ST502 Project 1

Xinqian Li, Ming-Hung Chen, Kexuan Zhou

2024-02-22

The Goal of Project

This simulation study has been devised to estimate the probability of success, denoted as p, from a Binomial distribution. Our primary focus is to assess the performance of various confidence interval procedures.

Methods

We compared six confidence interval methods:

- Wald interval
- Adjusted Wald (Agresti-Coull) interval
- Clopper-Pearson (exact) interval
- Score (Wilson) interval
- Raw percentile interval using a parametric bootstrap
- Bootstrap t interval using a parametric bootstrap

Data Creation

The data for this simulation study were created to reflect a range of scenarios where the true probability of success, p, varies, as does the sample size, n. Specifically, we generated $N = 1,500$ random samples from a binomial distribution for each combination of n and p. The sample sizes we considered were 15, 100, and 200, and we explored 30 different values of p ranging from 0.01 to 0.99, providing a comprehensive view across diverse conditions. The R function `rbinom` was used to generate the samples.

Generate Random Samples

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

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

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
# Parameters for simulation
n_values <- c(15, 100, 200)
p_values <- seq(0.01, 0.99, length.out = 30)
num_samples <- 1500
```

Calculation of Quantities

For each set of generated data, we calculated the 95% confidence intervals using six different methods. Here is how we computed each type of interval:

1. Wald Interval

The Wald interval is based on the normal approximation to the binomial distribution. The formula used is:

```r
waldCI <- function(y, n, alpha = 0.05) {
  p_hat <- y/n
  error_margin <- qnorm(1 - alpha/2) * sqrt((p_hat * (1 - p_hat)) / n)
  return(c(p_hat - error_margin, p_hat + error_margin))
}
```

```r
# Modified Wald Interval Function
waldCI <- function(y, n, alpha = 0.05) {
    p_hat <- y / n
    if (p_hat == 0) {
        return(c(0, 0))  # Return (0,0) when p_hat is 0
      }
    else if (p_hat == 1) {
        return(c(1, 1))  # Return (1,1) when p_hat is 1
      }
    else {
        error_margin <- qnorm(1 - alpha / 2) * sqrt(p_hat * (1 - p_hat) / n)
        lower <- p_hat - error_margin
        upper <- p_hat + error_margin
        return(c(lower, upper))
     }
}

# Initialize dataframe with columns for lower and upper CI bounds
results <- data.frame(n = integer(), p = numeric(), coverage = numeric(), lowerCI = numeric(), upperCI =

# Simulations
for (n in n_values) {
  for (p in p_values) {
    # Generate random samples
    sample_successes <- rbinom(num_samples, n, p)

    # Calculate confidence intervals and coverage
    ci_matrix <- t(sapply(sample_successes, waldCI, n))
    coverage <- mean(ci_matrix[,1] <= p & ci_matrix[,2] >= p)

    # Calculate average lower and upper CI bounds
    lowerCI <- mean(ci_matrix[,1])
    upperCI <- mean(ci_matrix[,2])
    average_length = mean(ci_matrix[,2]-ci_matrix[,1])
    # Append results
    results <- rbind(results, data.frame(n, p, coverage, lowerCI, upperCI, average_length))
  }
}

# Filter results for specific n values
waldCI_coverage_data_n15 <- results[results$n == 15, ]
waldCI_coverage_data_n100 <- results[results$n == 100, ]
waldCI_coverage_data_n200 <- results[results$n == 200, ]


waldCI_coverage_plot_n15 <- ggplot(waldCI_coverage_data_n15, aes(x = p, y = coverage)) +
  geom_line(color = "blue") +
  geom_hline(yintercept = 0.95, linetype = "solid", color = "black") +
  scale_y_continuous(limits = c(0, 1)) +  # Set y-axis limits from 0 to 1
  labs(title = "Proportion containing with n = 15 Wald CI",
       x = "Probability (p)", y = "Proportion") +
  theme_minimal()

waldCI_coverage_plot_n100 <- ggplot(waldCI_coverage_data_n100, aes(x = p, y = coverage)) +
  geom_line(color = "blue") +
  geom_hline(yintercept = 0.95, linetype = "solid", color = "black") +
  scale_y_continuous(limits = c(0, 1)) +  # Set y-axis limits from 0 to 1
  labs(title = "Proportion containing with n = 100 Wald CI",
       x = "Probability (p)", y = "Proportion") +
  theme_minimal()

waldCI_coverage_plot_n200 <- ggplot(waldCI_coverage_data_n200, aes(x = p, y = coverage)) +
  geom_line(color = "blue") +
  geom_hline(yintercept = 0.95, linetype = "solid", color = "black") +
  scale_y_continuous(limits = c(0, 1)) +  # Set y-axis limits from 0 to 1
  labs(title = "Proportion containing with n = 200 Wald CI",
       x = "Probability (p)", y = "Proportion") +
  theme_minimal()
```

2. Adjusted Wald Interval

The Adjusted Wald interval is a modification of the Wald interval that includes a correction factor. It is calculated as follows:

```r
# Function to calculate Adjusted Wald confidence intervals
adjustedwaldCI <- function(x, n) {
  # Adjusted estimate and standard error
  p_hat <- (x + 2) / (n + 4)
  se_hat <- sqrt(p_hat * (1 - p_hat) / (n + 4))
  z <- qnorm(0.975)

  # Confidence interval
  lower <- p_hat - z * se_hat
  upper <- p_hat + z * se_hat

  c(lower, upper)
}
```

```r
# Initialize dataframe with columns for lower and upper CI bounds
results <- data.frame(n = integer(), p = numeric(), coverage = numeric(), lowerCI = numeric(), upperCI =


# Simulations
for (n in n_values) {
  for (p in p_values) {
    # Generate random samples
    sample_successes <- rbinom(num_samples, n, p)

    # Calculate confidence intervals and coverage
    ci_matrix <- t(sapply(sample_successes, adjustedwaldCI, n))
    coverage <- mean(ci_matrix[,1] <= p & ci_matrix[,2] >= p)

    # Calculate average lower and upper CI bounds
    lowerCI <- mean(ci_matrix[,1])
    upperCI <- mean(ci_matrix[,2])
    average_length = mean(ci_matrix[,2]-ci_matrix[,1])

