Lab 3

Kevin Buck

2/1/2023

In Class

Today we are going to be using the human genes dataset from Chapter 4 of *Analysis of Biological Data* (Whitlock & Schluter) to explore how sampling effort influences uncertainty. As in the book, we are considering these genes to be a *population* of genes rather than a sample. These data represent the largest known transcript of each gene in the 93rd release of the **Human Genome Project**.

First, download the gene length data file ("chap04e1HumanGeneLengthsLongestTranscript.csv") from Canvas. We will again be using the **tidyverse** package, so load it using the **library()** function and read in the gene length data using **read_csv()**. Save it as an object named "gene". Make sure the data are in the same folder as your Rmd document to avoid directory issues.

```
setwd("~/Desktop/Biostats/Week03")
library(tidyverse) #importing tidyverse
gene <- read.csv(file="Gene.csv") #reading in the gene data from a csv file
```

Use **glimpse()** to take a look at the data. **How many genes are in this data set?**

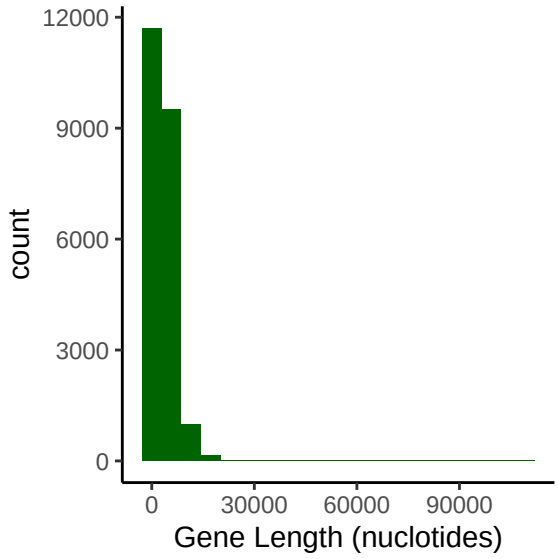
There are 22,385 rows in the data, which means there are 22,385 genes.

```
glimpse(gene) #peeking at the data
```

```
## Rows: 22,385
## Columns: 4
## $ gene        <chr> "ENSG00000000003.14", "ENSG00000000005.5", "ENSG0000000041~
## $ size        <int> 3796, 1339, 1161, 6364, 4355, 2729, 4127, 2356, 3813, 3811~
## $ name        <chr> "TSPAN6", "TNMD", "DPM1", "SCYL3", "C1orf112", "FGR", "CFH~
## $ description <chr> "tetraspanin", "tenomodulin", "dolichyl-phosphate", "SCY1"~
```

Plot a histogram of gene size using the skills you learned from last week's lab. Feel free to reference your code from last week.

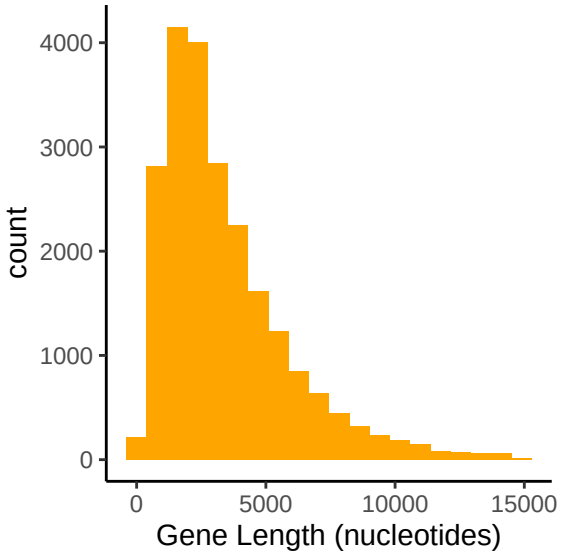
```
gene %>% #piping in the gene data
  ggplot(aes(size)) + #adding gene length as the response var to plot
  geom_histogram(fill="darkgreen",bins=20)+ #creating a green histogram of the data
  xlab("Gene Length (nuclotides)")+ #labeling the axis
  theme_classic() #making the background of the fig white
```



You'll notice that there is a long tail in the data (right skew), because there are a few genes that are very long. To recreate the graphics in Chapter 4 of W&S (Fig 4.1-1), we will want to reduce our data set to genes that have a length less than 15,000 nucleotides.

