

# SO-SP Coupling and Memory Consolidation — Effect Size Preprocessing

Author

2023-07-28

## Table of contents

Setting Functions . . . . .	2
Donnelly2022 and Helfrich2018 . . . . .	3
Schreiner2021 . . . . .	4
Denis2021a . . . . .	5
Hahn2020 . . . . .	8

```
## Load required packages
library(mosaic)
library(tidyverse)
library(knitr)
library(kableExtra)
library(tidyr)
library(Stat2Data)
library(dplyr)
library(meta)
library(metafor)
library(dmetar)
library(metaDigitise)
library(ICC)
library(wildmeta)
library(future)
library(shinyDigitise)
library(CircStats)
library(Directional)
options(digits = 4)
knitr::opts_chunk$set(fig.pos = "H", out.extra = "",
```

```

        tidy=FALSE, size="small")

## All papers that provided source data and/or reported correlation in graphs
## have undergone preprocess to normalize the correlation coefficient
## After preprocessing in R, all correlation coefficients reported as pearson's
## r were transformed to Fisher's z by the Practical Meta-Analysis Effect Size
## Calculator developed by Dr.Wilson

```

## Setting Functions

```

## Setting functions for the effect size calculation
## Function for calculating the circular linear correlation
circular_cor <- function(x, y, rads = TRUE) {
  circlin.cor <- circlin.cor(x, y, rads = rads)
  R_squared <- circlin.cor[, 1]
  Pearsons_r <- sqrt(R_squared)
  return(data.frame(Pearsons_r = Pearsons_r, R_squared = R_squared))
}

## Function for detecting and removing outliers
# Detect inputs
remove_outliers <- function(sleepchar, scale_columns, memory = NULL) {
  if (!is.null(memory) && length(scale_columns) > 0 && !all(scale_columns %in% colnames(sleepchar))) {
    scale_columns <- which(names(sleepchar) %in% scale_columns)
  }

  # Detect rows to remove
  rows_rem <- which(rowSums(abs(scale(sleepchar[, scale_columns]))) > 3) > 0)
  print(paste("Number of rows removed:", length(rows_rem)))

  if (length(rows_rem) > 0) {
    sleepchar_rem <- sleepchar[-rows_rem, ]
  } else {
    sleepchar_rem <- sleepchar
  }

  if (!is.null(memory)) {
    if (length(rows_rem) > 0) {
      memory_rem <- memory[-rows_rem, ]
    }
  }
}

```

```

    print("Rows removed in memory.")
  } else {
    memory_rem <- memory
    print("No rows to remove in memory.")
  }
  return(list(sleepchar_rem = sleepchar_rem, memory_rem = memory_rem))
}

return(sleepchar_rem)
}

## Function for building tables for the data classification
cortable <- function(author, cptype, flip = TRUE, ...) {
  # Combine the correlation matrices into a data frame and transpose it
  cor_group <- data.frame(...)

  if (flip) cor_group <- t(cor_group)
  # Create the table caption
  cptype <- switch(cptype, "CP Phase", "SP Peak Amplitude", "CP Strength", "CP Percentage")
  caption <- paste(author, cptype, "and Memory Pearson's r Correlation Table")

  # Generate the table calling knitr::kable
  knitr::kable(cor_group, format = "markdown", caption = caption)
}

save(circular_cor, remove_outliers, cortable, file = "preprocessing_fun.RData")

```

## Donnelly2022 and Helfrich2018

```

## Begin the scatterplot analysis for estimating effect sizes
## Import graphs to shinyDigitise
## Could be downloaded from the Github repository
## Donnelly2022 <- shinyDigitise("~/Desktop/SO-SP-Coupling/so-sp-coupling/Paper/Donnelly2022")
## Helfrich2018 <- shinyDigitise("~/Desktop/SO-SP-Coupling/so-sp-coupling/Paper/Helfrich2018")
## Processed data saved in the same folder

```

## Schreiner2021

```
## Begin the calculation of effect sizes by preprocessed data
## Import source data from Schreiner 2021
Schreiner2021 <- read_csv("https://raw.githubusercontent.com/Theaang/so-sp-coupling/main/P
## view(Schreiner2021)

## Calculate the circular linear correlation
sch_phase <- circular_cor(Schreiner2021$phase, Schreiner2021$retention)
knitr::kable(sch_phase, format = "markdown", caption = "Schreiner CP Phase and Memory Pearson'
```

