Exercises
R For Simulations Columbia University
EPIC 2015
(with 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.

```
> set.seed(111)
> rbinom(n=1,size=10,prob=0.2)
```

```
[1] 2
```

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

```
> set.seed(121)
> rbinom(n=300,size=10,prob=0.2)
```

```
  [1] 2 4 2 3 2 2 2 2 1 1 4 3 2 1 2 1 1 2 3 3 4 0 1 2 1 1 1 1 3 0 3 0 0 2 4 0 1
 [38] 4 3 2 2 2 2 0 2 0 3 4 3 3 1 0 3 0 1 2 1 2 1 2 4 3 2 4 1 0 0 2 2 5 3 1 3 1
 [75] 3 2 2 2 3 2 1 3 3 3 0 2 2 4 1 1 2 1 3 3 2 2 2 3 2 0 1 4 0 4 1 4 3 2 1 1 3
[112] 3 0 1 2 0 3 1 4 2 2 2 1 1 0 3 3 3 2 2 3 3 2 5 1 2 0 2 3 2 3 2 3 3 2 1 1 2
[149] 1 3 3 3 2 6 2 2 1 3 3 4 1 1 2 3 3 2 0 3 3 0 2 2 3 3 1 1 3 3 1 4 1 2 1 3 3
[186] 4 1 1 1 1 1 1 2 2 1 1 1 1 2 1 2 3 1 2 4 2 1 3 1 1 1 2 3 0 3 1 1 3 3 2 3 1
[223] 2 2 2 1 2 2 2 2 2 2 4 2 4 3 2 3 0 1 3 3 2 3 1 3 2 1 3 1 2 3 2 2 1 2 2 3 1
[260] 3 3 1 2 4 3 1 3 2 0 0 2 3 3 2 2 1 2 3 0 1 4 1 1 3 0 4 2 2 0 3 2 3 2 0 2 1
[297] 1 4 2 0
```

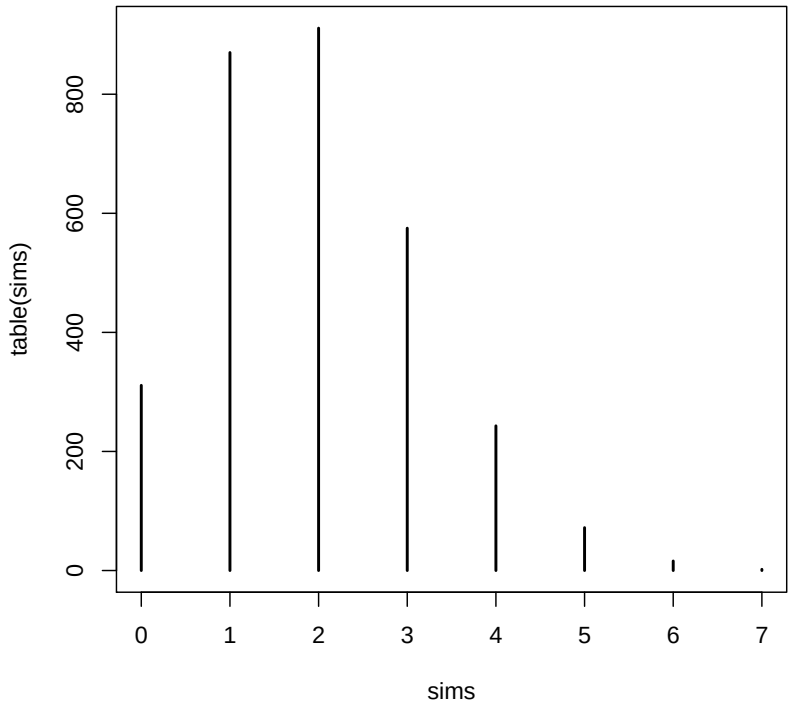
Now, run 3,000 simulations. Assign them to a variable called "sims". What is the mean and median of this series of simulations? Plot the frequency table of the object "sims".

```
> set.seed(131)
> sims<-rbinom(n=3000,size=10,prob=0.2)
> summary(sims)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   1.000   2.000   1.953   3.000   7.000
```

```
> plot(table(sims))
```

