Multilevel Meta-Analysis (Pearson r) 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` package in R. Specifically, this tutorial utilises Pearson's r and Fisher's Z as the effect size measure, to demonstrate how to convert between the two measures.

The analysis is based on data from Lua et al. (2023) and focuses on the relationship between overall relationship between the need for cognition and well-being. The original paper can be found here: https://doi.org/10.1007/s11031-023-10047-w.

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` 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 "NFCWB.csv", which contains the data drawn from Lua et al.(2023).

```
### Set Up --------------

# Set working directory to that of script's current location
setwd(dirname(rstudioapi::getActiveDocumentContext()$path))

# R version 4.5.0
```

```r
# 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)
```

```r
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
```

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

# Read in data drawn from Lua et al. (2023)
# Original paper: https://doi.org/10.1007/s11031-023-10047-w
mlmmeta_raw = read.csv("NFCWB.csv")
```

Prepare Data

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

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

Explanation of the Code

- The `corr_nfcwb` variable has to be reversed for studies with negative well-being, ensuring that the correlation reflects that the lower the correlation, the lower the well-being. The `ifelse()` function is used to reverse the correlation for studies with negative well-being.

- The `measure` function specifies the type of effect size to be calculated, in this case, "ZCOR" (Fisher's Z). ZCOR is used to calculate the effect size from the raw correlation coefficients instead of Pearson's r as Fisher's Z normalises the distribution of the effect sizes. Thus, Fisher's Z would be less affected by the sampling distribution skew.
- The ri variable `corr_nfcwb` is the column containing the raw correlation coefficients.
- The ni variable `sample_size` is the column containing the sample sizes for each study.
- Afterwards, the `escalc()` function computes the effect sizes (yi) and their corresponding sampling variances (vi) for each study.
- The publication type is categorized into "Published" and "Unpublished" based on the type of publication (e.g., journal article, conference paper, thesis/dissertation).
- The `mlmmeta` data frame is then sorted by the type of publication (published vs unpublished) to facilitate clearer visualization in the forest plot.

```
### Prepare Data --------------

# Clean data file (reverse correlation for negative well-being)
mlmmeta_raw$corr_nfcwb = with(mlmmeta_raw, ifelse(wellbeing_category == "Negative well-being", -corr_nf

# Compute effect sizes for each study
mlmmeta = escalc(
  # Type of effect size measure
  measure = "ZCOR",

  # Column for raw correlation coefficients
  ri = corr_nfcwb,

  # Column for sample sizes
  ni = sample_size,

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

# Categorise publication type into "published" and "unpublished"
# Published: Journal articles
# Unpublished: Conference, Panel Data, Thesis/dissertation, Unpublished data
mlmmeta$publication_type = ifelse(
  mlmmeta$publication_type == "Journal article",
  "Published",
  "Unpublished")

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

Computing the Overall Effect Size

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

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` function specifies the random effects structure, where `~ 1 | sample_id/meta_id`, indicates that random effects are nested within the sample and meta ID.
- The `yi` and `vi` functions specify the effect size estimates and their corresponding sampling variances, respectively.
- The `data` function 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.
- The `psych::fisherz2r()` function is used to convert the effect size estimates from Fisher's Z to Pearson's r.

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

# Effect size estimates
mlmmetaresults = rma.mv(
  # Effect size estimates
  yi = yi,
  # Sampling variances
  V = vi,
  # Include random effects for grouping variable (i.e., sample)
  random = ~ 1 | sample_id/meta_id,
  # Specify where to get the data from
  data = mlmmeta
)

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

```
##
## Multivariate Meta-Analysis Model (k = 108; method: REML)
##
##   logLik  Deviance       AIC       BIC      AICc
##  40.9843  -81.9687  -75.9687  -67.9502  -75.7357
##
## Variance Components:
##
##             estim    sqrt  nlvls  fixed             factor
## sigma^2.1  0.0076  0.0873     52     no          sample_id
## sigma^2.2  0.0156  0.1250    108     no  sample_id/meta_id
##
## Test for Heterogeneity:
## Q(df = 107) = 1573.7169, p-val < .0001
##
## Model Results:
##
## estimate      se    zval    pval   ci.lb   ci.ub
##   0.1977  0.0200  9.8908  <.0001  0.1585  0.2369  ***
```

```
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Convert from Fisher's Z to Pearson's r
mlmmetaresults$b |> psych::fisherz2r()
```

```
##           [,1]
## intrcpt 0.1952
```

