expectation seems to be met.
- Often called a *simulation study* in statistics journals, or a *(synthetic) data experiment* in machine learning proceedings

Why conduct a simulation study?

- Simulation studies are useful to **provide evidence that a statistic has a certain property.**
  - **For yourself:** if you have an idea for a new estimator, a good idea is to check first whether it works in practice!
  - **For consumers of your research:** if you develop your idea and write a paper or give a talk, simulation studies provide evidence that can convince other people that your method actually works.
- Simulation studies contrast with
  - **Intuition:** a sense that an idea ought to work, perhaps by comparison to existing ideas that are known to work
  - **Theory:** a mathematical proof that an idea works under certain assumptions (which may or may not be possible to check)
- Convincing research often uses all three of these forms of persuasion.

Why conduct a simulation study?

- Simulation studies also provide a way of **double-checking your work**.
  - What if your intuition is missing an important aspect of the problem you’re trying to solve?
  - What if your mathematical proof contains a mistake?
  - What if the code you wrote for your simulation study contains a bug that doesn’t throw an error but does result in the wrong statistic being calculated?
- **Realistically, we will never be 100% sure that our work is error-free.**
  - But the more forms of evidence we can provide that align with a consistent message, the more confident we (and consumers of our research) can be that our work is correct.

Outline

1. What are simulation studies?
2. **How to do a simulation study**
    1. Designing a setup
    2. Choosing an evaluation metric
    3. Presenting results
3. Example simulation study using the R simulator package
    1. Specifying our simulation study
    2. Carrying out the simulation study
    3. Presenting results
4. More advanced topics

Steps of a Simulation Study

- A simulation study consists of many iterations of the following procedure:
    1. We **generate a random data set** according to some specification.
    2. We **calculate one or more statistics** using the data.
    3. We **evaluate the statistic** according to one or more metrics.
- After all of the iterations, we **generate summaries of our findings** (plots, data tables, etc.) that we can use to present our results.
- Ideally, we also **preserve our work in some way and share it** so that others can check for themselves whether we made any mistakes.

Steps of a Simulation Study

- Coming up with a statistic to evaluate is often the goal of research (or comes up organically in the process of research—for example, you want to check some assumption you’re making).
- For the purposes of exploring simulation studies, today we will take for granted that you already have a statistic to evaluate (we will use the examples from before for demonstration purposes)
- We will focus on the other aspects of simulation studies:
    - Coming up with a setup for generating random data sets
    - Choosing an evaluation metric
    - Generating plots and tables of results

Outline

1. What are simulation studies?
2. How to do a simulation study
    1. Designing a setup
    2. Choosing an evaluation metric
    3. Presenting results
3. Example simulation study using the R simulator package
    1. Specifying our simulation study
    2. Carrying out the simulation study
    3. Presenting results
4. More advanced topics

Designing a Setup: Keep it Simple

- Try to think of the simplest possible setup that could make your point.
  - Try to avoid “unusual choices” that could make it look like your setup is “rigged” to work only in this specific setting, which makes your simulation study less convincing.
  - Unusual choices might also complicate your message and distract from the central point you are trying to communicate.
- **Example 1** (sample mean): On each simulation, we could generate n independent standard normally distributed observations. Maybe we could try a variety of choices for n and see if the estimator gets better as n gets larger.
- **Example 2** (positive semidefinite random matrix): On each simulation, we could generate an $n \times p$ matrix where each entry is an independent standard normally distributed random variable.

Let’s Write Some Code!

- Let’s write R code that generates data for each of these examples.

Designing a Setup: Include Some Variety

- Try to also vary your settings a bit, ideally in a way that you think might impact the results.
  - This serves the purpose of making your simulation study more convincing (both to you and consumers of your research)—it provides evidence that your method works in more than one specialized setting.
- **Example 1** (sample mean): Maybe consider distributions other than a normal distribution, like Poisson (heavier tails) or a discrete distribution.
- **Example 2** (positive semidefinite random matrix): Maybe try different ratios of n/p.

