Elementary Statistics with R: STAT 2670

Table of contents

Preface

This is an R-manual that accompanies the textbook Triola (2022) for the courses STAT 2670: Elementary Statistics offered at Auburn University at Montgomery.

Credits:

- Jerome Goddard (Chapters 2 and 3)
- Yi Wang (chapters 4 and 5; Overall editing)
- Wen Tang (chapters 6 and 7)
- Jieun Park (chapters 7, 8 and 10)

1 Setting-up Computing Environment

1.1 Setting up your own computing environment on a personal computer

This is the recommended way and the advantage is that it's easy to handle files.

- Go to the website https://posit.co/download/rstudio-desktop/.
- Follow the two steps: 1) download and install R: Choose the appropriate operating system, and then choose "base" to "install R for the first time". You can simply accept all default options.

2) download Rstudio Desktop and Install it.

After installation, start R-Studio, and you are ready to use it.

1.2 Use R-Studio Cloud (No setting-up needed)

Alternatively, one can save the hassle of setting up on a personal computer and use the R-Studio Cloud for **free**. Here are the steps:

- Go to the website https://login.rstudio.cloud/.
- Either create a new account using an email address such as your AUM email or simply "Log in using Google" or click on other log-in alternative.

After log-in to your account, you are ready to use R Studio.

2 Probability

2.1 Basic concepts of probability

In this code:

- We calculate the probability of drawing a Heart from a sample space (a deck of cards).
- We simulate random events such as a coin toss and rolling a six-sided die.
- We simulate multiple die rolls and visualize the resulting probability distribution.
- We calculate the probability of a specific outcome (rolling a 3).

```r
# Set a seed for reproducibility
set.seed(42)

# Define a sample space (e.g., a deck of cards)
sample_space <- c("Hearts", "Diamonds", "Clubs", "Spades")

# Calculate the probability of drawing a Heart from the sample space
probability_heart <- sum(sample_space == "Hearts") / length(sample_space)

cat("Probability of drawing a Heart:", probability_heart, "\n")
```

```
Probability of drawing a Heart: 0.25
```

```r
# Simulate a random event (e.g., coin toss)
coin_toss <- sample(c("Heads", "Tails"), size = 1)

cat("Result of a random coin toss:", coin_toss, "\n")
```

```
Result of a random coin toss: Heads
```

```r
# Simulate rolling a six-sided die
die_roll <- sample(1:6, size = 1)

cat("Result of rolling a die:", die_roll, "\n")
```

```
Result of rolling a die: 5
```

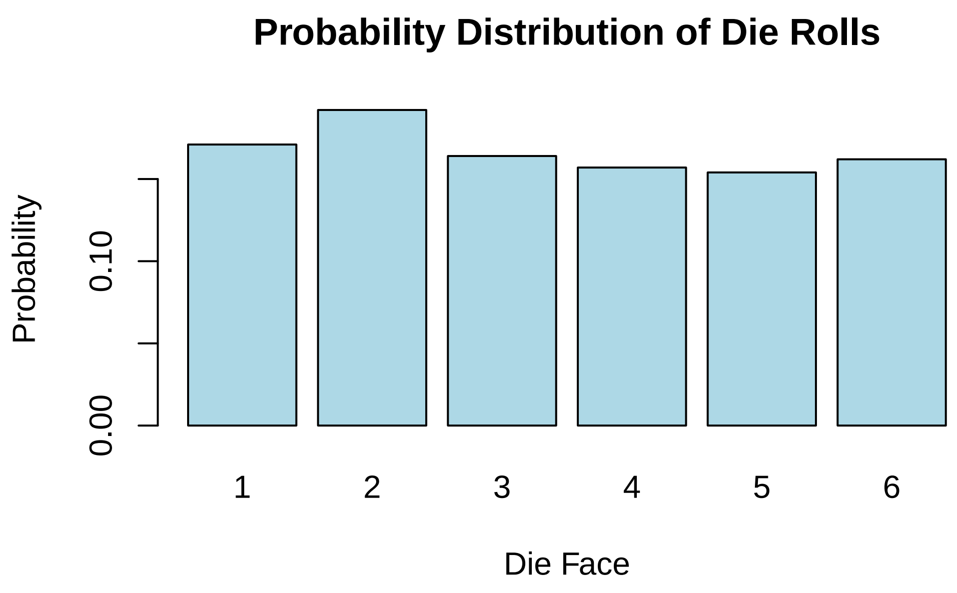
```r
# Simulate multiple die rolls and visualize the probability distribution
num_rolls <- 1000
die_rolls <- sample(1:6, size = num_rolls, replace = TRUE)

# Calculate the relative frequencies for each outcome
relative_frequencies <- table(die_rolls) / num_rolls
relative_frequencies
```

```
die_rolls
    1     2     3     4     5     6 
0.171 0.192 0.164 0.157 0.154 0.162 
```

```r
# Calculate the probability of rolling a 3
probability_roll_3 <- relative_frequencies[3]

cat("Probability of rolling a 3:", probability_roll_3, "\n")
```

```
Probability of rolling a 3: 0.164
```

```r
# Visualize the probability distribution with a bar plot
barplot(relative_frequencies, main = "Probability Distribution of Die Rolls",
        xlab = "Die Face", ylab = "Probability", col = "lightblue")
```



2.2 Addition rule and multiplication rule

2.3 Complements, conditional probability, and Bayes' theorem

2.4 Counting

2.4.1 Calculate factorial $n!$

R provides a built-in function to calculate factorial. You can use the `factorial()` function in R to compute the factorial of a number.

```r
n <- 5
factorial_result <- factorial(n)
cat("Factorial of", n, "is", factorial_result, "\n")
```

```
Factorial of 5 is 120
```

Replace the value of `n` with the number for which you want to calculate the factorial, and the `factorial()` function will return the result.

2.4.2 Find all permutations and the number of all permutations

To do this, we can use the `permutations` function from the **gtools** package. For any list of size `n`, this function computes all the different permutations $P(n, r)$ we can get when we select `r` items. Here are all the ways we can choose two numbers from a list consisting of `1,2,3`:

```r
library(gtools)
permutations(3, 2)
```

```
     [,1] [,2]
[1,]    1    2
[2,]    1    3
[3,]    2    1
[4,]    2    3
[5,]    3    1
[6,]    3    2
```

Notice that the order matters here: 3,1 is different than 1,3. Also, note that (1,1), (2,2), and (3,3) do not appear because once we pick a number, it can't appear again.

To get the actual number of permutations, one can use the R-function `nrow()` to find the total number of rows in the output of `permutations`:

```
library(gtools)
nrow(permutations(3,2))
```

```
[1] 6
```

Alternatively, we can add a vector `v` to indicate the objects that a permutation is performed on. If you want to see five random seven digit phone numbers out of all possible phone numbers (without repeats), you can type:

```
all_phone_numbers <- permutations(10, 7, v = 0:9) # Use digits 0, 1, ..., 9
n <- nrow(all_phone_numbers)
cat("total number of phone numbers n = ", n, "\n")
```

```
total number of phone numbers n =  604800
```

```
print("Randomly sample 5 phone numbers:")
```

```
[1] "Randomly sample 5 phone numbers:"
```

```
# Randomly sample 5 phone numbers
index <- sample(n, 5)
all_phone_numbers[index,]
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]    8    9    5    1    6    0    3
[2,]    4    0    2    3    5    7    8
[3,]    0    4    6    1    3    2    7
[4,]    5    8    2    6    4    0    3
[5,]    7    5    1    0    9    2    6
```

Instead of using the numbers 1 through 10, the default, it uses what we provided through `v`: the digits 0 through 9.

2.4.3 Find all combinations and the number of all combinations

How about if the order doesn't matter? For example, in Blackjack if you get an Ace and a face card in the first draw, it is called a *Natural 21* and you win automatically. If we wanted to compute the probability of this happening, we would enumerate the *combinations*, not the permutations, since the order does not matter.

```r
combinations(3,2)
```

```
     [,1] [,2]
[1,]    1    2
[2,]    1    3
[3,]    2    3
```

In the second line, the outcome does not include (2,1) because (1,2) already was enumerated. The same applies to (3,1) and (3,2).

