Augusta University: STAT 7630

Applied Linear Models

Dustin Pluta

2024 JAN 09

Lecture 4

- Confidence Intervals
- Multiple Testing

Confidence Intervals

- **Definition** A $(1-\alpha)100\%$ **confidence interval** for scalar parameter θ constructed from a sample X is an interval $(L(X), U(X)) \subset (-\infty, \infty)$ such that the probability of such an interval containing θ is $(1-\alpha)100\%$.
- **Definition** We refer to $(1-\alpha)100\%$ as the **coverage** or **coverage probability** of the confidence interval.
- The *coverage* of the interval is the expected proportion of times that the CI will contain the true value θ. This means that the $(1-\alpha)100\%$ confidence level is a probability statement with respect to the *distribution of confidence intervals* constructed this way.

Confidence Intervals

- **Note** In the frequentist framework, we assume paramters are *fixed*, not random.
- Consequently, for a given sample, the corresponding CI either contains θ, or it does not.
- In other words, once the sample is drawn, there is no more randomness, and we cannot make probability statements about the specific CI we constructed.

Confidence Intervals

Example: One-sample Normal, unknown mean and variance

Consider an iid sample $X_i \sim N(\mu, \sigma^2)$. We wish to construct a 95\% confidence interval for μ.

For the one-sample t-test, we use test statistic

$$T = \frac{\bar{X} - \mu_0}{s/\sqrt{n}} \overset{H_0}{\sim} t(n-1).$$

Confidence Intervals

Example: One-sample Normal, unknown mean and variance

To form a 95% percent confidence interval for μ, we invert this hypothesis test.

Note that if we replace μ_0 with the true parameter value μ, $T \sim t(n-1)$. Let $t_{1-\alpha/2}(n-1)$ be the $(1-\alpha/2)$ percentile of t.

$$P(-t_{1-\alpha/2}(n-1) < T < t_{1-\alpha/2}) = 1-\alpha$$

$$P(-t_{1-\alpha/2}(n-1) < \frac{\bar{X}-\mu}{s/\sqrt{n}} < t_{1-\alpha/2}) = 1-\alpha$$

$$P\left(\bar{X} - \frac{s}{\sqrt{n}} t_{1-\alpha/2}(n-1) < \mu < \bar{X} + \frac{s}{\sqrt{n}} t_{1-\alpha/2}(n-1)\right) = 1-\alpha$$

