Supplementary document: Examples of the effect size and MCID calculation

Contents

---

1 Introduction

This is a supplementary file for the article titled "Alternatives to the P value: connotations of significance" published in the *Korean Journal of Anesthesiology* (DOI: https://doi.org/10.4097/kja.23630). This document provides examples of effect size (based on Table 1), and minimal clinically important difference (MCID) calculations. Please note that all statistical analysis procedures presented in this document have been simplified for illustrative purposes, and the sample data used were artificially generated. When conducting an actual statistical analysis, it is important to consider the underlying assumptions required for each statistical method. Additionally, please note that the primary focus of this document is to describe how to calculate effect sizes and MCID.
This supplementary material was prepared and generated using R (version 4.3.2) and knitr (version 1.45), the final version was complied at 20:15 KST, 13 Apr 2024.

2 Common process

2.1 Working directory setting

For example, setwd(r"(C:\temp)")

```
setwd(r"(insert your working directory here)")
```

2.2 Sample data import

First, copy the supplementary data "sample_data.csv" into your working folder. The data can then imported into R using following code.

```
#Read CSV file
mydata = read.csv("sample_data.csv", header = TRUE, fileEncoding = "UTF-8-BOM", fill = TRUE)
```

2.3 Required package

The following libraries are required to calculate the examples.

- rcompanion (version 2.4.34, for effect size calculation)
- lsr (version 0.5.2, for effect size calculation)
- DescTools (version 0.99.50, for effect size calculation)
- apaTables (version 2.0.8, for effect size calculation)
- effectsize (version 0.8.6, for effect size calculation)
- afex (version 1.3-0, for type III SS calculation in ANOVA)
- psych (version 2.3.6 for Spearman's correlation)
- knitr (versin 1.43 for table)

- boot (version 1.3-28.1 for bootstrapped confidence interval estimation)

If any of the packages listed above are missing, the required packages can be installed on your computer using the following codes:

```
# Package install.
list.of.packages <- c("rcompanion",
                      "lsr",
                      "DescTools",
                      "apaTables",
                      "effectsize",
                      "afex",
                      "psych",
                      "car",
                      "knitr",
                      "boot")
req.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
if(length(req.packages)) install.packages(req.packages)
```

After installation, the required packges will be loaded as required.

3 Effect size calculation

3.1 Student's *t*-test

- From the sample data, extract the data for a *t*-test.

```
# Extract data
data.ttest <- subset(mydata, select = c(t_test.group,   t_test.value))

# Remove rows containing empty cell
data.ttest <- data.ttest[!apply(data.ttest == "",1,all), ]


summary(data.ttest)
```

```
##  t_test.group        t_test.value
##  Length:218         Min.   :35.00
##  Class :character   1st Qu.:56.00
##  Mode  :character   Median :63.00
##                     Mean   :64.28
##                     3rd Qu.:70.75
##                     Max.   :95.00
```

- Data structure (first 6 lines):

```
head(data.ttest)
```

```
##   t_test.group t_test.value
## 1            A           70
## 2            B           48
## 3            A           77
## 4            A           64
## 5            A           80
## 6            B           57
```

- Performing the *t*-test:

```r
res.t.test <- t.test(t_test.value ~ t_test.group,
       data = data.ttest)
res.t.test
```

```
##
##  Welch Two Sample t-test
##
## data:  t_test.value by t_test.group
## t = 8.9857, df = 191.3, p-value = 2.467e-16
## alternative hypothesis: true difference in means between group A and group B is not equal to 0
## 95 percent confidence interval:
##   9.158235 14.309655
## sample estimates:
## mean in group A mean in group B
##        70.14679        58.41284
```

3.1.1 Effect size calculation: Cohen's *d*

- The "lsr" package provides the cohensD function, which supports both numeric variables and formulas as input.

```r
lsr::cohensD(t_test.value ~ t_test.group, #formula with 'data' argument
             data = data.ttest)
```

```
## [1] 1.217175
```

- The cohenD function from the "DescTools" package in R can be used to calculate the effect size of Cohen's *d*, and corresponding confidence intervals for the effect size.

```r
groupa <- subset(data.ttest, subset = (t_test.group == "A"))
groupb <- subset(data.ttest, subset = (t_test.group == "B"))

DescTools::CohenD(groupa$t_test.value, groupb$t_test.value,
                  pooled = TRUE,
                  conf.level = 0.95)
```

```
##         d    lwr.ci    upr.ci
## 1.2171754 0.9267646 1.5051875
## attr(,"magnitude")
## [1] "large"
```

- Manual calculation of Cohen's *d* and 95% CI based on t-test results:

```r
# Mean, SD and size of each group
mean1 <- mean(data.ttest$t_test.value[data.ttest$t_test.group == "A"])
sd1 <- sd(data.ttest$t_test.value[data.ttest$t_test.group == "A"])
n1 <- length(data.ttest$t_test.value[data.ttest$t_test.group == "A"])

mean2 <- mean(data.ttest$t_test.value[data.ttest$t_test.group == "B"])
sd2 <- sd(data.ttest$t_test.value[data.ttest$t_test.group == "B"])
n2 <- length(data.ttest$t_test.value[data.ttest$t_test.group == "B"])

# Mean difference and pooled SD
mean_diff <- mean1 - mean2
pooled_sd <- sqrt(((n1 - 1) * sd1^2 + (n2 - 1) * sd2^2) / (n1 + n2 - 2))

# Calculate Cohen's d
```

```r
cohen_d <- mean_diff / pooled_sd

# Calculate standard error
se_d <- sqrt((n1 + n2) / (n1 * n2) + (cohen_d^2) / (2 * (n1 + n2)))

# Degrees of freedom
df <- n1 + n2 - 2

# Calculate 95% CI
ci_error <- qt(0.975, df) * se_d
ci_lower <- cohen_d - ci_error
ci_upper <- cohen_d + ci_error

# Return the results
list(cohen_d = cohen_d, ci_lower = ci_lower, ci_upper = ci_upper)
```

```
## $cohen_d
## [1] 1.217175
##
## $ci_lower
## [1] 0.9265163
##
## $ci_upper
## [1] 1.507834
```

3.1.2 BESD calculation

- Required package: "effectsize"

```r
effectsize::d_to_r(1.217175)
```

```
## [1] 0.5198795
```

- Manual calculation:

```r
cohen_d / (sqrt(cohen_d^2 + 4))
```

```
## [1] 0.5198797
```

3.2 Mann-Whitney rank sum test (Wilcoxon rank sum test, Mann-Whitney U test)

- Data preparation:

```r
data.mwrs <- na.omit(subset(mydata, select= c(mw.group, mw.value)))

# remove rows containing empty cell
data.mwrs <- data.mwrs[!apply(data.mwrs == "",1,all), ]

summary(data.mwrs)
```

```
##    mw.group           mw.value
##  Length:43          Min.   :0.100
##  Class :character   1st Qu.:0.800
##  Mode  :character   Median :1.100
##                     Mean   :1.077
##                     3rd Qu.:1.300
```

```
##                     Max.   :2.100
```

- Data structure (first 6 lines):

```
head(data.mwrs)
```

```
##   mw.group mw.value
## 1        B      1.1
## 2        B      1.3
## 3        A      0.8
## 4        A      0.8
## 5        A      0.1
## 6        A      1.1
```

- U test:

```
rs.mwrs <- wilcox.test(mw.value ~ mw.group,
            data = data.mwrs)

rs.mwrs
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  mw.value by mw.group
## W = 47, p-value = 7.318e-06
## alternative hypothesis: true location shift is not equal to 0
```

3.2.1 Effect size r

- Required package: "rcompanion"

```
rcompanion::wilcoxonR(x=data.mwrs$mw.value, g=data.mwrs$mw.group, ci = TRUE)
```

```
##        r lower.ci upper.ci
## 1 -0.686   -0.804   -0.514
```

- Manual calculation: The following calculation includes tied-rank correction and bootstrapping for confidence interval estimation. Detailed explanations of these conditions are not provided here. To make a robust bootstrap process, defining a custom function is crucial.

```
# calculation of r

  # Variables for group sizes and total size
  n1 <- sum(data.mwrs$mw.group == "A")
  n2 <- sum(data.mwrs$mw.group == "B")
  N <- n1 + n2

  # U statistic
  U <- rs.mwrs$statistic

  # Calculate mean and variance of U considering ties
  mu_U <- n1 * n2 / 2
  ranks <- rank(data.mwrs$mw.value)
  ties_sum <- sum((table(ranks)^3 - table(ranks)))
  sigma_U_corrected <- sqrt(n1 * n2 * (N + 1 - ties_sum / (N * (N - 1))) / 12)

  # Calculate the corrected Z value and effect size r
  Z <- (U - mu_U) / sigma_U_corrected
```

```r
  r <- Z / sqrt(N)

  # Calculated r
  print(r)
```

```
##          W
## -0.6856989
```

```r
# Bootstrapped r and 95%CI
  library(boot)

  # Custom function: calculate effect size r for bootstrap CI estimation
  calculate_effect_size <- function(data, indices) {

      # Sampling data for bootstrap
      sampled_data <- data[indices, ]

      # Separating groups
      group_A <- sampled_data$mw.value[sampled_data$mw.group == 'A']
      group_B <- sampled_data$mw.value[sampled_data$mw.group == 'B']

      # Executing Mann-Whitney U test
      test_result <- wilcox.test(group_A, group_B)

      # U statistic
      U <- test_result$statistic

      # Sizes of groups
      n1 <- length(group_A)
      n2 <- length(group_B)

      # Total data size
      N <- n1 + n2

      # Calculating mean and standard deviation
      mu_U <- n1 * n2 / 2
      ranks <- rank(data.mwrs$mw.value)
      ties_sum <- sum((table(ranks)^3 - table(ranks)))
      sigma_U_corrected <- sqrt(n1 * n2 * (N + 1 - ties_sum / (N * (N - 1))) / 12)

      # Calculating effect size r
      Z <- (U - mu_U) / sigma_U_corrected
      r <- Z / sqrt(N)

      return(r)
    }

  # Executing bootstrap
  set.seed(123) # Setting seed for reproducible results
  boot_result <- boot(data.mwrs, statistic=calculate_effect_size, R=1000)

  # Calculating 95% confidence intervals
  boot_ci <- boot.ci(boot_result, type=c("norm", "basic", "perc", "bca"))
```

```r
  #Estimated result
  list(effect_size_r = mean(boot_result$t), CI_low = boot_ci$bca[4], CI_high = boot_ci$bca[5])
```

```
## $effect_size_r
## [1] -0.6721086
##
## $CI_low
## [1] -0.7954948
##
## $CI_high
## [1] -0.5061475
```

3.2.2 VDA

- Required package: "rcompanion"

```r
rcompanion::vda(mw.value ~ mw.group, data = data.mwrs, ci = TRUE)
```

```
##     VDA lower.ci upper.ci
## 1 0.102   0.0304    0.197
```

