
Lesson Notes | Using ChatGPT for Data Analysis

GRAPH Network & WHO, supported by the Global Fund to fight HIV, TB & Malaria

September 2024

This lesson was created by the GRAPH Network, a non-profit headquartered at the University of Geneva Global Health Institute, in collaboration with the World Health Organization, under a Global Fund 2023 grant to create e-learning modules to build in-country data capacity for epidemiological data analysis

Introduction	
Models	
1. Explain Unfamiliar Code	
2. Debug Simple Errors	
3. Add Code Comments	
4. Reformat Code	
5. Make Plots	
6. Simple Data Wrangling	
7. Translate Language	
8. Translate Programming Languages	
9. Fluid Find and Replace	
Limitations of ChatGPT	

Introduction

ChatGPT, developed by OpenAI, is a language model that can be used to assist data analysts in various tasks. It can:

1. Explain unfamiliar code
2. Debug simple errors
3. Add code comments
4. Reformat code
5. Generate plots
6. Perform simple data wrangling
7. Translate languages
8. Translate programming languages
9. Perform a fluid find and replace operation

Let's get started and see how it works!

Models

As at our last check, ChatGPT comes in two main versions, both available at chat.openai.com:

Model	Availability	Speed	Task Complexity
GPT-3.5	Free	Fast	Simple tasks
GPT-4	Paid / Bing	Slower	Complex tasks

Both models have their advantages and use cases. GPT-3.5 is suitable for simple tasks and is free to use, while GPT-4, though slower and paid, can handle more complex tasks.

Bing's AI chat service also uses GPT-4 under the hood (last checked in September 2023), so you can go to Bing if you want GPT-4 access without paying. The Bing interface is a bit

clunky for data analysis tasks though, so if you can afford to pay for GPT-4, we highly recommend it.

1. Explain Unfamiliar Code

ChatGPT can explain unfamiliar code. For instance, if you encounter the following code:

```
trees %>%  
  select(Height) %>%  
  filter(Height > 80)
```

You can ask ChatGPT, “What is this code doing?” and paste in the code.

Here is ChatGPT3.5’s response:

The code you’ve provided appears to be written in R and is likely using the dplyr package for data manipulation. Here’s a step-by-step explanation of what this code does:

1. `trees %>%`: This part of the code uses the `%>%` operator, also known as the pipe operator, which is used to chain together multiple data manipulation operations in a more readable and concise manner. It takes the data frame or tibble `trees` and passes it as the first argument to the next operation.
2. `select(Height) %>%`: After the pipe operator, the `select()` function is used. It selects a subset of columns from the data frame. In this case, it selects only the “Height” column from the `trees` data frame.
3. `filter(Height > 80)`: Following the `select()` operation, the `filter()` function is used. This function filters rows based on a specified condition. In this case, it filters the data to keep only those rows where the “Height” column has a value greater than 80.

So, in summary, the code takes the `trees` data frame, selects only the “Height” column, and then filters the rows to retain only those where the height is greater than 80. The result will be a subset of the original data containing only rows where the tree height is above 80.

2. Debug Simple Errors

ChatGPT can assist in debugging straightforward errors. You can ask ChatGPT, “Please help me understand why the code is not working,” then paste in both your code and the error message.

For example:

```
data_subset_I <-  
  trees %>%  
  select(Height) %>%  
  filter(Height > 80)  
  
summary(data_subset_l)
```

Error message:

```
Error in summary(data_subset_I) : object 'data_subset_I' not found
```

ChatGPT3.5 response:

It seems like you have a typo in your code. In the code snippet you provided, you are creating a subset of data and storing it in the variable `data_subset_I`, but then you are trying to summarize a different variable `data_subset_l` (note the lowercase “l” at the end). The variable names are not consistent, which is likely causing the issue.

You should use the correct variable name `data_subset_I` when summarizing the data. Here's the corrected code:

```
data_subset_I <-  
  trees %>%  
  select(Height) %>%  
  filter(Height > 80)  
  
summary(data_subset_I)
```

With this corrected code, it should work as expected and summarize the `data_subset_I` data frame, which contains the heights of trees greater than 80.