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

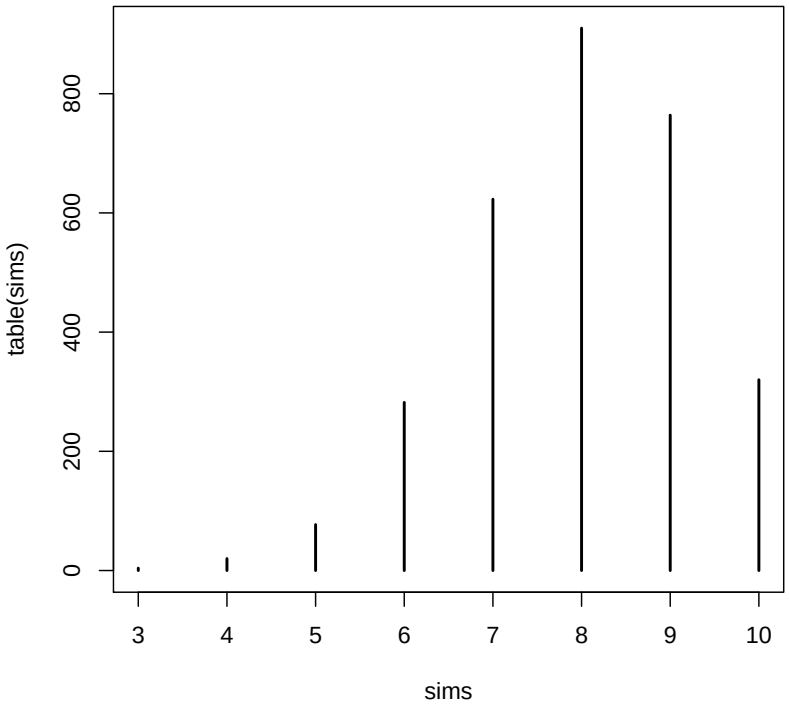


Re-run the simulations setting the probability to 0.8.

```
> set.seed(141)
> sims<-rbinom(n=3000,size=10,prob=0.8)
> summary(sims)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  3.000   7.000   8.000   7.962   9.000  10.000

> plot(table(sims))
```



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.

1.1 Simulating Risk Ratios

We can use simulations to approximate results for many problems when there is no direct or closed solution, which is often the case in Bayesian analysis, when the underlying distribution is unknown, or for small sample sizes. It may also be helpful in power calculations. To put the process in a context that is more familiar to epidemiologists, let's look at risk ratios. The observed rates of disease in exposed and unexposed populations can serve as the probabilities in two sets of simulated experiments using *rbinom()*. Divide those results by the number of simulations and use the mean and 0.025 tails for the point estimate and confidence limits. The process is sometimes referred to as "bootstrapping". In outline it looks like this:

1. simulate 5000 replicate bernoulli trials in sample size n_1 = exposed

2. divide those results by n_1 to get 5000 simulated risk estimates for the exposed group
3. repeat that process for the unexposed group n_2
4. divide 5000 simulated risks in exposed by 5000 simulated risks in unexposed to get 5000 simulate relative risks
5. calcualate mean and 0.25 tails from that population

Let's walk through the process with an example. In a ground breaking 1987 study, Hennekens and colleagues reported on the protective benefits of aspirin. There were 104 myocardial infarctions (fatal and non-fatal) among 11,037 people in the treatment group, and 189 MI's among 11,034 people in the placebo group. We calculate the risk ratio and confidence interval using the usual, log-approximated approach found in the "epitools" package.

```
> library(epitools)
> asa.tab<- matrix(c(104,11037,189,11034),2,2)
> epitab(asa.tab, method="riskratio")
```

```
$tab
          Outcome
Predictor  Disease1        p0 Disease2        p1 riskratio     lower     upper
  Exposed1      104 0.3549488      189 0.6450512 1.0000000        NA        NA
  Exposed2    11037 0.5000680    11034 0.4999320 0.7750269 0.7111913 0.8445924
          Outcome
Predictor       p.value
  Exposed1           NA
  Exposed2 8.978571e-07

$measure
[1] "wald"

$conf.level
[1] 0.95

$pvalue
[1] "fisher.exact"
```

We'll compare these results to those from a simulation-based approach. Begin by "setting a seed" for the random draws, so you can replicate your results later on. Then use *rbinom()* to repeat 5,000 times an experiment where we count the number of outcomes (MI's) in two populations, with the probability of the outcome in a population defined by the results of the Hennekens study.

```
> set.seed(151)
> tx <- rbinom(5000, 11037, 104/11037)
> plac <- rbinom(5000, 11034, 189/11034)
```

