Loopy for Loops and Silly for Samples - Hands-on!

Dr B and Daphne Hansell (BMC '24)

2023-10-11

Ex.1) run the same command above replacing `i` with `banana`.

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

So this loop printed each value for `i` (or `banana`). How would you write code that printed the values 6-10?

```
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
```

Ex.2) Write a loop that adds day to each day of the week so that the output of `days` gives you "Monday", "Tuesday", etc, one in each line.

```
## [1] "Monday"
## [1] "Tuesday"
## [1] "Wednesday"
## [1] "Thursday"
## [1] "Friday"
## [1] "Saturday"
## [1] "Sunday"
```

Ex.3) Now write a while loop that prints the value, then doubles it, starting at 1 and stopping at 100.

```
## [1] 1
## [1] 2
## [1] 4
## [1] 8
## [1] 16
## [1] 32
## [1] 64
```

Say you have a number and you want to find the smallest divisor that isn't 1. You could use a while loop to do this.

```
## [1] 7
```

Ex.4) Let's start with some basic sampling. Imagine you have a coin, equal chances of heads and tails. You want to flip it 10 times and see how many heads and how many tails. First make a vector called `coin_toss`, with "heads" and "tails" as elements. Then, sample it 10 times, with replacement:

```
##  [1] "tails" "heads" "heads" "heads" "tails" "heads" "heads" "heads" "heads"
## [10] "tails"
```

Now lets try and do this same thing with a for loop. Make an empty vector called all_tosses and use it to generate 10 coin tosses. Hint:Because you are looping over the sample you are only sampling one each time

Now I'm being sneaky. I challenge you to flip a coin but its weighted. It lands on heads 60% of the time. Use the same loop and add the probabilities to the sample function. Hint: Use the `prob` argument in the `sample` function.

Now let's see what happens if we flip both coins even more times. Use your original loop with the fair coin and toss is 100 times. Then take the weighted coin, change the variable names to make them different (add the word weighted or rigged) and toss that 100 times. Then count how many heads and how many tails came from each coin. Make sure to make an empty vector for each loop. hint: you can use the table function to get a count of heads and tails

```
## all_tosses
## Heads Tails
##    49    51
```

```
## all_tosses_weighted
## Heads Tails
##    59    41
```

Ex.5) Read in the `human_genes.csv` file and name it `human_genes`. Play around with it using your strong exploratory data analysis skills. Get a good feel for the data set.

Overall goal of this exercise is to compare sample means at different sample sizes to the population mean.

Its big right? So let's take some samples.

