BIOS6611 R Lab 1: Sampling, Simulations, and the Central Limit Theorem

Contents

Conceptual Overview 1

Simulation Basics with the Normal Distribution 2

Non-Normal Distributions 5

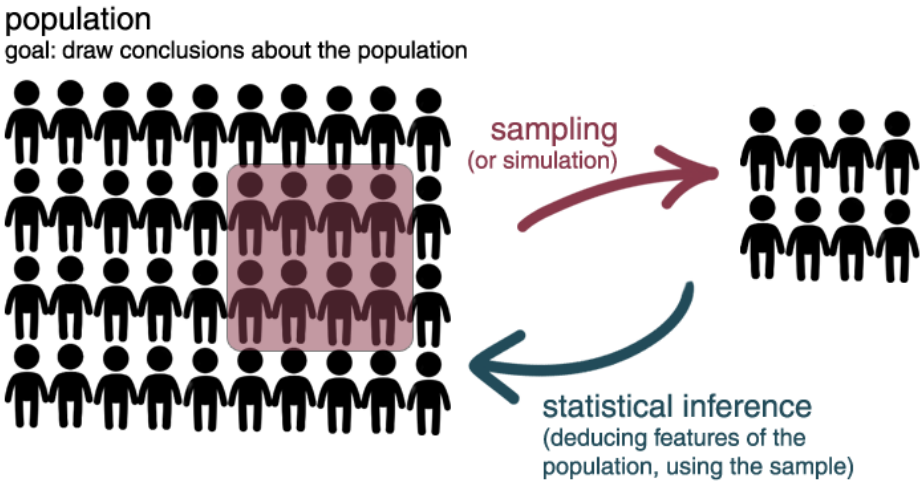
Other Distribution-Related Functions in R 8

The Sampling Distribution of the Mean 10

The Central Limit Theorem 12

Random Assignment and the sample() Function 12

Exercises 14



Conceptual Overview

The Scientific Process

We typically start a study with a scientific question for a population of interest. For example, a researcher may be interested in knowing the average level of educational attainment among the population of migrant farmers in the United States.

For practical reasons, we can virtually *never* collect information about the whole population of interest. In this example, it may be difficult to survey every farmer for their educational level due to cost,

the transient nature of the population, and complex sociopolitical factors.

However, we can use the data in our sample to try to estimate the features of the population, by assuming that the average educational attainment in the sample corresponds in some way to the average in the population. The act of using the sample data to a deduction about the population is **statistical inference**.

Most of the time in this course and (in practice) will be spent on inference. However, to fully understand the background behind inferential techniques, we have to spend a little time in the weeds understanding characteristics of samples to then understand how the information in samples is translated to information about the distribution.

Simulation

The purpose of a **simulation** is to mimic the sampling and data generation process: we attempt to tell the the story of how the data is generated, using mathematical language.

By pretending that we know the true population distribution and taking samples from that known distribution, we can:

1. See if the narrative we believe about the data generation process resembles the real data.
2. Assess how a statistical inference method performs–which we can quantitatively assess because we know the "truth" behind the synthetic data.

Simulation Basics with the Normal Distribution

Let's generate 20 mock samples from a normal distribution with a mean of 8 and a standard deviation of 1, to continue with our example about educational attainment[1].

In brief, a "distribution" is a mathematical depiction of shape describing how likely a variable is to take certain values. If we were to simulate years of educational attainment as a normal distribution, it would imply that **(1)** we think farmers in the population are equally likely to have educational attainment above and below eight years (the normal distribution is symmetrical) and **(2)** that the majority of workers have an educational attainment around eight years ("tails" of the distribution are smaller relative to the middle).

Your intuition may tell you that this isn't the case, and thus that we shouldn't use a normal distribution to model this variable: in Exercise 6, we delve into the real data.