In this next step, for each of the 5,000 experimental replicates, we divide the number of outcomes by the number of people in each population to get a 5,000 risk estimates for each group. Then, again for each of the 5,000 experiments or replicates, we divide the risk in the treatment group by the risk in the placebo group to get 5,000 estimates of the relative risk.

```
> r.tx<-tx/11037
> r.plac<-plac/11034
> rr.sim <- r.tx/r.plac
```

Now, we get the statistics in which we are interested.

```
> mean(rr.sim)

[1] 0.552865

> quantile(rr.sim, c(0.025, 0.975))

     2.5%     97.5%
0.4298172 0.6949400
```

The sample standard deviation is an estimate of the standard error.

```
> sd(rr.sim)

[1] 0.06748942
```

1.2 Bootstrap Function

Try writing a function to calculate bootstrap estimates of relative risks. Test the function using the aspirin example above.

```
> my.boot<-function(x){
+ p1<-rbinom(5000, x[1,2], x[1,1]/x[1,2])
+ p2<-rbinom(5000, x[2,2], x[2,1]/x[2,2])
+ r.p1<-p1/x[1,2]
+ r.p2<-p2/x[2,2]
+ rr.sim<-r.p1/r.p2
+ mu<-mean(rr.sim)
+ ci<-quantile(rr.sim, c(0.025, 0.975))
+ se<-sd(rr.sim)
+ list(rr.boot=mu, rr.ci=unname(ci), stand.err=se)
+ }
```

```
> asa.tab<-matrix(c(104, 11037, 189, 11034), 2, 2, byrow=T)
> my.boot(asa.tab)

$rr.boot
[1] 0.5545668

$rr.ci
[1] 0.4315711 0.6931526

$stand.err
[1] 0.06769169

>
```

You may find the following function, written by Tomas Aragon useful, though you can probably write a simpler version for our purposes.

```
> rr.boot <- function(x, conf.level = 0.95, replicates = 5000){ ##prepare input
+ x1 <- x[1,1]; n1 <- sum(x[1,])
+ x0 <- x[2,1]; n0 <- sum(x[2,])
+ ##calculate
+ p1 <- x1/n1 ##risk among exposed
+ p0 <- x0/n0 ##risk among unexposed
+ RR <- p1/p0
+ r1 <- rbinom(replicates, n1, p1)/n1
+ x0.boot <- x0.boot2 <- rbinom(replicates, n0, p0)
+ x0.boot[x0.boot2==0] <- x0.boot2 + 1
+ n0.denom <- rep(n0, replicates)
+ n0.denom[x0.boot2==0] <- n0.denom + 1
+ r0 <- x0.boot/n0.denom
+ rrboot <- r1/r0
+ rrbar <- mean(rrboot)
+ alpha <- 1 - conf.level
+ ci <- quantile(rrboot, c(alpha/2, 1-alpha/2))
+ ##collect
+ list(x = x, risks = c(p1 = p1, p0 = p0), risk.ratio = RR,
+ rrboot.mean = rrbar, conf.int = unname(ci), conf.level =
+ conf.level, replicates = replicates) }
```

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?

```
> sex <- rbinom (10, 1, .52)
> height <- ifelse (sex==0, rnorm (10, 69.1, 2.9), rnorm (10, 64.5, 2.7))
> avg.height <- mean(height)
> print (avg.height)
```

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.

```
> n.sims <- 1000
> avg.height <- rep (NA, n.sims)
> for (i in 1:n.sims){
+ sex <- rbinom (10, 1, .52)
+ height <- ifelse (sex==0, rnorm (10, 69.1, 2.9), rnorm (10, 64.5, 2.7))
+ avg.height[i] <- mean (height)
+ }
> hist (avg.height, main="Average height of 10 adults")
```

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.

```
> n.sims <- 1000
> max.height <- rep (NA, n.sims)
```

```
> for (i in 1:n.sims){
+ sex <- rbinom (10, 1, .52)
+ height <- ifelse (sex==0, rnorm (10, 69.1, 2.9), rnorm (10, 64.5, 2.7))
+ max.height[i] <- max(height)
+ }
> hist (max.height, main="Tallest height of 10 adults")
```

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?

```
> height.sim<-function(n.adults){
+ sex<-rbinom(n.adults, 1, 0.52)
+ height <- ifelse (sex==0, rnorm (10, 69.5, 2.9),
+ rnorm (10, 64.5, 2.7))
+ return (mean(height))
+ }
```