3. Add Code Comments

ChatGPT can also assist in adding comments to your code. For instance, you can say, “Add comments to the following code,” then paste in your code.

```
data_subset_2 <-  
  trees %>%  
  select(Height) %>%  
  filter(Height > 80)  
  
summary(data_subset_2)
```

ChatGPT3.5 response:

Certainly! Here's the code with comments added to explain each step:

```
# Select the 'Height' column from the 'trees' dataset
# and create a new dataset 'data_subset_2'
data_subset_2 <-
  trees %>%
  select(Height) %>%

# Filter 'data_subset_2' to retain only rows where 'Height' is greater than 80
  filter(Height > 80)

# Display a summary of 'data_subset_2'
# including statistics like mean, median, and quartiles
summary(data_subset_2)
```

These comments provide a clear explanation of what each line of code does, making it easier to understand the data manipulation and summary generation process.

4. Reformat Code

ChatGPT can help in reformatting code. For example, consider the following code:

```
temp<-      trees %>% select(Height) %>%filter(Height > 80)
```

You can ask ChatGPT, “Help me reformat this code nicely,” then paste in the code.

ChatGPT3.5 response:

Certainly, here's the reformatted code:

```
temp <- trees %>%
  select(Height) %>%
  filter(Height > 80)
```

I've added line breaks and indentation to make the code more readable and visually organized.

5. Make Plots

ChatGPT can assist in creating plots with ggplot2. For instance, you might ask:

“Use ggplot2 to create a scatter plot of height against girth for the dataset shown” then paste in the head of the data.

```
head(trees)
```

```
##      Girth Height Volume
## 1    8.3     70   10.3
## 2    8.6     65   10.3
## 3    8.8     63   10.2
## 4   10.5     72   16.4
## 5   10.7     81   18.8
## 6   10.8     83   19.7
```

ChatGPT3.5 response:

To create a scatter plot of height against girth for the dataset you provided using ggplot2 in R, you can follow these steps:

1. First, make sure you have the ggplot2 package installed. If you don't have it installed, you can install it using the following command:

```
install.packages("ggplot2")
```

2. Load the ggplot2 library:

```
library(ggplot2)
```

3. Create the scatter plot using the `ggplot()` function and the `geom_point()` function to add the points:

```
# Assuming you have already loaded the dataset
# If not, you can load it using data(trees)

# Create the scatter plot
ggplot(trees, aes(x = Girth, y = Height)) +
  geom_point() +
  labs(x = "Girth", y = "Height") +
  ggtitle("Scatter Plot of Height vs. Girth")
```

This code will create a scatter plot of height against girth using the `trees` dataset, with girth values on the x-axis and height values on the y-axis. Adjust the labels and title as needed for your specific requirements.

You can then ask ChatGPT a follow-up question like:

- Add a best-fit line and write the equation of the line in the caption of the plot.
- Mark the mean height and mean girth on the plot.
- Center the plot title.

Here we may see the difference between GPT3.5 and 4.

ChatGPT3.5 response:

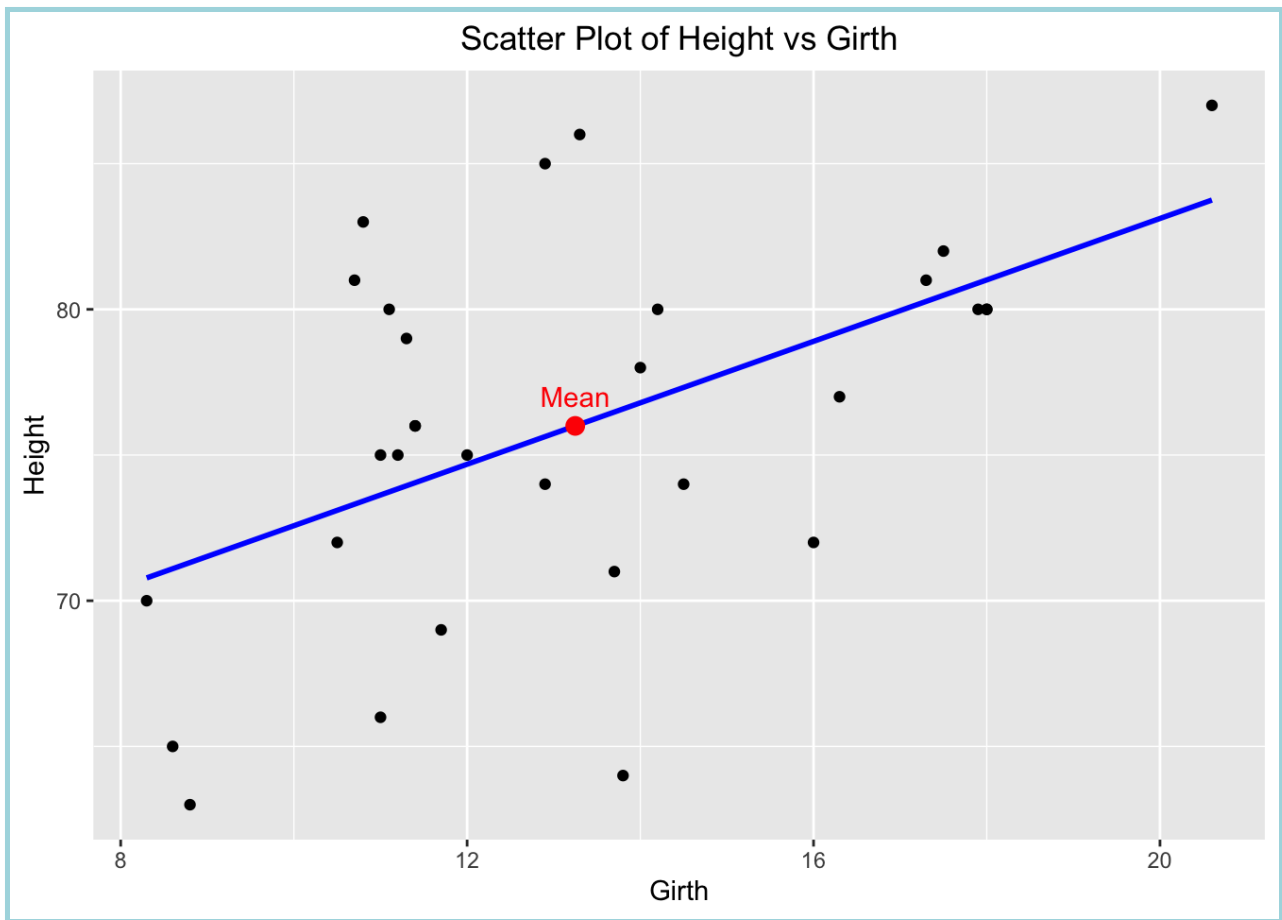
```
scatter_plot <- ggplot(trees, aes(x = Girth, y = Height)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +
  labs(x = "Girth", y = "Height") +
  ggtitle("Scatter Plot of Height vs Girth") +
  theme(plot.title = element_text(hjust = 0.5))

# Calculate mean values
mean_girth <- mean(trees$Girth)
mean_height <- mean(trees$Height)

# Add mean points
scatter_plot <- scatter_plot +
  geom_point(data = data.frame(Girth = mean_girth, Height = mean_height),
    color = "red", size = 3) +
  annotate("text", x = mean_girth, y = mean_height, label = "Mean",
    color = "red", vjust = -1)

# Print the scatter plot
print(scatter_plot)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

In the ChatGPT3.5 response, the equation of the line is missing from the caption.

Here is GPT4's response:

```
library(tidyverse)
library(scales)

