CSC100/CSC200 Homework #8: Power of Sampling

Caroline Hall

Fall 2021 | Data Science for the World | October 22, 2021

Please complete this notebook by filling in the cells provided. When you're done:

1. Remember to put your name in the header at the top of this notebook where it says `author`.
2. Select `Knit (Knit to Word)` from the toolbar menu.
3. Read that file! If any of your lines are too long and get cut off, we won't be able to see them, so break them up into multiple lines and knit again.
4. Save that Word document as a PDF file.
5. Submit BOTH this `.Rmd` file and the **PDF** file you generated to Gradescope. Some questions are autograded and you may improve your score on the tests given by resubmitting your work as many times as you like up to the deadline.
6. **Passing the automatic tests given does not guarantee full credit on any question.** The tests are provided to help catch some common mistakes, but it is *your* responsibility to answer the questions correctly.

If you cannot submit online, come to office hours for assistance. The office hours schedule appears on Blackboard.

This homework assignment is due **October 29 at 3:00PM**. Directly sharing answers is forbidden, but discussing problems with instructors and/or with classmates is encouraged.

Reading:

- Chapter 7 textbook

Run the cell below to prepare the notebook.

Part I: Sampling Basketball Players. This part uses salary data and game statistics for basketball players from the 2014-2015 NBA season. The data was collected from basketball-reference and spotrac.

Let's have a look at the data:

```
player_data

## # A tibble: 492 × 10
##    Name           Age Team  Games Rebounds Assists Steals Blocks Turnovers
Points
##    <chr>        <dbl> <chr> <dbl>    <dbl>   <dbl>  <dbl>  <dbl>     <dbl>
<dbl>
##  1 James Hard…    25 HOU      81      459     565    154     60       321
2217
```

```
## 2 Chris Paul     29 LAC      82       376      838     156      15         190
1564
## 3 Stephen Cu…    26 GSW      80       341      619     163      16         249
1900
## 4 Anthony Da…    21 NOP      68       696      149     100     200          95
1656
## 5 DeAndre Jo…    26 LAC      82      1226       61      81     183         109
946
## 6 Jimmy Butl…    25 CHI      65       379      212     114      36          93
1301
## 7 Damian Lil…    24 POR      82       378      507      97      21         222
1720
## 8 Russell We…    26 OKC      67       488      574     140      14         293
1886
## 9 Pau Gasol      34 CHI      78       919      210      25     147         158
1446
## 10 Kyrie Irvi…   22 CLE      75       237      389     114      20         186
1628
## # … with 482 more rows
```

```
salary_data
```

```
## # A tibble: 492 × 2
##    PlayerName          Salary
##    <chr>                <dbl>
##  1 Kobe Bryant       23500000
##  2 Amar'e Stoudemire 23410988
##  3 Joe Johnson       23180790
##  4 Carmelo Anthony   22458401
##  5 Dwight Howard     21436271
##  6 LeBron James      20644400
##  7 Chris Bosh        20644400
##  8 Chris Paul        20068563
##  9 Deron Williams    19754465
## 10 Rudy Gay          19317326
## # … with 482 more rows
```

Question 1. We would like to relate players' game statistics to their salaries. Compute a tibble called `full_data` using `dplyr` code that includes one row for each player who is listed in *both* `player_data` and `salary_data`. It should include all the columns from `player_data` and `salary_data`, except the `"PlayerName"` column.

```
full_data<- inner_join(player_data, salary_data,  by =
c("Name"="PlayerName"))
full_data
```

```
## # A tibble: 492 × 11
##    Name           Age Team  Games Rebounds Assists Steals Blocks Turnovers
Points
##    <chr>        <dbl> <chr> <dbl>    <dbl>   <dbl>  <dbl>  <dbl>     <dbl>
<dbl>
```

```
##   1 James Hard…    25 HOU       81        459       565      154      60         321
2217
##   2 Chris Paul     29 LAC       82        376       838      156      15         190
1564
##   3 Stephen Cu…    26 GSW       80        341       619      163      16         249
1900
##   4 Anthony Da…    21 NOP       68        696       149      100     200          95
1656
##   5 DeAndre Jo…    26 LAC       82       1226        61       81     183         109
946
##   6 Jimmy Butl…    25 CHI       65        379       212      114      36          93
1301
##   7 Damian Lil…    24 POR       82        378       507       97      21         222
1720
##   8 Russell We…    26 OKC       67        488       574      140      14         293
1886
##   9 Pau Gasol      34 CHI       78        919       210       25     147         158
1446
## 10 Kyrie Irvi…    22 CLE       75        237       389      114      20         186
1628
## # … with 482 more rows, and 1 more variable: Salary <dbl>
```

```
. = ottr::check("tests/sampling_players_q1.R")
```

```
## All tests passed!
```

Rather than getting data on every player, imagine that we had gotten data on only a smaller subset of the players. For 492 players, it's not so unreasonable to expect to see all the data, but usually we aren't so lucky. Instead, we often make *statistical inferences* about a large underlying population using a smaller sample.