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 need for cognition and well-being group
- 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 `cairo_pdf()` function starts the graphics device driver to create PDF files, and the `file` function specifies the name of the file. Specifically, the `cairo_pdf` function is used for font compatibility and in this case, it is for the author's names.
- The `width` and `height` function 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 `order` function 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` function sets the y-axis limits for the plot.
- The `ilab` function is used to add sample size information for need for cognition and well-being group into the forest plot.
- The `cbind()` function combines the columns indicating the sample size of the groups (dys and control).
- The `ilab.xpos` function specifies the horizontal arrangement of the columns.

- The `slab` function is used to label each effect size with its respective study, using the `paste()` function to combine the "Paper" and "Study" columns. The `paste()` function creates the label, and the `sep` function specifies the separator between the columns.
- The `xlim` function sets the x-axis limits for the plot.
- The `alim` function sets the confidence interval limits, and the `steps` function determines the number of intervals in the x-axis.
- The `efac` function changes the size of effect size polygons.
- The `header` function is set to FALSE to allow for manual specification of headers.
- The `xlab` function specifies the confidence interval label for the funnel plot, in this case, "Fisher's Z"
- 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` function in the `text()` function adjust the position of the headers, with the `x` function specifying the horizontal arrangement of the columns and the `y` function specifying the vertical arrangement of the columns.
- The `x` function specifies the horizontal arrangement of the columns and the `y` function specifies the vertical arrangement of the columns.
- The `font` function adjusts the font size.
- The `pos` function in the `text()` function specifies the position of the text relative to the specified coordinates.
- The `font` function 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 --------------

# Save the forest plot as a PDF file
# Name the pdf file of the forest plot
# cairo_pdf function used for font compatibility
# Adjust the width and height of the pdf file
cairo_pdf(file = "NFCWBforestplot.pdf", width = 14, height = 35)

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

  # Arrangement of studies
  order = "obs",

  # Add y-axis limits
  ylim = c(-3, 111),

  # Add sample size information for presence and absence of smartphones group
  # Values indicate the x-axis position of the sample size columns
  ilab = sample_size,
  ilab.xpos = -3,
```

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

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

  # Add confidence interval limits
  # Adjust intervals based on the number of steps
  alim = c(-1.5, 1.5),
  steps = 7,

  # 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, "Fisher's Z"
  xlab = "Fisher's Z"
)

# 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 = 110, "Author(s) Year", font = 2)

# Add "Sample Size" header
text(x = -3, y = 110, "Sample Size", font = 2)

# Add "r [95% CI]" header
text(x = 2.7, y = 110, "Z [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.

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` function specifies the name of the file.
- The `width` and `height` function adjust the dimensions of the PDF file.
- The `funnel()` function is used to create the funnel plot, and the `tradmetaresults` object contains the results of the meta-analysis.
- The `legend` function 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` function 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 #

# 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 = "NFCWBfunnelplot.pdf", width = 8, height = 5)
# funnel function to create the funnel plot, specify the data to create the plot
funnel(mlmmetaresults, legend = TRUE, xlab = "Fisher's Z")
# 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 `lmer()` function is used to fit a linear mixed-effects model, where the effect size weighted by the 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 / vi)` expression indicates that the effect size (yi) is divided by the standard error (vi), which is used to weight the effect sizes in the model.

- The `I(1 / vi)` expression indicates that the inverse of the standard error is included in the model as a predictor.
- The `1 | sample_id` expression indicates that random intercepts are included for each lab, accounting for the nested structure of the data.
- The `data` function 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 --------------------

# Eggers' Test #
lmer(
  # g weighted by SE is predicted by intercept and inverse SE
  # with random intercept by sample
  I(yi / vi) ~ 1 + I(1 / vi) + (1 | sample_id),
  data = mlmmeta
) |>
  # Estimate of interest is the intercept
  summary(correlation = TRUE)
```

```
## Warning: Some predictor variables are on very different scales: consider
## rescaling
## Warning: Some predictor variables are on very different scales: consider
## rescaling

## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: I(yi/vi) ~ 1 + I(1/vi) + (1 | sample_id)
##    Data: mlmmeta
##
## REML criterion at convergence: 1427
##
## Scaled residuals:
##    Min     1Q Median     3Q    Max
## -4.537 -0.103  0.022  0.110  5.077
##
## Random effects:
##  Groups    Name        Variance Std.Dev.
##  sample_id (Intercept) 42221    205
##  Residual              14832    122
## Number of obs: 108, groups:  sample_id, 52
##
## Fixed effects:
##             Estimate Std. Error       df t value            Pr(>|t|)
## (Intercept) -41.0231    35.3050  50.5712   -1.16               0.25
## I(1/vi)       0.2564     0.0202  45.8603   12.67 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##         (Intr)
```

```
## I(1/vi) -0.420
## fit warnings:
## Some predictor variables are on very different scales: consider rescaling
```