Designing a Setup: Can You Break Things?

- Try to find a setting where things break down
  - This helps you (and your audience) understand your message better.
- **Example 1** (sample mean): Maybe try dependent random variables.
- **Example 2** (positive semidefinite random matrix): Maybe try a discrete-valued distribution, so that two columns may be exactly equal (inducing linear dependence) with positive probability. Or try $p > n$ so that $\boldsymbol{X}^\top \boldsymbol{X}$ can't be full rank.
- If you are demonstrating your own statistic, make sure you also have settings that highlight its strengths!
  - Your audience may assume that you are showing the most favorable possible results for your statistic.

Outline

1. What are simulation studies?
2. How to do a simulation study
    1. Designing a setup
    2. Choosing an evaluation metric
    3. Presenting results
3. Example simulation study using the R simulator package
    1. Specifying our simulation study
    2. Carrying out the simulation study
    3. Presenting results
4. More advanced topics

Choosing an Evaluation Metric: Keep it Simple

- Similarly to your setup, try to choose a simple, conventional numeric metric to illustrate whether your expectation seems to hold. Try to avoid “unusual choices.”
- **Example 1** (sample mean): Our expectation was that the sample mean would do a good job of estimating the true mean.
  - The most traditional metric for estimation quality is probably squared error, $(\bar{Y} - \mu)^2$.
- **Example 2** (positive semidefinite random matrix): Our expectation was that the matrix $\boldsymbol{X}^\top \boldsymbol{X}$ would be symmetric and positive semidefinite.
  - On each simulation, we can just check if the matrix is symmetric. If yes, record a 1 for that simulation; if no, record a 0.
  - We could do the same thing to check for positive semi definiteness: calculate the smallest eigenvalue, and record whether or not it is nonnegative.
    - For more fine-grained information, we could consider recording the smallest eigenvalue itself on each simulation run.

Let’s Write Some Code!

- Let’s write R code that calculates each of these metrics.

Outline

1. What are simulation studies?
2. How to do a simulation study
    1. Designing a setup
    2. Choosing an evaluation metric
    3. Presenting results
3. Example simulation study using the R simulator package
    1. Specifying our simulation study
    2. Carrying out the simulation study
    3. Presenting results
4. More advanced topics

Presenting Results

- If possible, plots are always better than tables
  - Tables can also be good to convey results more precisely. Sometimes the appendix of your paper is the best place for these (or extra slides for your presentation that you will only pull up if someone asks a specific question)
- Try to convey both center and spread
  - Can present a boxplot of results
  - Can plot your mean and error bars

Outline

1. What are simulation studies?
2. How to do a simulation study
    1. Designing a setup
    2. Choosing an evaluation metric
    3. Presenting results
3. **Example simulation study using the R simulator package**
    1. Specifying our simulation study
    2. Carrying out the simulation study
    3. Presenting results
4. More advanced topics

The R `simulator` package

- We will walk through a simple example of a simulation study using the R `simulator` package (created by our department’s Prof. Jacob Bien).
- Install in R:

```
install.packages(“simulator”)
```

- We will also need the R `ggplot2` package for visualizations:

```
install.packages(“ggplot2”)
```

The R `simulator` package

- In the simulator package, we write functions for data **models**, **methods** (statistics), and **metrics**. Then the simulator runs a simulation study for us!
- Components
  - `model`: Specifies how data should be generated
    - `draw`: An individual data set created randomly according to the model
  - `method`: Specifies a statistic to calculate from each draw
    - `out`: A statistic calculated by a particular method on a particular draw
  - `metric`: Specifies a metric to evaluate each `out`

Outline

1. What are simulation studies?
2. How to do a simulation study
    1. Designing a setup
    2. Choosing an evaluation metric
    3. Presenting results
3. Example simulation study using the R simulator package
    1. Specifying our simulation study
    2. Carrying out the simulation study
    3. Presenting results
4. More advanced topics

Models

- In order to create a model, we first need to create a `simulate` function.
- The `simulate` function must have an argument `nsim`, which specifies the number of simulated data sets (`draws`) that it will create
  - It can also have arguments that parameterize the model
- It must return a list of length `nsim`.
  - Each list object is a `draw` from the model.