To get the actual number of combinations, one can do

```r
nrow(combinations(3,2))
```

```
[1] 3
```

(**optional**) Of course, one can define a R-function to calculate a permutation number.

```r
# Function to calculate permutation (nPr)
nPr <- function(n, r) {
  if (n < r) {
    return(0)
  } else {
    return(factorial(n) / factorial(n - r))
  }
}
nPr(3,2)
```

```
[1] 6
```

```r
# Function to calculate combination (nCr)
nCr <- function(n, r) {
  if (n < r) {
    return(0)
  } else {
    return(factorial(n) / (factorial(r) * factorial(n - r)))
  }
}
nCr(3,2)
```

```
[1] 3
```

3 Discrete probability distribution

3.1 Calculate mean, standard deviation and variance with equal probability

You can use R to calculate the mean, standard deviation, and variance of a given data set using built-in functions like `mean()`, `sd()`, and `var()`. Here's some sample R code to do that:

```r
# Sample data set
data_set <- c(12, 15, 18, 21, 24, 27, 30, 33, 36, 39)

# Calculate the mean
mean_value <- mean(data_set)
cat("Mean:", mean_value, "\n")
```

```
Mean: 25.5
```

```r
# Calculate the standard deviation
std_deviation <- sd(data_set)
cat("Standard Deviation:", std_deviation, "\n")
```

```
Standard Deviation: 9.082951
```

```r
# Calculate the variance
variance <- var(data_set)
cat("Variance:", variance, "\n")
```

```
Variance: 82.5
```

Just replace the data_set vector with your actual data, and this code will compute and print the mean, standard deviation, and variance for your data set. Note the results calculated by `mean()`, `sd()` and `var()` assumes each data points occurs with the equal probability $1/n$, where n is the number of data points.

3.2 Expectation and standard deviation with a given probability distribution

By definition,

```r
# Define the possible values and their corresponding probabilities
values <- c(1, 2, 3, 4, 5)
probabilities <- c(0.1, 0.2, 0.3, 0.2, 0.2)

# Calculate the mean (expected value)
mean_value <- sum(values * probabilities)

# Print the result
cat("Mean (Expected Value) =", mean_value, "\n")
```

```
Mean (Expected Value) = 3.2
```

Or one can use the following built-in function:

```r
wt <- c(5,  5,  4,  1)/15
x <- c(3.7,3.3,3.5,2.8)
xm <- weighted.mean(x, wt)
xm
```

```
[1] 3.453333
```

To calculate the variance of a probability distribution in R, you can use the Here's how you can do it:

```r
# Define the values of the random variable (x_i)
values <- c(1, 2, 3, 4, 5)

# Define the probabilities (P(x_i))
probabilities <- c(0.2, 0.3, 0.1, 0.2, 0.2)

# Calculate the mean (expected value) of the random variable
mean_x <- sum(values * probabilities)

# Calculate the variance using the formula
variance <- sum((values - mean_x)^2 * probabilities)

# Print the variance
cat("Variance:", variance, "\n")
```

```
Variance: 2.09
```

3.3 Median

```r
# Create a sample vector
data_vector <- c(12, 45, 23, 67, 8, 34, 19)

# Calculate the median
median_value <- median(data_vector)

# Print the median
cat("Median:", median_value, "\n")
```

```
Median: 23
```

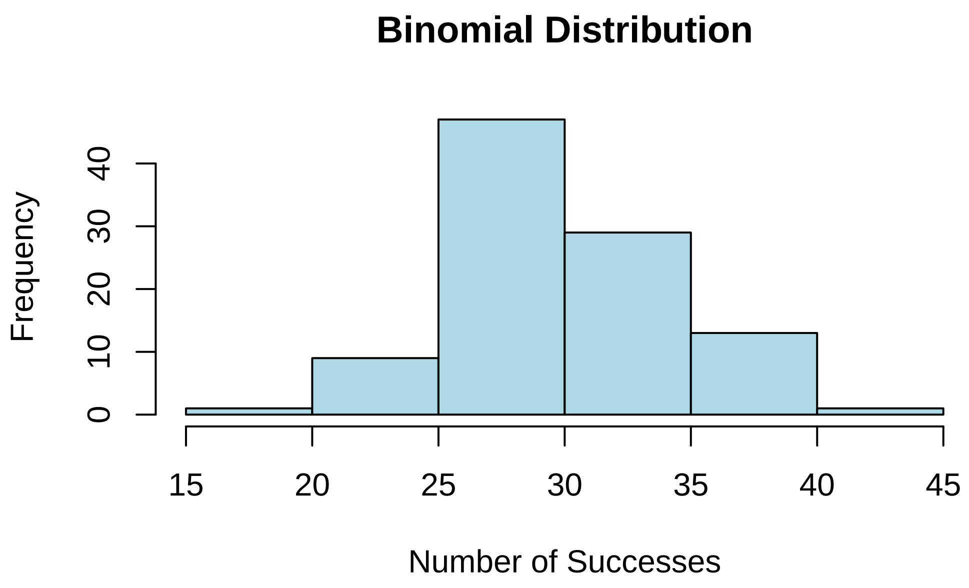
3.4 Binomial probability distributions

You can generate a data set with a binomial distribution in R using the `rbinom()` function. This function simulates random numbers following a binomial distribution. Here's an example code to generate a data set with a binomial distribution:

```r
# Set the parameters for the binomial distribution
n <- 100    # Number of trials
p <- 0.3    # Probability of success in each trial

# Generate a dataset with a binomial distribution
binomial_data <- rbinom(n, size = n, prob = p)

# Print the generated dataset
print(binomial_data)
```

```
  [1] 34 26 35 29 32 30 26 19 32 32 37 36 40 33 25 35 29 31 28 31 24 26 30 28 32
 [26] 26 24 32 30 34 27 34 25 28 31 27 26 21 30 29 32 30 30 28 36 37 36 37 31 28
 [51] 38 28 30 31 29 29 30 42 35 31 27 36 25 28 28 36 28 29 27 35 28 29 34 35 34
 [76] 28 25 27 30 30 37 27 35 29 26 28 32 31 25 28 24 30 36 33 28 36 29 32 29 33
```

```r
# Create a histogram to visualize the data
hist(binomial_data, main = "Binomial Distribution", xlab = "Number of Successes", ylab = "Fr
```



```r
# verify the mean =np, and var=npq
# Sample mean
mean(binomial_data)
```

```
[1] 30.39
```

```r
# Theoretical mean
n*p
```

```
[1] 30
```

```r
# Sample variance
var(binomial_data)
```

```
[1] 17.04838
```

```r
# Theoretical variance
n*p*(1-p)
```

```
[1] 21
```

You can calculate the probability of specific outcomes in a binomial distribution in R using the `dbinom()` function, which calculates the *probability mass function* (PMF) of the binomial distribution. Here's how to use it:

```r
# Set the parameters for the binomial distribution
x <- 2      # Number of successes (the outcome you want to calculate the probability for)
n <- 10     # Number of trials
p <- 0.3    # Probability of success in each trial

# Calculate the probability of getting 'x' successes in 'n' trials
probability <- dbinom(x, size = n, prob = p)

# Print the calculated probability
cat("Probability of", x, "successes in", n, "trials:", probability, "\n")
```

```
Probability of 2 successes in 10 trials: 0.2334744
```

The `pbinom()` function in R is used to calculate cumulative probabilities for a binomial distribution. Specifically, it calculates the cumulative probability that a random variable following a binomial distribution is less than or equal to a specified value. In other words, it gives you the *cumulative distribution function* (CDF) for a binomial distribution.

Here's the basic syntax of the `pbinom()` function:

```r
pbinom(q, size, prob, lower.tail = TRUE)
```

`q`: The value for which you want to calculate the cumulative probability.

`size`: The number of trials or events in the binomial distribution.

`prob`: The probability of success in each trial.

`lower.tail`: A logical parameter that determines whether you want the cumulative probability for values less than or equal to `q` (`TRUE`) or greater than `q` (`FALSE`). By default, it is set to `TRUE`.

The `pbinom()` function returns the cumulative probability for the specified value `q` based on the given parameters.

Here's an example of how to use `pbinom()`:

```r
# Calculate the cumulative probability that X is less than or equal to 3
cumulative_prob <- pbinom(3, size = 10, prob = 0.3)

# Print the cumulative probability
cat("Cumulative Probability:", cumulative_prob, "\n")
```

```
Cumulative Probability: 0.6496107
```

In this example, we're calculating the cumulative probability that a random variable following a binomial distribution with parameters `size = 10` and `prob = 0.3` is less than or equal to `3`. The result is stored in the cumulative_prob variable and printed to the console.