- Manual calculation with bootstrap confidence interval estimation.

```r
# Mann-Whitney U test and VDA calculation
calculate_vda <- function(data) {
  test_result <- wilcox.test(mw.value ~ mw.group, data = data)
  U <- test_result$statistic
  n1 <- sum(data$mw.group == "A")
  n2 <- sum(data$mw.group == "B")
  VDA <- U / (n1 * n2)
  return(VDA)
}

vda_original <- calculate_vda(data.mwrs)
cat("Original VDA:", vda_original, "\n")
```

```
## Original VDA: 0.1017316
```

```r
# 95% CI estimation using bootstrap
library(boot)

bootstrap_vda <- function(data, n_bootstrap=1000) {
  bootstrap_values <- replicate(n_bootstrap, {
    sample_data <- data[sample(nrow(data), replace = TRUE), ]
    calculate_vda(sample_data)
  })
  ci <- quantile(bootstrap_values, c(0.025, 0.975))
  mean_vda <- mean(bootstrap_values)
  return(list(mean_vda = mean_vda, ci = ci))
}

rs.vda <- bootstrap_vda(data.mwrs)

cat("Bootstrap Mean VDA:", rs.vda$mean_vda, "\n")
```

```
## Bootstrap Mean VDA: 0.1007622
```

```r
cat("95% CI for VDA:", rs.vda$ci, "\n")
```

```
## 95% CI for VDA: 0.03289474 0.193744
```

3.2.3 Cliff's *delta*

- Required package: "rcompanion"

```r
rcompanion::cliffDelta(mw.value ~ mw.group, data = data.mwrs, ci = TRUE)
```

```
##   Cliff.delta lower.ci upper.ci
## 1      -0.797   -0.948   -0.602
```

-Manual calculation

```r
# Cliff's Delta calculation

  # Extract values for each group
  group1_values <- data.mwrs$mw.value[data.mwrs$mw.group == "A"]
  group2_values <- data.mwrs$mw.value[data.mwrs$mw.group == "B"]

  # Perform all possible pairwise comparisons between the two groups

    # Number of cases where values in group1 are greater than those in group2, N_gt
    N_gt <- sum(outer(group1_values, group2_values, FUN = function(x, y) x > y))
    # Number of cases where values in group1 are less than those in group2, N_lt
    N_lt <- sum(outer(group1_values, group2_values, FUN = function(x, y) x < y))

  # Sample sizes for each group
  n1 <- length(group1_values)
  n2 <- length(group2_values)

  # Calculate Cliff's Delta
  original_delta <- (N_gt - N_lt) / (n1 * n2)

# Bootstrap procedure for CI estimation
library(boot)

  # Define a function to calculate the mean and 95% CI of Cliff's Delta using bootstrap procedure
  bootstrap_cliffs_delta <- function(data, n_bootstrap=1000) {
      deltas <- numeric(n_bootstrap)  # Initialize vector to store Delta values calculated by bootstrap

        for (i in 1:n_bootstrap) {
          # Generate samples from each group using random sampling with replacement
          sample_indices1 <- sample(length(group1_values), replace = TRUE)
          sample_indices2 <- sample(length(group2_values), replace = TRUE)

          # Calculate N_gt and N_lt for the bootstrap samples
          N_gt_b <- sum(outer(group1_values[sample_indices1], group2_values[sample_indices2],
                              FUN = function(x, y) x > y))
          N_lt_b <- sum(outer(group1_values[sample_indices1], group2_values[sample_indices2],
                              FUN = function(x, y) x < y))

          # Calculate Cliff's Delta for each bootstrap sample
          delta_b <- (N_gt_b - N_lt_b) / (n1 * n2)
          deltas[i] <- delta_b
```

```
    }

  # Calculate the mean and 95% CI of bootstrap Delta values
  mean_delta <- mean(deltas)
  ci <- quantile(deltas, probs = c(0.025, 0.975))

  return(list(mean_delta = mean_delta, ci = ci))
  }

# Calculate bootstrap results
bootstrap_results <- bootstrap_cliffs_delta(data.mwrs)
cat("Original Cliff's Delta:", original_delta, "\n")
```

```
## Original Cliff's Delta: -0.7965368
```

```
cat("Bootstrap Mean Cliff's Delta:", bootstrap_results$mean_delta, "\n")
```

```
## Bootstrap Mean Cliff's Delta: -0.7969091
```

```
cat("95% CI for Cliff's Delta:", bootstrap_results$ci, "\n")
```

```
## 95% CI for Cliff's Delta: -0.9393939 -0.5930195
```

3.3 Paired *t*-test

- Data preparation:

```
pairedt <- na.omit(subset(mydata, select = c(pair_t.pre,    pair_t.post)))

# remove rows containing empty cells
pairedt <- pairedt[!apply(pairedt == "",1,all), ]

summary(pairedt)
```

```
##    pair_t.pre      pair_t.post
##  Min.   : 15.00   Min.   :10.00
##  1st Qu.: 42.25   1st Qu.:19.75
##  Median : 55.00   Median :25.00
##  Mean   : 55.14   Mean   :25.32
##  3rd Qu.: 68.00   3rd Qu.:30.00
##  Max.   :123.00   Max.   :52.00
```

- Data structure (first 6 lines):

```
head(pairedt)
```

```
##   pair_t.pre pair_t.post
## 1         16          10
## 2         60          27
## 3         58          26
## 4         70          31
## 5         38          18
## 6         72          32
```

- Paired *t*-test:

```
t.test(pairedt$pair_t.post, pairedt$pair_t.pre, paired = TRUE)
```

```
##
##  Paired t-test
##
## data:  pairedt$pair_t.post and pairedt$pair_t.pre
## t = -23.263, df = 99, p-value < 2.2e-16
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  -32.36344 -27.27656
## sample estimates:
## mean difference
##          -29.82
```

3.3.1 Cohen's *d*

- Required package: "lsr"

```
lsr::cohensD(pairedt$pair_t.post, pairedt$pair_t.pre, method = "paired")
```

```
## [1] 2.326348
```

DescTools::CohenD does not support paired *t*-tests. Therefore, if you simply input the arguments into CohenD, the output will correspond to a *t*-test rather than a paired *t*-test.

```
DescTools::CohenD(pairedt$pair_t.post, pairedt$pair_t.pre, pooled = TRUE, conf.level = 0.95)
```

```
##         d    lwr.ci    upr.ci
## -1.928044 -2.262270 -1.590428
## attr(,"magnitude")
## [1] "large"
```

Let's do the modified calculation:

```
(mean(pairedt$pair_t.pre)-mean(pairedt$pair_t.post))/sd(pairedt$pair_t.pre-pairedt$pair_t.post)
```

```
## [1] 2.326348
```

If you want to calculate the CI of Cohen's *d*, use "DescTools:CohenD" with some modifications:

```
diff = pairedt$pair_t.pre - pairedt$pair_t.post #make a new variable of pre-post difference

DescTools::CohenD(diff, conf.level = 0.95) # Effect size of one-sample t-test
```

```
##        d   lwr.ci   upr.ci
## 2.326348 1.717275 2.924908
## attr(,"magnitude")
## [1] "large"
```

Also, you can calculate manually. Note that there are several different but similar formula for calculating the standard error of Cohen's *d*.

```
# Cohen's d calculation
mean_diff <- mean(diff) # Sample mean
sd_diff <- sd(diff) # Sample standard deviation
n <- length(diff) # Sample size
cohen_d <- mean_diff / sd_diff # Cohen's d

# 95%CI based on normal distribution

  # Calculate the standard error of Cohen's d
  se_d <- sqrt((n / (n - 1)) * (2 / n + (cohen_d^2) / (n - 3)))
```

```r
  # Calculate the z-score (for 95% confidence interval)
  z <- qnorm(1 - 0.025)

  # Calculate the 95% confidence interval for Cohen's d
  ci_lower <- cohen_d - z * se_d
  ci_upper <- cohen_d + z * se_d

  # Print the results
  print(paste("Cohen's d:", cohen_d))
```

```
## [1] "Cohen's d: 2.32634813492329"
```

```r
  print(paste("95% CI for Cohen's d:", ci_lower, ci_upper))
```

```
## [1] "95% CI for Cohen's d: 1.78404227964194 2.86865399020464"
```

3.4 Wilcoxon signed rank test (Wilcoxon Z test)

- Data preparation:

```r
wcsr <- na.omit(subset(mydata, select = c(case.no, wc.pre, wc.post)))
```

- Data structure (first 6 lines):

```r
head(wcsr)
```

```
##   case.no wc.pre wc.post
## 1       1     20      18
## 2       2     60      33
## 3       3     60      34
## 4       4     73      43
## 5       5     41      28
## 6       6     72      40
```

- Wilcoxon z test:

```r
wilcox.test(wcsr$wc.pre, wcsr$wc.post,
            paired = TRUE,
            conf.int = TRUE,
            conf.level = 0.95)
```

```
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  wcsr$wc.pre and wcsr$wc.post
## V = 4848, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
## 95 percent confidence interval:
##  22.49998 27.00007
## sample estimates:
## (pseudo)median
##       24.50002
```

Or long type transformation:

```r
wcsr.long <- tidyr::gather(wcsr,
                      time,
                      observed,
```

```
                      wc.pre:wc.post,
                      factor_key = TRUE
                      )
```

With long-type data, the Wilcoxon signed rank test is used:

```
wilcox.test(observed ~ time,
             data = wcsr.long,
             paired = TRUE,
            conf.int = TRUE,
            conf.level = 0.95)
```

```
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  observed by time
## V = 4848, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
## 95 percent confidence interval:
##  22.49998 27.00007
## sample estimates:
## (pseudo)median
##       24.50002
```

3.4.1 Matched-pairs rank biserial correlation coefficient

- Required package: "rcompanion"

```
rcompanion::wilcoxonPairedRC(x = wcsr.long$observed,
                 g = wcsr.long$time,
                 #type = "bca",
                 ci = TRUE,
                 verbose = TRUE)
```

```
##
## Levels: wc.pre wc.post
## zero.method: Wilcoxon
## n kept = 98
## Ranks plus = 3
## Ranks minus = 4848
## T value = 3
```

```
##      rc lower.ci upper.ci
## 1 0.999    0.994        1
```

- Manual calculation. Note that Z-transformation is used for 95%CI calculation.
  (Reference: King BM, Rosopa PJ, Minium EW. Statistical Reasoning in the Behavioral Sciences, 6th ed. 2011.John Wiley & Sons. PP392-4)

```
# Calculate differences
wcsr$diffs <- wcsr$wc.pre - wcsr$wc.post

# Exclude the cases of diffs = 0
wcsr_filtered <- wcsr[wcsr$diffs !=0,]

# Rank the absolute differences
wcsr_filtered$rank <- rank(abs(wcsr_filtered$diffs))
```

```r
# Calculate sums of positive and negative ranks
R_plus <- sum(wcsr_filtered$rank[wcsr_filtered$diffs > 0])
R_minus <- sum(wcsr_filtered$rank[wcsr_filtered$diffs < 0])

# N (Total number of pairs)
N <- nrow(wcsr_filtered)

# T (smaller one)
T <- min(R_plus, R_minus)

rc <- abs(4*T - N*(N+1)/2) / (N*(N+1)/2)


### Function to calculate rc from the original data
calc_rc <- function(data, indices) {
  # Sample the bootstrap data
  sample_data <- data[indices,]
  sample_data$diff <- sample_data$wc.pre - sample_data$wc.post
  sample_data <- sample_data[sample_data$diff != 0,]
  sample_data$rank <- rank(abs(sample_data$diff))
  R_plus <- sum(sample_data$rank[sample_data$diff > 0])
  R_minus <- sum(sample_data$rank[sample_data$diff < 0])
  N <- nrow(sample_data)
  T <- min(R_plus, R_minus)
  rc <- abs(4*T - N*(N+1)/2) / (N*(N+1)/2)
  return(rc)
}

# Calculate confidence intervals using bootstrapping
boot_data <- boot::boot(wcsr, calc_rc, R=2000)

# Calculate 95% confidence interval
boot_ci <- boot::boot.ci(boot_data, type=c("perc"))

# Print the results
print(paste("Matched-pairs rank biserial correlation coefficient: ",rc))
```

```
## [1] "Matched-pairs rank biserial correlation coefficient:  0.997526283240569"
```

```r
print(paste("Bootstrap coefficient: ", mean(boot_data$t)))
```

```
## [1] "Bootstrap coefficient:  0.996754113945055"
```

```r
print(boot_ci)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot::boot.ci(boot.out = boot_data, type = c("perc"))
##
## Intervals :
## Level     Percentile
## 95%   ( 0.9874,  1.0000 )
## Calculations and Intervals on Original Scale
```