Moderation Analysis

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

Explanation of the Code

- The `rma.mv()` function is used to compute the multilevel meta-analysis with moderators.
- The `yi` and `vi` functions specify the effect size estimates and their corresponding sampling variances, respectively.
- The `random` function specifies the random effects structure, where `~ 1 | sample_id/meta_id` indicates that random intercepts are included for both the lab and individual studies, accounting for the nested structure of the data.
- The `subset` function is used to specify the subset of data for each moderator analysis.
- The `data` function 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.

Note that if convergence issues arise, the `control` function 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 | sample_id/meta_id,
  # Specify categorical moderator (i.e., Published articles)
  subset = (publication_type == "Published"),
  data = mlmmeta,
  # To address convergence issues (if it exists)
  control=list(rel.tol=1e-8)
)
```

```
##
## Multivariate Meta-Analysis Model (k = 54; method: REML)
##
## Variance Components:
##
##             estim    sqrt  nlvls  fixed             factor
## sigma^2.1  0.0158  0.1257     37     no          sample_id
## sigma^2.2  0.0103  0.1016     54     no  sample_id/meta_id
```

```
##
## Test for Heterogeneity:
## Q(df = 53) = 1058.6768, p-val < .0001
##
## Model Results:
##
## estimate      se    zval    pval   ci.lb   ci.ub
##   0.1915  0.0275  6.9686  <.0001  0.1376  0.2453  ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
rma.mv(
  yi = yi,
  V = vi,
  random = ~ 1 | sample_id/meta_id,
  # Specify categorical moderator (i.e., Unpublished articles)
  subset = (publication_type == "Unpublished"),
  data = mlmmeta,
  # To address convergence issues (if it exists)
  control=list(rel.tol=1e-8)
)
```

```
##
## Multivariate Meta-Analysis Model (k = 54; method: REML)
##
## Variance Components:
##
##             estim    sqrt  nlvls  fixed             factor
## sigma^2.1  0.0040  0.0633     15     no          sample_id
## sigma^2.2  0.0170  0.1304     54     no  sample_id/meta_id
##
## Test for Heterogeneity:
## Q(df = 53) = 351.3640, p-val < .0001
##
## Model Results:
##
## estimate      se    zval    pval   ci.lb   ci.ub
##   0.2126  0.0289  7.3678  <.0001  0.1561  0.2692  ***
##
## ---
## 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 need for cognition and well-being group
- Custom headers for the plot
- Custom labels for the studies
- Summary effect sizes for each moderator

