Calculate and combine the effect-sizes

Damien Beillouin

2024-09-28

Contents

About

This training module is designed to provide an introduction to data exploration and visualization using R, with a particular focus on meta-analysis. The course is built around hands-on exercises aimed at developing essential skills for manipulating, summarizing, and presenting data in a clear, visually appealing format. By leveraging R packages such as ggplot2, cowplot, and rnaturalearth, participants will learn how to generate meaningful visual representations, such as histograms, maps, treemaps, heatmaps, and bubble plots.

The data used in this course comes from various scientific studies that analyze the impact of human interventions on various outcomes. The exercises cover a wide range of techniques, from simple bar charts to more complex visualizations like interactive maps and dynamic tables. The primary goal is to equip participants with practical tools to explore large datasets, summarize findings, and effectively communicate results in scientific contexts.

Whether you're new to R or looking to expand your data visualization skills, this module will provide you with a strong foundation in the fundamentals of data analysis and graphical presentation, emphasizing clarity, accuracy, and aesthetic quality in your work.

Chapter 1

What is an Effect Size?

An effect size is a metric that quantifies the **direction** and **magnitude** of an experimental or observational effect. It is essential in meta-analysis as it standardizes results across different studies to aggregate results.For example, if you are comparing the performance of two treatments (e.g., a control vs. a new intervention), the effect size tells you how much better or worse the new intervention performs compared to the control.Effect size should have the folowing characteristics:

- **Extracted directly** from publications or **calculated** from reported data.
- **Standardized** to be comparable across multiple primary studies.
- **Reliable**: Effect sizes should accurately represent the underlying data (e.g., caution is needed when using ratios with low denominators).

Effect sizes come in various forms depending on the type of data and research question (e.g., standardized mean difference, odds ratios, correlation coefficients).

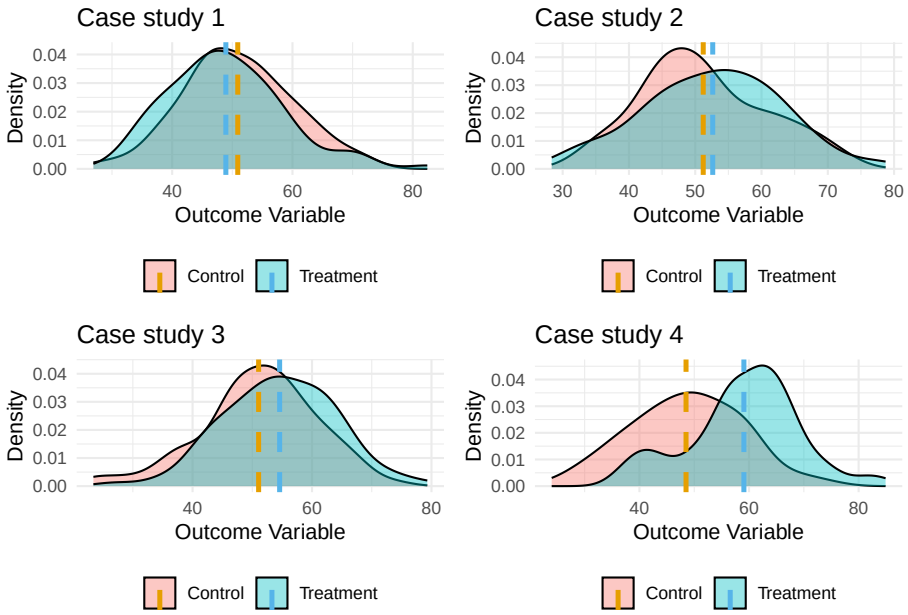
1.1 Visual representation of effect-sizes

Effect sizes are crucial for understanding how different treatments or conditions compare to each other in a study. To grasp their significance, it is helpful to visualize them, as graphical representation allows us to clearly see how much two groups overlap or differ. By illustrating varying levels of effect sizes, we can better understand what these numerical values mean in a practical context. Example Scenario: Comparing a Treatment vs. Control Group

Let's consider a hypothetical study where researchers are testing the effectiveness of a new treatment compared to a control group. For each group, a series

of measurements were collected, allowing us to estimate the average value and the statistical distribution for each population.

The goal is to determine if the treatment group exhibits a significant improvement or change compared to the control group. Here, the effect size is a standardized measure that quantifies how much the two groups differ. The larger the effect size, the more distinct the two groups are. Smaller effect sizes suggest that the two groups are relatively similar, while larger effect sizes indicate a more substantial difference



1.1.1 Different Metrics for Measuring Effect Sizes

Effect sizes can be calculated using a variety of metrics depending on the type of data and the comparisons being made. Below, we present a table showing some of the most common effect size metrics, calculated for the same hypothetical data used in the visual demonstration above.

Metric	Formula/Definition	Interpretation
Mean Difference	Mean_Treatment - Mean_Control	Difference in mean values between the groups.
Standardized Mean Difference (SMD)	(Mean_Treatment - Mean_Control) / SD_pooled	Cohen's d or Hedges' g for continuous variables.

Metric	Formula/Definition	Interpretation
Odds Ratio (OR)	(Odds_Treatment / Odds_Control)	Odds of an event occurring in the treatment vs. control.
Log Odds Ratio	log(OR)	Stabilized variance and more symmetrical distribution.
Risk Ratio (RR)	(Risk_Treatment / Risk_Control)	Ratio of the probability of an event between groups.
Log Risk Ratio	log(RR)	Similar to log OR, stabilizing variances.

Using the simulated data above, let's compare the groups using several of these metrics:

Effect Size Metric	Case study 1	Case study 2	Case study 3	Case study 4
Mean Difference	0	3.15	5.27	7.86
Standardized Mean Difference (SMD)	0	0.31	0.52	0.81
Odds Ratio (OR)	1	1.28	1.91	3.75
Log Odds Ratio	0	0.25	0.65	1.32
Risk Ratio (RR)	1	1.18	1.40	2.25
Log Risk Ratio	0	0.17	0.34	0.81

These values illustrate how different metrics capture varying aspects of the comparison between groups. We will delve into these metrics in greater detail in the next sections, exploring when and why to use each one based on your research question and data type.

Overall, effect sizes provide a powerful way to quantify the differences between groups, and choosing the appropriate metric ensures accurate and meaningful interpretations in meta-analyses and research synthesis.

1.1.2 How to Measure Variability

Variability in effect sizes can be measured using several approaches:

- **Confidence Intervals (CIs)**: These provide a range within which the true effect size is likely to fall. A 95% CI indicates that if the study were repeated many times, 95% of the calculated intervals would contain the true effect size.
- **Standard Error (SE)**: This metric quantifies the amount of variation in the effect size estimate due to sampling error. A smaller SE suggests that the effect size is a more reliable estimate.
- **Heterogeneity Statistics**: In meta-analyses, measures like I^2 quantify the degree of variability among effect sizes from different studies. A high I^2 value indicates substantial variability, suggesting that differences between studies may not be due to chance alone.

1.1.3 What Variability Means

- **High Variability**: Suggests that the effect of the treatment varies significantly across different studies or populations. This could indicate that certain factors (like participant characteristics or study design) influence the effectiveness of the intervention.
- **Low Variability**: Implies that the effect size is consistent across studies, leading to stronger generalizations about the treatment's effectiveness.

By understanding and measuring the variability of effect sizes, researchers can provide more nuanced interpretations of their findings and enhance the credibility of their conclusions. This approach leads to better-informed decisions in both research and practice.

1.1.4 Comparison of Effect Size and p-Value

Understanding the Concepts

- **Effect Size**: This metric quantifies the magnitude of a treatment effect or the strength of an association. It provides a clear indication of how substantial a difference is between groups, regardless of sample size. A larger effect size suggests a more meaningful difference, helping researchers understand the practical significance of their findings.

- **P-Value**: The p-value measures the probability that the observed results occurred by chance under the null hypothesis. It helps determine whether to reject the null hypothesis. A low p-value (typically < 0.05) indicates that the observed difference is statistically significant, but it does not inform us about the size or importance of the effect.