[1] The average level of education completed is often reported as eight years among migrant farmers: National Center for Farmworker Health. "Farmworker Factsheet", Jan 2016. <ncfh.org>

In R, we use the `rnorm()` function to simulate taking samples from a normal distribution. `rnorm()` stands for "random normal" and takes three arguments: `n`, `mean`, and `sd`.

```
# how many samples to simulate
mock_sample_size <- 20

# simulate twenty values from rnorm
simvals <- rnorm(n = mock_sample_size, mean = 8, sd = 1)

# display the first 5 values
simvals[1:5]
```

```
## [1] 7.373546 8.183643 7.164371 9.595281
## [5] 8.329508
```

```
# examine the distribution of the samples in a histogram
# the results are shown in figure 1
hist(simvals,
     main = "", # no title
     xlab = "Normal, N = 20", # x-axis label
     xlim = c(4, 12)) # limits for x-axis
```

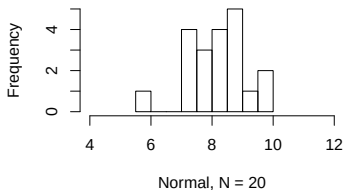


Figure 1: Histogram of our N = 20 samples simulated from a normal distribution with mean 8 and standard deviation of 1.

```
# check the mean
mean(simvals)
```

```
## [1] 8.190524
```

```
# and the standard deviation
sd(simvals)
```

```
## [1] 0.9132537
```

Two interesting things to note here:

1. If you were to run the exact same code used above, you would get a different set of values.
2. The histogram does not look like the normal distribution (bell-curved, symmetric), despite being generated from `rnorm()`. Similarly, the mean and sd are not exactly 8 and 1, respectively.

Let's discuss these two issues in order.

Issue 1: Seeds for Pseudo-Random Number Generators

The `set.seed()` function allows us to set the starting point for the sequence of random numbers generated. This function takes a single integer as its argument (choose your favorite). Setting the seed allows you to recreate results in the future and others to replicate your results.

The use of `rnorm()` in the below code is equivalent to grabbing three samples from a normal distribution with mean 0 and standard deviation of 2.

```
# run the random function with a seed = 999
set.seed(seed = 999)
rnorm(n = 3, mean = 0, sd = 2)
```

```
## [1] -0.5634803 -2.6251193  1.5903680
```

```
# re-run -- get different numbers than first time
rnorm(n = 3, mean = 0, sd = 2)
```

```
## [1]  0.5401410 -0.5546128 -1.1320475
```

```
# reset seed = 999, should get same as first time
set.seed(seed = 999)
rnorm(n = 3, mean = 0, sd = 2)
```

```
## [1] -0.5634803 -2.6251193  1.5903680
```

THE TAKEAWAY: For others to reproduce your work, set the seed before running random number generation.

Issue 2: Variation at Low Sample Sizes

At low sample sizes, there will be higher variability in the distribution of your sample. (The **sampling distribution** is distinct from that of the population.) Samples can look different from the population distribution that is being sampled from (and thus, look different from other samples) by chance and chance alone.

However, as the sample size increases, the sampling variability decreases (see **Figure 2** on next page).

Conceptually, we can think of this in terms of the extreme case in which the sample size grows to nearly be the size of the population: as the sample size increases, the sample resembles the population more and more.

THE TAKEAWAY: The sample itself has a distribution. One sample from the population may look different from another sample due to random sampling error (chance and chance alone).

```r
# histogram for N = 100
set.seed(555)
hist(x = rnorm(n = 100, mean = 8, sd = 1),
     main = "",
     xlab = "Normal, N = 100",
     xlim = c(4, 12))
```

```r
# histogram for n = 1,000
set.seed(555)
par(cex = 0.8)
hist(x = rnorm(n = 1000, mean = 8, sd = 1),
     main = "",
     xlab = "Normal, N = 1000",
     xlim = c(4, 12))
```

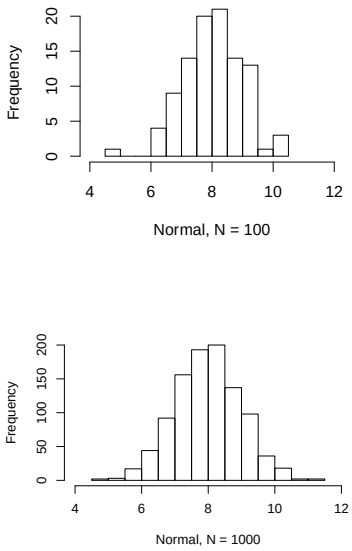


Figure 2: With increasing sample size (going from 20, to 100, to 1000), the distribution of our mock normal sample generally will look more like the population's distribution.

Non-Normal Distributions

In base R (i.e., without loading any extra packages), there are other "random" functions corresponding to around 20 probability distributions. We'll cover the four in **Table 1** for now.

Table 1: Four examples of random variable generators in R.