Table 1: Schreiner CP Phase and Memory Pearson's r Correlation Table

	Pearsons_r	R_squared
R-squared	0.3868	0.1496

```
## Calculate the coupling percentage and remove outlier(s)
Schreiner2021 <- Schreiner2021 |>
  mutate(
    spavg = (spobjects + spscenes)/2,
    cpavg = (cpobjects + cpscenes)/2,
    soavg = (soobjects + soscenes)/2,
    spsopct = cpavg/spavg,
    sosppct = cpavg/soavg)
Schreiner2021_rem <- remove_outliers(Schreiner2021, scale_columns = c("spsopct", "sosppct")
```

[1] "Number of rows removed: 0"

```
## Calculate summary statistics for the coupling percentage
favstats(~ spsopct, data = Schreiner2021_rem)
```

min	Q1	median	Q3	max	mean	sd	n	missing
0.1811	0.2234	0.2652	0.3309	0.4912	0.2775	0.07796	20	0

```
favstats(~ sosppct, data = Schreiner2021_rem)
```

min	Q1	median	Q3	max	mean	sd	n	missing
0.03134	0.1016	0.1157	0.1418	0.1769	0.1167	0.03836	20	0

```
## Test the normality condition for further interpretation
shapiro.test(Schreiner2021_rem$spsopct)
```

Shapiro-Wilk normality test

```
data: Schreiner2021_rem$spsopct
W = 0.91, p-value = 0.08
```

```
shapiro.test(Schreiner2021_rem$sospct)
```

Shapiro-Wilk normality test

```
data: Schreiner2021_rem$sospct
W = 0.96, p-value = 0.6
```

```
## Calculate the linear correlation between SO coupled SP and memory retention
cor1 <- cor(spsopct ~ retention, use = "complete", data = Schreiner2021_rem)
## Calculate the linear correlation between SP coupled SO and memory retention
cor2 <- cor(sospct ~ retention, use = "complete", data = Schreiner2021_rem)
cortable("Schreiner", 4, flip = FALSE, "SPcSO" = cor1, "SOcSP" = cor2)
```

Table 2: Schreiner CP Percentage and Memory Pearson's r Correlation Table

SPcSO	SOcSP
-0.1982	-0.3942

```
## Remove all unused variables
rm(list = ls())
load("preprocessing_fun.RData")
```

## Denis2021a

```
## Import source data from Denis 2021a
Denis2021a <- read_csv("https://raw.githubusercontent.com/Theaang/so-sp-coupling/main/Pape
## view(Denis2021a)
```

```

## Filter out the stress group and remove outlier(s) for coupling strength
Denis2021a_str <- Denis2021a |>
  dplyr::select(cond, neu_hit_fa:neg_hit_fa, n3_cp_str_all)|>
  filter(cond == 2)|>
  mutate(avg_hit_fa = (neu_hit_fa*100 + emo_hit_fa*200)/300)
Denis2021a_str_rem <- remove_outliers(Denis2021a_str, scale_columns = "n3_cp_str_all")

```

```
[1] "Number of rows removed: 0"
```

```

## Calculate summary statistics for the coupling percentage
favstats(~ n3_cp_str_all, data = Denis2021a_str_rem)

```

```

      min      Q1 median      Q3      max      mean      sd  n missing
0.2063 0.5096 0.6576 0.7219 0.8018 0.6002 0.1546 31          1

```

```

## Test the normality condition for further interpretation
shapiro.test(Denis2021a_str_rem$n3_cp_str_all)

```

Shapiro-Wilk normality test

```

data: Denis2021a_str_rem$n3_cp_str_all
W = 0.92, p-value = 0.02

```

```

## Note: The distribution of coupling strength data deviates significantly (p < 0.02)
## from the normal distribution

