Exercises
R For Simulations Columbia University
EPIC 2015
(no answers)

C DiMaggio

June 10, 2015

Contents

1 Sampling and Simulations

Let's start with some binomial simulations. [1] The following code is a single draw, from a sample of 10 with a probability of 0.4. Run it a few times.

```
> set.seed(100)
> rbinom(n=1,size=10,prob=0.4)
```

```
[1] 3
```

Think of this code as defining a binomial "experiment". Change the rules of the experiment so that the probability is 0.2, and run that a few times.

Each run is a "simulation". Do 300 simulations of the same "experiment". (You just need to change one number in the code.)

Now, run 3,000 simulations. Assign them to a variable called "sims". What is the mean and median of this series of simulations? Plot the frequency table of the object "sims".

Re-run the simulations setting the probability to 0.8.

In R, you can sample from most any probability distribution with the *rxxxx()* group of functions. For a simple random sample, you can use *sample()*. Note the specification for replacement.

2 Drawing Statistical Inferences on a Continuous Variable

About 52% of American adults are women. Their height is approximately normally distributed with a mean of 63.7 inches with a standard deviation of 2.7 inches. The average height of adult American men is 69.1 inches with a standard deviation of 2.9 inches.

2.1 Simulations With For Loops

2.1.1 Single sample: average

Write R code to draw a single random sample of 10 American adult heights. What is the average height of your 10 Americans?

[1] If you're interested in a more thorough introduction to using probability distributions in R, look here

2.1.2 Repeat samples: average

Repeat this random sample 1,000 times using a for loop and calculate the average height for each simulation. Create a histogram of your set of 1,000 average heights.

2.1.3 Repeat samples: Max

Write another for loop of 1000 height simulations of a 10-person sample. This time draw inferences on and plot the tallest heights.

2.2 Simulations with functions

2.2.1 Single sample: average

Write a function that returns the mean height of a random sample of 10 American adults. What is the mean height of your 10-person sample?

Repeat Samples: Average

Write code to repeat your height function 1000 times. Create a histogram of your simulations.

3 Simulation for GLM predictions: Arsenic in Bangladeshi Wells

Before demonstrating how to do predictions for a GLM, let's take a little time to fully understand the model we will be using.

3.1 Logistic Model for Switching Wells

A survey was taken of arsenic levels in wells in an area of Bangladesh. [2] Investigators returned a few years later to see if people drawing water from unsafe wells switched to nearby wells after being informed, or continued using their own contaminated wells. This dichotomous outcome can be modeled with a logistic regression model. Using the "wells" data set, run a logistic model with the outcome variable "switch" and a single predictor variable, the distance of the known closest safe well, "dist".

[2] Gelman and Hill, p86

The distance is measured in meters, so the coefficient, seems small. Try rescaling distance in 100 meter units. What is the resulting predictive model? (In the form of $Pr(switch) = logit^{-1}(\alpha - \beta \cdot distance)$) How to you interpret this model?

We might expect that that higher arsenic levels would increase the probability of switching, so our model may potentially be more informative by adding the effect of the arsenic level. Run a model that includes the variable "arsenic". How do you interpret this model?

We might expect some interaction between distance and arsenic level. Run a model with an interaction term. Interpret the results.

To interpret these results, we can use the divide by 4 rule, and mean values for dist100 (0.48) and arsenic (1.66):

3.2 Simulating Predictions for New Data

The approach to using simulations for predictions of complex GLMs proceeds as it does for linear regressions. Start by fitting the regression, create a matrix using the results of the regression, use *arm::sim()*, collect and summarize the results. In the code below, we will also plot the intercept vs. the predictor coefficient to illustrate the uncertainty in the coefficients as well as in the regression curve.

If you did not already do so in the previous section, load the "wells" data and run a simple logistic regression with "switch" as the outcome variable and "dist" as the predictor.

```
> library(arm)
> wells<-read.table("/Users/charlie/Dropbox/gelmanHillNotesSlides/ARM_Data/arsenic/wells
> fit.1 <- glm (switch ~ dist,family=binomial(link="logit"), data=wells)
> display(fit.1, digits=3)
```

Use *cbind()* to create a simple regression matrix called "X.tilde" that includes the number 1 in the first column (to represent a single-level regression) and the observations for the distance variable from the "wells" data set as the predictor variable. Create a variable called "n.tilde" to represent the number of observations in "X.tilde".

```
> X.tilde <- cbind (1, wells$dist)
> n.tilde <- nrow (X.tilde)
```

Use the*arm::sim()* function to run 1,000 simulations of the simple logistic regression model you ran above. Save the simulations into an object called "sim.1".

```
> sim.1 <- sim (fit.1, n.sims=1000)
```

To get a better sense of what *sim()* is doing behind the scenes, take a look at the structure of the "sim.1" object you just created. It is an S4-class object, so we use

the "@" delimiter for "slots". The "coef" slot holds the simulations, in this case the first column holds the intercept simulation, the second column holds the predictor variable simulations. Plot the intercept vs. the predictor.

```
> str(sim.1)
> plot(sim.1@coef[,1],sim.1@coef[,2],xlab=expression(intercept),
+ ylab=expression(distance))
```

We can use the simulation to illustrate the uncertainty in our regression line. In the following code, we start by setting an empty plot of the switching and distance observations from the "wells" data set.[3] Then we plot the first 20 simulations from the "sim1" object we created, and overlay the regression line from the from regression object.

```
> plot (wells$dist, wells$switch, type="n")
> for (s in 1:20){
+ curve (invlogit (sim.1@coef[s,1] + sim.1@coef[s,2]*x), col="gray", add=TRUE)}
> curve (invlogit (fit.1$coef[1] + fit.1$coef[2]*x), add=TRUE, col="red")
>
```

To predict the switching behavior of the *n.tilde* "new" households by simulation, set up a the matrix as we did with the linear regression of congressional voting districts, and run the simulation, only this time use the binomial distribution rather than the normal distribution to get the outcome variable. Finally, summarize the results in a way that makes sense to the question you are asking.

```
>         n.sims <- 1000
>         y.tilde <- array (NA, c(n.sims, n.tilde))
>         my.funct<-function()
+         for (s in 1:n.sims){
+           p.tilde <- invlogit (X.tilde %*% sim.1@coef[s,]) # probability of switching
+           y.tilde[s,] <- rbinom (n.tilde, 1, p.tilde) # whether switched or not
+         }
>         summary((apply(y.tilde[1:nrow(y.tilde),],1,sum))/n.tilde)        # summarize r
>
```

You can use similar approaches for Poisson regressions with rpois() and negative binomial models with rnegbin().

[3] You can see what the plot actually looks like by removing the *type="n"* option, but it's just a not very interesting or informative set of 0's and 1's.