

# TITLE HERE

A Thesis Proposal Presented to  
The Faculty of the SS  
univ

In Partial Fulfillment  
of the Requirements for the Degree of  
degree  
1st Semester A.Y. 2025-2026

by  
Nmae

# Abstract

# Contents

<b>Abstract</b>	<b>2</b>
<b>1 Methodology</b>	<b>4</b>
1.1 Parametric bootstrap . . . . .	4
1.2 Nonrank-based method . . . . .	5
1.3 Results . . . . .	6
<b>2 Introduction</b>	<b>9</b>
2.1 Background of the Study . . . . .	9
2.2 Statement of the Problem . . . . .	9
2.3 Objective of the Study . . . . .	9
2.4 Study Hypothesis . . . . .	9
2.5 Significance of the Study . . . . .	9
2.6 Scope and Limitation . . . . .	9
2.7 Definition of Terms . . . . .	9
<b>3 Background</b>	<b>9</b>
<b>References</b>	<b>9</b>
<b>Appendices</b>	<b>9</b>
Codes for algorithm 1 . . . . .	9
Codes for algorithm 2 . . . . .	13
Codes for simulation . . . . .	19

# 1 Methodology

Let  $\theta_1, \theta_2, \dots, \theta_K$  be the true parameter values and  $\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_K$  be the estimates obtained.

## 1.1 Parametric bootstrap

Let  $\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_K$  be independent but not identically distributed estimates. For this study, it is assumed that  $\hat{\theta}_k \sim N(\theta_k, \sigma_k^2)$ ,  $k = 1, 2, \dots, K$ , where  $\sigma_k^2$  is known. Denote the corresponding ordered values by  $\hat{\theta}_{(1)}, \hat{\theta}_{(2)}, \dots, \hat{\theta}_{(K)}$ .

---

### Algorithm 1 Computation of Joint Confidence Region via Parametric Bootstrap

---

1: **for**  $b = 1, 2, \dots, B$  **do**

2:   Generate  $\hat{\theta}_{bk}^* \sim N(\hat{\theta}_k, \sigma_k^2)$ ,  $i = 1, 2, \dots, K$  and let  $\hat{\theta}_{b(1)}, \hat{\theta}_{b(2)}, \dots, \hat{\theta}_{b(K)}$  be the corresponding ordered values

	$k = 1$	$k = 2$	$\dots$	$k = K$
$b = 1$	$\hat{\theta}_{1(1)}^*$	$\hat{\theta}_{1(2)}^*$	$\dots$	$\hat{\theta}_{1(K)}^*$
$b = 2$	$\hat{\theta}_{2(1)}^*$	$\hat{\theta}_{2(2)}^*$	$\dots$	$\hat{\theta}_{2(K)}^*$
$\vdots$	$\vdots$	$\vdots$	$\dots$	$\vdots$
$b = B$	$\hat{\theta}_{B(1)}^*$	$\hat{\theta}_{B(2)}^*$	$\dots$	$\hat{\theta}_{B(K)}^*$

3:   Compute

$$\hat{\sigma}_{b(k)}^* = \sqrt{\text{kth ordered value among } \{\hat{\theta}_{b1}^{*2} + \sigma_1^2, \hat{\theta}_{b2}^{*2} + \sigma_2^2, \dots, \hat{\theta}_{bK}^{*2} + \sigma_K^2\} - \hat{\theta}_{(k)}^{*2}}$$

4:   Compute  $t_b^* = \max_{1 \leq k \leq K} \left| \frac{\hat{\theta}_{b(k)}^* - \hat{\theta}_k^*}{\sigma_{b(k)}^*} \right|$

5: **end for**

6: Compute the  $(1 - \alpha)$ -sample quantile of  $t_1^*, t_2^*, \dots, t_B^*$ , call this  $\hat{t}$ .

