POL 345: Quantitative Analysis and Politics

Precept Handout 6

Week 7 (Verzani Chapter 5)

Remember to complete the entire handout and submit the precept questions to the Blackboard 24 hours before precept. In this handout, we cover the following new materials:

- Using `sample()` to calculate probability through simulation
- Using `pnorm()` and `pbinom` to evaluate cumulative distribution functions
- Using `rnorm()` and `rbinom()` to take random draws from the normal and binomial distribution
- Using `dnorm()` and `dbinom()` to evaluate probability density (distribution) functions (pdf) of the normal and binomial distribution
- Using `sort()` to sort vectors

1 Calculating Probability through Simulation

Recall that the probability can be thought of as the "limit" of repeated identical experiments. Using a loop to repeat an experiment, we may calculate an approximate probability of specified events.

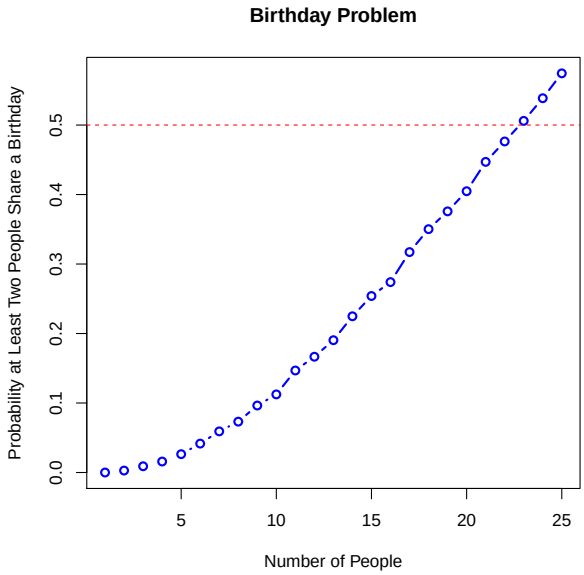
- The function `sample(X, Y, replace = TRUE, prob = P)` will randomly sample `Y` units from a vector `X`. Sampling may be done with or without replacement (`replace = TRUE` or `replace = FALSE`). The argument `prob = P` denotes a vector of probability for observing elements of vector `X`. The default is equal probability.

```
> Z <- seq(from = 2, to = 16, by = 2) # Create vector to draw samples from
> sample(Z, 7, replace = TRUE)  # Randomly draw 7 samples from Z, with replacement

[1]  8 16  2 12  4 16 10

> sample(Z, 7, replace = FALSE) # Randomly draw 7 samples from Z, without replacement

[1]  2  8 10 14  4 12  6

> ## Randomly draw 7 samples from Z, with replacement and unequal probabilities,
> ## where prob = is a vector of probability weights
> sample(Z, 7, replace = TRUE, prob = c(1,1,1,1,1,1,7,7))

[1] 16 16 12 14 14 16 14
```

- We explore calculating probability through simulation with the birthday problem. In this problem, we determine how many students must be in a class in order for the probability that at least two students have the same birthday to exceed 0.5. We answer this question by using a loop within a loop to repeat (i.e.: simulate) an experiment several times.

```
> sims <- 5000 # Specify number of simulations
> bday <- 1:365 # Create sequence to represent all possible birthdays
> ## Create empty container for our answers, where NA indicates
> ## missing data that we will fill with data generated by the loop
> answer <- rep(NA, 25)
> ## Generate a simulation through a loop nested within a loop
> ## The inner loop (indexed by i) is the simulation
> ## The outer loop (indexed by k) uses the results of the simulation to generate
> ## the probability that at least two share a birthday
> for (k in 1:25) {
+   count <- 0  # Start counter of simulations that meet condition
+   for (i in 1:sims) {
+     class <- sample(bday, k, replace = TRUE) # sampling with replacement
+     if (length(unique(class)) < length(class)) {
+       count <- count + 1 # add one to counter if any kids share the birthday
+     }
+   }
+   ## Store the answers (counter of simulations that meet condition
```

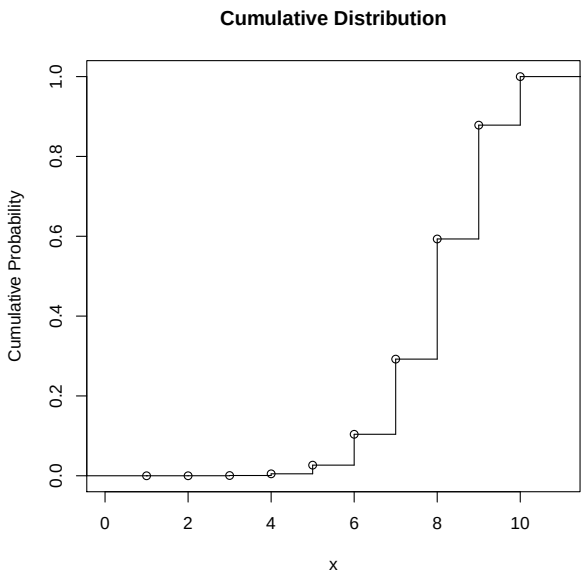
```
+   ## divided by total number of simulations)
+   answer[k] <- count/sims
+ }
> answer

 [1] 0.0000 0.0028 0.0090 0.0158 0.0264 0.0416 0.0592 0.0732 0.0964
[10] 0.1124 0.1468 0.1666 0.1904 0.2248 0.2540 0.2740 0.3172 0.3502
[19] 0.3758 0.4048 0.4470 0.4764 0.5060 0.5386 0.5744

> plot(1:25, answer, type = "b", col="blue", lwd = 2, xlab = "Number of People",
+      ylab = "Probability at Least Two People Share a Birthday",
+      main = "Birthday Problem")
> abline(h = 0.5, col = "red", lty=2)
```



