Traditional 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 traditional meta-analysis using the `metafor` package in R.

The analysis is based on data from Majeed et al. (2021) and focuses on the relationship between dyslexia and creativity. The original paper can be found here: https://doi.org/10.1002/dys.1677.

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` package is not already installed, use the `install.packages()` function to install it.

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` package.
- 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 "DYSCRE.csv", which contains the data drawn from Majeed et al. (2021).

```
### 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)
```

```
# Display settings (to disable scientific notation)
options(scipen = 9999, digits = 4)

# Read in data drawn from Majeed et al. 2021
tradmeta_raw = read.csv("DYSCRE.csv")
```

Prepare Data

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

The `tradmeta_raw` data frame does not include pre-computed effect sizes for each study. Therefore, the `escalc()` function from `metafor` package is used to calculate them. For more information on the `escalc()` function, refer to ?metafor::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 Hedges' 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 (yi) and their corresponding sampling variances (vi) for each study.
- The `tradmeta$Creativity.Measure_type` variable is converted to a factor with specified levels to ensure that the creativity measure types are ordered correctly in the forest plot.
- The `c()` function combines the creativity measure types into a character vector.
- The `tradmeta_raw` data frame is then sorted by the type of creativity measure 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
tradmeta = escalc(
  # Type of effect size measure
  measure = "SMD",

  # Columns for sample size of each group
  n1i = n_dys,
  n2i = n_control,

  # Columns for means of each group
  m1i = Mean_CRE_dys,
  m2i = Mean_CRE_control,
```

```r
  # Columns for standard deviation of each group
  sd1i = SD_CRE_dys,
  sd2i = SD_CRE_control,

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

# Convert Creativity.Measure_type to a factor with specified levels
tradmeta$Creativity.Measure_type = factor(
  tradmeta$Creativity.Measure_type,
  levels = c("Verbal", "Mixed", "Non-verbal")
)