7: The joint confidence region of  $\theta_{(1)}, \theta_{(2)}, \dots, \theta_{(K)}$  is given by

$$\mathfrak{R} = [\hat{\theta}_{(1)} \pm \hat{t} \times \hat{\sigma}_{(1)}] \times [\hat{\theta}_{(2)} \pm \hat{t} \times \hat{\sigma}_{(2)}] \times \dots \times [\hat{\theta}_{(K)} \pm \hat{t} \times \hat{\sigma}_{(K)}]$$

where  $\hat{\sigma}_{(k)}$  is computed as

$$\hat{\sigma}_{(k)} = \sqrt{\text{kth ordered value among } \{\hat{\theta}_1^2 + \sigma_1^2, \hat{\theta}_2^2 + \sigma_2^2, \dots, \hat{\theta}_K^2 + \sigma_K^2\} - \hat{\theta}_{(k)}^2}$$


---

---

**Algorithm 2** Computation of Coverage Probability for Parametric Bootstrap

---

For given values of  $\theta_1, \theta_2, \dots, \theta_K$  and thus  $\theta_{(1)}, \theta_{(2)}, \dots, \theta_{(K)}$

1: **for** replications = 1, 2,  $\dots$ , 5000 **do**

2:   Generate  $\hat{\theta}_k \sim N(\theta_k, \sigma_k^2)$ , for  $k = 1, 2, \dots, K$

3:   Compute the rectangular confidence region  $\mathfrak{R}$  using Algorithm 1.

4:   Check if  $(\theta_{(1)}, \theta_{(2)}, \dots, \theta_{(K)}) \in \mathfrak{R}$  and compute

$$\begin{aligned} T_1 &= \frac{1}{K} \sum_{k=1}^K |\Lambda_{Ok}| \\ T_2 &= \prod_{k=1}^K |\Lambda_{Ok}| \\ T_3 &= 1 - \frac{K + \sum_{k=1}^K |\Lambda_{Ok}|}{K^2} \end{aligned}$$

5: **end for**

6: Compute the proportion of times that the condition in step 4 is satisfied and the average of  $T_1, T_2$ , and  $T_3$ .

---

## 1.2 Nonrank-based method

The nonrank-based method assumes that  $\hat{\boldsymbol{\theta}} = (\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_K) \sim N(\boldsymbol{\theta}, \boldsymbol{\Sigma})$ . It accounts for potential correlation among items being ranked. For this case, an exchangeable correlation,  $\boldsymbol{\rho}$  (See Equation 1.1.), is assumed and used in the calculation of the variance covariance matrix (See Equation 1.2.).

$$\boldsymbol{\rho} = (1 - \rho) \mathbf{I}_K + \rho \mathbf{1}_K \mathbf{1}_K' \quad (1.1)$$

$$\boldsymbol{\Sigma} = \boldsymbol{\Delta}^{1/2} \boldsymbol{\rho} \boldsymbol{\Delta}^{1/2} \quad (1.2)$$

where  $\boldsymbol{\Delta} = \text{diag}\{\sigma_1^2, \sigma_2^2, \dots, \sigma_K^2\}$ , with known  $\sigma_k$ 's and  $\rho$  is studied for 0.1, 0.5, 0.9.

---

**Algorithm 3** Computation of Joint Confidence Region via Nonrank-based Method

---

Let the data consist of  $\hat{\theta}_1, \dots, \hat{\theta}_K$  and suppose  $\Sigma$  is known

- 1: **for**  $b = 1, 2, \dots, B$  **do**
- 2:     Generate  $\hat{\theta}_b^* \sim N_K(\hat{\theta}, \Sigma)$  and write  $\hat{\theta}_b^* = (\hat{\theta}_{b1}^*, \hat{\theta}_{b2}^*, \dots, \hat{\theta}_{bK}^*)'$
- 3:     Compute  $t_b^* = \max_{1 \leq j \leq K} \left| \frac{\hat{\theta}_{bj}^* - \hat{\theta}_j}{\sigma_j} \right|$
- 4: **end for**
- 5: Compute the  $(1 - \alpha)$ -sample quantile of  $t_1^*, t_2^*, \dots, t_B^*$ , call this  $\hat{t}$ .
- 6: The joint confidence region of  $\theta_1, \theta_2, \dots, \theta_K$  is given by

$$\mathfrak{R} = [\hat{\theta}_1 \pm \hat{t} \times \sigma_1] \times [\hat{\theta}_2 \pm \hat{t} \times \sigma_2] \times \dots \times [\hat{\theta}_K \pm \hat{t} \times \sigma_K]$$