3.4.2 *r*

- Required package: "rcompanion"

```
rcompanion::wilcoxonPairedR(x = wcsr.long$observed,
                  g = wcsr.long$time,
                  ci = TRUE)
```

```
##       r lower.ci upper.ci
## 1 0.868    0.863    0.869
```

- Manual Calculation: Use following formula to calculate Z statistic for the Wilcoxon signed-ranks test.

$$z = \frac{R_+ - 0.25n(n+1)}{\sqrt{n(n+1)(2n+1)/24}}$$

Where, R+: the sum of the ranks with a positive sign (R_plus), n: total number of pairs (N). We can use R_minus instead of R_plus.

```
# Define the function to calculate the Z value and effect size r
calculate_z_r <- function(N, R_plus) {
  # Calculate Z value using the formula
  Z <- (R_plus - 0.25 * N * (N + 1)) / (sqrt(N * (N + 1) * (2 * N + 1) / 24))

  # Calculate the effect size r
  r <- Z / sqrt(N)

  return(list(Z=Z, r=r))
}

# Define the function for bootstrap to calculate the 95% CI for the effect size r
bootstrap_ci <- function(diffs, n_bootstrap=10000, ci=95) {
  bootstrap_r_values <- replicate(n_bootstrap, {
    sample_diffs <- sample(diffs, length(diffs), replace=TRUE)
    sum_ranks <- sum(rank(abs(sample_diffs))[sample_diffs > 0])
    calculate_z_r(length(sample_diffs), sum_ranks)$r
  })

  lower_bound <- quantile(bootstrap_r_values, (100 - ci) / 2 / 100)
  upper_bound <- quantile(bootstrap_r_values, 1 - (100 - ci) / 2 / 100)

  return(c(lower=lower_bound, upper=upper_bound))
}

# Assuming that `diffs` is wcsr_filtered$diffs and you have calculated R_plus and N before

# Calculate the Z value and effect size r
results <- calculate_z_r(N, R_plus)

# Calculate the 95% CI for the effect size r using bootstrap
# Replace `wcsr_filtered$diffs` with the actual diffs
bootstrap_results <- bootstrap_ci(wcsr_filtered$diffs)

# Display results
print(results)
```

```
## $Z
## [1] 8.584315
##
## $r
## [1] 0.8671468
```

```
print(bootstrap_results)
```

```
##  lower.2.5% upper.97.5%
##   0.8628513   0.8682207
```

3.5 One-way ANOVA

- Data preparation:

```
anova.data <- na.omit(subset(mydata, select = c(case.no, anova.a, anova.b, anova.c)))
summary(anova.data)
```

```
##     case.no      anova.a         anova.b         anova.c
##  Min.   : 1   Min.   :230.0   Min.   :362.0   Min.   : 485.0
##  1st Qu.:15   1st Qu.:443.0   1st Qu.:497.0   1st Qu.: 648.0
##  Median :29   Median :524.0   Median :548.0   Median : 709.0
##  Mean   :29   Mean   :543.6   Mean   :560.8   Mean   : 724.5
##  3rd Qu.:43   3rd Qu.:653.0   3rd Qu.:630.0   3rd Qu.: 808.0
##  Max.   :57   Max.   :913.0   Max.   :795.0   Max.   :1007.0
```

- Data structure (first 6 lines):

```
head(anova.data)
```

```
##   case.no anova.a anova.b anova.c
## 1       1     565     574     741
## 2       2     477     519     674
## 3       3     712     668     853
## 4       4     325     422     557
## 5       5     517     544     704
## 6       6     552     566     731
```

- Long type data transformation:

```
anova.long <- tidyr::gather(anova.data,
                      group,
                      observed,
                      anova.a:anova.c,
                      factor_key = TRUE
                      )
summary(anova.long)
```

```
##     case.no       group       observed
##  Min.   : 1   anova.a:57   Min.   : 230.0
##  1st Qu.:15   anova.b:57   1st Qu.: 516.0
##  Median :29   anova.c:57   Median : 607.0
##  Mean   :29                Mean   : 609.6
##  3rd Qu.:43                3rd Qu.: 707.5
##  Max.   :57                Max.   :1007.0
```

```
head(anova.long)
```

```
##   case.no   group observed
```

```
## 1       1 anova.a      565
## 2       2 anova.a      477
## 3       3 anova.a      712
## 4       4 anova.a      325
## 5       5 anova.a      517
## 6       6 anova.a      552
```

- One-way ANOVA

```
anova.result <- aov(observed ~ group, data = anova.long)
summary(anova.result)
```

```
##              Df  Sum Sq Mean Sq F value   Pr(>F)
## group         2 1136553  568277   34.51 2.78e-13 ***
## Residuals   168 2766386   16467
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.5.1 Eta squared

- Required package: "lsr"

```
lsr::etaSquared(anova.result)
```

```
##          eta.sq eta.sq.part
## group 0.2912044   0.2912044
```

- Required package: "effectsize"

```
effectsize::eta_squared(anova.result,
            ci = 0.95,
            alternative = "two.sided")
```

```
## For one-way between subjects designs, partial eta squared is equivalent
##   to eta squared. Returning eta squared.
```

```
## # Effect Size for ANOVA
##
## Parameter | Eta2 |       95% CI
## ------------------------------
## group     | 0.29 | [0.18, 0.39]
```

- Manual calculation: The process of calculating eta squared is as follows:

1. Calculate the mean for each group and the overall mean for all data.
2. Calculate the sum of squares between groups (SSB). This is the sum of the squared differences between each group mean and the overall mean, multiplied by the number of cases in each group.
3. Calculate the total sum of squares (SST). This is the sum of the squared differences between each data point and the overall mean.
4. Eta squared is the value of SSB divided by SST, and represents the size of the treatment effect.
5. 95%CI calculation: The `effectsize::eta_squared()` function estimates confidence (compatibility) intervals using the noncentrality parameter method, also known as the pivot method. This method involves a complex process that is challenging to calculate manually. Instead of using the pivot method,

bootstrapped confidence intervals are presented here.However, using 95%CI of the pivot method is recommended.

```r
# Calculate group means
group_means <- colMeans(anova.data[, c("anova.a", "anova.b", "anova.c")])

# Calculate overall mean
overall_mean <- mean(c(anova.data$anova.a, anova.data$anova.b, anova.data$anova.c))

# Calculate between-group variance (SSB)
n <- nrow(anova.data) # Number of data points in each group,
                      # Note that this data is balanced number between groups.
SSB <- n * sum((group_means - overall_mean)^2)

# Calculate total variance (SST)
SST <- sum((c(anova.data$anova.a, anova.data$anova.b, anova.data$anova.c) - overall_mean)^2)

# Calculate Eta square
eta_square <- SSB / SST

# Load required packages
library(boot)

# Function to calculate Eta squared
eta_squared_func <- function(data, indices) {
  data_sample <- data[indices, ] # Bootstrap sample
  group_means <- colMeans(data_sample[, c("anova.a", "anova.b", "anova.c")])
  overall_mean <- mean(c(data_sample$anova.a, data_sample$anova.b, data_sample$anova.c))
  n <- nrow(data_sample)# Number of data points in each group
  SSB <- n * sum((group_means - overall_mean)^2)
  SST <- sum((c(data_sample$anova.a, data_sample$anova.b, data_sample$anova.c) - overall_mean)^2)
  eta_square <- SSB / SST
  return(eta_square)
}

# Perform bootstrap
set.seed(123) # Set seed for reproducibility
results <- boot(data = anova.data, statistic = eta_squared_func, R = 2000)

# Calculate 95% confidence interval
ci <- boot.ci(results, type = "bca")

# Print the result
eta_square
```

```
## [1] 0.2912044
```

```r
ci
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, type = "bca")
##
## Intervals :
```

```
## Level       BCa
## 95%   ( 0.2222,  0.3622 )
## Calculations and Intervals on Original Scale
```

3.6 Two-way ANOVA

- Data preparation:

```
tanova <- na.omit(subset(mydata, select = c(case.no, tanova.group, tanova.state, tanova.value)))
colnames(tanova) <- c("id", "group", "state", "value")
summary(tanova)
```

```
##        id           group              state               value
##  Min.   : 1.00   Length:60          Length:60          Min.   :346.0
##  1st Qu.:15.75   Class :character   Class :character   1st Qu.:607.0
##  Median :30.50   Mode  :character   Mode  :character   Median :680.0
##  Mean   :30.50                                         Mean   :692.2
##  3rd Qu.:45.25                                         3rd Qu.:830.5
##  Max.   :60.00                                         Max.   :999.0
```

- Data structure (first 6 lines):

```
head(tanova)
```

```
##   id group  state value
## 1  1 med.a normal   383
## 2  2 med.a normal   447
## 3  3 med.a normal   369
## 4  4 med.a normal   504
## 5  5 med.a normal   447
## 6  6 med.a normal   403
```

- Two-way ANOVA:

```
tanova.result <- aov(value ~ group + state + group*state,
                     data = tanova)
summary(tanova.result)
```

```
##             Df  Sum Sq Mean Sq F value Pr(>F)
## group        1    6222    6222   0.992 0.3236
## state        2 1312993  656496 104.710 <2e-16 ***
## group:state  2   61399   30699   4.896 0.0111 *
## Residuals   54  338561    6270
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.6.1 Eta squared (for the type I SS)

- Required package: "lsr"

```
lsr::etaSquared(tanova.result)
```

```
##                  eta.sq eta.sq.part
## group       0.003619189  0.01804618
## state       0.763734441  0.79500448
## group:state 0.035713969  0.15351183
```

-Required package: "effectsize"

```r
# Eta squared
effectsize::eta_squared(tanova.result,
                        partial = FALSE,
                        ci = 0.95,
                        alternative = "two.sided")
```

```
## # Effect Size for ANOVA (Type I)
##
## Parameter   |     Eta2 |       95% CI
## ------------------------------------
## group       | 3.62e-03 | [0.00, 0.09]
## state       |     0.76 | [0.65, 0.83]
## group:state |     0.04 | [0.00, 0.15]
```

```r
# Partial eta squared
effectsize::eta_squared(tanova.result,
                        partial = TRUE,
                        ci = 0.95,
                        alternative = "two.sided")
```

```
## # Effect Size for ANOVA (Type I)
##
## Parameter   | Eta2 (partial) |       95% CI
## ----------------------------------------
## group       |           0.02 | [0.00, 0.14]
## state       |           0.80 | [0.69, 0.85]
## group:state |           0.15 | [0.01, 0.32]
```

Eta squared from type I sums of squares(SS) reflects the magnitude and relevance of effects of explanatory variables based on the appearing order in the model. Thus the estimated effect size can vary depending on the order of explanatory variables. It is difficult to assess the independent contributions of corresponding variables.

