final syntax gill

Table of contents

```
basePath <- "C:/Users/P70066878/Documents/R_analyses/gill_2_meta"
##change to the folder that the files are in. To see, where you are: getwd()
dataPath <- basePath
workingPath <- file.path(basePath, "output")
#filenameToImport <- 'results for import.csv'

if(!require(ufs)){install.packages('ufs')}
```

```
Loading required package: ufs
```

```
if(!require(metafor)){install.packages('metafor')}
```

```
Loading required package: metafor

Loading required package: Matrix

Loading required package: metadat

Loading required package: numDeriv


Loading the 'metafor' package (version 4.0-0). For an
introduction to the package please type: help(metafor)
```

```r
if(!require(plyr)){install.packages('plyr')}
```

```
Loading required package: plyr
```

```r
if(!require(readxl)){install.packages('readxl')}
```

```
Loading required package: readxl
```

```r
if(!require(ggplot2)){install.packages('ggplot2')}
```

```
Loading required package: ggplot2
```

```r
if(!require(tidyverse)){install.packages('tidyverse')}
```

```
Loading required package: tidyverse
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.2     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.0
v lubridate 1.9.2     v tibble    3.2.1
v purrr     1.0.1     v tidyr     1.3.0
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::arrange()   masks plyr::arrange()
x purrr::compact()   masks plyr::compact()
x dplyr::count()     masks plyr::count()
x dplyr::desc()      masks plyr::desc()
x tidyr::expand()    masks Matrix::expand()
x dplyr::failwith()  masks plyr::failwith()
x dplyr::filter()    masks stats::filter()
x dplyr::id()        masks plyr::id()
x dplyr::lag()       masks stats::lag()
x dplyr::mutate()    masks plyr::mutate()
x tidyr::pack()      masks Matrix::pack()
x dplyr::rename()    masks plyr::rename()
x dplyr::summarise() masks plyr::summarise()
x dplyr::summarize() masks plyr::summarize()
x tidyr::unpack()    masks Matrix::unpack()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```r
correlationCoefficients <- c(.3, .5, .7)

### Import results
dat <- as.data.frame(readxl::read_excel("Copy of Current and original data updated 19022024.x

# Exclude Xu et al

dat <-dat %>% filter(studyID != "Xu, F., et al. (2017).")
```

```r
### Compute variance of Cohen's d, where it was supplied (if Cohen's d was
### supplied in the datafile, this means it was computed on the basis of
### changes between pre- and post-test, so we use post-test sample sizes).
dat$d_variance <- convert.d.to.variance(dat$d, dat$t1_n1, dat$t1_n2)


### Compute standardized mean change per group, based on on:
###    http://stats.stackexchange.com/questions/44745
dat$t1_mean1 <- as.numeric(dat$t1_mean1)
```

```
Warning: NAs introduced by coercion
```

```r
dat$t1_mean2 <- as.numeric(dat$t1_mean2)
```

```
Warning: NAs introduced by coercion
```

```r
dat$t0_mean1 <- as.numeric(dat$t0_mean1)
```

```
Warning: NAs introduced by coercion
```

```r
dat$t0_mean1 <- as.numeric(dat$t0_mean1)
dat$t0_mean2 <- as.numeric(dat$t0_mean2)
```

```
Warning: NAs introduced by coercion
```

```r
dat$t0_sd1 <- as.numeric(dat$t0_sd1)
```

```
Warning: NAs introduced by coercion
```

```r
dat$t0_sd2 <- as.numeric(dat$t0_sd2)

dat$dmc1 <- (dat$t1_mean1 - dat$t0_mean1) / dat$t0_sd1
dat$dmc2 <- (dat$t1_mean2 - dat$t0_mean2) / dat$t0_sd2
```