You can use the `pbinom()` function to answer questions like "What is the probability of getting at most 3 successes in 10 trials with a success probability of 0.3?" by specifying the appropriate values for q, size, and prob.

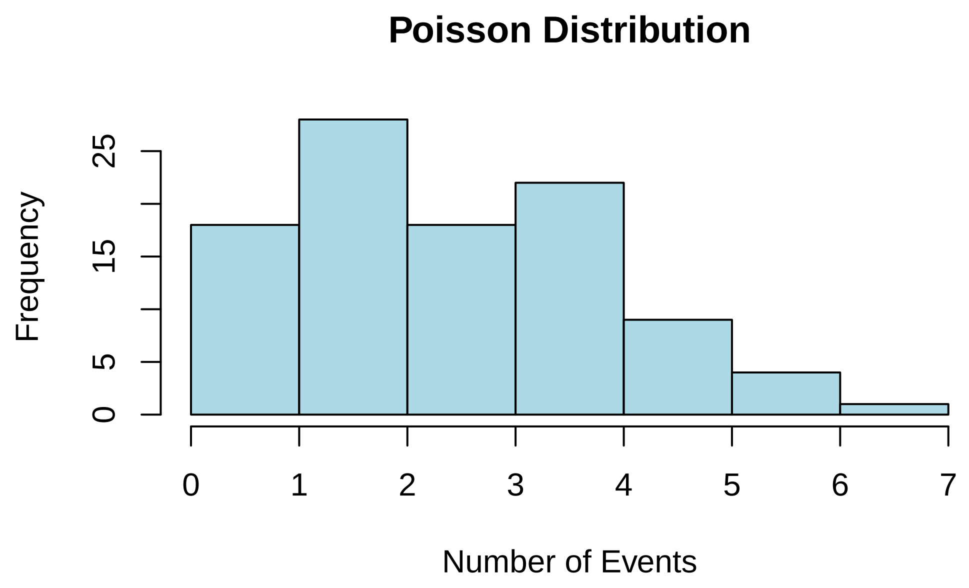
3.5 Poisson probability distributions (Optional)

To generate a data set with a Poisson distribution in R, you can use the `rpois()` function. The Poisson distribution is often used to model the number of events occurring in a fixed interval of time or space when the events happen with a known constant mean rate. Here's how you can use `rpois()`:

```r
# Set the parameters for the Poisson distribution
lambda <- 3  # Mean (average) rate of events

# Generate a dataset with a Poisson distribution
poisson_data <- rpois(n = 100, lambda = lambda)

# Print the generated dataset
print(poisson_data)
```

```
  [1] 1 5 2 4 2 3 4 2 1 6 4 3 3 0 4 4 4 2 2 2 5 1 1 3 2 6 2 4 2 2 2 2 1 7 4 0 1
 [38] 1 5 0 3 2 4 2 3 2 3 2 5 4 5 1 2 0 3 4 3 4 3 3 4 3 3 4 4 2 2 0 1 4 4 4 2 6
 [75] 2 1 5 6 2 4 2 5 1 3 2 2 4 2 3 2 3 4 1 2 1 5 5 4 3 3
```

```r
# Create a histogram to visualize the data
hist(poisson_data, main = "Poisson Distribution", xlab = "Number of Events", ylab = "Frequen
```



```r
# Verify the theoretical mean and variance
mean(poisson_data)
```

```
[1] 2.87
```

```r
#Theoretical mean = lambda

var(poisson_data)
```

```
[1] 2.356667
```

```r
#Theoretical variance = lambda
```

To calculate the probability of a specific value occurring in a Poisson distribution in R, you can use the `dpois()` function. This function calculates the *probability mass function* (PMF) of the Poisson distribution. Here's how to use it:

```r
# Set the parameters for the Poisson distribution
x <- 2     # The specific value for which you want to calculate the probability
lambda <- 3  # Mean (average) rate of events

# Calculate the probability of getting exactly 'x' events
probability <- dpois(x, lambda)
```

```r
# Print the calculated probability
cat("Probability of", x, "events:", probability, "\n")
```

```
Probability of 2 events: 0.2240418
```

To calculate the *cumulative distribution function* (CDF) for a Poisson distribution in R, you can use the `ppois()` function. This function calculates the cumulative probability that a Poisson random variable is less than or equal to a specified value. Here's how to use it:

```r
# Set the parameters for the Poisson distribution
x <- 2     # The specific value for which you want to calculate the cumulative probability
lambda <- 3  # Mean (average) rate of events

# Calculate the cumulative probability of getting less than or equal to 'x' events
cumulative_prob <- ppois(x, lambda)

# Print the calculated cumulative probability
cat("Cumulative Probability of less than or equal to", x, "events:", cumulative_prob, "\n")
```

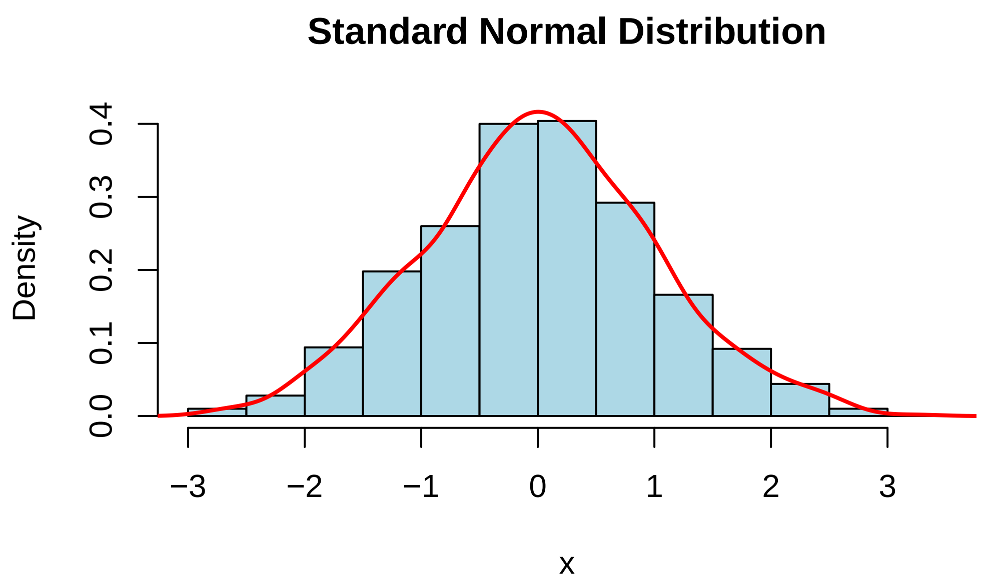
```
Cumulative Probability of less than or equal to 2 events: 0.4231901
```

4 NORMAL PROBABILITY DISTRIBUTION

4.0.1 THE standard normal distribution

4.0.1.1 Normal distribution graph (Displaying only).

```r
set.seed(123)                          # Set the seed for reproducibility
x <- rnorm(1000, mean = 0, sd = 1)  # Generate data for a standard normal distribution

# Plot the data with density curve
hist(x, prob = TRUE, col = "lightblue", main = "Standard Normal Distribution")
lines(density(x), col = "red", lwd = 2)
```



4.0.1.2 Find the probability (area) when z scores are given.

```r
# Find the area under the curve to the left of a certain value: P(z<1)
pnorm(1, mean = 0, sd = 1)
```

```
[1] 0.8413447
```

```r
# Find the area under the curve to the right of a certain value: P(z>1)
1-pnorm(1, mean = 0, sd = 1)
```

```
[1] 0.1586553
```

```r
# Find the area under the curve between two values: P(-1<z<1)
diff(pnorm(c(-1, 1), mean = 0, sd = 1))
```

```
[1] 0.6826895
```

4.0.1.3 Find z scores when the area is given.

```r
# Find the value with a certain area under the curve to its left: critical value
alpha <- 0.05
qnorm(1-alpha, mean = 0, sd = 1) # find the critical Z score.
```

```
[1] 1.644854
```

4.0.2 REAL application of normal distribution

4.0.2.1 Convert an individual x value to a z-score

```r
x <- 80  # the individual value
mu <- 75  # the mean of the distribution
sigma <- 10  # the standard deviation of the distribution

# Calculate z-scores for the individual value using scale()
z_scores <- scale(x, center = mu, scale = sigma)
cat("Z-score:", z_scores, "\n") # print the z-score
```

```
Z-score: 0.5
```

```r
z <- (x - mu) / sigma  # find the z-score by using the formula
cat("Z =", z, "\n") # print the z-score
```

```
Z = 0.5
```

