

Multilevel Meta-Analysis Tutorial

Hu, M., Koh, P.S., Soh, X.C., Hartanto, A., Majeed, N.M.

Introduction

The following script is designed to perform a multilevel meta-analysis using the `metafor` and `lmerTest` packages in R.

The analysis is based on data from Hartanto et al. (2024) and focuses on the relationship between smartphone presence and cognitive functions. The original paper can be found here: <https://doi.org/10.1037/tmb0000123>.

The script includes steps for data preparation, effect size calculation, overall effect size computation, forest plot generation, tests for publication bias, and moderation analysis.

The script is structured to be run in RStudio, and it includes comments to guide users through each step of the process.

Setting Up

This section sets up the working environment, installs necessary packages, loads the required libraries, and reads in the data.

If the `metafor` and `lmerTest` packages are not already installed, use the `install.packages()` function to install them.

Explanation of the Code

- The `setwd()` function sets the working directory to the location of the script, ensuring that all file paths are relative to the script's location.
- The `library()` function loads the `metafor` and `lmerTest` packages.
- The `options()` function is used to adjust the display settings, specifically to disable scientific notation and set the number of digits displayed.
- The `read.csv()` function reads in the data from a CSV file named "SPC.csv", which contains the data drawn from Hartanto et al. (2024).
- A new column named "ID" is created in the data frame to assign unique IDs to each row of data.

```
### Set Up -----  
  
# R version 4.5.0  
  
# Set working directory to that of script's current location  
setwd(dirname(rstudioapi::getActiveDocumentContext()$path))  
  
# Load packages  
library(metafor) # version 4.8-0
```

```
## Loading required package: Matrix
```

```
## Loading required package: metadat
## Loading required package: numDeriv
##
## Loading the 'metafor' package (version 4.8-0). For an
## introduction to the package please type: help(metafor)
library(lmerTest) # version 3.1-3

## Loading required package: lme4
##
## Attaching package: 'lmerTest'
## The following object is masked from 'package:lme4':
##
##     lmer
## The following object is masked from 'package:stats':
##
##     step
# Display settings (to disable scientific notation)
options(scipen = 9999, digits = 4)

# Read in data drawn from Hartanto et al. 2024
multilevelmeta_raw = read.csv("SPC.csv")

# Create new column with unique IDs
multilevelmeta_raw$ID = 1:nrow(multilevelmeta_raw)
```

Prepare Data

This section prepares the data for analysis by computing effect sizes for each study and organizing the data frame.

The `multimeta_raw` data frame does not include pre-computed effect sizes for each study. However, the data frame does include the necessary information to compute effect sizes, such as sample sizes, means, and standard deviations for each group (presence of smartphones and absence of smartphones). The `escalc()` function from `metafor` package can use this information to compute the effect sizes directly.

For more information on the `escalc()` function, refer to `?escalc` in R.

Explanation of the Code

- The `measure` argument specifies the type of effect size to be calculated, which in this case is “SMD” (Standardized Mean Difference). In the `metafor` package, specifying “SMD” computes Hedge’s g by default.
- The `n1i` and `n2i` arguments specify the columns for the sample sizes of each group.
- The `m1i` and `m2i` arguments specify the columns for the means of each group.
- The `sd1i` and `sd2i` arguments specify the columns for the standard deviations of each group.
- Afterwards, the `escalc()` function computes the effect sizes (y_i) and their corresponding sampling variances (v_i) for each study.

- The `multilevelmeta$publication` variable is converted to a factor with specified levels to ensure that the publication types (e.g., journal articles, thesis/dissertation, conference) are ordered correctly in the forest plot.
- The `c()` function combines the types of publication into a character vector.
- The `multilevelmeta` data frame is then sorted by the type of publication (e.g., journal articles, thesis/dissertation, conference) and corresponding effect sizes to facilitate clearer visualization in the forest plot. Do note that the comma before the closing square bracket is required, as it indicates that we are keeping the columns while reordering the rows.

```
### Prepare Data -----
# Compute effect sizes for each study
multilevelmeta = escalc(
  # Type of effect size measure
  measure = "SMD",

  # Columns for sample size of each group
  n1i = n_p,
  n2i = n_a,

  # Columns for means of each group
  m1i = cog_M_p,
  m2i = cog_M_a,

  # Columns for standard deviation of each group
  sd1i = cog_SD_p,
  sd2i = cog_SD_a,

  # Specify data.frame that the information will be extracted from
  data = multilevelmeta_raw
)

# Convert publication type to a factor with specified levels
multilevelmeta$publication = factor(
  multilevelmeta$publication,
  levels = c("Journal article", "Thesis/dissertation", "Conference")
)

# Order the data frame based on publication type and effect sizes (yi)
multilevelmeta = multilevelmeta[order(multilevelmeta$publication, multilevelmeta$yi), ]
```

Computing the Overall Effect Size

This section estimates the overall effect size using the `rma.mv()` function from the `metafor` package.

Specifically, we demonstrate the difference in calculating the overall effect size using a traditional meta-analysis approach versus a multilevel meta-analysis approach.

The traditional meta-analysis approach is not recommended for multilevel data, as it does not account for the nested structure of the data (i.e., studies nested within labs).

The multilevel meta-analysis approach is more appropriate for this type of data, as it accounts for the hierarchical structure and allows for the inclusion of random effects.