# Calculate means
mean_girth <- mean(trees$Girth)
mean_height <- mean(trees$Height)

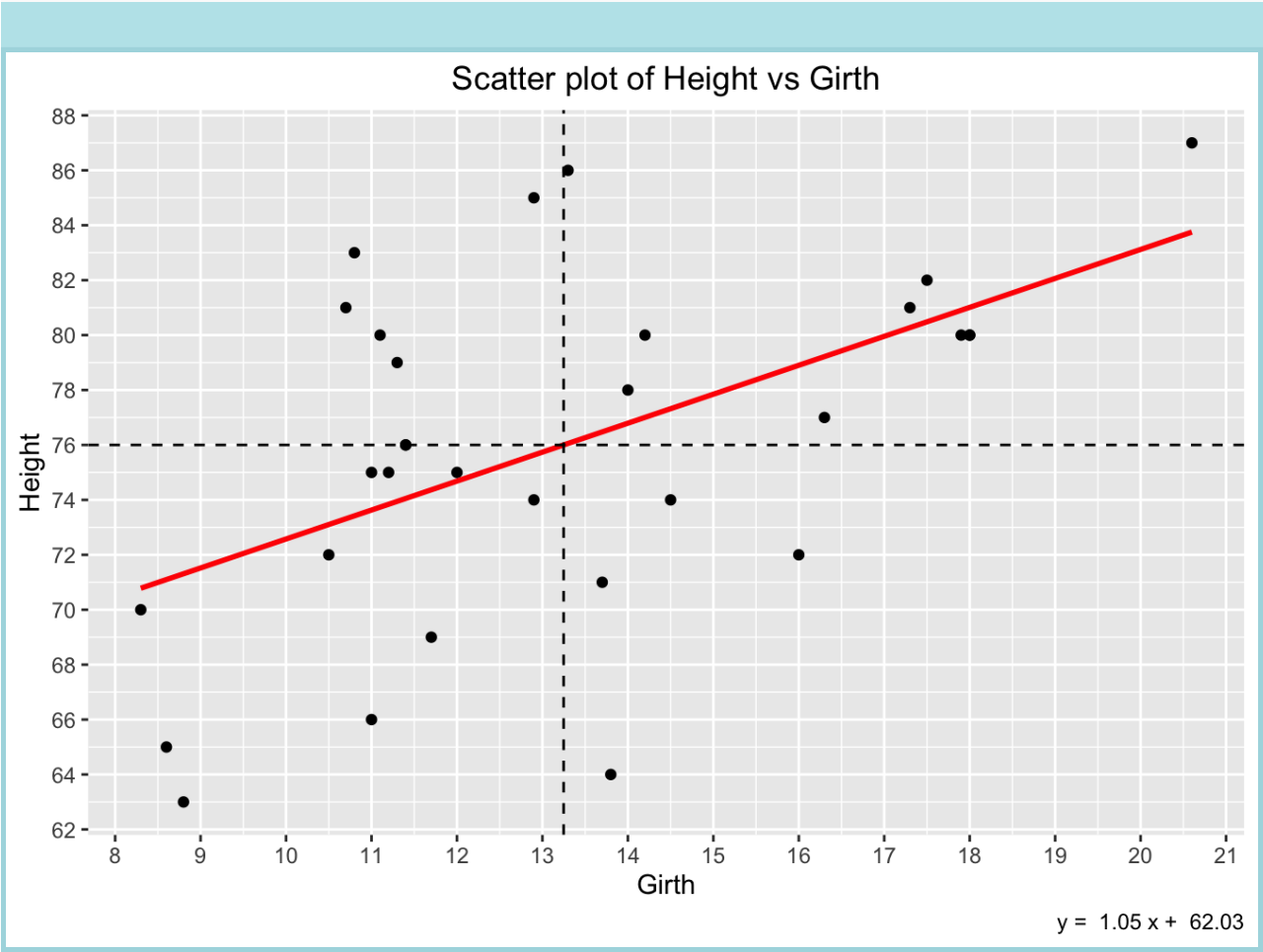
# Fit a linear model to calculate the equation of the line
fit <- lm(Height ~ Girth, data = trees)
slope <- coef(fit)[2]
intercept <- coef(fit)[1]
eq <- paste("y = ", round(slope, 2), "x + ", round(intercept, 2))

# Generate the plot
p <- ggplot(trees, aes(x = Girth, y = Height)) +
  geom_point() +
  geom_smooth(method = 'lm', se = FALSE, color = "red",
             formula = y ~ x) + # Add best-fit line (linear model)
  geom_vline(xintercept = mean_girth, linetype="dashed") +
  geom_hline(yintercept = mean_height, linetype="dashed") +
  ggtitle("Scatter plot of Height vs Girth") +
```

```

theme(plot.title = element_text(hjust = 0.5)) + # Center title
scale_x_continuous(breaks = pretty_breaks(n = 10)) +
scale_y_continuous(breaks = pretty_breaks(n = 10)) +
labs(x = "Girth", y = "Height",
caption = eq) # Add equation to the caption

```



With GPT-4, you get exactly what you want, including the equation in the caption at the bottom right of the plot.

6. Simple Data Wrangling

ChatGPT can help with simple data wrangling tasks. For instance:

Consider the following dataset:

```
head(trees)
```

##	Girth	Height	Volume
## 1	8.3	70	10.3
## 2	8.6	65	10.3
## 3	8.8	63	10.2

```
## 4 10.5 72 16.4
## 5 10.7 81 18.8
## 6 10.8 83 19.7
```

You might ask, “Give me dplyr code to find the mean and median girth, height, and volume.”

ChatGPT3.5 gives us this code:

```
# Use summarise_all to compute mean and median for each column
result <- trees %>%
  summarise_all(
    .funs = list(mean = mean, median = median),
    na.rm = TRUE
  )
print(result)
```

7. Translate Language

ChatGPT can help translate text in your tutorial or documentation from one language to another, while preserving the code. If you use other translators, they often translate your code as well, which is undesirable.

For example, you can ask:

“Translate the following section of a tutorial into Spanish”

dplyr is part of the core of tidyverse, so it is automatically loaded with:

```
library(tidyverse)
```

It can also be loaded individually.

```
library(dplyr)
```

ChatGPT 3.5 response:

dplyr forma parte del núcleo de tidyverse, por lo que se carga automáticamente con:

```
library(tidyverse)
```

También se puede cargar de forma individual.

```
library(dplyr)
```

You can verify the validity of this translation by translating it back into English in a different session.

8. Translate Programming Languages

ChatGPT can help translate between different programming languages. For example, you can say:

“Translate the following Stata code to R:”

```