Models

Let’s use the sample mean code we wrote earlier to write a `simulate` function!

Models

- Once we’ve written a simulate function, we can create a model object using the function `new_model()`.
- `new_model()` has the following arguments:
    - `name`: A short, computer-readable name for the model (like a name you would use for a variable in code)
    - `label`: A longer, human-readable name for the model that can include spaces and symbols (like you would use for a plot you might present)
    - `params`: The parameters of the model
    - `simulate`: The `simulate` function we just wrote
- Finally, we need to create a `make_model` function. The `make_model` function takes as arguments the parameters of the model, calls the function `new_model()` internally, and returns the model object created by `new_model()`.

Models

Let’s use the `simulate` function we wrote to write a `make_model` function that calls the function `new_model()`!

Methods

- Remember: with `models`, our first step was to create a `simulate` function that specified how the data would be simulated.
- Similarly, with `methods`, our first step is to write a `method` function that specifies how our statistic is calculated.
- This `method` function must have the following arguments:
  - `model`: This is a list of the parameters of the `model` (that we specified earlier) that will be passed to the `method` function.
  - `draw`: This is an individual `draw` that will be passed to the `method` function.

Methods

- The output of a `method` function should be a named list.
- The list should include our calculated statistic, but it can include other things too if they might be useful later when we are calculating a `metric`.
  - For example, we could include intermediate or secondary calculations we computed when we calculated the statistic. Or we could include any other information from the draw that we might want to pass to the metric.

Methods

Let’s use the code we wrote earlier to write a `method` function!

Methods

- Next, we create a `method` object. We do this with the function `new_method()`.
- `new_method()` is a lot like `new_model()`. It has three arguments:
  - `name`: A short, computer-readable name for the method (like a name you would use for a variable in code)
  - `label`: A longer, human-readable name for the method that can include spaces and symbols (like you would use for a plot you might present)
  - `method`: The `method` function we just wrote

Methods

Let’s use the `method` function we wrote to create a `method` object using the function `new_method()`!

Metrics

- Once again, our first step will be to create a `metric` function that calculates the metric we are interested in.
- The `metric` function must have two inputs:
  - `model`: Again, this is a list of the parameters of the model that will be passed to the metric function.
  - `out`: This is the list created by a `method` object from a single draw.
  - Note that `draw` is not an input to the `metric` function. If we want to be able to access parts of the `draw` directly in the `metric`, we can write our `method` to pass them as part of `out`.
- The output of the `metric` function should be a single number.

Metrics

Let’s use the code we wrote earlier to write a `metric` function!

Metrics

- Similarly to before, we next create a `metric` object with the function `new_metric()`.
- `new_metric()` is a lot like `new_model()`. It has three arguments:
  - `name`: A short, computer-readable name for the metric
  - `label`: A longer, human-readable name for the metric
  - `metric`: The `metric` function we just wrote

Metrics

Let’s use the `metric` function we wrote to create a `metric` object using the function `new_metric()`!

Outline

1. What are simulation studies?
2. How to do a simulation study
    1. Designing a setup
    2. Choosing an evaluation metric
    3. Presenting results
3. **Example simulation study using the R simulator package**
    1. Specifying our simulation study
    2. **Carrying out the simulation study**
    3. Presenting results
4. More advanced topics

Carrying Out the Simulation Study

- Now that we’ve created our model, method, and metric objects, carrying out a simulation study is relatively straightforward.
- Basic steps:
    1. Create a `Simulation` object
    2. Execute our `make_model` object and write the object to file
    3. Simulate data sets (`draws`) from the model
    4. Calculate our statistics on each draw
    5. Calculate our `metric` on each statistic
    6. Save the results
- Let’s walk through one step at a time.