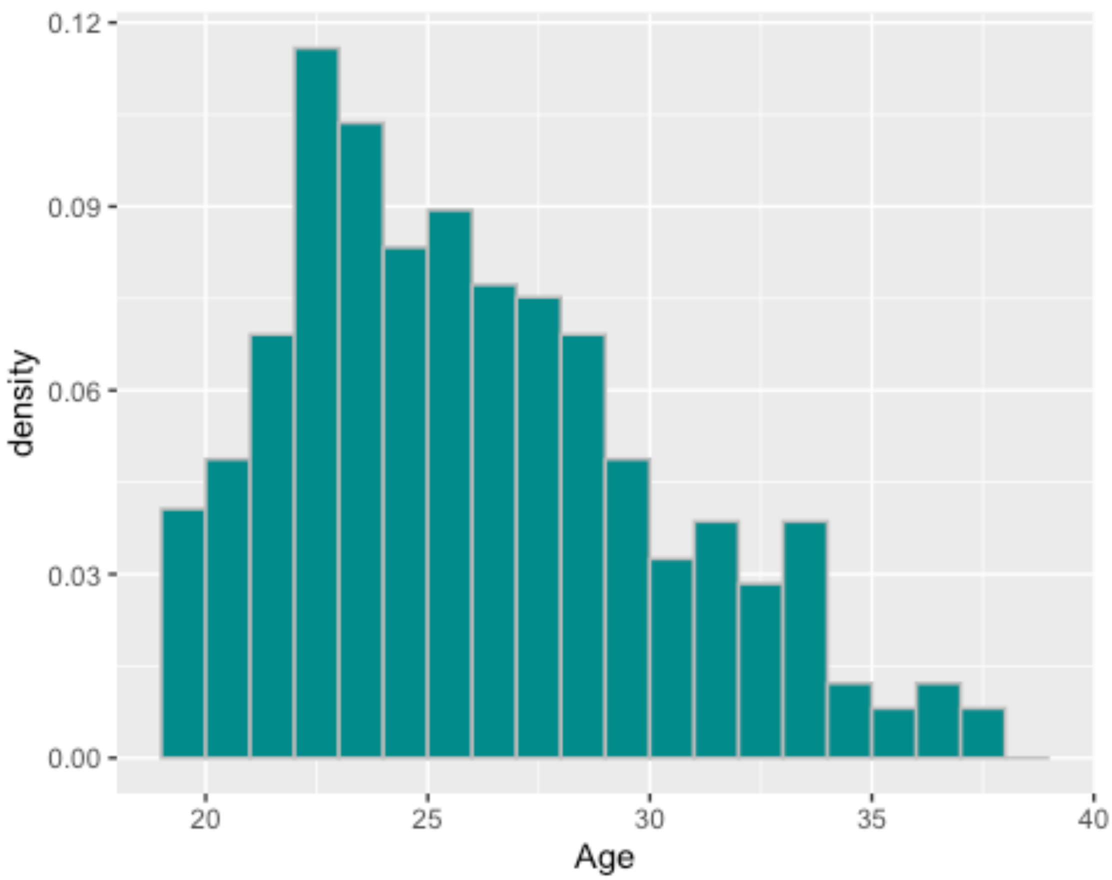
A statistical inference is a statement about some statistic of the underlying population, such as "the average salary of NBA players in 2014 was $3". You may have heard the word "inference" used in other contexts. It's important to keep in mind that statistical inferences, unlike, say, logical inferences, can be **wrong**!

A general strategy for inference using samples is to estimate *parameters* of the (unknown) population by computing it on a sample that we do have – this is what we call the *statistic*. This strategy sometimes works well and sometimes doesn't. The degree to which it gives us useful answers depends on several factors, and we'll touch lightly on a few of those in this assignment.

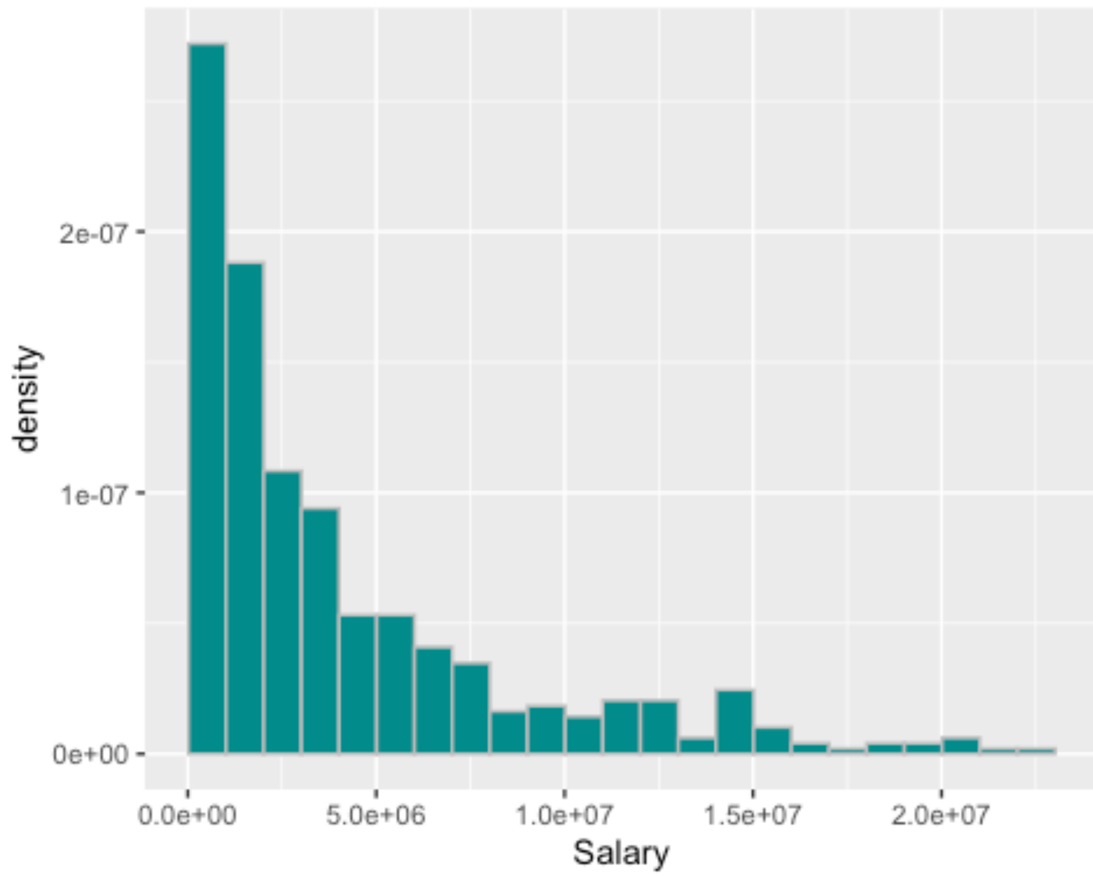
One very important factor in the utility of samples is how they were gathered. This is what we call a *sampling plan*, as discussed in Section 7.6. Let's see how they behave on the NBA player dataset.

