R Problem Set 2

Your name goes here

Due date: September 15, 2017

Contents

Instructions

Please print and submit a hardcopy of your completed problem set (the knitted .pdf) in class, 9:00AM Friday, September 15, 2017. Late submissions will not be accepted. *Note*: please do *not* email files. It will be easier to grade your problem sets if given the hardcopy.

Please do the following problems. When you see a line of code like `set.seed(123)`, just leave it. It is to make your assignments more easily comparable from student to student.

Please turn in partial answers and do your best if you get stuck somewhere. If you have code that will not run, but want to submit it and have the assignment knit, use the `eval=FALSE` argument in the R-codeblock `{r, echo=FALSE}`.

How to do this problem set

You will get more out of this if you try the assignment on your own first. If you get stuck, search online for a solution. The best way to do this is to google your error messages, or describe what you're trying to do, along with the letter R. For instance, "how do I create a data frame in R?" or "Error in '[.data.frame'(dat, 4) : undefined columns selected." If you get stuck, team work is permitted, but you must write up your own solutions.

R code goes in code chunks like this:

```
print("Hello World")
```

```
## [1] "Hello World"
```

They begin with three backticks (the key below the escape key in the upper left part of your keyboard) and {r}. They end with three backticks, so that the markdown compiler knows to go back to printing text, rather than evaluating your code. You may in the future want to pass additional arguments to knitr (the engine that compiles R markdown documents) by including additional arguments in the curly braces. For now, you can use the defaults. A keyboard shortcut for creating a new code chunk is Command-Option-I (Mac) or Control-Alt-I (Windows).

Put your solutions below the questions in this .Rmd file. It is a good practice to periodically compile the document using the "Knit PDF" button. This lets you check on whether the document has any errors in it.

You should also test your code as you are writing it by pressing Command-Enter (Mac) or Ctrl-Enter (Windows). If you have code selected, it will run that block of code. If you do not have any code selected, it will run the line of code where your cursor is.

Part 1

Optimizing OLS Regression

In Lab 4, we used optim to minimize the sum of squared errors for a linear regression with an intercept and one variable, while this approach is different from the matrix-based estimator we used to estimate OLS in the Lab 3 Homework, both approaches should (asymptotically) give us the same answers.

Let's revisit a bigger version of the data set we used during Lab 4 in a slightly new context. In particular, we'll use optim to find the parameters that minimize the sum of squared errors for a linear regression predicting "free.trade.support" with an intercept term and three variables: "income", "republican" and "democrat". That is, we're saying that we can predict whether or not someone supports limited protectionism based on their income and their partisanship (i.e., whether they are a republican, a democrat or an independent).

To start, load the R Data Frame "trade2.Rdata". This data frame can be found on Github. Use the "ls" command to find the name of the object you just loaded. Look over the data with the head() and str() functions. You can also look at the dataset in R Studio using View()

1a) Write a function, like our "trade.preferences" function in Lab 4, that you can pass to optim in order to minimize the sum of square errors for an OLS model predicting "free.trade.support" with the "income", "republican" and "democrat" variables. In particular, this will require you to write a function, "sum.squared.errors". This function should take 1) a vector of four parameter values at which to start optimizing, 2) a data frame, 3) a character vector with the name of the dependent variable and 4) a character vector with the names of the independent variables. This function should then return the sum of square errors based on the set of parameter values you pass to it.

1b) .Somewhere inside your function you should account for the missing data in this data frame. But, only drop observations if they are missing data for the variables that are included in the model (e.g., if a row is missing data on "education" in this case do not drop it in this case since that variable does not appear in the model). We suggest using either the "na.rm" or "complete.cases" functions to do this, but you are free to try other ways as well. ?complete.cases

1c) Once you've written the function "sum.squared.errors", use "optim" to find the parameter values that MINIMIZE the sum of square errors for a model predicting "free.trade.support" with the "income", "republican" and "democrat" variables.

1d) Use R's canned function "lm()" estimate the same model. You can use "summary()" on an lm object to pull out the resulting estimates.

1e)Using either a table or a plot, show the difference between the estimates based on the function you used to obtain them (i.e., "optim" estimates for beta0, beta1, beta2 and beta3 versus "lm" estimates for the same coefficients).

Maximum Likelihood

2a) In class Justin mentioned maximum likelihood estimation, a technique for developing estimators. Write a function, "mean.lik" for the log likelihood of the sample mean. This function should takes a vector of guesses for the value of the sample mean and a vector containing a single variable for which we want to estimate the sample mean. The function should return the resulting log-likelihood for a given guess of the sample mean.

NOTE 1

There is some missing data for "income". Make sure your function can handle missing data.

NOTE 2

The log likelihood function you want to maximize for this problem is found on Github in the document "likelihoodsforlab2.pdf".

