Welcome to EPI 560

Ashley I Naimi

January 2024

Contents

1 Welcome to EPI 560: Epidemiologic Methods IV

Welcome to EPI 560!

In this course, we will spend the next few months learning about the basic and intermediate concepts for analyzing epidemiologic data.

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

- the challenges of estimating causal effects with data
- basic and intermediate concepts behind probability and statistical inference
- regression modeling of epidemiologic data
- missing data: problems and solutions
- analyzing longitudinal data

2 Overview of Course Topics

In this course, we will cover the following topics:

- Introduction to the Datasets
- Causal Inference:
  - General Causal Inference
  - Potential Outcomes and the Fundamental Problem of Causal Inference
  - Estimands, Estimators, and Estimates
  - Effect Modification and CATEs
  - Identification Bias versus Estimation Bias
- Probability and Statistical Inference:
  - Frequentist and Bayesian
  - P-values, Neyman-Pearson Testing, and the P-Value Fallacy
  - S Values, Bayes Factors,
  - Confidence Intervals, Compatibility Intervals, Credible Intervals
  - Consequences of Heavy Tails
- Regression:
  - Fundamentals
  - Unadjusted, Conditionally Adjusted, and Marginally Adjusted Regression

  - Marginal Effects, Conditional Effects, and Noncollapsibility
  - Generalized Linear Models, Distributions and Link Functions
  - Marginally Adjusted Outcome Regression Models
  - Propensity Score Methods
  - Splines and Generalized Additive Models
  - Ridge Regression, LASSO, and Penalization
  - Quantile Regression

- Missing Data:

  - Relation to Exchangeability and Causal Inference Assumptions
  - Missing Completely at Random, Missing Not at Random, and Missing At Random
  - Monotone versus Nonmonotone Missingness
  - Imputation versus Weighting
  - Multiple Imputation via Chained Equations

- (Complex) Longitudinal Data:

  - Introduction to GEE and LMEMs
  - Causal Estimands for Time-Varying Data
  - Complex Longidutinal Data
  - IP-Weighting; g Computation
  - Structural Nested Models

3 Course Grading

The basis for the final grade will be determined as follows:

Section Assignments	60%
Final Assignment	40%

Section assignments will consist largely of short answer and data analysis questions to be submitted at the completion of each section in the lecture part of class.

Students **will be graded on only the 4 highest marked section assignments (out of 5 total),** making each of the four graded section assignments count for 15% of the overall grade.

Students may work in groups on the section assignments, however, **each student must submit their own independent section assignment.**

Additionally, these independently submitted assignments should reflect the student's own work (i.e., exact duplicate answers between students will not be accepted).

The final assignment will be a take-home "open-book" short answer and data-analysis project assigned during the final exam period. **The final assignment should reflect the student's independent work; group work will not be permitted.**

This final assignment will cover all of the material introduced in the course.

4 R Code Logistics

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 helpful to have a few packages pre-installed on your computer.

The easiest way to install these packages would be to paste the following code in the R console and run it. For the beginner, I highly recommend using 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 RStudio IDE. Here is a good getting started guide, provided by Garrett Grolemund.

If any of this is unclear to you, we will have opportunities to demonstrate some fundamentals with R and RStudio in the first few lab sessions.

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

```
##
## The downloaded binary packages are in
##  /var/folders/zm/rqfqp5xs0fs86qs2mcxk6q0r0000gr/T//Rtmp3CIbRf/downloaded_packages
```

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

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

```
install.packages("tidyverse", repos='http://cran.us.r-project.org')
library(tidyverse)
install.packages("here", repos='http://cran.us.r-project.org')
library(here)
install.packages("sandwich", repos='http://cran.us.r-project.org')
library(sandwich)
...
install.packages("broom", repos='http://cran.us.r-project.org')
library(broom)
```

4.1 Installing Development Packages Using `remotes`

We won't really need to install development packages for this course. However, if the need arises, it is good to know how to do this. There are a few options available to us. For example, you should first install the `remotes` package from CRAN:

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

Once installed, you can then proceed with installing your development package. For example, if the package is on GitHub:

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


library(AIPW)
```

Note that, as of 2023-12-23, the `remotes` package is still available and being maintained, but will be replaced by the `pak` package.[1]

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

Note also that, to install development packages from GitHub, you will need a GitHub account!

4.2 Manually Installing Dependencies

Sometimes, installing packages will not work, especially if they are development packages. There are a few strategies you could use to troubleshoot the

problems. First, try updating the packages you already have installed on your computer:

```
update.packages(ask = FALSE)
```

If you are not in the habit of regularly running the `update.packages()` function, your first time running it may take a while, so plan accordingly.

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 (available on GitHub) depends on the installation of the `imputeMissings` package, which (as of 2023-08-18) has been removed from CRAN.[2] 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[3]:

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

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

This issue is specific to `sl3` and `imputeMissings`, but the general strategy is often useful.

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. 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.[4]

[4] 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.