Key Differences

1. **Focus**:
   - **Effect Size**: Emphasizes the magnitude of the effect, providing context to the results.
   - **P-Value**: Emphasizes whether the effect is statistically significant, without indicating its size.
2. **Interpretation**:
   - **Effect Size**: A high effect size indicates a strong difference, while a low effect size suggests the groups are similar. It informs researchers about practical implications.
   - **P-Value**: A p-value tells us whether we can reject the null hypothesis. A significant p-value doesn't necessarily imply a large or important effect.
3. **Dependence on Sample Size**:
   - **Effect Size**: Remains stable regardless of sample size, providing a consistent measure of difference.
   - **P-Value**: Can be influenced by sample size; larger samples can yield statistically significant p-values even for trivial differences.

Chapter 2

Comparative meta-analytic methods for evidence synthesis

This chapter is designed to provide a comparative exploration of different meta-analytic approaches using the `metafor`, `meta`, and `brms` packages in R. The objective is to understand how the choice of methods (e.g., fixed-effect vs. random-effects models) and different effect size metrics (Standardized Mean Difference, Risk Ratios, etc.) can impact the interpretation of results. We will utilize a variety of datasets to illustrate how these choices influence conclusions, highlighting key interpretative aspects when heterogeneity and study-level variations are present.

By examining multiple datasets and applying different statistical techniques, we will underscore the importance of selecting appropriate methods based on the characteristics of the data, including heterogeneity among studies and the nature of the effect sizes being analyzed.

Prerequisites

You should have a basic understanding of meta-analysis concepts and be familiar with R. If you haven't yet installed the necessary R packages, run the following commands:

```
#install.packages(c("tidyverse", "metafor", "dmetar", "meta", "metadat", "ggplot2", "reactable",
```

2.1 Equal-Effects Model

2.1.1 Example Data

Consider the meta-analysis by Molloy et al. (2014), which examines the relationship between conscientiousness and medication adherence. We can compute the r-to-z transformed correlation coefficient and corresponding sampling variances using the `metafor` package:

```
library(metafor)
dat <- escalc(measure="ZCOR", ri=ri, ni=ni, data=dat.molloy2014)
```

The dataset contains the following columns: authors, year, sample size (ni), correlation coefficient (ri), transformed coefficient (yi), and sampling variance (vi).

2.1.2 Fitting the Equal-Effects Model

We can fit an equal-effects model with `rma()`:

```
res.ee <- rma(yi, vi, data=dat, method="EE")
summary(res.ee)
```

```
##
## Equal-Effects Model (k = 16)
##
##   logLik  deviance       AIC       BIC      AICc
##   5.4330   38.1595   -8.8659   -8.0933   -8.5802
##
## I^2 (total heterogeneity / total variability):   60.69%
## H^2 (total variability / sampling variability):  2.54
##
## Test for Heterogeneity:
## Q(df = 15) = 38.1595, p-val = 0.0009
##
## Model Results:
##
## estimate      se    zval    pval   ci.lb   ci.ub
##   0.1252  0.0170  7.3642  <.0001  0.0919  0.1585  ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This outputs the estimated effect size, heterogeneity statistics (I^2, H^2), and a test for heterogeneity.

Next, we fit the same model using `lm()` by specifying the inverse of the sampling variances as weights:

```
res.lm <- lm(yi ~ 1, weights = 1/vi, data=dat)
summary(res.lm)
```

```
##
## Call:
## lm(formula = yi ~ 1, data = dat, weights = 1/vi)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1919 -1.0719  0.6674  1.3173  2.4695
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.12518    0.02711   4.617 0.000335 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.595 on 15 degrees of freedom
```

2.1.3 Comparison of Results

1. **Coefficient Comparison**: The estimated intercept from `lm()` matches that from `rma()`, although it's rounded differently.
2. **Standard Errors**: The standard error from `lm()` differs because `lm()` assumes weights are only known up to a proportionality constant (^{2}e). This error can be demonstrated by extracting the estimated error variance from the `lm` object and refitting the model with `rma()`:

```
rma(yi, vi * sigma(res.lm)^2, data=dat, method="EE")
```

```
##
## Equal-Effects Model (k = 16)
##
## I^2 (total heterogeneity / total variability):   0.00%
## H^2 (total variability / sampling variability):  1.00
##
## Test for Heterogeneity:
```

```
## Q(df = 15) = 15.0000, p-val = 0.4514
##
## Model Results:
##
## estimate      se    zval    pval   ci.lb   ci.ub
##   0.1252  0.0271  4.6171  <.0001  0.0720  0.1783  ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This shows that adjusting the standard errors to account for the estimated error variance results in consistent estimates across both functions.

2.2 Random-Effects Model

2.2.1 Fitting the Random-Effects Model

We can fit a random-effects model using `rma()` as follows:

```
res.re <- rma(yi, vi, data=dat)
summary(res.re)
```

```
##
## Random-Effects Model (k = 16; tau^2 estimator: REML)
##
##   logLik  deviance       AIC       BIC      AICc
##   8.6096  -17.2191  -13.2191  -11.8030  -12.2191
##
## tau^2 (estimated amount of total heterogeneity): 0.0081 (SE = 0.0055)
## tau (square root of estimated tau^2 value):      0.0901
## I^2 (total heterogeneity / total variability):   61.73%
## H^2 (total variability / sampling variability):  2.61
##
## Test for Heterogeneity:
## Q(df = 15) = 38.1595, p-val = 0.0009
##
## Model Results:
##
## estimate      se    zval    pval   ci.lb   ci.ub
##   0.1499  0.0316  4.7501  <.0001  0.0881  0.2118  ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Next, we try fitting the same model with the `lme()` function from the `nlme` package, specifying a variance function for fixed variances:

```
library(nlme)
dat$study <- 1:nrow(dat)
res.lme <- lme(yi ~ 1, random = ~ 1 | study, weights = varFixed(~ vi), data=dat)
summary(res.lme)
```

```
## Linear mixed-effects model fit by REML
##   Data: dat
##         AIC       BIC   logLik
##   -8.781569 -6.657418 7.390784
##
## Random effects:
##  Formula: ~1 | study
##         (Intercept) Residual
## StdDev:  0.06818354 1.259548
##
## Variance function:
##  Structure: fixed weights
##  Formula: ~vi
## Fixed effects:  yi ~ 1
##                 Value  Std.Error DF  t-value p-value
## (Intercept) 0.1415557 0.03071906 16 4.608073   3e-04
##
## Standardized Within-Group Residuals:
##        Min         Q1        Med         Q3        Max
## -1.1596768 -0.6903414  0.1964221  0.7117538  1.4616798
##
## Number of Observations: 16
## Number of Groups: 16
```

Alternatively, we can use the `lmer()` function from the `lme4` package

```
library(lme4)
res.lmer <- lmer(yi ~ 1 + (1 | study), weights = 1/vi, data=dat,
                 control=lmerControl(check.nobs.vs.nlev="ignore", check.nobs.vs.nRE="ignore"))
summary(res.lmer)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: yi ~ 1 + (1 | study)
##    Data: dat
## Weights: 1/vi
## Control:
```

```
## lmerControl(check.nobs.vs.nlev = "ignore", check.nobs.vs.nRE = "ignore")
##
## REML criterion at convergence: -14.8
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.1597 -0.6903  0.1964  0.7117  1.4617
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  study    (Intercept) 0.004649 0.06818
##  Residual             1.586457 1.25955
## Number of obs: 16, groups:  study, 16
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  0.14156    0.03072   4.608
```

2.2.2 Comparison of Results

1. **Intercept Differences**: The estimated intercepts from `lme()` and `lmer()` do not differ from the estimate obtained via `rma()`.

2. **Standard Errors**: The standard errors, t-values, and p-values also show discrepancies for similar reasons as mentioned earlier—`lme()` and `lmer()` treat sampling variances as unknown beyond a proportionality constant.

To illustrate, we can factor this constant into the sampling variances and refit the model with `rma()`:

```
rma(yi, vi * sigma(res.lme)^2, data=dat)
```

```
##
## Random-Effects Model (k = 16; tau^2 estimator: REML)
##
## tau^2 (estimated amount of total heterogeneity): 0.0046 (SE = 0.0049)
## tau (square root of estimated tau^2 value):      0.0682
## I^2 (total heterogeneity / total variability):   36.82%
## H^2 (total variability / sampling variability):  1.58
##
## Test for Heterogeneity:
## Q(df = 15) = 24.0532, p-val = 0.0642
##
## Model Results:
```

```
##
## estimate      se    zval    pval   ci.lb   ci.ub
##   0.1416  0.0307  4.6081  <.0001  0.0813  0.2018  ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This yields results consistent with those from `lme()`.