Explanation of the Multilevel Meta-Analysis Code

- The `rma.mv()` function is used to compute the overall effect size, accounting for the nested structure of the data.
- The `random` argument specifies the random effects structure, where `~ 1 | lab_id / ID` indicates that random intercepts are included for both the lab and individual studies, accounting for the nested structure of the data.
- The `yi` and `vi` arguments specify the effect size estimates and their corresponding sampling variances, respectively.
- The `data` argument specifies the data frame containing the effect size estimates and variances.
- The `summary()` function is used to display the results of the meta-analysis, including the overall effect size estimate and its confidence interval.

```
### Compute Overall Effect Size -----

# Using multilevel code
# The code below is the recommended approach for multilevel meta-analysis
mlmmetaresults = rma.mv(
  # Effect size estimates
  yi = yi,
  # Sampling variances
  V = vi,
  # Include random effects for grouping variable (i.e., lab)
  random = ~ 1 | lab_id / ID,
  # Specify where to get the data from
  data = multilevelmeta
)

# summary function used to provide detailed results of the meta-analysis
summary(mlmmetaresults)

##
## Multivariate Meta-Analysis Model (k = 136; method: REML)
##
##   logLik  Deviance      AIC      BIC     AICc
##   3.5443   -7.0885   -1.0885    7.6273   -0.9053
##
## Variance Components:
##
##           estim      sqrt  nlvls  fixed    factor
## sigma^2.1  0.0305  0.1747    22     no    lab_id
## sigma^2.2  0.0000  0.0000   136     no  lab_id/ID
##
## Test for Heterogeneity:
## Q(df = 135) = 213.5023, p-val < .0001
##
## Model Results:
##
## estimate      se      zval    pval    ci.lb    ci.ub
##  -0.0315  0.0498  -0.6335  0.5264  -0.1291  0.0660
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```

# Using traditional meta code (refer to meta_traditional.R)
# The code below is for demonstration purposes only
# The code below is not recommended for multilevel meta-analysis
usingsimplemetacode = rma(
  yi = yi,
  vi = vi,
  data = multilevelmeta
)
summary(usingsimplemetacode)

##
## Random-Effects Model (k = 136; tau^2 estimator: REML)
##
##   logLik  deviance      AIC      BIC      AICc
##   -5.3141   10.6282   14.6282   20.4387   14.7191
##
## tau^2 (estimated amount of total heterogeneity): 0.0106 (SE = 0.0049)
## tau (square root of estimated tau^2 value):      0.1027
## I^2 (total heterogeneity / total variability):    25.36%
## H^2 (total variability / sampling variability):    1.34
##
## Test for Heterogeneity:
## Q(df = 135) = 213.5023, p-val < .0001
##
## Model Results:
##
## estimate      se      zval      pval      ci.lb      ci.ub
##  -0.0481  0.0181  -2.6651  0.0077  -0.0835  -0.0127  **
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Forest Plot

This section generates a forest plot to visually represent the effect sizes and confidence intervals for each study included in the meta-analysis.

The forest plot is created using the `forest()` function from the `metafor` package.

The plot includes the following features:

- Arrangement of studies by effect sizes
- Sample size information for both presence of smartphones and absence of smartphones groups
- Custom headers for the plot
- Custom labels for the studies

Saving the forest plot as a PDF file allows for easy sharing and presentation, and allows adjustment to the plot's dimensions.

Explanation of the Code

- The `pdf()` function starts the graphics device driver to create PDF files, and the `file` argument specifies the name of the file.

- The `width` and `height` arguments adjust the dimensions of the PDF file.
- The `forest()` function is used to create the forest plot, and the `mlmmetaresults` object contains the results of the meta-analysis.
- The `order` argument specifies the arrangement of studies, with “obs” indicating that the studies should be arranged by effect sizes. To organise by column, replace “obs” with the specific column in the data frame.
- The `ylim` argument sets the y-axis limits for the plot.
- The `ilab` argument is used to add extra columns of information to the forest plot beyond just the effect sizes. Here, we are adding sample size data for both the presence of smartphones and absence of smartphones groups.
- The `cbind()` function combines multiple columns side-by-side. In this case, the sample sizes of the groups (`n_p` and `n_a`) will appear side-by-side in the forest plot.
- The `ilab.xpos` argument specifies where the sample size columns appear horizontally on the plot. The negative values position the columns on the left side of the plot, just after the study names.
- The `slab` argument is used to label each effect size with its respective study.
- The `paste()` function creates the label by combining the “author” and “year_published” columns. The `sep` argument specifies the separator between the columns, which is set to “,” in this case. Hence, the label will be in the format “Author(s), Year_Published” (e.g., “Smith, 2020”).
- The `xlim` argument sets the x-axis limits for the plot.
- The `alim` argument sets the confidence interval limits, and the `steps` argument determines the number of intervals in the x-axis.
- The `efac` argument controls the size of the diamond shapes that represent the effect sizes in the plot.
- The `header` argument is set to `FALSE` to allow for manual specification of headers.
- The `xlab` argument specifies the confidence interval label for the forest plot, in this case, “Hedge’s g”.
- The `text()` function is used to manually include text within the plot, such as the “Author(s) Year” header and specific sample size column headers.
- The `x` and `y` arguments in the `text()` function adjust the position of the headers, with the `x` argument specifying the horizontal arrangement of the columns and the `y` argument specifying the vertical arrangement of the columns.
- The `font` argument adjusts the font size.
- The `dev.off()` function is used to close the graphics device and finalize the plot as a saved file.