    # Append results
    results <- rbind(results, data.frame(n, p, coverage, lowerCI, upperCI, average_length))
  }
}

adjustedWaldCI_coverage_data_n15 <- results[results$n == 15, ]
adjustedWaldCI_coverage_data_n100 <- results[results$n == 100, ]
adjustedWaldCI_coverage_data_n200 <- results[results$n == 200, ]

adjustedWaldCI_coverage_plot_n15 <- ggplot(adjustedWaldCI_coverage_data_n15, aes(x = p, y = coverage)) +
  geom_line(color = "red") +
  geom_hline(yintercept = 0.95, linetype = "solid", color = "black") +
  scale_y_continuous(limits = c(0, 1)) +  # Set y-axis limits from 0 to 1
  labs(title = "Proportion containing with n = 15 Adjusted Wald CI",
       x = "Probability (p)", y = "Proportion") +
  theme_minimal()

adjustedWaldCI_coverage_plot_n100 <- ggplot(adjustedWaldCI_coverage_data_n100, aes(x = p, y = coverage)
  geom_line(color = "red") +
  geom_hline(yintercept = 0.95, linetype = "solid", color = "black") +
  scale_y_continuous(limits = c(0, 1)) +  # Set y-axis limits from 0 to 1
  labs(title = "Proportion containing with n = 100 Adjusted Wald CI",
       x = "Probability (p)", y = "Proportion") +
  theme_minimal()

adjustedWaldCI_coverage_plot_n200 <- ggplot(adjustedWaldCI_coverage_data_n200, aes(x = p, y = coverage)
  geom_line(color = "red") +
  geom_hline(yintercept = 0.95, linetype = "solid", color = "black") +
  scale_y_continuous(limits = c(0, 1)) +  # Set y-axis limits from 0 to 1
  labs(title = "Proportion containing with n = 200 Adjusted Wald CI",
       x = "Probability (p)", y = "Proportion") +
  theme_minimal()
```

3. Clopper-Pearson (Exact) Interval

This interval is calculated using the cumulative probabilities from the beta distribution, corresponding to the cumulative probabilities of the binomial distribution:

```r
clopperPearsonCI <- function(y, n, alpha = 0.05) {
    if (y == 0) {
        return(c(0, 0))  # Return (0,0) when y is 0
    } else if (y == n) {
        return(c(1, 1))  # Return (1,1) when y is n
    } else {
        lower <- qbeta(alpha / 2, y, n - y + 1)
        upper <- qbeta(1 - alpha / 2, y + 1, n - y)
        return(c(lower, upper))
    }
}

# Initialize dataframe with columns for lower and upper CI bounds
results <- data.frame(n = integer(), p = numeric(), coverage = numeric(), lowerCI = numeric(), upperCI =


# Simulations
for (n in n_values) {
  for (p in p_values) {
    # Generate random samples
    sample_successes <- rbinom(num_samples, n, p)

    # Calculate confidence intervals and coverage
    ci_matrix <- t(sapply(sample_successes, clopperPearsonCI, n))
    coverage <- mean(ci_matrix[,1] <= p & ci_matrix[,2] >= p)

    # Calculate average lower and upper CI bounds
    lowerCI <- mean(ci_matrix[,1])
    upperCI <- mean(ci_matrix[,2])
    average_length = mean(ci_matrix[,2]-ci_matrix[,1])

    # Append results
    results <- rbind(results, data.frame(n, p, coverage, lowerCI, upperCI, average_length))
  }
}

Pearson_coverage_data_n15 <- results[results$n == 15, ]
Pearson_coverage_data_n100 <- results[results$n == 100, ]
Pearson_coverage_data_n200 <- results[results$n == 200, ]

Pearson_coverage_plot_n15 <- ggplot(Pearson_coverage_data_n15, aes(x = p, y = coverage)) +
  geom_line(color = "orange") +
  geom_hline(yintercept = 0.95, linetype = "solid", color = "black") +
  scale_y_continuous(limits = c(0, 1)) +
  labs(title = "Proportion containing with n = 15 Clopper-Pearson Interval",
       x = "Probability (p)", y = "Proportion") +
  theme_minimal()

Pearson_coverage_plot_n100 <- ggplot(Pearson_coverage_data_n100, aes(x = p, y = coverage)) +
  geom_line(color = "orange") +
  geom_hline(yintercept = 0.95, linetype = "solid", color = "black") +
  scale_y_continuous(limits = c(0, 1)) +
  labs(title = "Proportion containing with n = 100 Clopper-Pearson Interval",
       x = "Probability (p)", y = "Proportion") +
  theme_minimal()

Pearson_coverage_plot_n200 <- ggplot(Pearson_coverage_data_n200, aes(x = p, y = coverage)) +
  geom_line(color = "orange") +
  geom_hline(yintercept = 0.95, linetype = "solid", color = "black") +
  scale_y_continuous(limits = c(0, 1)) +
  labs(title = "Proportion containing with n = 200 Clopper-Pearson Interval",
       x = "Probability (p)", y = "Proportion") +
  theme_minimal()
```

4. Score Interval

The Wilson interval uses the score method, which is more accurate than the Wald method, especially for small sample sizes:

```
# Function to calculate Score confidence intervals Claire
scoreCI <- function(x, n, alpha = 0.05) {
  z <- qnorm(1 - alpha / 2)
  p_hat <- (x + z^2 / 2) / (n + z^2)
  se_hat <- sqrt(p_hat * (1 - p_hat) / (n + z^2))

  lower <- p_hat - z * se_hat
  upper <- p_hat + z * se_hat
  return(c(lower, upper))
}

# Initialize dataframe with columns for lower and upper CI bounds
results <- data.frame(n = integer(), p = numeric(), coverage = numeric(), lowerCI = numeric(), upperCI =

# Simulations
for (n in n_values) {
  for (p in p_values) {
    # Generate random samples
    sample_successes <- rbinom(num_samples, n, p)

    # Calculate confidence intervals and coverage
    ci_matrix <- t(sapply(sample_successes, scoreCI, n))
    coverage <- mean(ci_matrix[,1] <= p & ci_matrix[,2] >= p)

    # Calculate average lower and upper CI bounds
    lowerCI <- mean(ci_matrix[,1])
    upperCI <- mean(ci_matrix[,2])
    average_length = mean(ci_matrix[,2]-ci_matrix[,1])