NOTE: Your output will look different to the pdf because your samples will be different values. 5.1. First filter `human_genes` to only keep the size and name column.

```
## Rows: 22385 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): gene, name, description
## dbl (1): size
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

5.2. create a second data frame with summary statistics mean, median and SD of length. (what is interesting about this data??). Hint: use the `summarize` function.

5.3. Now let's take some samples. Take 3 samples of gene size with replacement on. Use n = 10, 100, 1000. Calculate the mean of these samples, and save each mean as mean10, mean100, and mean1000.

5.4. Compare these means to the population mean. Which is closest?

Ex.6) Now let's combine this with the skills you learned in loops. You are going to take 100 samples of each of those sample sizes to compare

6.1 First you want to create 3 vectors to store each of your results. Them mean_values_10 etc.

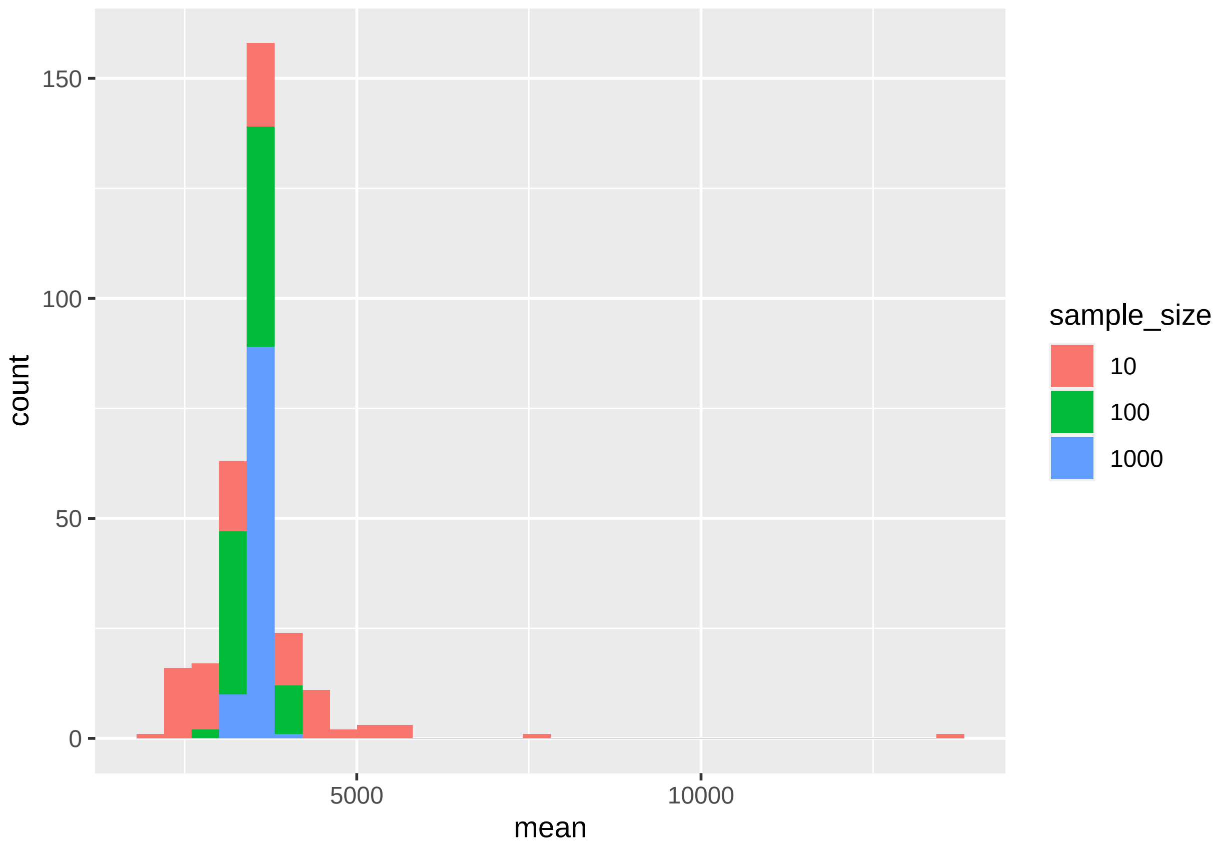
6.2 Now you write a for loop like you did before. Execept you can include all three of your new vectors in the same loop.