Repeat Samples: Average

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

```
> avg.height<-replicate(1000, height.sim(n.adults=10))
> hist(avg.height)
```

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

[2] Gelman and Hill, p86

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".

```
> 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)

            coef.est coef.se
(Intercept)  0.606    0.060
dist        -0.006    0.001
---
  n = 3020, k = 2
  residual deviance = 4076.2, null deviance = 4118.1 (difference = 41.9)
```

The distance is measured in meters, so the coefficient, -0.006, seems small. Try rescaling distance in 100 meter units.

```
> wells$dist100<-wells$dist/100
> fit.2 <- glm (switch ~ dist100,family=binomial(link="logit"), data=wells)
> display(fit.2)

            coef.est coef.se
(Intercept)  0.61     0.06
dist100     -0.62     0.10
```

The predictive model then is $Pr(switch) = logit^{-1}(0.61 - 0.62 \cdot dist100)$:

- The intercept is interpreted as the probability of switching when the distance between wells is zero (nonsensical, I know ...), the is $logit^{-1}(0.61) = 0.65$. We can use the the *invlogit()* function in "arm" to obtain the value of a 65% probability of switching when uncontaminated well is right next to you.
- We can evaluate the coefficient as the effect of distance. Gelmen and Hill present a handy rule of thumb (see page 89) with *the divide by 4 rule*. Simply divide the coefficient on the log scale by 4 to obtain the probability. In our model, $-0.62/4 = -0.15$. So, adding 1 to $dist100$ (i.e. 100 meters to nearest uncontaminated well) decreases the probability of switching by 15%

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"

```
>         fit.3<-glm (switch ~ dist100 + arsenic,family=binomial(link="logit"), data=wel
>         display(fit.3)

            coef.est coef.se
(Intercept)  0.00     0.08
```

```
dist100     -0.90     0.10
arsenic      0.46     0.04
```

Both coefficients are statistically significant and in the expected direction. Comparing two wells with the same arsenic level, each distance increment of 100 meters translates to a $-90/4 = -.22$ or a 22% lower probability of switching. Conversely, for two equidistant wells, each 1 unit increase in arsenic level translates to a $0.46/4$ or 11% increase in the probability of switching. We might expect some interaction between distance and arsenic level. Run a model with such an interaction term.

```
>         fit.4<-glm (switch ~ dist100 + arsenic + dist100:arsenic,family=binomial(link=
>         display(fit.4)

                coef.est coef.se
(Intercept)     -0.15     0.12
dist100         -0.58     0.21
arsenic          0.56     0.07
dist100:arsenic -0.18     0.10
```

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

- The *intercept* is probability of switching if the distance to the nearest well is 0 and arsenic level is 0, ($logit^{-1}(-0.15) = 0.47$). Since this is non-interpretable, we substitute the mean terms for distance and arsenic: $logit^{-1}(-0.15 - 0.58 \cdot 0.48 + 0.56 \cdot 1.66 - 0.18 \cdot 0.48 \cdot 1.66) = 0.59$
- The *coefficient for distance* compares two wells if the arsenic level for both is zero. Again, probably not a useful interpretation. We again substitute the average value for arsenic into the equation to get the linear term $-0.58 - 0.18 \cdot 1.66 = -0.88$, and use the rule of 4, $-0.88/4$ to interpret a 22% decrease in the probability of switching for each 100 meters if the arsenic is at the mean level
- The *coefficient for arsenic* compares two wells that differ by 1 unit of arsenic level if the distance to the nearest safe well is 0. Again, substituting the mean value for distance, we get $0.56 - 0.18 \cdot 0.48 = 0.47$, and $0.47/4 = 12\%$ increased probability of switching wells for every 1 unit increase in arsenic level for two wells spaced 48 meters apart
- The *interaction term* adds -0.18 to the coefficient for distance for each additional unit of arsenic. So, adding -0.18 to the -0.88 we calculated for the effect of distance, we may conclude effect (or "importance") of distance increases for households with higher levels of arsenic. *Alternatively* by adding -0.18 to the coefficient for arsenic for each 100 meters of distance to the 0.47 effect we calculated for the effect of arsenic, we may conclude that importance of arsenic decreases for wells farther away. You may note that the interaction term is not

quite statistically significant. But, it makes sense and is in the right direction, so we should probably keep it.

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.