- Manual calculation: Calcuation of 95%CIs for the effect sizes can be performed using a similar process as described in the one-way ANOVA section. However, it is crucial to note that while manual calculation is possible, the pivot method is preferred for estimating CIs due to its accuracy and reliability. Hence, the code for calculating bootstrap CIs will not be provided here and thereafter.

```r
# Extract Type I SS
summary.aov(tanova.result)
```

```
##             Df  Sum Sq Mean Sq F value Pr(>F)
## group        1    6222    6222   0.992 0.3236
## state        2 1312993  656496 104.710 <2e-16 ***
## group:state  2   61399   30699   4.896 0.0111 *
## Residuals   54  338561    6270
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Calculate total SS
total_ss <- sum(summary.aov(tanova.result)[[1]]$"Sum Sq")

# Calculate SS for each factor (based on Type I SS)
group_ss <- summary.aov(tanova.result)[[1]]$"Sum Sq"[1]
state_ss <- summary.aov(tanova.result)[[1]]$"Sum Sq"[2]
interaction_ss <- summary.aov(tanova.result)[[1]]$"Sum Sq"[3]
error_ss <- summary.aov(tanova.result)[[1]]$"Sum Sq"[4]
```

```r
# Calculate Eta squared
eta_squared_group <- group_ss / total_ss
eta_squared_state <- state_ss / total_ss
eta_squared_interaction <- interaction_ss / total_ss

# Partial Eta squared
eta_squared_group_p <- group_ss / (group_ss + error_ss)
eta_squared_state_p <- state_ss / (state_ss + error_ss)
eta_squared_interaction_p <- interaction_ss / (interaction_ss + error_ss)


# Output
cat("Eta squared for 'group':", eta_squared_group, "\n")
```

```
## Eta squared for 'group': 0.003619189
```

```r
cat("Eta squared for 'state':", eta_squared_state, "\n")
```

```
## Eta squared for 'state': 0.7637344
```

```r
cat("Eta squared for 'interaction':", eta_squared_interaction,"\n")
```

```
## Eta squared for 'interaction': 0.03571397
```

```r
cat("Partial Eta squared for 'group':", eta_squared_group_p, "\n")
```

```
## Partial Eta squared for 'group': 0.01804618
```

```r
cat("Partial Eta squared for 'state':", eta_squared_state_p, "\n")
```

```
## Partial Eta squared for 'state': 0.7950045
```

```r
cat("Partial Eta squared for 'interaction':", eta_squared_interaction_p,"\n")
```

```
## Partial Eta squared for 'interaction': 0.1535118
```

3.6.2 Cohen's *f*

Cohen's f is primarily used as an indicator of effect size and is employed in calculating sample sizes for ANOVA. This metric aids in determining the appropriate sample size based on the anticipated effect size during the initial stages of research design. Cohen's f is closely related to conventional eta squared.

- Required package: "effectsize"

```r
effectsize::cohens_f(tanova.result,
         partial = FALSE,
         ci = 0.95,
         alternative = "two.sided")
```

```
## # Effect Size for ANOVA (Type I)
##
## Parameter   | Cohen's f |       95% CI
## -------------------------------------
## group       |      0.06 | [0.00, 0.32]
## state       |      1.80 | [1.36, 2.22]
## group:state |      0.19 | [0.00, 0.43]
```

- Manual calculations: Cohen's f is interchangable with eta square

```r
# Cohen's f of group
sqrt(eta_squared_group/(1-eta_squared_group))
```

```
## [1] 0.06026886
```

```r
# Cohen's f of state
sqrt(eta_squared_state/(1-eta_squared_state))
```

```
## [1] 1.797923
```

```r
# Cohen's f of interaction
sqrt(eta_squared_interaction/(1-eta_squared_interaction))
```

```
## [1] 0.1924492
```

3.6.3 Eta squared, partial eta squared, generalized eta squared for type III SS

Eta squared for type III SS: Type III SS considers the importance of the effect of an explanatory variable by separating it from all other effects and interactions. Type III SS is used to independently measure the magnitude and importance of an effect.
Type III SS can be calculated with the "Anova" function in "Car" package.

```r
tanova.iii <- car::Anova(tanova.result, type = 3)
```

- Eta squared for type III SS:

```r
effectsize::eta_squared(tanova.iii,
            partial = FALSE,
            generalized = FALSE, # Conventional eta squared calculation
            ci = 0.95,
            alternative = "two.sided")
```

```
## Type 3 ANOVAs only give sensible and informative results when covariates
##   are mean-centered and factors are coded with orthogonal contrasts (such
##   as those produced by `contr.sum`, `contr.poly`, or `contr.helmert`, but
##   *not* by the default `contr.treatment`).
```

```
## # Effect Size for ANOVA (Type III)
##
## Parameter   | Eta2 |       95% CI
## ---------------------------------
## group       | 0.02 | [0.00, 0.14]
## state       | 0.69 | [0.54, 0.78]
## group:state | 0.05 | [0.00, 0.17]
```

- Partial eta squared (Type III SS)

```r
effectsize::eta_squared(tanova.iii,
            partial = TRUE,
            generalized = FALSE,
            ci = 0.95,
            alternative = "two.sided")
```

```
## Type 3 ANOVAs only give sensible and informative results when covariates
##   are mean-centered and factors are coded with orthogonal contrasts (such
##   as those produced by `contr.sum`, `contr.poly`, or `contr.helmert`, but
##   *not* by the default `contr.treatment`).
```

```
## # Effect Size for ANOVA (Type III)
##
```

```
## Parameter   | Eta2 (partial) |       95% CI
## ------------------------------------------
## group       |           0.06 | [0.00, 0.22]
## state       |           0.73 | [0.60, 0.81]
## group:state |           0.15 | [0.01, 0.32]
```

- Generalized eta squared (Type III SS)

```
effectsize::eta_squared(tanova.iii,
            partial = FALSE,
            generalized = TRUE,
            ci = 0.95,
            alternative = "two.sided")
```

```
## Type 3 ANOVAs only give sensible and informative results when covariates
##   are mean-centered and factors are coded with orthogonal contrasts (such
##   as those produced by `contr.sum`, `contr.poly`, or `contr.helmert`, but
##   *not* by the default `contr.treatment`).
```

```
## # Effect Size for ANOVA (Type III)
##
## Parameter   | Eta2 (generalized) |       95% CI
## ---------------------------------------------
## group       |               0.06 | [0.00, 0.22]
## state       |               0.73 | [0.60, 0.81]
## group:state |               0.15 | [0.01, 0.32]
##
## - Observed variables: All
```

- Manual calculation:

```
# The ANOVA results.
tanova.iii
```

```
## Anova Table (Type III tests)
##
## Response: value
##              Sum Sq Df  F value    Pr(>F)
## (Intercept) 5178242  1 825.9220 < 2.2e-16 ***
## group         22378  1   3.5693   0.06423 .
## state        926359  2  73.8765 3.503e-16 ***
## group:state   61399  2   4.8965   0.01111 *
## Residuals    338561 54
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Calculate the sum of squares.
group_ss <- tanova.iii$"Sum Sq"[2]
state_ss <- tanova.iii$"Sum Sq"[3]
interaction_ss <- tanova.iii$"Sum Sq"[4]
error_ss <- tanova.iii$"Sum Sq"[5]
total_ss <-group_ss + state_ss + interaction_ss + error_ss

# Calculate eta squared based on Type III SS.
eta_squared_group <- group_ss / total_ss
eta_squared_state <- state_ss / total_ss
eta_squared_interaction <- interaction_ss / total_ss
```

```r
# Print the results.
cat("Eta squared for 'group':", eta_squared_group, "\n")
```

```
## Eta squared for 'group': 0.01659235
```

```r
cat("Eta squared for 'state':", eta_squared_state, "\n")
```

```
## Eta squared for 'state': 0.686855
```

```r
cat("Eta squared for 'group:state interaction':", eta_squared_interaction, "\n")
```

```
## Eta squared for 'group:state interaction': 0.04552433
```

3.7 Kruskal-Wallis ANOVA on ranks (Kruskal-Wallis H)

- Data preparation:

```r
kw <- na.omit(subset(mydata, select = c(case.no, kw.group, kw.value)))

dim(kw)
```

```
## [1] 30  3
```

```r
summary(kw)
```

```
##     case.no        kw.group           kw.value
##  Min.   : 1.00   Length:30          Min.   : 1.000
##  1st Qu.: 8.25   Class :character   1st Qu.: 8.000
##  Median :15.50   Mode  :character   Median : 9.000
##  Mean   :15.50                      Mean   : 8.533
##  3rd Qu.:22.75                      3rd Qu.:11.000
##  Max.   :30.00                      Max.   :14.000
```

```r
head(kw)
```

```
##   case.no kw.group kw.value
## 1       1        A        1
## 2       2        A        6
## 3       3        A        8
## 4       4        A        1
## 5       5        A        9
## 6       6        A        4
```

- Kruskal-Wallis test:

```r
kruskal.test(kw.value ~ kw.group,
             data = kw)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  kw.value by kw.group
## Kruskal-Wallis chi-squared = 22.325, df = 2, p-value = 1.42e-05
```

3.7.1 Eta squared

- Required package: "rcompanion"

```
rcompanion::ordinalEtaSquared(x = kw$kw.value,
                g = kw$kw.group,
                ci = TRUE)
```

```
##   eta.squared lower.ci upper.ci
## 1       0.753    0.539    0.881
```

- Manual calculation

```
# Eta squared for ordinal variable
kw.rs <- kruskal.test(kw.value ~ kw.group, data = kw)

(kw.rs$statistic[[1]]-3+1)/(30-3)
```

```
## [1] 0.7527859
```

```
H <- kw.rs$statistic[[1]]
k <- length(unique(kw$kw.group))
n <- length(kw$kw.value)

kw.ets <- (H-k+1)/(n-k)

# 95%CI using bootstrap
set.seed(123) # Set seed for reproducibility

bootstrap_ets <- replicate(1000, { # Perform 1000 bootstrap iterations
    # Resample with replacement from the original data
    boot_sample <- kw[sample(nrow(kw), replace = TRUE), ]

    # Perform Kruskal-Wallis test on the bootstrap sample
    boot_kw_rs <- kruskal.test(kw.value ~ kw.group, data = boot_sample)

    # Calculate eta squared for the bootstrap sample
    H <- boot_kw_rs$statistic[[1]]
    k <- length(unique(boot_sample$kw.group))
    n <- length(boot_sample$kw.value)
    ets <- (H - k + 1) / (n - k)

    return(ets) # Return the calculated eta squared
})

  # Estimate the 95% confidence interval
ci_lower <- quantile(bootstrap_ets, probs = 0.025)
ci_upper <- quantile(bootstrap_ets, probs = 0.975)

# Print the results

cat("Eta squared and 95% CI:",kw.ets," [", ci_lower, ", ", ci_upper, "]\n")
```

```
## Eta squared and 95% CI: 0.7527859  [ 0.52904 ,  0.8739311 ]
```

3.7.2 Epsilon squared

-Required package: "rcompanion"

```
rcompanion::epsilonSquared(x = kw$kw.value,
                g = kw$kw.group,
```

```
                ci = TRUE)
```

```
##   epsilon.squared lower.ci upper.ci
## 1            0.77    0.568    0.889
```

- Manual calculation

```
## Epsilon squared calculation
ess.org <- H/((n^2-1)/(n+1))

## Bootstrap 95%CI
set.seed(123) # Set seed for result reproducibility

# Number of bootstrap iterations
B <- 1000

# Initialize vector to store bootstrap epsilon squared values
epsilon_squared_values <- numeric(B)

for (i in 1:B) {
  # Generate bootstrap sample
  boot_indices <- sample(1:nrow(kw), replace = TRUE)
  boot_sample <- kw[boot_indices, ]

  # Perform Kruskal-Wallis test on the bootstrap sample
  kw_test <- kruskal.test(kw.value ~ kw.group, data = boot_sample)

  # H statistic
  Hb <- kw_test$statistic

  # Size of the bootstrap sample
  nb <- length(boot_sample$kw.value)

  # Calculate Epsilon squared
  ess <- Hb / ((nb^2 - 1) / (nb + 1))

  # Store the calculated value
  epsilon_squared_values[i] <- ess
}

# Calculate the 95% confidence interval
ci_lower <- quantile(epsilon_squared_values, probs = 0.025)
ci_upper <- quantile(epsilon_squared_values, probs = 0.975)

# Print the confidence interval
cat("epsilon squared and 95% CI:",ess.org,"[",ci_lower, ", ", ci_upper, "]\n")
```

```
## epsilon squared and 95% CI: 0.7698352 [ 0.56152 ,  0.8826255 ]
```