    # Append results
    results <- rbind(results, data.frame(n, p, coverage, lowerCI, upperCI, average_length))
  }
}
```

```r
scoreCI_coverage_data_n15 <- results[results$n == 15, ]
scoreCI_coverage_data_n100 <- results[results$n == 100, ]
scoreCI_coverage_data_n200 <- results[results$n == 200, ]

scoreCI_coverage_plot_n15 <- ggplot(scoreCI_coverage_data_n15, aes(x = p, y = coverage)) +
  geom_line(color = "black") +
  geom_hline(yintercept = 0.95, linetype = "solid", color = "black") +
  scale_y_continuous(limits = c(0, 1)) +  # Set y-axis limits from 0 to 1
  labs(title = "Proportion containing with n = 15 Score CI ",
       x = "Probability (p)", y = "Proportion") +
  theme_minimal()

scoreCI_coverage_plot_n100 <- ggplot(scoreCI_coverage_data_n100, aes(x = p, y = coverage)) +
  geom_line(color = "black") +
  geom_hline(yintercept = 0.95, linetype = "solid", color = "black") +
  scale_y_continuous(limits = c(0, 1)) +  # Set y-axis limits from 0 to 1
  labs(title = "Proportion containing with n = 100 Score CI",
       x = "Probability (p)", y = "Proportion") +
  theme_minimal()

scoreCI_coverage_plot_n200 <- ggplot(scoreCI_coverage_data_n200, aes(x = p, y = coverage)) +
  geom_line(color = "black") +
  geom_hline(yintercept = 0.95, linetype = "solid", color = "black") +
  scale_y_continuous(limits = c(0, 1)) +  # Set y-axis limits from 0 to 1
  labs(title = "Proportion containing with n = 200 Score CI",
       x = "Probability (p)", y = "Proportion") +
  theme_minimal()
```

5. Raw Percentile Interval Using a Parametric Bootstrap

For the bootstrap intervals, we resampled the data with replacement to create a distribution of estimates, from which we derived the percentile and t intervals:

```r
# Excluding cases where bootstrap samples result in proportions of 0 or 1
bootstrapCI <- function(y, n, alpha = 0.05, B = 200) {
    if (y == 0) {
        return(c(0, 0))  # Return (0,0) when y is 0
    } else if (y == n) {
        return(c(1, 1))  # Return (1,1) when y is n
    } else {
        p_hat <- y / n
        bootstrap_samples <- replicate(B, sum(rbinom(n, 1, p_hat)) / n)

        # Filter out proportions that are 0 or 1
        valid_samples <- bootstrap_samples[bootstrap_samples > 0 & bootstrap_samples < 1]

        # If all samples are invalid, return NA
        if (length(valid_samples) == 0) {
            return(c(NA, NA))
        } else {
            lower <- quantile(valid_samples, alpha / 2)
            upper <- quantile(valid_samples, 1 - alpha / 2)
            return(c(lower, upper))
        }
    }
}

# Initialize dataframe with columns for lower and upper CI bounds
results <- data.frame(n = integer(), p = numeric(), coverage = numeric(), lowerCI = numeric(), upperCI

# Simulations
for (n in n_values) {
  for (p in p_values) {
    # Generate random samples
    sample_successes <- rbinom(num_samples, n, p)

    # Calculate confidence intervals and coverage
    ci_matrix <- t(sapply(sample_successes, bootstrapCI, n))
    coverage <- mean(ci_matrix[,1] <= p & ci_matrix[,2] >= p)

    # Calculate average lower and upper CI bounds
    lowerCI <- mean(ci_matrix[,1])
    upperCI <- mean(ci_matrix[,2])
    average_length = mean(ci_matrix[,2]-ci_matrix[,1])

    # Append results
    results <- rbind(results, data.frame(n, p, coverage, lowerCI, upperCI, average_length))

  }
}

bootstrapCI_coverage_data_n15 <- results[results$n == 15, ]
bootstrapCI_coverage_data_n100 <- results[results$n == 100, ]
bootstrapCI_coverage_data_n200 <- results[results$n == 200, ]
```

```r
bootstrapCI_coverage_plot_n15 <- ggplot(bootstrapCI_coverage_data_n15, aes(x = p, y = coverage)) +
  geom_line(color = "green") +
  geom_hline(yintercept = 0.95, linetype = "solid", color = "black")+
  scale_y_continuous(limits = c(0, 1)) +
  labs(title = "Proportion containing with n = 15 Raw Percentile Bootstrap ",
       x = "Probability (p)", y = "Proportion") +
  theme_minimal()

bootstrapCI_coverage_plot_n100 <- ggplot(bootstrapCI_coverage_data_n100, aes(x = p, y = coverage)) +
  geom_line(color = "green") +
  geom_hline(yintercept = 0.95, linetype = "solid", color = "black")+
  scale_y_continuous(limits = c(0, 1)) +
  labs(title = "Proportion containing with n = 100 Raw Percentile Bootstrap",
       x = "Probability (p)", y = "Proportion") +
  theme_minimal()

bootstrapCI_coverage_plot_n200 <- ggplot(bootstrapCI_coverage_data_n200, aes(x = p, y = coverage)) +
  geom_line(color = "green") +
  geom_hline(yintercept = 0.95, linetype = "solid", color = "black")+
  scale_y_continuous(limits = c(0, 1)) +
  labs(title = "Proportion containing with n = 200 Raw Percentile Bootstrap",
       x = "Probability (p)", y = "Proportion") +
  theme_minimal()
```

6. Bootstrap t Interval Using a Parametric Bootstrap

```r
bootstrapT <- function(y, n, B = 200, alpha = 0.05) {
  p_hat <- mean(y)
  t_statistics <- numeric(B)

  for (i in 1:B) {
    sample_y <- sample(y, size = n, replace = TRUE)
    sample_p_hat <- mean(sample_y)
    se <- sd(sample_y) / sqrt(n)
    if (se > 0) {
      t_statistics[i] <- (sample_p_hat - p_hat) / se
    } else {
      t_statistics[i] <- NA  # Exclude the sample if SE is 0
    }
  }

  t_statistics <- t_statistics[!is.na(t_statistics) & !is.infinite(t_statistics)]
  t_quantiles <- quantile(t_statistics, c(alpha / 2, 1 - alpha / 2))
  lower_ci <- p_hat - t_quantiles[2] * sd(y) / sqrt(n)
  upper_ci <- p_hat - t_quantiles[1] * sd(y) / sqrt(n)

  return(c(lower_ci, upper_ci))
}

set.seed(123)  # For reproducibility

results <- data.frame(n = integer(), p = numeric(), coverage = numeric(),
                      lowerCI = numeric(), upperCI = numeric(), average_length = numeric())

ci_values <- matrix(nrow = num_samples, ncol = 2)

for (n in n_values) {
  for (p in p_values) {
    coverage_count <- 0
    ci_values <- matrix(nrow = num_samples, ncol = 2)  # Reset ci_values for each (n, p) pair

    for (i in 1:num_samples) {
      y <- rbinom(n, 1, p)
      ci <- bootstrapT(y, n, B = 200, alpha = 0.05)
      ci_values[i, ] <- ci