---

---

**Algorithm 4** Computation of Coverage Probability for Nonrank-based Method

---

For given values of  $\theta_1, \theta_2, \dots, \theta_K$  and  $\Sigma$

- 1: **for** replications = 1, 2,  $\dots$ , 5000 **do**
  - 2:     Generate  $\hat{\theta} \sim N_K(\theta, \Sigma)$
  - 3:     Compute the rectangular confidence region  $\mathfrak{R}$  using Algorithm 3.
  - 4:     Check if  $(\theta_1, \theta_2, \dots, \theta_K) \in \mathfrak{R}$  and compute  $T_1, T_2$ , and  $T_3$ .
  - 5: **end for**
  - 6: Compute the proportion of times that the condition in step 4 is satisfied and the average of  $T_1, T_2$ , and  $T_3$ .
- 

### 1.3 Results

---

K	sd	corr	Cov_nonrankbased	Cov_independent	Cov_bonferroni
51	2.0	0.1	0.8944	0.8994	0.9048
40	2.0	0.1	0.8916	0.8968	0.9012
30	2.0	0.1	0.9000	0.9036	0.9088
20	2.0	0.1	0.8988	0.9026	0.9082
10	2.0	0.1	0.8996	0.9016	0.9060
5	2.0	0.1	0.8984	0.9008	0.9046

---

51	3.6	0.1	0.8944	0.8994	0.9048
40	3.6	0.1	0.8916	0.8968	0.9012
30	3.6	0.1	0.9000	0.9036	0.9088
20	3.6	0.1	0.8988	0.9026	0.9082
10	3.6	0.1	0.8996	0.9016	0.9060
5	3.6	0.1	0.8984	0.9008	0.9046
51	6.0	0.1	0.8944	0.8994	0.9048
40	6.0	0.1	0.8916	0.8968	0.9012
30	6.0	0.1	0.9000	0.9036	0.9088
20	6.0	0.1	0.8988	0.9026	0.9082
10	6.0	0.1	0.8996	0.9016	0.9060
5	6.0	0.1	0.8984	0.9008	0.9046
51	2.0	0.5	0.9086	0.9492	0.9516
40	2.0	0.5	0.8920	0.9328	0.9358
30	2.0	0.5	0.8910	0.9312	0.9338
20	2.0	0.5	0.9032	0.9378	0.9408
10	2.0	0.5	0.9038	0.9322	0.9346
5	2.0	0.5	0.9042	0.9218	0.9260
51	3.6	0.5	0.9086	0.9492	0.9516
40	3.6	0.5	0.8920	0.9328	0.9358
30	3.6	0.5	0.8910	0.9312	0.9338
20	3.6	0.5	0.9032	0.9378	0.9408
10	3.6	0.5	0.9038	0.9322	0.9346
5	3.6	0.5	0.9042	0.9218	0.9260
51	6.0	0.5	0.9086	0.9492	0.9516
40	6.0	0.5	0.8920	0.9328	0.9358
30	6.0	0.5	0.8910	0.9312	0.9338
20	6.0	0.5	0.9032	0.9378	0.9408
10	6.0	0.5	0.9038	0.9322	0.9346
5	6.0	0.5	0.9042	0.9218	0.9260

51	2.0	0.9	0.8928	0.9866	0.9868
40	2.0	0.9	0.8996	0.9862	0.9870
30	2.0	0.9	0.8960	0.9802	0.9806
20	2.0	0.9	0.8980	0.9758	0.9766
10	2.0	0.9	0.8962	0.9682	0.9690
5	2.0	0.9	0.9032	0.9574	0.9586
51	3.6	0.9	0.8928	0.9866	0.9868
40	3.6	0.9	0.8996	0.9862	0.9870
30	3.6	0.9	0.8960	0.9802	0.9806
20	3.6	0.9	0.8980	0.9758	0.9766
10	3.6	0.9	0.8962	0.9682	0.9690
5	3.6	0.9	0.9032	0.9574	0.9586
51	6.0	0.9	0.8928	0.9866	0.9868
40	6.0	0.9	0.8996	0.9862	0.9870
30	6.0	0.9	0.8960	0.9802	0.9806
20	6.0	0.9	0.8980	0.9758	0.9766
10	6.0	0.9	0.8962	0.9682	0.9690
5	6.0	0.9	0.9032	0.9574	0.9586

---

tbl\_summary(trial) %>% as\_flex\_table()



## 2 Introduction

### 2.1 Background of the Study

### 2.2 Statement of the Problem

### 2.3 Objective of the Study

### 2.4 Study Hypothesis

### 2.5 Significance of the Study

### 2.6 Scope and Limitation

### 2.7 Definition of Terms

## 3 Background

THIS IS Rizzo (2008) and Klein et al. (2020)