4.0.2.2 Find the probability when x value is given (page 269 Pulse Rates Question)

```r
x1 <- 60
x2 <- 80
mu <- 69.6
sigma <- 11.3
# Find the probability that X is less than 60: P(X<60)
pnorm(x1, mean = mu, sd = sigma)
```

```
[1] 0.1977856
```

```r
# Find the probability that X is great than 80: P(X>80)
1-pnorm(x2, mean = mu, sd = sigma)
```

```
[1] 0.1786939
```

```r
# Find the probability between two values: P(60<X<80)
diff(pnorm(c(x1, x2), mean = mu, sd = sigma))
```

```
[1] 0.6235205
```

4.0.2.3 Convert a z-score back to x value

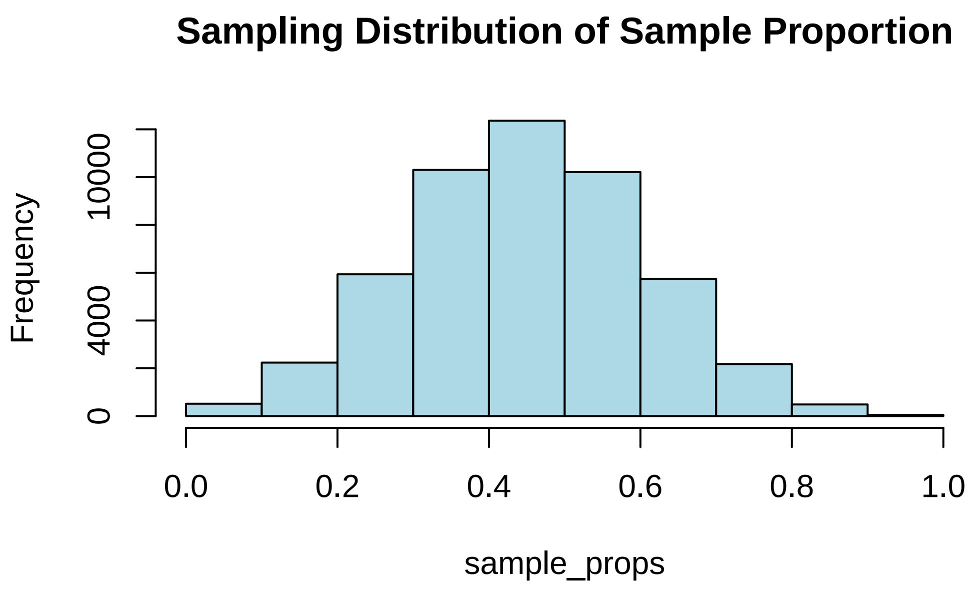
```r
z <- 1.96  # the z-score
mu <- 100  # the mean of the distribution
sigma <- 15  # the standard deviation of the distribution
x <- z * sigma + mu  # convert the z score to individual x value using formula
cat("X =", x, "\n")  # print the individual x value
```

```
X = 129.4
```

4.0.3 SAMPLING distributions and estimators (Displaying only/Optional)

4.0.3.1 general behavior of sampling distribution of the sample proportion

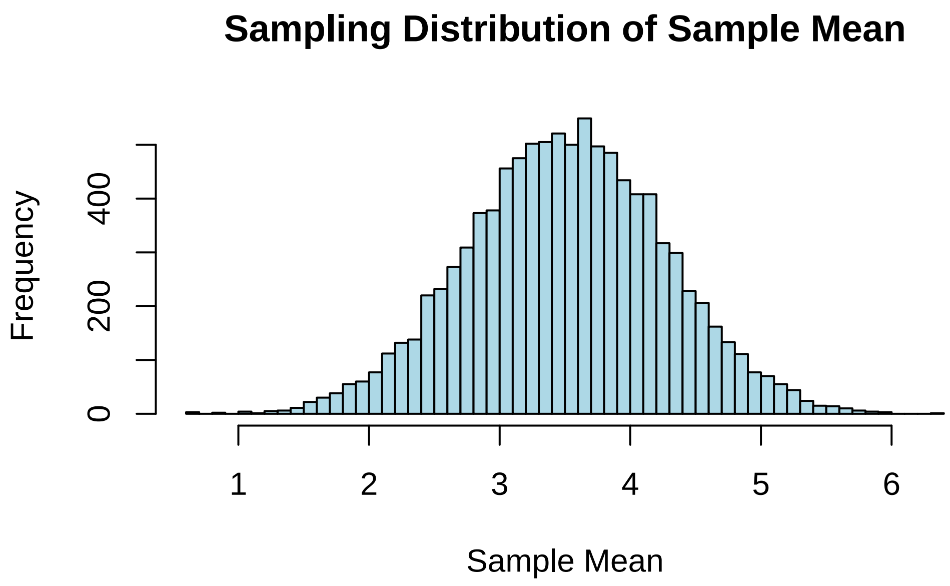
```
# Set the seed for reproducibility
set.seed (123)
# Generate data
n <- 10  # sample size
p <- 0.5  # population proportion
samples <- replicate(50000, rbinom(1, size = n, prob = p))

# Calculate sample proportions
sample_props <- samples / n

# Plot the histogram

hist(sample_props, breaks = seq( 0, 1, by = 0.1 ), col = "lightblue", main = "Sampling Distr
```



4.0.3.2 general behavior of sampling distribution of the sample mean

```
#input the parameter values
mu <- 3.5
```

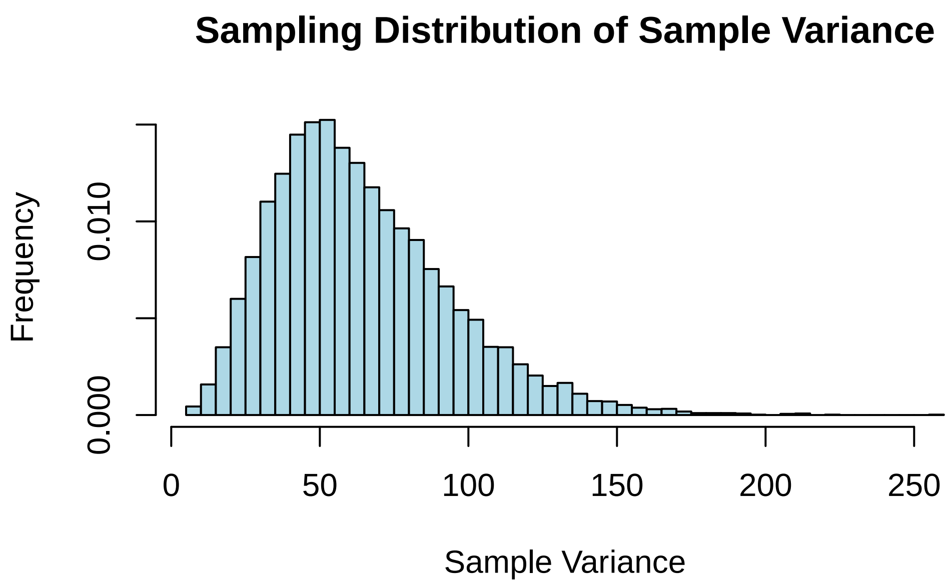
```r
sigma <- 1.7
n <- 5
# Simulate sampling distribution
sample_means <- replicate(10000, mean(rnorm(n, mu, sigma)))

# Create a histogram of the sampling distribution of the sample mean
hist(sample_means, breaks ="FD",  main = "Sampling Distribution of Sample Mean", xlab = "Sam
```



4.0.3.3 general behavior of sampling distribution of the sample variance

```r
mu <- 4    # True population mean
sigma <- 8       # Population standard deviation
sample_size <- 10         # Sample size
num_samples <- 10000       # Number of samples
# Function to calculate sample variance
sample_variance <- function(sample) {
  n <- length(sample)
  mean_sample <- mean(sample)
  sum_squared_deviations <- sum((sample - mean_sample)^2)
  return(sum_squared_deviations / (n - 1))
}
# Simulate sampling distribution
sample_variances <- replicate(num_samples, sample_variance(rnorm(sample_size, mu, sigma)))
```

```r
# Create a histogram of the sampling distribution of sample variance
hist(sample_variances, breaks = "FD", freq = FALSE, main = "Sampling Distribution of Sample
     xlab = "Sample Variance", ylab = "Frequency", col = "lightblue", border = "black")
```