      # Check for NA before evaluating the condition
      if (!is.na(ci[1]) && !is.na(ci[2]) && ci[1] <= p && p <= ci[2]) {
        coverage_count <- coverage_count + 1
      }
    }

    coverage <- coverage_count / num_samples
    lowerCI_avg <- mean(ci_values[, 1], na.rm = TRUE)
    upperCI_avg <- mean(ci_values[, 2], na.rm = TRUE)
    average_length <- mean(ci_values[, 2] - ci_values[, 1], na.rm = TRUE)

    results <- rbind(results, data.frame(n, p, coverage, lowerCI_avg, upperCI_avg, average_length))
  }
}

bootstrapT_CI_coverage_data_n15 <- results[results$n == 15, ]
bootstrapT_CI_coverage_data_n100 <- results[results$n == 100, ]
bootstrapT_CI_coverage_data_n200 <- results[results$n == 200, ]
```

```r
bootstrapT_CI_coverage_plot_n15 <- ggplot(bootstrapT_CI_coverage_data_n15, aes(x = p, y = coverage)) +
  geom_line(color = "purple") +
  geom_hline(yintercept = 0.95, linetype = "solid", color = "black") +
  scale_y_continuous(limits = c(0, 1)) +
  labs(title = "Proportion containing with n = 15 Bootstrap t",
       x = "Probability (p)", y = "Proportion") +
  theme_minimal()

bootstrapT_CI_coverage_plot_n100 <- ggplot(bootstrapT_CI_coverage_data_n100, aes(x = p, y = coverage))
  geom_line(color = "purple") +
  geom_hline(yintercept = 0.95, linetype = "solid", color = "black") +
  scale_y_continuous(limits = c(0, 1)) +
  labs(title = "Proportion containing with n = 100 Bootstrap t",
       x = "Probability (p)", y = "Proportion") +
  theme_minimal()