```
### Forest Plot -----

# Start creating the forest plot itself
# Specify dataset
forest(
  mlmmetaresults,

  # Arrangement of studies
  order = "obs",

  # Add y-axis limits
  ylim = c(-3, 140),

  # Add sample size information for presence and absence of smartphones groups
```

```

# -3 for presence of smartphones (n_p)
# -2.55 for absence of smartphones (n_a)
ilab = cbind(n_p, n_a),
ilab.xpos = c(-3, -2.55),

# Label studies on the forest plot
slab = paste(author, year_published, sep = ", "),

# Add x-axis limits
xlim = c(-5, 3),

# Add confidence interval limits
# Adjust intervals based on the number of steps
alim = c(-2, 2),
steps = 9,

# Change size of effect size polygons
efac = 0.3,

# Show (TRUE) or hide (FALSE) default headers
# Hide when we want to manually specify our own headers
header = FALSE,

# Add label for confidence interval, in this case, "Hedge's g"
xlab = "Hedge's g"
)

# For the following lines of code,
# Use text function to manually include text within the plot

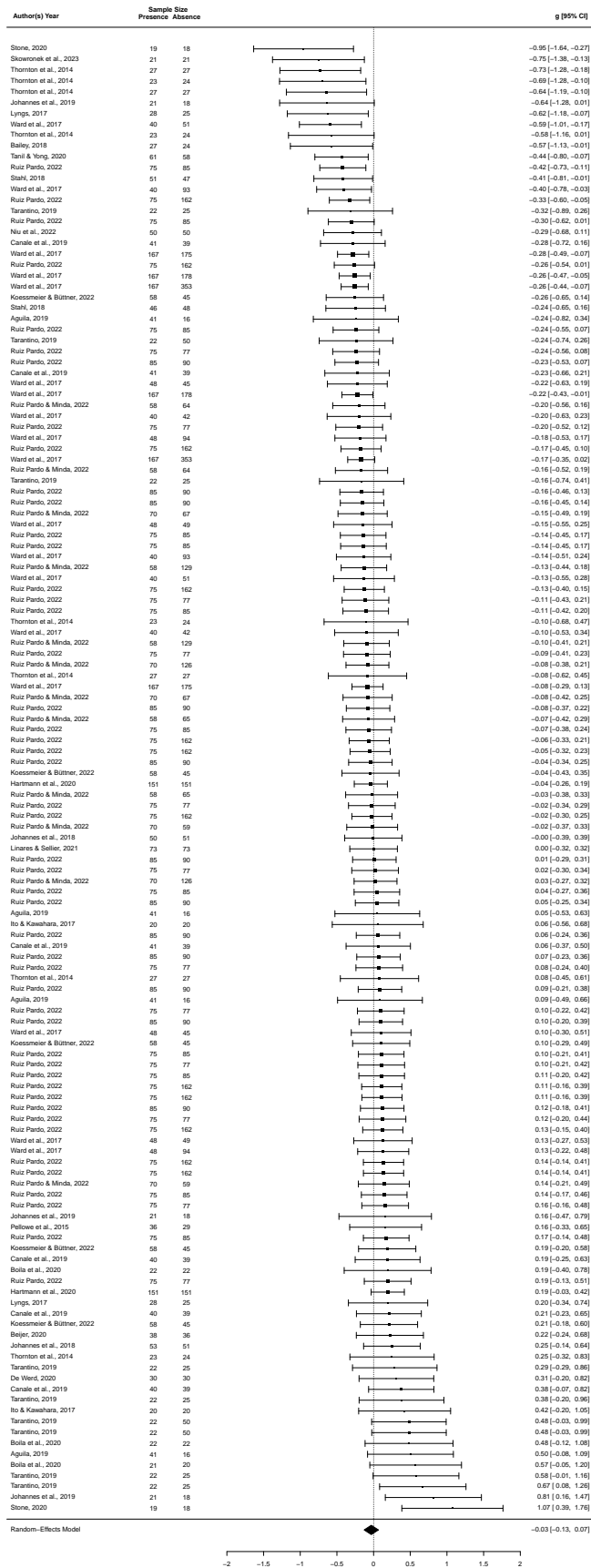
# Add "Author(s) Year" header
text(x = -4.6, y = 139, "Author(s) Year", font = 2)

# Add "Sample Size" header
text(x = -2.8, y = 139.6, "Sample Size", font = 2)

# Add specific sample size column headers, "Presence" and "Absence"
# x = -3 for for presence of smartphones
# x = -2.55 for absence of smartphones
# y = 139 for both
text(c(x = -3, x = -2.55), y = 139, c("Presence", "Absence"), font = 2)

# Add "g [95% CI]" header
text(x = 2.7, y = 139, "g [95% CI]", font = 2)

```



Optional: Saving the Forest Plot as a Separate File

Explanation of the Code

- To save the forest plot as a PDF file, the plotting code can be enclosed within `pdf()` and `dev.off()` functions:
- The `pdf()` function starts the graphics device driver to create PDF files, and the `file` argument specifies the name of the file.
- The `width` and `height` arguments adjust the dimensions of the PDF file.
- Following the `pdf()` function, the same code used to create the forest plot is repeated to generate the plot within the PDF file. The graphical output will be directed to the specified PDF file instead of the RStudio plotting window.
- The `dev.off()` function is used to close the graphics device and finalise the plot as a saved file.

