Welcome to IMCS

Ashley I Naimi

Summer 2024

Contents

1 Welcome to Introduction to Monte Carlo Simulation

Welcome to IMCS!

In this short course, we will spend the next few days learning about the basics behind using Monte Carlo simulations in data science settings.

By the end of this course you should have a solid understanding of:

- What a Monte Carlo simulation is, and when / why you should do one.
- Example questions that can be answered via simulation studies.
- Key Distributions in R that you can use.
- Using DAGs to frame your simulated data.
- Defining and quantifying your estimand.
- Choosing and deploying performance measures (bias, mean squared error, etc).
- Presenting results using specialized plots (nested loop, zipper)

Simulation studies are preeminently useful in any quantitative science that uses statistics. Unfortunately, there is not a lot of literature on conducting Monte Carlo simulations in a wide range of settings, and it's challenging to try to put together a coherent understanding of this disparate work. This course seeks to help address these gaps.

2 Overview of the Course

Over the next few days, we will cover the following topics:

2.1 Section 1: Background and Context

- Why simulate? Example questions that can be asked and answered via simulation studies;
- Some example questions;
- An Overview of Simulation Designs;

2.2 Section 2: Basic Tools

- Key Distributions in the R stats and other packages
- Constructing user defined functions in R
- For Loops and the Apply Family of Functions

- Seeds in R
- Regression Models and Distributions for Simulation Studies

2.3 Section 3: Simulation Design

- The Aims of a simulation study
- Defining your data generating mechanism using DAGs
- What is your Estimand? No, really, What is your Estimand?;
- The true value and Monte Carlo Integration
- Evaluating Estimators

2.4 Section 4: Performance Measures

- Bias
- Mean squared error
- Confidence interval coverage and length
- Power

2.5 Section 5: Presenting Results

- Nested Loop Plots
- Zipper Plots

2.6 Section 6: Simulation in Practice

- Example 1: How does IP-weighting compare to marginal standardization?
- Example 2: Simple Regression in an RCT
- Example 3: Causal Mediation Estimators

2.7 Section 7: Computing Efficiency

- Computation: when do small differences become big?
- Benchmarking Tools
- Parallel Processing

3 Some Logistics

All the materials for this short course should have been provided to you via email. If you do not have them, please email info@statisticalhorizons.com.

4 R Code Logistics

In this short course, we will be using the R programming language throughout.[1] In order to run the code we will be using during the course of the workshop, it will be important to have a few packages pre-installed on your computer.

[1] For the beginner, I highly recommend using RStudio as your IDE of choice, particularly if you are less familiar with R and R programming. There are many excellent resources for installing and setting up R and the RStudio IDE. Here is a good getting started guide, provided by Garrett Grolemund: https://rstudio-education.github.io/hopr/starting.html

The easiest way to install these packages would be to paste the following code in the R console and run it:

```r
## first, if not installed, then
## install pacman
install.packages("pacman", repos = "http://lib.stat.cmu.edu/R/CRAN")
```

```
##
## The downloaded binary packages are in
##  /var/folders/zm/rqfqp5xs0fs86qs2mcxk6q0r0000gr/T//RtmpX22hI4/downloaded_packages
```

Once `pacman` is installed, you can use it to install and load the packages we'll need in the short course.

```r
pacman::p_load(here, skimr, tidyverse, parallel,
    lmtest, sandwich, MASS, gridExtra, broom,
    extraDistr, ExtDist, mvtnorm)
```

If the above code does not work for you, consider installing each package separately:

```r
install.packages("tidyverse", repos = "http://lib.stat.cmu.edu/R/CRAN")
library(tidyverse)
install.packages("here", repos = "http://lib.stat.cmu.edu/R/CRAN")
library(here)
```

```r
install.packages("sandwich", repos = "http://lib.stat.cmu.edu/R/CRAN")
library(sandwich)
...
install.packages("mvtnorm", repos = "http://lib.stat.cmu.edu/R/CRAN")
library(mvtnorm)
```

4.1 Installing Development Packages in R

Sometimes, there is a need to install packages that are not located on the CRAN repository. GitHub is a common location for such development packages. Installing development packages can be done with:

```r
install.packages("remotes", warning = F,
    message = F, eval = F)
library(remotes)

remotes::install_github("yqzhong7/AIPW")
library(AIPW)
```

However, using `install_github()` can lead to installation errors and problems. If you encounter these errors, the best recommendation I can give is to try using `devtools` instead of `remotes`:

```r
install.packages("devtools")
library(devtools)

devtools::install_github("yqzhong7/AIPW")
library(AIPW)
```

4.2 Manually Installing Dependencies

If this doesn't work, the next option is to look carefully through the log at the error messages. Often, these errors arise because a certain dependency could not be installed. It would help to try to install those dependencies first, and then try again. This could mean sometimes installing compilation libraries on your computer first. For example,

```
installation of package ‘igraph’ had non-zero exit status
```

After running a successful `install.packages` for the `igraph` package, you could try installing `AIPW` again.

4.3 Addressing GitHub API Limits

There are two strategies one can pursue to deal with this error. First, the error arises because a single call to `install_github()` is attempting to install numerous packages at once. Without an authenticated API, this could easily reach the limit of request calls.

The easiest way to address this issue is to use a Github personal access token (PAT). There are a number of ways to do this, and it's important to read the basic information on PATs. Within R and RStudio, one straightforward way to manage PATs is to install and use the `usethis` package, which has a suite of functions available for creating and integrating PATs. Once you've installed `usethis`, you can:

- Use `usethis::browse_github_pat()` to create a GitHub token
- Use `usethis::edit_r_environ()` and add the environment variable by adding the following line to the R environment file: `GITHUB_PAT = 'your_github_token'`.
- Restart R (so that the GITHUB_PAT is read) and try to reinstall the packages that were resulting in the API limit error.

Be aware: **your Github PAT is a password, and should be treated as such.**

Note that current recommendations regarding the use and inclusion of PATs into R differs from the procedure I outline above.[2] However, a number of students have had trouble implementing this procedure, while the former procedure works well.

[2] See, for example: `https://bit.ly/41yJKKK`

4.4 Addressing Other Issues

There are many other reasons why errors in installing development packages such as the `AIPW` can arise. Too many to cover here. The final route you can take, if it is available to you, is to contact the IT person at your school or work-place. These folks usually know how to resolve cryptic errors, especially those due to missing compilers.