Question 2. Complete the `plot_age_salary_histograms()` function, which takes a tibble as an argument that has columns `Age` and `Salary`, and draws a histogram for each one. The histograms should be drawn in density scale. Use the bins provided (`age_bins` and `salary_bins`) in your geom histogram call.

```r
plot_age_salary_histograms <- function(tib) {
  age_bins <- seq(min(tib %>% pull(Age)), max(tib %>% pull(Age)) + 1)
  salary_bins <- seq(min(tib %>% pull(Salary)), max(tib %>% pull(Salary))+ 1,
1e6)

gg1<- ggplot(tib, aes(x=Age))+
  geom_histogram(aes(y=..density..),
                 fill="darkcyan",
                 color="gray",
                 breaks=age_bins)

gg2<- ggplot(tib, aes(x=Salary))+
  geom_histogram(aes(y=..density..),
                 fill="darkcyan",
                 color="gray",
                 breaks=salary_bins)
print(gg1)
print(gg2)
}

plot_age_salary_histograms(full_data) # an example call
```



0.12
0.09
0.06
0.03
0.00
density
20
25
30
35
40
Age



Question 3. Create a function called `compute_statistics()` that takes a tibble as an argument containing ages and salaries and:

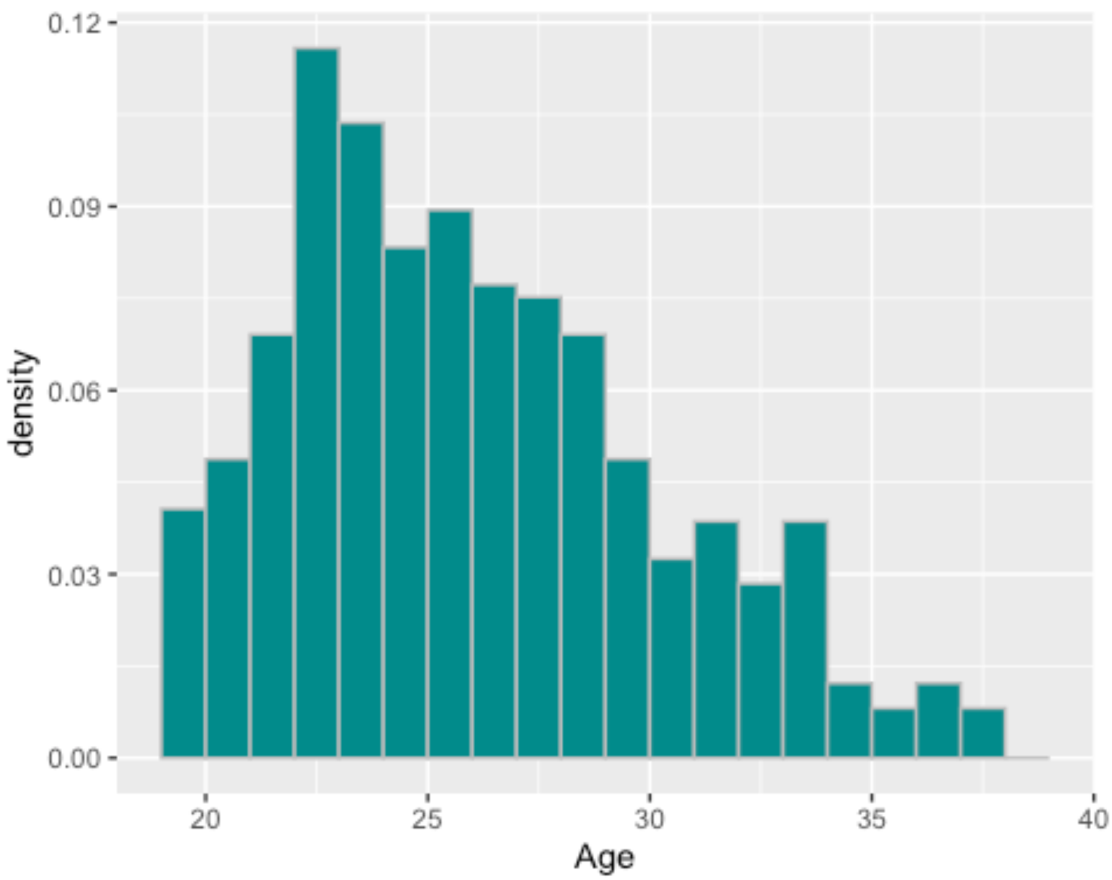
- Draws a histogram of ages
- Draws a histogram of salaries
- Return a two-element *vector* containing the average age and average salary

You can call your `plot_age_salary_histograms()` function to draw the histograms!

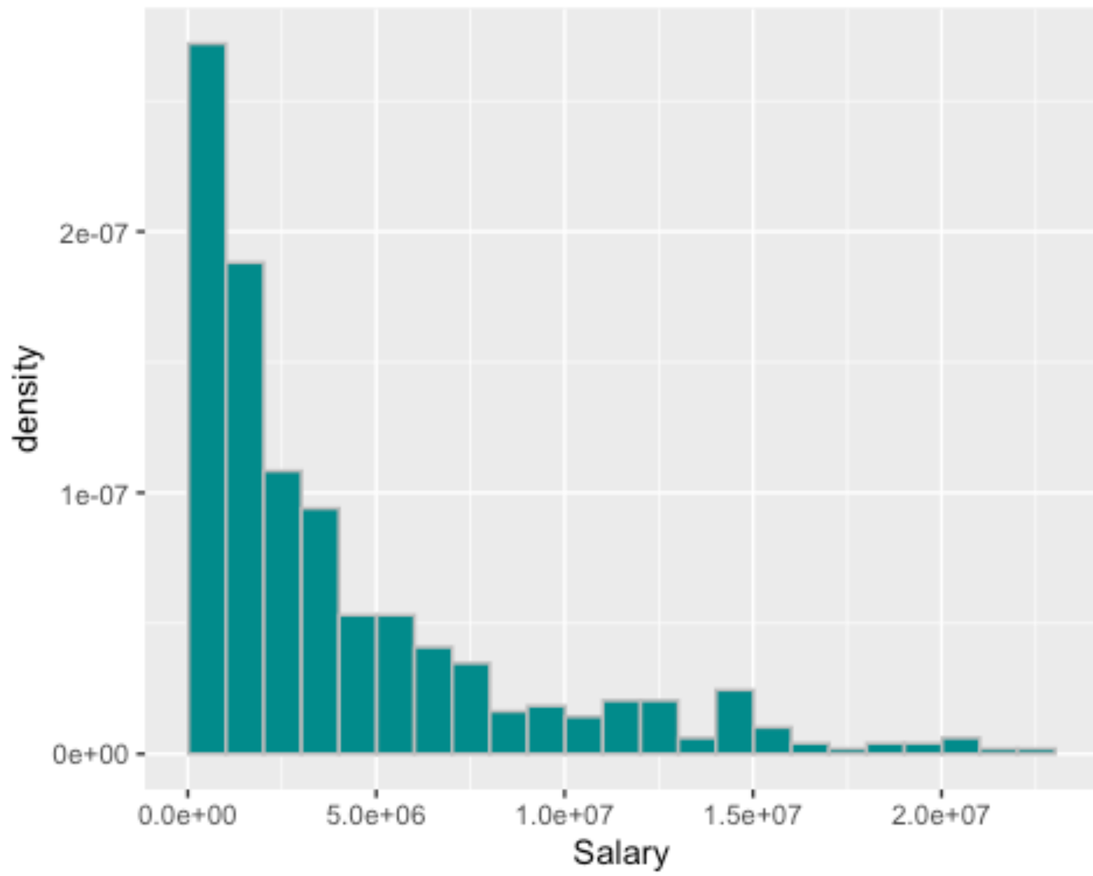
```
compute_statistics <- function(tib) {
  plot_age_salary_histograms(tib)
  c(mean(pull(tib, Age)), mean(pull(tib, Salary)))
}
```