```
### Saving the Forest Plot as a Separate File (Optional) -----
```

```
# Save the forest plot as a PDF file
# Name the pdf file of the forest plot
pdf(file = "mlmforestplot.pdf", width = 15, height = 40)

# Same forest plot code as above
forest(
  mlmmetaresults,
  order = "obs",
  ylim = c(-3, 140),
  ilab = cbind(n_p, n_a),
  ilab.xpos = c(-3, -2.55),
  slab = paste(author, year_published, sep = ", "),
  xlim = c(-5, 3),
  alim = c(-2, 2),
  steps = 9,
  efac = 0.3,
  header = FALSE,
  xlab = "Hedge's g"
)

text(x = -4.6, y = 139, "Author(s) Year", font = 2)

text(x = -2.8, y = 139.6, "Sample Size", font = 2)

text(c(x = -3, x = -2.55), y = 139, c("Presence", "Absence"), font = 2)

text(x = 2.7, y = 139, "g [95% CI]", font = 2)

# Close the forest plot and finalise it as a saved file
dev.off()

## pdf
## 2
```

Tests for Publication Bias

This section performs tests for publication bias, including a funnel plot and Egger's test.

For multilevel meta-analysis, we do not recommend conducting a rank correlation test as it is prone to Type 1 error.

Researchers may refer to this article by Fernández-Castilla et al. (2019) on a detailed discussion of the limitations of rank correlation tests in multilevel meta-analysis, as well as an overview on other publication bias tests: <https://doi.org/10.1080/00220973.2019.1582470>:

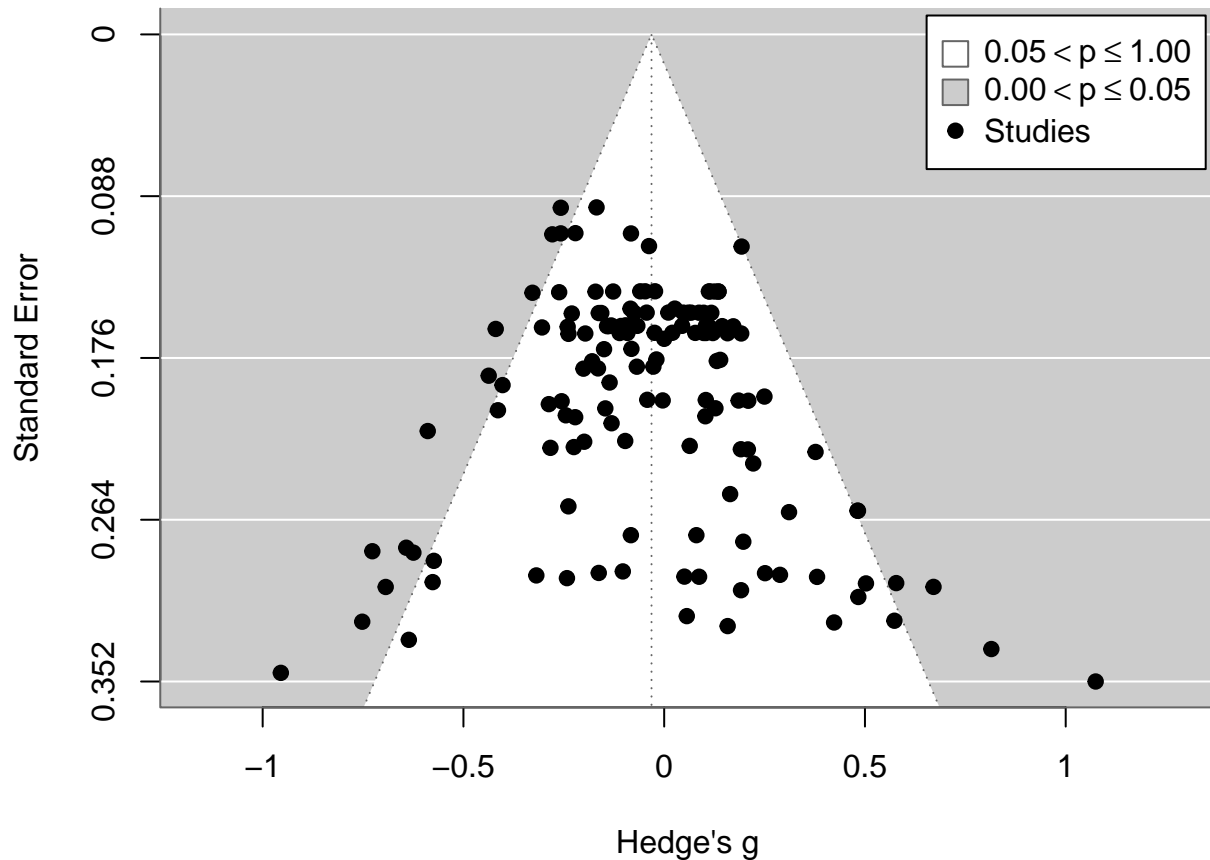
Funnel Plot

The funnel plot visually represents the distribution of effect sizes and their standard errors, allowing for the identification of potential publication bias. Saving the funnel plot as a PDF file allows for easy sharing and presentation, and allows adjustment to the plot's dimensions.

Explanation of the Code

- The `pdf()` function starts the graphics device driver to create PDF files, and the `file` argument specifies the name of the file.
- The `width` and `height` arguments adjust the dimensions of the PDF file.
- The `par()` function is used to adjust the margins of the funnel plot, with the `mar` argument specifying the bottom, left, top, and right margins.
- The `funnel()` function is used to create the funnel plot, and the `mlmetaresults` object contains the results of the meta-analysis.
- The `legend` argument specifies whether to include a legend in the plot. `TRUE` indicates that a legend should be included, `FALSE` indicates that it should not.
- The `xlab` argument specifies the confidence interval label for the funnel plot, in this case, "Hedge's g".
- The `dev.off()` function is used to close the graphics device and finalize the plot as a saved file.