1. Create a `Simulation` Object

- We create a `Simulation` object using the function `new_simulation()`. It only requires two arguments:
  - `name`: A short, computer-readable name for the simulation
  - `label`: A longer, human-readable name for the simulation
- We will continue modifying and passing the simulation object to a series of functions in order to carry out the simulation study.

2. Generate a Model

- Next, we modify our `Simulation` object using the function `generate_model()`. This function executes our `make_model` object and writes the object to file.
- `generate_model()` requires two arguments:
  - The `Simulation` object we just created
  - `make_model`: Our `make_model` function we created earlier
- Lastly, the arguments of the `make_model` function should also be arguments of `generate_model()`. (We specify these parameters when we call `generate_model()`.)

3. Simulate From the Model

- Next, we simulate `draws` from the model using the function `simulate_from_model()`.
- `simulate_from_model()` requires two arguments:
  - The `Simulation` object that we just modified with `generate_model()`
  - `nsim`: The number of `draws` to simulate.

4. Calculate Statistics

- Next, we calculate statistics using the function `run_method()`.
- `run_method()` requires two arguments:
  - The `Simulation` object that we just modified with `simulate_from_model()`
  - `methods`: A list of `methods` that we want to apply to each `draw`.

5. Calculate Metrics

- Finally, we calculate metrics using the function `evaluate()`.
- `evaluate()` requires two arguments:
  - The `Simulation` object that we just modified with `run_method()`
  - `metrics`: A list of `metrics` that we want to apply to each `out`.

6. Save the results

- Lastly, we can save the results of the simulation using the function `save_simulation()`. The only argument is the `Simulation` object that we just modified with `evaluate()`.
  - Saving the simulation allows you to access the results later, if you want to generate more plots or tables, etc.
  - We can save the `Simulation` object at any point in the process. This is useful if you're worried that your computer could lose power or crash, etc.
  - We won't go over this today (see the documentation for details), but you can also easily add new methods, metrics, or simulations to a saved `Simulation` object rather than having to start from scratch if you want to make a change. (This is very useful because real simulation studies will typically be very computationally expensive, taking hours if not days to complete.)

Outline

1. What are simulation studies?
2. How to do a simulation study
    1. Designing a setup
    2. Choosing an evaluation metric
    3. Presenting results
3. **Example simulation study using the R simulator package**
    1. Specifying our simulation study
    2. Carrying out the simulation study
    3. **Presenting results**
4. More advanced topics

Generating Plots

- Generating plots is easy! Use the function `plot_eval()`. There are two required arguments.
  - The first argument is the simulation object (must have already run `evaluate()`).
  - `metric_name`: the name of the metric you want to plot.
- `plot_eval()` generates a `ggplot2` plot. There is a lot to learn about `ggplot2`, and I recommend learning about it because it’s a very powerful package for data visualization. But we won’t go over it today.

Generating Tables

- You can also generate a LaTeX table of results, which you can conveniently copy and paste directly into a LaTeX paper. Use the function `tabulate_eval()`, which has two required arguments:
  - The Simulation object
  - `metric_name`: the name of the metric you want to create a table for.
- Warning: the output won't print out properly if you use `tabulate_eval()` within a script that you source. Instead, call `tabulate_eval()` directly in the console after running a simulation (when the `Simulation` object is still stored in memory).

Creating a data.frame of results

- Lastly, you can also create a convenient data.frame containing the value of each metric for each draw.
  - Use the function `evals()`, which takes the `Simulation` object as an argument.
  - Pass the output of `evals()` to `as.data.frame()` in order to store the results as a data.frame
- This is useful for calculating summary statistics for the metric (mean, standard deviation, etc.), as well as doing statistical inference on the results.