Let us call the function you wrote on `full_data`, which is the *population* of 492 players.

```
full_data_stats <- compute_statistics(full_data)
```



0.12
0.09
0.06
0.03
0.00
density
20
25
30
35
40
Age




```
full_data_stats

## [1] 2.653659e+01 4.269776e+06
```

Part II: Convenience sampling. One sampling methodology, which is **generally a bad idea**, is to choose players who are somehow convenient to sample. For example, you might choose players from one team that's near your house, since it's easier to survey them. This is called *convenience sampling*.

Suppose you survey only *relatively new* players with ages less than 22. Sadly, the experienced players didn't bother to answer your surveys about their salaries…

Question 1. Assign `convenience_sample_data` to a subset of `full_data` that contains only the rows for players under the age of 22.

```
convenience_sample_data<- full_data %>%
  filter(Age < 22)
convenience_sample_data

## # A tibble: 44 × 11
##    Name           Age Team  Games Rebounds Assists Steals Blocks Turnovers
Points
##    <chr>        <dbl> <chr> <dbl>    <dbl>   <dbl>  <dbl>  <dbl>     <dbl>
<dbl>
```

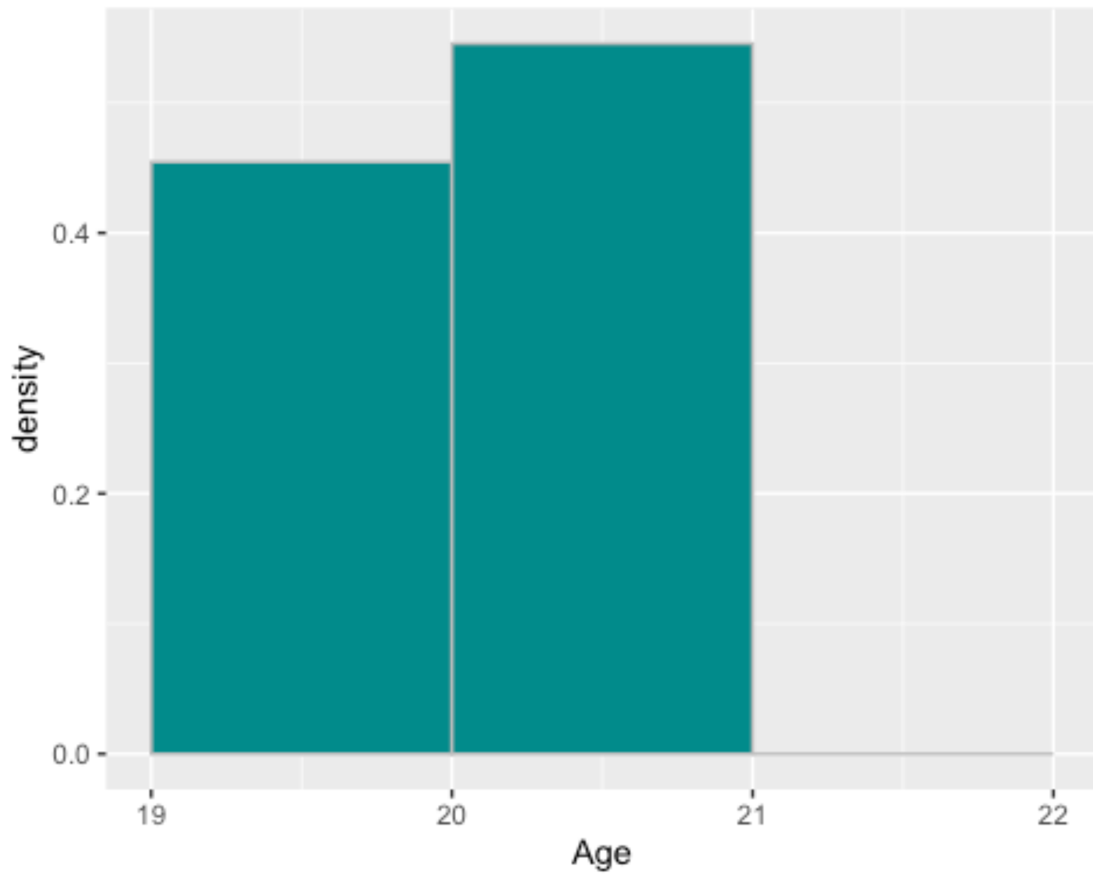
```
##   1 Anthony Da…    21 NOP       68        696       149      100      200          95
1656
##   2 Andre Drum…    21 DET       82       1104        55       73      153         120
1130
##   3 Giannis An…    20 MIL       81        542       207       73       85         173
1030
##   4 Steven Ada…    21 OKC       70        523        66       38       86          99
537
##   5 Nerlens No…    20 PHI       75        611       128      133      142         146
744
##   6 Michael Ki…    21 CHO       55        416        77       30       38          63
598
##   7 Bradley Be…    21 WAS       63        241       194       76       18         123
962
##   8 Alex Len       21 PHO       69        454        32       34      105          74
432
##   9 Marcus Sma…    20 BOS       67        222       208       99       18          90
523
## 10 Kentavious…    21 DET       82        255       109       93       18          94
1043
## # … with 34 more rows, and 1 more variable: Salary <dbl>
```

```
. = ottr::check("tests/convenience_q1.R")
```