```
### Tests for Publication Bias -----  
  
# Funnel Plot #  
  
# Adjust margins of the funnel plot  
# Set the bottom, left, top, and right margins  
par(mar = c(4, 4, 0.3, 1))  
# Create the funnel plot  
funnel(mlmetaresults, legend = TRUE, xlab = "Hedge's g")
```



Optional: Saving the Funnel Plot as a Separate File

Explanation of the Code

- To save the funnel plot as a PDF file, the plotting code can be enclosed within `pdf()` and `dev.off()` functions:
- The `pdf()` function starts the graphics device driver to create PDF files, and the `file` function specifies the name of the file.
- The `width` and `height` arguments adjust the dimensions of the PDF file.
- Following the `pdf()` function, the same code used to create the funnel plot is repeated to generate the plot within the PDF file. The graphical output will be directed to the specified PDF file instead of the RStudio plotting window.
- The `dev.off()` function is used to close the graphics device and finalize the plot as a saved file.

Saving the Funnel Plot as a Separate File (Optional) -----

Funnel Plot

Save the funnel plot as a PDF file

Name the pdf file of the funnel plot

Adjust the width and height of the pdf file

`pdf(file = "mlmfunnelplot.pdf", width = 8, height = 5)`

Same funnel plot code as above

```

par(mar = c(4, 4, 0.3, 1))
funnel(mlmmetaresults, legend = TRUE, xlab = "Hedge's g")

# Close the funnel plot and finalise it as a saved file
dev.off()

## pdf
## 2

```

Egger's Test

The Egger's test is a statistical test that quantifies the degree of asymmetry in the funnel plot, providing a more formal assessment of publication bias.

Explanation of the Code

- The `multilevelmeta$sei_corrected` variable is created to store the corrected standard error for each effect size, calculated using the formula $\sqrt{\frac{(n_p + n_a)}{(n_p \cdot n_a)}}$. Using Egger's test (unadjusted) on SMDs results in inflated Type 1 error as SMD and SE are not independent. Hence, use corrected formula for SE.
- The `lmer()` function is used to fit a linear mixed-effects model, where the effect size weighted by the corrected standard error is predicted by the intercept and the inverse of the corrected standard error. `metafor::rma.mv` does not have a `weights` argument, and `metafor::regtest` does not support `rma.mv` objects. For three (or more) level meta-analysis, use `lmerTest::lmer` instead.
- The `I(yi / sei_corrected)` expression indicates that the effect size (`yi`) is divided by the corrected standard error (`sei_corrected`), which is used to weight the effect sizes in the model.
- The `I(1 / sei_corrected)` expression indicates that the inverse of the corrected standard error is included in the model as a predictor.
- The `1 | lab_id` expression indicates that random intercepts are included for each lab, accounting for the nested structure of the data.
- The `data` argument specifies the dataset to be used for the analysis.
- The `summary()` function provides the results of the Egger's test, including the slope estimate and its significance.

```

### Tests for Publication Bias -----

# Egger's Test #

# Calculate standard error (SE)
multilevelmeta$sei_corrected = with(
  multilevelmeta,
  sqrt((n_p + n_a) / (n_p * n_a)))
lmer(
  # g weighted by SE is predicted by intercept and inverse SE
  # with random intercept by sample
  I(yi / sei_corrected) ~ 1 + I(1 / sei_corrected) + (1 | lab_id),
  data = multilevelmeta
) |>
  # Estimate of interest is the intercept
  summary(correlation = FALSE)

## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]

```

```
## Formula: I(yi/sei_corrected) ~ 1 + I(1/sei_corrected) + (1 | lab_id)
## Data: multilevelmeta
##
## REML criterion at convergence: 444.7
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.6417 -0.7225  0.0334  0.7713  2.6991
##
## Random effects:
## Groups   Name      Variance Std.Dev.
## lab_id   (Intercept) 0.445    0.667
## Residual                1.335    1.155
## Number of obs: 136, groups: lab_id, 22
##
## Fixed effects:
##              Estimate Std. Error    df t value Pr(>|t|)
## (Intercept)      0.6753     0.4536 61.7763    1.49   0.142
## I(1/sei_corrected) -0.1803     0.0874 79.4062   -2.06   0.042 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Moderation Analysis

This section performs moderation analysis to explore the influence of categorical and continuous moderators on the effect sizes.

For continuous moderators, use meta-regression. Specifically, the `mods` function specifies the moderator variable (for continuous moderators).

For categorical moderators, use subgroup analysis. Specifically, the `subset` argument is used to specify the subset of data for categorical moderators, allowing for separate analyses for each category.

Explanation of the Code

- The `rma.mv()` function is used to perform the moderation analysis.
- The `yi` and `vi` arguments specify the effect size estimates and their corresponding sampling variances, respectively.
- The `random` argument specifies the random effects structure, where `~ 1 | lab_id / ID` indicates that random intercepts are included for both the lab and individual studies, accounting for the nested structure of the data.
- The `mods` argument specifies the moderator variable (for continuous moderators).
- The `subset` argument is used to specify the subset of data for categorical moderators, allowing for separate analyses for each category.
- The `method` argument specifies the method used to estimate heterogeneity, in this case, “REML” (Restricted Maximum Likelihood).
- The `data` argument specifies the dataset to be used for the analysis.
- The `summary()` function is used to display the results of the meta-analysis, including the overall effect size estimate and its confidence interval.