```

```

## Calculate the effect size for each emotional condition
cor1 <- cor(neu_hit_fa ~ n3_cp_str_all, use = "complete", data = Denis2021a_str_rem)
cor2 <- cor(emo_hit_fa ~ n3_cp_str_all, use = "complete", data = Denis2021a_str_rem)
cor3 <- cor(pos_hit_fa ~ n3_cp_str_all, use = "complete", data = Denis2021a_str_rem)
cor4 <- cor(neg_hit_fa ~ n3_cp_str_all, use = "complete", data = Denis2021a_str_rem)
## Calculate the weighted effect size for all conditions
cor5 <- cor(avg_hit_fa ~ n3_cp_str_all, use = "complete", data = Denis2021a_str_rem)
cortable("Denis", 3, flip = FALSE, "Neutral" = cor1, "Emotional" = cor2, "Positive" = cor3)

```

Table 3: Denis CP Strength and Memory Pearson's r Correlation Table

Neutral	Emotional	Positive	Negative	Weighted.Average
0.1981	0.2964	0.3467	0.1949	0.2832

```
## Filter out the stress group and remove outlier(s) for coupling percentage
Denis2021a_per <- Denis2021a |>
  dplyr::select(cond, neu_hit_fa:neg_hit_fa, n3_cp_per_all)|>
  filter(cond == 2)|>
  mutate(avg_hit_fa = (neu_hit_fa*100 + emo_hit_fa*200)/300)
Denis2021a_per_rem <- remove_outliers(Denis2021a_per, scale_columns = "n3_cp_per_all")
```

```
[1] "Number of rows removed: 0"
```

```
## Calculate summary statistics for the coupling percentage
favstats(~ n3_cp_per_all, data = Denis2021a_per_rem)
```

```
   min   Q1 median   Q3  max  mean   sd  n missing
5.009 12.32   16.8 20.89 24.2 16.14 5.134 31         1
```

```
## Test the normality condition for further interpretation
shapiro.test(Denis2021a_per_rem$n3_cp_per_all)
```

Shapiro-Wilk normality test

```
data: Denis2021a_per_rem$n3_cp_per_all
W = 0.97, p-value = 0.5
```

```
## Calculate the effect size for each emotional condition
cor1 <- cor(neu_hit_fa ~ n3_cp_per_all, use = "complete", data = Denis2021a_per_rem)
cor2 <-cor(emo_hit_fa ~ n3_cp_per_all, use = "complete", data = Denis2021a_per_rem)
cor3 <-cor(pos_hit_fa ~ n3_cp_per_all, use = "complete", data = Denis2021a_per_rem)
cor4 <-cor(neg_hit_fa ~ n3_cp_per_all, use = "complete", data = Denis2021a_per_rem)
## Calculate the weighted effect size for all conditions
cor5 <-cor(avg_hit_fa ~ n3_cp_per_all, use = "complete", data = Denis2021a_per)
cortable("Denis", 4, flip = FALSE, "Neutral" = cor1, "Emotional" = cor2, "Positive" = cor3
```

Table 4: Denis CP Percentage and Memory Pearson's r Correlation Table

Neutral	Emotional	Positive	Negative	Weighted.Average
0.1915	-0.0754	-0.1246	-0.0142	0.0085

```
## Test robustness by the bootstrap method for nonnormality data
#> num_sim <- 10000
#> set.seed(1821)
#> bootstrap_result <- do(num_sim) *
#>   cor(avg_hit_fa ~ n3_cp_str_all, data = resample(Denis2021a_str))
#> summary(bootstrap_result)

#> bootstrap_result <- as.numeric(bootstrap_result$cor)
#> ggplot(data.frame(x = bootstrap_result), aes(x = x)) +
#>   geom_histogram(binwidth = 0.05, color = "black", fill = "lightblue") +
#>   labs(title = "Histogram of Bootstrap Results",
#>         x = "Bootstrap Results (Pearson's r correlation)",
#>         y = "Frequency") +
#>   geom_vline(xintercept = mean(bootstrap_result), color = "black", linetype = "dashed")
#>   geom_vline(xintercept = avg_cor, color = "black", linetype = "dashed")

## Remove all unused variables
rm(list = ls())
load("preprocessing_fun.RData")
```