2.2.2.1 Comparison with the `meta` Package

The `meta` package provides a robust alternative for conducting meta-analyses, featuring a range of built-in functions that cater to both fixed-effect and random-effects models. This package streamlines the analysis of various effect sizes, making it accessible for researchers. For example, we can execute a meta-analysis using the `metagen()` function from the `meta` package as follows:

```
library(meta)
res.meta <- metagen(TE = yi, seTE = sqrt(vi), data = dat, sm = "SMD")
summary(res.meta)
```

```
##        SMD            95%-CI %W(common) %W(random)
## 1   0.1892 [-0.0011; 0.3796]        3.1        5.7
## 2   0.1634 [ 0.0917; 0.2352]       21.6       10.5
## 3   0.3541 [ 0.0823; 0.6259]        1.5        3.6
## 4   0.3316 [ 0.1395; 0.5238]        3.0        5.6
## 5   0.2769 [ 0.0409; 0.5128]        2.0        4.4
## 6   0.0000 [-0.2489; 0.2489]        1.8        4.1
## 7   0.1768 [ 0.0269; 0.3267]        4.9        7.1
## 8   0.0500 [-0.0590; 0.1591]        9.3        8.9
## 9   0.2661 [ 0.0018; 0.5304]        1.6        3.8
## 10  0.0100 [-0.0607; 0.0807]       22.2       10.6
## 11 -0.0902 [-0.3595; 0.1790]        1.5        3.7
## 12  0.3884 [ 0.1795; 0.5974]        2.5        5.1
## 13  0.0000 [-0.1844; 0.1844]        3.3        5.9
## 14  0.1511 [ 0.0663; 0.2360]       15.4       10.0
## 15  0.2448 [ 0.0873; 0.4022]        4.5        6.8
## 16  0.0400 [-0.2089; 0.2889]        1.8        4.1
##
## Number of studies: k = 16
##
##                          SMD           95%-CI    z  p-value
## Common effect model  0.1252 [0.0919; 0.1585] 7.36 < 0.0001
## Random effects model 0.1499 [0.0881; 0.2118] 4.75 < 0.0001
```

```
##
## Quantifying heterogeneity:
##  tau^2 = 0.0081 [0.0017; 0.0378]; tau = 0.0901 [0.0412; 0.1944]
##  I^2 = 60.7% [32.1%; 77.2%]; H = 1.59 [1.21; 2.10]
##
## Test of heterogeneity:
##      Q d.f. p-value
##  38.16   15  0.0009
##
## Details on meta-analytical method:
## - Inverse variance method
## - Restricted maximum-likelihood estimator for tau^2
## - Q-Profile method for confidence interval of tau^2 and tau
```

One of the key advantages of the `meta` package is its user-friendly syntax, which simplifies the process of conducting meta-analyses. Additionally, it integrates functions for visualizing results, such as forest plots, directly within its framework, enhancing the interpretability of the findings.

2.2.2.2 Bayesian Meta-Analysis

Bayesian meta-analysis presents a flexible framework that accommodates the incorporation of prior distributions and quantifies uncertainty in a comprehensive manner. By employing the `brms` package, we can specify a Bayesian random-effects model as follows:

```r
library(brms)

# Define the response variable and the standard error
# Replace 'yi' with your actual response variable and 'se' with the appropriate standa
response_var <- "yi"  # Change this to your response variable name
se_var <- "sqrt(vi)"  # Calculate the standard error (if vi is the variance)

# Fit the Bayesian random-effects model with specified priors
bayes_model <- brm(
  formula = paste0(response_var, " | se(", se_var, ") ~ 1 + (1 | study)"),
  data = dat,
  family = gaussian,
  prior = c(
    prior(normal(0, 1), class = "Intercept"),   # Prior for the intercept
    prior(cauchy(0, 1), class = "sd")            # Prior for the standard deviation of
  ),
  iter = 200,
  warmup = 100,
  cores = 4,
```

```
  chains = 2,
  seed = 14
)
```

```
## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## using C compiler: 'Apple clang version 15.0.0 (clang-1500.3.9.4)'
## using SDK: 'MacOSX14.4.sdk'
## clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Library
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/library/Sta
## In file included from /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/library/Rcp
## In file included from /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/library/Rcp
## /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/library/RcppEigen/include/Eigen/s
## #include <cmath>
##          ^~~~~~~
## 1 error generated.
## make: *** [foo.o] Error 1
```

```
# Summarize the results
summary(bayes_model)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: yi | se(sqrt(vi)) ~ 1 + (1 | study)
##    Data: dat (Number of observations: 16)
##   Draws: 2 chains, each with iter = 200; warmup = 100; thin = 1;
##          total post-warmup draws = 200
##
## Multilevel Hyperparameters:
## ~study (Number of levels: 16)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     0.11      0.04     0.05     0.19 1.00       63      113
##
## Regression Coefficients:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     0.15      0.04     0.09     0.24 1.00      180      186
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.00      0.00     0.00     0.00   NA       NA       NA
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
## 40%]  (Warmup)
## Chain 2: Iteration:  80 / 200 [ 40%]  (Warmup)
## Chain 1: Iteration: 100 / 200 [ 50%]  (Warmup)
## Chain 1: Iteration: 101 / 200 [ 50%]  (Sampling)
## Chain 2: Iteration: 100 / 200 [ 50%]  (Warmup)
## Chain 2: Iteration: 101 / 200 [ 50%]  (Sampling)
## Chain 1: Iteration: 120 / 200 [ 60%]  (Sampling)
## Chain 2: Iteration: 120 / 200 [ 60%]  (Sampling)
## Chain 1: Iteration: 140 / 200 [ 70%]  (Sampling)
## Chain 2: Iteration: 140 / 200 [ 70%]  (Sampling)
## Chain 1: Iteration: 160 / 200 [ 80%]  (Sampling)
## Chain 2: Iteration: 160 / 200 [ 80%]  (Sampling)
## Chain 1: Iteration: 180 / 200 [ 90%]  (Sampling)
## Chain 2: Iteration: 180 / 200 [ 90%]  (Sampling)
## Chain 1: Iteration: 200 / 200 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.035 seconds (Warm-up)
## Chain 1:                0.025 seconds (Sampling)
## Chain 1:                0.06 seconds (Total)
## Chain 1:
## Chain 2: Iteration: 200 / 200 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.033 seconds (Warm-up)
## Chain 2:                0.027 seconds (Sampling)
## Chain 2:                0.06 seconds (Total)
## Chain 2:
```

This Bayesian approach enables us to derive posterior distributions for the effect sizes, facilitating a more nuanced interpretation of uncertainty compared to traditional frequentist methods. By leveraging Bayesian techniques, researchers can integrate prior knowledge and obtain more robust estimates, especially in cases where sample sizes are limited or when study results exhibit significant variability.

2.2.3 Meta-Forest Analysis: A Comparison with Classical Meta-Analytic Models

Meta-forest analysis is a machine-learning-based extension of meta-analysis that addresses the limitations of traditional meta-analytic models. In standard approaches like meta-regression, we assume a linear relationship between moderators and the effect sizes, which can oversimplify the true patterns within the data. Meta-forest, on the other hand, leverages the power of random forests to capture **nonlinear interactions** and **complex relationships** between moderators and outcomes, providing a deeper understanding of heterogeneity.

2.2.3.1 Key Differences and Advantages

1. **Handling Complex and Nonlinear Relationships**:
   Traditional meta-regression models fit a single linear equation to explain the relationship between effect sizes and moderators. This approach works well for straightforward datasets but struggles when the effects of moderators are nonlinear or when there are interactions between variables. Meta-forest overcomes these limitations by utilizing decision trees, which split the data into multiple regions and fit localized models, effectively capturing nonlinearities and interactions that would be missed otherwise.

