1 Calculate Probability through Simulation

- New commands:
  1. sample()
  2. unique()

1.1 Basic Concepts and Commands

- 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 certain events.
- The function `sample(X, Y, replace = TRUE, prob = P)` will let you sample $Y$ units from a vector $X$ with or without replacement (replace = TRUE or replace = FALSE) using a vector of probability $P$ (the default is equal probability).
- The function `unique()` will return the unique elements of a vector or a dataframe.

```r
> ## Create vector to draw samples from
> Z <- seq(from=2, to=20, by=2)
> Z

[1]  2  4  6  8 10 12 14 16 18 20

> ## Randomly draw 8 samples from Z, with replacement
> sample(Z, 8, replace = TRUE)

[1] 12 16 10 18 20  4 16  2

> ## Randomly draw 8 samples from Z, without replacement
> sample(Z, 8, replace = FALSE)

[1] 10  6 20  8 14 12 18  4

> ## Randomly draw 8 samples from Z, with replacement
> ## and unequal probabilities, where prob = is a vector of
> ## probability weights
> sample(Z, 8, replace = TRUE, prob = c(1,1,1,1,1,1,1,6,6))
```
1.2 Birthday Problem

- How many people do you need in order for the probability that at least two people have the same birthday exceeds 0.5?

- We answer this question by using a loop within a loop to repeat (i.e.: simulate) an experiment several times.

```r
> ## Specify number of simulations
> sims <- 15000
> ## Create sequence to represent all possible birthdays
> bday <- 1:365

> ## 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 our quantity of interest (probability that at least two share a birthday)
> for (k in 1:25) {
+ ## Start counter of simulations that meet condition
+ count <- 0
+ for (i in 1:sims) {
+   ## sampling with replacement
+   class <- sample(bday, k, replace = TRUE)
+   if (length(unique(class)) < length(class)) {
+     ## add one to counter if any kids share the same birthday
+     count <- count + 1
+   }
+ }
+ }
+ 
+ ## Store the answers (counter of simulations that meet condition divided by total number of simulations
+ answer[k] <- count/sims
+ }
> ## Number of students in class before 50/50 chance that at least two share the same birthday
> sum(answer < 0.5)
[1] 22
```
2 Random Draws from Probability Distributions

- New commands:
  1. `rnorm()`
  2. `rbinom()`

- 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.

```r
> ## Ten random draws from N(-1, 0.2)
> rnorm(n=10, mean=-1, sd=0.2)

[1] -1.2333936  -1.0642076  -1.1180926  -1.0228324  -0.8942401
[7] -0.6623037  -1.0476683  -0.9094923  -0.7178358
```
The function \( \texttt{rbinom(n, s, p)} \) 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.

```r
> ## Twenty draws from B(7,0.55)
> rbinom(n=20, s=7, p=0.55)
[1]  5  4  6  4  6  4  3  2  7  2  3  4  3  4  3  4  3  7  3
```

3 Probability Density Functions

- New commands:
  1. \texttt{dnorm()}
  2. \texttt{dbinom()}

- The probability density function (pdf), denoted by \( f(x) \), is the function that equals the likelihood of taking a value \( x \).

- The function \( \texttt{dnorm(x, mean, sd)} \) 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.

- The function \( \texttt{dbinom(x, s, p)} \) 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 \( p \).

```r
> data <- read.csv("turnout.csv", header=TRUE)
> turnout <- data$Turnout / data$VEP
> density <- dnorm(turnout, mean(turnout), sd(turnout))
> turnout ## Proportion of Voting Eligible Population that went to the polls
[1] 0.6196124 0.6851535 0.5666132 0.5390454 0.6248765 0.7034940 0.6758841
[8] 0.6641816 0.6090150 0.6802923 0.6172263 0.5073455 0.6453062 0.6342732
[15] 0.6054873 0.7017127 0.6296375 0.5887549 0.6267981 0.7265119 0.6818243
[22] 0.669165 0.6937776 0.7848446 0.6148484 0.6854747 0.6710364 0.6348207
[29] 0.5868780 0.7216296 0.6690453 0.6056322 0.5857124 0.6646767 0.6595854
[36] 0.6759426 0.5711639 0.6846795 0.6485576 0.6301750 0.5876669 0.6472208
[43] 0.5775653 0.5519162 0.5433659 0.6705004 0.6823415 0.6772415 0.5189949
[50] 0.7285860 0.6576737
> density ## Density estimate, based on normal distribution assumption
[22] 6.2187095 4.4690915 0.2856249 6.3226681 5.0667325 5.996131 6.924634
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.

> #-1 never occurs, hence the zero probability
> x <- -1:4
> dbinom(x, 5, 0.4)

[1] 0.00000 0.07776 0.25920 0.34560 0.23040 0.07680

> a1 <- rnorm(20, mean=2, sd=4)
> a2 <- rnorm(500, mean=2, sd=4)
> ## Create reference distribution, values range 3sd above and below mean
> x <- seq(from=-10, to=14, length.out=1000)
> # length.out indicates desired length of sequence
> y <- dnorm(x, 2, 4)
```r
> plot(x, y, type="l", col="red", lwd=3, ylim=c(0, 0.14), ylab="Density")
> lines(density(a1), col="purple", lty=2, lwd=2)
> lines(density(a2), col="darkgreen", lty=4, lwd=2)
> legend("topright", c("reference N(2,4)", "20 draws from N(2,4)",
+ "500 draws from N(2,4)")",
+ lty=c(1,2,4), lwd=c(3,2,2),
+ col=c("red", "purple", "darkgreen"))
```