bootstrapT_CI_coverage_plot_n200 <- ggplot(bootstrapT_CI_coverage_data_n200, aes(x = p, y = coverage))
  geom_line(color = "purple") +
  geom_hline(yintercept = 0.95, linetype = "solid", color = "black") +
  scale_y_continuous(limits = c(0, 1)) +
  labs(title = "Proportion containing with n = 200 Bootstrap t",
       x = "Probability (p)", y = "Proportion") +
  theme_minimal()
```

Visualization of the results

1. Plot the Proportion

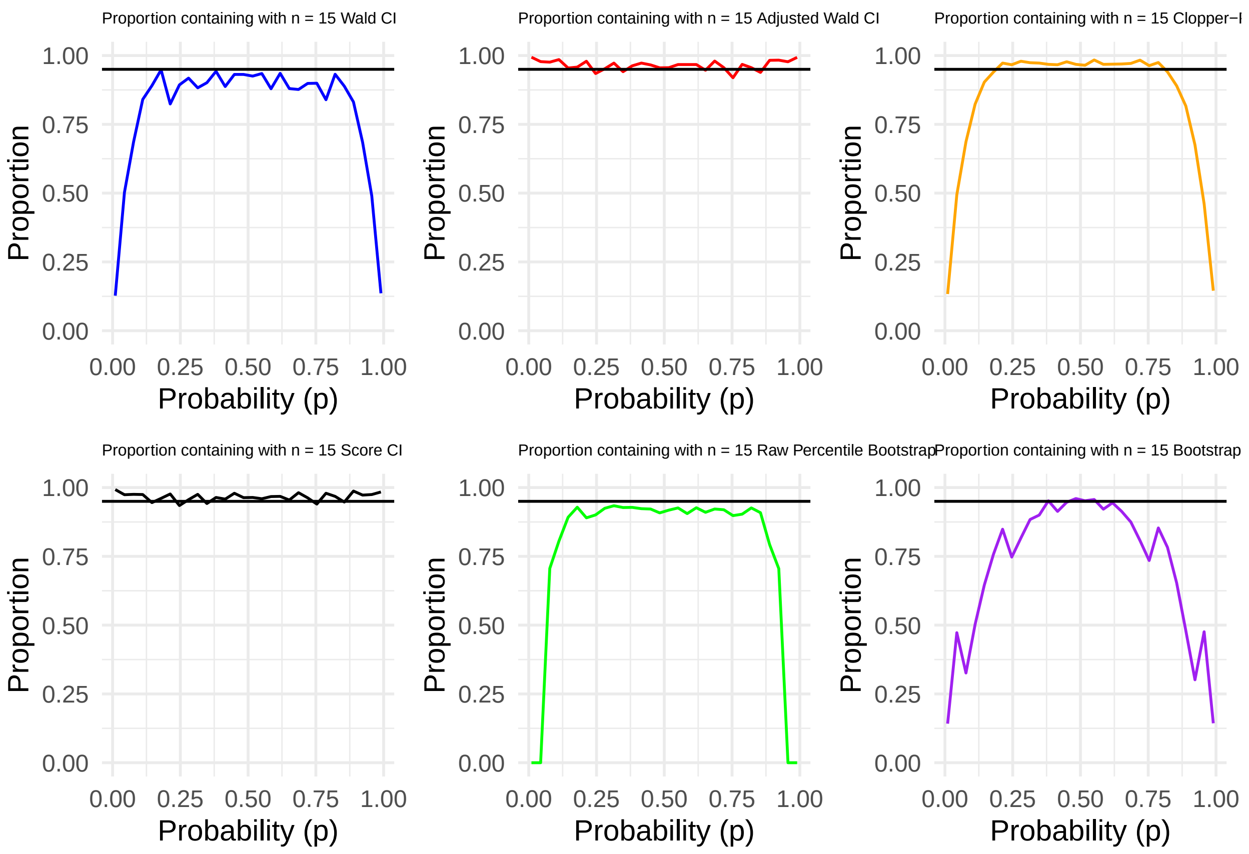
1.1 Proportion Plots when n = 15

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

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

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

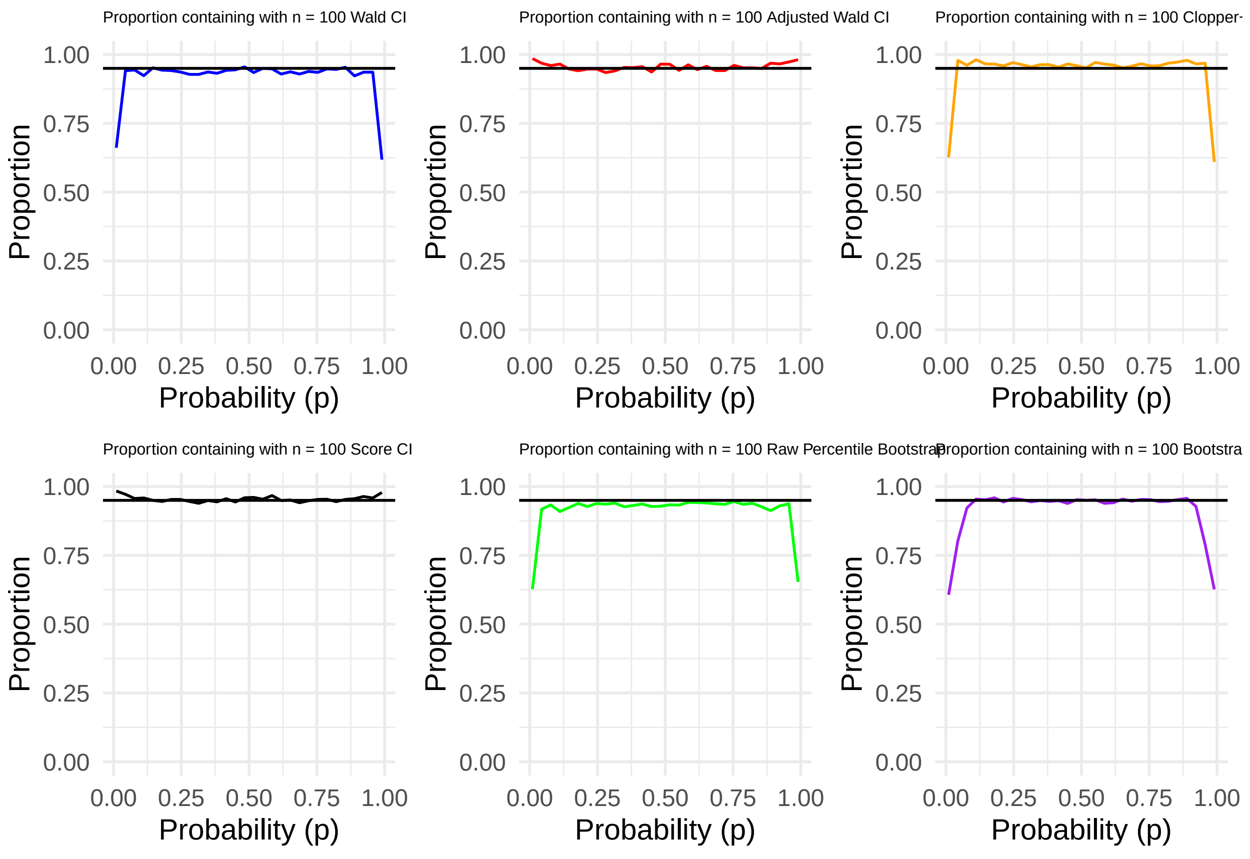
```r
waldCI_coverage_plot_n15 <- waldCI_coverage_plot_n15 + theme(plot.title = element_text(size = 6))
adjustedWaldCI_coverage_plot_n15 <- adjustedWaldCI_coverage_plot_n15 + theme(plot.title = element_text(
Pearson_coverage_plot_n15 <- Pearson_coverage_plot_n15 + theme(plot.title = element_text(size = 6))
```

```r
scoreCI_coverage_plot_n15 <- scoreCI_coverage_plot_n15 + theme(plot.title = element_text(size = 6))
bootstrapCI_coverage_plot_n15 <- bootstrapCI_coverage_plot_n15 + theme(plot.title = element_text(size =
bootstrapT_CI_coverage_plot_n15 <- bootstrapT_CI_coverage_plot_n15 + theme(plot.title = element_text(si


plots <- list(waldCI_coverage_plot_n15, adjustedWaldCI_coverage_plot_n15, Pearson_coverage_plot_n15,
              scoreCI_coverage_plot_n15, bootstrapCI_coverage_plot_n15, bootstrapT_CI_coverage_plot_n15