## References

Klein, M., Wright, T., & Wieczorek, J. (2020). *A joint confidence region for an overall ranking of populations.*

Rizzo, M. (2008). *Statistical computing with r.*

## Appendices

### Codes for algorithm 1

```
get_independent_ci <- function(theta_hat,  
                                S,  
                                alpha){
```

```

K <- length(theta_hat)
gamma = 1-(1-alpha)^(1/K)
z = qnorm(1-gamma/2)
ci_lower <- theta_hat - z*S
ci_upper <- theta_hat + z*S
return(list(
  ci_lower = ci_lower,
  ci_upper = ci_upper
))
}

```

```

get_bonferroni_ci <- function(theta_hat,
                               S,
                               alpha){
  K <- length(theta_hat)
  z = qnorm(1-(alpha/K)/2)
  ci_lower <- theta_hat - z*S
  ci_upper <- theta_hat + z*S
  return(list(
    ci_lower = ci_lower,
    ci_upper = ci_upper
  ))
}

```

```

get_parametric_ci <- function(B,
                               theta_hat,
                               S,
                               alpha) {
  K <- length(theta_hat)

```

```

# step 1a =====
thetahat_star <- sapply(seq_len(K), function(i) {
  rnorm(B, mean = theta_hat[i], sd = S[i])
})
colnames(thetahat_star) <- paste0("thetahat_star",
                                sprintf("%02d", 1:K))
sorted_thetahat_star <- t(apply(thetahat_star, 1, sort))
colnames(sorted_thetahat_star) <- paste0("sorted_thetahat_star",
                                sprintf("%02d", 1:K))

# step 1b =====
variance_vector <- S^2
minuend <- thetahat_star^2 + rep(
  variance_vector, each = nrow(thetahat_star))
sigma_hat_star <- sqrt(
  t(apply(minuend, 1, sort)) - sorted_thetahat_star^2)

# step 1c =====
sorted_theta_hat <- sort(theta_hat)
t_star <- apply(
  abs(
    (
      sorted_thetahat_star - rep(
        sorted_theta_hat,
        each = nrow(sorted_thetahat_star)
      )
    )/sigma_hat_star
  ),
  1,
  max)

# step 2 =====

```

```

t_hat <- quantile(t_star, probs = 1 - alpha)

# step 3 =====

sigma_hat <- sqrt(
  sort(theta_hat^2 + variance_vector) - sorted_theta_hat^2)

# step 6 =====

ci_lower <- sorted_theta_hat - t_hat*sigma_hat
ci_upper <- sorted_theta_hat + t_hat*sigma_hat
return(list(
  ci_lower = ci_lower,
  ci_upper = ci_upper
))
}

```

```

get_nonrankbased_ci <- function(B,
                                theta_hat,
                                alpha,
                                varcovar_matrix) {
  K <- length(theta_hat)

  # step 1a =====

  generate_data <- function(){MASS::mvrnorm(n = 1,
                                             mu = theta_hat,
                                             Sigma = varcovar_matrix)}

  thetahat_star <- t(replicate(B, generate_data()))

  # step 1b =====

  t_star <- apply(thetahat_star,
                  1,
                  function(x) max(abs((x - theta_hat) / sqrt(
                    diag(varcovar_matrix))))))

  # step 2 =====

```

```

t_hat <- quantile(t_star, probs = 1 - alpha)

# step 3 =====
ci_lower <- theta_hat - t_hat*sqrt(diag(varcovar_matrix))
ci_upper <- theta_hat + t_hat*sqrt(diag(varcovar_matrix))

return(list(
  ci_lower = ci_lower,
  ci_upper = ci_upper
))
}

```

## Codes for algorithm 2

```

source("../R/compute_ci.R")
library("doRNG")

get_ranks <- function(k, tuple_list){
  Lambda_lk <- which(
    tuple_list[,2]<=tuple_list[k,1])
  Lambda_lk <- Lambda_lk[Lambda_lk != k]
  Lambda_Ok <- which(
    tuple_list[,2]>tuple_list[k,1] & tuple_list[k,2] > tuple_list[,1])
  Lambda_Ok <- Lambda_Ok[Lambda_Ok != k]
  ranks <- seq(
    length(unique(Lambda_lk)) + 1,
    length(unique(Lambda_lk)) + length(unique(Lambda_Ok)) + 1,
    1
  )
  return(list(
    ranks = ranks,

