Exercise-05

2024-02-27

CHALLANGE 1

Step 1: Load and read dataset IMDB-movies.csv

```
library (tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
f <- "https://raw.githubusercontent.com/difiore/ada-2024-datasets/main/IMDB-movies.csv"
d <- read_csv(f, col_names = TRUE)
```

```
## Rows: 28938 Columns: 10
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (6): tconst, titleType, primaryTitle, genres, nconst, director
## dbl (4): startYear, runtimeMinutes, averageRating, numVotes
##
## 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.
```

```
attach(d)
ncol(d)
```

```
## [1] 10
```

```
nrow(d)
```

```
## [1] 28938
```

Step 2: Use a one-line statement to filter the dataset to include just movies from 1920 to 1979 and movies that are between 1 and 3 hours long (runtimeMinutes >= 60 and runtimeMinutes <= 180), and add a new column that codes the startYear into a new variable, decade ("20s", "30s", . . . "70s"). If you do this correctly, there should be 5651 movies remaining in the dataset.

```r
filtered_dataset <- d %>%
  filter(startYear >= 1920 & startYear <= 1979, runtimeMinutes >= 60 & runtimeMinutes <= 180) %>%
  mutate(decade = paste0(floor((startYear - 1) / 10) * 10, "s"))
print(filtered_dataset)
```

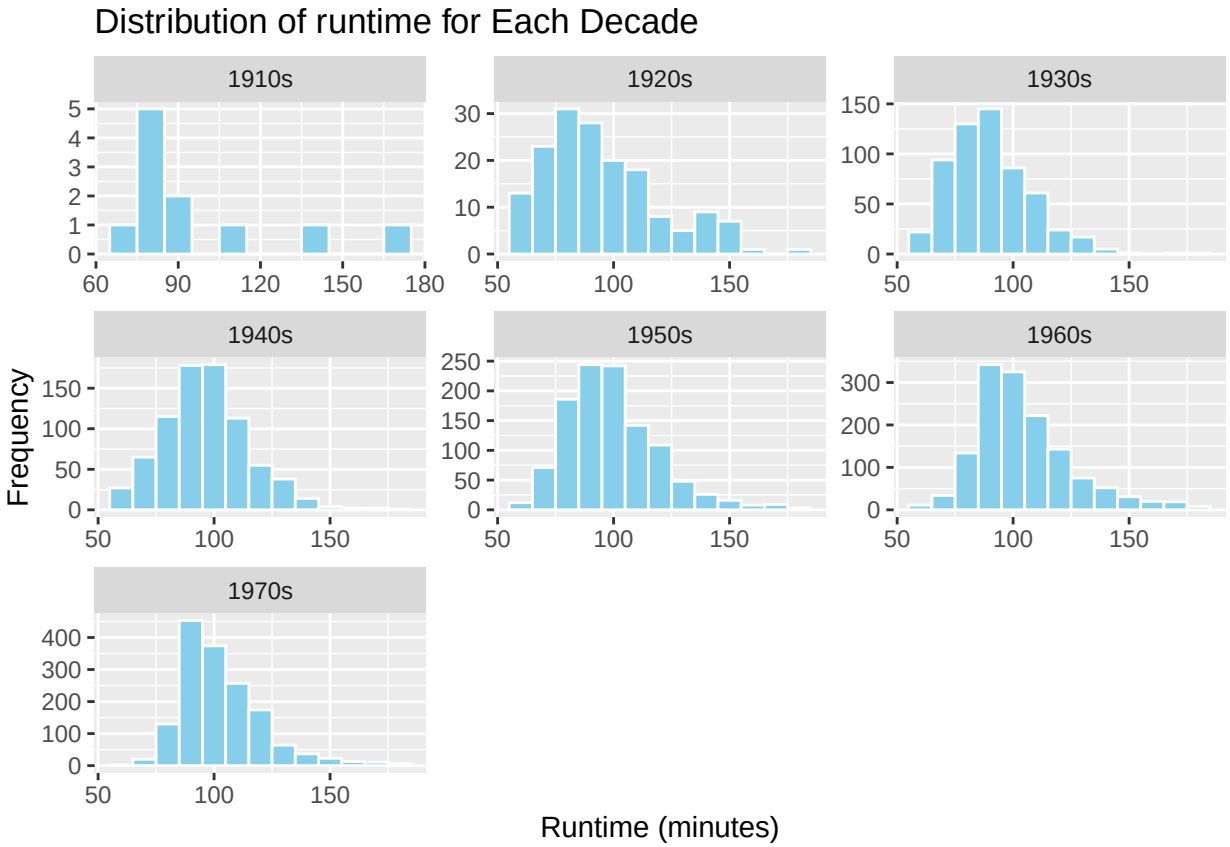
```
## # A tibble: 5,651 x 11
##    tconst   titleType primaryTitle startYear runtimeMinutes genres averageRating
##    <chr>    <chr>     <chr>            <dbl>          <dbl> <chr>          <dbl>
##  1 tt00103~ movie     The Cabinet~      1920             76 Fanta~           8.1
##  2 tt00110~ movie     Leaves From~      1920            167 Drama            6.7
##  3 tt00111~ movie     Dr. Jekyll ~      1920             82 Drama~           7
##  4 tt00112~ movie     The Golem         1920             76 Fanta~           7.2
##  5 tt00113~ movie     The Last of~      1920             73 Actio~           6.7
##  6 tt00114~ movie     The Mark of~      1920            107 Adven~           7.1
##  7 tt00115~ movie     The Penalty       1920             90 Crime~           7.4
##  8 tt00116~ movie     The Saphead       1920             77 Comedy           6.2
##  9 tt00118~ movie     Way Down Ea~      1920            145 Drama~           7.4
## 10 tt00118~ movie     Why Change ~      1920             90 Comed~           6.7
## # i 5,641 more rows
## # i 4 more variables: numVotes <dbl>, nconst <chr>, director <chr>,
## #   decade <chr>
```

```r
nrow(filtered_dataset)
```

```
## [1] 5651
```

Step 3:Use {ggplot2} (which is part of {tidyverse}) to plot histograms of the distribution of runtimeMinutes for each decade.

```r
filtered_dataset %>%
  ggplot(aes(x = runtimeMinutes)) +
  geom_histogram(binwidth = 10, fill = "skyblue", color = "white", position = "identity") +
  facet_wrap(~decade, scales = "free") +
  labs(title = "Distribution of runtime for Each Decade",
       x = "Runtime (minutes)",
       y = "Frequency")
```



Step 4: Use a one-line statement to calculate the population mean and population standard deviation in runtimeMinutes for each decade and save the results in a new dataframe called results.

```
results <- filtered_dataset %>%
  group_by(decade) %>%
  summarize(mean_runtime = mean(runtimeMinutes), sd_runtime = sd(runtimeMinutes))
print(results)
```

```
## # A tibble: 7 x 3
##   decade mean_runtime sd_runtime
##   <chr>         <dbl>      <dbl>
## 1 1910s          96.5       31.3
## 2 1920s          95.8       25.0
## 3 1930s          90.8       17.5
## 4 1940s          97.2       18.5
## 5 1950s         100.        20.0
## 6 1960s         105.        20.9
## 7 1970s         104.        18.0
```

Step 5: Draw a single sample of 100 movies, without replacement, from each decade and calculate the single sample mean and single sample standard deviation in runtimeMinutes for each decades. Recall that your single sample mean for each decade is an estimate of the population mean for each decade.

```
single_sample_results <- filtered_dataset %>%
  group_by(decade) %>%
```

```
  slice_sample(n = 100, replace = FALSE) %>%
  summarize(sample_mean = mean(runtimeMinutes), sample_sd = sd(runtimeMinutes))
print(single_sample_results)
```

```
## # A tibble: 7 x 3
##   decade sample_mean sample_sd
##   <chr>        <dbl>     <dbl>
## 1 1910s         96.5      31.3
## 2 1920s         95.6      24.6
## 3 1930s         91        18.8
## 4 1940s         96.3      18.0
## 5 1950s         98.7      17.4
## 6 1960s        105.       22.5
## 7 1970s        107.       19.8
```

Step 6: Calculate for each decade the standard error around your estimate of the population mean runtimeMinutes based on the standard deviation and sample size (n=100 movies) of your single sample.

```
single_sample_results <- single_sample_results %>%
  mutate(se = sample_sd / sqrt(100))
print(single_sample_results)
```

```
## # A tibble: 7 x 4
##   decade sample_mean sample_sd    se
##   <chr>        <dbl>     <dbl> <dbl>
## 1 1910s         96.5      31.3  3.13
## 2 1920s         95.6      24.6  2.46
## 3 1930s         91        18.8  1.88
## 4 1940s         96.3      18.0  1.80
## 5 1950s         98.7      17.4  1.74
## 6 1960s        105.       22.5  2.25
## 7 1970s        107.       19.8  1.98
```

Step 7: Compare these estimates to the actual population mean runtimeMinutes for each decade and to the calculated SE in the population mean for samples of size 100 based on the population standard deviation for each decade.

```
comparison_results <- left_join(results, single_sample_results, by = "decade") %>%
  mutate(mean_difference = mean_runtime - sample_mean,
         se_difference = mean_difference / se)
print(comparison_results)
```

```
## # A tibble: 7 x 8
##   decade mean_runtime sd_runtime sample_mean sample_sd    se mean_difference
##   <chr>         <dbl>      <dbl>       <dbl>     <dbl> <dbl>           <dbl>
## 1 1910s          96.5       31.3        96.5      31.3  3.13           0
## 2 1920s          95.8       25.0        95.6      24.6  2.46           0.271
## 3 1930s          90.8       17.5        91        18.8  1.88          -0.179
## 4 1940s          97.2       18.5        96.3      18.0  1.80           0.933
## 5 1950s         100.        20.0        98.7      17.4  1.74           1.27
## 6 1960s         105.        20.9       105.       22.5  2.25           0.806
## 7 1970s         104.        18.0       107.       19.8  1.98          -2.78
## # i 1 more variable: se_difference <dbl>
```

Step 8: Generate a sampling distribution of mean runtimeMinutes for each decade by [a] drawing 1000 random samples of 100 movies from each decade, without replacement, and, for each sample, [b] calculating the mean runtimeMinutes and the standard deviation in runtimeMinutes for each decade.

```
library(purrr)
library(mosaic)
```

```
## Registered S3 method overwritten by 'mosaic':
##   method                           from
##   fortify.SpatialPolygonsDataFrame ggplot2
```

```
##
## The 'mosaic' package masks several functions from core packages in order to add
## additional features.  The original behavior of these functions should not be affected by this.
```

```
##
## Attaching package: 'mosaic'
```

```
## The following object is masked from 'package:Matrix':
##
##     mean
```

```
## The following objects are masked from 'package:dplyr':
##
##     count, do, tally
```

```
## The following object is masked from 'package:purrr':
##
##     cross
```

```
## The following object is masked from 'package:ggplot2':
##
##     stat
```

```
## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cor.test, cov, fivenum, IQR, median, prop.test,
##     quantile, sd, t.test, var
```