4.0.4 THE central limit theorem

4.0.4.1 Find the probability when individual value is used (Page 292 Ejection Seat Question)

```r
mu <- 171 # population mean
sigma <- 46 # population standard deviation
n <- 25 # sample size
x_lower <- 140
x_upper <- 211

# Find the probability between two X values
probability_range <- diff(pnorm(c(x_lower, x_upper), mean = mu, sd = sigma))
probability_range
```

```
[1] 0.5575477
```

4.0.4.2 Find the probability when sample mean is used (Page 292 Ejection Seat Question)

```r
# Find the probability between two mean values $x/bar$ (CLT)
standard_error <- sigma / sqrt(n) # Calculate the standard error of the sample mean
probability_range <- diff(pnorm(c(x_lower, x_upper), mean = mu, sd = standard_error))# Find
probability_range
```

```
[1] 0.9996167
```

4.1 ESTIMATING PARAMETERS AND DETERMINGING SAMPLE SIZES

4.1.1 ESTIMATING a population proportion (Page 313 Online Course Example)

4.1.1.1 Getting the CI directly

```r
p_hat <- 0.53 # 0.53 for 53% sample proportion
n <- 950 # sample size
success <- n*p_hat # number of success

# Calculate a 95% confidence interval for the population proportion
result <- prop.test(success, n, conf.level = 0.95)

# Extract the confidence interval
conf_interval <- result$conf.int
# Print the confidence interval
cat("Confidence Interval:", conf_interval[1], "to", conf_interval[2], "\n")
```

```
Confidence Interval: 0.4976792 to 0.5620751
```

4.1.1.2 Getting the CI step by step

1.Critical value

```
# Confidence level (e.g., 0.95 for 95% confidence)
confidence_level <- 0.95
# get alpha value
alpha <- 1-confidence_level

# Find the critical Z-value using qnorm()
critical_z <- qnorm (1 - alpha/2)
# Print the result
cat("Critical Z =", critical_z, "\n")
```

```
Critical Z = 1.959964
```

2. Margin of error

```
# Calculate the standard error
standard_error <- sqrt((p_hat * (1 - p_hat)) / n)
# Calculate the margin of error
margin_of_error <- critical_z * standard_error
# Print the result
cat("E=", margin_of_error, "\n")
```

```
E= 0.03173753
```

3. Confidence interval

```
# Calculate the confidence interval
confidence_interval <- c (p_hat - margin_of_error,
                          p_hat + margin_of_error)

# Print the confidence interval
cat("Confidence Interval:", confidence_interval[1], "to", confidence_interval[2], "\n")
```

```
Confidence Interval: 0.4982625 to 0.5617375
```

4.1.2 ESTIMATING a population mean

4.1.2.1 Get the CI directly with Original data values are given. (Page 343 Mercury question)

```
# Calculate a 98% confidence interval for the population mean
#Sample data
mercury <- c(0.56, 0.75, 0.10, 0.95, 1.25, 0.54, 0.88)
result <- t.test(mercury,conf.level = 0.98)

# Extract the confidence interval
conf_interval <- result$conf.int

# Print the confidence interval
cat("Confidence Interval:", conf_interval[1], "to", conf_interval[2], "\n")
```

```
Confidence Interval: 0.2841145 to 1.153028
```

4.1.2.2 Get the CI step by step with given mean and standard deviation (Page 341 Hershey kisses question)

1. Critical value

```
confidence_level <- 0.99  # Confidence level (e.g., 0.99 for 99% confidence)
alpha <- 1- confidence_level
n <- 32          # Sample size

# Calculate the degrees of freedom
degrees_of_freedom <- n - 1

# Find the critical t-value using qt()
critical_t <- qt(1 - alpha/ 2, df = degrees_of_freedom)

# Print the result
cat("Critical t-value for degrees of freedom =", degrees_of_freedom, "and confidence level =
```

```
Critical t-value for degrees of freedom = 31 and confidence level = 0.99 : 2.744042
```

2. Margin of error

```
# Given sample standard deviation (this is s value)
sample_standard_deviation <- 0.1077

# Calculate the standard error
standard_error <- sample_standard_deviation / sqrt(n)

# Calculate the margin of error
margin_of_error <- critical_t * standard_error
```

```r
# Print the result
cat("Margin of Error for confidence level =", confidence_level, "and sample size =", n, ":",
```

```
Margin of Error for confidence level = 0.99 and sample size = 32 : 0.0522434
```

3. Confidence interval

```r
x_bar<- 4.5210       # Sample mean

# Calculate the lower and upper bounds of the confidence interval
lower_bound <- x_bar - margin_of_error
upper_bound <- x_bar + margin_of_error

# Print the result
cat("Confidence Interval:", lower_bound, "to", upper_bound, "\n")
```

```
Confidence Interval: 4.468757 to 4.573243
```

4.1.3 ESTIMATING a population variance (body temperature example page 353)

4.1.3.1 Critical values

```r
confidence_level <- 0.95  # Confidence level ( 0.95 for 95% confidence)
alpha <- 1- confidence_level
sample_size <- 106          # Sample size
degrees_of_freedom <- sample_size - 1  # Degrees of freedom for the chi-squared distribution

# Find the critical values using the chi-squared distribution
lower_critical_value <- qchisq(1-alpha/2, df = degrees_of_freedom)
upper_critical_value <- qchisq(alpha/2, df = degrees_of_freedom)

# Print the results
cat("Lower Critical Value:", lower_critical_value, "\n")
```

```
Lower Critical Value: 135.247
```

```r
cat("Upper Critical Value:", upper_critical_value, "\n")
```

```
Upper Critical Value: 78.5364
```

4.1.3.2 Confidence interval

```
sample_standard_deviation <- 0.62                    # sample standard deviation s
sample_variance <- sample_standard_deviation^2       # Sample variance

# Calculate the confidence interval for variance
confidence_interval <- c(((sample_size - 1) * sample_variance) / lower_critical_value,
                         ((sample_size - 1) * sample_variance) / upper_critical_value)

# Print the confidence interval
confidence_interval
```

```
[1] 0.2984318 0.5139273
```

4.2 SAMPLE QUESTIONS FOR CHAPTER 6 AND 7

4.2.1 SECTION 6.1

4.2.1.1 Bone density scores are normally distributed with a mean of 0 and a standard deviation of 1. Find the probability of the given bone density test scores. Please use r instead of table and round your answers to four decimal places.

1. Less than -2.00
2. Greater than 2.33
3. Between -0.77 and 1.42

4.2.1.2 Bone density scores are normally distributed with a mean of 0 and a standard deviation of 1.Find the bone desity test scores corresponding to the given information. Round your answer to two decimal places.

1. Find the 99th percentile P_{99}. This is the bone density score separating the bottom 99% from the top 1%.
2. Find the bone density scores that are the three Quartiles: Q_1, Q_2,Q_3.

4.2.1.3 Find the indicated critical value. Round results to two decimal places.

1. $Z_{0.25}$
2. $Z_{0.02}$
3. $Z_{0.06}$

4.2.2 SECTION 6.2

4.2.2.1 The IQ test scores of adults are normally distributed with a mean of 100 and a standard deviation of 15 (As on the Wechsler IQ test).

1. Find the probability that a person has IQ score greater than 125.
2. Find the probability that a person has IQ score between 90 and 105.
3. Find the P_{90}, which is the IQ score to separating the bottom 90% from the top 10%.

4.2.3 SECTION 6.4

4.2.3.1 Assume that weights of men are normally distributed with a mean of 189 lb and a standard deviation of 39 lb.

1. If one man is randomly selected, What is the probability that his weight exceeds 140 lb.
2. If 30 men are randomly selected, what is the probability that their mean weight exceeds 140 lb.

4.2.4 SECTION 7.1

4.2.4.1 One of Mendel's famous genetics experiments yielded 580 peas, with 428 of them green and 152 yellow.

Find a 99% confidence interval estimate of the percentage of green peas. a) Find the critical value. b) Find the margin of error. c) Find the confidence interval

4.2.5 SECTION 7.2

4.2.5.1 The summary statistics for the weights of Pepsi in randomly selected cans are n=36, $\bar{x}$=0.82410 lb, s=0.00570 lb. Use a confidence level of 95%

a) Find the critical value.
b) Find the margin of error.
c) Find the confidence interval

4.2.6 SECTION 7.3

4.2.6.1 Assume the weights of dollar coins are normally distributed. Find the 95% confidence interval given n=20, s= 0.04111.

a) Find the critical values
b) Find the confidence interval.