Distribution	R Function	Arguments
binomial	`rbinom()`	`n`, `size`, `prob`
normal	`rnorm()`	`n`, `mean`, `sd`
poisson	`rpois()`	`n`, `lambda`
uniform	`runif()`	`n`, `min`, `max`

Note that each distribution has an `n` argument that specifies how many samples to simulate. The other arguments correspond to the population **parameters**–features of each distribution that entirely specify the distribution's shape[2].

For the normal distribution, the parameters are the (1) mean and (2) standard deviation because these two features can be used to specify everything about the shape of a normal distribution.

[2] Because they tell us everything about the shape of the distribution, parameters are used for notational short-hand: e.g., "X is normally distributed with mean of 8 and standard deviation of 1" is equivalent to $X \sim \text{Normal}(8, 1)$, where "$\sim$" means "distributed as".

In contrast, the shape of the **uniform distribution** is determined entirely by its (1) minimum and (2) maximum. A variable that follows the uniform distribution displayed in **Figure 3** would have a 0.10 probability of being between 0 and 0.1, as well as a 0.10 probability of being between 0.9 to 1.0. The probability of being between 0.25 and 0.45 is double the "width" of our other examples and thus has 0.20 probability.

Because the probability of taking any value in range is equally likely, `min` and the `max` are the only parameters needed to specify the entire shape of uniform distribution.

```
# for reproducibility
set.seed(100)

# simulate 10,000 samples from a uniform distribution
# that varies between 0 and 1
uniform_example <- runif(n = 10000, min = 0, max = 1)

# print first 8 observations in vector
uniform_example[1:8]
```

```
## [1] 0.30776611 0.25767250 0.55232243
## [4] 0.05638315 0.46854928 0.48377074
## [7] 0.81240262 0.37032054
```

```
# plot shape of distribution
par(cex = 0.8)
hist(uniform_example, main = "",
     xlab = "Uniform(0, 1) Sample")
```

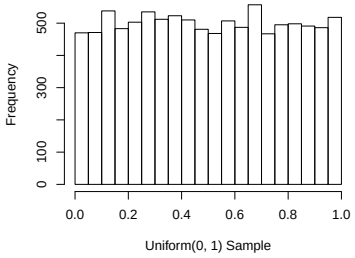


Figure 3: Mock data (N = 10,000) simulated from a uniform distribution varying between min = 0 and max = 1.

Note that the normal distribution and uniform distribution are **continuous** variables, meaning they can take on decimal values. However, we might instead want to simulate data from a **discrete** distribution. Discrete distributions may represent variables that take on only categorical values (e.g., "smoker", "non-smoker", "former smoker") or counts (e.g., number of hospitalizations per day can take values of 0, 1, 2, 3).

The **binomial distribution** represents the number of "successes" of a binary outcome. It can be visualized as an experiment where we flip a coin repeatedly and count the number of times we rolled heads.

The following code is equivalent to a single experiment flipping a fair coin (`prob = 0.5`; i.e., 50% chances of heads, 50% chances of tails) ten times (`size = 10`), then counting the total number of times we rolled heads (R's output / the binomial variable). The binomial

variable is discrete because it can take values of 0, 1, 2, $\cdots$, number of coin-flips per experiment.[3]

[3] Note that R calls the number of coin-flips or trials "size". However, you'll see the number of trials referred to as "n" most of the time in theory texts! It's tricky, but try to keep the sample size argument (`n =`) and the number of trials argument (`size =`) separate.

```
# binomial distribution example
set.seed(111)
rbinom(n = 1, size = 10, prob = 0.5)
```

```
## [1] 5
```

Maybe we repeat this experiment involving ten coin-flips (`size = 10`) two-hundred times (`n = 200`), because we don't have any other hobbies. A visual summary of the resulting mock data is shown in **Figure 4**.

```
# example to highlight difference between "n" and "size"
set.seed(111)
par(cex = 0.8)
binom_example <- rbinom(n = 200, size = 10, prob = 0.5)

# note: setting "breaks" argument to change x-axis defaults,
# in order to explicitly show that this is *discrete*
hist(binom_example, main = "",
     xlab = "Binomial(n = 10, p = 0.5)",
     breaks = seq(-0.5, 10.5, 0.5))
```