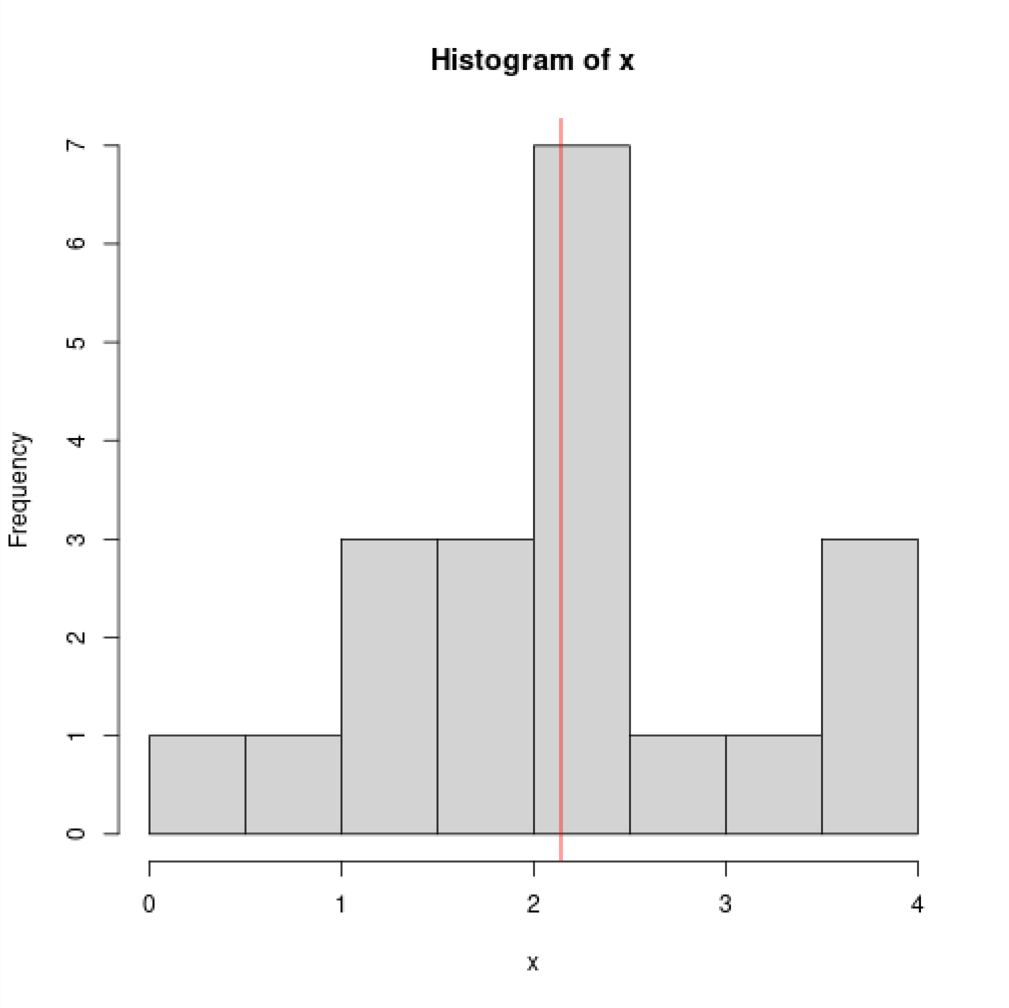
Confidence Intervals

Example: One-sample Normal, unknown mean and variance

```
set.seed(123)
n ← 20
mu ← 2
sigma ← 1
x ← rnorm(n, mu, sigma)
x_bar ← mean(x)
s ← sd(x)
```

Confidence Intervals

Example: One-sample Normal, unknown mean and variance



Confidence Intervals

Example: One-sample Normal

```
alpha <- 0.05
lwr <- x_bar - s / sqrt(n) * pt(1 - alpha / 2, n - 1)
upr <- x_bar + s / sqrt(n) * pt(1 - alpha / 2, n - 1)

cat(lwr, upr)

## 1.9613 2.321947
```

We see this confidence interval contains the true value $\mu = 2$

- What happens if we sample X many times and form a confidence interval for each?