2b) Using a sequence of guesses for the sample mean ranging from 30000 to 80000, use your function to find the resulting log likelihood values for the "income" variable in the "trade" data frame. Plot the results with the sequence of parameter guesses on the x-axis and the resulting log likelihoods on the y-axis. Label the axes in the graph and provide a title.

2c) Using "optimize" and your function, "mean.lik" , find the value that maximizes this log likelihood function. At what value is the log likelihood of the sample mean of trade$income maximized?

2d) Check your work using the "mean" function on trade$income. Do you get the same estimate from "optimize" and "mean"?

2e) Draw a vertical line at the value of your answer to 2D on your graph of the log likelihood from 2B.

Part 2

Introduction

Load in the data set HorseKicks.csv.

These data were collected by von Bortkiewicz (1898) (source: http://www.math.uah.edu/stat/data/HorseKicks.html), and convey the number of Prussian calvarymen being killed by the kick of a horse. The data set is famous for the degree to which the distribution of horse kicks corresponds to a Poisson distribution. In this homework, we will try to verify this claim.

Recall that the probability mass function (PMF) of a poisson distribution of a discrete random variable X is:

$$P(X = k) = p(k) = e^{-\lambda}\frac{\lambda^k}{k!},$$

for $k \in \{0, 1, 2...\}$ and 0 otherwise.

The rate parameter λ is the single parameter we need to define a poisson distribution.

Problems

Problem 1

The data contain a column of years, and then data on the number of horse deaths in several different corps of the army. There are a total of 280 observations, each of which represents the number of deaths recorded in a year, in a corps.Reshape the data into a dataset that has just two columns: "year" and "deaths". Stack each corps'data on top of each other. In other words, the reshaped dataset should still have 280 observations. The first 20 rows show the death of corps 1 between 1875 and 1894, the next 20 rows show the death of corps 2 in the same time period, etc.

Hint: there are multiple ways to do this reshape. One way is to use the relevant functions in the "reshape" package. Please note you may need to install the package if you have not yet.

Problem 2

It turns out that the maximum likelihood estimator for λ (the rate parameter) in a Poisson distribution is the sample mean. Take the mean of the "deaths" column to estimate λ and save it in an object called lambda.hat.

Problem 3

Program a function that takes an input "x", which is a scalar equal to the number of times an event occurs, and another input "lambda", which is a scalar equal to the rate parameter of a Poisson distribution. Have the function return the probability that x events occur in a given period, given that x is distributed Poisson. In other words, program the probability mass function of a Poisson random variable.

Problem 4

Now we wanna confirm that your function produces the same results as the canned function (e.g. dpois, ppois etc.). Plot the PMF and CMF of poisson distribution above (set the rate parameter equal to lambda.hat) over the range of 0 to 10 with your own function and with the canned function in R respectively.

Problem 5

Use your function to construct a data frame of probabilities. Specifically, the table should have a column called "Number of Deaths", and a column called "Probability." Have the column "Number of Deaths" include the integers between 0 and 10. In the "Probability" column, display the output of your function given that x = each integer in "Number of Deaths", and given lambda=lambda.hat. What is the probability of 4 calvarymen experiencing death by horse kick in a year given these parameters? Of 6 calvarymen? How about the probability that 3 or fewer calvarymen are killed in a year?

Problem 6

Add a column to your results dataframe called "Expected occurances" that is equal to the estimated number of occurances in a data set of this size that we should expect each value of "Number of Deaths" to appear, given these probabilities.

Problem 7

Add another column to the table called "Actual occurances" that is equal to the number of observations in which the value in "Number of Deaths" appears in the data. How does the predicted number of occurances compare to the observed number of occurances? Make an additional column in the results data frame called "Error" that is equal to the difference between the actual number of occurances and expected number of occurances.

Problem 8

We calculated lambda.hat based on our sample of 280 observations. But we might expect that if we collected many such samples, and estimated lambda.hat each time, we'd get a slightly different result. Write a loop that draws a sample of dd$deaths of the same size as the original data 10,000 times with replacement, estimates lambda.hat each of those times using the same method as before, and saves the estimate in a vector called lambda.boot. Set your seed to 55.

Problem 9

Plot the distribution of estimated lambda.hats in a density histogram with 30 bins. Label the x axis "Lambda Hat" and the y axis "Frequency" and give the plot a main title. Bonus points if you can use the Greek letter lambda with the hat symbol above it (i.e. $\hat{\lambda}$) in the x-axis label instead of writing "Lambda Hat".

Problem 10

What famous distribution does this resemble? Overlay this famous distribution in a line on top of the histogram and discuss how well it fits. Please note you need to redraw the historgram in problem 9 first and set the vertical axis as probability densities rather than frequencies.

Add vertical lines to the histogram that bound the 95% confidence interval around the mean estimate of Lambda. What are the upper and lower bounds of the 95% confidence interval?

Problem 11

Finally, re-generate the prediction table you made earlier using the mean estimate of Lambda you obtained via simulation. How does this estimate compare to the analytic estimate we began working with? Does it generate similar predictions?