# Order the data frame by Creativity.Measure_type and effect sizes (yi)
tradmeta = tradmeta[order(tradmeta$Creativity.Measure_type, tradmeta$yi), ]
```

Computing the Overall Effect Size

This section estimates the overall effect size using the `rma()` function from the `metafor` package, based on the computed effect sizes (yi) and their corresponding sampling variances (vi).

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

The results are summarized to provide detailed information about the meta-analysis.

Explanation of the Code

- The `tradmeta` data frame contains the computed effect sizes (yi) and their corresponding sampling variances (vi).
- The `yi` and `vi` arguments specify the effect size estimates and their corresponding sampling variances, respectively.
- The `rma()` function is used to perform a random-effects meta-analysis, which accounts for the variability between studies.
- The `method` argument specifies the method used to estimate heterogeneity, in this case, "REML" (Restricted Maximum Likelihood).
- The `summary()` function is used to provide detailed results of the meta-analysis, including the overall effect size estimate, confidence intervals, and heterogeneity statistics.

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

tradmetaresults = rma(
  # Effect size estimates
  yi = yi,
  # Sampling variances
  vi = vi,
  # Specify method to estimate heterogeneity
  method = "REML",
  # Specify where to get the data from
  data = tradmeta
)
```

```r
# summary function used to provide detailed results of the meta-analysis
summary(tradmetaresults)
```

```
##
## Random-Effects Model (k = 13; tau^2 estimator: REML)
##
##   logLik  deviance       AIC       BIC      AICc
## -11.0151   22.0302   26.0302   27.0000   27.3635
##
## tau^2 (estimated amount of total heterogeneity): 0.2443 (SE = 0.1373)
## tau (square root of estimated tau^2 value):      0.4943
## I^2 (total heterogeneity / total variability):   77.16%
## H^2 (total variability / sampling variability):  4.38
##
## Test for Heterogeneity:
## Q(df = 12) = 42.9777, p-val < .0001
##
## Model Results:
##
## estimate      se    zval    pval    ci.lb   ci.ub
##   0.1106  0.1613  0.6856  0.4930  -0.2056  0.4268
##
## ---
## 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 dyslexia and control groups
- Custom headers for the plot
- Custom labels for the studies

After running this code, the forest plot will appear as shown on page 7 of this handbook.

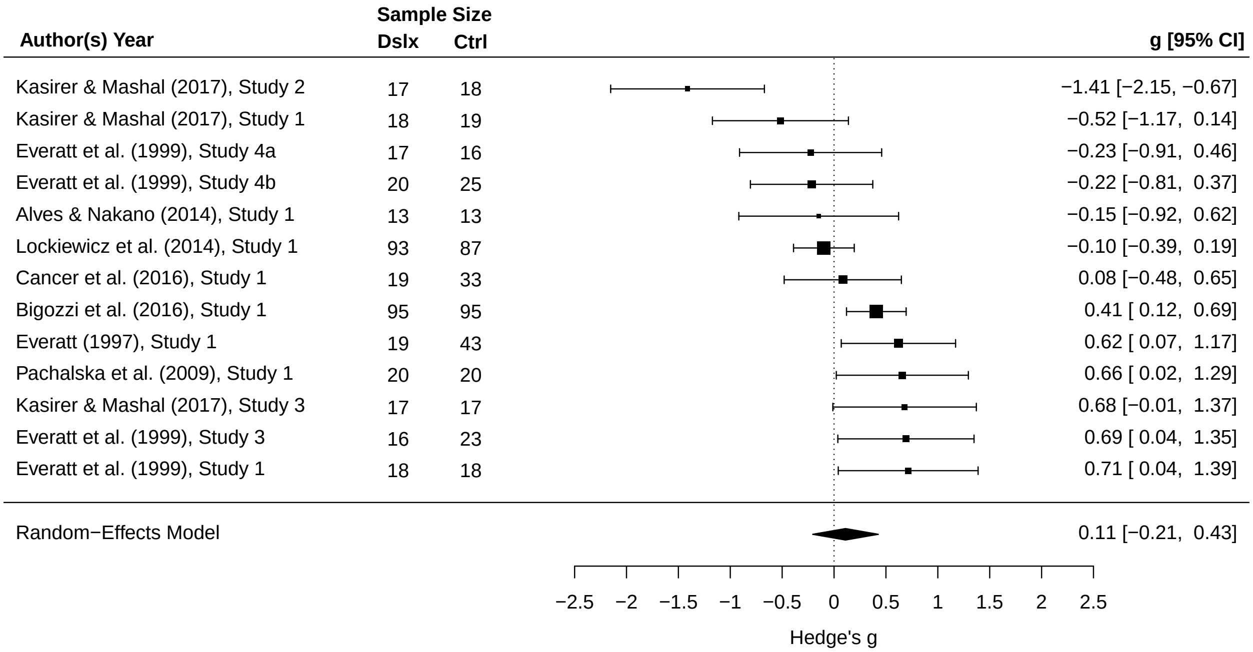
Explanation of the Code

- The `forest()` function is used to create the forest plot, and the `tradmetaresults` 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 dyslexia and control groups.

- The `cbind()` function combines multiple columns side-by-side. In this case, the sample sizes of the groups (dys and control) 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 labels.
- The `slab` argument is used to label each effect size with its respective study.
- The `paste()` function creates the label by combining the Paper (i.e., authors of the paper) and Study columns. Since the Study column in the dataset only contains numbers, we added the word "Study" before the number to make it clearer. The `sep` argument specifies the separator between paper and the study label, which is set to "," in this case. Hence, the label will be in the format "Paper, Study X" (e.g., "Smith et al., Study 1").
- 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 `header` argument is set to FALSE to allow for manual specification of headers. Otherwise, the default headers will be used if the `header` argument is set to TRUE.
- 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 `c()` function inside the `text()` function creates a vector of x-positions, where the labels "Dslx" and "Ctrl" will be placed. The first x-value (-4.2) determines the horizontal position of the "Dslx" label, and the second x-value (-3.5) positions the "Ctrl" label.

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

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

  # Arrangement of studies
  order = "obs",

  # Add y-axis limits
  ylim = c(-2, 16),

  # Add sample size information for dyslexia (n_dys) and control (n_control) group
  # -4.2 for Dslx (Dyslexia Group)
  # -3.5 for Ctrl (Control Group)
  ilab = cbind(n_dys, n_control),
  ilab.xpos = c(-4.2, -3.5),

  # Label studies on the forest plot
  slab = paste(Paper, paste("Study", Study), sep = ", "),

  # Add x-axis limits
```

```r
  xlim = c(-8, 4),

  # Add confidence interval limits
  # Adjust intervals based on the number of steps
  alim = c(-2.5, 2.5),
  steps = 11,

  # 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 = -7.2, y = 14.5, "Author(s) Year", font = 2)

# Add "Sample Size" header
text(x = -3.85, y = 15.3, "Sample Size", font = 2)

# Add specific sample size column headers for dyslexia and control groups
# x = -4.2 for Dslx (Dyslexia Group)
# x = -3.5 for Ctrl (Control Group)
# y values indicate the vertical arrangement of the columns
# y = 14.5 for both
text(c(x = -4.2, x = -3.5), y = 14.5, c("Dslx", "Ctrl"), font = 2)

# Add "g [95% CI]" header
text(x = 3.5, y = 14.5, "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.