Confidence Intervals

Example: One-sample Normal

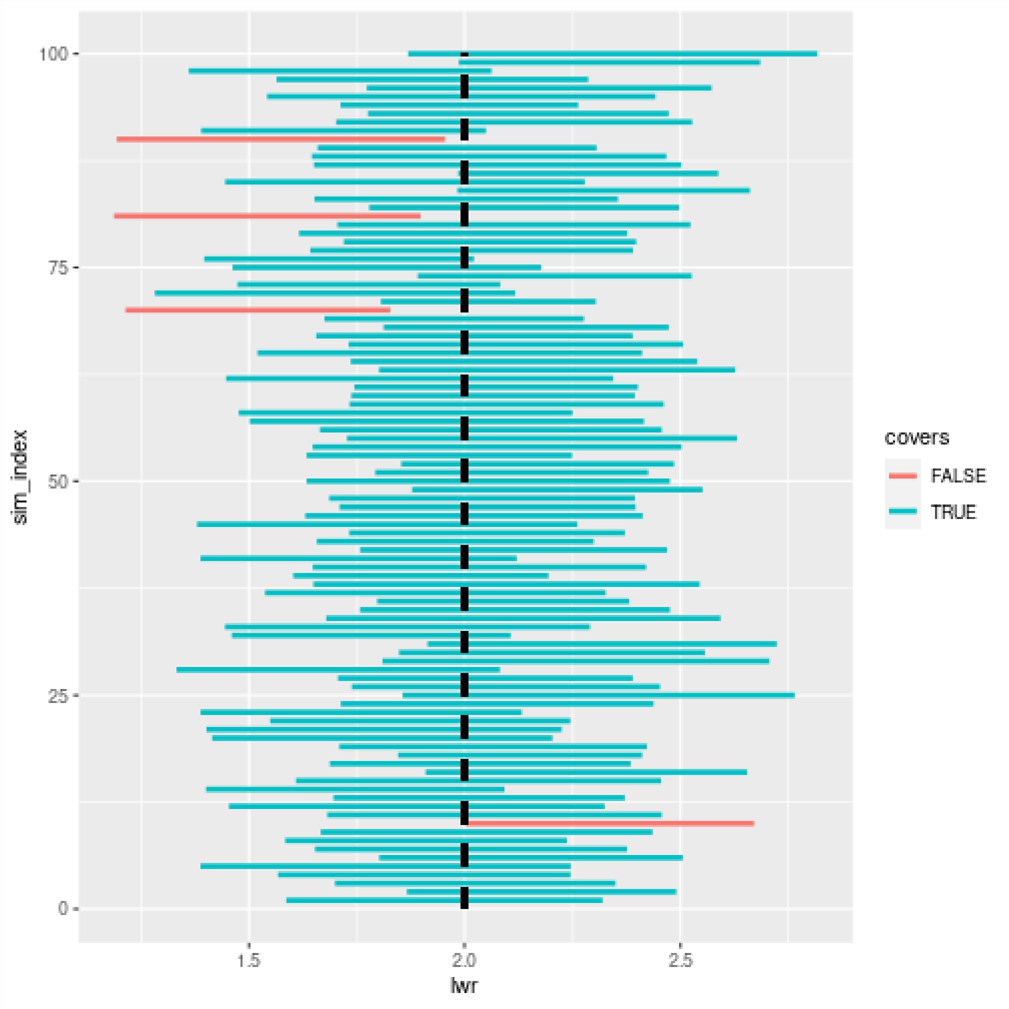
```
conf_int_simulation ← function(n_sims, n, mu, sigma, alpha = 0.05) {
  results ← data.frame(lwr = rep(NA, n_sims), upr = rep(NA, n_sims))

  for (k in 1:n_sims) {
    x ← rnorm(n, mu, sigma)
    x_bar ← mean(x)
    s ← sd(x)
    lwr ← x_bar - s / sqrt(n) * qt(1 - alpha / 2, n - 1)
    upr ← x_bar + s / sqrt(n) * qt(1 - alpha / 2, n - 1)
    results[k, ] ← c(lwr, upr)
  }
  results$sim_index ← 1:nrow(results)
  results$covers ← (mu > results$lwr) & (mu < results$upr)

  return(results)
}
```

Confidence Intervals

```r
set.seed(123)
n_sims <- 100
n <- 30
mu <- 2
sigma <- 1
alpha <- 0.05

sim_results <- conf_int_simulation(n_sims, n, mu, sigma, alpha)
head(sim_results)
```

```
##        lwr      upr sim_index covers
## 1 1.586573 2.319219         1   TRUE
## 2 1.866496 2.490180         2   TRUE
## 3 1.699634 2.349207         3   TRUE
## 4 1.567479 2.244743         4   TRUE
## 5 1.387571 2.245269         5   TRUE
## 6 1.802118 2.505315         6   TRUE

## Coverage:  0.96
```