grid.arrange(
  grobs = plots,
  nrow = 2,
  ncol = 3
)
```



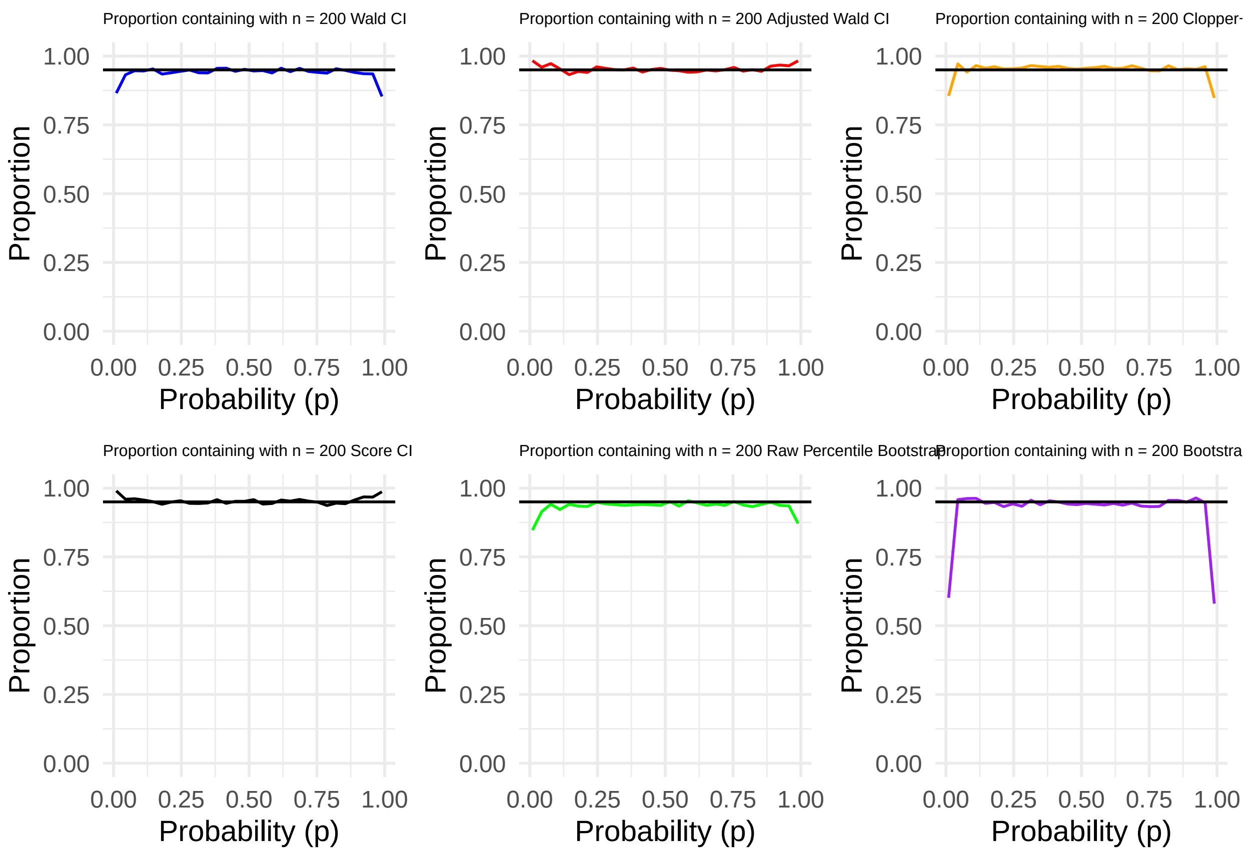
1.2 Proportion Plots when n = 100

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

waldCI_coverage_plot_n100 <- waldCI_coverage_plot_n100 + theme(plot.title = element_text(size = 6))
adjustedWaldCI_coverage_plot_n100 <- adjustedWaldCI_coverage_plot_n100 + theme(plot.title = element_tex
Pearson_coverage_plot_n100 <- Pearson_coverage_plot_n100 + theme(plot.title = element_text(size = 6))
scoreCI_coverage_plot_n100 <- scoreCI_coverage_plot_n100 + theme(plot.title = element_text(size = 6))
bootstrapCI_coverage_plot_n100 <- bootstrapCI_coverage_plot_n100 + theme(plot.title = element_text(size
bootstrapT_CI_coverage_plot_n100 <- bootstrapT_CI_coverage_plot_n100 + theme(plot.title = element_text(
plots <- list(waldCI_coverage_plot_n100, adjustedWaldCI_coverage_plot_n100, Pearson_coverage_plot_n100,
              scoreCI_coverage_plot_n100, bootstrapCI_coverage_plot_n100, bootstrapT_CI_coverage_plot_n

grid.arrange(
  grobs = plots,
  nrow = 2,
  ncol = 3
)
```



1.3 Proportion Plots when n = 200

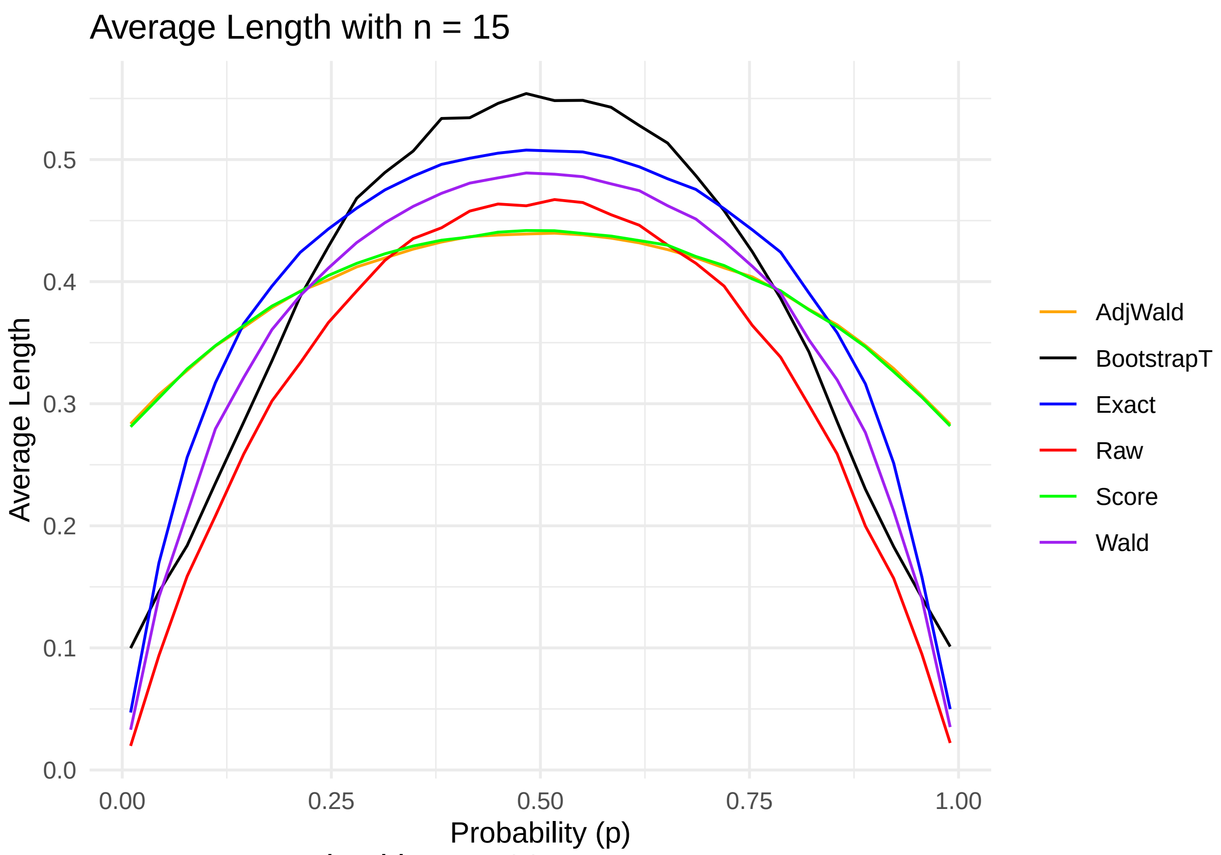
```
library(ggplot2)
library(dplyr)
library(gridExtra)

waldCI_coverage_plot_n200 <- waldCI_coverage_plot_n200 + theme(plot.title = element_text(size = 6))
adjustedWaldCI_coverage_plot_n200 <- adjustedWaldCI_coverage_plot_n200 + theme(plot.title = element_tex
Pearson_coverage_plot_n200 <- Pearson_coverage_plot_n200 + theme(plot.title = element_text(size = 6))
scoreCI_coverage_plot_n200 <- scoreCI_coverage_plot_n200 + theme(plot.title = element_text(size = 6))
bootstrapCI_coverage_plot_n200 <- bootstrapCI_coverage_plot_n200 + theme(plot.title = element_text(size
bootstrapT_CI_coverage_plot_n200 <- bootstrapT_CI_coverage_plot_n200 + theme(plot.title = element_text(

plots <- list(waldCI_coverage_plot_n200, adjustedWaldCI_coverage_plot_n200, Pearson_coverage_plot_n200,
              scoreCI_coverage_plot_n200, bootstrapCI_coverage_plot_n200, bootstrapT_CI_coverage_plot_n

grid.arrange(
  grobs = plots,
  nrow = 2,
  ncol = 3
)
```