2. **No Assumptions on Functional Form**:
   Classical models require specifying the functional form of the relationship a priori (e.g., linear or quadratic). If the chosen form is incorrect, the model can lead to biased or misleading conclusions. Meta-forest avoids this issue by being entirely data-driven, learning the shape of the relationships directly from the data without requiring these assumptions.

3. **Resilience to Overfitting**:
   While adding more moderators to a standard meta-regression model can quickly lead to overfitting, especially in small datasets, meta-forest models are designed to mitigate overfitting through cross-validation and ensemble averaging. This robustness is crucial when dealing with a large number of potential moderators, allowing researchers to explore multiple variables without risking model instability.

4. **Variable Importance and Feature Selection**:
   One of the most valuable outputs from a meta-forest model is the **variable importance** score. Unlike classical models, which may produce ambiguous results when variables are collinear or interact with each other, meta-forest provides a clear ranking of which moderators contribute the most to explaining heterogeneity. This feature is particularly beneficial when performing exploratory analyses or when the moderator space is large and uncertain.

5. **Partial Dependence Plots for Interpretation**:
   Meta-forest offers visual tools like **partial dependence plots** that show the marginal effect of each moderator while accounting for the influence of other variables. This allows researchers to interpret the role of each moderator in context, which is challenging to achieve with standard meta-regression models.

2.2.3.2 Potential Drawbacks of Meta-Forest

Despite its strengths, meta-forest also has some drawbacks:

- **Lack of Inference Statistics**:
  Unlike classical models, meta-forest does not provide p-values or confidence intervals for effect size estimates. While this is acceptable for exploratory analyses, it can be limiting if formal hypothesis testing is required.
- **Higher Complexity and Computational Cost**:
  The complexity of meta-forest models requires more computational resources and time, especially for large datasets or numerous moderators. Researchers may need to balance the improved flexibility against increased computational demand.
- **Limited Applicability for Small Datasets**:
  Meta-forest is most effective when there is a substantial number of studies and a diverse set of moderators. For small datasets, classical methods like mixed-effects models might be more appropriate due to their simpler structure and ease of interpretation.

2.2.3.3 Practical Application: When to Use Meta-Forest?

Meta-forest is an ideal tool when dealing with **exploratory analyses** or when the **relationships between moderators and effect sizes are unknown or suspected to be nonlinear**. It is particularly useful for:

1. Exploring datasets with **many potential moderators** where interactions are likely.
2. Identifying key predictors of heterogeneity when prior knowledge is limited.
3. Providing a preliminary understanding of complex datasets before fitting more traditional models.

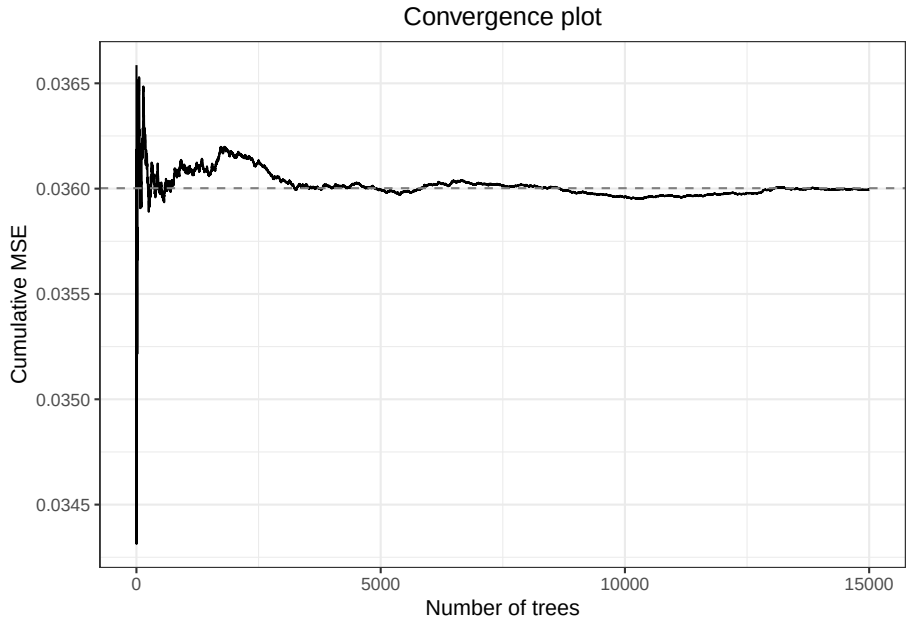
However, for datasets where linear assumptions are well-supported and computational efficiency is a concern, classical methods like **meta-regression** remain preferable.

2.2.3.4 Example of Implementing a Meta-Forest Model

Below is a demonstration of how to fit a meta-forest model and assess the relative importance of moderators:

```
# Load required libraries
library(metaforest)
library(caret)
```

```r
# Hypothetical dataset: Replace this with your own meta-analysis data
set.seed(123)
data_agro <- data.frame(
  yi = rnorm(60, 0.3, 0.2),          # Simulated effect sizes
  vi = min(0.01,rnorm(60, 0.1, 0.05)),         # Variances of the effect sizes
  Intervention = sample(c("Fruit", "Mixed", "Timber"), 60, replace = TRUE),
  StudyID = rep(1:15, each = 4),     # 15 unique studies
  Region = sample(c("Tropical", "Temperate"), 60, replace = TRUE)
)

# Step 1: Run initial MetaForest model with many trees to check convergence
initial_model <- MetaForest(
  formula = yi ~ Intervention + Region,
  data = data_agro,
  study = "StudyID",
  whichweights = "random",           # Use random-effects weighting scheme
  num.trees = 15000                  # Set a high number of trees for convergence
)

# Step 2: Plot convergence trajectory (Mean Squared Error vs. Number of Trees)
plot(initial_model)
```



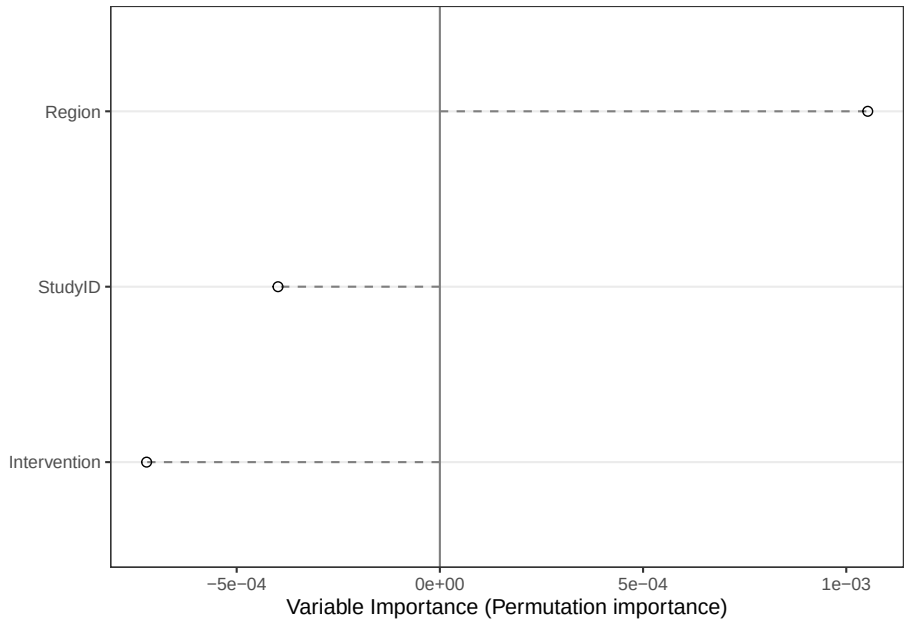
```r
# Step 3: Tune MetaForest model using cross-validation with `caret`
# Define the parameter grid for tuning
tuning_grid <- expand.grid(
  whichweights = c("random", "fixed", "unif"),
  mtry = 1:2,                        # Number of variables to possibly split at in each
  min.node.size = 2:5                # Minimum size of terminal nodes
)

# Setup 10-fold cross-validation
grouped_cv <- trainControl(method = "cv", index = groupKFold(data_agro$StudyID, k = 10)

# Train MetaForest model using `caret` for optimal tuning
tuned_model <- train(
  y = data_agro$yi,
  x = data_agro[, c("Intervention", "Region", "StudyID", "vi")],
  method = ModelInfo_mf(),            # MetaForest method setup for `caret`
  trControl = grouped_cv,             # Cross-validation setup
  tuneGrid = tuning_grid,             # Grid of tuning parameters
  num.trees = 5000                    # Number of trees for final model
)

# Step 4: Examine the optimal tuning parameters and model performance
best_params <- tuned_model$results[which.min(tuned_model$results$RMSE), ]
print(best_params)
```

```
##    whichweights mtry min.node.size      RMSE  Rsquared       MAE     RMSESD
## 20         unif    1             5 0.1813407 0.1106552 0.1465791 0.03107815
##    RsquaredSD      MAESD
## 20  0.1610125 0.03347646
```