5 Hypothesis Testing

5.0.1 Basic of Hypothesis Testing

We will use the following functions to perform hypothesis tests.

```r
library(BSDA)
```

```
Warning: package 'BSDA' was built under R version 4.2.3
```

```r
# prop.test(x, n, p = NULL,
#           alternative = c("two.sided", "less", "greater"),
#           conf.level = 0.95, correct = TRUE)

# t.test(x, y = NULL,
#       alternative = c("two.sided", "less", "greater"),
#       mu = 0, paired = FALSE, var.equal = FALSE,
#       conf.level = 0.95, ...)

# z.test(
#   x, y = NULL,
#   alternative = "two.sided",
#   mu = 0, sigma.x = NULL, sigma.y = NULL,
#   conf.level = 0.95)
```

We use `qnorm()` and `qt()` functions to calculate critical values. For example, we can obtain $z_{0.05}$ using the `qnorm(0.95)` for a normal distribution, and the critical value $t_{0.05,5}$ using `qt(0.95, 5)` for a t-distribution with 5 degree of freedom with $\alpha = 0.05$ as below.

```r
qnorm(0.95)
```

```
[1] 1.644854
```

```r
qt(0.95, 5)
```

```
[1] 2.015048
```

5.0.2 Testing a Claim About a Proportion

`mtcars` dataset has data for 32 automobiles in 1973-1974 with 11 variables. Among these variable, we are interested to check if the proportion of V-shaped engine (`vs = 0`) is 0.5. That is, $H_0 : p = 0.5$. We set the null hypothesis as follows: the population proportion of cars with a V-shaped engine (vs = 0) among all automobiles in 1973-1974 is equal to 0.5. We first check if we can use a normal approximation to perform a proportion test. With a sample size of $n = 32$ and a proportion of interest $p = 0.5$, both the expected number of successes and failures are $np = n(1-p) = 32 \cdot 0.5 = 16$. Since they are greater than 5, we can apply the proportion test using a normal approximation. In our sample, the number of success (`vs=0`) is 18 and the sample proportion is 0.56.

```
data(mtcars)
attach(mtcars)
table(vs)
```

```
vs
 0  1 
18 14 
```

```
prop.table(table(vs))
```

```
vs
     0      1 
0.5625 0.4375 
```

We use one sample proportion test with `prop.test()` function if $np \geq 5$ and $n(1-p) \geq 5$ where p is the null hypothesized proportion and n is the sample size. The syntax is below if we want to test with a sample vector (categorical variable with two levels) for $H_0 : p = p_0$ with $\alpha = 0.05$. `x` is the number of success, `n`is the sample size, and `p_0` is the null hypothesized proportion.

```
# prop.test(x, n, p = p_0, conf.level=0.95, alternative=c("two.sided", "less", "greater"))
```

Depending on the alternative hypothesis H_1, we can choose one among `two.sided`, `less`, and `greater`. Under $H_0 : p = p_0$, we can use each `alternative` option for `prop.test()` function.

1. $H_1 : p = p_0$: `alternative = "two.sided"`
2. $H_1 : p = p_0$:`alternative = "less"`

3. $H_1 : p = p_0$: `alternative = "less"`

For the proportion of `vs` , we test for the proportion of `vs = 0` with $H_0 : p = 0.5$ and $\alpha = 0.05$.

5.0.2.1 Two-sided Proportion Test

$$H_0 : p = 0.5 \quad \text{vs} \quad H_1 : p \neq 0.5$$

```
res <- prop.test(x=18, n=32, p = 0.50, alternative = "two.sided", conf.level = 0.95)
res
```

```

	1-sample proportions test with continuity correction

data:  18 out of 32, null probability 0.5
X-squared = 0.28125, df = 1, p-value = 0.5959
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.3788033 0.7316489
sample estimates:
     p 
0.5625 
```

Decision:

- **P-Value**: we fail to reject the null hypothesis since p-value 0.596 is greater than $\alpha = 0.05$.
- **Critical Value**: the z-test statistic $z = 0.53$ is closer to 0 than the critical values. Thus, we fail to reject the null hypothesis.

```
# the critical value can be calculated by the following code.
c(qnorm(0.025), qnorm(0.975))
```

```
[1] -1.959964  1.959964
```

- **Confidence Interval**: the claimed proportion 0.5 falls within the confidence interval of (0.379, 0.732). Thus we fail to reject the null hypothesis.

5.0.2.2 One-sided Proportion Test

$$H_0 : p = 0.5 \quad \text{vs} \quad H_1 : p > 0.5$$

```r
res <- prop.test(x=18, n=32, p = 0.50, alternative = "greater", conf.level = 0.95)
res
```

```
	1-sample proportions test with continuity correction

data:  18 out of 32, null probability 0.5
X-squared = 0.28125, df = 1, p-value = 0.2979
alternative hypothesis: true p is greater than 0.5
95 percent confidence interval:
 0.4041836 1.0000000
sample estimates:
     p 
0.5625 
```

Decision:

- **P-Value**: we fail to reject the null hypothesis since p-value 0.298 is greater than $\alpha = 0.05$.

- **Critical Value**: the test statistic $z = 0.53$ does not fall in the critical region which is greater than $z_{0.05} = 1.645$. Thus, we fail to reject the null hypothesis.

```r
# the critical value can be calculated by the following code.
qnorm(0.95)
```

```
[1] 1.644854
```

- **Confidence Interval**: the claimed proportion 0.5 falls within the confidence interval of (0.404, 1). Thus we fail to reject the null hypothesis.

5.0.3 Tesing a Claim About a Mean

5.0.3.1 Unknown σ with Normality Assumption

We use one sample t-test with `t.test()` function when we assume normality for population or the sample size is large enough. The syntax is below if we want to test with a sample vector (variable) `x` for $H_0 : \mu = m$ with $\alpha = 0.05$.

```
# t.test(x, mu= m, conf.level=0.95, alternative=c("two.sided", "less", "greater"))
```

Depending on the alternative hypothesis H_1, we can choose one among `two.sided`, `less`, and `greater`. Under $H_0 : \mu = m$, use each `alternative` option for `t.test()` function.

1. $H_1 : \mu \neq m$: `alternative = "two.sided"`
2. $H_1 : \mu < m$:`alternative = "less"`
3. $H_1 : \mu > m$: `alternative = "less"`

As an example, we test for `mpg` with $H_0 : \mu = 22$. That is, we test if the population mean of `mpg` is equal to 22. `mtcars` cars have 32 samples and we can understand that we have a large enough sample to use t-test with $\alpha = 0.05$.

5.0.3.1.1 Two-sided t-test

$$H_0 : \mu = 22 \quad \text{vs} \quad H_1 : \mu \neq 22$$

```
res <- t.test(mpg, mu=22, alternative = "two.sided", conf.level = 0.95)
res
```

```

	One Sample t-test

data:  mpg
t = -1.7921, df = 31, p-value = 0.08288
alternative hypothesis: true mean is not equal to 22
95 percent confidence interval:
 17.91768 22.26357
sample estimates:
mean of x 
 20.09062 
```

Decision:

- **P-Value**: we fail to reject the null hypothesis since p-value 0.083 is greater than $\alpha = 0.05$.

- **Critical Value**: the test statistic t = -1.792 is closer to 0 than the critical values. Thus, we fail to reject the null hypothesis.

```r
# the critical value can be calculated by the following code.
c(qt(0.025, df=31), qt(0.975, df=31))
```

```
[1] -2.039513  2.039513
```

- **Confidence Interval**: the claimed mean 22 falls within the confidence interval of (17.918, 22.264). Thus we fail to reject the null hypothesis.

5.0.3.1.2 One-sided t-test $H_1 : \mu < m$

$$H_0 : \mu = 22 \quad \text{vs} \quad H_1 : \mu < 22$$

```r
res <- t.test(mpg, mu=22, alternative = "less", conf.level = 0.95)
res
```

```

    One Sample t-test

data:  mpg
t = -1.7921, df = 31, p-value = 0.04144
alternative hypothesis: true mean is less than 22
95 percent confidence interval:
     -Inf 21.89707
sample estimates:
mean of x 
 20.09062 
```

Decision:

- **P-Value**: we reject the null hypothesis since p-value 0.041 is less than $\alpha = 0.05$.