2 Distribution Functions

Recall from the lecture that the **(cumulative) distribution function** of the random variable X, denoted by $F(x)$, is equal to the probability of X taking a value less than or equal to x.

- The function `pbinom(q, size, prob, lower.tail = TRUE)` will take in a vector of values, `q`, and report the corresponding proportion of times we would expect to see `q` or fewer successes when we have sample size `size` and probability of success `prob`. The `lower.tail` default is `TRUE`, which specifies the quantity of interest is proportion of all observations having values of `q` or lower, given the mean and sd. Choosing `lower.tail = FALSE` produces probability of values *greater* than `q`.



- Similarly, the function `pnorm(q, mean, sd, lower.tail=TRUE)` will take in a vector of values, `q`, and report the proportion of all observations we would expect to witness having values `q` or lower, given that the distribution is normal with the specified `mean` and `sd`.

- We return to the 2008 Presidential Election example. We again rely on the data file `e08.RData`, available on Blackboard. The data description is available from Precept 4.

- We begin by calculating the mean poll predicted margin of victory for Obama within each state as well as the actual margin of victory. Assume that each poll has the sample size of $n = 1000$ and that the probability of winning a state is equal to $\Pr(X > 0)$ where X is normally distributed with mean equal to the mean margin of victory from the poll (converted from percentage points to probabilities) and the standard deviation is equal to $2\sqrt{p(1-p)/mn}$ where m is the number of polls for that state and $p = (x + 1)/2$ with x being the mean margin of victory (again, in probabilities not percentage points; see the lecture slides if you are not sure where these expressions come from). We use this information to calculate Obama's probability of winning for each state.

```
> load("e08.RData")
> ## strip out poll results
> e08.polls <- e08[e08$DaysToElection != 0, ]
> e08.results <- e08[e08$DaysToElection == 0, ]
> ## calculate the mean poll margin by state
> O.margin.polls <- tapply(e08.polls$Dem - e08.polls$GOP, e08.polls$State, mean)
> ## convert margin to percentage
> means <- O.margin.polls * 0.01
> ## calculate p and sd using formulas from lecture
> p <- (0.01 * O.margin.polls + 1)/2
> poll.n <- table(e08.polls$State)
> sds <- 2*sqrt(p*(1-p)/(poll.n*1000))
> ## determine Obama's probability of winning by state
```

```
> prob.O.win <- pnorm(0, means, sds, lower.tail = FALSE)
> ## Let's look at Obama's probability of winning in Missouri
> prob.mo <- prob.O.win[names(prob.O.win) == "Missouri"]
> prob.mo

Missouri
0.388648
```

3 Random Draws from Probability Distributions

- The function `rbinom(n, s, p)` will create a vector of length `n` containing independent, random draws from a **binomial** distribution with the size of `s` and the probability of success `p`.
- In the following example, `n` is equal to 20, `s` is equal to 7, and `p` is equal to 0.55. In other words, we take twenty independent, random draws from a binomial distribution with a size of 7 and a probability of success equal to 0.55.

```
> rbinom(n = 20, s = 7, p = 0.55)

 [1] 4 4 3 2 3 3 6 5 4 5 4 3 5 5 4 2 3 4 5 5
```

- The function `rnorm(n, mean, sd)` will create a vector of length `n` containing independent, random draws from a **normal** distribution with the specified `mean` and `sd` (standard deviation).
- In the following example, `n` is equal to 10, `mean` is equal to -1, and `sd` is equal to 0.2. In other words, we take 10 independent, random draws from normal distribution with a mean of -1 and a standard deviation of 0.2.

```
> rnorm(n = 10, mean = -1, sd = 0.2)

 [1] -0.712168 -1.348671 -0.934798 -0.701572 -0.964257 -0.558023
 [7] -1.048107 -1.168315 -0.927232 -1.041468
```

4 Probability Density Functions

Recall from the lecture that the **probability density function (pdf)**, denoted by $f(x)$, is the function that equals the likelihood of taking a value x.