```r
### 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 = "tradforestplot.pdf", width = 11, height = 6.5)

# Same forest plot code as above
forest(
  tradmetaresults,
  order = "obs",
  ylim = c(-2, 16),
  ilab = cbind(n_dys, n_control),
  ilab.xpos = c(-4.2, -3.5),
  slab = paste(Paper, paste("Study", Study), sep = ", "),
  xlim = c(-8, 4),
  alim = c(-2.5, 2.5),
  steps = 11,
  header = FALSE,
```

```r
  xlab = "Hedge's g"
)

text(x = -7.2, y = 14.5, "Author(s) Year", font = 2)
text(x = -3.85, y = 15.3, "Sample Size", font = 2)
text(c(x = -4.2, x = -3.5), y = 14.5, c("Dslx", "Ctrl"), font = 2)
text(x = 3.5, y = 14.5, "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, rank correlation test and Egger's test.

Funnel Plot

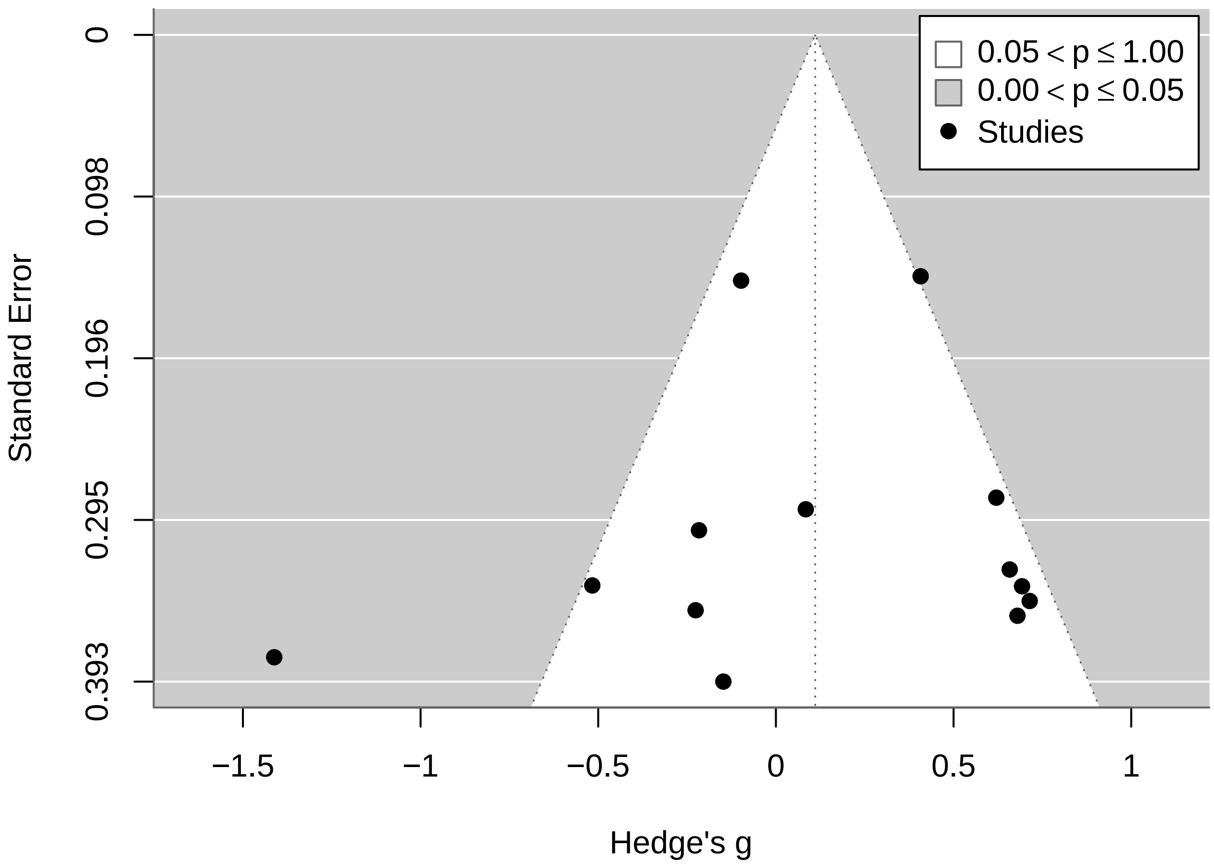
The funnel plot visually represents the distribution of effect sizes and their standard errors, allowing for the identification of potential publication bias.