```r
### Compute variances of these estimates (for each correlation
### specified in correlationCoefficients).
for (i in correlationCoefficients) {
  dat[, paste0('dmc1_v_', i)] <- (2 * sum(1,-i))/(dat$t1_n1 + (dat$dmc1^2 / (2 * dat$t1_n1)))
  dat[, paste0('dmc2_v_', i)] <- (2 * sum(1,-i))/(dat$t1_n2 + (dat$dmc2^2 / (2 * dat$t1_n2)))
}

### Compute final effect size estimate, based on:
###    http://stats.stackexchange.com/questions/190712
dat$es <- dat$dmc1 - dat$dmc2
```

```r
dat[, paste0('es_v_', correlationCoefficients)] <-
  dat[, paste0('dmc1_v_', correlationCoefficients)] +
  dat[, paste0('dmc2_v_', correlationCoefficients)]


### Now add the manually computed effect sizes
dat$es <- ifelse(is.na(dat$es) & !is.na(dat$d), dat$d, dat$es)


### Store unique variable names (after trimming)
dat$variable <- gdata::trim(dat$`coder 3 (consensus with 3rd coder)`)


### Invert some variables
dat$es_raw <- dat$es;
dat$es <- ifelse(grepl('inverted', dat$variable), -1 * dat$es_raw, dat$es_raw)


### Remove 'inverted' from the variable names
dat$variable_raw <- dat$variable;
dat$variable <- sub(' \\(inverted\\)', '', dat$variable)


### Merge QOL variables into one
dat$variable <- sub('.*QOL.*', 'QOL', dat$variable)
```

```r
##make the names tolower

dat$variable <- tolower(dat$variable)
```

```r
### Now, for every question (every study category represents a different
### research question), we need to average effect sizes within each study.
### Therefore, we first create a new variable where we combine these two.

dat$study_category_var <- factor(paste0(dat$studyID, "_",
                                        dat$study_category, "_",
                                        dat$variable))
```

```r
### Now loop through the correlations and conduct the meta-analyses

rma.withinStudy <- list();
dat.rma <- list();
rma.overStudies <- list();
for (currentCorrelation in correlationCoefficients) {
  ### Conduct and store within-study meta-analyses
  rma.withinStudy[[paste0("r_", currentCorrelation)]] <-
    by(dat, dat$study_category_var, function(dat) {
      if (!all(is.na(dat[, paste0('es_v_', currentCorrelation)]))) {
        return(rma(yi=dat$es, vi=dat[, paste0('es_v_', currentCorrelation)]));
      }
    })

  ### Extract effect sizes and variances
  dat.rma[[paste0("r_", currentCorrelation)]] <-
    ldply(rma.withinStudy[[paste0("r_", currentCorrelation)]], function(x) {
      if(!is.null(x)) {
        return(data.frame(yi=coef(x)[1], vi=vcov(x)[1]));
      }
    })



  ### Split study/category/variable columns
  dat.rma[[paste0("r_", currentCorrelation)]] <-
    cbind(dat.rma[[paste0("r_", currentCorrelation)]],
          matrix(unlist(strsplit(dat.rma[[paste0("r_", currentCorrelation)]][, '.id'], "_"))
                 ncol=3, byrow=TRUE))
  colnames(dat.rma[[paste0("r_", currentCorrelation)]]) <-
```

```
    c('study_category_var', 'yi', 'vi', 'study', 'category', 'variable')

  ### Add category_variable variable (because we should conduct different
  ### meta-analyses for each variable, and for each category
  dat.rma[[paste0("r_", currentCorrelation)]]$category_variable <-
    paste0(dat.rma[[paste0("r_", currentCorrelation)]]$category,
           "_",
           dat.rma[[paste0("r_", currentCorrelation)]]$variable)

  ### Conduct real over-study meta-analysis
  rma.overStudies[[paste0("r_", currentCorrelation)]] <-
    by(dat.rma[[paste0("r_", currentCorrelation)]],
       dat.rma[[paste0("r_", currentCorrelation)]]$category_variable,
       function(dat) {
         return(rma(yi=dat$yi, vi=dat$vi))
       })
}
```

```
Warning: Studies with NAs omitted from model fitting.

Warning: Studies with NAs omitted from model fitting.

Warning: Studies with NAs omitted from model fitting.