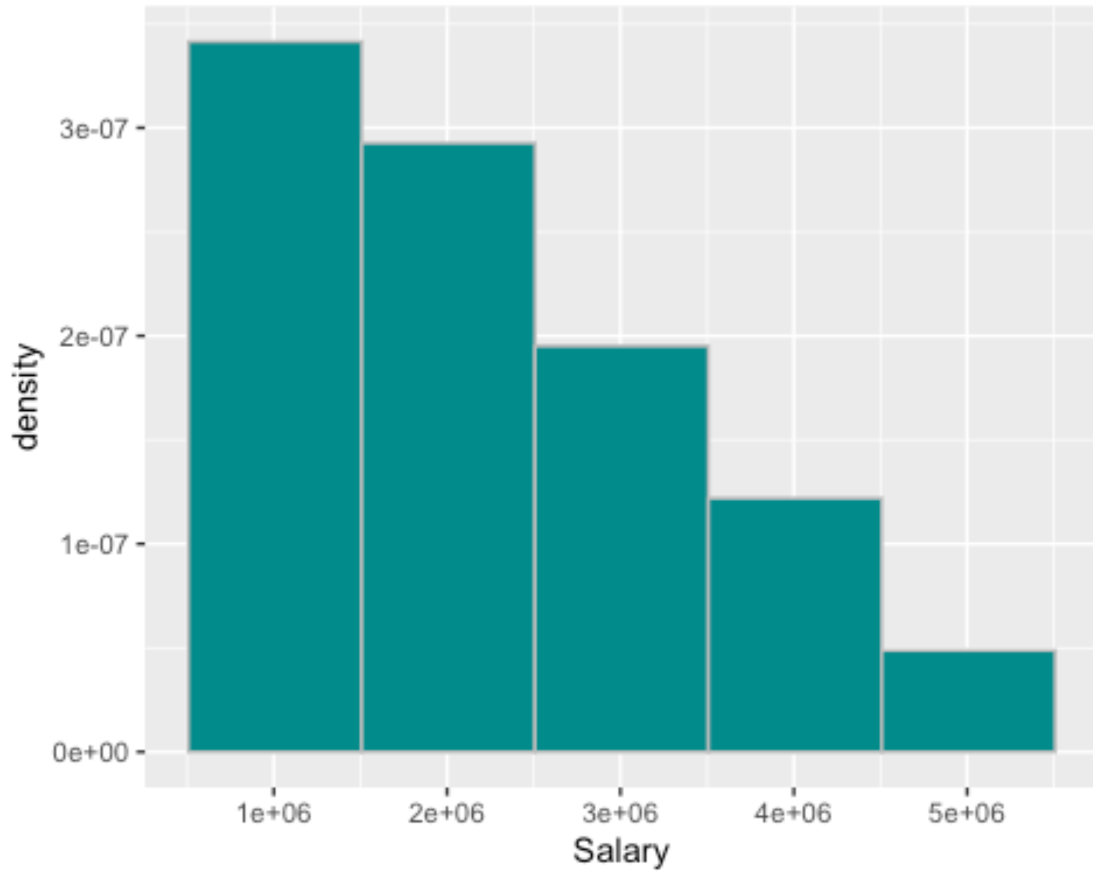
```
## All tests passed!
```

Question 2. Assign `convenience_stats` to a *vector* of the average age and average salary of your convenience sample, using the `compute_statistics` function you wrote earlier. Since they're computed on a sample, these are called *sample means*.

```
convenience_stats <- compute_statistics(convenience_sample_data)
```