Let's use the tidyverse function **filter()** to create a new data set that only includes genes with lengths less than 15,000 nucleotides. To create an object from a piped data set with a function, we decide on a name and assign it to the "top" of the pipe. We'll name this new object "gene_short". Then plot a histogram using the object "gene_short".

```
#creating a new object from gene data piped into a filter function removing rows where size exceeds 150
gene_sort <- gene %>%
  filter(size < 15000)
#making histogram of the gene length data
gene_sort %>% #piping in the truncated gene data
  ggplot(aes(size)) + #adding gene length as the response var to plot
  geom_histogram(fill="orange",bins=20)+ #creating an orange histogram of the data
  xlab("Gene Length (nucleotides)")+ #labeling the axis
  theme_classic() #making the background of the fig white
```



We can calculate summary statistics in tidyverse using the function **summarize()**. Calculate the mean and median of the full "gene" dataset.

The mean is 3511.457, the median is 2744.

```
#calculating mean and median gene size from gene data
summarize(.data=gene,mean=mean(size),median=median(size))
```

```
##       mean median
## 1 3511.457   2744
```

Apply your skills

Let's examine how the mean and median change when we remove genes that are greater than 15,000 nucleotides from the dataset. Try that here: first filter the "gene" dataset for lengths less than 15,000 nucleotides, then put put another %>%, press ENTER, and use the summarize function on the filtered dataset to calculate the mean and median.

The mean of the sorted data is 3407.496, and the median is 2727.5

```
 #creating new variable for stats w sorted data, sending gene data into start of pipe
gene_sorted_stats <- gene %>%
  #filtering data to be only size < 15000
  filter(size < 15000) %>%
  #calculating mean and median of the filtered data to be stored
  summarize(mean=mean(size),median=median(size))
print(gene_sorted_stats)
```

```
##       mean median
## 1 3407.496 2727.5
```

Assignment

Don't forget to answer all parts of each question and to comment your code thoroughly!

Question 1:

Which of the calculated summary statistics (mean or median) changed more when you filtered the "gene" dataset to consist of genes with lengths less than 15,000 nucleotides ("gene_short")? Explain why you think this is.

Answer 1: The mean changed more than the median because by eliminating genes with lengths more than 15000 nucleotides, the data became more skewed left with the right lying (larger) outlier removed. The mean changed by about -104, while the median only changed by -16.5.

Question 2: How does sampling effort influence the summary statistics of the gene data set?

a. Calculate the mean, standard deviation, and standard error for three different tibbles using genes that vary in the number of sampled observations (10, 100, 1000).

- To do this, first randomly sample "gene" using the **sample_n()** function in a pipe workflow. The only argument the **sample_n()** function needs is the number of samples. Then pipe to **summarize()**.
- Within **summarize()**, first calculate the mean and standard deviation (sd). Then calculate the number of observations (n). Finally calculate the standard error as se = sd / **sqrt(**n**)**. You can use the file "SummaryStats_BasicFunctions.R" to find the appropriate functions.

- You'll notice the function **set.seed()** specified at the beginning in this and some of the following code chunks. The "seed" is a random number (in this case 1256) that ensures that R is starting from the same starting point to randomly sample. This means that you should get the same values each time you use **sample_n()**, even though it is still a random sampling process.
- Make sure to run the full code chunk using the green arrow at the top right of the chunk before interpreting output to ensure that the "seed" was set before the random samples were taken.