3.8 One-way RM ANOVA

- Data preparation:

```
rmanova.data <- na.omit(subset(mydata, select = c(case.no, btws.factor, subject, ws.factor, value)))

summary(rmanova.data)
```

```
##     case.no      btws.factor          subject           ws.factor
##  Min.   : 1.00   Length:72          Length:72          Length:72
##  1st Qu.:18.75   Class :character   Class :character   Class :character
##  Median :36.50   Mode  :character   Mode  :character   Mode  :character
##  Mean   :36.50
##  3rd Qu.:54.25
##  Max.   :72.00
##      value
##  Min.   :1639
##  1st Qu.:1978
##  Median :2000
##  Mean   :2003
##  3rd Qu.:2036
##  Max.   :2294
```

```
head(rmanova.data)
```

```
##   case.no btws.factor subject ws.factor value
## 1       1           A       a        t1  2000
## 2       2           A       a        t2  1978
## 3       3           A       a        t3  1962
## 4       4           A       a        t4  1873
## 5       5           A       a        t5  1782
## 6       6           A       a        t6  1737
```

- One-way RM ANOVA (one repeated factor, one within-subject factor):

```
one.rmanova.result <- aov(value ~ ws.factor +Error(subject/ws.factor),
                          data = rmanova.data)

summary(one.rmanova.result)
```

```
##
## Error: subject
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals 11 762805   69346
##
## Error: subject:ws.factor
##           Df Sum Sq Mean Sq F value Pr(>F)
## ws.factor  5   1469     294   0.052  0.998
## Residuals 55 309033    5619
```

3.8.1 Eta squared, partial eta squared, general eta squared

- Required package: "effectsize"
- Eta squared:

```
effectsize::eta_squared(one.rmanova.result,
            partial = FALSE,
            ci = 0.95,
            alternative = "two.sided")
```

```
## # Effect Size for ANOVA (Type I)
##
## Group             | Parameter |     Eta2 |       95% CI
```

```
## ------------------------------------------------------------
## subject:ws.factor | ws.factor | 1.37e-03 | [0.00, 0.00]
```

- Partial eta squared:

```
effectsize::eta_squared(one.rmanova.result,
                        partial = TRUE,
                        ci = 0.95,
                        alternative = "two.sided")
```

```
## # Effect Size for ANOVA (Type I)
##
## Group              | Parameter | Eta2 (partial) |       95% CI
## -----------------------------------------------------------------
## subject:ws.factor | ws.factor |       4.73e-03 | [0.00, 0.00]
```

- Manual calculation

```
rmanova.table <- summary(one.rmanova.result)

# Extract the sum of squares for ws.factor
ss_factor <- rmanova.table[["Error: subject:ws.factor"]][[1]]$`Sum Sq`[1]

# Extract the sum of squares for error within factor
ss_error_within_factor <- rmanova.table[["Error: subject:ws.factor"]][[1]]$`Sum Sq`[2]

# Extract the sum of squares for subject error
ss_subject <- rmanova.table[["Error: subject"]][[1]]$`Sum Sq`[1]

# Total sum of squares calculation (variation due to ws.factor + within-subject error variation + subje
total_ss <- ss_factor + ss_error_within_factor + ss_subject

# Eta squared calculation
eta_squared <- ss_factor / total_ss

# Partial eta squared calculation (variation due to ws.factor / (variation due to ws.factor + within-su
partial_eta_squared <- ss_factor / (ss_factor + ss_error_within_factor)

# Print the results
cat("Eta Squared:", eta_squared, "\n")
```

```
## Eta Squared: 0.001368266
```

```
cat("Partial Eta Squared:", partial_eta_squared, "\n")
```

```
## Partial Eta Squared: 0.004729664
```

3.9 Two-way RM ANOVA (one repeated [within-subject] & one between-subject factor)

- Data: data used in one-way RM ANOVA:

```
two.rmanova.result <- aov(value ~ btws.factor*ws.factor + Error(subject/ws.factor),
                          data = rmanova.data)

summary(two.rmanova.result)
```

```
##
```

```
## Error: subject
##             Df Sum Sq Mean Sq F value Pr(>F)
## btws.factor  2 374297  187149   4.335  0.048 *
## Residuals    9 388508   43168
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Error: subject:ws.factor
##                       Df Sum Sq Mean Sq F value   Pr(>F)
## ws.factor              5   1469     294   0.128    0.985
## btws.factor:ws.factor 10 205587   20559   8.943 7.01e-08 ***
## Residuals             45 103446    2299
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.9.1 Eta squared, partial eta squared, general eta squared

- Required package: "effectsize"
- Eta squared

```
effectsize::eta_squared(two.rmanova.result,
            partial = FALSE,
            generalized = FALSE,
            ci = 0.95,
            alternative = "two.sided")
```

```
## # Effect Size for ANOVA (Type I)
##
## Group             |             Parameter |     Eta2 |       95% CI
## ------------------------------------------------------------------
## subject           |           btws.factor |     0.35 | [0.00, 0.66]
## subject:ws.factor |             ws.factor | 1.37e-03 | [0.00, 0.00]
## subject:ws.factor | btws.factor:ws.factor |     0.19 | [0.00, 0.32]
```

- Partial eta squared

```
effectsize::eta_squared(two.rmanova.result,
            partial = TRUE,
            generalized = FALSE,
            ci = 0.95,
            alternative = "two.sided")
```

```
## # Effect Size for ANOVA (Type I)
##
## Group             |             Parameter | Eta2 (partial) |       95% CI
## ------------------------------------------------------------------------
## subject           |           btws.factor |           0.49 | [0.00, 0.75]
## subject:ws.factor |             ws.factor |           0.01 | [0.00, 0.00]
## subject:ws.factor | btws.factor:ws.factor |           0.67 | [0.45, 0.76]
```

- Generalized eta squared

```
effectsize::eta_squared(two.rmanova.result,
            partial = FALSE,
            generalized = TRUE,
            ci = 0.95,
            alternative = "two.sided")
```

```
## # Effect Size for ANOVA (Type I)
##
## Group             |             Parameter | Eta2 (generalized) |       95% CI
## ------------------------------------------------------------------------------
## subject           |           btws.factor |               0.43 | [0.00, 0.71]
## subject:ws.factor |             ws.factor |          2.98e-03 | [0.00, 0.00]
## subject:ws.factor | btws.factor:ws.factor |               0.29 | [0.00, 0.41]
##
## - Observed variables: All
```

- Manual calculation

```
# ANOVA table
two.rmanova.table <- summary(two.rmanova.result)

# Extract the sum of squared for between-subject factor
ss_btws <- two.rmanova.table$`Error: subject`[[1]]$`Sum Sq`[1]

# Extract the sum of squared for within-subject factor
ss_ws <- two.rmanova.table$`Error: subject:ws.factor`[[1]]$`Sum Sq`[1]

# Extract the sum of squared for between-subject and within-subject factors interaction
ss_btws_ws <- two.rmanova.table$`Error: subject:ws.factor`[[1]]$`Sum Sq`[2]

# Extract the sum of squared for the error of between-subject factor
ss_error_btws <- two.rmanova.table$`Error: subject`[[1]]$`Sum Sq`[2]

# Extract the sum of squared for the common error of within-subject factor and interaction
ss_error_btws_ws <- two.rmanova.table$`Error: subject:ws.factor`[[1]]$`Sum Sq`[3]

# Total sum of squared
ss_total <- ss_btws + ss_ws + ss_btws_ws + ss_error_btws + ss_error_btws_ws

# Eta squared calculation
eta_squared_btws = ss_btws / ss_total
eta_squared_ws = ss_ws / ss_total
eta_squared_btws_ws = ss_btws_ws / ss_total

# Partial eta squared calculation
eta_p_squared_btws = ss_btws / (ss_btws + ss_error_btws)
eta_p_squared_ws = ss_ws / (ss_ws + ss_error_btws_ws)
eta_p_squared_btws_ws = ss_btws_ws / (ss_btws_ws + ss_error_btws_ws)

# Print the results
cat("Eta Squared, between-subject factor (type I):", eta_squared_btws, "\n")
```

```
## Eta Squared, between-subject factor (type I): 0.3487328
```

```
cat("Eta Squared, within-subject factor (type I):", eta_squared_ws, "\n")
```

```
## Eta Squared, within-subject factor (type I): 0.001368266
```

```
cat("Eta Squared, intercation (type I):", eta_squared_btws_ws, "\n")
```

```
## Eta Squared, intercation (type I): 0.1915454
```

```r
cat("Partial Eta Squared, between-subject factor (type I):", eta_p_squared_btws, "\n")
```

```
## Partial Eta Squared, between-subject factor (type I): 0.4906853
```

```r
cat("Partial Eta Squared, within-subject factor (type I):", eta_p_squared_ws, "\n")
```

```
## Partial Eta Squared, within-subject factor (type I): 0.01399774
```

```r
cat("Partial Eta Squared, intercation (type I):", eta_p_squared_btws_ws, "\n")
```

```
## Partial Eta Squared, intercation (type I): 0.6652587
```

3.9.2 Eta squared, Partial eta squared, Generalized eta suqared (type III ss)

Type III SS: The object RM-ANOVA result is not compatible with the car::Anova function. Another package ("afex") is needed to calculate the type III SS:

```r
two.rmanova.iii <- afex::aov_car(value ~ btws.factor*ws.factor + Error(subject/ws.factor),
                        data = rmanova.data)
```

```
## Converting to factor: btws.factor
```

```
## Contrasts set to contr.sum for the following variables: btws.factor
```

```r
two.rmanova.iii
```

```
## Anova Table (Type 3 tests)
##
## Response: value
##                   Effect          df      MSE        F  ges p.value
## 1           btws.factor        2, 9 43167.56   4.34 * .432    .048
## 2             ws.factor 1.90, 17.12  6041.15     0.13 .003    .872
## 3 btws.factor:ws.factor 3.81, 17.12  6041.15 8.94 *** .295   <.001
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '+' 0.1 ' ' 1
##
## Sphericity correction method: GG
```

```r
#detailed ANOVA table
summary(two.rmanova.iii)
```

```
##
## Univariate Type III Repeated-Measures ANOVA Assuming Sphericity
##
##                          Sum Sq num Df Error SS den Df   F value    Pr(>F)
## (Intercept)           288980834      1   388508      9 6694.3981 3.082e-14 ***
## btws.factor              374297      2   388508      9    4.3354   0.04802 *
## ws.factor                  1469      5   103446     45    0.1278   0.98534
## btws.factor:ws.factor    205587     10   103446     45    8.9432 7.008e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Mauchly Tests for Sphericity
##
##                       Test statistic   p-value
## ws.factor                  0.0051573 0.0010314
## btws.factor:ws.factor      0.0051573 0.0010314
##
```