```

```
### Extract confidence intervals and means, first looping over
### the correlation sizes
rma.overStudies.confInts <- ldply(rma.overStudies, .id='correlation', function(x) {
  ### Then looping over the research question types
  return(ldply(x, .id='variable', function (x) {
    return(data.frame(lb=x$ci.lb, es=x$b, ub=x$ci.ub))
  }))
})

write.csv(rma.overStudies.confInts,
          file.path(workingPath,
                    paste0('Within study meta-analysis confidence intervals.csv')))
```

```
# Create directory if it doesn't exist
output_dir <- "C:/Users/P70066878/Documents/R_analyses/gill_2_meta/output/"
if (!file.exists(output_dir)) {
  dir.create(output_dir, recursive = TRUE)
```

```
}

# Write CSV file to the specified directory
write.csv(rma.overStudies.confInts,
          file.path(output_dir, "Within study meta-analysis confidence intervals.csv"))
```

```
### DiamondPlot functions zitten nu in userfriendlyscience versie 0.4-2, zorg dat die is gel

### Extract those for the first research question and r=.3
dat1 <- rma.overStudies.confInts[1:9, ]
dat1$y <- 1:nrow(dat1);
dat1$variable <- sub('.*_(.*)', '\\1', dat1$variable);
dat1$variable <- gsub("(?<=\\b)([a-z])", "\\U\\1", tolower(dat1$variable), perl=TRUE);
row.names(dat1) <- dat1$variable


## save as excel file

library(openxlsx)


write.xlsx(dat1, file = "dat1.xlsx")
```

```
dat2 <- rma.overStudies.confInts[10:18, ]
dat2$y <- 1:nrow(dat2);
dat2$variable <- sub('.*_(.*)', '\\1', dat2$variable);
dat2$variable <- gsub("(?<=\\b)([a-z])", "\\U\\1", tolower(dat2$variable), perl=TRUE);
row.names(dat2) <- dat2$variable;


# write an excel file

write.xlsx(dat2, file = "dat2.xlsx")
```

```
dat3 <- rma.overStudies.confInts[19:23, ]
dat3$y <- 1:nrow(dat3);
dat3$variable <- sub('.*_(.*)', '\\1', dat3$variable);
dat3$variable <- gsub("(?<=\\b)([a-z])", "\\U\\1", tolower(dat3$variable), perl=TRUE);
row.names(dat3) <- dat3$variable;


write.xlsx(dat3, file = "dat3.xlsx")
```

```r
#import files

dat1 <-as.data.frame(readxl::read_excel("dat1.xlsx"))


# Reorder the dataframe
reordered_dat1 <- dat1[nrow(dat1):1, ]

# Reset row names
rownames(reordered_dat1 ) <- NULL



dat2 <- as.data.frame(readxl::read_excel("dat2.xlsx"))