After running this code, the funnel plot will appear as shown on page 9 of this handbook.

Explanation of the Code

- 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 `tradmetaresults` 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"

```r
### 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(tradmetaresults, legend = TRUE, xlab = "Hedge's g")
```



Optional: Saving the Funnel Plot as a Separate File

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

- 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.

```r
### 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 = "tradfunnelplot.pdf", width = 7, height = 3.5)
```

```
# Same funnel plot code as above
par(mar = c(4, 4, 0.3, 1))
funnel(tradmetaresults, legend = TRUE, xlab = "Hedge's g")

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

```
## pdf
##   2
```

Rank Correlation Test

The rank correlation test serves as a complementary method to the funnel plot in assessing publication bias by examining the correlation between effect sizes and their standard errors.

Explanation of the Code

- The `ranktest()` function computes the Kendall tau value, which indicates the strength and direction of the association between the ranks of effect sizes and their standard errors.
- The `tradmetaresults` object contains the results of the meta-analysis, and the `ranktest()` function returns the Kendall tau value and its significance level.

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

# Rank Correlation Test #
ranktest(tradmetaresults)
```

```
##
## Rank Correlation Test for Funnel Plot Asymmetry
##
## Kendall's tau = -0.2308, p = 0.3062
```

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 `tradmeta$sei_corrected` variable is created to store the corrected standard error for each effect size, calculated using the formula $\sqrt{\frac{(n_{dys}+n_{control})}{(n_{dys}\cdot n_{control})}}$. Using Egger' test (unadjusted) on SMDs results in inflated type 1 error as SMD and SE are not independent. Hence, use the corrected formula for SE.
- The `rma()` function is used to perform Egger's test, with the `yi` and `vi` arguments specifying the effect size estimates and their corresponding sampling variances.
- The `mods` argument specifies the moderator variable, which is the corrected standard error in this case.
- The `weights` argument specifies the weight for each effect size, which is the inverse of the corrected standard error squared.
- 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.

```r
### Tests for Publication Bias --------------

# Egger's Test #

# Calculate standard error (SE)
tradmeta$sei_corrected = with(
  tradmeta,
  sqrt((n_dys + n_control) / (n_dys * n_control))
)
rma(
  # Effect size estimates
  yi = yi,
  # Sampling variance
  vi = vi,
  # Indicate moderator which is SE/sei_corrected
  mods = ~ sei_corrected,
  # Indicate weight which is inverse SE^2
  weights = 1 / sei_corrected^2,
  # Specify dataset
  data = tradmeta
) |>
  # Estimate of interest is the intercept
  summary()
```

```
##
## Mixed-Effects Model (k = 13; tau^2 estimator: REML)
##
##   logLik  deviance       AIC       BIC      AICc
## -10.5945   21.1890   27.1890   28.3827   30.6176
##
## tau^2 (estimated amount of residual heterogeneity):     0.2800 (SE = 0.1613)
## tau (square root of estimated tau^2 value):             0.5291
## I^2 (residual heterogeneity / unaccounted variability): 77.97%
## H^2 (unaccounted variability / sampling variability):   4.54
## R^2 (amount of heterogeneity accounted for):            0.00%
##
## Test for Residual Heterogeneity:
## QE(df = 11) = 42.5985, p-val < .0001
##
## Test of Moderators (coefficient 2):
## QM(df = 1) = 0.0673, p-val = 0.7954
##
## Model Results:
##
##                  estimate      se     zval    pval    ci.lb   ci.ub
## intrcpt            0.2830  0.7048   0.4016  0.6880  -1.0984  1.6645
## sei_corrected     -0.6105  2.3539  -0.2594  0.7954  -5.2240  4.0030
##
## ---
## 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. Specficially, the `mods` argument 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()` 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 `mods` argument specifies the moderator variable (for continuous moderators).
- The `method` argument specifies the method used to estimate heterogeneity, in this case, "REML" (Restricted Maximum Likelihood).
- The `subset` argument is used to specify the subset of data for categorical moderators, allowing for separate analyses for each category.
- The `data` argument specifies the dataset to be used for the analysis.