```
##
## Greenhouse-Geisser and Huynh-Feldt Corrections
##  for Departure from Sphericity
##
##                          GG eps Pr(>F[GG])
## ws.factor              0.38052  0.8715814
## btws.factor:ws.factor 0.38052  0.0004904 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##                            HF eps   Pr(>F[HF])
## ws.factor              0.4797885 0.9116910816
## btws.factor:ws.factor 0.4797885 0.0001154626
```

- Required package: "effectsize"
- eta squared

```
effectsize::eta_squared(two.rmanova.iii,
            partial = FALSE,
            generalized = FALSE,
            ci = 0.95,
            alternative = "two.sided")
```

```
## # Effect Size for ANOVA (Type III)
##
## Parameter             |     Eta2 |       95% CI
## -------------------------------------------------
## btws.factor           |     0.28 | [0.00, 0.61]
## ws.factor             | 1.08e-03 | [0.00, 0.00]
## btws.factor:ws.factor |     0.15 | [0.00, 0.00]
```

- Partial eta squared (type III ss)

```
effectsize::eta_squared(two.rmanova.iii,
            partial = TRUE,
            generalized = FALSE,
            ci = 0.95,
            alternative = "two.sided")
```

```
## # Effect Size for ANOVA (Type III)
##
## Parameter             | Eta2 (partial) |       95% CI
## -----------------------------------------------------
## btws.factor           |           0.49 | [0.00, 0.75]
## ws.factor             |       3.77e-03 | [0.00, 0.00]
## btws.factor:ws.factor |           0.35 | [0.00, 0.43]
```

- Generalized eta squared (type III ss)

```
effectsize::eta_squared(two.rmanova.iii,
            partial = FALSE,
            generalized = TRUE,
            ci = 0.95,
            alternative = "two.sided")
```

```
## # Effect Size for ANOVA (Type III)
##
```

```
## Parameter             | Eta2 (generalized) |       95% CI
## ------------------------------------------------------------
## btws.factor           |               0.33 | [0.00, 0.65]
## ws.factor             |           1.89e-03 | [0.00, 0.00]
## btws.factor:ws.factor |               0.21 | [0.00, 0.00]
##
## - Observed variables: All
```

3.9.3 Omega squared (type III ss)

- Required package: "effectsize"

```
effectsize::omega_squared(two.rmanova.iii,
            partial = TRUE,
            generalized = FALSE,
            ci = 0.95,
            alternative = "two.sided")
```

```
## # Effect Size for ANOVA (Type III)
##
## Parameter             | Omega2 (partial) |       95% CI
## ------------------------------------------------------------
## btws.factor           |             0.24 | [0.00, 0.58]
## ws.factor             |             0.00 | [0.00, 0.00]
## btws.factor:ws.factor |             0.00 | [0.00, 0.00]
```

3.10 Friedman test

- Data preparation:

```
ft.data <- na.omit(subset(mydata, select = c(ft.object, ft.time, ft.value)))

summary(ft.data)
```

```
##   ft.object           ft.time            ft.value
##  Length:40          Length:40          Min.   : 357.0
##  Class :character   Class :character   1st Qu.: 579.5
##  Mode  :character   Mode  :character   Median : 709.5
##                                        Mean   : 739.8
##                                        3rd Qu.: 913.2
##                                        Max.   :1105.0
```

```
head(ft.data)
```

```
##   ft.object ft.time ft.value
## 1         A   time1      363
## 2         A   time2      598
## 3         A   time3      357
## 4         A   time4      634
## 5         A   time5      427
## 6         A   time6      566
```

- Friedman test:

```
ft.result <- friedman.test(ft.value ~ ft.time|ft.object,
                           data =ft.data)

ft.result
```

```
##
##  Friedman rank sum test
##
## data:  ft.value and ft.time and ft.object
## Friedman chi-squared = 15.533, df = 7, p-value = 0.02974
```

3.10.1 Kendall's *W*

- First, the effect size calculation requires a cross-table of data:

```
ft.table <- tidyr::spread(data = ft.data, key = ft.time, value = ft.value)

ft.table
```

```
##   ft.object time1 time2 time3 time4 time5 time6 time7 time8
## 1         A   363   598   357   634   427   566  1099   529
## 2         B   824   553   937   939   871   696  1022  1010
## 3         C   723   444   670   838   793   985  1105   954
## 4         D   530   408   649   494   584   688   637   639
## 5         E   871   696   822   910   923   844   905  1054
```

Then, we can calculate the Kendall's *W* with the package "DescTools". Note that *transposed data excluding object column* is used for calculation.

```
DescTools::KendallW(t(ft.table[,-1]),
         correct = TRUE,
         test = TRUE)
```

```
##
##  Kendall's coefficient of concordance Wt
##
## data:  t(ft.table[, -1])
## Kendall chi-squared = 15.533, df = 7, subjects = 8, raters = 5, p-value
## = 0.02974
## alternative hypothesis: Wt is greater 0
## sample estimates:
##        Wt
## 0.4438095
```

Or, with the package "rcompanion":

```
rcompanion::kendallW(t(ft.table[,-1]),
         correct = TRUE,
         ci = TRUE)
```

```
##       W lower.ci upper.ci
## 1 0.444    0.341    0.794
```

Or, with the packave "rstatix":

```
rstatix::friedman_effsize(ft.data, ft.value ~ ft.time|ft.object)
```

```
## # A tibble: 1 x 5
##   .y.          n effsize method    magnitude
## * <chr>    <int>   <dbl> <chr>     <ord>
## 1 ft.value     5   0.444 Kendall W moderate
```

- Manual calculation:

```r
# Total sample size (N), total number of groups (k)
N <- length(unique(ft.data$ft.object))
k <- length(unique(ft.data$ft.time))

# Friedman test statistic
chi <- ft.result$statistic[[1]]

# Apply Kendall's W formula
W <- chi/(N*(k - 1))

# Print Kendall's W
print(W)
```

```
## [1] 0.4438095
```

3.11 Chi-squared test

- Data preparation:

```r
# Select columns for chi-squared test
chisq.data <- subset(mydata, select = c(level, observed))

summary(chisq.data)
```

```
##     level              observed
##  Length:218         Length:218
##  Class :character   Class :character
##  Mode  :character   Mode  :character
```

```r
dim(chisq.data)
```

```
## [1] 218   2
```

```r
head(chisq.data)
```

```
##   level observed
## 1     A      yes
## 2     A      yes
## 3     A      yes
## 4     A      yes
## 5     A      yes
## 6     A      yes
```

- Chi-squared test of independence:

Making tabulated data:

```r
chisq.table <- xtabs(~ level + observed,
              data =chisq.data)
chisq.table
```

```
##      observed
## level  no yes
##     A  21 121
##     B  22  54
```

Chi-squared test:

```r
chi.rst <- chisq.test(chisq.table,
           correct = TRUE)

print(chi.rst)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  chisq.table
## X-squared = 5.4052, df = 1, p-value = 0.02008
```

3.11.1 Cramér's V

- Required package: "effectsize"

```r
effectsize::cramers_v(chisq.data$level, chisq.data$observed, ci = 0.95, alternative = "two.sided")
```

```
## Cramer's V (adj.) |       95% CI
## --------------------------------
## 0.16              | [0.00, 0.30]
```

- Required package: "rcompanion"

```r
rcompanion::cramerV(chisq.table, ci = TRUE)
```

```
##   Cramer.V lower.ci upper.ci
## 1   0.1696  0.03465   0.3073
```

- Manual calculation:

```r
# Extract test statistic
chi <- chi.rst$statistic[[1]]

# Sample size
n <- sum(chisq.table)

# Smaller one between numbers of row and column
k <- min(nrow(chisq.table), ncol(chisq.table))

# Cramer's V calculation

V <- sqrt(chi/(n*(k-1)))

# Result output
cat("Cramer's V: ", V, "\n")
```

```
## Cramer's V:  0.1574629
```

3.12 Fisher's exact test

- Data preparation:

```r
# Select columns for Fisher's exact test
fisher.data <- subset(mydata, select = c(area, response))

# remove rows containing empty cells
fisher.data <- fisher.data[!apply(fisher.data == "",1,all), ]
```

```r
summary(fisher.data)
```

```
##      area             response
##  Length:40          Length:40
##  Class :character   Class :character
##  Mode  :character   Mode  :character
```

```r
head(fisher.data)
```

```
##   area response
## 1    A Positive
## 2    A Positive
## 3    A Positive
## 4    A Negative
## 5    A Negative
## 6    A Negative
```

- Fisher's exact test:

```r
fisher.table <- xtabs(~area + response, data = fisher.data)
fisher.table
```

```
##     response
## area Negative Positive
##    A       11        3
##    B        9       17
```

```r
fisher.rst <- fisher.test(fisher.table)
fisher.rst
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  fisher.table
## p-value = 0.0187
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##   1.291512 46.264752
## sample estimates:
## odds ratio
##   6.567572
```

3.12.1 Phi coefficient

- Required package: "effectsize"

```r
effectsize::phi(fisher.table)
```

```
## Phi (adj.) |       95% CI
## ------------------------
## 0.39       | [0.00, 1.00]
##
## - One-sided CIs: upper bound fixed at [1.00].
```

- Manual calculation:

```r
# Extract chi-sqaured statistic (corrected)
chi <- chisq.test(fisher.table,correct = TRUE)$statistic[[1]]
```

```r
# total number
N <- sum(fisher.table)

# phi coefficient calculation
round(sqrt(chi/N),2)
```

```
## [1] 0.37
```

3.13 Two-proportions z-test

- Data preparation:

```r
# Select columns for z test
zdata <- subset(mydata, select = c(test, outcome))

# remove rows containing empty cell
zdata <- zdata[!apply(zdata == "",1,all), ]
summary(zdata)
```

```
##      test             outcome
##  Length:80          Length:80
##  Class :character   Class :character
##  Mode  :character   Mode  :character
```

```r
dim(zdata)
```

```
## [1] 80  2
```

```r
head(zdata)
```

```
##   test outcome
## 1    B success
## 2    B success
## 3    B success
## 4    B    fail
## 5    A success
## 6    B success
```

- Tabulation:

```r
ztable <- xtabs(~test + outcome, data = zdata)
addmargins(ztable)
```

```
##      outcome
## test  fail success Sum
##   A      4      31  35
##   B     13      32  45
##   Sum   17      63  80
```

- Two-proportions z-test:

```r
### with prop.test(): this R basic statistical function uses the chi-squared test.

# Calculate the number of success and fail for each test level
success_count <- table(zdata$test, zdata$outcome)[,"success"]
total_count <- table(zdata$test)

# Calculate success rate
```

```r
success_rate <- success_count / total_count

# Perform z-test using prop.test
# Setting correct = FALSE option deactivates Yates' continuity correction for a result closer to z-test
test_result <- prop.test(success_count, total_count, correct = FALSE)

# Print the results
print(success_rate)
```

```
##
##         A         B
## 0.8857143 0.7111111
```

```r
print(test_result)
```

```
##
##  2-sample test for equality of proportions without continuity correction
##
## data:  success_count out of total_count
## X-squared = 3.5866, df = 1, p-value = 0.05825
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  0.00534935 0.34385700
## sample estimates:
##    prop 1    prop 2
## 0.8857143 0.7111111
```

When performing a rate comparison using a z-test, it is essential to have a sufficient number of successes and failures to justify using an approximated normal distribution. The minimum required sample size depends on the experimental context, varying based on the hypothesis being tested and the level of statistical rigor.

Rule of 10: This criterion is relatively accurate for approximating proportions using a normal distribution. However, it can be too conservative in some cases.

Rule of 5: This less strict rule is often applied in chi-squared tests and can also be used for rate comparisons.

Keep in mind that both the "Rule of 10" and the "Rule of 5" are general guidelines. The most appropriate choice should be based on the specific experiment and analysis context.