Answer 2a: For 10 random data points, mean = 3913.8, sd = 7529.799, and se = 2381.131 For 100 random data points, mean = 3366.59, sd =3168.298, and se = 316.8298 For 1000 random data points, mean = 3504.069, sd =2682.986, and se= 84.84346

```
# Makes sure the random sample is the same each time you run your code.
# Make sure you run this line of code!
set.seed(23) #making sure I am using same random sample each time

#stats for sample size = 10
gene %>% #piping gene data in
  sample_n(size=10) %>% #sampling 10 random data points
  #calculating mean, sd, se from this sampled data
  summarize(mean=mean(size),sd=sd(size),obs=length(size),se=sd(size)/sqrt(obs))
```

```
##     mean       sd obs       se
## 1 3913.8 7529.799  10 2381.131
```

```
#stats for sample size = 100
gene %>% #piping gene data in
  sample_n(size=100) %>% #sampling 100 random data points
  #calculating mean, sd, se from this sampled data
  summarize(mean=mean(size),sd=sd(size),obs=length(size),se=sd(size)/sqrt(obs))
```

```
##      mean       sd obs       se
## 1 3366.59 3168.298 100 316.8298
```

```
#stats for sample size = 1000
gene %>% #piping gene data in
  sample_n(size=1000) %>% #sampling 1000 random data points
  #calculating mean, sd, se from this sampled data
  summarize(mean=mean(size),sd=sd(size),obs=length(size),se=sd(size)/sqrt(obs))
```

```
##       mean       sd  obs       se
## 1 3504.069 2682.986 1000 84.84346
```

b. How do the mean, standard deviation, and standard error change with different sample sizes? Based on what you know from Chapter 4, do you think that the changes for each of the summary statistics (mean, standard deviation, standard error) with sample size are due to chance alone (*i.e.* due to random sampling)? Why?

Answer 2b:

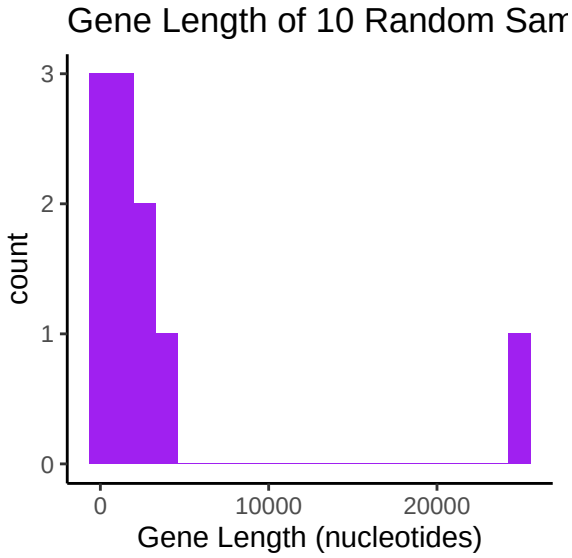
Basically, when sample size (and statistical power) increases, the standard deviation and standard error decrease, as the data is more representative of the real life parameter because it represents more and more of the true variance of values in the real population (sampling distribution increases). There appears to be no real change in mean that scales with size. This makes sense, as the variation in the mean should be coming from randomness/stochasticity and not any trend (as mean should be about the same in any representative sample).

c. Plot a histogram of gene size using just 10 random samples and compare with a histogram of gene size with 1000 random samples.