Note that if convergence issues arise, the `control` argument can be used to address it. Researchers may also refer more to the `metafor` package documentation for more information on how to address convergence issues.

```

### Moderation Analysis -----

# Categorical variable (i.e., publication type)
rma.mv(
  yi = yi,
  V = vi,
  random = ~ 1 | lab_id / ID,
  # Specify categorical moderator (i.e., Journal Article)
  subset = (publication == "Journal article"),
  data = multilevelmeta,
  # To address convergence issues (if it exists)
  control = list(rel.tol=1e-8)
)

##
## Multivariate Meta-Analysis Model (k = 66; method: REML)
##
## Variance Components:
##
##      estim      sqrt  nlvls  fixed      factor
## sigma^2.1 0.0270  0.1644    14     no    lab_id
## sigma^2.2 0.0000  0.0000    66     no  lab_id/ID
##
## Test for Heterogeneity:
## Q(df = 65) = 100.7856, p-val = 0.0029
##
## Model Results:
##
## estimate      se      zval      pval      ci.lb      ci.ub
## -0.0476  0.0571  -0.8328  0.4050  -0.1596  0.0644
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

rma.mv(
  yi = yi,
  V = vi,
  random = ~ 1 | lab_id / ID,
  # Specify categorical moderator (i.e., Thesis/dissertation)
  subset = (publication == "Thesis/dissertation"),
  data = multilevelmeta
)

##
## Multivariate Meta-Analysis Model (k = 68; method: REML)
##
## Variance Components:
##
##      estim      sqrt  nlvls  fixed      factor
## sigma^2.1 0.0321  0.1792     8     no    lab_id
## sigma^2.2 0.0029  0.0543    68     no  lab_id/ID
##
## Test for Heterogeneity:
## Q(df = 67) = 100.9032, p-val = 0.0047
##

```

```

## Model Results:
##
## estimate      se      zval      pval      ci.lb      ci.ub
## 0.0131 0.0872 0.1506 0.8803 -0.1578 0.1841
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

rma.mv(
  yi = yi,
  V = vi,
  random = ~ 1 | lab_id / ID,
  # Specify categorical moderator (i.e., Conference)
  subset = (publication == "Conference"),
  data = multilevelmeta
)

## Warning: Single-level factor(s) found in 'random' argument. Corresponding
## 'sigma2' value(s) fixed to 0.

##
## Multivariate Meta-Analysis Model (k = 2; method: REML)
##
## Variance Components:
##
##          estim      sqrt  nlvls  fixed      factor
## sigma^2.1 0.0000 0.0000      1    yes      lab_id
## sigma^2.2 0.2598 0.5097      2     no  lab_id/ID
##
## Test for Heterogeneity:
## Q(df = 1) = 4.3422, p-val = 0.0372
##
## Model Results:
##
## estimate      se      zval      pval      ci.lb      ci.ub
## -0.2118 0.4108 -0.5155 0.6062 -1.0170 0.5934
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# Continuous variable (i.e., female proportion)
rma.mv(
  yi = yi,
  V = vi,
  random = ~ 1 | lab_id / ID,
  # Specify continuous moderator (i.e., female proportion)
  mods = ~ female_proportion,
  method = "REML",
  data = multilevelmeta
) |>
  summary()

##
## Multivariate Meta-Analysis Model (k = 136; method: REML)
##
##      logLik  Deviance      AIC      BIC      AICc