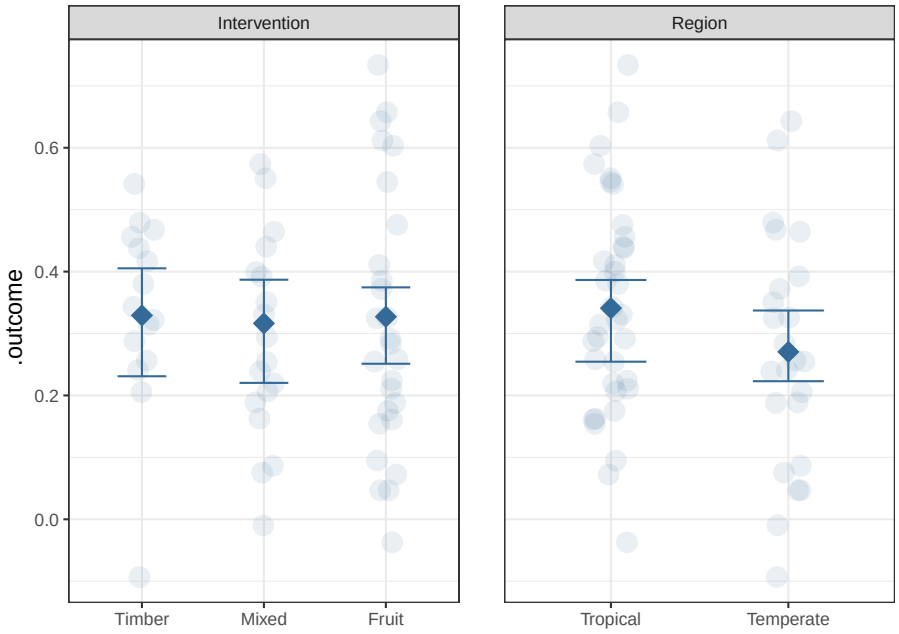
```r
# Step 5: Assess the final model and visualize results
final_model <- tuned_model$finalModel

# R² from out-of-bag predictions (OOB R² indicates model fit for unseen data)
r2_oob <- final_model$forest$r.squared
cat("OOB R-squared: ", r2_oob)
```

```
## OOB R-squared:  -0.07849894
```

```r
# Variable importance plot to identify influential moderators
VarImpPlot(final_model)
```




```
# Step 6: Visualize partial dependence of top moderators
PartialDependence(final_model, vars = names(final_model$forest$variable.importance)[1:2], rawdata
```



2.2.3.5 Handling Random Structures and Weights in Meta-Forest Models

Meta-forest models handle **random structures** and **weights** differently from traditional meta-analysis models:

- **Random Effects**: Traditional models explicitly incorporate random effects to model between-study variability. Meta-forest, however, relies on the structure of decision trees to capture variability implicitly. While this can account for some heterogeneity, it does not separate fixed and random components as effectively as mixed-effects models.
- **Weights**: Classical meta-analyses use **inverse variance weighting** to prioritize studies with higher precision. Meta-forest can integrate study weights similarly during the tree-fitting process, ensuring that precise studies have more influence. However, unlike static weights in classical models, meta-forest adjusts them adaptively during iterations, making it more suitable for complex interactions and nonlinear patterns.

2.3 Conclusion

In summary, while the `lm()`, `lme()`, and `lmer()` functions can fit various linear models, they are not appropriate for meta-analysis due to their assumptions about the error variances. The `rma()` function is specifically designed for meta-analytic contexts, ensuring that known sampling variances are correctly utilized, resulting in reliable estimates and standard errors. Additionally, the `meta` package provides an accessible alternative for meta-analyses, while Bayesian methods using `brms` allow for flexible modeling and uncertainty quantification.

2.4 Practical implementations

2.4.1 Example 1: fixed vs. random Meta-Analysis models

The first dataset explores the impact of two effect size measures: Standardized Mean Difference (SMD) and Log Risk Ratios (lnRR). We apply different models and compare the results to identify variations in interpretations:

- **Calculation of effect-sizes**

```r
dat <- metadat::dat.curtis1998

# Standardized mean differences
    SMD <- escalc(measure = "SMD", n1i = dat$n1i, n2i = dat$n2i, m1i = dat$m1i, m2i = dat$m2i, sd

    # Ln(RR)

lnRR <- escalc(measure = "ROM", n1i = dat$n1i, n2i = dat$n2i, m1i = dat$m1i, m2i = dat$m2i, sd1i
```

- **Model Comparison:**

```r
# Fixed-Effect Model (SMD):
fixed_SMD <- rma(yi = SMD$yi, vi = SMD$vi, method = "FE", data = dat)
fixed_lnRR <- rma(yi = lnRR$yi, vi = lnRR$vi, method = "FE", data = dat)

# random effect models
random_m_SMD <- rma(yi = SMD$yi, vi = SMD$vi, method = "REML", data = dat)
random_m_lnRR <- rma(yi = lnRR$yi, vi = lnRR$vi, method = "REML", data = dat)
```

- **Extraction and comparisons of the results**

```r
# Extract relevant results for comparison
model_comparison <- data.frame(
  Model = c("Fixed Effect (SMD)", "Random Effect (SMD)", "Fixed Effect (lnRR)", "Random Effect (l
  Estimate = c(fixed_SMD$b, random_m_SMD$b, fixed_lnRR$b, random_m_lnRR$b),
  `95% CI Lower` = c(fixed_SMD$ci.lb, random_m_SMD$ci.lb, fixed_lnRR$ci.lb, random_m_lnRR$ci.lb),
  `95% CI Upper` = c(fixed_SMD$ci.ub, random_m_SMD$ci.ub, fixed_lnRR$ci.ub, random_m_lnRR$ci.ub),
  Tau2 = c(NA, random_m_SMD$tau2, NA, random_m_lnRR$tau2),
  I2 = c(NA, random_m_SMD$I2, NA, random_m_lnRR$I2),
  p.value = c(fixed_SMD$pval, random_m_SMD$pval, fixed_lnRR$pval, random_m_lnRR$pval)
)

# Affichage des résultats sous forme de tableau interactif avec reactable
model_comparison
```

```
##                  Model  Estimate X95..CI.Lower X95..CI.Upper       Tau2
## 1   Fixed Effect (SMD) 1.0200088     0.9104184      1.129599         NA
## 2  Random Effect (SMD) 1.2845731     1.0619587      1.507187 0.74922116
## 3  Fixed Effect (lnRR) 0.2088297     0.1981535      0.219506         NA
## 4 Random Effect (lnRR) 0.2552978     0.2164785      0.294117 0.02620568
##         I2      p.value
## 1       NA 2.382399e-74
## 2 69.73831 1.173970e-29
## 3       NA 0.000000e+00
## 4 88.89735 5.133963e-38
```

- **Interpretation:** The contrast in point estimates and confidence intervals between the SMD and lnRR models underscores the impact of effect size metrics on study-level variability. This comparison reveals how different metrics can shift the weight of certain studies and alter pooled estimates.