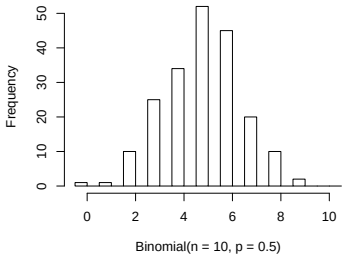
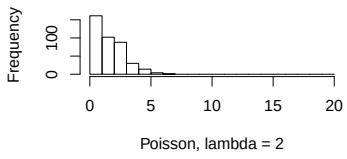


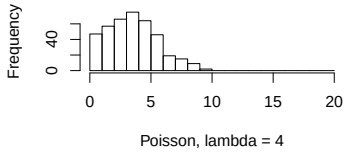
Figure 4: Simulating n = 200 experiments in which we flip a fair coin twenty times (prob = 0.5, size = 10), and count the total number of heads. Note the discrete nature of this distribution.

The **Poisson distribution** is another common discrete distribution that can take values of 0, 1, $\cdots$, ∞. The Poisson distribution has an interesting feature: its mean–denoted as lambda (λ)–is equal to its variance. The Poisson distribution tends to be right-skewed at low values of λ, and is used very frequently to model count data.

```
# figure 5-top
set.seed(303)
hist(rpois(n = 400, lambda = 2),
     main = "", xlab = "Poisson, lambda = 2",
     breaks = seq(0, 20, 1))
```




```
# figure 5-mid
hist(rpois(n = 400, lambda = 4),
     main = "", xlab = "Poisson, lambda = 4",
     breaks = seq(0, 20, 1))
```



```r
# figure 5-bot
hist(rpois(n = 400, lambda = 10),
     main = "", xlab = "Poisson, lambda = 10",
     breaks = seq(0, 20, 1))
```

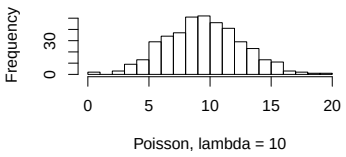


Figure 5: Depictions of Poisson distributions with different lambda values.

Other Distribution-Related Functions in R

We've used the "r" set of functions like `rnorm()` to simulate sampling from a distribution of interest. There are three other functions that concern themselves with properties of distributions, corresponding to `d` (density), `p` (cumulative distribution function), and `q` (quantile).

d-Functions: Probability Density Function (PDF)

The probability density function for a continuous variable X is typically denoted with $f(x)$, where $P(a \leq X \leq b) = \int_a^b f(x)dx$.

Think of a histogram: histograms are essentially the way we try to visualize the density function with our sample. Just as the heights of histogram bins contain information about the proportion of subjects in a sample between two given values, the density represents this for a theoretical distribution.

```r
# density function - dnorm()
x_values <- seq(4, 12, 0.1)
density_of_normal <- dnorm(x = x_values,
                           mean = 8, sd = 1)
plot(x_values, density_of_normal,
     xlab = "X-Value", ylab = "Density", type = "l")
```

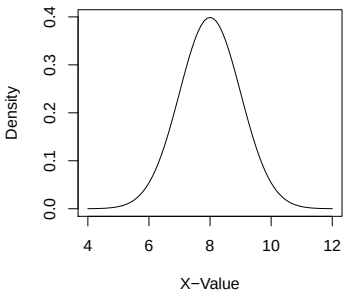


Figure 6: Density plot for a normal distribution.

Discrete variables have an analogous "density" that is a bit easier to interpret. It is known as the probability mass function $p(x)$ for which $p(x) = P(X = x)$. The probability of being between two given values a and b is therefore $P(a \leq X \leq b) = \Sigma_{i=a}^{b} p(x)$.

Make sure you understand the following output from `dbinom()`:

```r
# dbinom() example 1
dbinom(x = 0, size = 2, prob = 0.5) +
  dbinom(x = 1, size = 2, prob = 0.5) +
  dbinom(x = 2, size = 2, prob = 0.5)
```

```
## [1] 1
```

```r
# dbinom() example 2
dbinom(0.4, size = 2, prob = 0.5)
```

```
## Warning in dbinom(0.4, size = 2, prob = 0.5):
## non-integer x = 0.400000
```

```
## [1] 0
```

p-Functions: Cumulative Distribution Function (CDF)

For a random variable X, the cumulative distribution function (CDF) describes the following probability: $P(X \leq x)$. As it is a probability, the CDF ranges between 0 and 1.

```r
# cumulative probability (distribution function)
x_values <- seq(4, 12, 0.1)
density_of_normal <- pnorm(q = x_values,
                           mean = 8, sd = 1)
plot(x_values, density_of_normal,
     xlab = "X-Value", ylab = "CDF", type = "l")
```