```

```
##    5.4145  -10.8291  -2.8291    8.7623  -2.5190
##
## Variance Components:
##
##          estim      sqrt  nlvls  fixed      factor
## sigma^2.1  0.0246  0.1567    22     no      lab_id
## sigma^2.2  0.0000  0.0000   136     no  lab_id/ID
##
## Test for Residual Heterogeneity:
## QE(df = 134) = 200.0850, p-val = 0.0002
##
## Test of Moderators (coefficient 2):
## QM(df = 1) = 5.1951, p-val = 0.0227
##
## Model Results:
##
##              estimate      se      zval      pval      ci.lb      ci.ub
## intrcpt             -0.3041  0.1286  -2.3650  0.0180  -0.5562  -0.0521  *
## female_proportion    0.4104  0.1801   2.2793  0.0227   0.0575   0.7634  *
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Forest Plot of Moderators

This section generates a forest plot that includes moderators to visually represent the effect sizes and confidence intervals for each study included in the meta-analysis.

The plot is saved as a PDF file for easy sharing and presentation.

The forest plot includes the following features:

- Arrangement of studies by effect sizes and types of moderators
- Sample size information for both presence of smartphones and absence of smartphones groups
- Custom headers for the plot
- Custom labels for the studies
- Summary effect sizes for each moderator

Explanation of the Code

- The `pdf()` function starts the graphics device driver to create PDF files, and the `file` argument specifies the name of the file.
- The `width` and `height` arguments adjust the dimensions of the PDF file.
- The `forest()` function is used to create the forest plot, and the `mlmmetaresults` object contains the results of the meta-analysis.
- The `rows` argument specifies the arrangement of studies by publication type, and in ascending order of effect sizes per publication type.
- The `ylim` argument sets the y-axis limits for the plot.
- The `ilab` argument is used to add extra columns of information to the forest plot beyond just the effect sizes. Here, we are adding sample size data for both the presence of smartphones and absence of smartphones groups.

- The `cbind()` function combines multiple columns side-by-side. In this case, the sample sizes of the groups (`n_p` and `n_a`) will appear side-by-side in the forest plot.
- The `ilab.xpos` argument specifies where the sample size columns appear horizontally on the plot. The negative values position the columns on the left side of the plot, just after the study names.
- The `slab` argument is used to label each effect size with its respective study.
- The `paste()` function creates the label by combining the “author” and “year_published” columns. The `sep` argument specifies the separator between the columns, which is set to “,” in this case. Hence, the label will be in the format “Author(s), Year” (e.g., “Smith, 2020”).
- The `xlim` argument sets the x-axis limits for the plot.
- The `alim` argument sets the confidence interval limits, and the `steps` argument determines the number of intervals in the x-axis.
- The `efac` argument controls the size of the diamond shapes that represent the effect sizes in the plot.
- The `header` argument is set to `FALSE` to allow for manual specification of headers.
- The `xlab` argument specifies the label for the confidence interval, in this case, “Hedge’s g”.
- The `text()` function is used to manually include text within the plot, such as the “Author(s) Year” header and specific sample size column headers.
- The `x` and `y` arguments in the `text()` function adjust the position of the headers, with `x` argument specifying the horizontal arrangement of the columns and the `y` specifying the vertical arrangement of the columns.
- The `pos` argument specifies the position of the text relative to the specified coordinates.
- The `font` argument adjusts the font size.
- The `rma.mv()` function is used to perform moderation analysis for each publication type, with the `subset` argument specifying the subset of data for each category.
- The `rest.j`, `rest.t`, and `rest.c` variables store the results of the moderation analysis for journal articles, thesis/dissertations, and conference papers, respectively.
- The `random` argument specifies the random effects structure, where `~ 1 | lab_id / ID` indicates that random intercepts are included for both the lab and individual studies, accounting for the nested structure of the data.
- The `subset` argument is used to specify the subset of data for categorical moderators, allowing for separate analyses for each category.
- The `addpoly()` function is used to add summary effect sizes for each of the moderators, with the `row` argument specifying the position of the summary in the plot.
- The `dev.off()` function is used to close the graphics device and finalize the plot as a saved file.

```
### Forest Plot of Moderators -----
```

```
# Start creating the forest plot itself
# Specify dataset
forest(
  mlmmetaresults,
  # Manually arrange effect sizes by publication type
  # - Journal article: Rows 143 to 79
  # - Thesis/Dissertations: Rows 75 to 7
  # - Conference: Rows 3 to 2
  # The arrangement must consider spacing and must end at row 2
```

```

rows = c(143:79, 75:7, 3:2),

# Add y-axis limits
ylim = c(-3, 147),

# Add sample size information for presence and absence of smartphones groups
# -4.2 for presence of smartphones (n_p)
# -3.6 for absence of smartphones (n_a)
ilab = cbind(n_p, n_a),
ilab.xpos = c(-4.2, -3.6),

# Label studies on the forest plot
slab = paste(author, year_published, sep = ", "),

# Add x-axis limits
xlim = c(-7, 4),

# Add confidence interval limits
# Adjust intervals based on the number of steps
alim = c(-1.5, 1.5),
steps = 11,

# Change size of effect size polygons
efac = 0.3,

# Remove headers (if any), for manual input
header = FALSE,

# Add label for confidence interval, in this case, "Hedge's g"
xlab = "Hedge's g"
)

# For the following lines of code,
# Use text function to manually include text within the plot

# Add text labels for moderator (type of publication)
# Labels for different publication type (Moderator Analysis)
# - "Journal article" at y = 7
# - "Thesis/Dissertations" at y = 15
# - "Non-Conference" at y = 21
text(
  x = -7,
  y = c(4, 76, 144),
  pos = 4,
  c("Conference", "Thesis/dissertation", "Journal article"),
  font = 2
)

# Moderation analysis
res.j = rma.mv(
  yi,
  vi,
  random = ~ 1 | lab_id / ID,

```

```

subset = (publication == "Journal article"),
data = multilevelmeta,
# To address convergence issues (if it exists)
control = list(rel.tol=1e-8)
)
res.t = rma.mv(
  yi,
  vi,
  random = ~ 1 | lab_id / ID,
  subset = (publication == "Thesis/dissertation"),
  data = multilevelmeta
)
res.c = rma.mv(
  yi,
  vi,
  random = ~ 1 | lab_id / ID,
  subset = (publication == "Conference"),
  data = multilevelmeta
)

```

```

## Warning: Single-level factor(s) found in 'random' argument. Corresponding
## 'sigma2' value(s) fixed to 0.

```

```

# Add summary effect sizes for each of the moderators
addpoly(res.c, row = 1) # summary effect for "Conference" group
addpoly(res.t, row = 6) # summary effect for "Thesis/Dissertation" group
addpoly(res.j, row = 78) # summary effect for "Journal article" group