Exercise: Model comparison

1. **Task**: Fit three different models (Fixed, Random, and Three-Level) using the dataset.
2. **Objective**: Compare their tau² estimates and overall pooled effect sizes.
3. **Guiding questions**:
    - How does heterogeneity (I^2) change across these models?
    - Does the model choice impact the significance of moderators?

2.4.2 Example 2: Comparing Heterogeneity across models

Here, we calculate Risk Differences (RD) and explore the impact of between-study variance estimators (`DL`, `REML`) on heterogeneity.

- **Random-Effects Model Using DerSimonian-Laird (DL)**:

```
res_dl <- rma(yi = lnRR$yi, vi = lnRR$vi, method = "DL", data = dat)
```

- **Random-Effects Model Using REML**:

```
res_reml <- rma(yi = lnRR$yi, vi = lnRR$vi, method = "REML", data = dat)
```

- **Extraction and comparisons of the results**

```
# Extract relevant results for comparison
model_comparison2 <- data.frame(
  Model = c("DL", "REML"),
  Estimate = c(res_dl$b, res_reml$b),
  `95% CI Lower` = c(res_dl$ci.lb, res_reml$ci.lb),
  `95% CI Upper` = c(res_dl$ci.ub, res_reml$ci.ub),
  Tau2 = c(res_dl$tau2, res_reml$tau2),
  I2 = c(res_dl$I2, res_reml$I2),
  p.value = c(res_dl$pval, res_reml$pval)
)

# Affichage des résultats sous forme de tableau interactif avec reactable
model_comparison2
```

```
##   Model  Estimate X95..CI.Lower X95..CI.Upper       Tau2       I2      p.value
## 1    DL 0.2530579     0.2168455     0.2892704 0.02164714 86.86638 1.065093e-42
## 2  REML 0.2552978     0.2164785     0.2941170 0.02620568 88.89735 5.133963e-38
```

Interpretation: The DL estimator is more sensitive to small-study effects, potentially overestimating between-study heterogeneity, while the REML tends to yield more conservative estimates, resulting in narrower confidence intervals. This comparison is crucial for understanding when each method might be more appropriate.

Exercise: Model estimator impact

1. **Task**: Apply multiple heterogeneity estimators (`DL`, `HE`, `SJ`, `REML`) to the dataset.
2. **Objective**: Compare I^2, tau^2, and Q-statistics across estimators.
3. **Guiding Questions**:
    - Which estimator produces the most conservative tau^2 estimate?
    - Does the choice of estimator affect the overall significance?

2.4.3 Example 3: Multi-Level meta-Analysis with hierarchical Data

In hierarchical data structures, three-level models allow us to account for dependencies within and across clusters (e.g., experiments and individual observations).

- **Three-Level model setup**:

```r
dat<-cbind(dat, lnRR)
# Three-level meta-analysis
three_level_m <- rma.mv(yi = yi, V = vi, random = list(~1 | id, ~1 | paper), data = dat)

# Two level meta-analysis
two_level_m <- rma.mv(yi = yi, V = vi, random = list(~1 | id), data = dat)
```

1. Faire le tableau de comparaison de outputs
2. AIC pour sélection de modèles

Interpretation: The three-level model reveals the extent of within- and between-experiment variance. Visualizing these results with orchard plots helps disentangle which levels contribute most to the observed heterogeneity.

Exercise: multi-Level meta-Analysis

1. **Task**: Fit a series of multi-level models with different group-level random effects.
2. **Objective**: Determine how accounting for more complex structures impacts model fit and tau² partitioning.
3. **Guiding Questions**:
    - How does the inclusion of nested random effects change the intraclass correlation?
    - Which levels (experiment vs. within-group) contribute most to the variance?

2.4.4 Model Selection in Meta-Analysis

Choosing the best model in meta-analysis involves comparing multiple competing models using selection criteria like AIC (Akaike Information Criterion), AICc (Corrected AIC), BIC (Bayesian Information Criterion), and other model performance metrics. This helps identify a model that balances goodness-of-fit and model complexity. The following section covers strategies for selecting the most appropriate model, including fixed and random effects, based on theoretical knowledge or statistical tests.

2.4.4.1 Selection Criteria for Model Comparison

- **AIC (Akaike Information Criterion)**: Measures the trade-off between model fit and complexity:

$$\text{AIC} = 2k - 2\log(\hat{L})$$

  where k is the number of model parameters, and $\hat{L}$ is the maximum likelihood value. A lower AIC indicates a more parsimonious model.

- **AICc (Corrected AIC)**: Adjusted for small sample sizes to avoid overfitting:

$$\text{AICc} = \text{AIC} + \frac{2k(k+1)}{n-k-1}$$

  It should be used when n/k is small (< 40), where n is the number of observations and k the number of parameters.

- **BIC (Bayesian Information Criterion)**: Applies a stronger penalty for the number of parameters:

$$\text{BIC} = k\log(n) - 2\log(\hat{L})$$

BIC often favors simpler models, making it suitable when working with large sample sizes.

- **Likelihood Ratio Tests and ANOVA**: To directly compare nested models, likelihood ratio tests or `anova` can be used. This approach quantifies if adding (or removing) parameters significantly improves model fit.

2.4.4.2 Choosing Between Fixed and Random-Effects Models

The decision between using a fixed-effect or random-effects model depends on either *theoretical considerations* or *statistical testing*:

- **Theoretical Justification**: Choose a *fixed-effect model* if you expect a common effect size across all studies (e.g., homogeneous study contexts or interventions). Select a *random-effects model* when heterogeneity among studies is anticipated (e.g., varying contexts, differing methods).
- **Statistical Testing**:
  - **Heterogeneity Tests**: Use tests like Cochran's Q-test or I^2 statistics to detect significant variability between studies. High I^2 or a significant Q-test indicates that a random-effects model is likely more appropriate.
  - **ANOVA for Model Comparison**: When comparing nested models (e.g., fixed-effects vs. random-effects), `anova()` can be used to test whether the inclusion of random effects significantly improves the model fit.
-

2.4.4.3 Comparing Models: Fixed and Random Structures

When fitting mixed-effects models, choosing between different random and fixed structures is a crucial step. Typically, the process involves:

1. **Start with the Random Structure**:
   - Define the random effects first to capture variance due to grouping or clustering (e.g., study-level random effects, site-level random effects).
   - Use the likelihood ratio test (`anova()`) to compare models with and without each random component:

```
model1 <- rma.mv(yi, vi, random = ~ 1 | paper, data = dat)
model2 <- rma.mv(yi, vi, random = ~ 1 | paper/id, data = dat)
anova(model1, model2)
```

```
##
##          df      AIC      BIC     AICc   logLik      LRT   pval       QE
## Full      3  -8.2204  -0.3750  -7.9730   7.1102                769.0185
## Reduced   2 104.0570 109.2872 104.1794 -50.0285 114.2774 <.0001 769.0185
```

This step allows you to determine if additional random components (e.g., study vs. nested study/site effects) are justified.

Here, `anova()` will test if the more complex model (`model_random`) explains significantly more variance than the simpler one (`model_fixed`).

2. **Incorporate Fixed Effects**:

   - After selecting the optimal random structure, test different combinations of fixed effects (e.g., covariates, moderators).
   - Compare models using AIC, AICc, and BIC. Include one fixed effect at a time and use `anova()` to see if adding predictors improves fit:

```
model_fixed1 <- rma.mv(yi, vi, mods = ~ species, random= ~ 1 | paper/id, data
model_fixed2 <- rma.mv(yi, vi, mods = ~ species + fungrp,random= ~ 1 | paper/
anova(model_fixed1, model_fixed2)
```

```
##
##          df     AIC      BIC     AICc logLik    LRT   pval       QE
## Full     38 73.7811 156.9879 183.5588 1.1095              334.3528
## Reduced  37 71.3274 152.9011 168.2930 1.3363 0.0000 1.0000 334.9388
```

3. **Test for Interactions**:

   - After defining main effects, consider adding interaction terms and use AIC or likelihood ratio tests to assess their contribution.