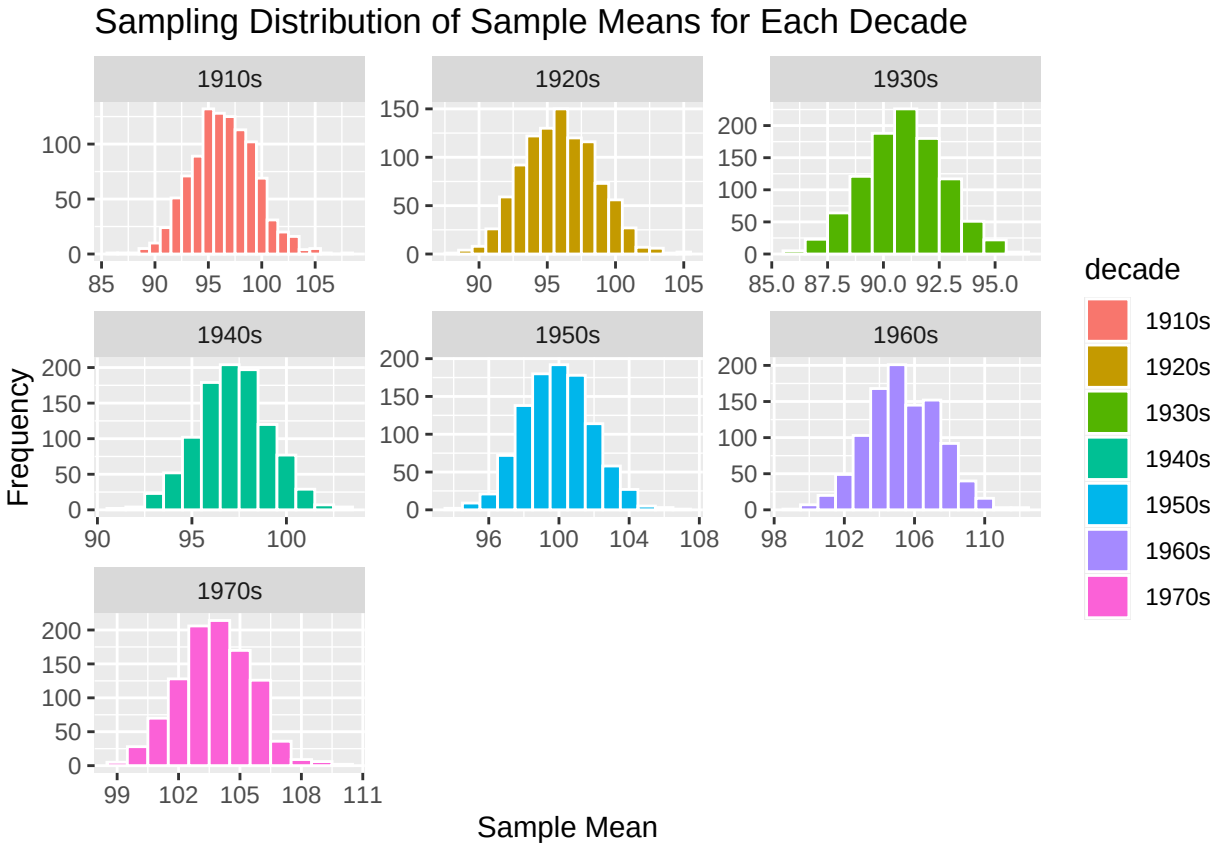
```
## The following objects are masked from 'package:base':
##
##     max, mean, min, prod, range, sample, sum
```

```
reps <- 1000
sampling_distributions <- do(reps) * {
  filtered_dataset %>%
    group_by(decade) %>%
    slice(sample(seq(n()), size = 100, replace = TRUE)) %>%
    summarize(sample_mean = mean(runtimeMinutes), sample_sd = sd(runtimeMinutes))
}
sampling_distributions
```

```
## # A tibble: 7,000 x 5
##    decade sample_mean sample_sd  .row .index
##    <chr>        <dbl>     <dbl> <int>  <dbl>
##  1 1910s         99.8      31.0     1      1
##  2 1920s         97.7      26.0     2      1
##  3 1930s         92.8      17.6     3      1
##  4 1940s         94.7      18.7     4      1
##  5 1950s        100.       21.6     5      1
##  6 1960s        107.       22.3     6      1
##  7 1970s        101.       16.0     7      1
##  8 1910s         99.1      31.4     1      2
##  9 1920s         98.4      26.6     2      2
## 10 1930s         90.2      16.9     3      2
## # i 6,990 more rows
```

Step 9: Then, calculate the mean and the standard deviation of the sampling distribution of sample means for each decade (the former should be a very good estimate of the population mean, while the latter is another estimate of the standard error in our estimate of the population mean for a particular sample size) and plot a histogram of the sampling distribution for each decade. What shape does it have?

```
library(ggplot2)
sampling_means <- sampling_distributions %>%
  group_by(decade) %>%
  summarize(sampling_mean_mean = mean(sample_mean),
            sampling_mean_sd = sd(sample_mean))
ggplot(sampling_distributions, aes(x = sample_mean, fill = decade)) +
  geom_histogram(binwidth = 1, color = "white", position = "identity") +
  facet_wrap(~decade, scales = "free") +
  labs(title = "Sampling Distribution of Sample Means for Each Decade",
       x = "Sample Mean",
       y = "Frequency")
```



The shapes are normally distributed curve (Bell curve).

Step 10:Finally, compare the standard error in runtimeMinutes for samples of size 100 from each decade [1] as estimated from your first sample of 100 movies, [2] as calculated from the known population standard deviations for each decade, and [3] as estimated from the sampling distribution of sample means for each decade.

```
# Standard error from the first sample of 100 movies
se_from_first_sample <- filtered_dataset %>%
  group_by(decade) %>%
  slice_sample(n = 100, replace = FALSE) %>%
  summarize(se_from_first_sample = sd(runtimeMinutes) / sqrt(100))

# Merge with results for comparison
results_comparison <- left_join(results, se_from_first_sample, by = "decade")

# Standard error calculated from known population standard deviations
results_comparison <- results_comparison %>%
  mutate(se_from_known_population_sd = sd_runtime / sqrt(100))

# Standard error estimated from the sampling distribution of sample means
results_comparison <- left_join(results_comparison, sampling_means, by = "decade") %>%
  mutate(se_from_sampling_distribution = sampling_mean_sd)

print(results_comparison[, c("decade", "se_from_first_sample", "se_from_known_population_sd", "se_from_
```

```
## # A tibble: 7 x 4
```

```
##   decade se_from_first_sample se_from_known_population_sd se_from_sampling_dis~1
##   <chr>                 <dbl>                       <dbl>                   <dbl>
## 1 1910s                  3.13                        3.13                    2.97
## 2 1920s                  2.52                        2.50                    2.64
## 3 1930s                  1.64                        1.75                    1.77
## 4 1940s                  1.86                        1.85                    1.90
## 5 1950s                  1.85                        2.00                    1.95
## 6 1960s                  2.07                        2.09                    2.06
## 7 1970s                  1.79                        1.80                    1.78
## # i abbreviated name: 1: se_from_sampling_distribution
```

CHALLANGE 2

Step 1:Using the {tidyverse} read_csv() function, load the "zombies.csv" dataset from this URL as a "tibble" named z.

```
library (tidyverse)
f <- "https://raw.githubusercontent.com/difiore/ada-2024-datasets/main/zombies.csv"
z <- read_csv(f, col_names = TRUE)
```

```
## Rows: 1000 Columns: 10
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): first_name, last_name, gender, major
## dbl (6): id, height, weight, zombies_killed, years_of_education, age
##
## 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.
```

```
attach(z)
ncol(z)
```

```
## [1] 10
```

```
nrow(z)
```

```
## [1] 1000
```

Step 2: Calculate the population mean and standard deviation for each quantitative random variable in the dataset (height, weight, age, number of zombies killed, and years of education).

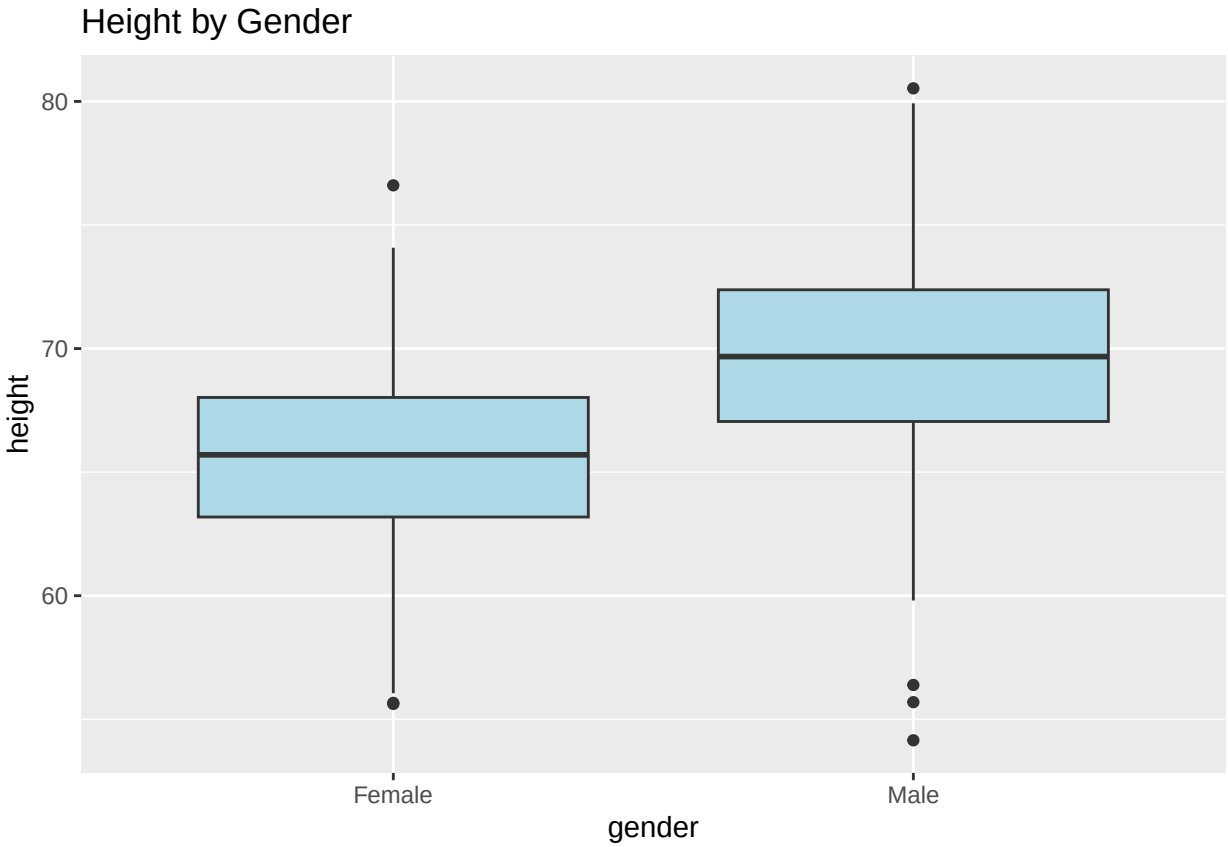
```
pop_mean <- function(x) sum(x, na.rm = TRUE) / length(x)
pop_sd <- function(x) sqrt(sum((x - pop_mean(x))^2, na.rm = TRUE) / length(x))
summary_stats <- z %>%
  summarize(
    mean_height = pop_mean(height),
    sd_height = pop_sd(height),
    mean_weight = pop_mean(weight),
    sd_weight = pop_sd(weight),
    mean_age = pop_mean(age),
    sd_age = pop_sd(age),
    mean_zombies_killed = pop_mean(zombies_killed),
```