recode age (18/35=1) (36/50=2), into(agegroup)
```

ChatGPT 3.5 response:

In R, you can achieve the same recoding of the “age” variable into an “agegroup” variable using the `cut()` function. Here’s the equivalent R code:

```
# Assuming you have a dataframe called "data" with an "age" variable
data$agegroup <- cut(data$age, breaks = c(18, 35, 50), labels = c(1, 2),
                    include.lowest = TRUE)
```

In this code:

- `data$age` refers to the “age” variable in your dataframe.
- `cut()` is used to create the “agegroup” variable.
- `breaks` specifies the breakpoints for age groups (18, 35, and 50).
- `labels` assigns labels 1 and 2 to the corresponding age groups.
- `include.lowest = TRUE` ensures that values exactly equal to the breakpoints are included in the specified group.

This code will create a new “agegroup” variable in your dataframe, similar to what the Stata code does.

9. Fluid Find and Replace

Lastly, ChatGPT can help you refactor your code in situations where a simple find and replace with your IDE would be time-consuming. For example, if “Afghanistan” is written in three different ways (with a capital “A”, in lowercase letters, and as a country code in the title, “AF”), you could replace all instances with “Ghana”.

You can ask ChatGPT:

“Replace ‘afghanistan’ with ‘ghana’ for the code below:”

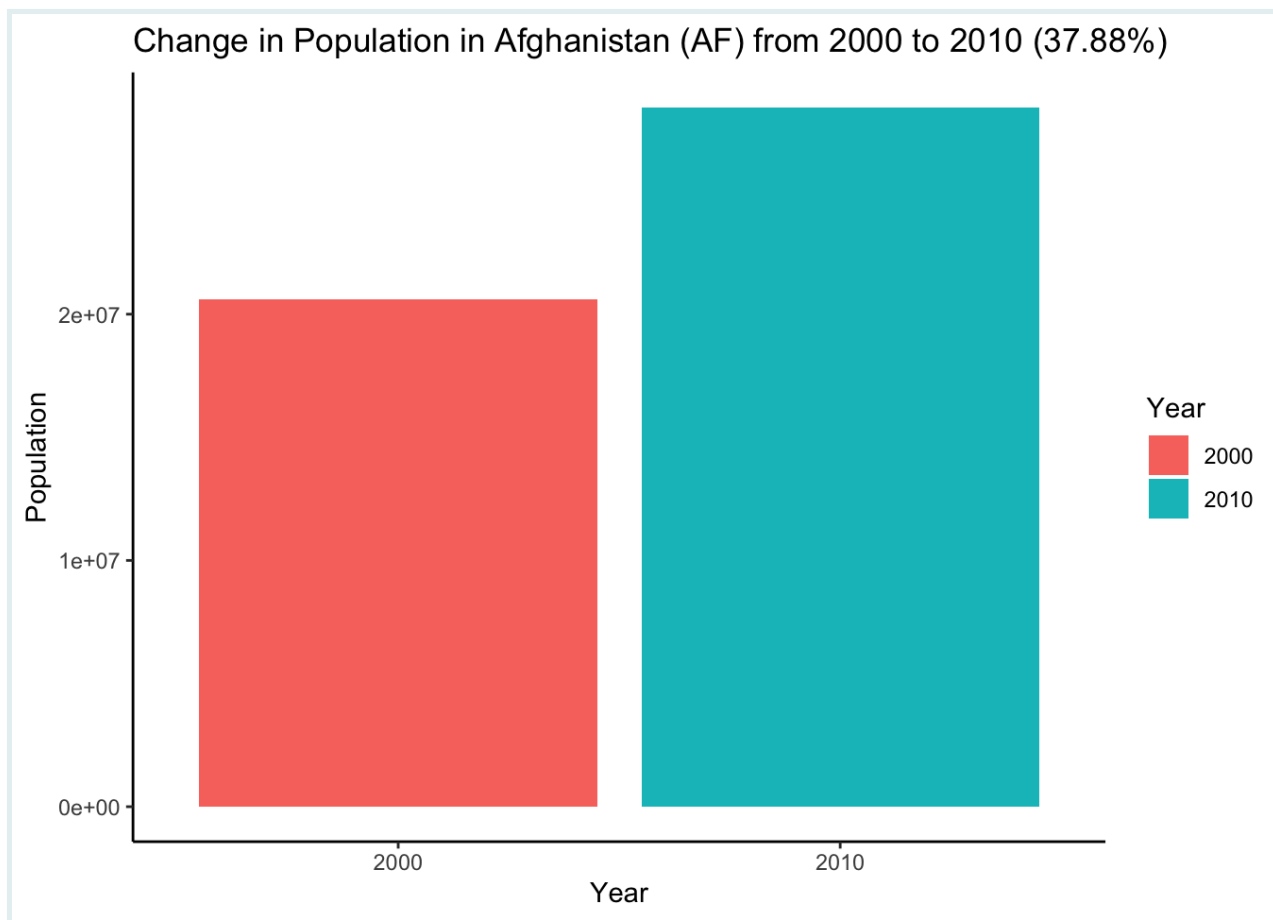
```
afghanistan_2000 <- population %>%
  filter(country == "Afghanistan", year == 2000) %>%
  pull(population)