Outline

1. What are simulation studies?
2. How to do a simulation study
    1. Designing a setup
    2. Choosing an evaluation metric
    3. Presenting results
3. Example simulation study using the R simulator package
    1. Specifying our simulation study
    2. Carrying out the simulation study
    3. Presenting results
4. **More advanced topics**

Varying model parameters

- We might want to see how the performance of a statistic varies as we change a model parameter.
  - For example, we expect the sample mean to better estimate the true mean as the sample size n increases.
- We can do this in the `generate_model()` function:
  - For the parameter we want to vary, provide a list of values: instead of `n=50`, type `n=list(50, 500)`
  - Add the optional argument `vary_along=“n”`
- Later, we can use `plot_eval_by()` to plot the metric values as function of the sample size
  - `plot_eval_by()` has the same arguments as `plot_eval()`, except it also has an added argument varying which you use to specify the parameter you want to vary. In this case, you'd type `varying=“n”`

Real Data Experiments

- Sometimes you may want to run a simulation study on a real data set
  - This provides evidence that your method works on real-world data sets, not just synthetic data that follows a nice distribution perfectly
- Typical setup: take a subsample of the data as a *training set* to calculate the statistic on, then use the rest of the data as a *test set* to evaluate the metric on
- My advice:
  - Using the `simulator` package, let each draw be a subset of the indices of the rows. These rows will form your training set, and the remaining rows will be a test set.
  - Let the method access the full data set, but only make use of the rows in the training set.
  - Pass the test set indices as part of `out` so that the metric evaluates only using the test set data.
- This approach avoids creating `nsim` copies of the data set, which wastes memory.

Documentation and Reproducibility

- These go hand-in-hand!
- Documentation: it’s a good idea to document your approach.
    - Fortunately, the simulator makes this easy—you can see exactly the settings that you used, the code you use to run everything, etc.
    - The “modularity” of the simulator (separate functions for models, methods, metrics, etc.) also helps facilitate documentation.
    - Also document: the version of R and the packages that you used to run your code; maybe even details of your computer, like the processor, number of cores, etc.

Documentation and Reproducibility

- Reproducibility: it’s always good to make your work reproducible
  - Set a seed when you run your simulation studies so that you (or whoever runs your code) get the same results every time
  - It’s okay to tweak your simulation study over time, but after it’s set in place, run it one more time from scratch. That way someone who runs your code later will get identical results to yours.
  - Share your code publicly so that others can see for themselves that the code does what you say it does

Other Topics

- Parallel processing: the `simulator` makes parallel processing easy.
  - This is really useful because each simulation is independent of the others—great setting to take advantage of parallelism for speed
  - See documentation for details
- Statistical inference: it's a good idea to do some sort of inference on your results
  - Example: t-test to evaluate whether different performance of two methods is statistically significant
  - You can easily do this once you've collected the data in a data.frame, as described earlier.

Thank you!

Additional Resources for the R `simulator` package

- Details on CRAN (official source for vetted R packages): https://cloud.r-project.org/web/packages/simulator/index.html
  - Includes documentation for the functions we talked about today
- Website for the simulator package: https://jacobbien.github.io/simulator/index.html
  - An example "vignette" walking through a simulation study: https://jacobbien.github.io/simulator/articles/lasso.html
- More advanced information in Prof. Bien's paper on the simulator: https://cloud.r-project.org/web/packages/simulator/index.html

Additional Resources

- Some other example simulation studies, both using the `simulator` package and not, for a recent paper Prof. Bien and I wrote: https://github.com/gregfaletto/presto
  - In this code, I make use of parallel processing, statistical inference, etc. I also do a real data experiment along the lines that I described earlier in the slides.
  - Also an example of reproducibility—if you run the code in the repo (using the same version of R and the packages), you will generate the exact plots that appear in the published version of the paper.
- A paper with more practical information on simulation studies: https://onlinelibrary.wiley.com/doi/pdfdirect/10.1002/sim.8086