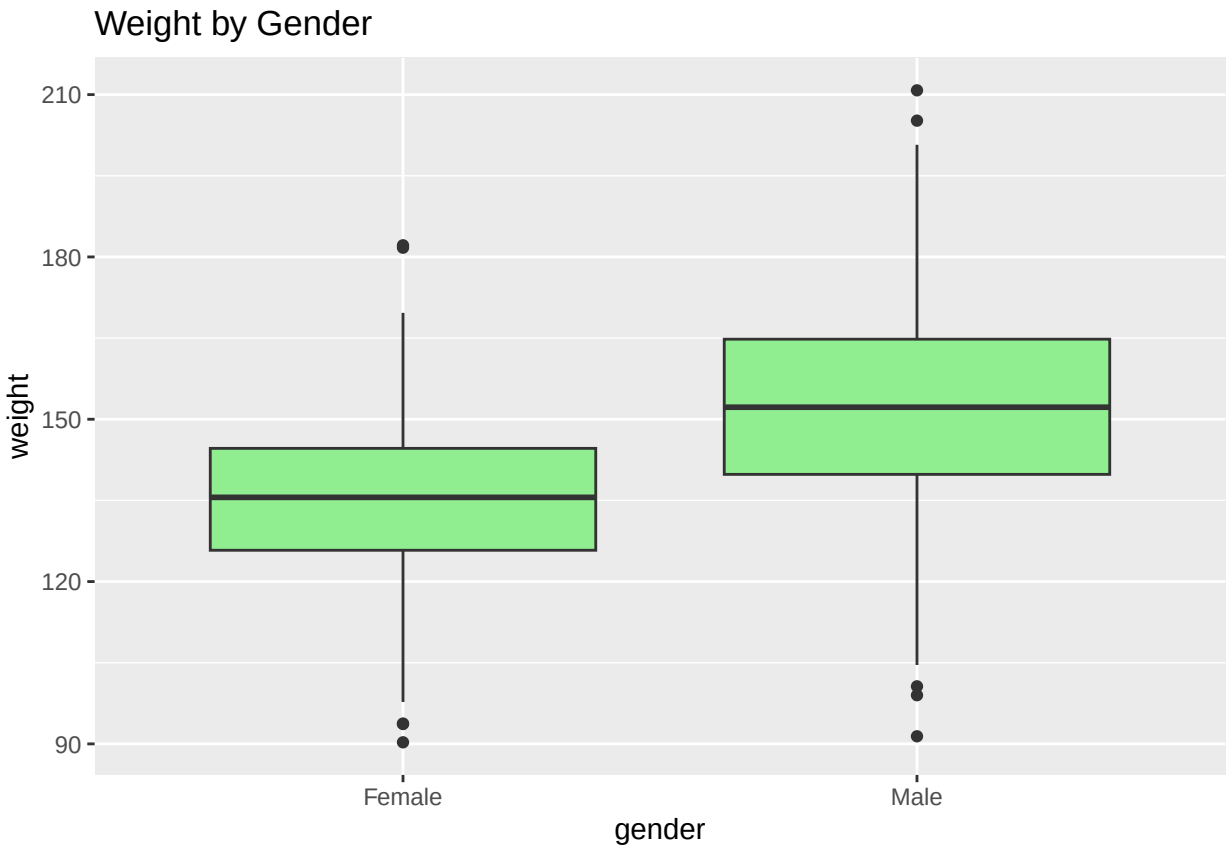
```
    sd_zombies_killed = pop_sd(zombies_killed),
    mean_years_of_education = pop_mean(years_of_education),
    sd_years_of_education = pop_sd(years_of_education)
  )

print(summary_stats)
```

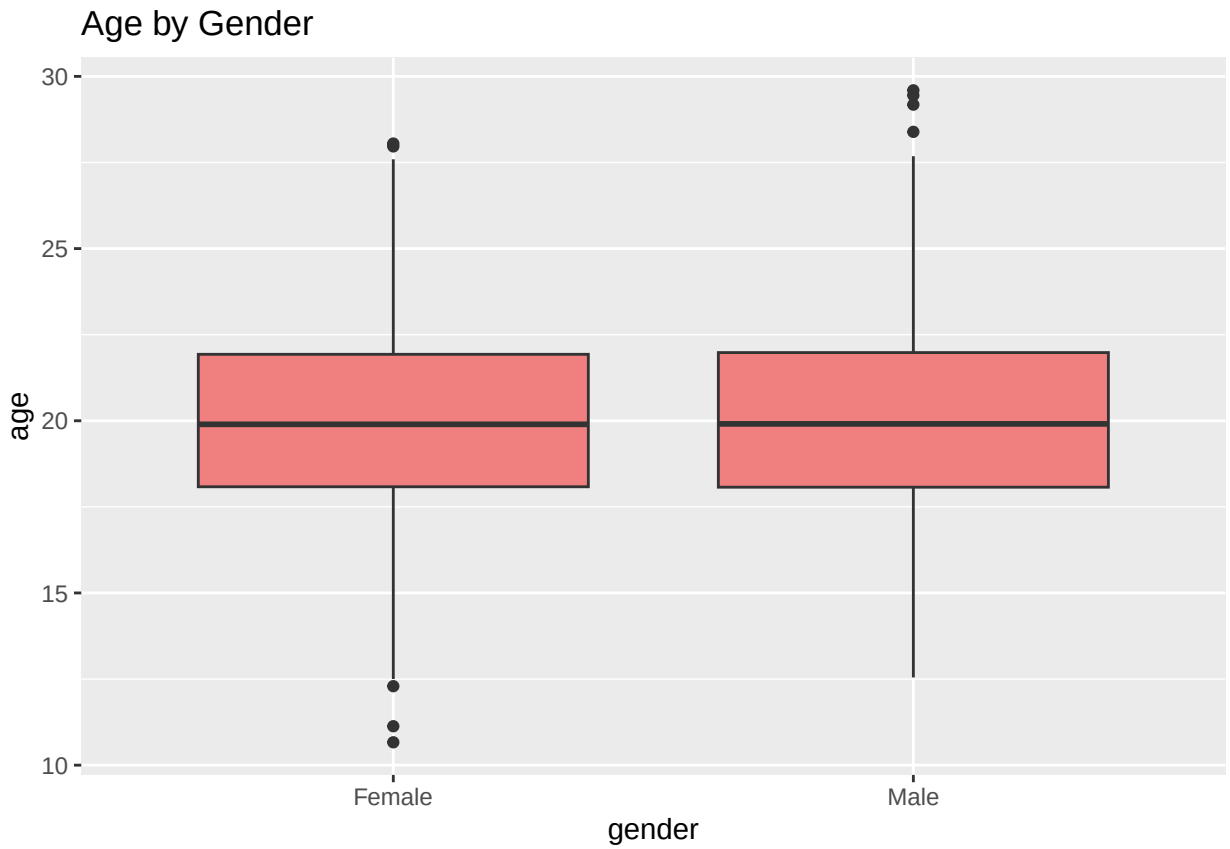
```
## # A tibble: 1 x 10
##   mean_height sd_height mean_weight sd_weight mean_age sd_age
##         <dbl>     <dbl>       <dbl>     <dbl>    <dbl>  <dbl>
## 1        67.6      4.31        144.      18.4     20.0   2.96
## # i 4 more variables: mean_zombies_killed <dbl>, sd_zombies_killed <dbl>,
## #   mean_years_of_education <dbl>, sd_years_of_education <dbl>
```

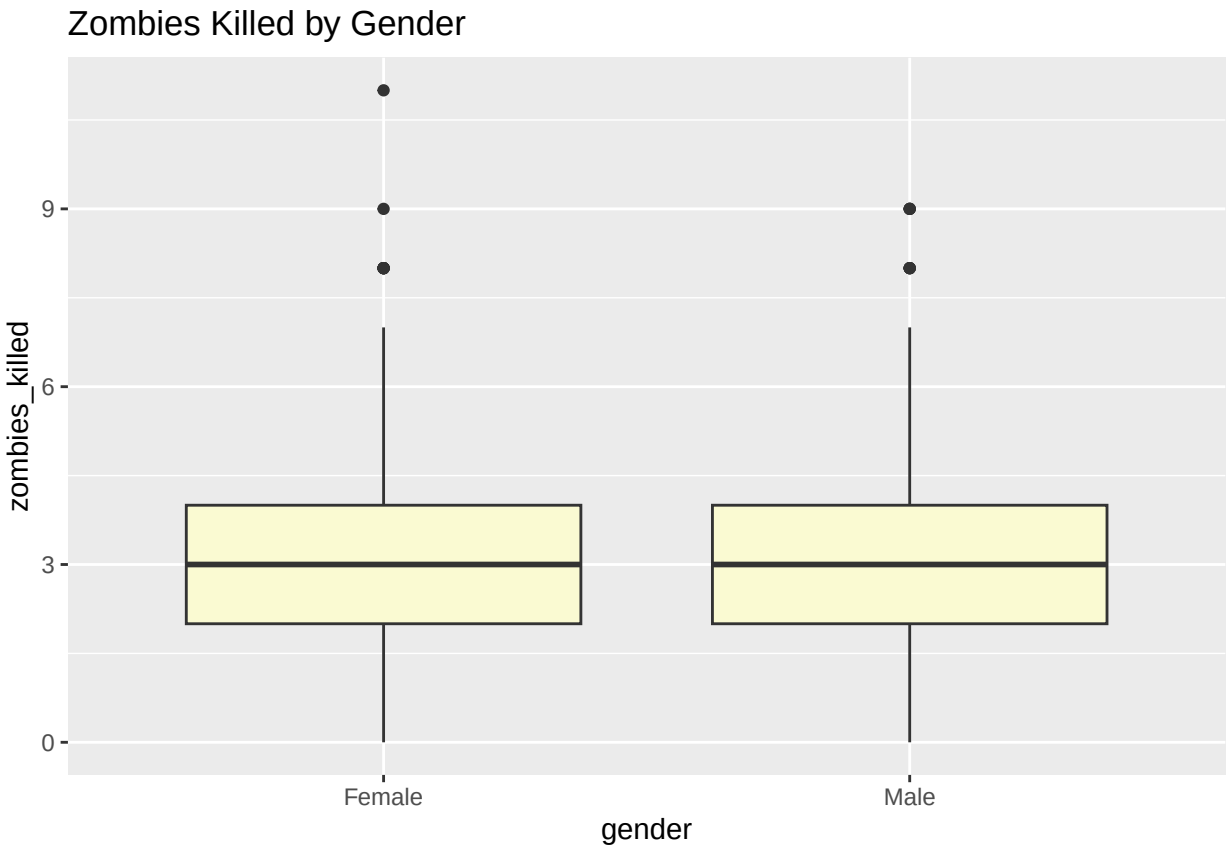
Step 3:Use {ggplot} and make boxplots of each of these variables by gender.