afghanistan_2010 <- population %>%
  filter(country == "Afghanistan", year == 2010) %>%
  pull(population)

afghanistan_population <- data.frame(
  year = c(2000, 2010),
  population = c(afghanistan_2000, afghanistan_2010)
)

percent_change <-
  ((afghanistan_2010 - afghanistan_2000) / afghanistan_2000) * 100

ggplot(afghanistan_population,
  aes(
    x = factor(year),
    y = population,
    fill = factor(year)
  )) +
  geom_bar(stat = "identity") +
  labs(
    x = "Year",
    y = "Population",
    fill = "Year",
    title = paste0(
      "Change in Population in Afghanistan (AF) from 2000 to 2010 (",
      round(percent_change, 2),
      "%)"
    )
  ) +
  theme_classic()
```



ChatGPT 3.5 response:

```
ghana_2000 <- population %>%
  filter(country == "Ghana", year == 2000) %>%
  pull(population)

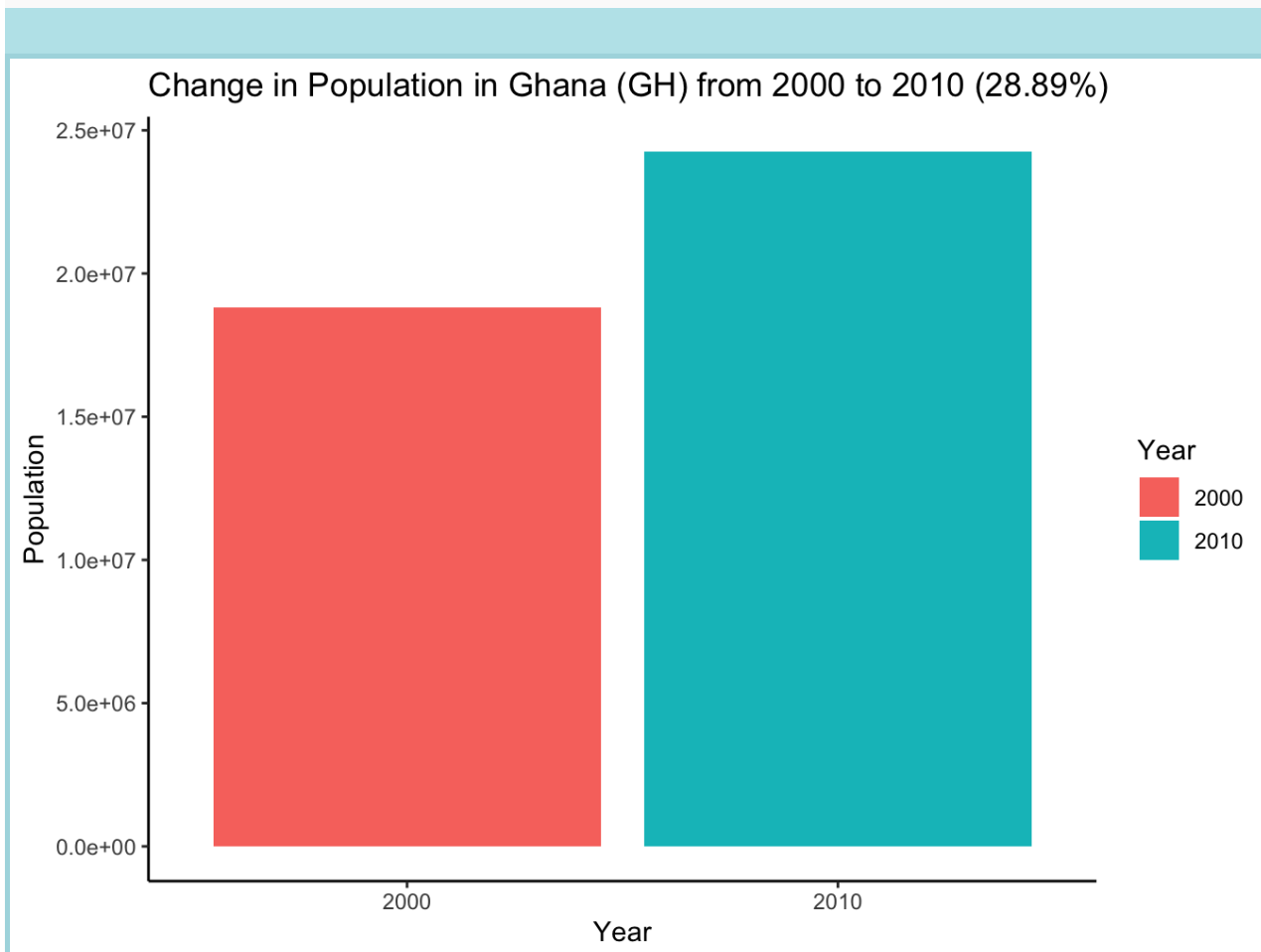
ghana_2010 <- population %>%
  filter(country == "Ghana", year == 2010) %>%
  pull(population)

ghana_population <- data.frame(
  year = c(2000, 2010),
  population = c(ghana_2000, ghana_2010)
)

percent_change <- ((ghana_2010 - ghana_2000) / ghana_2000) * 100

ggplot(ghana_population, aes(
  x = factor(year),
  y = population,
  fill = factor(year)
)) +
  geom_bar(stat = "identity") +
  labs(
    x = "Year",
    y = "Population",
```

```
fill = "Year",
title = paste0("Change in Population in Ghana (GH) from 2000 to 2010 (",
               round(percent_change, 2), "%)",
               theme = theme_classic())
```



This refactored code now refers to "Ghana" instead of "Afghanistan".

Limitations of ChatGPT

While ChatGPT is a powerful tool for data analysts, it has some limitations:

1. **Lag in Learning:** ChatGPT may struggle with newer software or libraries.
2. **Hallucinations:** Always verify the output of ChatGPT as it can sometimes generate outputs that are incorrect or nonsensical.
3. **Limited Input Length:** ChatGPT cannot process very long prompts. To avoid this, start new conversations frequently.
4. **Weak Math Skills:** At the moment, ChatGPT is not ideal for complex calculations or data analysis.