Confidence Intervals



Confidence Intervals

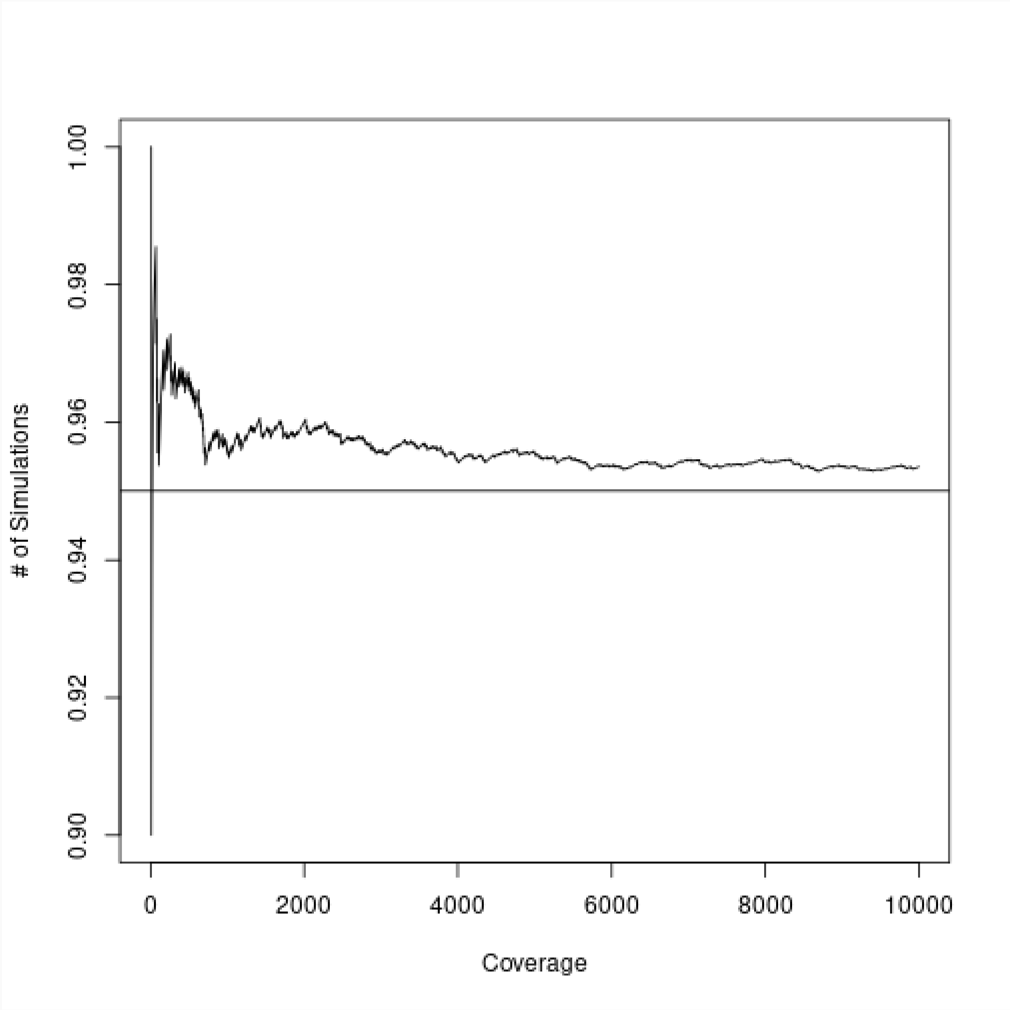
Example: One-sample Normal

```r
set.seed(123)
n_sims <- 10000
n <- 30
mu <- 2
sigma <- 1
alpha <- 0.05

sim_results <- conf_int_simulation(n_sims, n, mu, sigma, alpha)
mean(sim_results$covers)
```

```
## [1] 0.9535
```