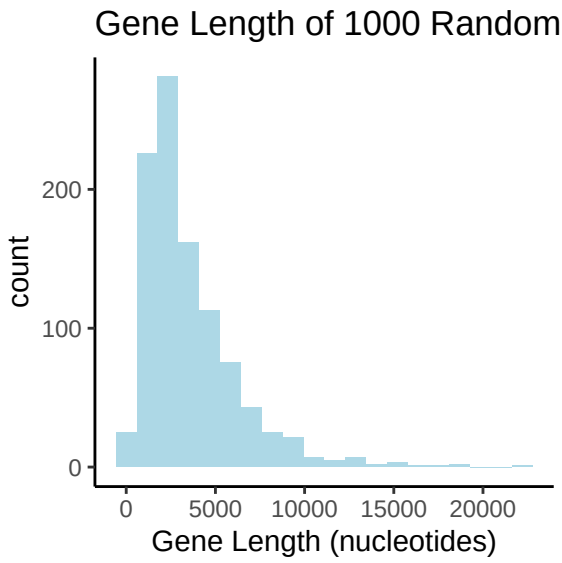
- Note: you can pipe (%>%) directly from the **sample_n()** function to your **ggplot()** function.
- Add a vertical line to your graph to denote the mean of the sampled values using the add-on **geom_vline()**. **geom_vline()** needs its own **aes()**, within which, only the argument "xintercept" must be specified. Use the **mean()** function to take the mean of gene size ("size") in your data set as the input for xintercept.
- Make sure to run the full code chunk using the green arrow at the top right of the chunk before interpreting output to ensure that the "seed" was set before the random samples were taken.

```
# Makes sure the random sample is the same each time you run your code.
# Make sure you run this line of code!
set.seed(23)

#making histogram of the 10 random samples data
gene %>%
  sample_n(size=10) %>%
  ggplot(aes(size)) + #adding gene length as the response var to plot
  geom_histogram(fill="purple",bins=20)+ #creating a purple histogram of the data
  xlab("Gene Length (nucleotides)")+ #labeling the axis
  ggtitle("Gene Length of 10 Random Samples") + #adding title
  theme_classic() #making the background of the fig white
```




```
#making histogram of the 1000 random samples data
gene %>%
  sample_n(size=1000) %>%
  ggplot(aes(size)) + #adding gene length as the response var to plot
  geom_histogram(fill="lightblue",bins=20)+ #creating a light blue histogram of the data
  xlab("Gene Length (nucleotides)")+ #labeling the axis
  ggtitle("Gene Length of 1000 Random Samples") + #adding title
  theme_classic() #making the background of the fig white
```



Are 10 samples adequate to get a good estimate of the population mean gene size?

Answer 2c:

10 samples are too little to get a good estimate of the population mean gene size parameter sought out here. The histogram of the 10 samples here is skewed and includes an outlier data point. Larger samples mean more statistical power and a better estimate.

Question 3: How does sampling effort influence the uncertainty of an estimate?

Following methods in Chapter 4.1 (pg. 101-102), we took 10,000 sets of random samples of the gene data *each* for three different scenarios that varied the sample size (n = 20, 100, 500). For each set of random samples, we calculated the sample mean of the "size" variable and stored it in a tibble.

a. Download and read in this data file into R ("iterative_samples.csv") and save as an object.
   - Use **glimpse()** to take a look at the data to understand how they are structured.
   - Create a faceted histogram graph (*i.e.* three histograms next to each other). To do this, write code for a histogram that shows the distribution of sample means for the entire data set and at the end, include the add-on **facet_wrap(vars(n_samples))**.