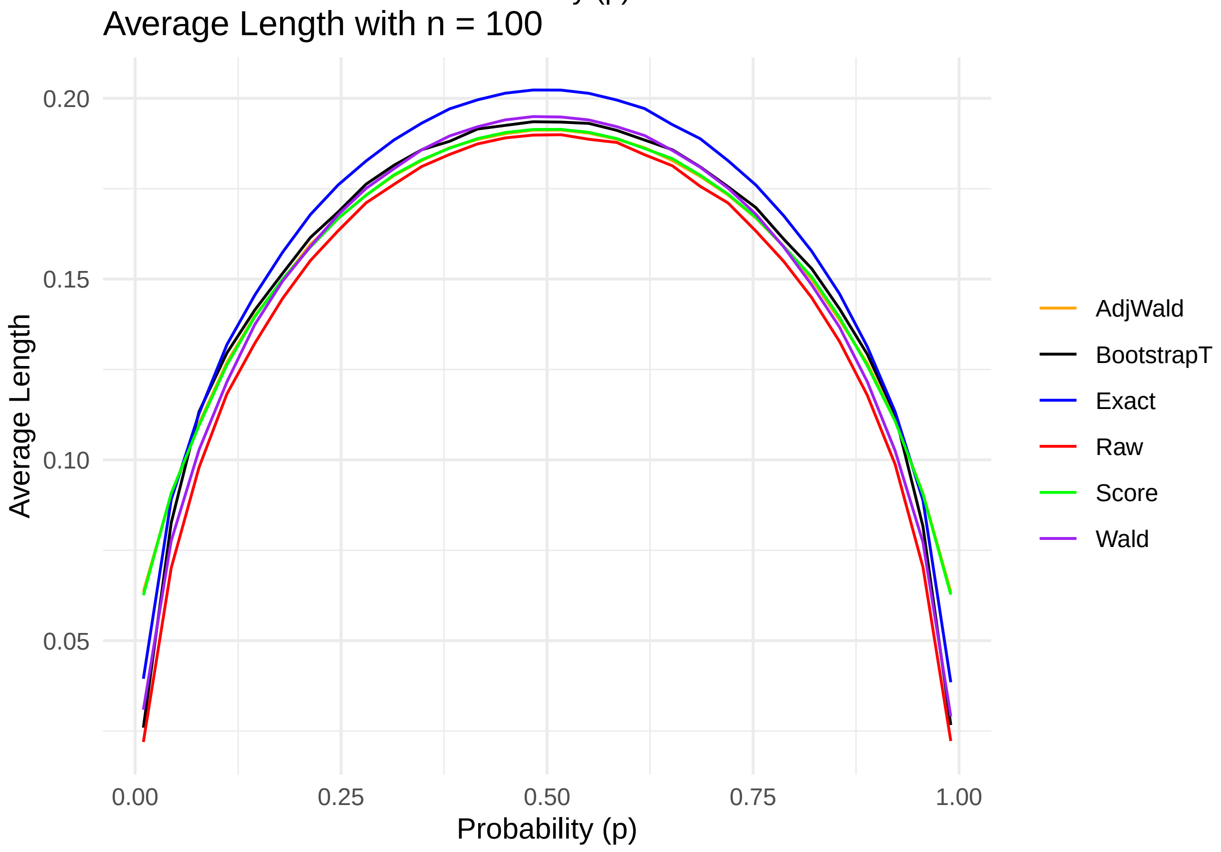


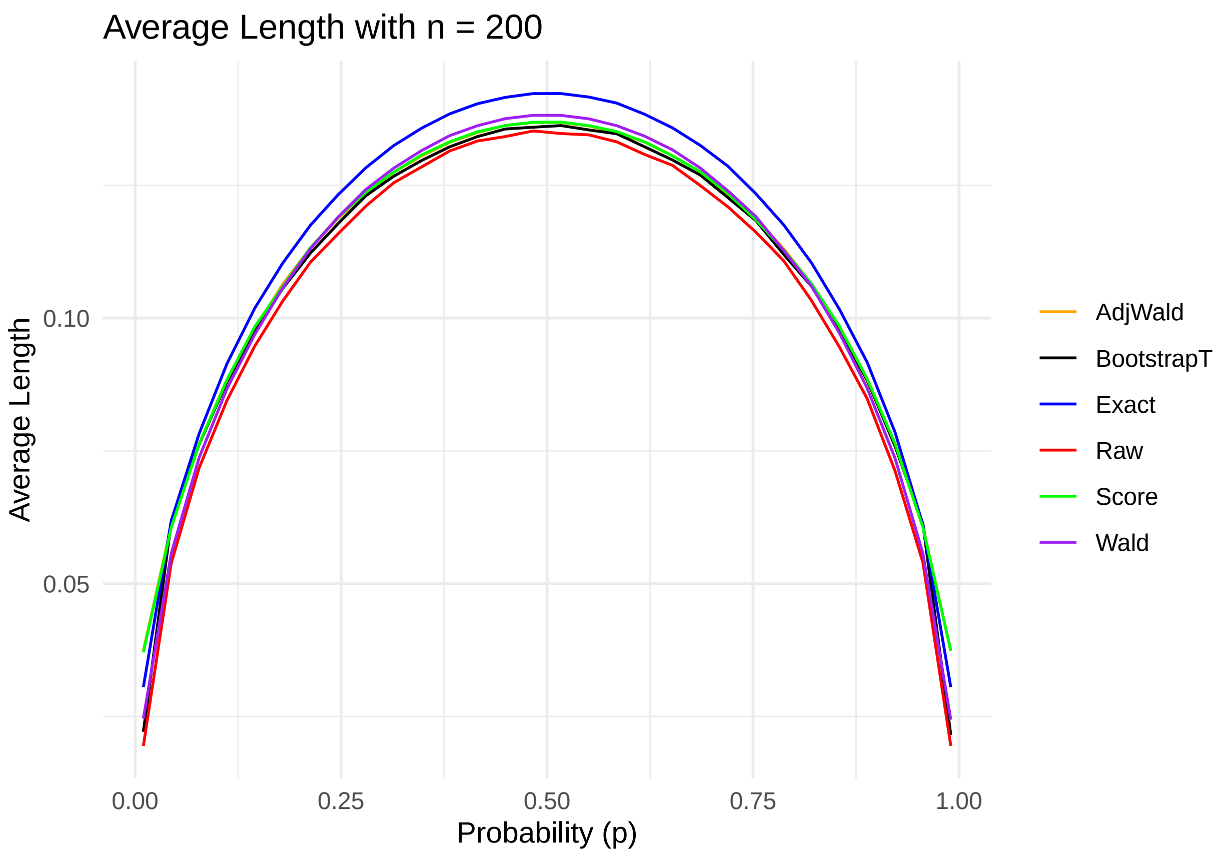
2. Calculate Average Length and Plot

```
library(ggplot2)
library(dplyr)

# Assign 'Method' to each dataset
datasets <- list(
  waldCI_coverage_data_n15 = 'Wald',
  adjustedWaldCI_coverage_data_n15 = 'AdjWald',
  Pearson_coverage_data_n15 = 'Exact',
  scoreCI_coverage_data_n15 = 'Score',
  bootstrapCI_coverage_data_n15 = 'Raw',
  bootstrapT_CI_coverage_data_n15 = 'BootstrapT',
  waldCI_coverage_data_n100 = 'Wald',
  adjustedWaldCI_coverage_data_n100 = 'AdjWald',
  Pearson_coverage_data_n100 = 'Exact',
  scoreCI_coverage_data_n100 = 'Score',
  bootstrapCI_coverage_data_n100 = 'Raw',
  bootstrapT_CI_coverage_data_n100 = 'BootstrapT',
  waldCI_coverage_data_n200 = 'Wald',
  adjustedWaldCI_coverage_data_n200 = 'AdjWald',
```

```r
  Pearson_coverage_data_n200 = 'Exact',
  scoreCI_coverage_data_n200 = 'Score',
  bootstrapCI_coverage_data_n200 = 'Raw',
  bootstrapT_CI_coverage_data_n200 = 'BootstrapT'
)

# Apply 'Method' to each dataframe
for (name in names(datasets)) {
  assign(name, within(get(name), Method <- datasets[[name]]))
}

# Now the loop to combine and plot for each n value
n_values <- c(15, 100, 200)

for (n in n_values) {
  combined_df <- bind_rows(
    get(paste0("waldCI_coverage_data_n", n)),
    get(paste0("adjustedWaldCI_coverage_data_n", n)),
    get(paste0("Pearson_coverage_data_n", n)),
    get(paste0("scoreCI_coverage_data_n", n)),
    get(paste0("bootstrapCI_coverage_data_n", n)),
    get(paste0("bootstrapT_CI_coverage_data_n", n))
  )

  # Plot
  p <- ggplot(combined_df, aes(x = p, y = average_length, group = Method, color = Method)) +
    geom_line() +
    scale_color_manual(values = c('Wald' = 'purple', 'AdjWald' = 'orange', 'Exact' = 'blue',
                                  'Score' = 'green', 'Raw' = 'red', 'BootstrapT' = 'black')) +
    labs(x = 'Probability (p)', y = 'Average Length', title = paste('Average Length with n =', n)) +
    theme_minimal() +
    theme(legend.title = element_blank())

  print(p)
}
```


Average Length with n = 15
Average Length