Confidence Intervals

Example: One-sample Normal, unknown mean and variance



Multiple Testing

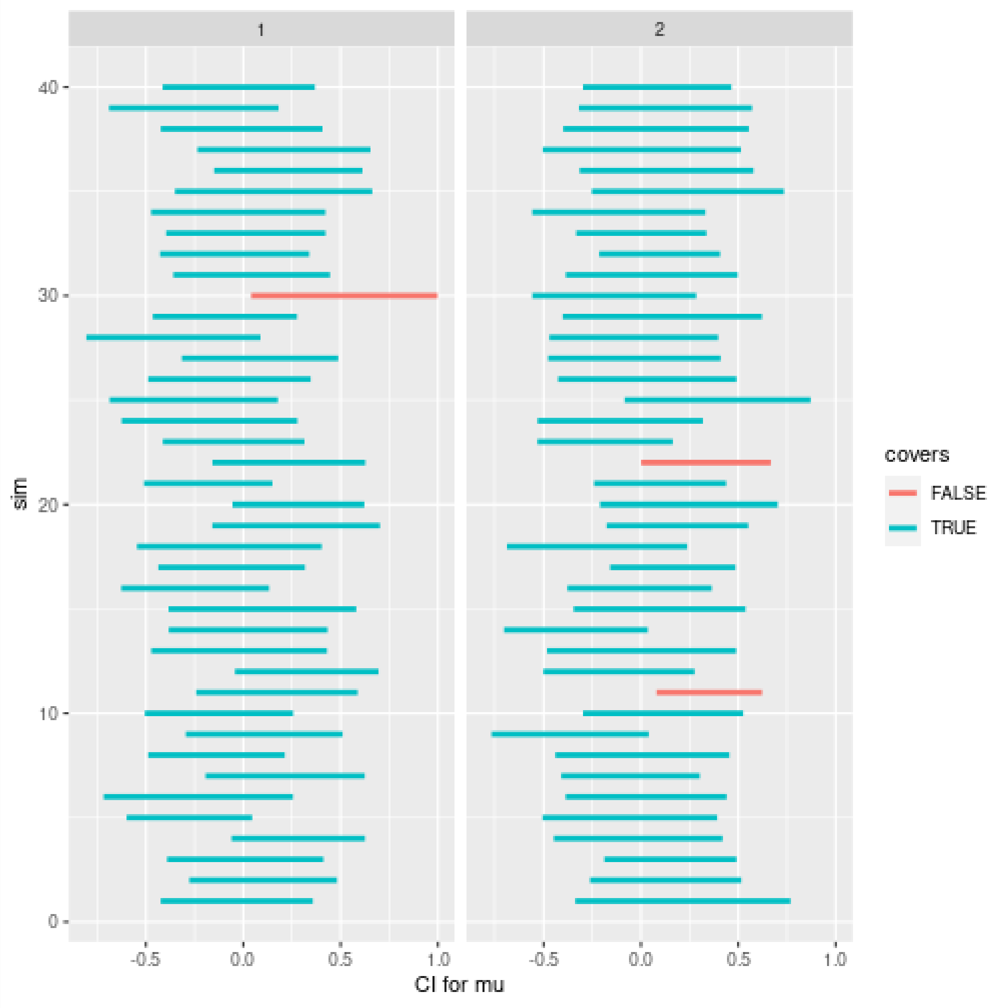
```
set.seed(123)
n_sims ← 40
n ← 25
mu1 ← 0
mu2 ← 0
sigma ← 1
alpha ← 0.05

sim_results1 ← conf_int_simulation(n_sims, n, mu1, sigma, alpha)
sim_results2 ← conf_int_simulation(n_sims, n, mu2, sigma, alpha)
combined_results ← data.frame(covers1 = sim_results1$covers, covers2 = sim_results2$covers
combined_results$covers_all ← combined_results$covers1 & combined_results$covers2
```

```
## Combined coverage probability:  0.925
```

```
## Group 1 coverage:  0.975
```

```
## Group 2 coverage:  0.95
```

Multiple Testing



Multiple Testing

```
set.seed(123)
n_sims <- 40
n <- 30
mu1 <- 0
mu2 <- 0
mu3 <- 0
sigma <- 1
alpha <- 0.05
```

```
## Combined coverage probability:  0.9
```

```
## Group 1 coverage:  0.975
```

```
## Group 2 coverage:  0.975
```

```
## Group 3 coverage:  0.925
```

Multiple Testing

```
set.seed(123)
n_sims <- 10000
n <- 30
mu1 <- 0
mu2 <- 0
mu3 <- 0
mu4 <- 0
sigma <- 1
alpha <- 0.05
```

```
## Combined coverage probability:  0.8179
```

```
## Group 1 coverage:  0.9535
```

```
## Group 2 coverage:  0.9504
```

```
## Group 3 coverage:  0.9474
```

```
## Group 4 coverage:  0.9516
```

Bonferroni Correction

One method for controlling the familywise Type I error rate is the *Bonferroni correction*.

- When conduction K hypothesis tests simultaneously, the Bonferroni method adjusts the significance level from α to α/K.
- When constructing multiple confidence intervals for the one-sample normal case, this has the form

$$\bar{X} \pm \frac{s}{\sqrt{n}} t_{1-\alpha/(2K)}(n-1).$$

- The name of this method is from the Bonferroni inequality from probability theory, which can be stated as

$$P\left(\cup_{k=1}^{K} A_k\right) \leq \sum_{k=1}^{K} P(A_k),$$

for some set of events $A_k, k = 1, \ldots, K$.