- **Critical Value**: the test statistic t = -1.792 falls in the critical region which is less than $t_{0.05,31}$ = -1.696. Thus, we reject the null hypothesis.

```
# the critical value can be calculated by the following code.
qt(0.05, df=31)
```

```
[1] -1.695519
```

- **Confidence Interval**: the claimed mean does not fall within the confidence interval of $(-\infty, 21.897)$. Thus we reject the null hypothesis.

5.0.3.2 Known σ with Normality Assumption

We use one sample z-test or normal test with `z.test()` function when we assume normality for population with known population standard deviation σ. The syntax is below if we want to test with a sample vector (variable) `x` for $H_0 : \mu = m$ with $\alpha = 0.05$ and known `sigma`.

```
#library(BSDA)
# z.test(x, mu = m, sigma.x = sigma, conf.level = 0.95, alternative = c("two.sided", "less",
```

Depending on the alternative hypothesis H_1, we can choose one among `two.sided`, `less`, and `greater`. Under $H_0 : \mu = m$, use each `alternative` option for `t.test()` function.

1. $H_1 : \mu \neq m$: `alternative = "two.sided"`
2. $H_1 : \mu < m$:`alternative = "less"`
3. $H_1 : \mu > m$: `alternative = "less"`

For example, we test for `mpg` with $H_0 : \mu = 22$. Assume `mpg` follows a normal distribution with $\sigma = 6$, then we can use z-test with $\alpha = 0.05$.

5.0.3.2.1 Two-sided z-test

$$H_0 : \mu = 22 \quad \text{vs} \quad H_1 : \mu \neq 22$$

```
library(BSDA)
res <- z.test(mpg, mu=22, sigma.x = 6, alternative = "two.sided", conf.level = 0.95)
res
```

```

    One-sample z-Test

data:  mpg
z = -1.8002, p-value = 0.07183
alternative hypothesis: true mean is not equal to 22
95 percent confidence interval:
 18.01177 22.16948
sample estimates:
mean of x
 20.09062
```

Decision:

- **P-Value**: we fail to reject the null hypothesis since p-value 0.072 is greater than $\alpha = 0.05$.

- **Critical Value**: the test statistic $z =$ -1.8 is closer to 0 than the critical values. Thus, we fail to reject the null hypothesis.

```r
# the critical value can be calculated by the following code.
c(qnorm(0.025), qnorm(0.975))
```

```
[1] -1.959964  1.959964
```

- **Confidence Interval**: the claimed mean 22 falls within the confidence interval of (18.012, 22.169). Thus we fail to reject the null hypothesis.

5.0.3.2.2 One-sided z-test $H_1 : \mu < m$

$$H_0 : \mu_{mpg} = 22 \quad \text{vs} \quad H_1 : \mu_{mpg} < 22$$

```r
res <- z.test(mpg, mu=22, sigma.x = 6, alternative = "less", conf.level = 0.95)
res
```

```

    One-sample z-Test

data:  mpg
z = -1.8002, p-value = 0.03592
alternative hypothesis: true mean is less than 22
```

```
95 percent confidence interval:
       NA 21.83526
sample estimates:
mean of x
 20.09062
```

Decision:

- **P-Value**: we reject the null hypothesis since p-value 0.036 is less than $\alpha = 0.05$.
- **Critical Value**: the test statistic z = -1.8 falls in the critical region which is less than $z_{0.05}$ = -1.645. Thus, we reject the null hypothesis.

```r
# the critical value can be calculated by the following code.
qnorm(0.05)
```

```
[1] -1.644854
```

- **Confidence Interval**: the claimed mean does not fall within the confidence interval of $(-\infty, 21.835)$. Thus we reject the null hypothesis.

5.1 Correlation and Regression

5.1.1 Correlation

We check if a linear correlation exists between two variables using `cor()` function.

```r
# We can calculate the correlation coefficient between x and y with the following code.
# cor(x, y)
```

```r
library(tidyverse)
```

```
Warning: package 'tidyverse' was built under R version 4.2.3
```

```
Warning: package 'ggplot2' was built under R version 4.2.3
```

```
Warning: package 'tibble' was built under R version 4.2.3
```

```
Warning: package 'tidyr' was built under R version 4.2.3
```

```
Warning: package 'readr' was built under R version 4.2.3
```

```
Warning: package 'purrr' was built under R version 4.2.3
```

```
Warning: package 'dplyr' was built under R version 4.2.3
```

```
Warning: package 'stringr' was built under R version 4.2.3
```

```
Warning: package 'forcats' was built under R version 4.2.3
```

```
Warning: package 'lubridate' was built under R version 4.2.3
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.3     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.0
v ggplot2   3.4.3     v tibble    3.2.1
v lubridate 1.9.2     v tidyr     1.3.0
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 beco
```

```r
library(patchwork)
```

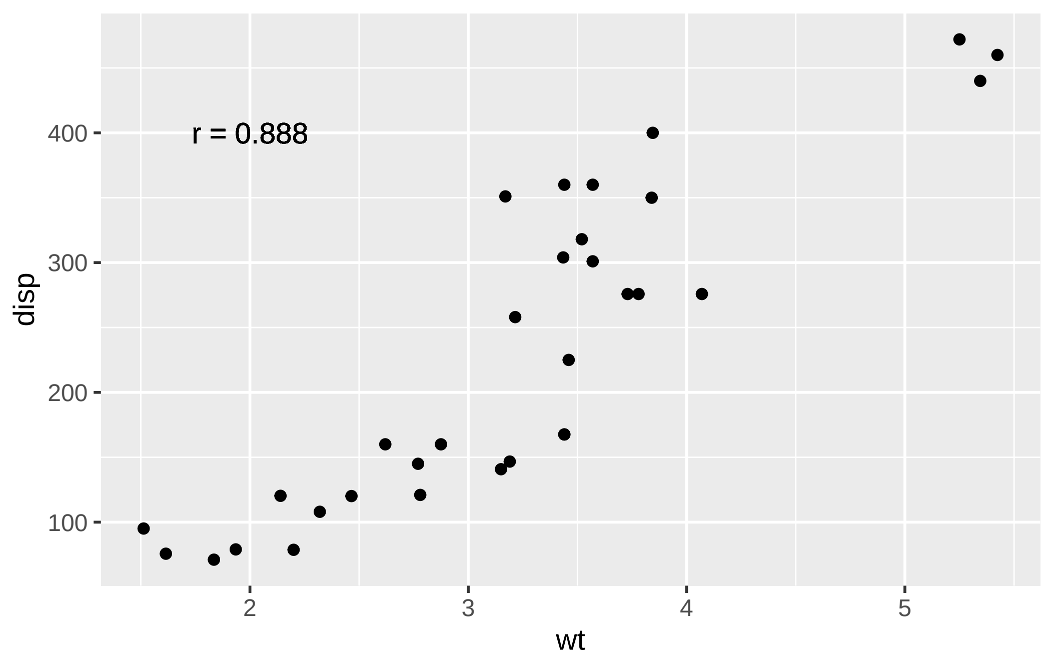
```
Warning: package 'patchwork' was built under R version 4.2.3
```

```r
data("mtcars")
names(mtcars)
```

```
 [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
[11] "carb"
```

```r
attach(mtcars)
# positive correlation
qplot(wt, disp, data = mtcars) +
  geom_text(aes(x=2, y=400, label="r = 0.888"))
```