2.4.4.4 Selecting the Optimal Model: Practical Considerations

- **Model Complexity vs. Parsimony**: Start with a simple model and gradually add parameters, using AIC, AICc, and BIC to avoid overfitting.
- **Check Model Assumptions**: Ensure that the chosen model meets meta-analytic assumptions (e.g., homogeneity of variance, normality).
- **Visual Evaluation**: Use diagnostic plots (e.g., residual plots, forest plots) to visually inspect model fit.

In practice, the ideal strategy involves:

1. **Define the Random Structure First**:

   - Use theoretical and statistical justifications to select between study-level or multi-level random effects.

- Select the most parsimonious random structure using `anova()` and AIC/BIC.

2. **Add Fixed Effects Incrementally**:
   - Start with primary predictors and gradually include covariates, using `anova()` to compare nested models.
   - Check for interactions only after establishing main effects.
3. **Assess the Final Model**:
   - Compare the best candidate models (fixed vs. random) using AIC, AICc, BIC, and likelihood ratio tests.
   - Select the model that balances fit and interpretability.

This structured approach ensures that both theoretical considerations and statistical criteria are employed for robust model selection in meta-analyses.

Chapter 3

Analysing model variability

3.0.1 Using Parametric and Non-Parametric Bootstrapping to Estimate Confidence Intervals in Meta-Analyses

Bootstrapping is a statistical technique used to estimate the sampling distribution of an estimator by repeatedly resampling from the observed data. In meta-analysis, bootstrapping is commonly applied to construct more robust confidence intervals (CIs) for parameters such as the overall effect size ( ) and the between-study variance (2 ^2 2). This chapter will demonstrate how to implement both parametric and non-parametric bootstrapping methods using the `boot` package in R, emphasizing best practices for obtaining reliable results.

- **Parametric Bootstrapping**: Assumes a specific distributional form for the data (e.g., normal distribution of residuals). The data are generated based on parameter estimates from the fitted model.
- **Non-Parametric Bootstrapping**: Does not rely on specific distributional assumptions. The bootstrap samples are created directly by resampling the original data with replacement.

3.0.2 Parametric Bootstrapping: Step-by-Step Example

In parametric bootstrapping, we generate new datasets by simulating values from a specified distribution using the parameter estimates from the fitted model. This process requires defining two functions:

1. A function to compute the statistics of interest (e.g., the mean effect size and the between-study variance 2 ^2 2) based on the bootstrap data.

2. A function to generate the bootstrap datasets.

Let's illustrate this approach with a random-effects model:

Defining the Function

The function `boot.func()` calculates the effect size and variance components based on each bootstrap dataset:

```
# Load necessary libraries
library(metafor)
library(boot)


# 1. Fit the initial random-effects model
initial_model <- rma(yi, vi, data=dat)

# Extract estimated parameters for later use
mu_estimate <- coef(initial_model)
tau2_estimate <- initial_model$tau2

# 2. Define the Statistic Function for Bootstrapping
boot.func <- function(data.boot) {
  # Fit the random-effects model to the bootstrap data
  res <- try(suppressWarnings(rma(yi, vi, data=data.boot)), silent=TRUE)

  # Return NA if the model did not converge
  if (inherits(res, "try-error")) {
    return(rep(NA, 4))  # Return a vector of NAs
  } else {
    # Extract the estimated effect size (mu), its variance, tau², and its variance
    return(c(coef(res), diag(vcov(res)), res$tau2, res$se.tau2^2))
  }
}

# 3. Define the Data Generation Function for Bootstrapping
data.gen <- function(dat, mle) {
  # Generate effect sizes based on the estimated mu and tau²
  data.frame(yi = rnorm(nrow(dat), mle$mu, sqrt(mle$tau2 + dat$vi)), vi = dat$vi)
}

# 4. Running the Parametric Bootstrap
set.seed(1234)  # For reproducibility
res.boot <- boot::boot(dat,
                       statistic = boot.func,
                       R = 100,
```

```
                        sim = "parametric",
                        ran.gen = data.gen,
                        mle = list(mu = mu_estimate, tau2 = tau2_estimate))

# Check results
print(res.boot)
```

```
##
## PARAMETRIC BOOTSTRAP
##
##
## Call:
## boot::boot(data = dat, statistic = boot.func, R = 100, sim = "parametric",
##     ran.gen = data.gen, mle = list(mu = mu_estimate, tau2 = tau2_estimate))
##
##
## Bootstrap Statistics :
##         original        bias     std. error
## t1* 2.552978e-01  1.086396e-03 1.908634e-02
## t2* 3.922816e-04 -9.255823e-06 5.149671e-05
## t3* 2.620568e-02 -7.892906e-04 4.629509e-03
## t4* 2.819226e-05 -8.527037e-07 7.784439e-06
```

Extracting Confidence Intervals

After running the bootstrap, we can calculate the confidence intervals for the mean effect size ( ) and the between-study variance (2 ^2 2) using different bootstrap methods (normal, basic, studentized, percentile):

```
# Confidence intervals for mu
boot.ci(res.boot, type=c("norm", "basic", "stud", "perc"), index=1:2)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 100 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = res.boot, type = c("norm", "basic", "stud",
##     "perc"), index = 1:2)
##
## Intervals :
## Level      Normal              Basic
## 95%   ( 0.2168,  0.2916 )   ( 0.2137,  0.2933 )
##
## Level    Studentized          Percentile
```

```
## 95%   ( 0.2123,  0.2956 )   ( 0.2173,  0.2969 )
## Calculations and Intervals on Original Scale
## Some basic intervals may be unstable
## Some studentized intervals may be unstable
## Some percentile intervals may be unstable
```

```
# Confidence intervals for tau^2
boot.ci(res.boot, type=c("norm", "basic", "stud", "perc"), index=3:4)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 100 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = res.boot, type = c("norm", "basic", "stud",
##     "perc"), index = 3:4)
##
## Intervals :
## Level      Normal              Basic
## 95%   ( 0.0179,  0.0361 )   ( 0.0160,  0.0373 )
##
## Level    Studentized          Percentile
## 95%   ( 0.0184,  0.0432 )   ( 0.0152,  0.0364 )
## Calculations and Intervals on Original Scale
## Some basic intervals may be unstable
## Some studentized intervals may be unstable
## Some percentile intervals may be unstable
```

These commands compute CIs based on various bootstrap methods, allowing comparison of the interval estimates.

3.0.3 Non-Parametric Bootstrapping: Step-by-Step Example

Non-parametric bootstrapping involves generating new datasets by resampling the original data with replacement. We only need to define a single function for this purpose:

Defining the Function

```
# Load necessary libraries
library(metafor)
library(boot)
```

```r
# 1. Fit the initial random-effects model
initial_model <- rma(yi, vi, data=dat)

# Extract estimated parameters for later use
mu_estimate <- coef(initial_model)
tau2_estimate <- initial_model$tau2

# 2. Define the Statistic Function for Bootstrapping
boot.func <- function(data.boot, indices) {
   # Resample the data based on the given indices
   sel <- data.boot[indices, ]

   # Fit the random-effects model to the resampled data
   res <- try(suppressWarnings(rma(yi, vi, data=sel)), silent=TRUE)

   # Return NA if the model did not converge
   if (inherits(res, "try-error")) {
      return(rep(NA, 4))  # Return a vector of NAs
   } else {
      # Extract the estimated effect size (mu), its variance, tau², and its variance
      return(c(coef(res), diag(vcov(res)), res$tau2, res$se.tau2^2))
   }
}

# 3. Define the Data Generation Function for Bootstrapping
data.gen <- function(dat, mle) {
   # Generate effect sizes based on the estimated mu and tau²
   data.frame(yi = rnorm(nrow(dat), mle$mu, sqrt(mle$tau2 + dat$vi)), vi = dat$vi)
}

# 4. Running the Parametric Bootstrap
set.seed(1234)  # For reproducibility
res.boot <- boot::boot(dat,
                       statistic = boot.func,
                       R = 100,
                       sim = "parametric",
                       ran.gen = data.gen,
                       mle = list(mu = mu_estimate, tau2 = tau2_estimate))

# Check results
print(res.boot)
```

```
##
## PARAMETRIC BOOTSTRAP
##
```

```
##
## Call:
## boot::boot(data = dat, statistic = boot.func, R = 100, sim = "parametric",
##     ran.gen = data.gen, mle = list(mu = mu_estimate, tau2 = tau2_estimate))
##
##
## Bootstrap Statistics :
##         original        bias     std. error
## t1* 2.552978e-01  1.086396e-03 1.908634e-02
## t2* 3.922816e-04 -9.255823e-06 5.149671e-05
## t3* 2.620568e-02 -7.892906e-04 4.629509e-03
## t4* 2.819226e-05 -8.527037e-07 7.784439e-06
```