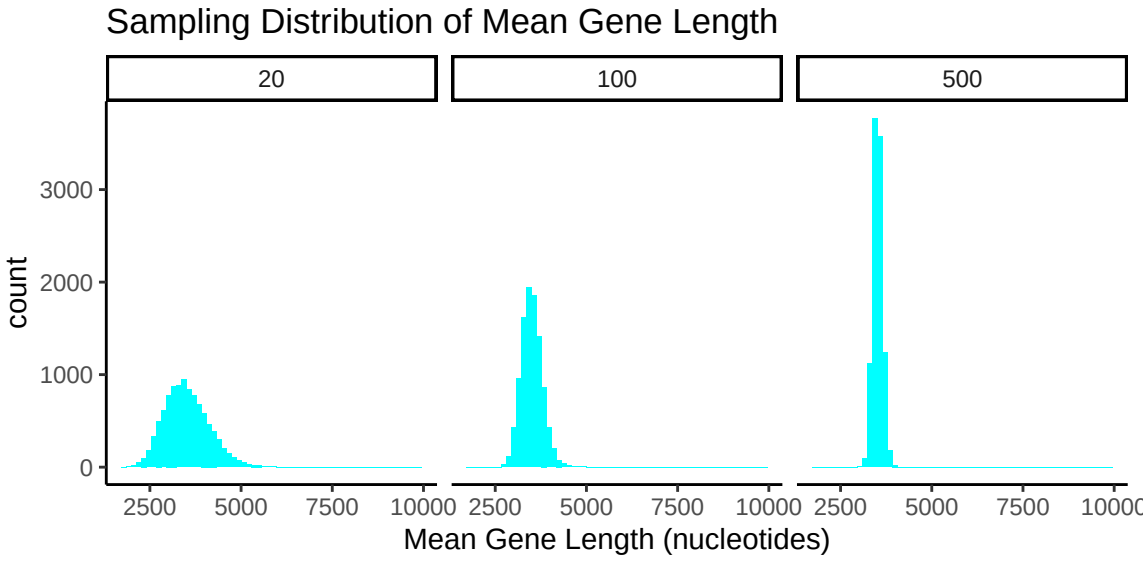
Answer 3a:

See code below!

```
iterative <- read.csv(file="iterative_samples.csv") #importing the data
glimpse(iterative) #peeking at the data
```

```
## Rows: 30,000
## Columns: 2
## $ n_samples   <int> 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20~
## $ sample_mean <dbl> 3970.60, 3296.55, 4063.35, 4433.35, 4232.45, 2656.50, 3457~
```

```
iterative %>% #piping in the data
  ggplot(aes(sample_mean)) + #adding sample mean as the response var to plot
  geom_histogram(fill="cyan",bins=60)+ #creating a cyan histogram of the data
  xlab("Mean Gene Length (nucleotides)")+ #labeling the axis
  ggtitle("Sampling Distribution of Mean Gene Length") + #adding title
  theme_classic() +  #making the background of the fig white
  facet_wrap(vars(n_samples))
```



b. Explain how the spread in the distributions changes across the three graphs.

Answer 3b:

As the sample size increased, the spread of the sampling distribition got tighter as an increase in statistical power decreased the standard error and standard deviation.

c. Why is the shape of the distributions in 3a not the same as the shape of the distribution of the gene data?

Answer 3c:

I just made graphs of what the mean for the multiple random samples would be, this it yielded a normal samplin distribution with a bell shape. The graphs in the distribution of gene data was from 1 sampling event, so they most had a slight skew. This averages to a bell shape overall seen in the 3a graphs.

Question 4: Calculating a confidence interval

a. Calculate the "quick and dirty" confidence interval (W&S pg. 106) for the population mean (mean - 2 * se, mean + 2 * se) using a random sample of 200 genes from the "gene" dataset.

- First, use the **summarize()** function to calculate the mean and standard error as in 2a.
- Then create two new summary statistics "lower" and "upper" that calculate the lower and upper bounds of the confidence interval, respectively.
- Make sure to run the full code chunk using the green arrow at the top right of the chunk before interpreting output to ensure that the "seed" was set before the random samples were taken.

Answer 4a:

```
# Makes sure the random sample is the same each time you run your code.
# Make sure you run this line of code!
set.seed(23) #making sure I use the correct randomization
gene %>% #piping in gene data
  sample_n(size=200) %>% #sampling 200 random genes
  #calculating lower and upper confidence interval, and std error necessary for that formula
  summarize(obs=length(size),se=sd(size)/sqrt(obs),lower=mean(size)-2*se,upper=mean(size)+2*se)
```

```
##   obs       se    lower    upper
## 1 200 253.7144 3159.741 4174.599
```

b. Explain in your own words what this confidence interval means.

Answer 4b:

I am 95% confident that the true mean gene length in the dataset lies between 3160 and 4175 nucleotides .