Applied Statistical Programming - Exam 2022

4/6/2022

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

Rules of the Road

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 class on Wednesday, April 13. 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 regular 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

Practicalities:

- 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, drop me a note on slack. But be aware that I am not always be available especially since I am at a conference. I will do my best. But you can also contact Ryan if I am not responsive. If you are in a total panic text me at 919-559-6255.

Notes:

- Nearly the first thing I will do is test that your function gives back expected results. Make sure your function actually executes as expected.

Introduction to the problem

In this exam, we will be working with the a one parameter likelihood function for data we assume to be distributed according to a Poisson distribution. Some of you have less training in probability theory than others. That's OK. The goal here is to give you enough instructions to complete the project with a limited understanding of the actual math or statistics.

Poisson models are typically used in cases where the data (y) is all count data. Thus, the data should be all positive integers (or zero). For data point i, we assume the probability of observing a value y is:

$$Pr(Y_i = y_i) = \frac{\lambda^{y_i} \exp(-\lambda)}{y_i!}$$

So if $y_i = 2$ and $\lambda = 4$, then

$$Pr(Y_i = 2) = \frac{4^2 \exp(-4)}{2 \cdot 1}$$

The basic idea here is that we have *observed* a vector of data (y) and we are trying to *estimate* λ. Your goal in this exam is to develop a package to estimate λ for a set of observed data. Don't worry, we'll tell you exactly how to do it.

Details

Class definition: PoisMLE

You need to define and S4 class that represents a fitted model. The class should have room for the following slots.

- y: The original data
- MLE: The maximum likelihood estimator for this dataset
- LL: The log likelihood calculated from the observed data assuming the MLE is correct.
- SE: The standard error for the MLE
- SEtype: The method used to calculate the standard error.

logLik

You will make a `logLik` function that calculates a log likelihood for the observed data.

The function should take in the following inputs:

- `y`: The *vector* of observed data
- `lambda`: The assumed value of λ

The function should produce as an output:

- The loglikelihood for the observed data conditioned on the value of λ. Mathematically this is:

$$LL(\lambda) = -n\lambda - \sum_{i=1}^{n} \ln(y_i!) + \ln(\lambda) \sum_{i=1}^{n} y_i$$

where n is the number of observations.

mle

You will make a `mle` function that calculates the maximum likelihood estimator for λ. It will take in the vector of data and output the MLE.

The formula for the MLE is:

$$MLE = \frac{\sum_{i=1}^{n} y_i}{n}$$

standardError

You will also make a function that calculates the standard error. This will take in a vector of the data but also an argument for `SEtype`. `SEtype` can take on two different values: `basic` and `bootstrap`.

`basic` If the standard error is basic, the function should return:

$$\sqrt{\frac{MLE}{n}}$$

`bootstrap` Here is the tricky part. If the user chooses the `bootstrap` option you are going to calculate as follows. Let B be the number of bootstrapped resamplings (you will have to set this or let the user set it). You are going to:

1. Create B samples from y where you sample **WITH REPLACEMENT** a sample of size n from the original vector y. The result should be a matrix that is n by B. Ask questions if this is confusing to you. Wikipedia is also helpful.
2. Calculate the MLE for each of these samples. So you will have a vector of MLEs of length B.
3. Find the standard deviation of the results from step 2.

The output of step 3 is the bootstrapped standard error.

estimatePois

The final function you will make will make use of the various functions you defined above. It will take in data and return an object of class `PoisMLE` (see above). That means it should return an object structured like this and of the appropriate class:

- y: The original data
- MLE: The maximum likelihood estimator for this dataset
- LL: The log likelihood calculated from the observed data assuming the MLE is correct. (So you will put in the MLE for the lambda argument.)
- SE: The standard error for the MLE
- SEtype: The method used to calculate the standard error.

The user should put in the data (y) and the option for the standard errors. You may want to include other options depending on how you implement the functions above. (For instance, you may want to let the user set the number of bootstrapped samples.)

Extra credit

For extra credit, create a plotting function for `PoisMLE` that shows the MLE plus and minus 1.96 standard errors. Be creative.