## Hahn2020

```
## Import source data from Hahn 2020
Hahn_beh <- read_csv("https://raw.githubusercontent.com/Theaang/so-sp-coupling/main/Paper/
Hahn_chphase <- read_csv("https://raw.githubusercontent.com/Theaang/so-sp-coupling/main/Pa
Hahn_champ <- read_csv("https://raw.githubusercontent.com/Theaang/so-sp-coupling/main/Pape
Hahn_chstr <- read_csv("https://raw.githubusercontent.com/Theaang/so-sp-coupling/main/Pape
Hahn_adphase <- read_csv("https://raw.githubusercontent.com/Theaang/so-sp-coupling/main/Pa
Hahn_adamp <- read_csv("https://raw.githubusercontent.com/Theaang/so-sp-coupling/main/Pape
Hahn_adstr <- read_csv("https://raw.githubusercontent.com/Theaang/so-sp-coupling/main/Pape
Hahn_pct <- read_csv("https://raw.githubusercontent.com/Theaang/so-sp-coupling/main/Paper/
#> view(Hahn_beh)
#> view(Hahn_chphase)
#> view(Hahn_champ)
```



```

#> view(Hahn_chstr)
#> view(Hahn_adphase)
#> view(Hahn_adamp)
#> view(Hahn_adstr)
#> view(Hahn_pct)

## Coupling Phase Preprocessing
## Calculate the mean preferred phase in each specified channel location (F,C,P,O) for both
Hahn_chphase <- Hahn_chphase |>
  rowwise() |>
  mutate(
    Favg = mean(c(F3, Fz, F4)),
    Cavg = mean(c(C3, Cz, C4)),
    POavg = mean(c(P3, Pz, P4, O1, O2)))
#> view(Hahn_chphase)
Hahn_adphase <- Hahn_adphase |>
  rowwise() |>
  mutate(
    Favg = mean(c(F3, Fz, F4)),
    Cavg = mean(c(C3, Cz, C4)),
    POavg = mean(c(P3, Pz, P4, O1, O2)))
#> view(Hahn_adphase)

## Calculate the circular linear correlation for the child group
variables <- c("Favg", "Cavg", "POavg")
effect_sizech <- data.frame()
for (var in variables) {
  effect_varch <- circular_cor(Hahn_chphase[[var]], Hahn_beh$ch_diff)
  effect_sizech <- rbind(effect_sizech, effect_varch)
}
rownames(effect_sizech) <- c("Frontal", "Central", "Parietal and Occipital")
knitr::kable(effect_sizech, format = "markdown", caption = "Hahn Child CP Phase and Memory")

```

Table 5: Hahn Child CP Phase and Memory Pearson's r Correlation Table

	Pearsons_r	R_squared
Frontal	0.2156	0.0465
Central	0.3706	0.1374
Parietal and Occipital	0.4017	0.1613

```

## Calculate the circular linear correlation for the adolescent group
variables <- c("Favg", "Cavg", "POavg")
effect_sizead <- data.frame()
for (var in variables) {
  effect_varad <- circular_cor(Hahn_adphase[[var]], Hahn_beh$ad_diff)
  effect_sizead <- rbind(effect_sizead, effect_varad)
}
rownames(effect_sizead) <- c("Frontal", "Central", "Parietal and Occipital")
knitr::kable(effect_sizead, format = "markdown", caption = "Hahn Adolescent CP Phase and Memory Pearson's r Correlation Table")

```

Table 6: Hahn Adolescent CP Phase and Memory Pearson's r Correlation Table

	Pearsons_r	R_squared
Frontal	0.1287	0.0166
Central	0.2758	0.0761
Parietal and Occipital	0.5787	0.3348

```

## Coupling Strength Preprocessing
## Calculate the mean strength in each specified channel location (F,C,P,O) for both groups
Hahn_chstr <- Hahn_chstr |>
  rowwise() |>
  mutate(
    Favg = mean(c(F3, Fz, F4)),
    Cavg = mean(c(C3, Cz, C4)),
    POavg = mean(c(P3, Pz, P4, O1, O2))) |>
  dplyr::select(Favg, Cavg, POavg)
#> view(Hahn_chstr)
Hahn_adstr <- Hahn_adstr |>
  rowwise() |>
  mutate(
    Favg = mean(c(F3, Fz, F4)),
    Cavg = mean(c(C3, Cz, C4)),
    POavg = mean(c(P3, Pz, P4, O1, O2))) |>
  dplyr::select(Favg, Cavg, POavg)
#> view(Hahn_adstr)

