Welcome to Analytic Methods for Health Disparities Research

Ashley I Naimi

September 2024

Contents

1 Welcome to Analytic Methods for Health Disparities Research 2

2 Overview of Course Topics 2

3 Some Logistics 3

4 R Code Logistics 4

4.1 Installing and Managing Packages 4

4.2 Installing Development Packages from GitHub (Optional) 5

4.3 Manually Installing Dependencies (Optional) 6

4.4 Addressing GitHub API Limits (Optional) 6

1 Welcome to Analytic Methods for Health Disparities Research

Welcome to AMHDR!

In this short course, we will learn some intermediate and advanced skills to do data science and data analytics, with a specific focus on health disparities research.

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

- how to use R and RStudio to effectively organize and share an analytic project
- some intermediate and advanced uses of regression modeling, particularly with the R programming language
- key concepts in mediation analysis, including causal mediation estimands, and regression based methods to estimate them
- how to combine skills learnt in R/RStudio, regression, and mediation to conduct cutting edge research in health disparities

The literature on ML and causal inference is very expansive and complex. There are some key essential concepts that are not easy to understand for those with little to no formal technical background. The purpose of this course is to help you with your efforts in filling this gap.

2 Overview of Course Topics

In this course, we will cover the following topics:

- Introduction to the Datasets
- Using R
  - Installing R/RStudio and Basic Functions
  - Using RStudio Projects and Writing Good Programs
  - Conditionally Adjusted Regression in R
    - Estimating Risk Differences, Risk Ratios and Odds Ratios
    - Estimating Appropriate Standard Errors
    - The Bootstrap
- Estimating Contrasts using Parametric Regression

  - Problems with Conditionally Adjusted Regression
  - Marginal Standardization via:
    * G Computation
    * Inverse Probability Weighting
- Mediation Analysis
  - Standard Mediation Methods
    * Product Method
    * Difference Method
  - Causal Mediation Analysis
    * Controlled Direct Effect
    * Direct and Indirect Effects
- Health Disparities Research
  - Counterfactual Disparity Measures and Relation to Mediation
  - Applied Example: Health Disparities in Pregnancy Outcomes
    * Explained by Nutrition
    * Explained by Physical Activity
  - Applied Example: TBD

3 Some Logistics

All the materials for this short course are available in the following GitHub Repository: `https://github.com/ainaimi/CDC_ShortCourse`

If you have git installed on your computer, you can download all the materials via the command line:

```
git clone https://github.com/ainaimi/CDC_ShortCourse.git
```

Or you can use a GUI of your choice (e.g., GitKraken) to clone the repo.

If you don't have `git` installed on your computer, you can simply download all the materials by visiting `https://github.com/ainaimi/CDC_ShortCourse.git`, clicking the "clone" button, and selecting "Download ZIP".

4 R Code Logistics

4.1 Installing and Managing Packages

In this short course, we will be using the R programming language throughout. In order to run the code we will be using during the course of the workshop, it will be important to know how to install R and Posit.

The easiest way to install these packages would be to paste the following code in the R console[1] and run it:

```r
install.packages("pacman", repos = "http://cran.us.r-project.org")

pacman::p_load(tidyverse, here, sandwich, lmtest,
               survey, boot, ggplot2, broom)
```

[1] For the beginner, I highly recommend using Posit (formerly RStudio) as your IDE of choice, particularly if you are not familiar with R and R programming. There are many excellent resources for installing and setting up R and the Posit IDE. Here is a good getting started guide, provided by Garrett Grolemund: https://rstudio-education.github.io/hopr/starting.html

The above code uses a package called `pacman`, which is installed on your computer using the first line of code.[2]

[2] See this link for some details on `pacman`.

Once `pacman` is installed, we next use the `p_load` function from the package to install and load a host of different packages into R.

If the above code does not work for you, consider installing and loading each package separately directly from CRAN. For example:

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

install.packages("here")
library(here)

install.packages("sandwich")
library(sandwich)

...

install.packages("broom")
library(broom)
```

4.2 Installing Development Packages from GitHub (Optional)

Note that we will not need to install development packages in this course, but I include this here for reference.

Sometimes, you may need to install packages that are not available on CRAN. These packages may be in development, and are typically hosted on external repository sites such as GitHub.

There are a few packages that can be used to install development packages easily. One of these is the `remotes` package. To install it, you can use the standard procedure:

```
install.packages("remotes")
library(remotes)
```

You can then use the `install_github()` function to install development packages directly from GitHub. For example:

```
remotes::install_github("tlverse/tlverse")
library(tmle3)
library(sl3)
```

Sometimes, installing development packages (i.e., 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`:

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

devtools::install_github("tlverse/tlverse")
library(tmle3)
library(sl3)
```

Note that, as of 2024-08-02, the `remotes` package is still available and being maintained. However, there may soon be a replacement package known as `pak`.[3]

[3] https://pak.r-lib.org/

4.3 Manually Installing Dependencies (Optional)

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 `tlverse` again.

Sometimes, a dependency may no longer be available on the CRAN repository, which can create challenges for installing things in a straightforward way. For example, the `sl3` package depends on the installation of the `imputeMissings` package, which (as of 2023-08-18) has been removed from CRAN.[4] At this point, the maintainers of the `sl3` package have to decide whether to remove `imputeMissings` as a dependency for their package. In the meantime, one solution to this problem is to install an archived version of the `imputeMissings` package. For example, one can run the following code[5]:

[4] See: https://bit.ly/46yYQ3R

[5] See: https://bit.ly/3GhJTIT

```
install.packages("https://cran.r-project.org/src/contrib/
                 Archive/imputeMissings/imputeMissings_0.0.3.tar.gz")
```

Which should install the dependency from the CRAN archive. Once `imputeMissings` is successfully installed, one should be able to proceed with the `sl3` installation as usual.

4.4 Addressing GitHub API Limits (Optional)

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. 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.

In the past, instructions were provided on how to add a PAT to your `.Renviron` file. However, this practice is no longer recommended.[6]

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

- If you don't already have one, the first step is to create a GitHub PAT (see "Get a personal access token" in the practical instructions of the link provided in the margins).

- Once you have a token, copy it to your clipboard. You can then call the `gitcreds::gitcreds_set()` function, which should return a list of options to your RStudio console:

```
-> Your current credentials for 'https://github.com':

  protocol: https
  host    : github.com
  username: ainaimi
  password: <-- hidden -->

-> What would you like to do?

1: Abort update with error, and keep the existing credentials
2: Replace these credentials
3: See the password / token
```

Select "Replace these credentials" and paste you PAT in the alloted location.

Note that **your Github PAT is a password, and should be treated as such.** Additionally, one can specify a time at which a particular PAT expires. If you're PAT is expired, simply follow the instructions here to renew them.