dat3 <- as.data.frame(readxl::read_excel("dat3.xlsx"))
```

```r
ylabs =  c("Psychological disorders", "Inhibition", "Mood", "Outcome Expectations", "Quality
           "Selective Attention", "Self-Efficacy", "Self-Esteem", "Strength",
           "Stress", "Intention")
c("Inhibition", "Mood", "Outcome Expectations", "Quality of life", "Selective Attention", "Se
  "Strength", "Stress", "Intention", "Psychological disorders")
```

```
 [1] "Inhibition"              "Mood"
 [3] "Outcome Expectations"    "Quality of life"
 [5] "Selective Attention"     "Self-Efficacy"
 [7] "Self-Esteem"             "Strength"
 [9] "Stress"                  "Intention"
[11] "Psychological disorders"
```

```r
# Helper function
compute_effect_sizes <- function(data) {

  try <- data %>%
    select(studyID, study_category, variable) %>%
    group_by(study_category, variable) %>%
    summarise(num_effect_sizes = n()) %>%
    as.data.frame() %>%
    rename(K = num_effect_sizes)
```

```
  effects_list <- list()
  for (i in unique(try$study_category)) {
    effects_list[[paste0("effects_cat", i)]] <- try %>% filter(study_category == i)
  }

  return(effects_list)
}

rez <-compute_effect_sizes(dat)
```

```
`summarise()` has grouped output by 'study_category'. You can override using
the `.groups` argument.
```

```
#FIRST RQ

first_cat_k <- rez[[1]]

row_names_1 <- rownames(dat1[["variable"]])



dat_full1 <-  left_join(dat1, first_cat_k )
```

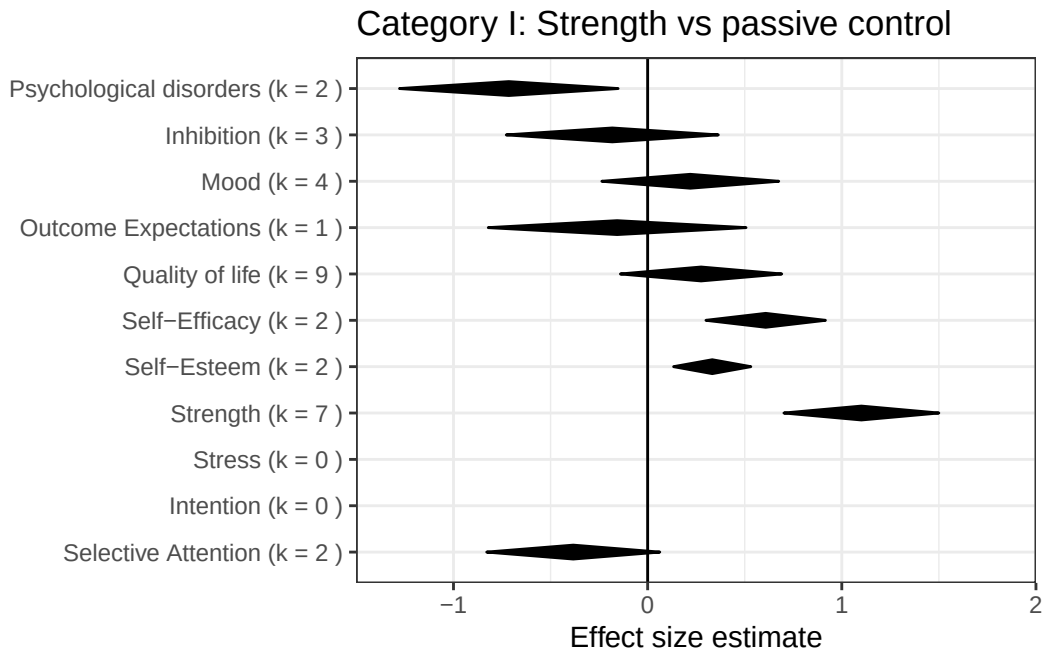
```
Joining with `by = join_by(variable)`
```

```
dat_full1 <- merge(dat1, first_cat_k)



dat1 <-as.data.frame(readxl::read_excel("dat1 - Copy.xlsx"))



dat1 <-as.data.frame(readxl::read_excel("dat1 - Copy.xlsx", sheet = 2))



#plot

dat1$combined_column <- paste(dat1$variable,  "(k =", dat1$k, ")", sep = " ")
```

```
diam1 <-diamondPlot(dat1[, 3:6], colourCol = 2,
            xlab = 'Effect size estimate',
            yLabels =  dat1$combined_column ,
            generateColours=c("red", "green")) +

  scale_x_continuous(limits = c(-1.5, 2), breaks = seq(-1, 2,  by = 1),  expand = c(0, 0)) +

  ggtitle("Category I: Strength vs passive control") +
  theme_bw() + geom_vline(xintercept=0);
```

```