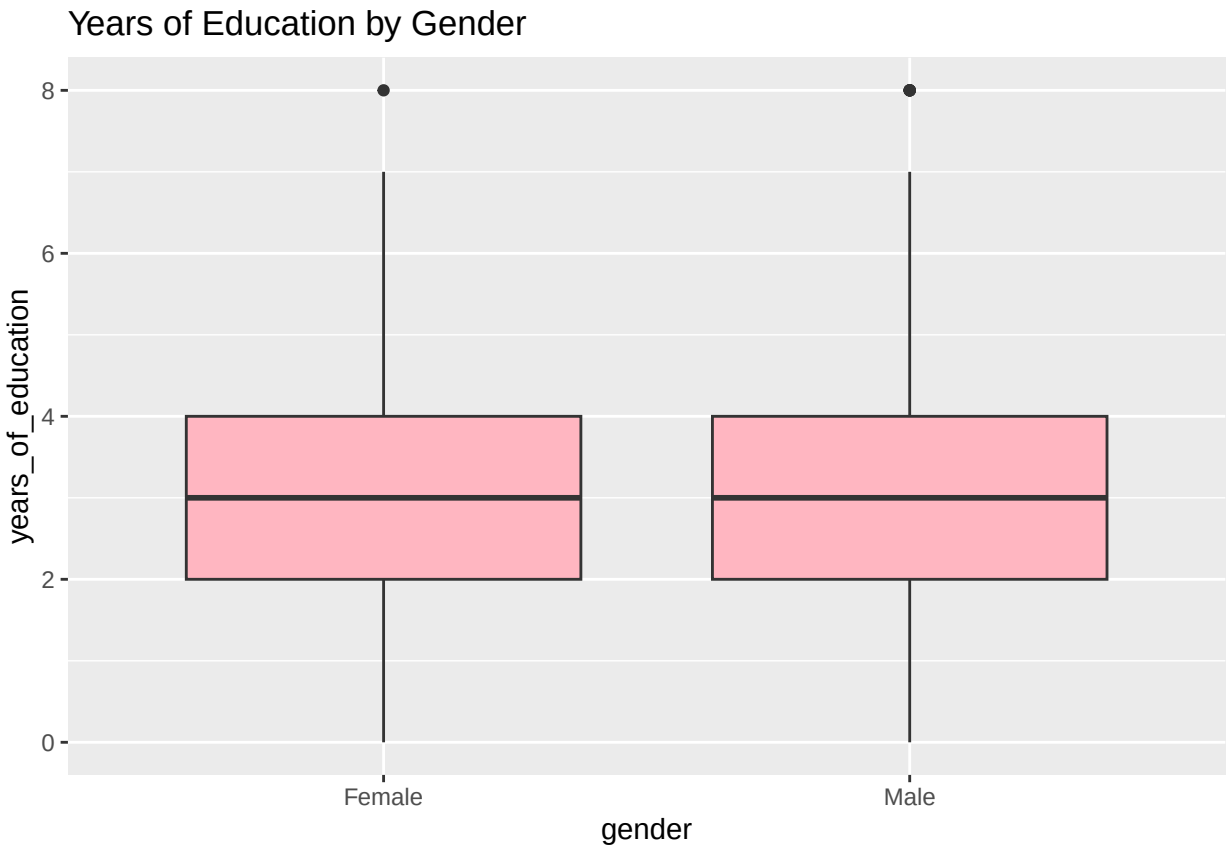
```
library(ggplot2)
boxplot_plots <- list(
  ggplot(z, aes(x = gender, y = height)) + geom_boxplot(fill = "lightblue") + ggtitle("Height by Gender
  ggplot(z, aes(x = gender, y = weight)) + geom_boxplot(fill = "lightgreen") + ggtitle("Weight by Gende
  ggplot(z, aes(x = gender, y = age)) + geom_boxplot(fill = "lightcoral") + ggtitle("Age by Gender"),
  ggplot(z, aes(x = gender, y = zombies_killed)) + geom_boxplot(fill = "lightgoldenrodyellow") + ggtitl
  ggplot(z, aes(x = gender, y = years_of_education)) + geom_boxplot(fill = "lightpink") + ggtitle("Years
)

# Display the boxplots
for (plot in boxplot_plots) {
  print(plot)
}
```


Height by Gender
80
70
60
height
Female
Male
gender


Weight by Gender
210
180
150
120
90
weight
Female
Male
gender


Age by Gender
30
25
20
15
10
age
Female
Male
gender

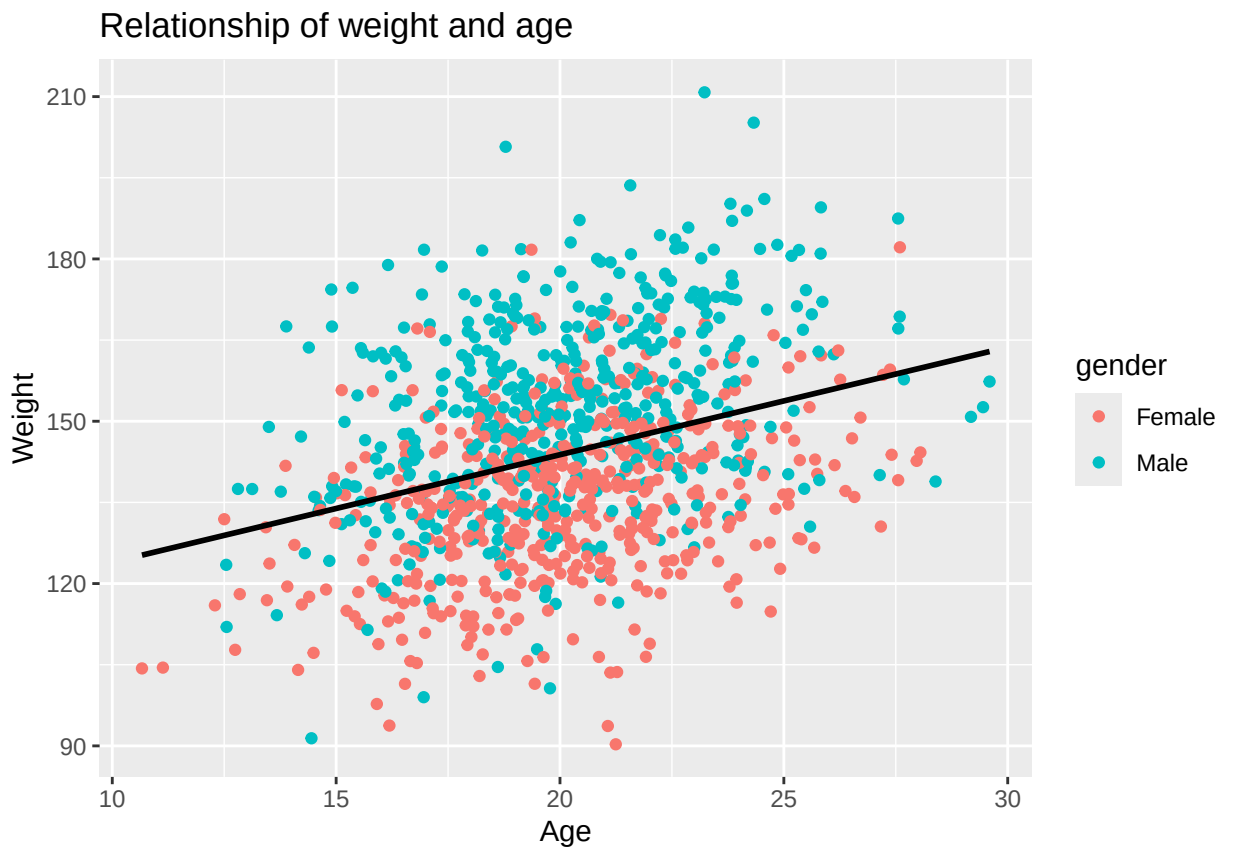

Zombies Killed by Gender
zombies_killed
9
6
3
0
Female
Male
gender



Step 4:Use {ggplot} and make scatterplots of height and weight in relation to age (i.e., use age as the variable), using different colored points for males versus females. Do these variables seem to be related? In what way?

```
scatterplot_height <- ggplot(z, aes(x = age, y = height, color = gender)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "black", formula = y ~ x) +
  ggtitle("Relationship of height and age") +
  xlab("Age") +
  ylab("Height")

scatterplot_weight <- ggplot(z, aes(x = age, y = weight, color = gender)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "black", formula = y ~ x) +
  ggtitle("Relationship of weight and age") +
  xlab("Age") +
  ylab("Weight")

print(scatterplot_height)
```


```
print(scatterplot_weight)
```



Yes, these variables are related. There is positive linear relationship of weight and weight with age (As age increases, height and weight also increases)

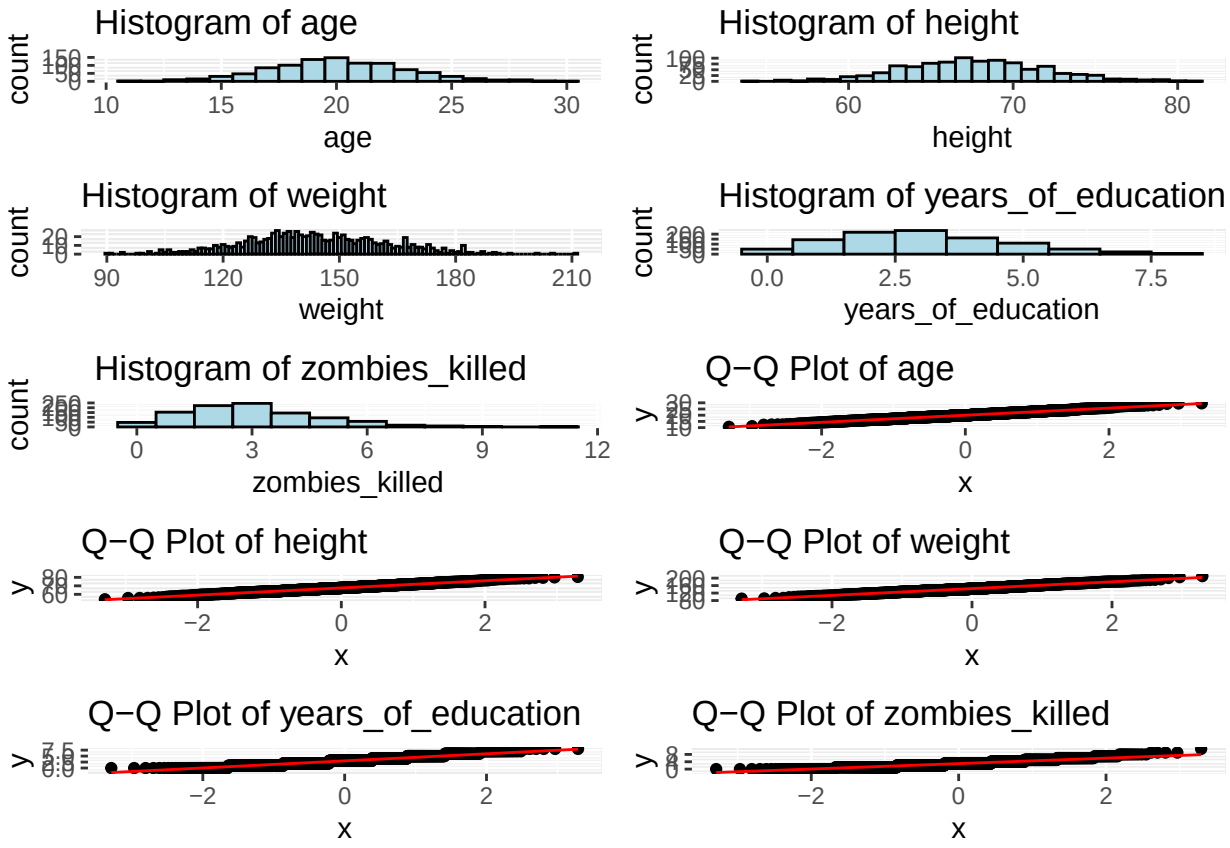
Step 5: Using histograms and Q-Q plots, check whether each of the quantitative variables seem to be drawn from a normal distribution. Which seem to be and which do not?