```

```

    Lambda_Ok = Lambda_Ok
  ))
}

get_t1 <- function(v) mean(v)

get_t2 <- function(v) prod(v)^(1/length(v))

get_t3 <- function(v) {
  1 - ((length(v)+sum(v))/(length(v)^2))
}

get_coverage <- function(ci_lower,
                          ci_upper,
                          true_theta) {
  return(all(ci_lower<=true_theta) & all(true_theta<=ci_upper))
}

algo2_nonrankbased <- function(
  true_theta,
  K,
  reps = 5, # step 4
  B=100,
  alpha= 0.10,
  varcovar_matrix){
  foreach(iter = 1:reps,
    .combine = rbind,
    .packages = c("foreach", "arrow", "MASS"),
    .export = c("get_nonrankbased_ci", "get_independent_ci",

```

```

        "get_bonferroni_ci", "get_ranks", "get_coverage",
        "get_t1", "get_t2", "get_t3")
) %dorning% {

# step 1 =====
theta_hat <- mvrnorm(n = 1,
                    mu = true_theta,
                    Sigma = varcovar_matrix)

# step 2 =====
S <- sqrt(diag(varcovar_matrix))

ci_methods <- list(
  nonrankbased = function() get_nonrankbased_ci(B, theta_hat, alpha,
                                                varcovar_matrix),
  independent = function() get_independent_ci(theta_hat, S, alpha),
  bonferroni = function() get_bonferroni_ci(theta_hat, S, alpha)
)

ci_results <- lapply(ci_methods, function(f) f())

coverages <- lapply(ci_results, function(res) {
  get_coverage(
    ci_lower = res$ci_lower,
    ci_upper = res$ci_upper,
    true_theta = true_theta
  )
})

```

```

process_ci_result <- function(result, K) {
  tuple_list <- t(apply(
    data.frame(
      ci_lower = result$ci_lower,
      ci_upper = result$ci_upper
    ),
    1,
    function(row) as.numeric(row)
  ))

  rank_range_length <- sapply(1:K, function(x)
    length(get_ranks(x, tuple_list)$ranks)
  )

  list(
    t1 = get_t1(rank_range_length),
    t2 = get_t2(rank_range_length),
    t3 = get_t3(rank_range_length)
  )
}

processed <- lapply(ci_results, process_ci_result, K = K)

data.frame(
  t1_nonrankbased = processed$nonrankbased$t1,
  t2_nonrankbased = processed$nonrankbased$t2,
  t3_nonrankbased = processed$nonrankbased$t3,
  coverage_nonrankbased = coverages$nonrankbased,
  t1_independent = processed$independent$t1,

```



```

t2_independent = processed$independent$t2,
t3_independent = processed$independent$t3,
coverage_independent = coverages$independent,
t1_bonferroni = processed$bonferroni$t1,
t2_bonferroni = processed$bonferroni$t2,
t3_bonferroni = processed$bonferroni$t3,
coverage_bonferroni = coverages$bonferroni
)
}
}

```

```

algo2_parametric <- function(
  true_theta,
  K,
  reps = 5, # step 4
  B=100,
  alpha= 0.10,
  S){
  foreach(iter = 1:reps,
    .combine = rbind,
    .packages = c("foreach", "arrow", "MASS"),
    .export = c("get_parametric_ci", "get_ranks", "get_coverage",
      "get_t1", "get_t2", "get_t3")
  ) %dornrg% {

    # step 1 =====
    theta_hat <- rnorm(
      n = K,

```

```

    mean = true_theta,
    sd   = S
)

# step 2 =====
result <- get_parametric_ci(B,
                           theta_hat,
                           S,
                           alpha)

# step 3 =====
sorted_true_theta <- sort(true_theta)
coverage <- get_coverage(ci_lower = result$ci_lower,
                        ci_upper = result$ci_upper,
                        true_theta = sorted_true_theta)

tuple_list <- t(apply(
  data.frame(ci_lower = result$ci_lower,
            ci_upper = result$ci_upper), 1, function(row) as.numeric(row)))
rank_range_length <- sapply(1:K, function(x) length(
  get_ranks(x, tuple_list)$ranks))
t1 <- get_t1(rank_range_length)
t2 <- get_t2(rank_range_length)
t3 <- get_t3(rank_range_length)

data.frame(
  t1_parametric = t1,
  t2_parametric = t2,
  t3_parametric = t3,