0.4
0.2
0.0
density
19
20
21
22
Age

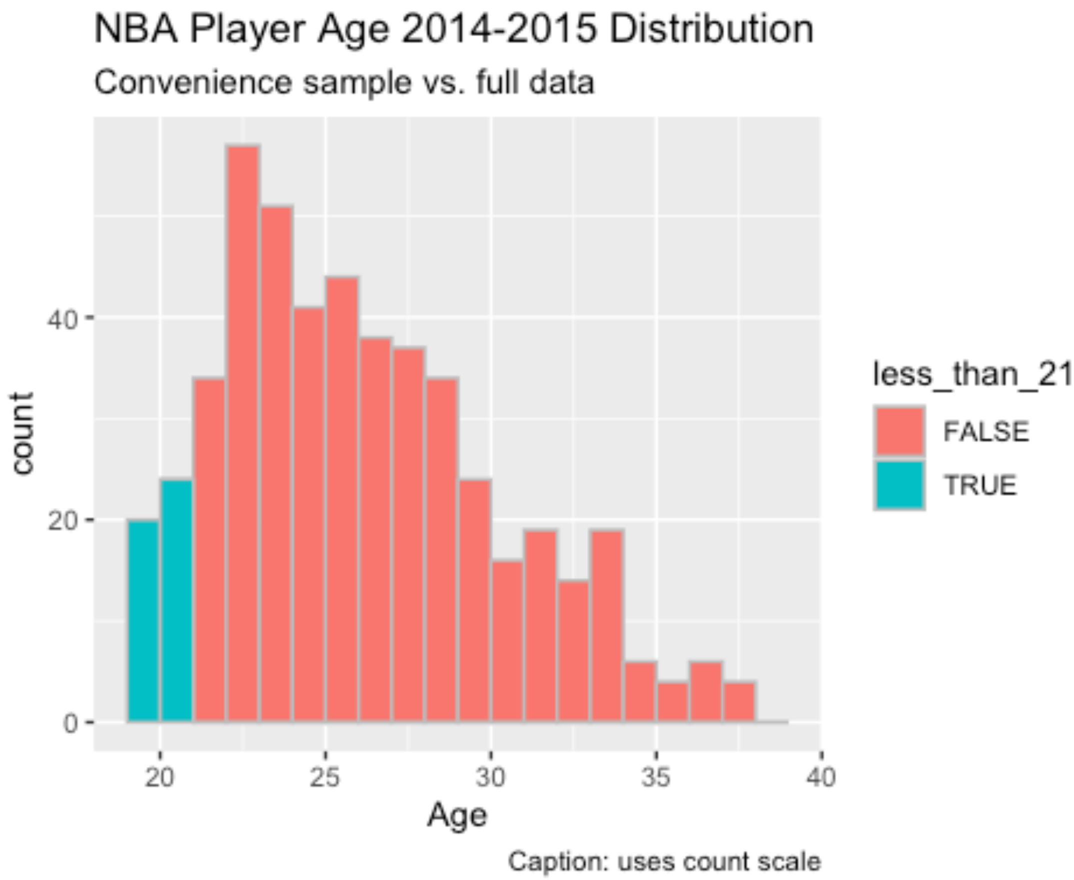



```
convenience_stats
```

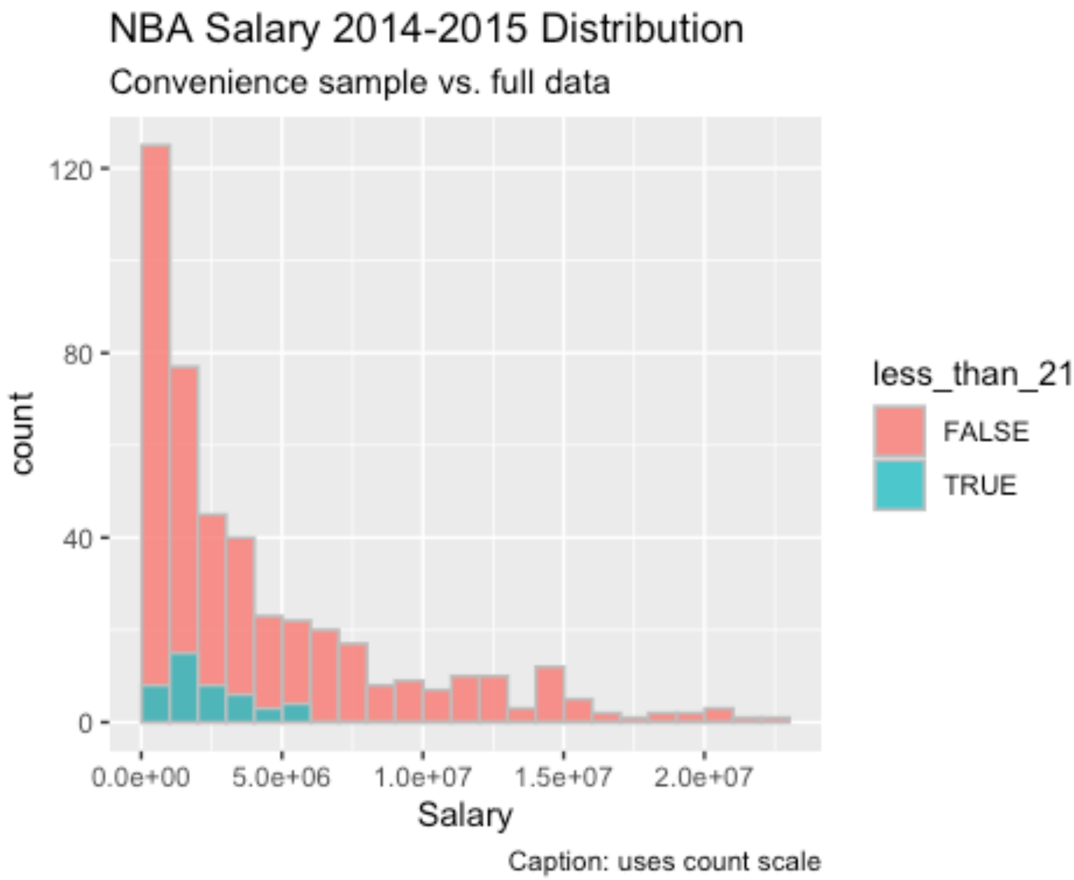
```
## [1] 2.036364e+01 2.383534e+06
```

Next, we'll compare the convenience sample salaries with the full data salaries in a single histogram. To do that, we'll plot the histogram **this time in count scale** rather than density scale. The following cell should not require any changes; just run it.

```
age_bins <- seq(min(full_data$Age), max(full_data$Age) + 1)
salary_bins <- seq(min(full_data$Salary), max(full_data$Salary) + 1, 1e6)

annotated <- full_data %>% mutate(less_than_21 = Age < 22)

# FYI: Why doesn't this plot need a positional adjustment?
ggplot(annotated,
       aes(x = Age, fill = less_than_21)) +
  geom_histogram(breaks = age_bins, color = "gray") +
  labs(title = "NBA Player Age 2014-2015 Distribution",
       subtitle = "Convenience sample vs. full data",
       caption = "Caption: uses count scale")
```




```
ggplot(annotated,
       aes(x = Salary, fill = less_than_21)) +
  geom_histogram(breaks = salary_bins, color = "gray",
                 alpha = 0.8, position = "identity") +
  labs(title = "NBA Salary 2014-2015 Distribution",
       subtitle = "Convenience sample vs. full data",
       caption = "Caption: uses count scale")
```



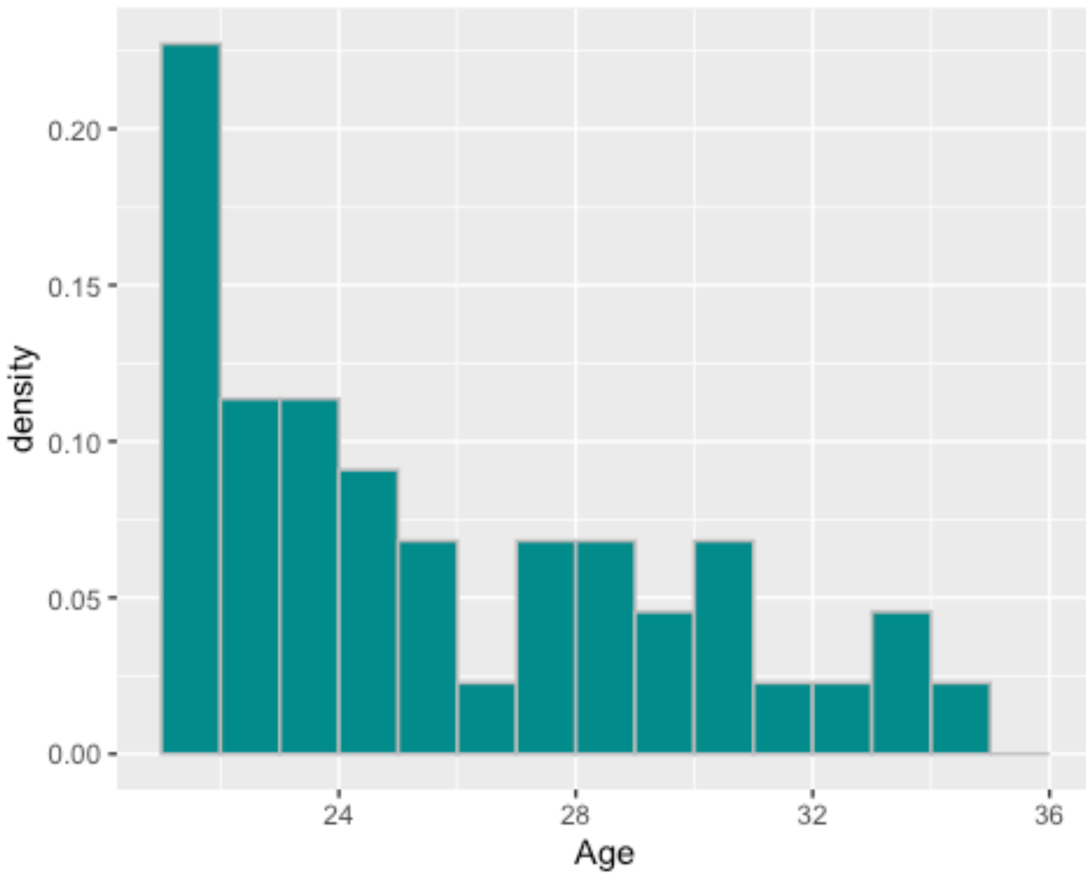
Question 3. Does the convenience sample give us an accurate picture of the age and salary of the full population of NBA players in 2014-2015? Would you expect it to, in general? Provide a short explanation. You can refer to the statistics calculated above in `convenience_stats` and compare that to `full_data_stats`.