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
create_distribution_plots <- function(data, variable) {
  # Histogram
  hist_plot <- ggplot(data, aes(x = !!sym(variable))) +
    geom_histogram(binwidth = 1, fill = "lightblue", color = "black") +
    ggtitle(paste("Histogram of", variable))

  # Q-Q plot
  qq_plot <- ggplot(data, aes(sample = !!sym(variable))) +
    geom_qq() +
    geom_qq_line(color = "red") +
    ggtitle(paste("Q-Q Plot of", variable))
```

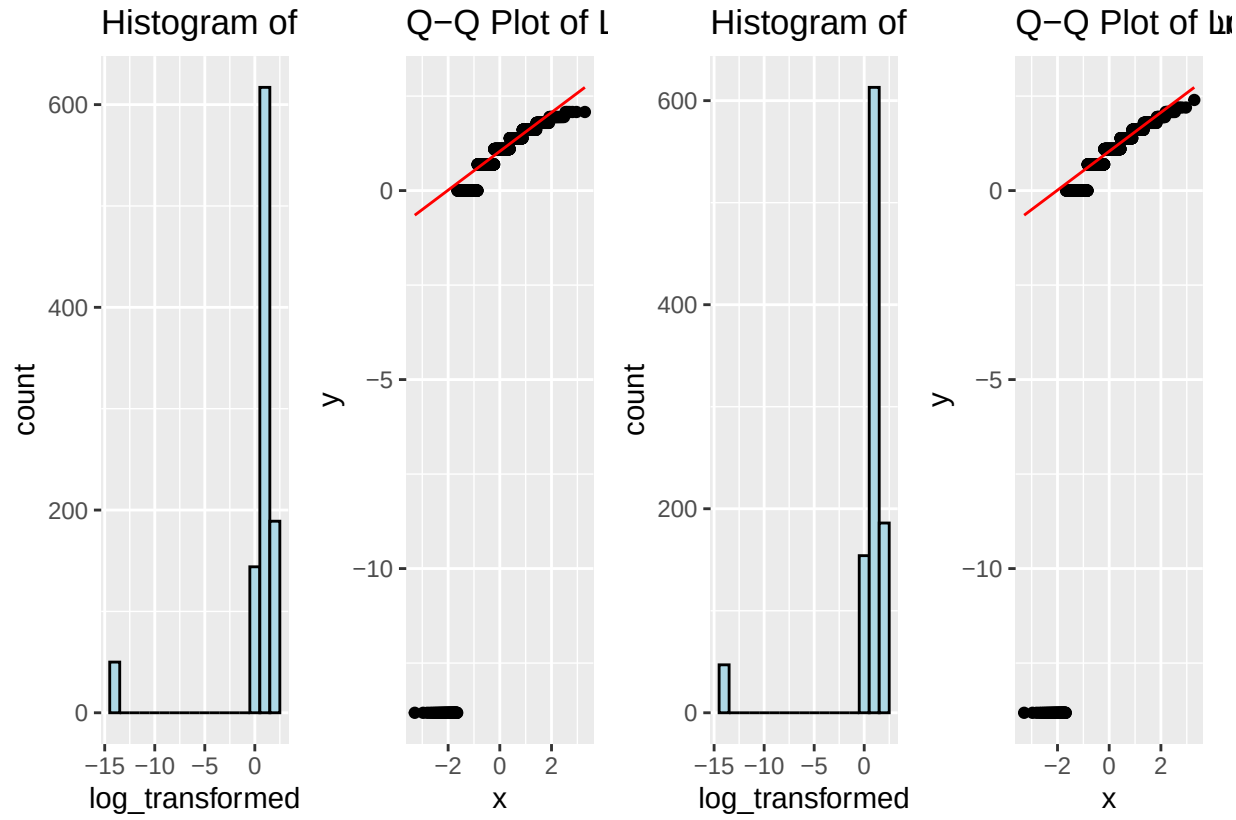
```r
  return(list(hist_plot = hist_plot, qq_plot = qq_plot))
}

quantitative_variables <- c("age", "height", "weight", "years_of_education", "zombies_killed")
plots_list <- lapply(quantitative_variables, function(var) create_distribution_plots(z, var))
hist_plots <- lapply(plots_list, function(plots) plots$hist_plot)
qq_plots <- lapply(plots_list, function(plots) plots$qq_plot)
grid.arrange(grobs = c(hist_plots, qq_plots), ncol = 2)
```



Age, height, and weight exhibit a normal distribution, whereas years of education and zombies killed display positive skewness in their distributions. To assess normality, log transformation can be applied to the latter two variables.

```r
library(ggplot2)
library(patchwork)
# Function to create histogram and Q-Q plot with log transformation
create_distribution_plots <- function(data, variable) {
  # Add a small constant to avoid log transformation issues
  epsilon <- 1e-6
  data$log_transformed <- log(data[[variable]] + epsilon)

  # Histogram
  hist_plot <- ggplot(data, aes(x = log_transformed)) +
    geom_histogram(binwidth = 1, fill = "lightblue", color = "black") +
    ggtitle(paste("Histogram of Log-transformed", variable))
```

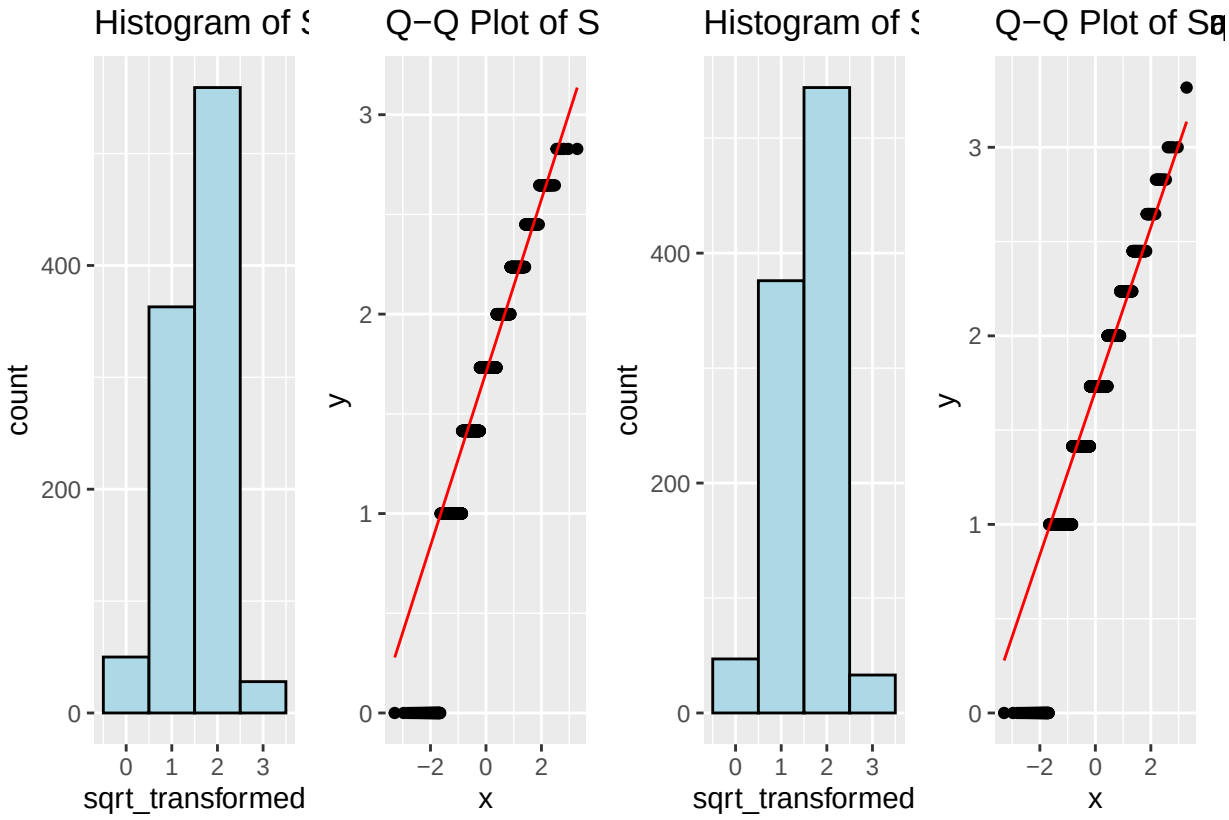
```r
  # Q-Q plot
  qq_plot <- ggplot(data, aes(sample = log_transformed)) +
    geom_qq() +
    geom_qq_line(color = "red") +
    ggtitle(paste("Q-Q Plot of Log-transformed", variable))
  combined_plot <- hist_plot + qq_plot
  return(combined_plot)
}

skewed_variables <- c("years_of_education", "zombies_killed")
combined_plots_list <- lapply(skewed_variables, function(var) create_distribution_plots(z, var))
combined_plots <- wrap_plots(combined_plots_list, ncol = 2)
print(combined_plots)
```



In this scenario, log-transformation of the data did not contribute to achieving a normal distribution. An alternative approach involves attempting a square root transformation as a different method of data transformation.

```r
library(ggplot2)
library(patchwork)

# Function to create histogram and Q-Q plot with square root transformation
create_distribution_plots_sqrt <- function(data, variable) {
  data$sqrt_transformed <- sqrt(data[[variable]])
```

```r
  # Histogram
  hist_plot <- ggplot(data, aes(x = sqrt_transformed)) +
    geom_histogram(binwidth = 1, fill = "lightblue", color = "black") +
    ggtitle(paste("Histogram of Square Root Transformed", variable))

  # Q-Q plot
  qq_plot <- ggplot(data, aes(sample = sqrt_transformed)) +
    geom_qq() +
    geom_qq_line(color = "red") +
    ggtitle(paste("Q-Q Plot of Square Root Transformed", variable))

  combined_plot <- hist_plot + qq_plot
  return(combined_plot)
}

# Apply the function to the specified variables
skewed_variables <- c("years_of_education", "zombies_killed")
combined_plots_list_sqrt <- lapply(skewed_variables, function(var) create_distribution_plots_sqrt(z, va
combined_plots_sqrt <- wrap_plots(combined_plots_list_sqrt, ncol = 2)
print(combined_plots_sqrt)
```



The implementation of the square root transformation has successfully resulted in a normal distribution for the variable

Step 6: Now use the `sample_n()` or `slice_sample()` function from {dplyr} to sample ONE subset of 50 zombie apocalypse survivors (without replacement) from this population and calculate the mean and sample

standard deviation for each variable. Also estimate the standard error for each variable based on this one sample and use that to construct a theoretical 95% confidence interval for each mean. You can use either the standard normal *or* a Student's t distribution to derive the critical values needed to calculate the lower and upper limits of the CI.