- The function `dbinom(x, s, p)` will take in a vector of values, `x`, and report the probability of seeing exactly the number of successes denoted by each value of `x` when we have sample size `s` and probability of success `p`.
- In the example below, our vector `x` contains elements that range from -1 to 4. For each element of x, **R** reports the probability of seeing exactly the number of successes denoted by each element of `x` when we have sample size of 5 and probability of success equal to 0.4.

```
> x <- -1:4
> dbinom(x, 5, 0.4)
```

```
[1] 0.00000 0.07776 0.25920 0.34560 0.23040 0.07680
```

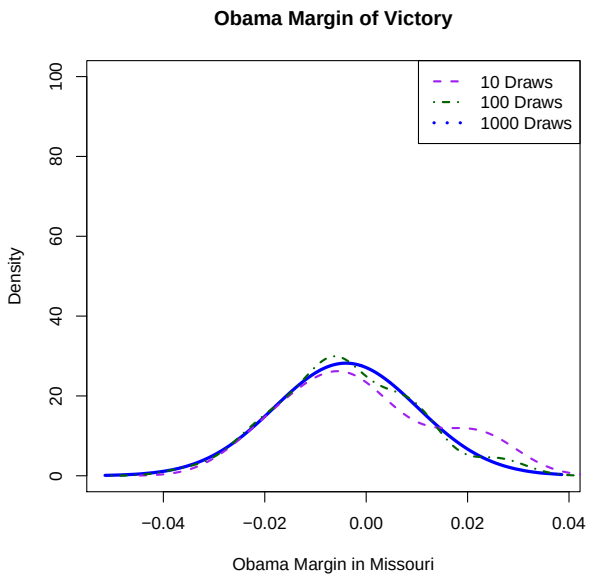
- The 2008 Minnesota Senate race was very close. The difference between Democratic candidate Franken and Republican candidate Coleman was only 312 votes. Most elections, however, are not close. Anthony Downs famously set forth the voter's paradox: for a rational, self-interested citizen, the costs of voting will exceed the benefits of voting – unless the citizen's vote is decisive. If we assume that each voter is equally likely to vote for either the Democrat or the Republican and that each vote is independent, what is the probability of a given vote being decisive? We use the example from lecture to illustrate the voter's paradox:

```
> dbinom(x = 200000, size = 400000, prob = 0.5)

[1] 0.00126157
```

- The function `dnorm(x, mean, sd)` will take in a vector of values, `x`, and report the value of the density function at point `x` for the normal distribution with a specific `mean` and `sd`.

- We again return to the 2008 election example and produce the distribution of Missouri using random draws from normal distribution with the mean and standard deviation from the state. The function `sort()` will sort a vector of elements in a descending or ascending (default is ascending) order.

```
> mo.mean <- means[names(means) == "Missouri"]
> mo.sd <- sds[names(sds) == "Missouri"]
> draw.10 <- rnorm(10, mo.mean, mo.sd)
> draw.100 <- rnorm(100, mo.mean, mo.sd)
> draw.1000 <- rnorm(1000, mo.mean, mo.sd)
> ## generate density for 1000 draws
> draw.1000 <- sort(draw.1000) # sort data to aid graphing
> y.mo <- dnorm(draw.1000, mo.mean, mo.sd)
> plot(draw.1000, y.mo, type = "l", col = "blue", lwd = 3, ylim = c(0, 100),
+      main = "Obama Margin of Victory", ylab = "Density",
+      xlab = "Obama Margin in Missouri")
> lines(density(draw.10), col="purple", lty=2, lwd=2)
> lines(density(draw.100), col="darkgreen", lty=4, lwd=2)
> legend("topright", c("10 Draws", "100 Draws", "1000 Draws"),
+        lty = c(2, 4, 3), lwd = c(2, 2, 3), col=c("purple", "darkgreen", "blue"))
```



5 Precept Questions

In preparation for precept, please answer the following questions. Submit your code following the instructions given in the syllabus.

1. Getting into the Ph.D. program of the Politics Department is known to be very difficult, and there is considerable uncertainty about the admission process. Every year, a secretary types each applicant's name on a separate card together with a matching envelope. Then, he drops the pile from the window of his office on the second floor of Corwin Hall. Finally, he goes downstairs and places the cards randomly in the envelopes. If the name on a card matches with a name on the envelope in which the card is placed, the applicant will be admitted. What is the probability of nobody getting accepted? Does this probability vary as you change the number of applicants from 50 to 500 (by 25)? Create a graph that is similar to the one created above for the birthday problem. **Hint:** Use `sample(..., replace = FALSE)` to represent the random process.

2. In the example above, we calculated the probability of Obama's victory based on the normal distribution for each state, i.e., `prob.O.win`. Simulate the election 5,000 times where you draw the winner for each state according to this probability and allocate electoral votes for the winner (**Hint:** use `rbinom()`). Store the total number of electoral votes for each simulation and plot them as a histogram after the simulation. Add a vertical dashed line indicating the actual election result. According to the simulation result, what is the predicted probability of Obama's victory (calculate this probability by computing the proportion of simulated elections where Obama won the majority of electoral votes)? Comment on the performance of this prediction.