The convenience sample does not give us an accurate picture of the age and salary of the full population. We can see that the average age for the full population is 26.5 years old, and the average salary is 4.2 million dollars per year. However, in the sample of players less than 22 years old, the average age is 20.4 years old, and the average salary is 2.4 million dollars per year. These sample statistics are much lower than the full data statistics, and thus this sample is not an accurate representation of the full data.

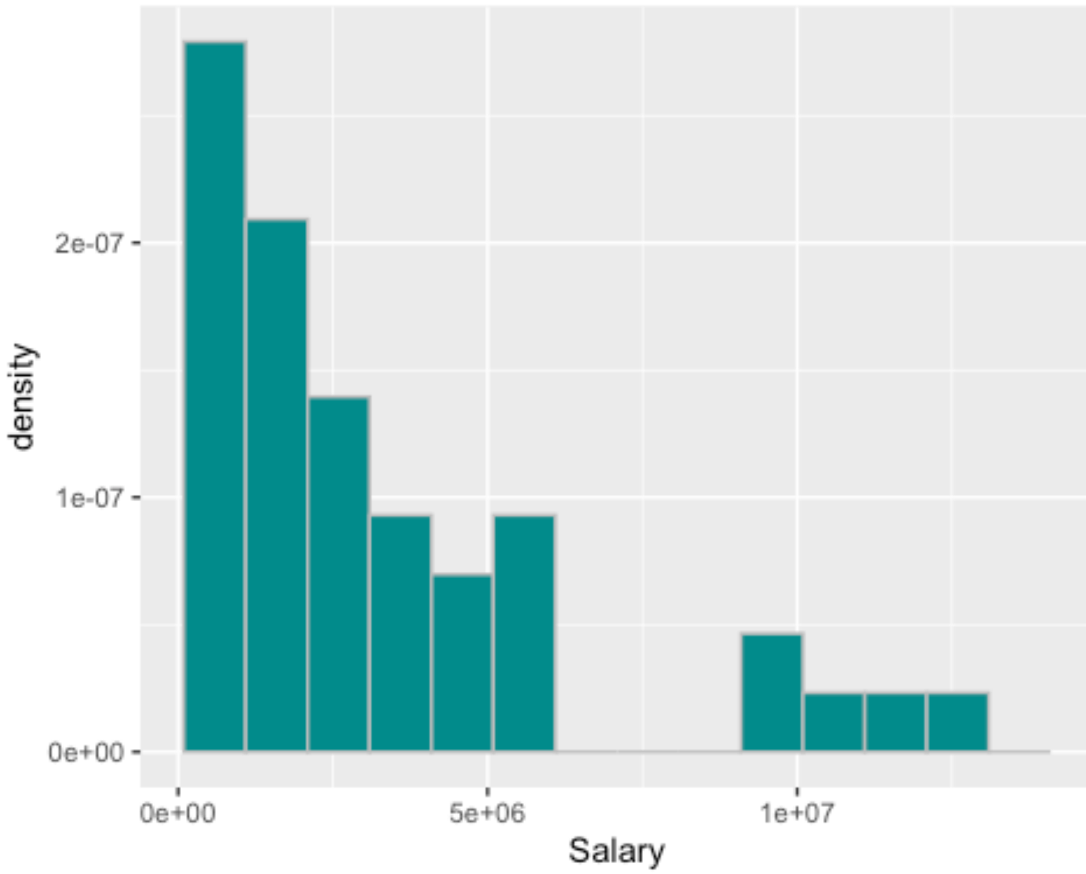
Part III: Simple random sampling. A more principled approach is to sample uniformly at random from the players. If we ensure that each player is selected at most once, this is a *simple random sample without replacement*, sometimes abbreviated to "simple random sample". Imagine writing down each player's name on a card, putting the cards in an urn, and shuffling the urn. Then, pull out cards one by one and set them aside, stopping when the specified *sample size* is reached.

Question 1. Produce a simple random sample *without replacement* of size 44 from `full_data`. Run `compute_statistics()` on this sample and store the returned vector in a name called `small_stats`.

```r
small_stats<- compute_statistics(
  full_data %>%
  slice_sample(n=44, replace = FALSE))
```