```
sampled_data <- z %>%
  sample_n(50, replace = FALSE)

# Calculate the mean and sample standard deviation for each variable
summary_stats_sample <- sampled_data %>%
  summarise(
    mean_height = mean(height),
    sd_height = sd(height),
    mean_weight = mean(weight),
    sd_weight = sd(weight),
    mean_age = mean(age),
    sd_age = sd(age),
    mean_zombies_killed = mean(zombies_killed),
    sd_zombies_killed = sd(zombies_killed),
    mean_years_of_education = mean(years_of_education),
    sd_years_of_education = sd(years_of_education)
  )
print(summary_stats_sample)
```

```
## # A tibble: 1 x 10
##   mean_height sd_height mean_weight sd_weight mean_age sd_age
##         <dbl>     <dbl>       <dbl>     <dbl>    <dbl>  <dbl>
## 1        67.7      3.52        145.      15.7     19.5   2.47
## # i 4 more variables: mean_zombies_killed <dbl>, sd_zombies_killed <dbl>,
## #   mean_years_of_education <dbl>, sd_years_of_education <dbl>
```

```
# Calculate the standard error for each variable
# Function to calculate standard error for a variable
calculate_se <- function(x) {
  se_value <- sd(x, na.rm = TRUE) / sqrt(sum(!is.na(x)))
  return(se_value)
}

# Specify the variables to calculate standard errors
variables_of_interest <- c("height", "age", "weight", "years_of_education", "zombies_killed")

# Apply the function to calculate standard error for specific variables
se_sample <- sampled_data %>%
  summarise(across(all_of(variables_of_interest), calculate_se, .names = "se_{.col}"))
print(se_sample)
```

```
## # A tibble: 1 x 5
##   se_height se_age se_weight se_years_of_education se_zombies_killed
##       <dbl>  <dbl>     <dbl>                 <dbl>             <dbl>
## 1     0.498  0.349      2.22                 0.284             0.253
```

```r
# Calculate the critical values for a 95% confidence interval using t-distribution
t_critical_value <- qt(0.975, df = 49)  # degrees of freedom: 50 - 1 = 49
print(t_critical_value)
```

```
## [1] 2.009575
```

```r
# Calculate the confidence intervals for each variable

confidence_intervals <- se_sample %>%
  mutate(across(starts_with("se_"),
                list(
                  lower_limit = ~. - t_critical_value * .,
                  upper_limit = ~. + t_critical_value * .
                ),
                .names = "{.col}_{.fn}"
  ))

# Combine results into a data frame
confidence_interval_results <- cbind(summary_stats_sample, confidence_intervals)
print(confidence_interval_results)
```

```
##   mean_height sd_height mean_weight sd_weight mean_age   sd_age
## 1    67.72688  3.521235    144.6639  15.66641 19.46481 2.469598
##   mean_zombies_killed sd_zombies_killed mean_years_of_education
## 1                2.94          1.788968                    3.14
##   sd_years_of_education se_height    se_age se_weight se_years_of_education
## 1               2.01028 0.4979778 0.3492539  2.215565             0.2842965
##   se_zombies_killed se_height_lower_limit se_height_upper_limit
## 1         0.2529983            -0.5027461              1.498702
##   se_age_lower_limit se_age_upper_limit se_weight_lower_limit
## 1         -0.3525981           1.051106             -2.236779
##   se_weight_upper_limit se_years_of_education_lower_limit
## 1              6.667909                         -0.2870187
##   se_years_of_education_upper_limit se_zombies_killed_lower_limit
## 1                         0.8556117                    -0.2554209
##   se_zombies_killed_upper_limit
## 1                     0.7614176
```

Step 7: Then draw another 199 random samples of 50 zombie apocalypse survivors out of the population and calculate the mean for each of the these samples. Together with the first sample you drew out, you now have a set of 200 means for each variable (each of which is based on 50 observations), which constitutes a sampling distribution for each variable. What are the means and standard deviations of the **sampling distribution** for each variable? How do the standard deviations of the sampling distribution for each variable compare to the standard errors estimated from your first sample of size 50?

```r
# Set the seed for reproducibility
set.seed(123)

# Number of additional samples
num_samples <- 199

# Number of survivors in each sample
```

```r
sample_size <- 50

# Function to calculate mean for each sample
calculate_sample_mean <- function(sample_data) {
  return(data.frame(
    height = mean(sample_data$height),
    weight = mean(sample_data$weight),
    age = mean(sample_data$age),
    zombies_killed = mean(sample_data$zombies_killed),
    years_of_education = mean(sample_data$years_of_education)
  ))
}

# Draw additional samples and calculate means
additional_means <- replicate(num_samples, {
  sampled_data <- sample_n(z, sample_size, replace = FALSE)
  calculate_sample_mean(sampled_data)
}, simplify = FALSE)

# Combine the first sample with the additional samples
all_means <- c(summary_stats_sample, additional_means)

# Calculate means and standard deviations of the sampling distribution for each variable
sampling_distribution_means <- colMeans(do.call(rbind, all_means))
sampling_distribution_sds <- apply(do.call(rbind, all_means), 2, sd)
print(sampling_distribution_means)
```

```
##             height             weight                age     zombies_killed
##          65.625278         138.094840          20.364482           4.093934
## years_of_education
##           4.126469
```

```r
print(sampling_distribution_sds)
```

```
##             height             weight                age     zombies_killed
##          13.069342          27.020695           9.721287          10.843220
## years_of_education
##          10.839026
```

Standard deviation are higher in the second draw than the first one.

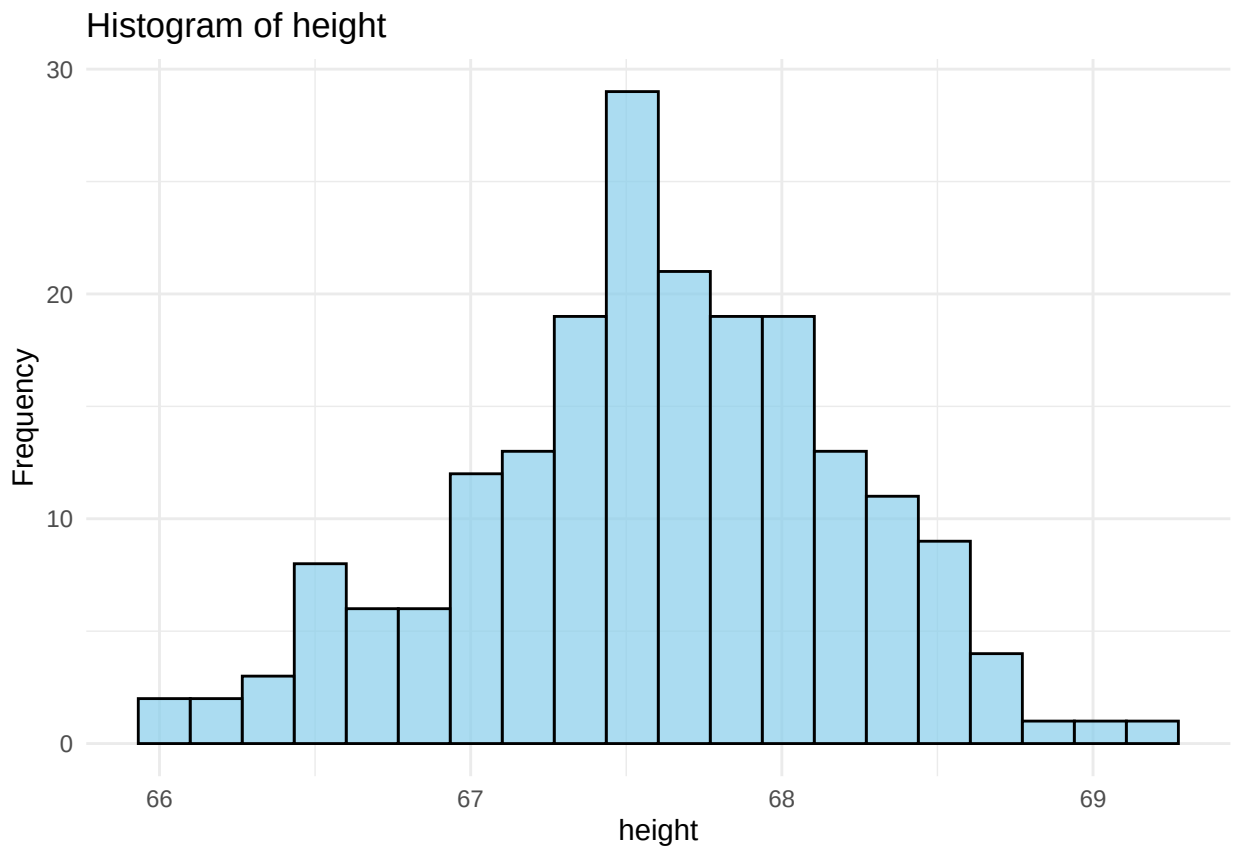
Step 8: Plot the sampling distributions for each variable mean. What do they look like? Are they normally distributed? What about for those variables that you concluded were not originally drawn from a normal distribution?

```r
library(ggplot2)
library(dplyr)

# Combine means from the first sample with the additional samples
all_means_df <- bind_rows(summary_stats_sample, additional_means)

# Function to plot histogram and normal distribution curve
```

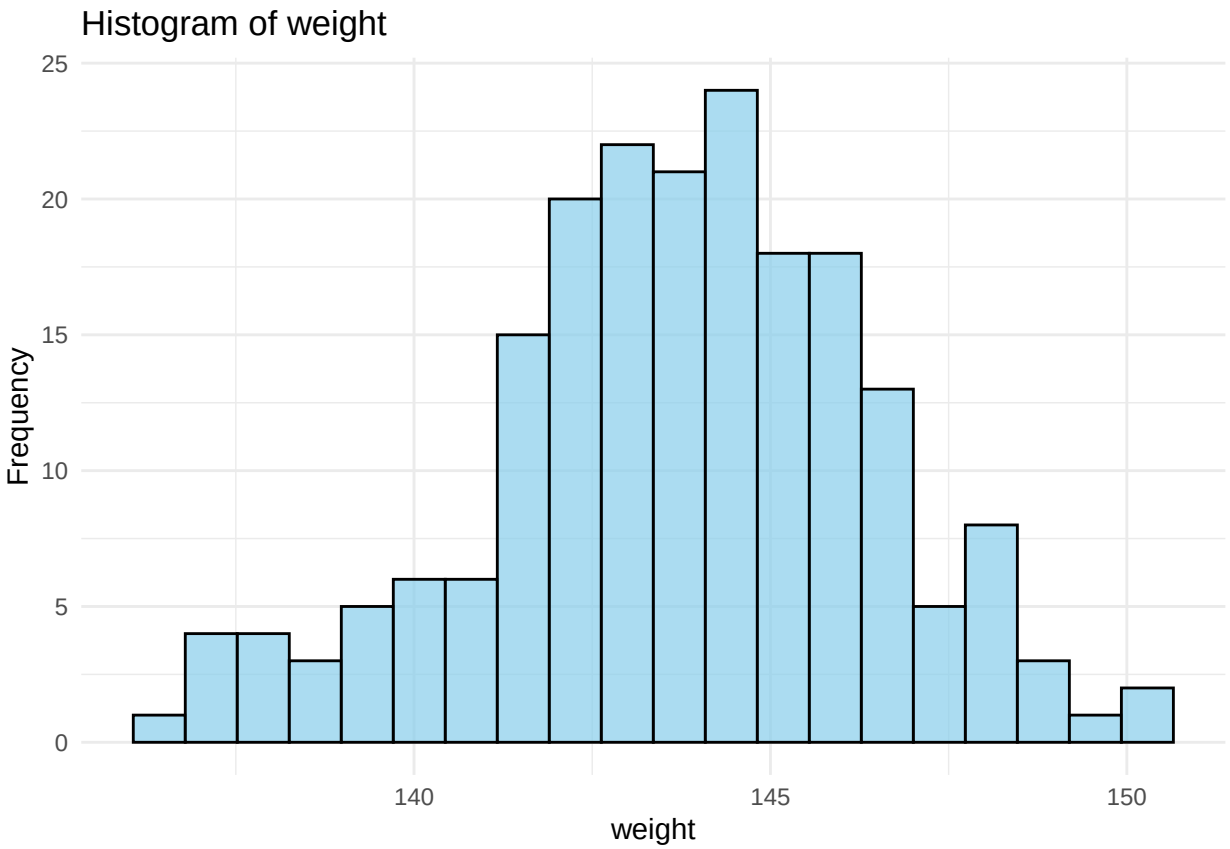
```r
plot_histogram <- function(data, variable) {
  ggplot(data, aes(x = !!sym(variable))) +
    geom_histogram(fill = "skyblue", color = "black", bins = 20, alpha = 0.7) +
    stat_function(fun = dnorm, args = list(mean = mean(data[[variable]]), sd = sd(data[[variable]])), co
    labs(title = paste("Histogram of", variable),
         x = variable,
         y = "Frequency") +
    theme_minimal()
}