Ex.7). We want to compare our samples to see which sample size got closest to our population mean. You are going to make a histogram of these means.

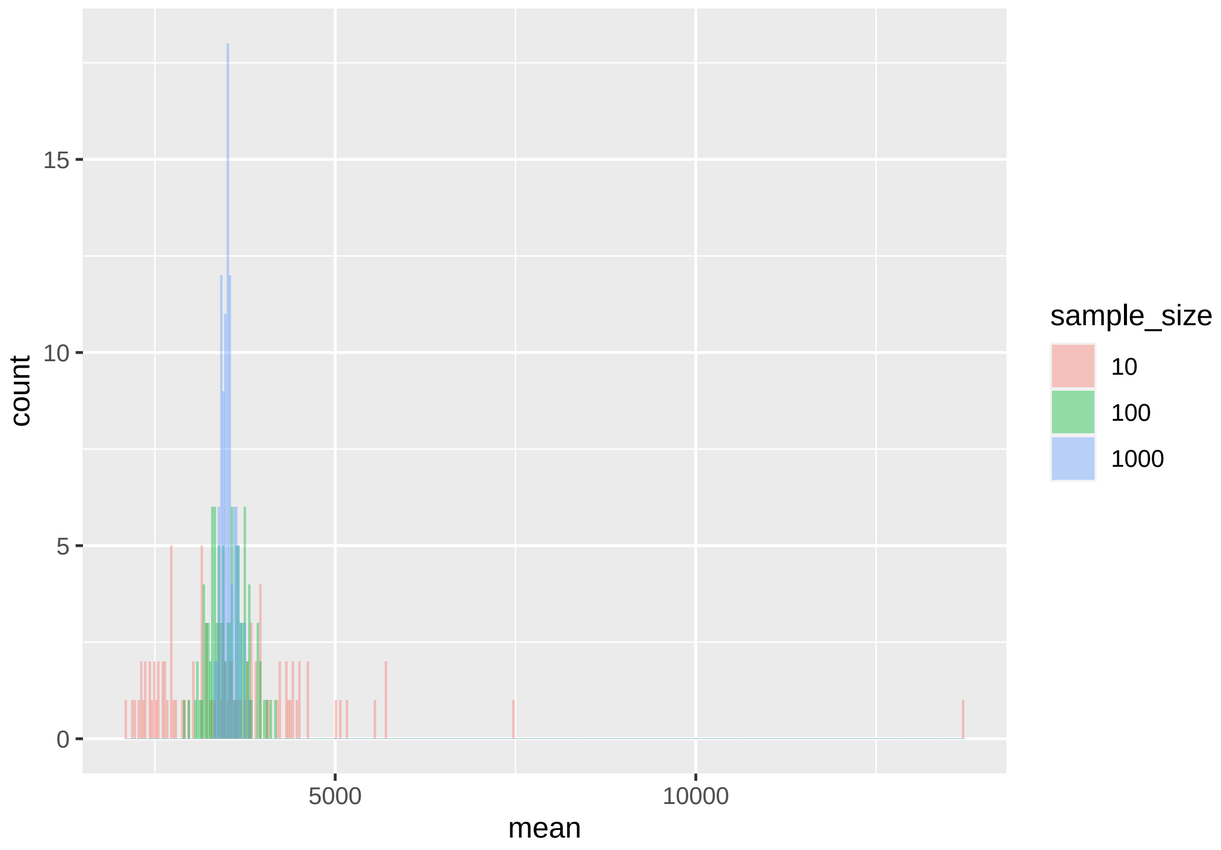
7.1 It's probably a good idea to start with making a data frame of our values. Take the vectors you just made and make one column called "mean" and another column with the corresponding sample size called "sample_size." You're going to have to make this a factor for the graphs to turn out right. You also need to nestle the functions `rep` inside your factor call, with the argument `each = 100`. Call your data frame `mean_data`.

7.2 Lets make a basic histogram. x axis is mean, fill by sample_size.

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

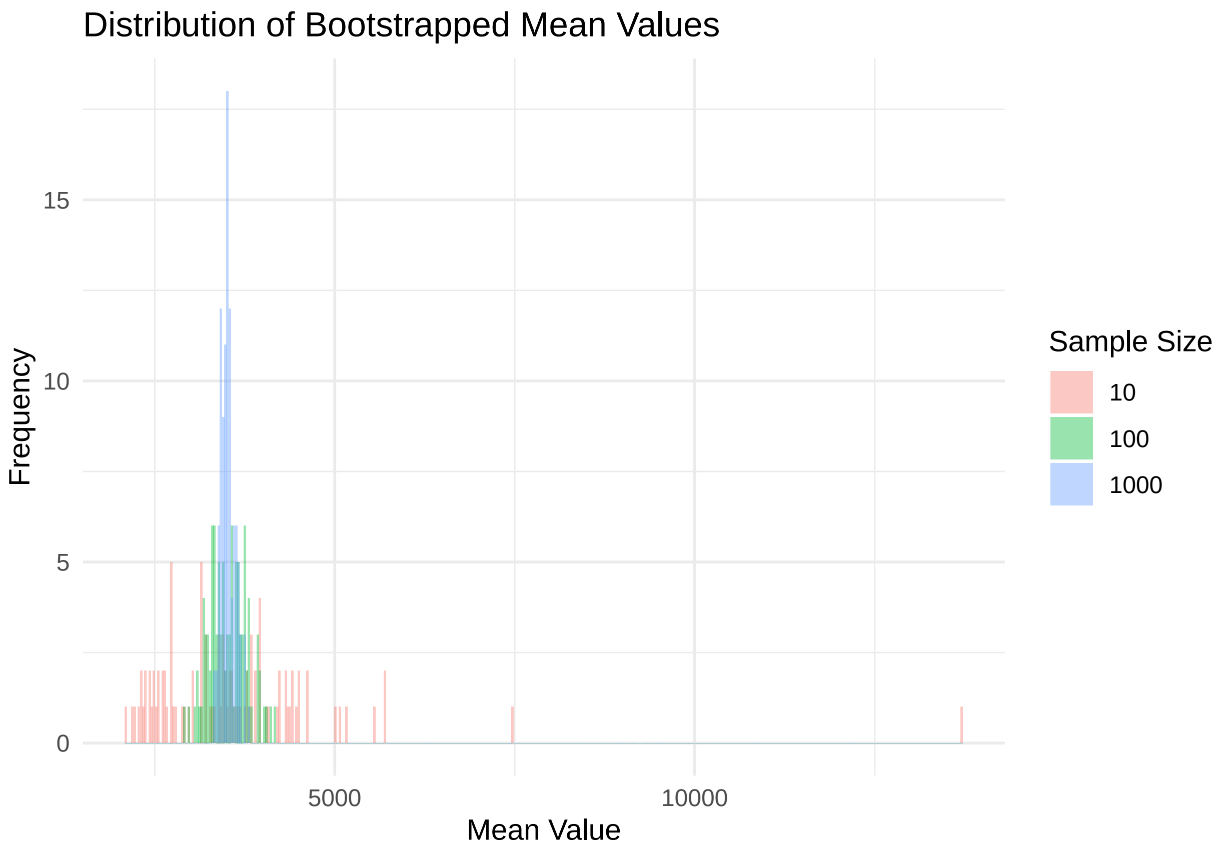


7.3 This is a bit hard to read. Let's add some detail. inside geom_histogram, change position to "fill", and both alpha to 0.4 and binwidth to 30.



7.4

Much better! Ok add some labels on your axis and legend. perhaps even add a theme if you feel silly


Distribution of Bootstrapped Mean Values
Frequency
Mean Value
Sample Size