0.0
0.1
0.2
0.3
0.4
0.5
0.00
0.25
0.50
0.75
1.00
Probability (p)
AdjWald
BootstrapT
Exact
Raw
Score
Wald


Average Length with n = 100
Average Length
0.05
0.10
0.15
0.20
0.00
0.25
0.50
0.75
1.00
Probability (p)
AdjWald
BootstrapT
Exact
Raw
Score
Wald



Conclusion

Based on the analysis of the confidence interval methods for a sample size of $n = 15$, our findings are as follows:

1. Coverage Probability

The graphical analysis revealed that most methods, with the exception of the Wald method, exhibit high and consistent coverage probabilities across varying values of p. This suggests robustness in capturing the true parameter value. The Adjusted Wald and Score methods, in particular, demonstrated near-ideal coverage probability, which aligns well with the theoretical confidence level.

2. Interval Length

The evaluation of interval lengths indicated that the Score and Adjusted Wald methods provide the shortest intervals across most values of p, suggesting a greater degree of precision in the estimation. The Bootstrap t and Raw Percentile Bootstrap methods generated longer intervals, which may translate to less precise estimates.

3. Final conclusion

Considering the balance between coverage probability and precision, the Adjusted Wald and Score methods emerge as the preferred choices for our given sample size. They offer a desirable trade-off, with consistent coverage and more precise interval lengths compared to the other methods analyzed.

The Wald method displayed significant shortcomings, particularly with lower coverage probabilities near the extremes of p. The Exact method, while exhibiting good coverage, tended to produce longer intervals, which could be less practical for precise estimation needs.

In conclusion, the Adjusted Wald and Score methods are recommended for small sample sizes where the balance of interval precision and coverage probability is crucial. Further research may include an exploration of these methods' performance with larger sample sizes or under different distributions, to extend the generalizability of these findings.