```r
### Moderation Analysis --------------

# Continuous variable (i.e., female proportion)
rma(
  yi = yi,
  vi = vi,
  # Specify continuous moderator (i.e., sex)
  mods = ~ Proportion.of.female,
  method = "REML",
  data = tradmeta
)
```

```
## Warning: 2 studies with NAs omitted from model fitting.
```

```
##
## Mixed-Effects Model (k = 11; tau^2 estimator: REML)
##
## tau^2 (estimated amount of residual heterogeneity):     0.3157 (SE = 0.1942)
## tau (square root of estimated tau^2 value):             0.5619
## I^2 (residual heterogeneity / unaccounted variability): 79.02%
## H^2 (unaccounted variability / sampling variability):   4.77
## R^2 (amount of heterogeneity accounted for):            0.00%
##
## Test for Residual Heterogeneity:
## QE(df = 9) = 40.1340, p-val < .0001
##
## Test of Moderators (coefficient 2):
## QM(df = 1) = 0.8519, p-val = 0.3560
##
## Model Results:
##
```

```
##                        estimate      se     zval    pval    ci.lb   ci.ub
## intrcpt                -0.6211  0.7645  -0.8124  0.4165  -2.1196  0.8773
## Proportion.of.female    1.7058  1.8481   0.9230  0.3560  -1.9163  5.3279
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Categorical variable (i.e., type of creativity measure)
rma(
  yi = yi,
  vi = vi,
  # Specify categorical moderator (i.e., verbal)
  subset = (Creativity.Measure_type == "Verbal"),
  method = "REML",
  data = tradmeta
)
```

```
##
## Random-Effects Model (k = 3; tau^2 estimator: REML)
##
## tau^2 (estimated amount of total heterogeneity): 0.9624 (SE = 1.0886)
## tau (square root of estimated tau^2 value):      0.9810
## I^2 (total heterogeneity / total variability):   88.45%
## H^2 (total variability / sampling variability):  8.65
##
## Test for Heterogeneity:
## Q(df = 2) = 16.6463, p-val = 0.0002
##
## Model Results:
##
## estimate      se     zval    pval    ci.lb   ci.ub
##  -0.4111  0.6024  -0.6824  0.4950  -1.5917  0.7696
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
rma(
  yi = yi,
  vi = vi,
  # Specify categorical moderator (i.e., mixed)
  subset = (Creativity.Measure_type == "Mixed"),
  method = "REML",
  data = tradmeta
)
```

```
##
## Random-Effects Model (k = 5; tau^2 estimator: REML)
##
## tau^2 (estimated amount of total heterogeneity): 0.0690 (SE = 0.0860)
## tau (square root of estimated tau^2 value):      0.2626
## I^2 (total heterogeneity / total variability):   60.28%
## H^2 (total variability / sampling variability):  2.52
##
## Test for Heterogeneity:
## Q(df = 4) = 10.4551, p-val = 0.0334
##
```

```
## Model Results:
##
## estimate      se    zval    pval    ci.lb   ci.ub
##   0.2947  0.1568  1.8799  0.0601  -0.0125  0.6019  .
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
rma(
  yi = yi,
  vi = vi,
  # Specify categorical moderator (i.e., non-verbal)
  subset = (Creativity.Measure_type == "Non-verbal"),
  method = "REML",
  data = tradmeta
)
```

```
##
## Random-Effects Model (k = 5; tau^2 estimator: REML)
##
## tau^2 (estimated amount of total heterogeneity): 0.1221 (SE = 0.1676)
## tau (square root of estimated tau^2 value):      0.3495
## I^2 (total heterogeneity / total variability):   51.69%
## H^2 (total variability / sampling variability):  2.07
##
## Test for Heterogeneity:
## Q(df = 4) = 8.2822, p-val = 0.0818
##
## Model Results:
##
## estimate      se    zval    pval    ci.lb   ci.ub
##   0.1600  0.2178  0.7344  0.4627  -0.2669  0.5868
##
## ---
## 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 forest plot includes the following features:

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

After running this code, the forest plot of moderators will appear as shown on page 18 of this handbook.