Bonferroni Correction

```
bonferroni_simulation ← function(n_sims, n, mu, sigma, K, alpha = 0.05) {
  results ← data.frame(lwr = rep(NA, n_sims), upr = rep(NA, n_sims))

  for (k in 1:n_sims) {
    x ← rnorm(n, mu, sigma)
    x_bar ← mean(x)
    s ← sd(x)
    lwr ← x_bar - s / sqrt(n) * qt(1 - alpha / (2 * K), n - 1)
    upr ← x_bar + s / sqrt(n) * qt(1 - alpha / (2 * K), n - 1)
    results[k, ] ← c(lwr, upr)
  }
  results$sim_index ← 1:nrow(results)
  results$covers ← (mu > results$lwr) & (mu < results$upr)

  return(results)
}
```

Bonferroni Correction

```
set.seed(12)
n_sims ← 40
n ← 25
mu1 ← 0
mu2 ← 0
sigma ← 1
K ← 2
alpha ← 0.05
```

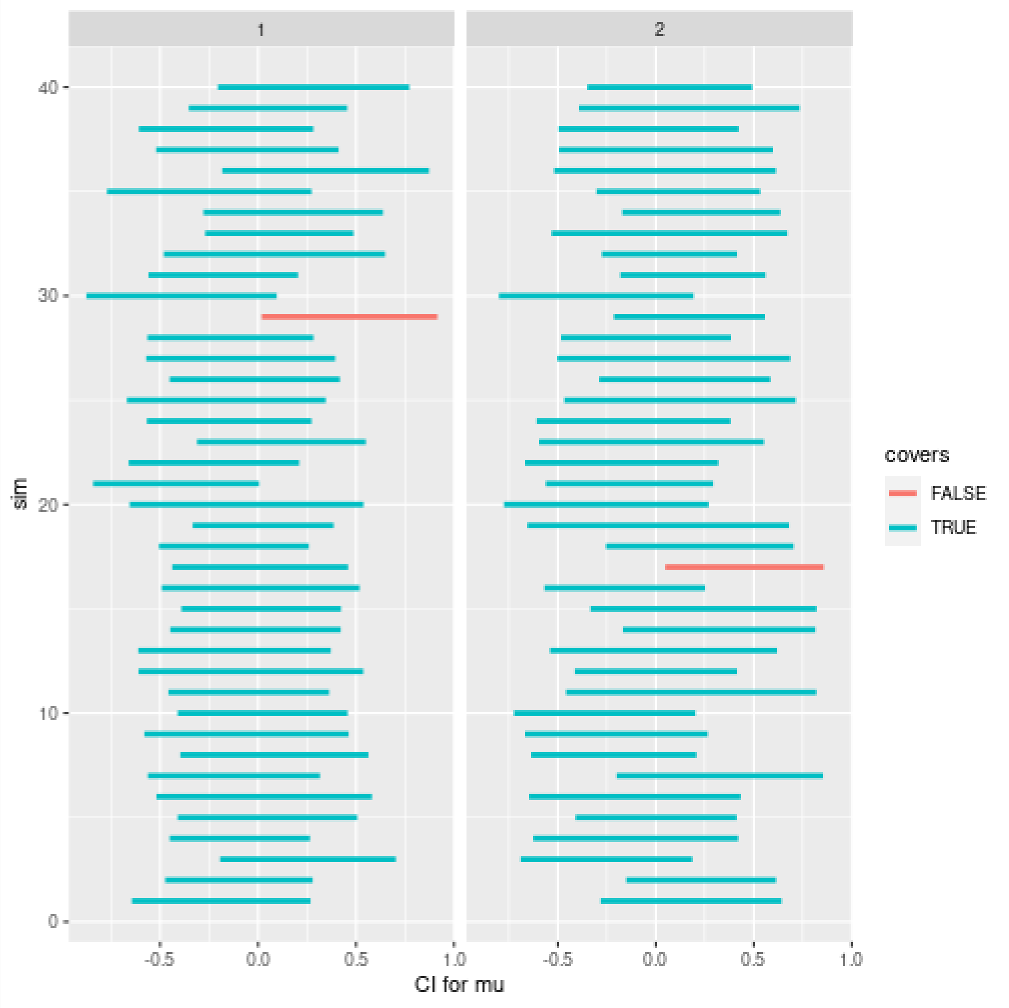
```
## Combined coverage probability:  0.95
```

```
## Group 1 coverage:  0.975
```

```
## Group 2 coverage:  0.975
```