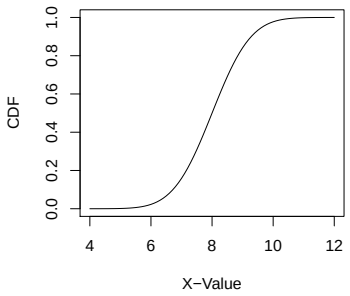


Figure 7: Plot of the cumulative distribution function (CDF) for a normal distribution.

```r
# example 1 of using cdf
pnorm(q = 7, mean = 8, sd = 1)
```

```
## [1] 0.1586553
```

Interpretation: $P(X \leq 7) \approx 0.159$. A given worker randomly sampled from this population would approximately have a 15.9% probability of reaching an educational level of 7 years or less.

```r
# example 2 of using cdf
pbinom(q = 17, size = 20, prob = 0.5) -
  pbinom(q = 15, size = 20, prob = 0.5)
```

```
## [1] 0.005707741
```

Interpretation: $P(X = 16 \text{ or } 17) = P(16 \leq X \leq 17) = P(X \leq 17) - P(X \leq 15) \approx 0.0057$. If we were to flip a fair coin twenty times, the probability that we observe 16 or 17 heads in the course of this process is about 0.0057, or 0.6%.

q-Functions: Quantiles

```r
qnorm(p = 0.95, mean = 8, sd = 1)
```

```
## [1] 9.644854
```

Interpretation: Continuing with our educational attainment example, if educational attainment were distributed as a $Normal(\mu = 8, \sigma = 1)$, then the 95th percentile would be 9.6 years of education. Equivalently, someone with 9.6 years of schooling would have more schooling than 95% of the population.

The Sampling Distribution of the Mean

We mentioned that the sample has its own distribution, with random samples potentially differing from one another by chance and chance alone ("sampling error").

As a result, the sample mean (denoted as $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ for sample observations $x_1, x_2, \cdots, x_n$) itself varies from sample to sample and thus itself has a distribution. This distribution is known as the sampling distribution of the mean.

To see this, we'll use a for-loop[4] to generate a sample of size n = 20, five-hundred times (i.e., 500 simulation iterations).

[4] For loops are frowned upon in R, but the syntax is relatively easy to parse. We'll get to better programming practice later in the course, as you become more familiar with R.

```r
# simulation parameters
number_of_sims <- 500
sample_size <- 20
set.seed(888)

# empty vector to store sample means
vector_of_sample_means <- rep(-9, number_of_sims)

# for i = 1, 2, ..., 500
# calculate the sample mean, and store in vector
for (i in 1:number_of_sims) {
  vector_of_sample_means[i] <-
    mean( rnorm(n = sample_size, mean = 8, sd = 1) )
}

# check to make sure that our simulation worked
# print the first five entires
vector_of_sample_means[1:5]
```

```
## [1] 7.752596 8.253554 7.946046 7.850417
## [5] 8.000252
```

```r
# are the first five entries equal to -9?
vector_of_sample_means[1:5] == -9
```

```
## [1] FALSE FALSE FALSE FALSE FALSE
```

```r
# this should return zero if there are no -9s in whole vector
# in R, TRUE values are coerced into 1s, FALSE into 0s
sum(vector_of_sample_means == -9)
```

```
## [1] 0
```

```r
# plot the sampling distribution of the mean
hist(vector_of_sample_means, xlim=c(7,9),
     xlab = "Values of Sample Mean",
     main = "")
```

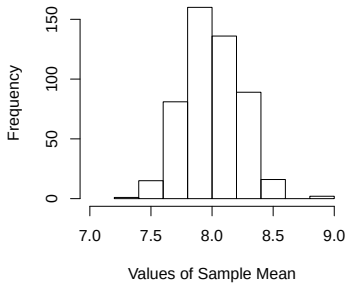


Figure 8: Sampling distribution of the mean, for samples with n = 20 from a population that is normally distributed with mean 8 and standard deviation 1.

As n increases, the sampling distribution of the mean changes. Namely, when the population is distributed as a normal with mean μ and standard deviation σ, denoted as $X \sim Normal(\mu, \sigma)$, the sample mean is $\bar{X} \sim Normal(\mu, \sigma/n)$.