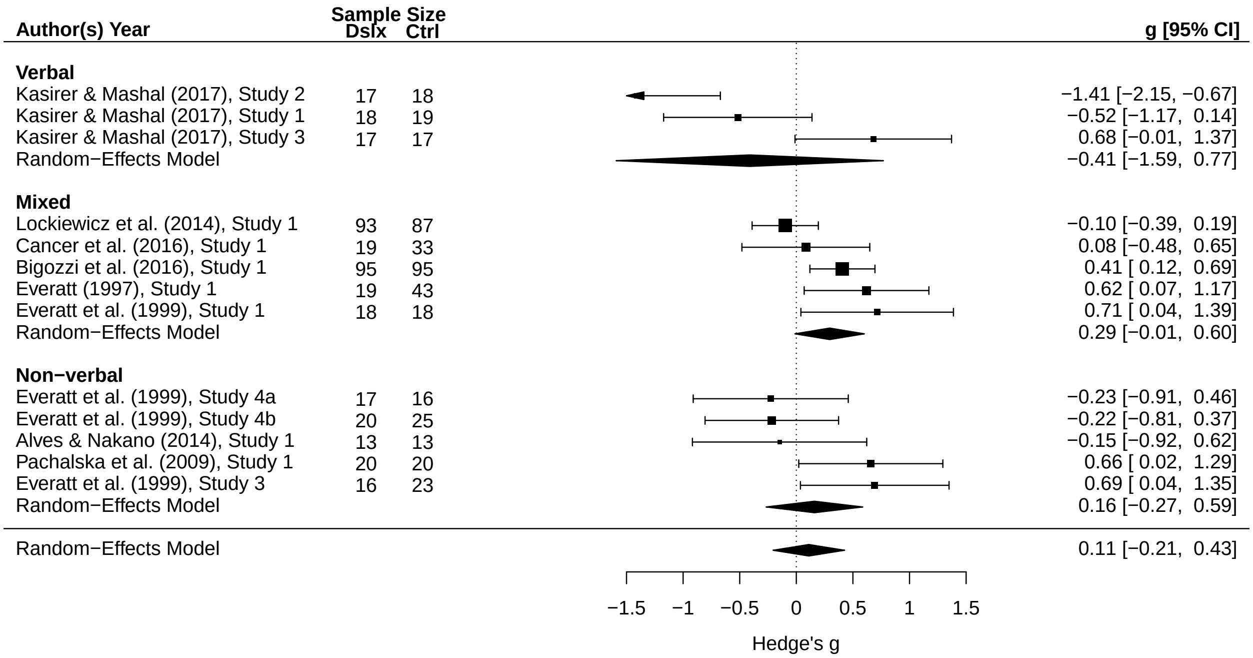
Explanation of the Code

- 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 `tradmetaresults` object contains the results of the meta-analysis.
- The `rows` argument specifies the arrangement of studies by creativity measure type, and in ascending order of effect sizes per creativity measure 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 dyslexia and control groups.
- The `cbind()` function combines multiple columns side-by-side. In this case, the sample sizes of the groups (dys and control) 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 labels.
- The `slab` argument is used to label each effect size with its respective study.
- The `paste()` function creates the label by combining the Paper (i.e., authors of the paper) and Study columns. Since the Study column in the dataset only contains numbers, we added the word "Study" before the number to make it clearer. The `sep` argument specifies the separator between paper and the study label, which is set to "," in this case. Hence, the label will be in the format "Paper, Study X" (e.g., "Smith et al., Study 1").
- 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 `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 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 `pos` argument specifies the position of the text relative to the specified coordinates.
- The `rma()` function is used to perform moderation analysis for each type of creativity measure.
- The `res.v`, `res.n`, and `res.m` variables store the results of the moderation analysis for verbal, non-verbal, and mixed creativity measures, respectively.
- 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 `c()` function within the `text()` function creates a vector of x-positions, where the labels "Dslx" and "Ctrl" will be placed. The first x-value (-3.8) determines the horizontal position of the "Dslx" label, and the second x-value (-3.3) positions the "Ctrl" label.
- 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(
  tradmetaresults,