```r
### Comparing proportions of two populations with binomial distribution

# Calculate the number of successes and attempts
success_a <- sum(zdata$test == "A" & zdata$outcome == "success")
total_a <- sum(zdata$test == "A")
success_b <- sum(zdata$test == "B" & zdata$outcome == "success")
total_b <- sum(zdata$test == "B")

# Calculate success rates
p1 <- success_a / total_a
p2 <- success_b / total_b

# Availability check

  # For group A
  ifelse(total_a * p1 > 5 & total_a * (1 - p1) > 5, 'Available', 'Not available')
```

```
## [1] "Not available"
```

```r
  # For group B
  ifelse(total_b * p2 > 5 & total_b * (1 - p2) > 5, 'Available', 'Not available')
```

```
## [1] "Available"
```

```r
# Calculate pooled proportion
p <- (success_a + success_b) / (total_a + total_b)

# Calculate standard error
se <- sqrt(p * (1 - p) * (1/total_a + 1/total_b))

# Calculate z-score
z <- (p1 - p2) / se

# Calculate p-value
p_value <- 2 * (1 - pnorm(abs(z)))

# Print the results
cat("z-score: ", z, "\n")
```

```
## z-score:  1.893837
```

```r
cat("p-value: ", p_value, "\n")
```

```
## p-value:  0.0582466
```

3.13.1 Cohen's h

- Required package: "effectsize"

```r
# Use table format: Groups are COLUMNS

ztable # Groups are ROWS
```

```
##     outcome
## test fail success
##    A    4      31
##    B   13      32
```

```r
t(ztable) # Groups are COLUMNS
```

```
##          test
## outcome    A  B
##   fail     4 13
##   success 31 32
```

```r
effectsize::cohens_h(t(ztable), ci = 0.95, alternative = "two.sided")
```

```
## Cohen's h |         95% CI
## -------------------------
## -0.45     | [-0.89,  0.00]
```

- We can change the column order for a positive Cohen's h:

```r
zdata$outcome <- factor(zdata$outcome, levels = c("success", "fail"))
ztable_r <- xtabs(~test + outcome, data = zdata)
ztable_r # Groups are ROWS
```

```
##     outcome
```

```
## test success fail
##    A      31    4
##    B      32   13
```

```
t(ztable_r) # Groups are COLUMNS
```

```
##          test
## outcome    A  B
##   success 31 32
##   fail     4 13
```

```
effectsize::cohens_h(t(ztable_r), ci = 0.95, alternative = "two.sided")
```

```
## Cohen's h |       95% CI
## ------------------------
## 0.45      | [0.00, 0.89]
```

- Manual calculation:

```
abs(2 * (asin(sqrt(p1)) - asin(sqrt(p2))))
```

```
## [1] 0.4451878
```

3.14 Correlation test

- Data preparation:

```
cor.data <- na.omit(subset(mydata, select = c(case.no, cor.1, cor.2, cor.3)))

# remove rows containing empty cell
cor.data <- cor.data[!apply(cor.data == "",1,all), ]

dim(cor.data)
```

```
## [1] 32  4
```

```
summary(cor.data)
```

```
##     case.no          cor.1           cor.2           cor.3
##  Min.   : 1.00   Min.   :1.510   Min.   :10.40   Min.   :1.000
##  1st Qu.: 8.75   1st Qu.:2.583   1st Qu.:15.43   1st Qu.:1.000
##  Median :16.50   Median :3.330   Median :19.20   Median :2.000
##  Mean   :16.50   Mean   :3.218   Mean   :20.09   Mean   :2.094
##  3rd Qu.:24.25   3rd Qu.:3.610   3rd Qu.:22.80   3rd Qu.:3.000
##  Max.   :32.00   Max.   :5.420   Max.   :33.90   Max.   :3.000
```

```
head(cor.data)
```

```
##   case.no cor.1 cor.2 cor.3
## 1       1  2.62  18.1     2
## 2       2  2.88  17.8     2
## 3       3  2.32  15.5     1
## 4       4  3.22  17.3     2
## 5       5  3.44  19.7     3
## 6       6  3.46  21.0     2
```

- Correlation test:

Pearson's correlation test:

```r
cor.test(cor.data$cor.1, cor.data$cor.2, method = "pearson")
```

```
##
##  Pearson's product-moment correlation
##
## data:  cor.data$cor.1 and cor.data$cor.2
## t = 9.0336, df = 30, p-value = 4.617e-10
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.7215518 0.9273028
## sample estimates:
##       cor
## 0.8551016
```

Spearman's non-parametric correlation test:

```r
sp.cor <- psych::corr.test(cor.data$cor.2, cor.data$cor.3, method = "spearman", ci = TRUE)
sp.cor
```

```
## Call:psych::corr.test(x = cor.data$cor.2, y = cor.data$cor.3, method = "spearman",
##     ci = TRUE)
## Correlation matrix
## [1] 0.91
## Sample Size
## [1] 32
## These are the unadjusted probability values.
##   The probability values  adjusted for multiple tests are in the p.adj object.
## [1] 0
##
##  To see confidence intervals of the correlations, print with the short=FALSE option
```

```r
# To see the confidence intervals of the correlations, print with the short=FALSE option:
print(sp.cor$ci, short = FALSE)
```

```
##          lower        r     upper            p
## NA-NA 0.827059 0.912575 0.9568047 3.512499e-13
```

4 Minimal Clinical Important Difference (MCID)

- Sample data for MCID calculation example. This imaginary data has 131 cases who have received a certain intervention for postoperative pain control. Pain scores before and after the intervention were assessed using 10-points visual analogue scale (VAS), and the changes were recorded. All patients also reported their level of improvement using a 5-level global assessment scale (GAS: -2: much worse, -1: worse, 0: no change, 1: better, 2: much better). When doing clinical research, using a GAS with more detailed levels than this example is recommended.
- This example demonstrates a simple triangulating method (anchor-based and distribution-based methods). Some articles have reported an MCID that is further treated after applying the triangulating method, such as the mean or weighted mean values of all estimated MCIDs. However, none of these methods are generally accepted yet.
- Reference: Malec JF, Ketchum JM. A standard method for determining the minimal clinically important difference for rehabilitation measures. Arch Phys Med Rehabil 2020; 101: 1090-4.

4.1 Data preparation

```
# Importing data for MCID calculation example
mcid.data <- na.omit(subset(mydata, select = c(vas1, vas2, change, gas )))

# Remove the empty cells
mcid.data <- mcid.data[!apply(mcid.data == "",1,all), ]

# Check the imported data
summary(mcid.data)
```

```
##       vas1             vas2            change            gas
##  Min.   : 4.000   Min.   : 0.000   Min.   :-3.000   Min.   :-2.0000
##  1st Qu.: 6.000   1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.: 0.0000
##  Median : 8.000   Median : 4.000   Median : 3.000   Median : 0.0000
##  Mean   : 7.649   Mean   : 4.168   Mean   : 3.481   Mean   : 0.3969
##  3rd Qu.: 9.000   3rd Qu.: 7.000   3rd Qu.: 7.000   3rd Qu.: 1.0000
##  Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   : 2.0000
```

Variable dictionary:
- vas1: pre-treatment VAS score
- vas2: post-treatment VAS score
- change: changes in VAS scores (vas1 - vas2)
- gas: global assessment scale

```
dim(mcid.data)
```

```
## [1] 131   4
```

```
head(mcid.data)
```

```
##   vas1 vas2 change gas
## 1    4    4      0   0
## 2   10    5      5   0
## 3    9    6      3   0
## 4    7    7      0   0
## 5    6    1      5   0
## 6    9   10     -1  -1
```

4.2 Correlation analysis

- First, we must investigate the correlation between the change in the VAS score and the GAS score. Because the GAS will be used as an anchor for determining MCID of a postoperative intervention for pain, a high correlation coefficient (≥ 0.7) is required.
- Required package: "psych"

```
mcid.cor <- psych::corr.test(mcid.data$gas, mcid.data$change, method = "spearman", ci = TRUE)
mcid.cor
```

```
## Call:psych::corr.test(x = mcid.data$gas, y = mcid.data$change, method = "spearman",
##     ci = TRUE)
## Correlation matrix
## [1] 0.89
## Sample Size
## [1] 131
## These are the unadjusted probability values.
##   The probability values  adjusted for multiple tests are in the p.adj object.
```

```
## [1] 0
##
##  To see confidence intervals of the correlations, print with the short=FALSE option
```

```
print(mcid.cor$ci, short = FALSE)
```

```
##           lower         r     upper            p
## NA-NA 0.8426782 0.8861228 0.9181033 6.483957e-45
```

4.3 Distribution based indices

Variable dictionary:
- sd.pre: baseline standard deviation
- se_m: standardized error of measurement(SE_M)
- sd.5: 1/2 of baseline standard deviation
- se_m196: $1.96 \times SE_M$
- rci: reliable change index

- Required package: "DescTools" for the Cronbach alpha calculation

```
# baseline SD
sd.pre <- sd(mcid.data$vas1)

# se_m

calpha <- DescTools::CronbachAlpha(subset(mcid.data, select = c(change, gas)), cond = TRUE, conf.level =

r <- calpha[1]$unconditional[[1]]

se_m <- sd.pre*sqrt(1-r[1])

# Other indicators
sd.5 <- sd.pre*0.5
se_m196 <- se_m*1.96
rci <- 1.96*sd.pre*sqrt(2*(1-r))

# arrange for output
distribution.index <- data.frame(Indicator = c("SE_M",
                                               "1/2 pretreatment SD",
                                               "1.96×SE_M",
                                               "RCI",
                                               "Pretreatment SD"),
                                 Value = c(se_m, sd.5, se_m196, rci, sd.pre))
distribution.index
```

```
##             Indicator     Value
## 1                SE_M 1.1096453
## 2 1/2 pretreatment SD 0.9383366
## 3          1.96×SE_M 2.1749048
## 4                 RCI 3.0757799
## 5     Pretreatment SD 1.8766733
```

4.4 Anchor-based response variable

- For MCID calculation using the Anchor-based method, we will create a dichotomous GAS response variable. The variable will have two levels: 0 for much worse, worse, and no changes and 1 for better

and much better.

- For convenience, we will create a response variable for each distribution-based index at the same time.