# Plot histograms and normal distribution curves for each variable
for (variable in c("height", "weight", "age", "zombies_killed", "years_of_education")) {
  print(plot_histogram(all_means_df, variable))
}
```

```
## Warning: Removed 1 row containing non-finite outside the scale range
## (`stat_bin()`).
```

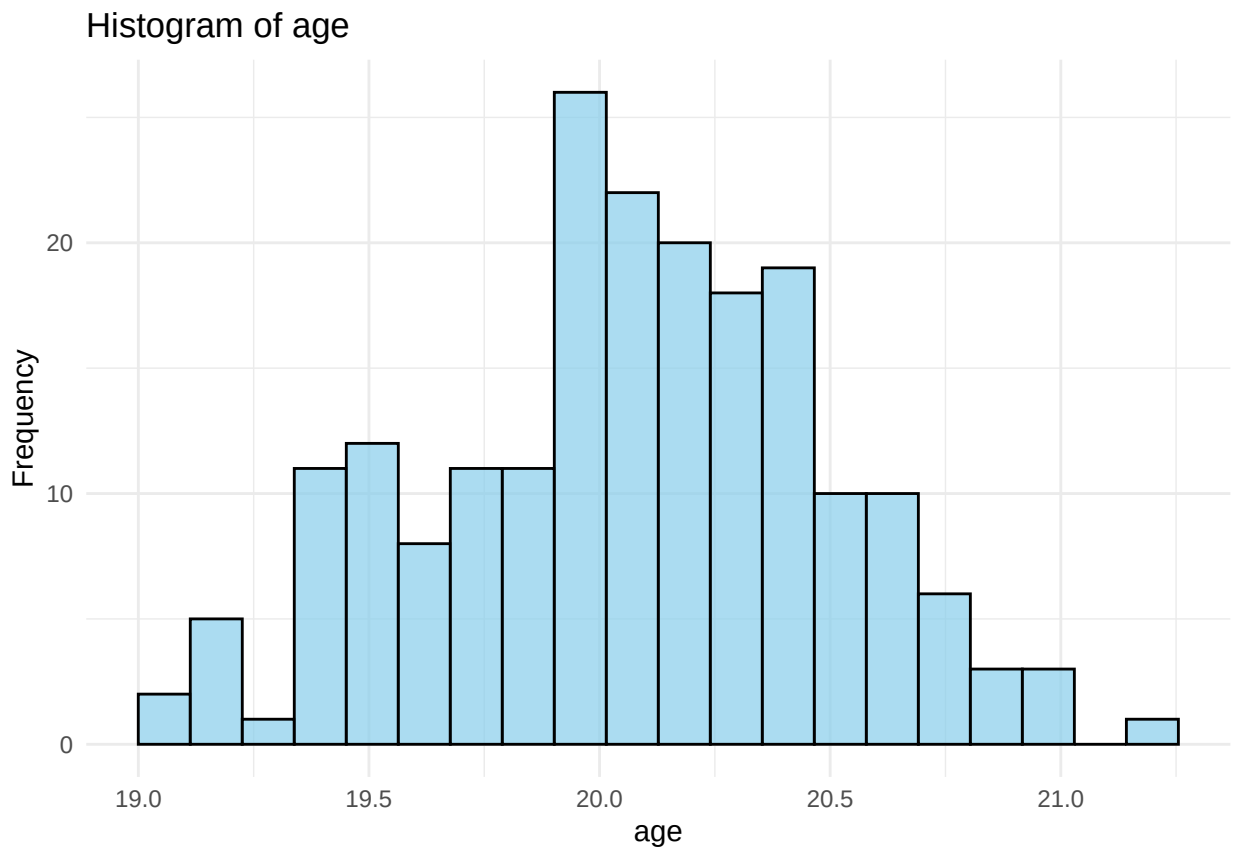
```
## Warning: Removed 101 rows containing missing values or values outside the scale range
## (`geom_function()`).
```



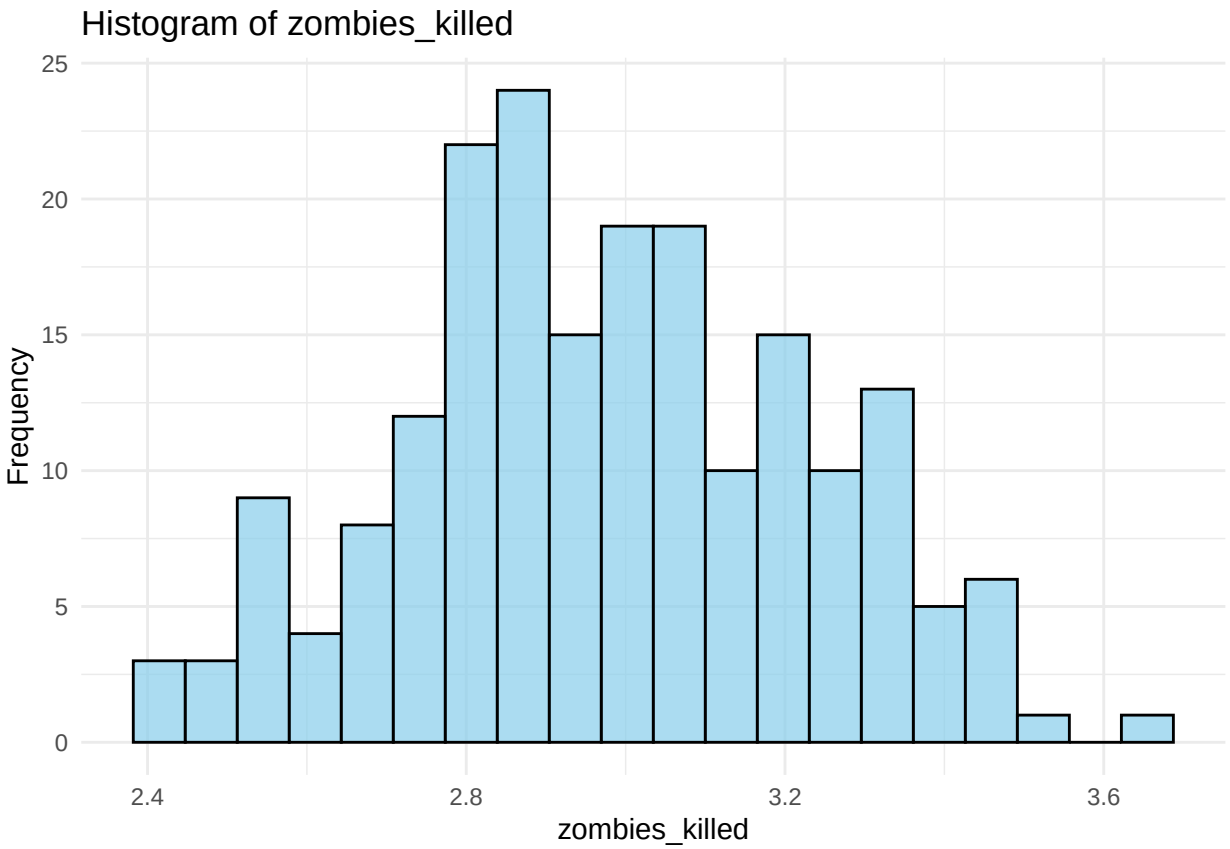

```
## Warning: Removed 1 row containing non-finite outside the scale range (`stat_bin()`).
## Removed 101 rows containing missing values or values outside the scale range
## (`geom_function()`).
```



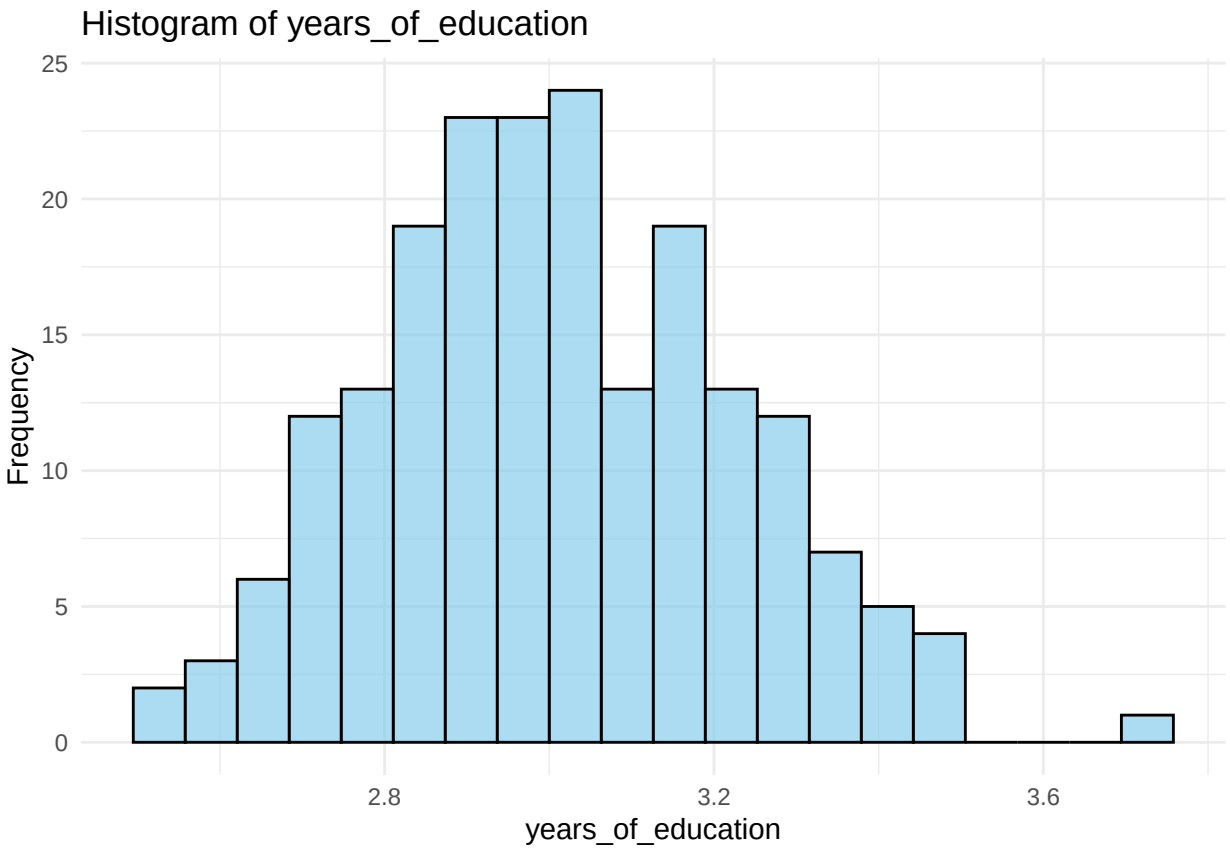

```
## Warning: Removed 1 row containing non-finite outside the scale range (‘stat_bin()‘).
## Removed 101 rows containing missing values or values outside the scale range
## (‘geom_function()‘).
```




```
## Warning: Removed 1 row containing non-finite outside the scale range (‘stat_bin()‘).
## Removed 101 rows containing missing values or values outside the scale range
## (‘geom_function()‘).
```




