Statistical Computing
Midterm 2017

Instructions

We are going to make an S4 Package named `easyRasch`. It will require making multiple new generics and methods (specified below).

Some notes:

- I am expecting a complete gitHub repository containing the package and a development file. The development file should walk through the entire process of creating and documenting the package and include some example code showing how each function works. That is, you should provide some examples that make use of the functionality in the package.
- You have until the beginning of the next class. I do not expect that this will take anyone more than six hours total, but you are allowed to space that six hours over any 8-hour period. This allows you to go eat or whatever. But if you are going to work the entire time, please limit yourself to six hours.
- We will begin the time from the moment you start your GitHub Repostory.
- You will be graded on:
  - Comments
  - Correct/frequent use of GitHub with lots of commits.
  - Elegance of code (e.g., apply rather than loops, speed of functionality, etc.)
  - Readability of the code/stability of naming conventions
  - Documentation/full completion of package structure
- Email the TA with your repository when done. Include a start/end time in that email. Note any breaks.
- If you are totally lost and confused, email me. But be aware that I am not always be available. You can text me at (919) 559-6255 if you are desparate.

Introducing the problem

In this exam, you will be working with a one parameter item response model, often called the Rasch model. Think of it like an exam. For person j answering exam question i, we assume their probability of answering the question correctly is:

$$P_{ij} \equiv P(y_{ij} = 1|\theta_j, a_i) = \frac{\exp(\theta_j - a_i)}{1 + \exp(\theta_j - a_i)}, \tag{1}$$

where θ_j is the "ability" of the respondent (e.g., how good they are at math) and a_i is the "difficulty" of the problem.

Your goal in this exam is to develop a package to estimates the θ_j parameter for a respondent. We are going to assume that we *know* all n of the difficulty parameters $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ and we have observed their answers to *all* of their answers to the test, $\mathbf{y}_j = (y_{1j}, y_{2j}, \ldots, y_{nj})$ (we are assuming there is no missing data).

Details

Rasch class

You need to define an S4 class that represents a student who has completed an exam. The class should have room for the following slots:

- The name of the test taker
- A vector of question-item parameters $\mathbf{a}$
- A vector of answers for the respondent $\mathbf{y}_j$

(Extra credit) As part of this class, you will need to set up a validation method to test that it is structured correctly.

Probability

You will make a probability function that corresponds with P_{ij} in Equation (1). For convenience, we will also define $Q_{ij} = 1 - P_{ij}$, which is the probability they got it wrong.

The function should take in the following inputs:

- `raschObj`: An object of class Rasch (see above)
- `theta`: A proposed value of θ_j

The function should create the following outputs:

- A vector of length n that represents P_{ij} for each question.
- A vector of length n (called PQ) that represents P_{ij} IF they got the question correct and Q_{ij} IF they got the question wrong. That is, $y_{ij} = 1$ then $PQ_{ij} = P_{ij}$. If $y_{ij} = 0$, then it should be $PQ_{ij} = Q_{ij}$

Likelihood

We are going to calculate what is called the "likelihood" of a proposed value of θ_j given the responses we have observed from the test taker.

$$L(\theta_j|\mathbf{y}_j) = \prod_{i=1}^{n} \left(P_{ij}^{y_{ij}} Q_{ij}^{(1-y_{ij})} \right) \tag{2}$$

Basically, when they got the answer correct we say that the probability of seeing that response (given θ_j) is P_{ij}. The probability they got it wrong is Q_{ij}. The likelihood is just the joint probability of all of the observed responses given our guess of θ_j. To make this perfectly obvious, the likelihood is just the product of the vector PQ_j described above.

The function should take in the following inputs:

- `raschObj`: An object of class Rasch (see above)
- `theta`: A proposed value of θ_j

The function should create the following outputs:

- The calculated likelihood from Equation (2)

Prior

One problem with this model is that sometimes finding the value of θ_j that maximizes the likelihood (the normal goal in this branch of statistics) is not defined. Imagine that we have a test taker that got all of the questions right. In that case, the likelihood will be maximized when $\theta_j = \infty$. So we are going to be working in a Bayesian framework, which will often give a more reasonable estimate.

To do this, we need a prior. I'm going to keep this simple, and a assume that the prior is *always* going to be a normal distribution with $\mu = 0$, and $\sigma = 3$. You are going to write a function that calculates the height of the normal curve (using `dnorm`) for a proposed value of θ.

The function should take in the following inputs:

- `theta`: A proposed value of θ_j

The function should create the following outputs:

- The height of the specified normal curve evaluated at θ_j

To be perfectly clear, the function should return the output from *dnorm* where $x = \theta$, the mean is zero, and the standard deviation is 3.

EAP

We can calculate the expected a posteriori value for θ_j (our estimate of θ_j) using the following formula. Let $\pi(\theta_j)$ represent the prior function.

$$\hat{\theta}_j = \frac{\int_{-\infty}^{\infty} \theta L(\theta_j|\mathbf{y}_j)\pi(\theta_j)d\theta_j}{\int_{-\infty}^{\infty} L(\theta_j|\mathbf{y}_j)\pi(\theta_j)d\theta_j} \tag{3}$$

HINT: If we let $f(\theta_j) = L(\theta_j|\mathbf{y}_j) \times \pi(\theta_j)$ then the denominator could be written as

$$\int_{-\infty}^{\infty} f(\theta_j)d\theta_j$$

Likewise, we could make a function $g(\theta_j) = \theta L(\theta_j|\mathbf{y}_j)\pi(\theta_j)$ and approximate the integral of that.

We are going to approximate this integral using the `integrate` function in `R`. Note that `integrate` takes in a function as one its arguments (just like `apply`). So you will want to make a function for the numerator and the denominator and pass them into the `integrate` function. Not that, just like apply, you can pass in arguments to the function you are integrating. So, you will be integrating over θ but you should be able to pass in a `raschObj` as well.

The function should take in the following inputs:

- `raschObj`: An object of class Rasch (see above)
- `lower`: The lower limits of integration (default this to -6)
- `upper`: The lower limits of integration (default this to 6)

The function should create the following outputs:

- The value of $\hat{\theta}_j$ as specified in Equation (3).

Print

Make a `print` method for the `raschObj` that takes in a `raschObj`, calculates the EAP, and prints name of the test taker and the EAP results.

Notes for the confused

These functions will be easiest to create in a hierarchy. That is, higher level functions will be calling lower level functions. This will be a lot easier if you avoid coding the same functionality more than once.

- The likelihood function should call the probability function.
- The EAP should will at some point call both likelihood and prior
- The print method will call the EAP method.

Notes for the confused 2

If I have a function like this:

$$f(\theta_j) = L(\theta_j|\mathbf{y}_j) \times \pi(\theta_j)$$

This means that I should put in a potential value for θ and get back a number. In this case, $f(\theta)$ will be the result of multiplying two functions together.

1. I will put in a value of θ_j into the likelihood function (along with the `raschObject`, which contains the person's answers to the test). I will get back a number.

2. I will put in a value of θ_j into the prior function and get back a number.

3. I will multiply the numbers I got back from steps (1) and (2) and that will the output of $f(\theta)$.

4. Just like the Trapezoid integration method you worked on in the problem set, the `integrate` function works by evaluating the function $f(\theta)$ for different values of θ and doing some calculations to guess the area under the curve. So if you make a function called `myFunction` that takes in an argument θ and γ and I want to integrate over θ but hold $\gamma = 2$, I would write:

   ```
   integrate(myFunction, gamma=2, lower=6, upper=6)
   ```