```

```

        coverage_parametric = coverage
    )
}
}

```

## Codes for simulation

```

#3:37PM
source("../R/implement_algo2.R")

mean <- 23.8
df <- readRDS("../data/mean_travel_time_ranking_2011.rds")
cl=parallel::makeCluster(15)
registerDoParallel(cl)

sds <- c(2, 3.6, 6)
Ks <- c(51, 40, 30, 20, 10, 5)
corrs <- c(0.1,0.5,0.9)
alphas <- c(0.1)#c(0.05, 0.1, 0.15, 0.2)

for (sd in sds) {
  for (K in Ks) {
    set.seed(123974)
    true_theta <- rnorm(K, mean, sd)
    true_sds <- df$S[1:K]

    for (alpha in alphas) {

```

```

tic("Running parametric...")
coverage_parametric_df <- algo2_parametric(true_theta,
                                           K,
                                           reps = 5000,
                                           B=500,
                                           alpha= alpha,
                                           S=true_sds)

toc()

saveRDS(coverage_parametric_df, paste0("output/coverage_parametric_",
                                       K, "_", sd, "_", alpha, ".rds"))

for (corr in corrs) {
  corr_matrix <- (1 - corr) * diag(K) + corr * matrix(1, K, K)
  variance_vector <- true_sds^2
  delta <- diag(variance_vector)
  varcovar_matrix <- delta^(1/2) %*% corr_matrix %*% delta^(1/2)

  tic("Running nonrankbased...")
  coverage_output_df <- algo2_nonrankbased(
    true_theta,
    K,
    reps = 5000,
    B = 500,
    alpha=alpha,
    varcovar_matrix = varcovar_matrix)
  toc()

  saveRDS(coverage_output_df, paste0("output/coverage_probability_",
                                       K, "_", sd, "_", corr, "_",
                                       alpha, ".rds"))
}

```

```

    }
  }
}
}

```

```
stopCluster(cl)
```

```
param_grid <- expand.grid(K = Ks, sd = sds, corr = corrs, alpha = alphas)
```

```
results <- do.call(rbind, lapply(seq_len(nrow(param_grid)), function(i) {
```

```
  K <- param_grid$K[i]
```

```
  sd <- param_grid$sd[i]
```

```
  corr <- param_grid$corr[i]
```

```
  alpha <- param_grid$alpha[i]
```

```
  a <- readRDS(paste0("output/coverage_probability_",
                      K, "_", sd, "_", corr, "_", alpha, ".rds"))
```

```
data.frame(
```

```
  K = K, sd = sd, corr = corr, alpha = alpha,
```

```
  Cov_nonrankbased = mean(a$coverage_nonrankbased),
```

```
  Cov_independent = mean(a$coverage_independent),
```

```
  Cov_bonferroni = mean(a$coverage_bonferroni),
```

```
  T1_nonrankbased = mean(a$t1_nonrankbased),
```

```
  T1_independent = mean(a$t1_independent),
```

```
  T1_bonferroni = mean(a$t1_bonferroni),
```

```
  T2_nonrank = mean(a$t2_nonrankbased),
```

```
  T2_independent = mean(a$t2_independent),
```

```
  T2_bonferroni = mean(a$t2_bonferroni),
```

```

    T3_independent = mean(a$t3_independent),
    T3_nonrankbased = mean(a$t3_nonrankbased),
    T3_bonferroni = mean(a$t3_bonferroni)
  )
}))

param_grid <- expand.grid(K = Ks, sd = sds, alpha = alphas)

results1 <- do.call(rbind, lapply(seq_len(nrow(param_grid)), function(i) {
  K <- param_grid$K[i]
  sd <- param_grid$sd[i]
  alpha <- param_grid$alpha[i]

  a <- readRDS(paste0("output/coverage_parametric_",
                      K, "_", sd, "_", alpha, ".rds"))

  data.frame(
    K = K,
    sd = sd,
    alpha = alpha,
    Cov_parametric = mean(a$coverage_parametric),
    T1_parametric = mean(a$t1_parametric),
    T2_parametric = mean(a$t2_parametric),
    T3_parametric = mean(a$t3_parametric)
  )
}))

save(results, results1, file = "simulation_results.RData")

```