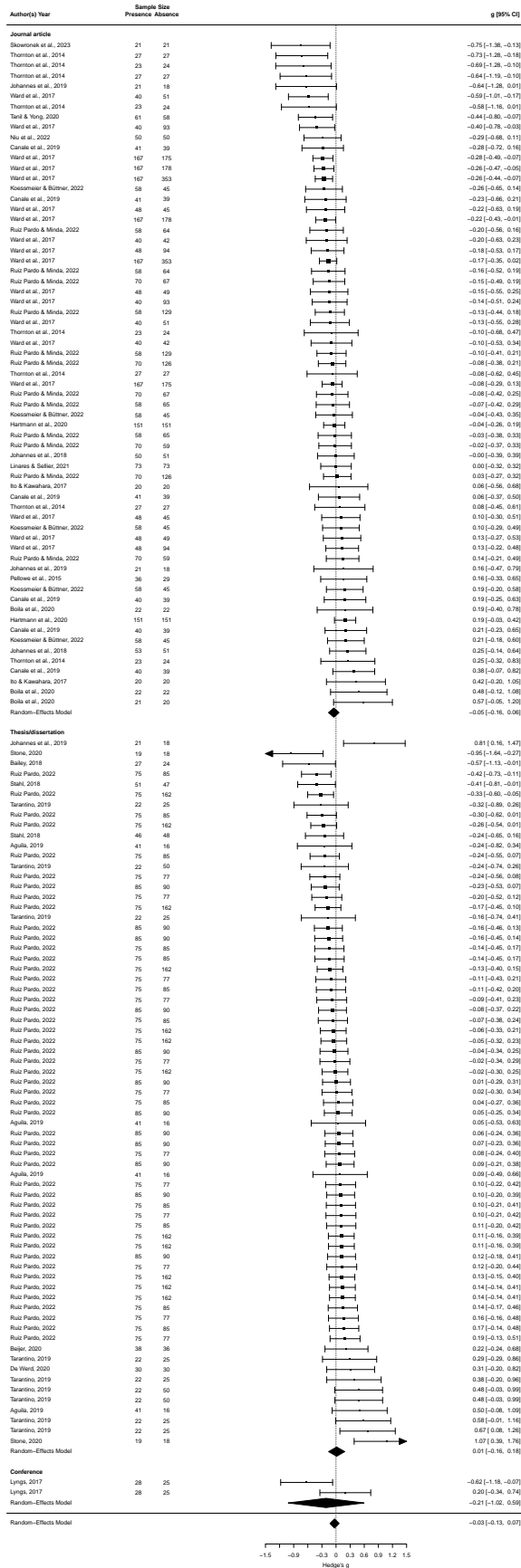
# Add "Author(s) Year" header
text(x = -6.5, y = 146, "Author(s) Year", font = 2)

# Add "Sample Size" header
text(x = -3.9, y = 146.7, "Sample Size", font = 2)

# Add specific sample size column headers, "Presence" and "Absence"
# x = -4.2 for for presence of smartphones
# x = -3.6 for absence of smartphones
# y = 146 for both
text(c(x = -4.2, x = -3.6), y = 146, c("Presence", "Absence"), font = 2)

# Add "g [95% CI]" header
text(x = 3.6, y = 146, "g [95% CI]", font = 2)

```



Optional: Saving the Forest Plot of Moderators as a Separate File

Explanation of the Code

- To save the forest plot of moderators as a PDF file, the plotting code can be enclosed within `pdf()` and `dev.off()` functions:
- The `pdf()` function starts the graphics device driver to create PDF files, and the `file` argument specifies the name of the file.
- The `width` and `height` arguments adjust the dimensions of the PDF file.
- Following the `pdf()` function, the same code used to create the forest plot of moderators is repeated to generate the plot within the PDF file. The graphical output will be directed to the specified PDF file instead of the RStudio plotting window.
- The `dev.off()` function is used to close the graphics device and finalise the plot as a saved file.

```
### Forest Plot of Moderators -----

# Save the forest plot as a PDF file
# Name the pdf file of the forest plot
# Adjust the width and height of the pdf file
pdf(file = "mlmforestplotwithmoderators.pdf", width = 15, height = 45)

# Same forest plot code as above
forest(
  mlmmetaresults,
  rows = c(143:79, 75:7, 3:2),
  ylim = c(-3, 147),
  ilab = cbind(n_p, n_a),
  ilab.xpos = c(-4.2, -3.6),
  slab = paste(author, year_published, sep = ", "),
  xlim = c(-7, 4),
  alim = c(-1.5, 1.5),
  steps = 11,
  efac = 0.3,
  header = FALSE,
  xlab = "Hedge's g"
)

text(
  x = -7,
  y = c(4, 76, 144),
  pos = 4,
  c("Conference", "Thesis/dissertation", "Journal article"),
  font = 2
)

res.j = rma.mv(
  yi,
  vi,
  random = ~ 1 | lab_id / ID,
  subset = (publication == "Journal article"),
  data = multilevelmeta,
  control = list(rel.tol=1e-8)
)
```

```

res.t = rma.mv(
  yi,
  vi,
  random = ~ 1 | lab_id / ID,
  subset = (publication == "Thesis/dissertation"),
  data = multilevelmeta
)
res.c = rma.mv(
  yi,
  vi,
  random = ~ 1 | lab_id / ID,
  subset = (publication == "Conference"),
  data = multilevelmeta
)

```

```

## Warning: Single-level factor(s) found in 'random' argument. Corresponding
## 'sigma2' value(s) fixed to 0.

```

```

addpoly(res.c, row = 1)
addpoly(res.t, row = 6)
addpoly(res.j, row = 78)

text(x = -6.5, y = 146, "Author(s) Year", font = 2)

text(x = -3.9, y = 146.7, "Sample Size", font = 2)

text(c(x = -4.2, x = -3.6), y = 146, c("Presence", "Absence"), font = 2)

text(x = 3.6, y = 146, "g [95% CI]", font = 2)

# Close the forest plot and finalise it as a saved file
dev.off()

## pdf
## 2

```

End of Code