That is, the variability in sample means that you'll observe from sample-to-sample decreases as the sample size increases. (This is consistent with our initial exploration of this phenomenon through simulation in the "Variation at Low Sample Sizes" section.)

Similarly, note that the x-axis is between 7 and 9 whereas a histogram of the sample values themselves was presented with x-axis limits of 4 to 12 (see **Figure 2**). The variability in the sample mean is less than the variability in any given sample (in fact, $\frac{1}{n}$-fold less). We reiterate that the "distribution of a sample" is different than the "distribution of a sample's mean".

As an aside, the code used above, `sum(vector_of_sample_means == -9)`, exploits R's *coercion* principles. Coercion is an R phenomenon we mentioned in R Lab 0 as something that is often convenient (but also scary, because it R doesn't yield a warning message). The command `vector_of_sample_means == -9` returns a logical vector (`TRUE`s and `FALSE`s) that–once we ask R to `sum()` the items in the vector–are converted to the numerical double type (1s and 0s, respectively).[5]

[5] To more explicitly see this phenomenon, run the code `T + T + F + T + F + T` in your R console.

The Central Limit Theorem

The Central Limit Theorem (CLT) is a surprising result that applies to *any* population distribution with a finite mean (that is, when $E(X) = \mu$ exists). At large sample sizes ("asymptotically"), the sampling distribution of the mean converges in distribution to a normal distribution.

Or, more mathematically, for a population with mean μ and standard deviation σ, the CLT says that as $n \to \infty$, $\bar{X} \to N(\mu, \frac{\sigma}{n})$.[6]

A common point of confusion: The CLT is a statement about the sampling distribution of the mean ($\bar{X}$), not the distribution of the sample (X-values). The distribution of the sample becomes more similar to the distribution of the population as n increases.

The CLT is *extremely* useful because at high sample size we now can apply techniques developed for statistical inference on normally distributed populations–even if the population distribution is extremely skewed or we don't know the underlying distribution.

What sample size is "large"? Large is heuristically defined as $n \geq 30$. This rule-of-thumb is generally well-known, but due to compuing advances, there might be better alternatives if your worried that your sample size is too small[7].

[6] This distribution strongly resembles the sampling distribution of the mean that we discussed in the previous section. The distinction is that $\bar{X} \sim N(\mu, \frac{\sigma}{n})$ holds true regardless of sample size for a normally distributed population. In contrast, this statement is only true at large sample sizes for non-normal populations.

[7] Namely, non-parametric methods or bootstrap / permutation methods. BIOS6611 students will read Hesterberg, Tim. "It's Time To Retire the 'n >= 30' rule." *Proceedings of the Joint Statistical Meetings*, Alexandria, VA (2008) let in the semester.

Random Assignment and the sample() Function

Let's step away from theory for a moment. Although there aren't many variables that are uniformly distributed in the "real world," the `runif()` function is still extremely useful in simulations and study design. Namely, we can use it to select *approximately* 5 of the 50 states (stored in `state.name`) using the following code.

This code works by creating a fifty numbers between 0 and 1. We then check if each value in this numerical vector is < 0.1, resulting in a vector of `TRUE` and `FALSE` values. This logical vector is saved as `select_with_uniform`. Then, we use the brackets to indicate that we want to subset the `state.names` vector.[8]

[8] Here's a simpler example:

```
sentence <- c("dogs", "are",
"super", "neat!")
last_word_only <- c(F, F, F, T)
sentence[last_word_only]
```

```
# select approximately 5 with runif()
set.seed(123)
select_with_uniform <- runif(n = 50) < 0.1
state.name[select_with_uniform]
```

```
## [1] "Colorado"  "Louisiana" "Ohio"
```

Alternatively, we could imagine flipping a single coin for every state, with a heads indicating that we should select the state. To get approximately five, let's use a really "unfair" coin, with a 0.1 probability of heads and a 0.9 probability of tails. (This code works in a similar way as above.)

```
# select approximately 5 with rbinom()
set.seed(321)
select_with_binom <-
  rbinom(n = 50, size = 1, prob = 0.1) == 1
state.name[select_with_binom]
```

```
## [1] "Alabama"       "Alaska"
## [3] "Michigan"      "Minnesota"
## [5] "Nebraska"      "North Dakota"
## [7] "West Virginia"
```

Alternatively, we could use the `sample()` function to select exactly 5 of the 50 states. Note that we also `set.seed()` when using this function for reproducibility, as another random operation is happening.

```
# select exactly 5 with sample()
set.seed(123321)
sample(x = state.name, # features we want to sample from
       size = 5, # sample size
       replace = F) # don't do it with replacement
```

```
## [1] "South Carolina" "Missouri"
## [3] "Wyoming"        "Vermont"
## [5] "North Carolina"
```

Although it may seem farfetched in this example, there are many situations where we may want an *approximate* proportion of our sample to have a feature versus an *exact* proportion.