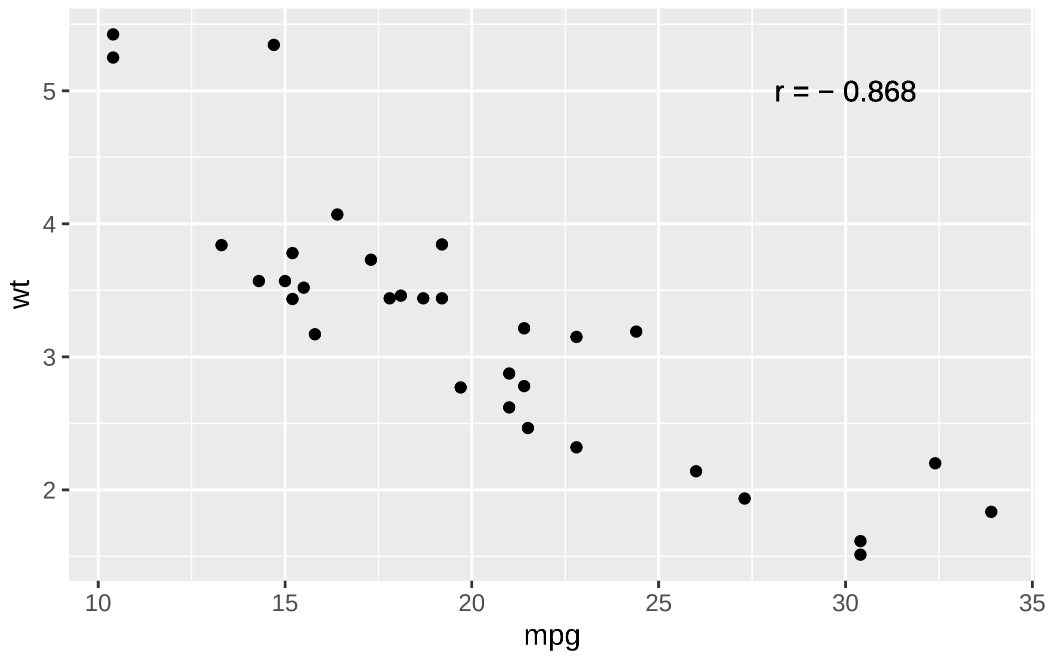
```
Warning: `qplot()` was deprecated in ggplot2 3.4.0.
```



```
cor(wt, disp)
```

```
[1] 0.8879799
```

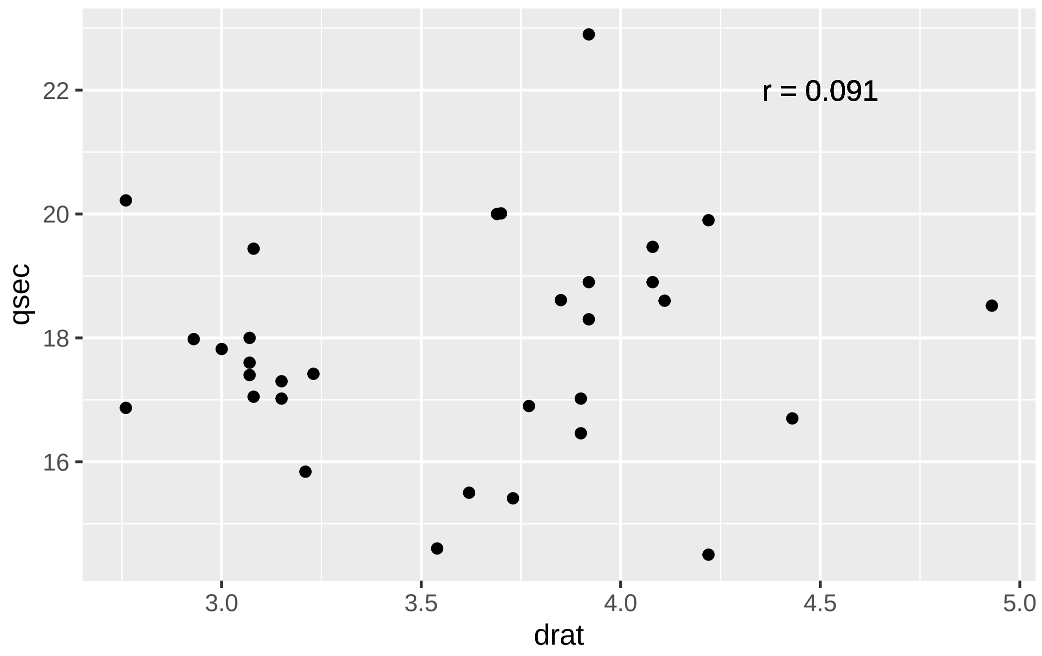
```
# negative correlation
qplot(mpg, wt, data = mtcars)  +
  geom_text(aes(x=30, y=5, label="r = - 0.868"))
```



```
cor(mpg, wt)
```

```
[1] -0.8676594
```

```
# no correlation
qplot(drat, qsec, data = mtcars)  +
  geom_text(aes(x=4.5, y=22, label="r = 0.091"))
```



```r
cor(drat, qsec)
```

```
[1] 0.09120476
```

- `wt` and `disp` have a positive correlation with r =0.888.
- `wt` and `disp` have a negative correlation with r = -0.868.
- `wt` and `disp` does not have a significant correlation with r = -0.175.

5.1.2 Regression

Assume we have a data set `data` with `x` and `y` variables and we check their linear relationship. We can find the slope and the intercept of the estimated regression line using the following code.

```r
# res <- lm(y ~ x, data)
# summary(res)
```

For example, we can find the regression line equation between `disp`(x, predictor) and `wt`(y, response) as below.

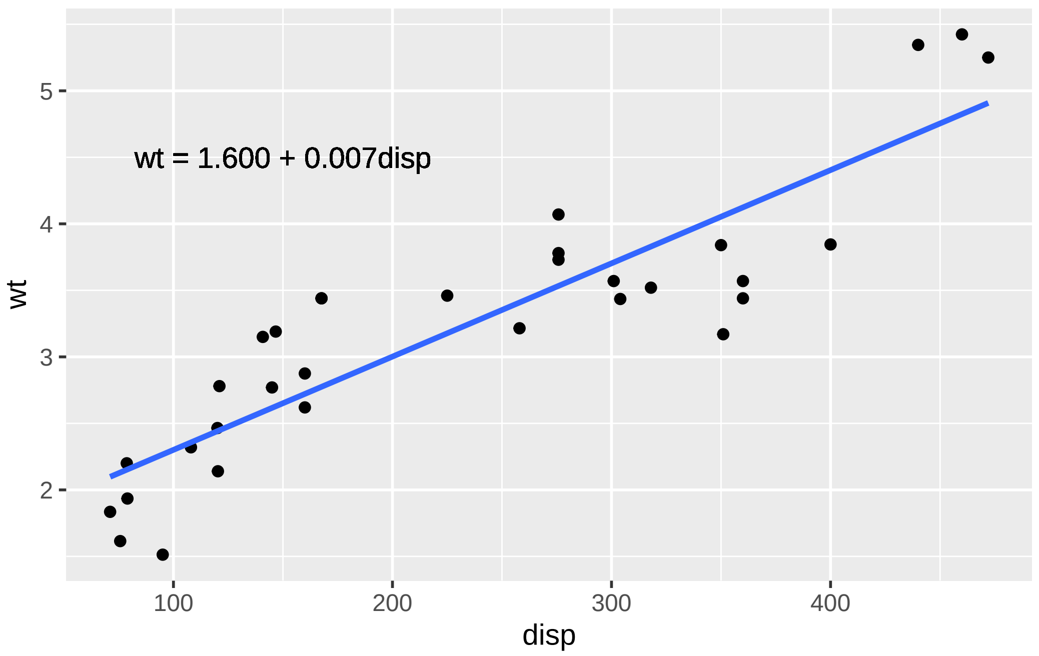
```r
library(tidyverse)
data("mtcars")

res <- lm(wt ~ disp, mtcars)
summary(res)
```

```
Call:
lm(formula = wt ~ disp, data = mtcars)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.89044 -0.29775 -0.00684  0.33428  0.66525 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 1.5998146  0.1729964   9.248 2.74e-10 ***
disp        0.0070103  0.0006629  10.576 1.22e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4574 on 30 degrees of freedom
Multiple R-squared:  0.7885,	Adjusted R-squared:  0.7815 
F-statistic: 111.8 on 1 and 30 DF,  p-value: 1.222e-11
```

The estimated regression line is $\text{wt} = 1.600 + 0.007\text{disp}$ since the intercept is 1.6 and the slope is 0.007. Both of them are significantly different from 0 with a significance level $\alpha = 0.05$. It means that one inch increase in `disp` (displacement) makes 7 lbs increase in `wt` (weight). On average, if a car has a one-inch longer displacement, it is 7 pounds heavier.

If a car has 200 inches displacement, then its estimated weight can be calculated as

$$1.600 + 0.007 \cdot 200 = 3000 \text{ lbs}$$


wt = 1.600 + 0.007disp
wt
5
4
3
2
100
200
300
400
disp

6 Summary

In summary, TBA.

References

Triola, Mario F. 2022. *Elementary Statistics.* USA: Pearson.