Warning in ggplot2::geom_polygon(tmpDf, mapping = ggplot2::aes_string(x = "x", : Ignoring unk
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
```

```
diam1
```

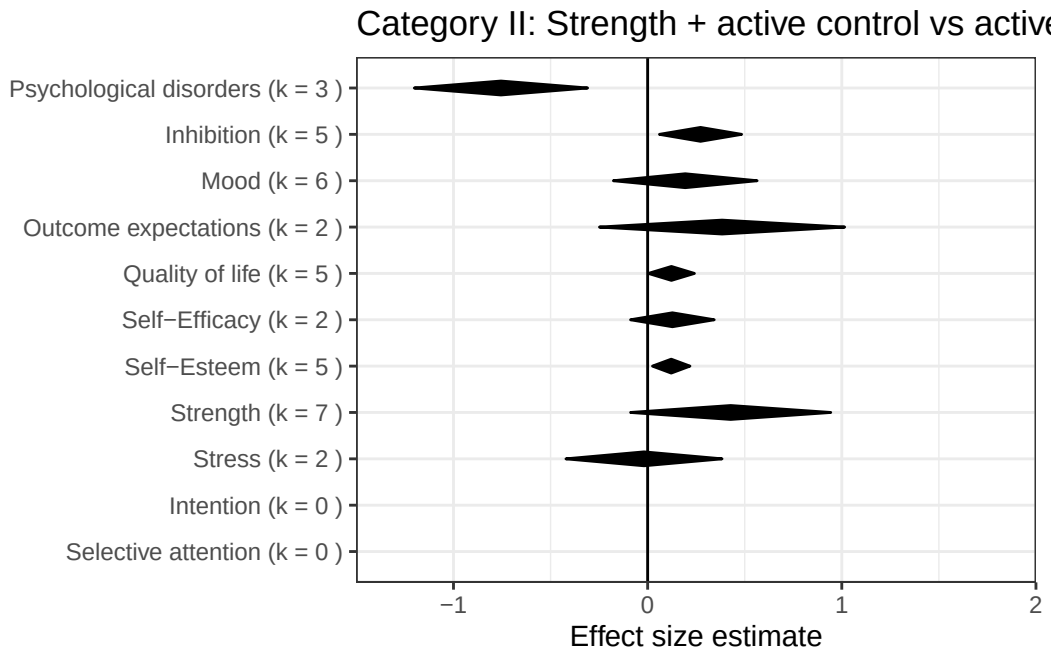



```
ggsave(filename = "RQ1.jpg", plot = diam1, width = 6, height = 4, dpi = 300)
```

```
### SECOND Rq


dat2 <-as.data.frame(readxl::read_excel("dat2 - Copy.xlsx", sheet = 2))
second_cat_k <- rez[[2]]

dat2$combined_column <- paste(dat2$variable,  "(k =", dat2$k, ")", sep = " ")


#plot

diam2 <-diamondPlot(dat2[, 3:6], colourCol = 2,
                    xlab = 'Effect size estimate',
                    yLabels =  dat2$combined_column ,
                    generateColours=c("red", "green")) +

  scale_x_continuous(limits = c(-1.5, 2), breaks = seq(-1, 2,  by = 1),  expand = c(0, 0)) +

  ggtitle("Category II: Strength + active control vs active control") +
  theme_bw() + geom_vline(xintercept=0);
```

```
Warning in ggplot2::geom_polygon(tmpDf, mapping = ggplot2::aes_string(x = "x", : Ignoring unk
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
```

```
diam2
```



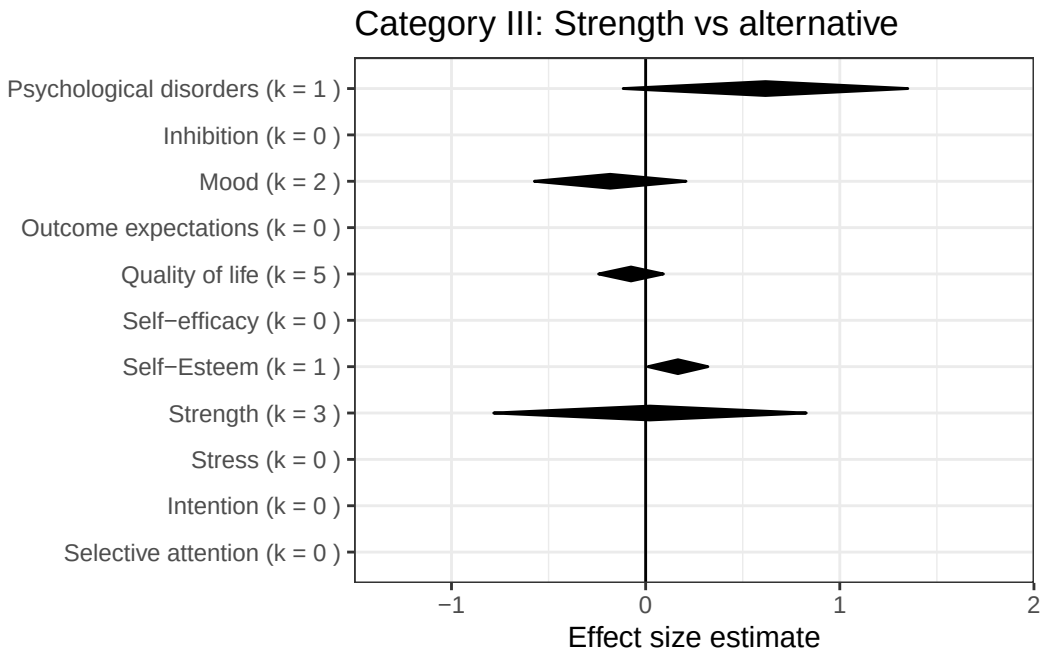

```
ggsave(filename = "RQ2.jpg", plot = diam2, width = 7, height = 4, dpi = 300)
```

```
#### RQ3

#retrieve k s for the third q

third_cat_k <- rez[[3]]
```

```r
#import df


dat3 <-as.data.frame(readxl::read_excel("dat3 - Copy.xlsx", sheet = 2))

#combined col


dat3$combined_column <- paste(dat3$variable,  "(k =", dat3$k, ")", sep = " ")


### plot 3

diam3 <-diamondPlot(dat3[, 3:6], colourCol = 2,
                    xlab = 'Effect size estimate',
                    yLabels =  dat3$combined_column ,
                    generateColours=c("red", "green")) +

 scale_x_continuous(limits = c(-1.5, 2), breaks = seq(-1, 2,  by = 1),  expand = c(0, 0)) +

  ggtitle("Category III: Strength vs alternative") +

  theme_bw() + geom_vline(xintercept=0);
```

```
Warning in ggplot2::geom_polygon(tmpDf, mapping = ggplot2::aes_string(x = "x", : Ignoring unk
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
Ignoring unknown parameters: `colourCol` and `generateColours`
```

```r
diam3
```




```
ggsave(filename = "RQ3.jpg", plot = diam3, width = 7, height = 4, dpi = 300)
```