Multiple Testing

Bonferroni Correction



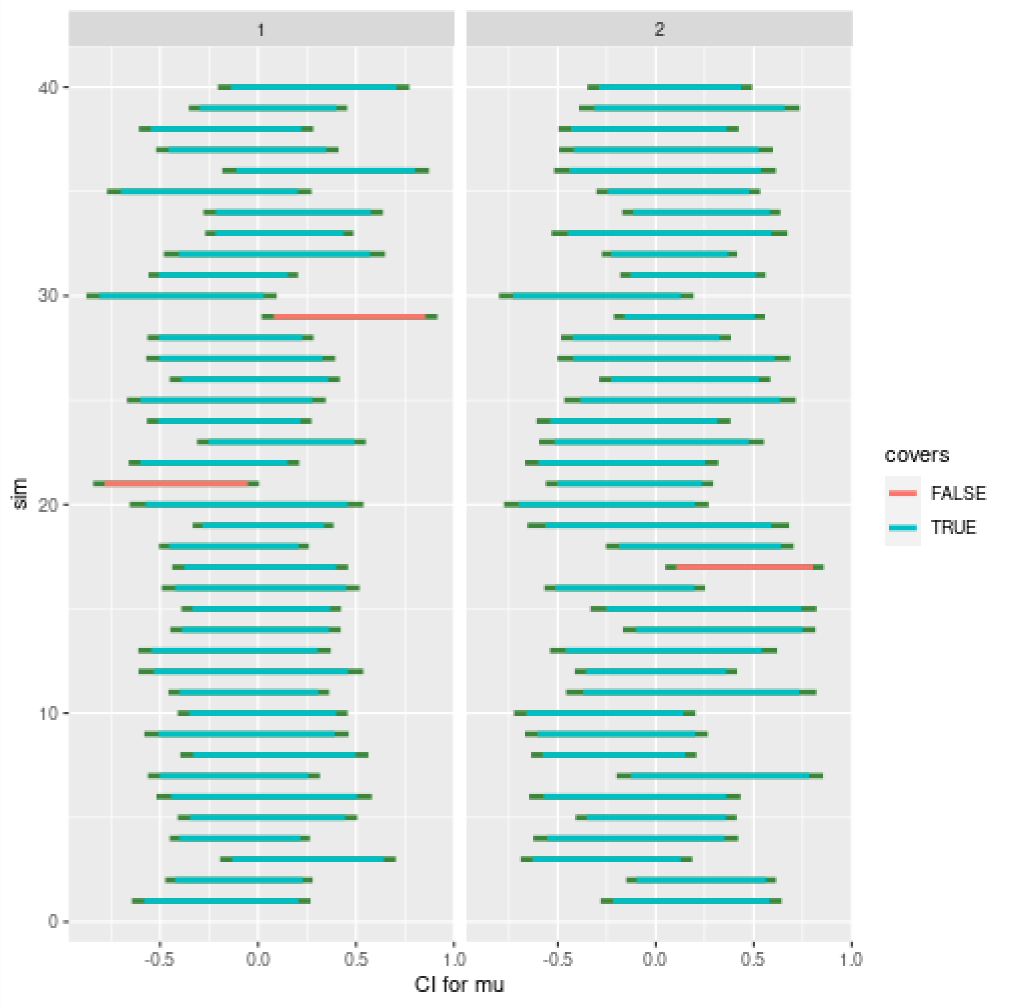
Bonferroni Correction

```
set.seed(12)
n_sims <- 40
n <- 25
mu1 <- 0
mu2 <- 0
sigma <- 1
K <- 2
alpha <- 0.05
```

```
set.seed(12)
bonf_sim_results1 <- bonferroni_simulation(n_sims, n, mu1, sigma, K, alpha)
bonf_sim_results2 <- bonferroni_simulation(n_sims, n, mu2, sigma, K, alpha)
set.seed(12)
sim_results1 <- conf_int_simulation(n_sims, n, mu1, sigma, alpha)
sim_results2 <- conf_int_simulation(n_sims, n, mu2, sigma, alpha)
```

Multiple Testing

Bonferroni Correction



Bonferroni Correction

```
set.seed(123)
n_sims ← 5000
n ← 30
mu1 ← 0
mu2 ← 0
mu3 ← 0
mu4 ← 0
sigma ← 1
alpha ← 0.05
K ← 4
```

```
## Combined coverage probability:  0.9538
```

```
## Group 1 coverage:  0.9888
```

```
## Group 2 coverage:  0.984
```

```
## Group 3 coverage:  0.9898
```

```
## Group 4 coverage:  0.9898
```