

Inescapable Human Errors of the Data Generation Process*

Jessica Im

April 3, 2024

1 Prompt

Allow that the true data generating process is a Normal distribution with mean of one, and standard deviation of 1. We obtain a sample of 1,000 observations using some instrument. Simulate the following situation:

1. Unknown to us, the instrument has a mistake in it, which means that it has a maximum memory of 900 observations, and begins over-writing at that point, so the final 100 observations are actually a repeat of the first 100.
2. We employ a research assistant to clean and prepare the dataset. During the process of doing this, unknown to us, they accidentally change half of the negative draws to be positive.
3. They additionally, accidentally, change the decimal place on any value between 1 and 1.1, so that, for instance 1 becomes 0.1, and 1.1 would become 0.11.

You finally get the cleaned dataset and are interested in understanding whether the mean of the true data generating process is greater than 0.

2 Hypothesizing and Understanding the Task

Before simulating, I wanted to gain a better understanding of the question and how the distribution would be manipulated by the changes made in steps 1-3. To do so, I sketched what the distribution would look like before and throughout each change. This sketch can be found at [Sketches.png](#). In my sketching, I realized that step 1, in which 100 of the 1000 observations

*Code and data is available at: https://github.com/jimessica/data_generation/tree/main

are actually a repeat of the first 100 would not affect the normality of the distribution, since these first 100 observations should still be randomly distributed along the data. However, if I were to analyze this data under the impression that each observation is independent, my analysis would not be valid.

3 Simulating the Data

3.1 Step 1: Create the dataset with the first 100 observations repeated

To simulate the data, I used (R Core Team 2023) and (Wickham et al. 2019). I first set the seed at 250 and then created a normal distribution with 900 observations and a mean and standard deviation of one by using `rnorm()`. Next, I needed to duplicate the first 100 observations and duplicate them. To do so, I created a variable of the first 100 observations and created a completed dataframe of the initially simulated and repeated data called `full_sample`.

```
#### Simulate Data ####
## Step 1: Simulate initial sample of 900 observations from a normal distribution and then
initial_sample <-
  tibble(
    obs_value = rnorm(900, mean = 1, sd = 1)
  )

# Repeat the first 100 observations to create the final 100 observations
repeated_obs <-
  tibble(
    obs_value = initial_sample$obs_value[1:100]
  )
full_sample <-
  bind_rows(initial_sample, repeated_obs)
```

3.2 Step 2: Change half of the negative values to positive

Next, I needed to randomly select half of the negative observations and make them positive. I used ChatGPT to find out how to select all the negative values. Once I had all negative observations, I found the number of these observations using `length()` and divided that by 2. Next, I randomly sampled half of the number of observations from all of the negative observations and created a variable called `num_to_change`. Finally, I asked ChatGPT how to change the `num_to_change` numbers in `full_sample` to positive numbers.

```

## Step 2: Change half of the negative values to positive
# Step 2.1: Select negative observations
negative_obs <-
  full_sample$obs_value[full_sample$obs_value < 0]
#Step 2.2: Randomly select half of negative_obs
num_to_change <-
  sample(negative_obs, size = length(negative_obs) / 2)
#Step 2.3: Change num_to_change to positive values
full_sample <- full_sample |>
  mutate(obs_value = ifelse(obs_value %in% num_to_change, abs(obs_value), obs_value))

```

3.3 Step 3: Move the decimal place on values between 1 and 1.1

To change the decimal place on the values between 1 and 1.1, I need to remove 1 from all the values. First, I create a variable that encompasses all the numbers between 1 and 1.1. Then I followed the same method that was used to change the negative numbers.

```

## Step 3: Change the decimal place on values between 1 and 1.1
#Step 3.1: Select the numbers between 1 and 1.1
one_obs <-
  full_sample |>
  filter(obs_value >= 1 & obs_value <= 1.1)
#Step 3.2: Move the decimal in one_obs
full_sample <-
  full_sample |>
  mutate(obs_value = ifelse(obs_value %in% one_obs$obs_value, (obs_value - 1), obs_value))

```

3.4 Step 4: Check whether the mean of the distribution is above 0

Finally, to check the mean and whether it is above 0. I created a variable called `mean_cleaned_data` which is the mean of the `full_sample`.

```

## Step 4: Calculate the mean of the cleaned dataset
mean_cleaned_data <- mean(full_sample$obs_value)
mean_cleaned_data

```

```
[1] 1.031841
```

4 Reflection

It is essential to consider human error when simulating and cleaning data. This exercise demonstrated the possible human errors that can occur and alter your data. Remember that building models is not building the “truth”, but an attempt to better understand the world within a series of unavoidable constraints and biases. To lessen the impact and presence of these constraints and biases, it is always ideal to check both manually and using automated tests.

5 LLM Usage

Me: If I have a dataset called `full_sample` with an unknown number of negative numbers and an unknown number of positive numbers in R, how do I select all of the negative numbers in this dataset.

ChatGPT: To select all the negative numbers from a dataset called `full_sample` in R, you can use the following code: `negative_numbers <- full_sample[full_sample < 0]`. This code will create a new vector or subset called `negative_numbers` containing only the negative numbers from the `full_sample` dataset.

Me: This is my code so far: [Insert my code from beginning to Step 2.1]. How would I change the `num_to_change` in `full_sample` from their current negative numbers to positive in R using only the package `tidyverse`?

ChatGPT: To change the negative numbers in `num_to_change` to positive within the `full_sample` dataframe using the `tidyverse` package, you can directly use the `mutate()` function along with `ifelse()` to conditionally replace the values. Here’s how you can do it: `full_sample <- full_sample %>% mutate(obs_value = ifelse(obs_value %in% num_to_change, abs(obs_value), obs_value))`. In this code: `mutate()` is used to modify the `obs_value` column of the `full_sample` dataframe. `ifelse()` is a conditional function that checks if each value in the `obs_value` column is present in `num_to_change`. If it is, it replaces the value with its absolute value using `abs()`. If not, it keeps the original value unchanged. This code will effectively change half of the negative values in `num_to_change` to positive within the `full_sample` dataframe.

6 References

Shirley Chen helped finalize this paper by acting as a consultant and suggesting edits.

R Core Team. 2023. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. <https://www.R-project.org/>.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Golemund, et al. 2019. “Welcome to the tidyverse.” *Journal of Open Source Software* 4 (43): 1686. <https://doi.org/10.21105/joss.01686>.