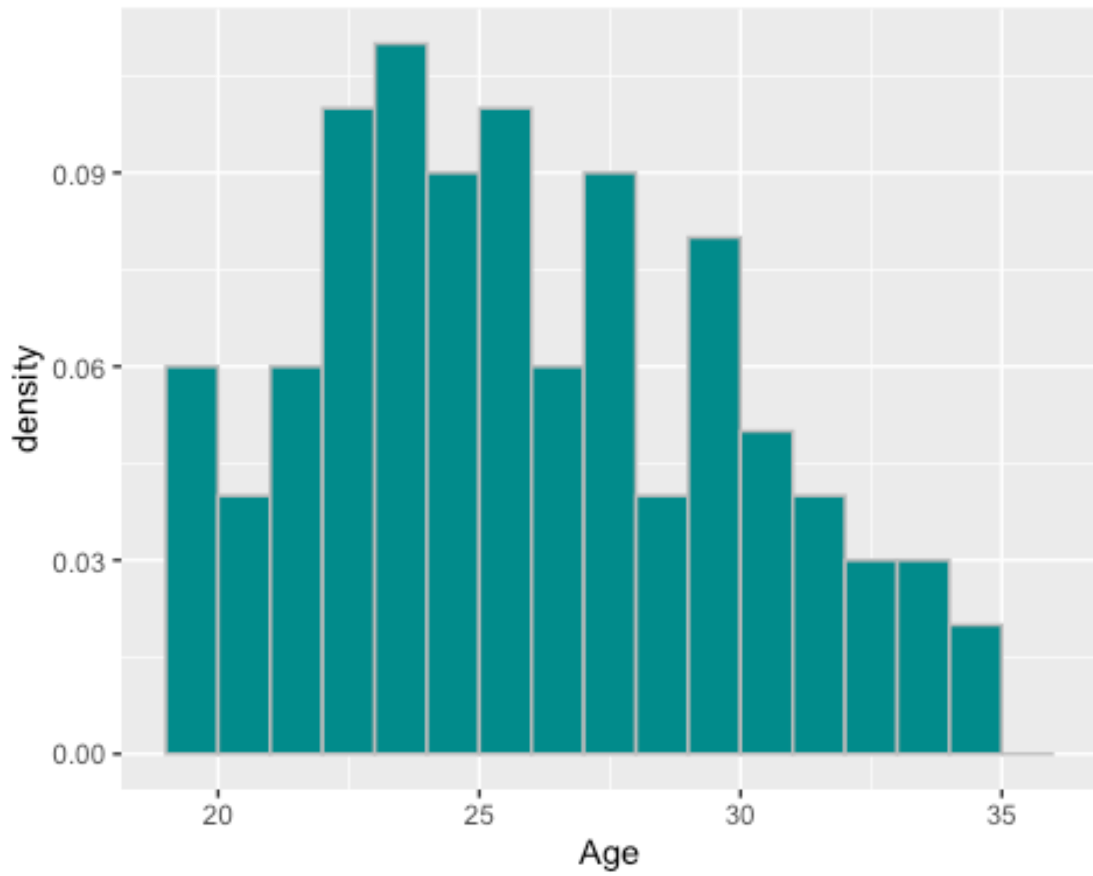
0.20
0.15
0.10
0.05
0.00
density
24
28
32
36
Age



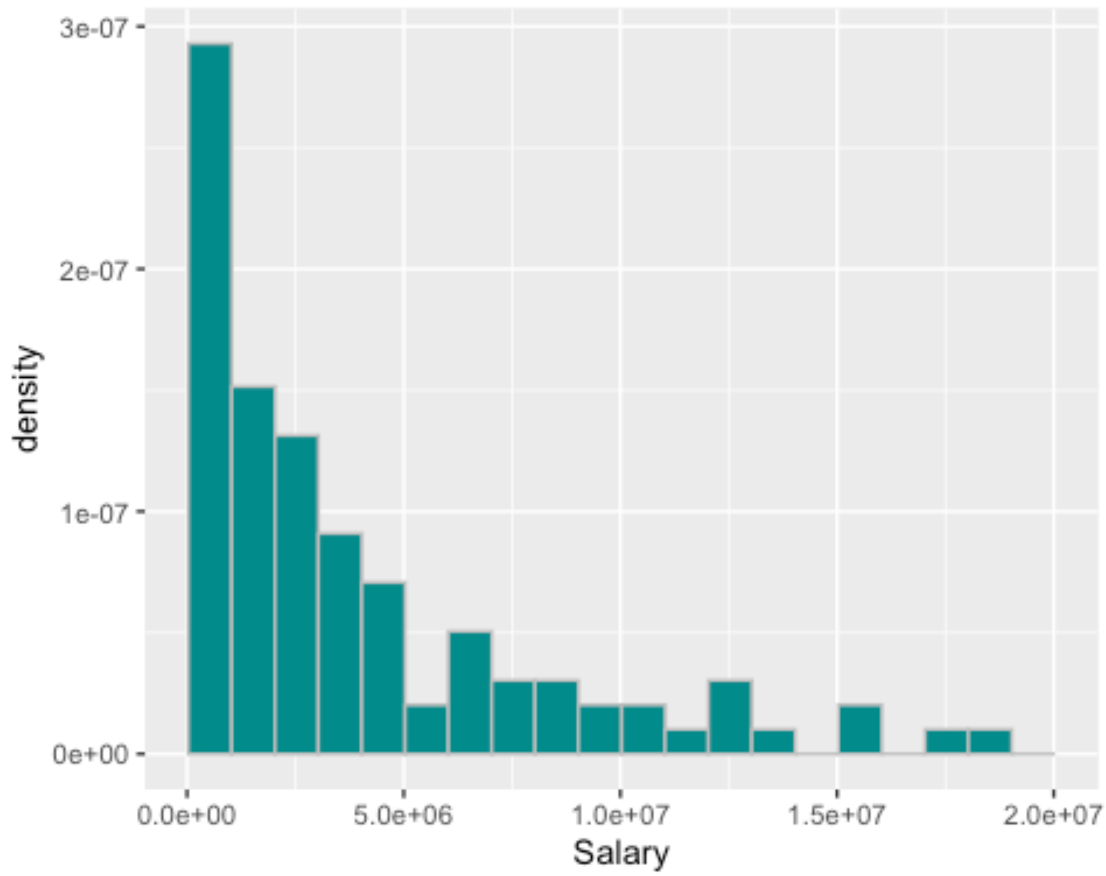

```
small_stats

## [1] 2.613636e+01 3.551649e+06
```

Question 2. Ditto the previous question, but now a simple random sample of size 100 from `full_data`. Store the resulting vector in a name called `large_stats`.

```
large_stats<- compute_statistics(
  full_data %>%
  slice_sample(n=100, replace = FALSE))
```



density
0.09
0.06
0.03
0.00
20
25
30
35
Age




```
large_stats
## [1]      26.37 4148493.99
```

For your convenience, here are all the statistics you should have computed:

```
print(small_stats)
## [1] 2.613636e+01 3.551649e+06
print(large_stats)
## [1]      26.37 4148493.99
print(convenience_stats) # from the convenience sample
## [1] 2.036364e+01 2.383534e+06
print(full_data_stats) # statistic computed directly from the population
## [1] 2.653659e+01 4.269776e+06
```

You should analyze several simple random samples of sizes 44 and 100 to get a good sense of how much the statistics vary in each analysis. Then proceed to the following question:

Question 3. Do the mean and histogram statistics seem to change more or less across samples of size 100 than across samples of size 44? For which sample size are the sample means and histograms closer to their true values for age and for salary? Do these results surprise you or is this what you expected to see? Please explain.

The mean and histogram statistics do seem to change more from the true values across a sample of size 44 than a sample of size 100. In general, we know that the larger a sample size is, the closer the statistics will be to their true values. Therefore, I did expect to see the statistics of a sample of size 100 to be closer to the true statistics for age and salary than the statistics of a sample of size 44.