```
## Transformation of the dataframe mcid.data

# Creating the variable gas.rsp: 1 if gas > 0, else 0
mcid.data$gas.rsp <- ifelse(mcid.data$gas > 0, 1, 0)

# Creating the variable pred.se_m: 1 if change > se_m, else 0
mcid.data$pred.se_m <- ifelse(mcid.data$change > se_m, 1, 0)

# Creating the variable pred.sd.5: 1 if change > sd.5, else 0
mcid.data$pred.sd.5 <- ifelse(mcid.data$change > sd.5, 1, 0)

# Creating the variable pred.se_m196: 1 if change > se_m196, else 0
mcid.data$pred.se_m196 <- ifelse(mcid.data$change > se_m196, 1, 0)

# Creating the variable pred.rci: 1 if change > rci, else 0
mcid.data$pred.rci <- ifelse(mcid.data$change > rci, 1, 0)

# Creating the variable pred.sd.pre: 1 if change > sd.pre, else 0
mcid.data$pred.sd.pre <- ifelse(mcid.data$change > sd.pre, 1, 0)

# Displaying the first 6 rows
head(mcid.data)
```

```
##   vas1 vas2 change gas gas.rsp pred.se_m pred.sd.5 pred.se_m196 pred.rci
## 1    4    4      0   0       0         0         0            0        0
## 2   10    5      5   0       0         1         1            1        1
## 3    9    6      3   0       0         1         1            1        0
## 4    7    7      0   0       0         0         0            0        0
## 5    6    1      5   0       0         1         1            1        1
## 6    9   10     -1  -1       0         0         0            0        0
##   pred.sd.pre
## 1           0
## 2           1
## 3           1
## 4           0
## 5           1
## 6           0
```

4.5 ROC analysis

- ROC analysis with a response (GAS response) and a predictor (distribution indices response):

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
# Distribution based: se_m
roc.se_m <- roc(mcid.data, gas.rsp, pred.se_m)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
# Distribution based: 1/2 pretreatment SD
roc.sd.5 <- roc(mcid.data, gas.rsp, pred.sd.5)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
# Distribution based: 1.96*se_m
roc.se_m196 <- roc(mcid.data, gas.rsp, pred.se_m196)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
# Distribution based: RCI
roc.rci <- roc(mcid.data, gas.rsp, pred.rci)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
# Distribution based: sd.pre
roc.sd.pre <- roc(mcid.data, gas.rsp, pred.sd.pre)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
roc.result <- data.frame(Index = c("SE_M",
                                   "1/2 pretreatment SD",
                                   "1.96×SE_M",
                                   "RCI",
                                   "Pretreatment SD"),
                         AUC = c(round(roc.se_m$auc[1], 2),
                                 round(roc.sd.5$auc[1], 2),
                                 round(roc.se_m196$auc[1], 2),
                                 round(roc.rci$auc[1], 2),
                                 round(roc.sd.pre$auc[1], 2)),
                         CI.low = c(round(ci(roc.se_m)[1], 2),
                                    round(ci(roc.sd.5)[1], 2),
                                    round(ci(roc.se_m196)[1], 2),
                                    round(ci(roc.rci)[1], 2),
                                    round(ci(roc.sd.pre)[1], 2)),
                         CI.upp = c(round(ci(roc.se_m)[3], 2),
                                    round(ci(roc.sd.5)[3], 2),
                                    round(ci(roc.se_m196)[3], 2),
                                    round(ci(roc.rci)[3], 2),
                                    round(ci(roc.sd.pre)[3], 2))
                         )

roc.result
```

```
##                  Index  AUC CI.low CI.upp
## 1                 SE_M 0.82   0.76   0.87
## 2 1/2 pretreatment SD 0.72   0.66   0.78
```

```
## 3          1.96×SE_M 0.84   0.78   0.89
## 4                RCI 0.85   0.79   0.91
## 5     Pretreatment SD 0.82   0.76   0.87
```

4.6 Sensitivity, specificity, Youden index

- The sensitivity, specificity, accuracy, and Youden index can be calculated from a 2×2 table:

Table 1: Contingency table of GAS and index

	≤index	>index
GAS ≤ 0	True response(-)	False response(+)
GAS > 1	False response(-)	True response(+)

Sensitivity $= \frac{\text{True response(+)}}{\text{False response(-)+True response(+)}}$: Ratio of accurately classified response(+) number among GAS > 1 cases

Specificity $= \frac{\text{True response(-)}}{\text{True response(-)+False respone(+)}}$: Ratio of accurately classified response(-) number among GAS ≤ 0 cases

Accuracy $= \frac{\text{True response(+)+True response(-)}}{\text{Total response}}$: Accurately classified response number over all cases

Youden Index = Sensitivity + Specificity − 1

```
### tabulation of GAS and SE_M
t.se_m <- table(mcid.data$gas.rsp, mcid.data$pred.se_m)

# sensitivity
t.se_m.sen <- round(t.se_m[2,2]/(t.se_m[2,1]+t.se_m[2,2]), 3)

# specificity
t.se_m.spe <- round(t.se_m[1,1]/(t.se_m[1,1]+t.se_m[1,2]), 3)

# Classification accuraty
t.se_m.acc <- round(mean(mcid.data$gas.rsp == mcid.data$pred.se_m),3)

#Younde index
t.se_m.yi <- t.se_m.sen + t.se_m.spe -1


### tabulation of GAS and 1/2 pretreatment SD
t.sd.5 <- table(mcid.data$gas.rsp, mcid.data$pred.sd.5 )

# sensitivity
t.sd.5.sen <- round(t.sd.5[2,2]/(t.sd.5[2,1]+t.sd.5[2,2]), 3)

# specificity
t.sd.5.spe <- round(t.sd.5[1,1]/(t.sd.5[1,1]+t.sd.5[1,2]), 3)

# Classification accuraty
t.sd.5.acc <- round(mean(mcid.data$gas.rsp == mcid.data$pred.sd.5), 3)

#Younde index
t.sd.5.yi <- t.sd.5.sen + t.sd.5.spe -1
```

```r
### tabulation of GAS and 1.96×SE_M
t.se_m196 <- table(mcid.data$gas.rsp, mcid.data$pred.se_m196 )

# sensitivity
t.se_m196.sen <- round(t.se_m196[2,2]/(t.se_m196[2,1]+t.se_m196[2,2]), 3)

# specificity
t.se_m196.spe <- round(t.se_m196[1,1]/(t.se_m196[1,1]+t.se_m196[1,2]), 3)

# Classification accuraty
t.se_m196.acc <- round(mean(mcid.data$gas.rsp == mcid.data$pred.se_m196), 3)

#Younde index
t.se_m196.yi <- t.se_m196.sen + t.se_m196.spe -1


### tabulation of GAS and RCI
t.rci <- table(mcid.data$gas.rsp, mcid.data$pred.rci )

# sensitivity
t.rci.sen <- round(t.rci[2,2]/(t.rci[2,1]+t.rci[2,2]), 3)

# specificity
t.rci.spe <- round(t.rci[1,1]/(t.rci[1,1]+t.rci[1,2]), 3)

# Classification accuraty
t.rci.acc <- round(mean(mcid.data$gas.rsp == mcid.data$pred.rci), 3)

#Younde index
t.rci.yi <- t.rci.sen + t.rci.spe -1


### tabulation of GAS and pretreatment SD
t.sd.pre <- table(mcid.data$gas.rsp, mcid.data$pred.sd.pre)

# sensitivity
t.sd.pre.sen <- round(t.sd.pre[2,2]/(t.sd.pre[2,1]+t.sd.pre[2,2]), 3)

# specificity
t.sd.pre.spe <- round(t.sd.pre[1,1]/(t.sd.pre[1,1]+t.sd.pre[1,2]), 3)

# Classification accuraty
t.sd.pre.acc <- round(mean(mcid.data$gas.rsp == mcid.data$pred.sd.pre), 3)

#Younde index
t.sd.pre.yi <- t.sd.pre.sen + t.sd.pre.spe -1




# Arranging the result as a table
yi.result <- data.frame(Index = c("SE_M",
                                  "1/2 pretreatment SD",
```

```
                                  "1.96xSE_M",
                                  "RCI",
                                  "Pretreatment SD"),
                                Sensitivity = c(t.se_m.sen,
                                                t.sd.5.sen,
                                                t.se_m196.sen,
                                                t.rci.sen,
                                                t.sd.pre.sen),
                                Specificity = c(t.se_m.spe,
                                                t.sd.5.spe,
                                                t.se_m196.spe,
                                                t.rci.spe,
                                                t.sd.pre.spe),
                                Accuracy = c(t.se_m.acc,
                                                t.sd.5.acc,
                                                t.se_m196.acc,
                                                t.rci.acc,
                                                t.sd.pre.acc),
                                Youden_Index = c(t.se_m.yi,
                                                t.sd.5.yi,
                                                t.se_m196.yi,
                                                t.rci.yi,
                                                t.sd.pre.yi)
)

yi.result
```

```
##                 Index Sensitivity Specificity Accuracy Youden_Index
## 1                SE_M       1.000       0.634    0.802        0.634
## 2 1/2 pretreatment SD       1.000       0.437    0.695        0.437
## 3           1.96xSE_M       0.983       0.690    0.824        0.673
## 4                 RCI       0.867       0.831    0.847        0.698
## 5     Pretreatment SD       1.000       0.634    0.802        0.634
```

4.7 Result

- Let's compare the results:

Table 2: MCID results by the Triangulating method

Index	Sensitivity	Specificity	Accuracy	Youden_Index	AUC	CI.low	CI.upp
SE_M	1.000	0.634	0.802	0.634	0.82	0.76	0.87
1/2 pretreatment SD	1.000	0.437	0.695	0.437	0.72	0.66	0.78
1.96xSE_M	0.983	0.690	0.824	0.673	0.84	0.78	0.89
RCI	0.867	0.831	0.847	0.698	0.85	0.79	0.91
Pretreatment SD	1.000	0.634	0.802	0.634	0.82	0.76	0.87

MCID: minimal clinical important difference, SE_M: standard error of measurement, SD: standard deviation, RCI: reliable change index.

4.8 Instructions for the sensitivity analysis

This is an example of performing sensitivity analysis.
In the anchor-based analysis, we transformed the GAS into a dichotomous response variable. The original

Table 3: Sensitivity analysis of MCID results using the triangulating method

Index	Sensitivity	Specificity	Accuracy	Youden_Index	AUC	CI.low	CI.upp
All GAS response							
SE_M	1.000	0.634	0.802	0.634	0.82	0.76	0.87
1/2 pretreatment SD	1.000	0.437	0.695	0.437	0.72	0.66	0.78
1.96xSE_M	0.983	0.690	0.824	0.673	0.84	0.78	0.89
RCI	0.867	0.831	0.847	0.698	0.85	0.79	0.91
Pretreatment SD	1.000	0.634	0.802	0.634	0.82	0.76	0.87
GAS response excluding "much better"							
SE_M	1.000	0.634	0.723	0.634	0.82	0.76	0.87
1/2 pretreatment SD	1.000	0.437	0.574	0.437	0.72	0.66	0.78
1.96×SE_M	1.000	0.690	0.766	0.690	0.85	0.79	0.90
RCI	1.000	0.831	0.872	0.831	0.92	0.87	0.96
Pretreatment SD	1.000	0.634	0.723	0.634	0.82	0.76	0.87
GAS response excluding "better"							
SE_M	1.000	0.634	0.759	0.634	0.82	0.76	0.87
1/2 pretreatment SD	1.000	0.437	0.630	0.437	0.72	0.66	0.78
1.96×SE_M	0.973	0.690	0.787	0.663	0.83	0.77	0.89
RCI	0.784	0.831	0.815	0.615	0.81	0.73	0.89
Pretreatment SD	1.000	0.634	0.759	0.634	0.82	0.76	0.87

MCID: minimal clinical important difference, SE_M: standard error of measurement, SD: standard deviation, RCI: reliable change index.

GAS has 5 levels (-2: much worse, -1: worse, 0: no change, 1: better, 2: much better), and the transformed variable has two levels, combining "much better" and "better" as 1 and th others as 0.
The simplest approach to conducting a sensitivity analysis involves systematically varying key parameters. In this context, GAS is the key parameter, and we can perform the sensitivity analysis by systematically transforming GAS. Initially, the analysis includes using the transformed variable as described above. Subsequently, additional transformed variables are constructed from the original GAS. For example, a variable with two levels, where "better" is coded as 1 and all others besides much better is coded as 0 and another variable with two levels, where "much better" is coded as 1, all others besides better is coded as 0. These two additional transformed variables exclude either the "better" or "much better" response. Then, perform the exact same analysis process described above. Sensitivity is assessed by evaluating the impact of variations in the key parameter on the outcomes.
A detailed process of conducting the sensitivity analysis is not provided here, because the same code will be used with different variables names.
Table 3 shows the results of the sensitivity analysis.