For example, we may be trying to simulate mock data resembling a cohort study where we recruit 100 patients, then follow-up later to see if they have developed a disease that typically occurs in half of patients. Because we're not guaranteed to observe 50 of 100 patients get the disease, it's better to simulate their disease status with `rbinom(n = 100, size = 1, prob = 0.5)` than simply `sample()` 50 of the 100 patients.

(Real data is random, so we generally want our mock data have a bit of randomness as well.)

See Example 5 and 6 to put these applications of random functions and `sample()` into practice.

Exercises

Excercise 1: Getting Started

1a. Reproducibly simulate a sample of 10,000 from each of the following distributions.

- Normal(μ=125, σ=8)
- Poisson(λ=1.5)
- Binomial(n=5, p=0.15)

1b. Determine the theoretical mean and standard deviation for each distribution and verify that the generated numbers have approximately the correct mean and standard deviation.

1c. Create a histogram and boxplot depicting each of the mock samples.

Exercise 2: Sampling Distributions for Other Statistics

2a. For a population that is normally distributed with mean 40 and standard deviation 10, generate histograms showing the sampling distribution of the mean, median, and variance. Use 1,000 simulation iterations and a sample size of n = 10.

2b. Based on theory, what is the distribution of the sample mean and sample median in this case (e.g., uniform, exponential, gamma, normal, etc. distributed)?

2c. When the underlying population is normally distributed, the sampling distribution of $(n-1)s^2/\sigma^2$ is chi-squared with n - 1 degrees of freedom ($n-1$ = 9 in this case).

Use the base R `plot()` and `dchisq()` function to plot the *theoretical* sampling distribution of the variance (i.e., instead of a histogram, either predict the density and present either density curve or set of points which could be connected together to form an approximate curve). Compare it to your result in 2a (hint: for ease of comparison,

you might want to multiply your sample variance vector from 2a by a factor of $9/10^2$).

Exercise 3: The Central Limit Theorem

2a. Generate and save a vector containing 500 sample means (i.e., five-hundred simulation iterations) of sample size 10 from a $Binomial(n = 1, p = 0.15)$ population (recall, in `rbinom()` size=1 and n=10).

2b. Repeat for sample sizes of n = 20, n = 30, n = 40, and n = 50.

2c. Calculate the mean and standard deviation associated with each of the five sets of $\bar{x}$ values.

2d. Create histograms of the sampling distribution of the mean, for each sample size n.

2e. Is there a value of the sample size n (i.e. 10, 20, 30, 40, or 50) where the distributions begin to look normal?

Exercise 4: The CLT and the Cauchy Distribution

The Cauchy distribution is famous in part because its mean and variance is undefined. As a result, the CLT does not apply.

Repeat the simulation exercises in Exercise 3, using sample size values of n = 10, n = 50, n = 100, and n = 1000 substituting the random function `rcauchy()` and no other arguments besides the sample size `n` parameter[9].

How do the trends you observe differ relative to the Exercise 3?

[9] If you run `?rcauchy` to pull up the documentation on this function, you'll see that `rcauchy` has the `location = 0` and `scale = 1` arguments by default. That is, we'll use a location parameter of 0 and a scale parameter of 1 if we don't specify one explicitly when we call the function.

(These last two exercise sets are long problems for practicing your random assignment & data frame manipulation skills.)

Exercise 5: Study Design Sprints

5a. You are auditing two year-old patients' immunization records at a rural health clinic. There are 250 two year-olds with medical records numbered from 9001 to 9250. **(i)** Generate a vector of record IDs, then **(ii)** reproducibly sample 30 records for your audit.

5b. `names.txt` on the Canvas site contains the names of the students of a sixth grade physical education class. To prevent the psychological trauma associated with picking teams, randomly assign students to either the Red Team or Blue Team.