  # Manually arrange effect sizes by creativity measure type
  # - Verbal: Rows 20 to 18
  # - Mixed: Rows 14 to 10
  # - Non-verbal: Rows 6 to 2
  # The arrangement must consider spacing and must end at row 2
  rows = c(20:18, 14:10, 6:2),

  # Add y-axis limits
  ylim = c(-2, 24.5),

  # Add sample size information for dyslexia (n_dys) and control (n_control) group
  # -3.8 for Dslx (Dyslexia Group)
  # -3.3 for Ctrl (Control Group)
  ilab = cbind(n_dys, n_control),
  ilab.xpos = c(-3.8, -3.3),

  # Label studies on the forest plot
  slab = paste(Paper, paste("Study", Study), 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 = 7,

  # 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 creativity task)
# Labels for different creativity task types (Moderator Analysis)
# - "Non-verbal" at y = 7
# - "Mixed" at y = 15
# - "Verbal" at y = 21
```

```r
text(
  x = -7,
  y = c(7, 15, 21),
  pos = 4,
  c("Non-verbal", "Mixed", "Verbal"),
  font = 2
)

# Moderation analysis
res.v = rma(
  yi,
  vi,
  subset = (Creativity.Measure_type == "Verbal"),
  data = tradmeta
)
res.n = rma(
  yi,
  vi,
  subset = (Creativity.Measure_type == "Non-verbal"),
  data = tradmeta
)
res.m = rma(
  yi,
  vi,
  subset = (Creativity.Measure_type == "Mixed"),
  data = tradmeta
)

# Add summary effect sizes for each of the moderators
addpoly(res.n, row = 1) # summary effect for "non-verbal" group
addpoly(res.m, row = 9) # summary effect for "mixed" group
addpoly(res.v, row = 17) # summary effect for "verbal" group

# Add "Author(s) Year" header
text(x = -6.3, y = 23, "Author(s) Year", font = 2)

# Add "Sample Size" header
text(x = -3.6, y = 23.7, "Sample Size", font = 2)

# Add specific sample size column headers for dyslexia and control groups
# x = -3.8 for Dslx (dyslexia Group)
# x = -3.3 for Ctrl (control Group)
# y = 23 for both
text(c(x = -3.8, x = -3.3), y = 23, c("Dslx", "Ctrl"), font = 2)

# Add "g [95% CI]" header
text(x = 3.5, y = 23, "g [95% CI]", font = 2)
```



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

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

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.

```
### 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 = "tradforestplotwithmoderators.pdf", width = 11, height = 6.5)

# Same forest plot code as above
forest(
  tradmetaresults,

  rows = c(20:18, 14:10, 6:2),
  ylim = c(-2, 24.5),
  ilab = cbind(n_dys, n_control),
  ilab.xpos = c(-3.8, -3.3),
```

```r
  slab = paste(Paper, paste("Study", Study), sep = ", "),
  xlim = c(-7, 4),
  alim = c(-1.5, 1.5),
  steps = 7,
  header = FALSE,
  xlab = "Hedge's g"
)

text(
  x = -7,
  y = c(7, 15, 21),
  pos = 4,
  c("Non-verbal", "Mixed", "Verbal"),
  font = 2
)

res.v = rma(
  yi,
  vi,
  subset = (Creativity.Measure_type == "Verbal"),
  data = tradmeta
)
res.n = rma(
  yi,
  vi,
  subset = (Creativity.Measure_type == "Non-verbal"),
  data = tradmeta
)
res.m = rma(
  yi,
  vi,
  subset = (Creativity.Measure_type == "Mixed"),
  data = tradmeta
)

addpoly(res.n, row = 1)
addpoly(res.m, row = 9)
addpoly(res.v, row = 17)

text(x = -6.3, y = 23, "Author(s) Year", font = 2)

text(x = -3.6, y = 23.7, "Sample Size", font = 2)

text(c(x = -3.8, x = -3.3), y = 23, c("Dslx", "Ctrl"), font = 2)

text(x = 3.5, y = 23, "g [95% CI]", font = 2)

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

```
## pdf
##   2
```

End of Code