```
## Warning: Removed 1 row containing non-finite outside the scale range (‘stat_bin()‘).
## Removed 101 rows containing missing values or values outside the scale range
## (‘geom_function()‘).
```



Zombies killed and years of education exhibit a positive skew in their distributions, whereas height, weight, and age follow a normal distribution pattern. A potential approach to explore the data's distribution involves applying log transformation to the skewed variables and observing any alterations in the data patterns.

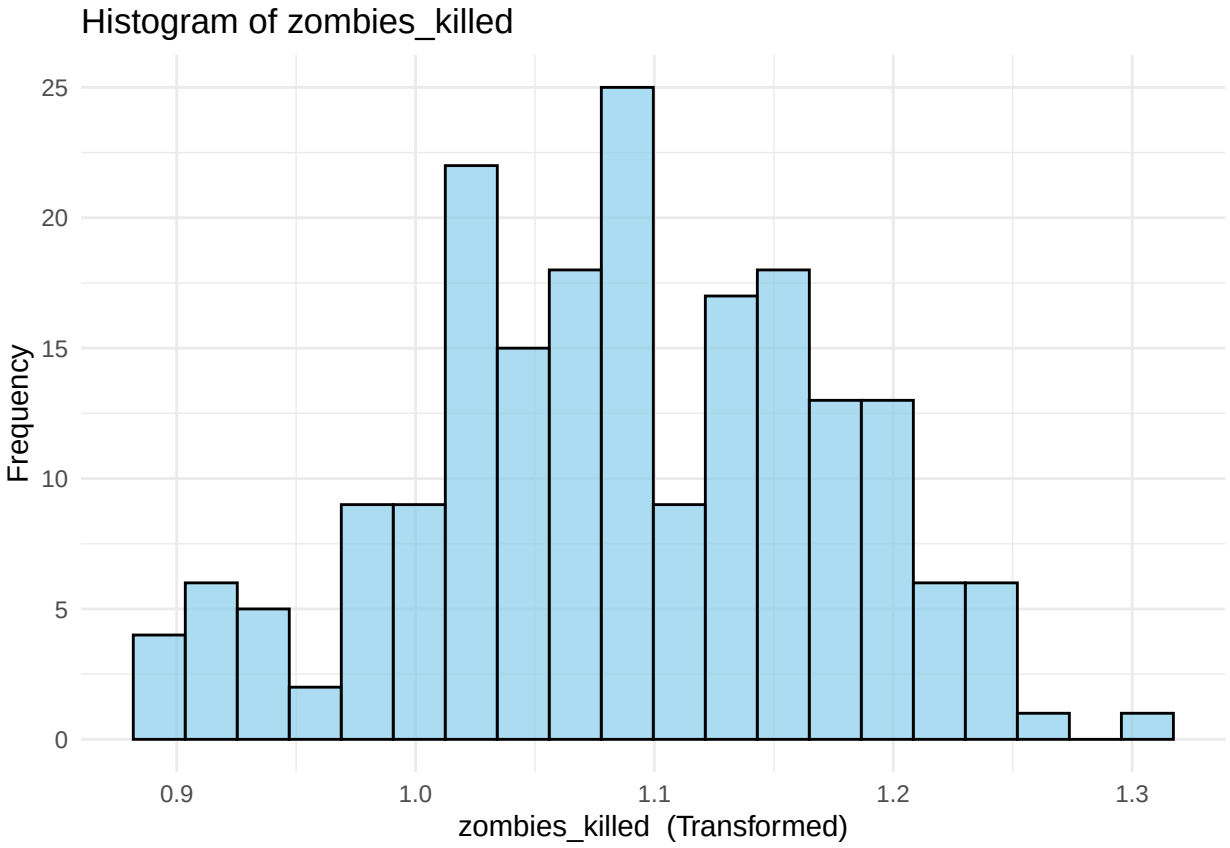
```
# Combine means from the first sample with the additional samples
all_means_df <- bind_rows(summary_stats_sample, additional_means)

# Function to plot histogram and normal distribution curve with transformation
plot_histogram_transformed <- function(data, variable, transformation = NULL) {
  ggplot(data, aes(x = transformation(!!sym(variable)))) +
    geom_histogram(fill = "skyblue", color = "black", bins = 20, alpha = 0.7) +
    stat_function(fun = dnorm, args = list(mean = mean(data[[variable]]), sd = sd(data[[variable]])), c
    labs(title = paste("Histogram of", variable),
         x = paste(variable, " (Transformed)"),
         y = "Frequency") +
    theme_minimal()
}

# Plot histograms and normal distribution curves for each variable, applying log transformation
for (variable in c("zombies_killed", "years_of_education")) {
  print(plot_histogram_transformed(all_means_df, variable, log))
}
```

```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
## This warning is displayed once every 8 hours.
```

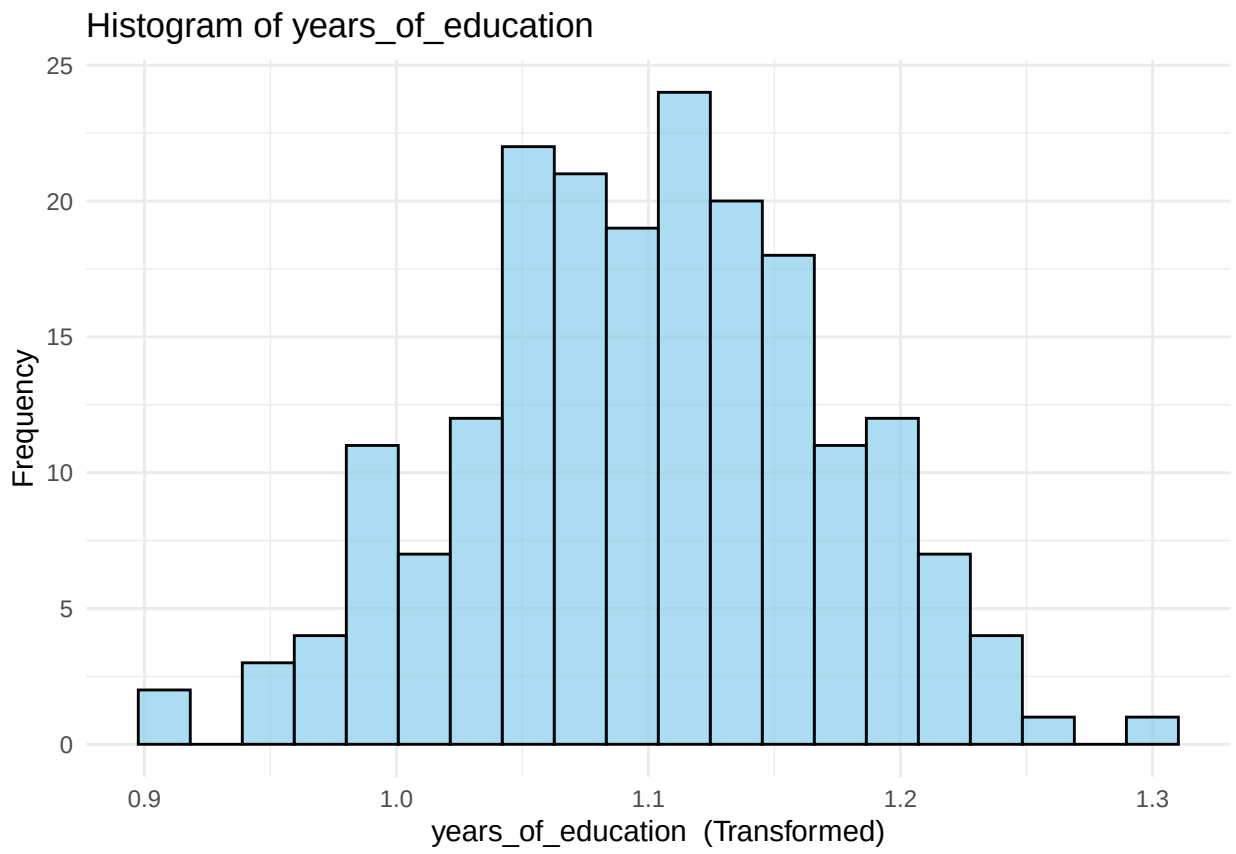
```
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
## Warning: Removed 1 row containing non-finite outside the scale range
## (‘stat_bin()‘).
```

```
## Warning: Removed 101 rows containing missing values or values outside the scale range
## (‘geom_function()‘).
```




```
## Warning: Removed 1 row containing non-finite outside the scale range (‘stat_bin()‘).
## Removed 101 rows containing missing values or values outside the scale range
## (‘geom_function()‘).
```



The log transformation applied to the dataset has resulted in the normalization of the variables "Years of Education" and "Zombies Killed."

Step 9: Construct a 95% confidence interval for each mean **directly from the sampling distribution** of sample means using the central 95% that distribution (i.e., by setting the lower and upper CI bounds to 2.5% and 97.5% of the way through that distribution).

```
# Specify the significance level (alpha) for the confidence interval
alpha <- 0.05

# Specify the variables for which you want to calculate confidence intervals
variables_of_interest <- c("height", "weight", "age", "zombies_killed", "years_of_education")

# Calculate and display 95% confidence intervals for each variable
for (variable in variables_of_interest) {
  lower_bound <- quantile(all_means_df[[variable]], alpha / 2, na.rm = TRUE)
  upper_bound <- quantile(all_means_df[[variable]], 1 - alpha / 2, na.rm = TRUE)

  cat("95% Confidence Interval for", variable, ": [", lower_bound, ", ", upper_bound, "]\n")
}
```

```
## 95% Confidence Interval for height : [ 66.31536 ,  68.66555 ]
## 95% Confidence Interval for weight : [ 137.6618 ,  148.5229 ]
## 95% Confidence Interval for age : [ 19.20699 ,  20.85638 ]
## 95% Confidence Interval for zombies_killed : [ 2.499 ,  3.461 ]
## 95% Confidence Interval for years_of_education : [ 2.637 ,  3.422 ]
```

Step 10: Finally, use bootstrapping to generate a 95% confidence interval for each variable mean **by resampling 1000 samples, with replacement, from your original sample** (i.e., by setting the lower and upper CI bounds to 2.5% and 97.5% of the way through the sampling distribution generated by bootstrapping).

```
# Function to perform bootstrapping and calculate mean
bootstrap_mean <- function(x) {
  bootstrap_sample <- sample(x, replace = TRUE)
  return(mean(bootstrap_sample))
}

# Number of bootstrap samples
num_bootstrap_samples <- 1000

# Create an empty data frame to store bootstrap samples
bootstrap_samples_df <- data.frame(matrix(nrow = num_bootstrap_samples, ncol = length(variables_of_inte

# Perform bootstrapping for each variable
for (i in seq_along(variables_of_interest)) {
  # Get the original sample for the current variable
  original_sample <- all_means_df[[variables_of_interest[i]]]

  # Perform bootstrapping
  bootstrap_means <- replicate(num_bootstrap_samples, bootstrap_mean(original_sample))

  # Store the bootstrap samples in the data frame
  bootstrap_samples_df[, i] <- bootstrap_means
}

# Rename the columns of the data frame
colnames(bootstrap_samples_df) <- variables_of_interest

# Calculate and display 95% confidence intervals from bootstrapping
for (variable in variables_of_interest) {
  lower_bound <- quantile(bootstrap_samples_df[[variable]], 0.025, na.rm = TRUE)
  upper_bound <- quantile(bootstrap_samples_df[[variable]], 0.975, na.rm = TRUE)

  cat("Bootstrapped 95% Confidence Interval for", variable, ": [", lower_bound, ", ", upper_bound, "]\n
}
```

```
## Bootstrapped 95% Confidence Interval for height : [ 67.50607 ,  67.68191 ]
## Bootstrapped 95% Confidence Interval for weight : [ 143.3493 ,  144.0893 ]
## Bootstrapped 95% Confidence Interval for age : [ 20.00848 ,  20.11809 ]
## Bootstrapped 95% Confidence Interval for zombies_killed : [ 2.9459 ,  3.007635 ]
## Bootstrapped 95% Confidence Interval for years_of_education : [ 2.975448 ,  3.0399 ]
```