Explanation of the Code

- The `cairo_pdf()` function starts the graphics device driver to create PDF files, and the `file` function specifies the name of the file. Specifically, the `cairo_pdf` function is used for font compatibility and in this case, it is for the author's names.
- The `width` and `height` function 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` function specifies the arrangement of studies by publication type, and in ascending order of effect sizes per publication type.
- The `ylim` function sets the y-axis limits for the plot.
- The `ilab` function is used to add sample size information for both presence of smartphones and absence of smartphones groups into the forest plot.
- The `cbind()` function combines the columns indicating the sample size of the groups (dys and control).
- The `ilab.xpos` function specifies the horizontal arrangement of the columns, while the `xlim` function sets the x-axis limits for the plot.
- The `slab` function is used to label each effect size with its respective study, using the `paste()` function to combine the "Paper" and "Study" columns.
- The `paste()` function creates the label, and the `sep` function specifies the separator between the columns.
- The `alim` function sets the confidence interval limits, and the `steps` function determines the number of intervals in the x-axis.
- The `header` function is set to FALSE to allow for manual specification of headers.
- The `xlab` function 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` functions in the `text()` function adjust the position of the headers.
- The `x` function specifies the horizontal arrangement of the columns and the `y` function specifies the vertical arrangement of the columns.
- The `font` function adjusts the font size.
- The `pos` function specifies the position of the text relative to the specified coordinates
- The `rma.mv()` function is used to perform moderation analysis for each publication type, with the `subset` function specifying the subset of data for each category.

- The `rest.j`, `res.t`, and `res.c` variables store the results of the moderation analysis for journal articles, thesis/dissertations, and conference papers, respectively.
- The `subset` function 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` function 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 --------------

# Save the forest plot as a PDF file
# Name the pdf file of the forest plot
# cairo_pdf function used for font compatibility
# Adjust the width and height of the pdf file
cairo_pdf(file = "NFCWBforestplotwithmod.pdf", width = 13, height = 35)

# Start creating the forest plot itself
# Specify dataset
forest(
  mlmmetaresults,
  # Manually arrange effect sizes by publication type
  # - Unpublished: Rows 108 to 51
  # - Published: Rows 50 to 48
  # The arrangement must consider spacing and must end at row 2
  rows = c(112:40, 36:2),

  # Add y-axis limits
  ylim = c(-3, 116),

  # Add sample size information for presence and absence of smartphones group
  # Values indicate the x-axis position of the sample size columns
  ilab = sample_size,
  ilab.xpos = -4.2,

  # Label studies on the forest plot
  slab = paste(author, year, 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,

  # 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, "Fisher's Z"
```

```
  xlab = "Fisher's Z"
)

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

# Add text labels for moderator (type of publication)
# x values to indicate the horizontal arrangement of the text
# Labels for different publication types (Moderator Analysis)
# y values indicate the vertical arrangement of the text

# - "Unpublished" (Unpublished data, Panel Data, Thesis/Dissertations) at
# - "Published" (Journal Articles, Conference) at
# `pos = 4` ensures left alignment of the text
text(
  x = -7,
  y = c(37, 113),
  pos = 4,
  c("Unpublished", "Published"),
  font = 2
)

# Moderation analysis
res.p = rma.mv(
  yi,
  vi,
  random = ~ 1 | sample_id/meta_id,
  subset = (publication_type == "Published"),
  data = mlmmeta,
  # To address convergence issues (if it exists)
  control=list(rel.tol=1e-8)
)
res.u = rma.mv(
  yi,
  vi,
  random = ~ 1 | sample_id/meta_id,
  subset = (publication_type == "Unpublished"),
  data = mlmmeta,
  # To address convergence issues (if it exists)
  control=list(rel.tol=1e-8)
)

# Add summary effect sizes for each of the moderators
addpoly(res.u, row = 1) # summary effect for "Unpublished" group
addpoly(res.p, row = 39) # summary effect for "Published" group

# Add"Author(s) Year" header
text(x = -6.4, y = 115,"Author(s) Year", font = 2)

# Add "Sample Size" header
text(x = -4.0, y = 115,"Sample Size", font = 2)

# Add "r [95% CI]" header
```

```r
text(x = 3.6, y = 115, "Z [95% CI]", font = 2)

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

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

END OF CODE