Using the `read.table()` function, **(i)** read the names into in R as a data frame called "`names`" and **(ii)** reproducibly assign exactly half of the students to each team, displaying the student's assignment in a column called `team` in the `names` data frame. Then, **(iii)** display the first 10 records of your data frame and **(iv)** run the code `table(names$team)`.

5c. Create a data frame representing 10 subjects with: **(i)** a column named `id`, valued 1, 2, $\cdots$, 10 and **(ii)** a column called `age` that is uniformly distributed between 20 and 60.

Split your data frame two different data frame, one called `older` the other called `younger`, based on whether the subject is $\geq$ 45 years old.

5d. Create a data frame named `assignments` representing 100 subjects with three columns respectively named `id` (1 through 100), `dietary_intervention`, and `pharma_intervention` using the following criteria:

Use the `runif()` function to make it so *approximately* 30% of the patients have a value of "D" for `dietary_intervention` and the others say "ND". Then, use `rbinom()` function to make *approximately* 30% of the patients have a value of "P" for `pharma_intervention` and the remainder "NP".

Finally, run the following command:

```
table(assignments$dietary_intervention,
      assignments$pharma_intervention)
```

5e. Create a data frame reflecting the following:

- There are 5 hospitals involved in a clinical trial (numbered 1 to 5).
- Each of the hospitals has 40 patients.
- Each patient has a 0.7 probability of improving.

Create a **(i)** data frame reflecting the above specification and **(ii)** use a for-loop to calculate the number of patients that improved for each hospital. Then, **(iii)** calculate the mean number of patients that improved across all hospitals.

Exercise 6: NAWS Farm Worker Survey Simulation

6a. Read `NAWS2014.csv` from the Canvas site into R with the name `NAWS`. This is survey data publicly available from the Department of Labor[10].

[10] See the "Public Data" tab at `doleta.gov/naws`. The data was filtered to contain only surveys from 2014 and remove subjects without known educational levels.

6b. Plot a histogram of the `A09` column, which asks how many years of school migrant farmers have completed (the same population referenced earlier in R Lab 1). Label the title and axes appropriately.

6c. As mentioned earlier in this document, it is commonly reported that the average educational attainment among migrant farmers is 8 years.In your opinion, does reporting just this average tell the whole story? Why or why not? (Feel free to speculate on why you think the histogram has its unique shape.)

6d. Create a new column in the dataset called `category_edu` with four possible values: "00 - 05", "06 - 08", "09 - 11", or "12+".

Here's some example code to make the first two columns:

```
NAWS$category_edu <- "00 - 05"
NAWS[NAWS$A09 >= 6 & NAWS$A09 <= 8, ]$category_edu <- "06 - 08"
```

6e. Use the command `table(NAWS$category_edu)` to return a vector containing counts of subjects in each category. Combine the `table()` command with the division operator (`/`) to summarize the proportion of subjects in each category. You should get 0.248, 0.289, 0.212, and 0.251 if you've performed the calculation correctly.

6f. Reproducibly generate a 800-subject data frame named `mockdata` by following these steps:

- Run the following command:

```
mockdata <- data.frame(subject = 1:800,
                       random = runif(n = 800))
```

- Use the `mockdata$random` column together with the proportions found in 5d to randomly assign the subjects to "00 - 05", "06 - 08", "09 - 12", or "12+". That is, if `random` is in the range 0 and 0.248, assign each the subject to "00 - 05". If `random` is in the range 0.248 to 0.248 + 0.289, assign the subject to "06 - 08". (And so on.) Save these classifications to a column called `educ_cat`.
- For subjects in the respective `educ_cat` categories, assign a value from a Uniform(0, 6), Uniform(6, 9), Uniform(9, 12), and Uniform(12, 17). Save these assignments as `educ_years`.

- Assign every subject a randomly sampled value from a $Binomial(1, 0.8)$ distribution and save this as the column `edu_stop_yn`.
- For subjects in the `6 - 8` category and `edu_stop_yn == 1`, set `educ_years` to 6. Similarly, assign `9` to "09 - 11" and `12` to "12+" category.
- Transform the `educ_years` variable into counts by running the following code, where as.integer rounds down to the nearest integer (e.g., 5.7 becomes 5):

```
mockdata$educ_years <- as.integer(mockdata$educ_years)
```

6g. Finally, plot a histogram of `mockdata$educ_years`. Does your mock data set look similar to the data it is based off of?