```

```

## Detect and Remove outlier(s)
chresult <- remove_outliers(Hahn_chstr, scale_columns = c("Favg", "Cavg", "POavg"), memory = "no")

```

```
[1] "Number of rows removed: 1"
```

```
[1] "Rows removed in memory."
```

```
Hahn_chstr_rem <- chresult$sleepchar_rem
Hahn_chbeh_rem <- chresult$memory_rem
#> view(Hahn_chstr_rem)
#> view(Hahn_chbeh_rem)
adresult <- remove_outliers(Hahn_adstr, scale_columns = c("Favg", "Cavg", "POavg"), memory
```

```
[1] "Number of rows removed: 1"
```

```
[1] "Rows removed in memory."
```

```
Hahn_adstr_rem <- adresult$sleepchar_rem
Hahn_adbeh_rem <- adresult$memory_rem
#> view(Hahn_adstr_rem)
#> view(Hahn_adbeh_rem)

# Calculate correlation coefficients for the child group in each channel location
cor_ch <- c(
  Frontal = cor(Hahn_chbeh_rem$ch_diff ~ Hahn_chstr_rem$Favg, use = "complete"),
  Central = cor(Hahn_chbeh_rem$ch_diff ~ Hahn_chstr_rem$Cavg, use = "complete"),
  "Parietal and Occipital" = cor(Hahn_chbeh_rem$ch_diff ~ Hahn_chstr_rem$POavg, use = "com
)

# Calculate correlation coefficients for the adolescent group in each channel location
cor_ad <- c(
  Frontal = cor(Hahn_adbeh_rem$ad_diff ~ Hahn_adstr_rem$Favg, use = "complete"),
  Central = cor(Hahn_adbeh_rem$ad_diff ~ Hahn_adstr_rem$Cavg, use = "complete"),
  "Parietal and Occipital" = cor(Hahn_adbeh_rem$ad_diff ~ Hahn_adstr_rem$POavg, use = "com
)

# Create the table
cortable("Hahn", 3, flip = FALSE, Child = cor_ch, Adolescent = cor_ad)
```

Table 7: Hahn CP Strength and Memory Pearson's r Correlation Table

	Child	Adolescent
Frontal	-0.0379	0.3374
Central	0.2270	-0.1520
Parietal and Occipital	-0.1909	-0.0093

```

## Coupling Percentage Preprocessing
## Detect and Remove outlier(s)
chpct <- remove_outliers(Hahn_pct, scale_columns = c("ch_n2", "ch_n3"), memory = Hahn_beh)

[1] "Number of rows removed: 0"
[1] "No rows to remove in memory."

Hahn_chpct_rem <- chpct$sleepchar_rem
Hahn_chbeh_rem <- chpct$memory_rem

adpct <- remove_outliers(Hahn_pct, scale_columns = c("ad_n2", "ad_n3"), memory = Hahn_beh)

[1] "Number of rows removed: 0"
[1] "No rows to remove in memory."

Hahn_adpct_rem <- adpct$sleepchar_rem
Hahn_adbeh_rem <- adpct$memory_rem

# Calculate correlation coefficients for the child group in each sleep stage
cor_ch <- c(
  N2 = cor(Hahn_chbeh_rem$ch_diff ~ Hahn_chpct_rem$ch_n2, use = "complete"),
  N3 = cor(Hahn_chbeh_rem$ch_diff ~ Hahn_chpct_rem$ch_n3, use = "complete")
)

# Calculate correlation coefficients for the adolescent group in each sleep stage
cor_ad <- c(
  N2 = cor(Hahn_chbeh_rem$ad_diff ~ Hahn_adpct_rem$ad_n2, use = "complete"),
  N3 = cor(Hahn_chbeh_rem$ad_diff ~ Hahn_adpct_rem$ad_n3, use = "complete")
)

# Create the table
cortable("Hahn", 4, Child = cor_ch, Adolescent = cor_ad)

```

Table 8: Hahn CP Percentage and Memory Pearson's r Correlation Table

	N2	N3
Child	-0.0453	0.0852
Adolescent	-0.2881	-0.1404

N2	N3
----	----

```
## Remove all unused variables  
rm(list = ls())  
load("preprocessing_fun.RData")
```