Extracting Confidence Intervals

After running the bootstrap, we can calculate the confidence intervals for the mean effect size ( ) and the between-study variance (2 ^2 2) using different bootstrap methods (normal, basic, studentized, percentile):

```
# Confidence intervals for mu
boot.ci(res.boot, type=c("norm", "basic", "stud", "perc"), index=1:2)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 100 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = res.boot, type = c("norm", "basic", "stud",
##     "perc"), index = 1:2)
##
## Intervals :
## Level      Normal              Basic
## 95%   ( 0.2168,  0.2916 )   ( 0.2137,  0.2933 )
##
## Level    Studentized          Percentile
## 95%   ( 0.2123,  0.2956 )   ( 0.2173,  0.2969 )
## Calculations and Intervals on Original Scale
## Some basic intervals may be unstable
## Some studentized intervals may be unstable
## Some percentile intervals may be unstable
```

```
# Confidence intervals for tau²
boot.ci(res.boot, type=c("norm", "basic", "stud", "perc"), index=3:4)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 100 bootstrap replicates
```

```
##
## CALL :
## boot.ci(boot.out = res.boot, type = c("norm", "basic", "stud",
##     "perc"), index = 3:4)
##
## Intervals :
## Level      Normal              Basic
## 95%   ( 0.0179,  0.0361 )   ( 0.0160,  0.0373 )
##
## Level    Studentized          Percentile
## 95%   ( 0.0184,  0.0432 )   ( 0.0152,  0.0364 )
## Calculations and Intervals on Original Scale
## Some basic intervals may be unstable
## Some studentized intervals may be unstable
## Some percentile intervals may be unstable
```

Comparing Bootstrap Methods

The choice between parametric and non-parametric bootstrapping depends on the underlying assumptions and the sample size:

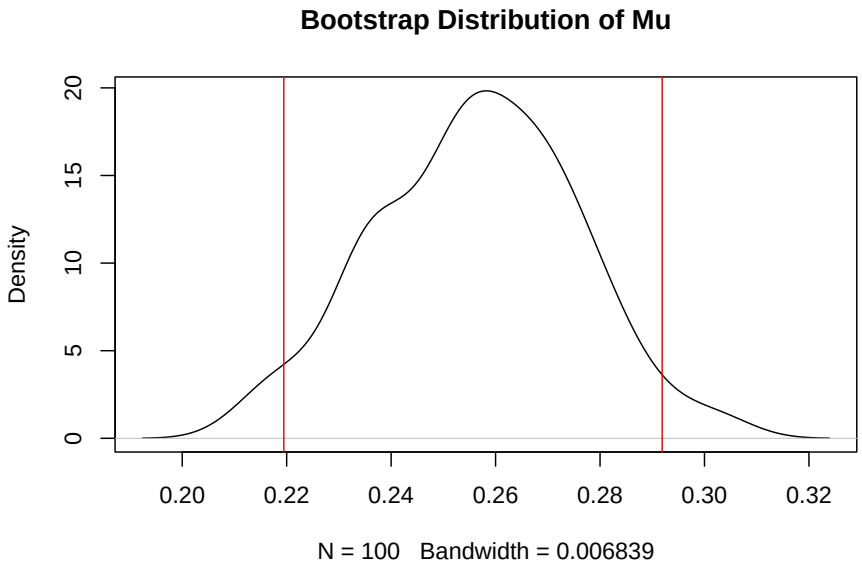
- **Parametric Bootstrapping** is more appropriate when we have strong assumptions about the distribution of effect sizes.

- **Non-Parametric Bootstrapping** is recommended when the sample size is relatively small or when we want to avoid making strict assumptions.

Visualization of Bootstrap Distributions

To visualize the bootstrap distributions, we can create kernel density plots to inspect the variability and shape of the bootstrap samples:

```r
# Visualize the bootstrap distribution for mu
plot(density(res.boot$t[,1]), main="Bootstrap Distribution of Mu")
abline(v=quantile(res.boot$t[,1], probs=c(0.025, 0.975)), col="red")
```



3.0.4 Conclusion

Bootstrapping provides a flexible and powerful method for estimating confidence intervals in meta-analysis. However, it is important to consider potential issues such as non-convergence of models during the bootstrap process, as well as the assumptions underlying each method. When applying bootstrap methods, carefully evaluate the coverage of different interval types and compare the results to standard methods (e.g., Wald-type CIs or Knapp-Hartung adjustments).

In the next section, we will delve deeper into **prediction intervals**, which provide a complementary measure of uncertainty, reflecting the variability in effect sizes for new studies.

Chapter 4

Plotting meta-analytical results

4.0.1 Overview

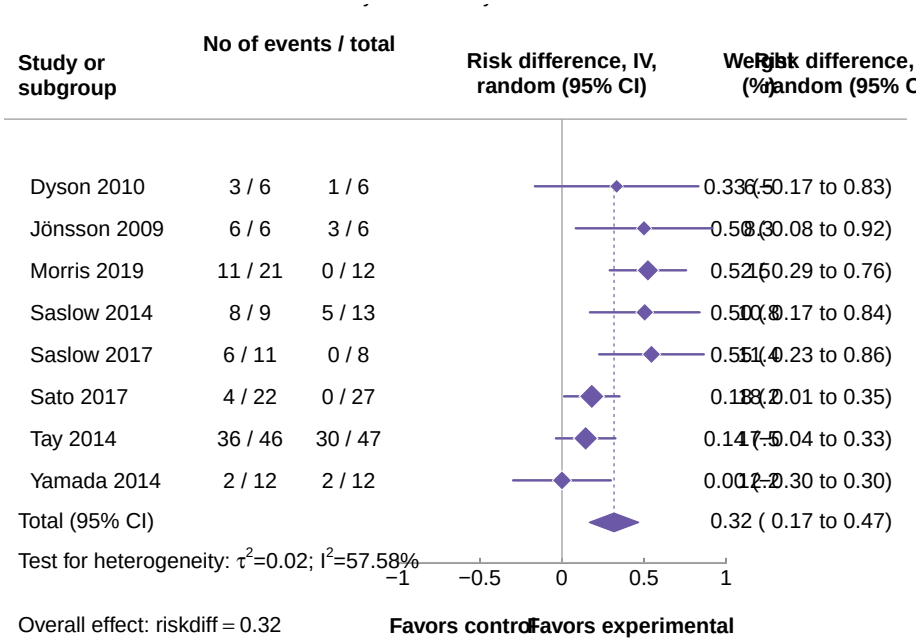
This chapter focuses on the visualization of meta-analytical results, which is crucial for effectively communicating findings and insights drawn from aggregated data. Visualization techniques help researchers and stakeholders quickly grasp complex relationships and patterns within the data. In this chapter, we will cover various plotting methods, including forest plots, funnel plots, and advanced visualizations using R packages such as `ggplot2`, `metafor`, and `dmetar`.

4.0.2 Example 1: Forest plot with metafor

```r
dat <- data.frame(author = c("Dyson", "Jönsson", "Morris", "Saslow", "Saslow", "Sato", "Tay", "Ya
                  year   = c(2010, 2009, 2019, 2014, 2017, 2017, 2014, 2014),
                  ai     = c(3, 6, 11, 8, 6, 4, 36, 2), ##Nb event experimental group
                  n1i    = c(6, 6, 21, 9, 11, 22, 46, 12), ## Total experimental group
                  ci     = c(1, 3, 0, 5, 0, 0, 30, 2),  ## Nb events control group
                  n2i    = c(6, 6, 12, 13, 8, 27, 47, 12))  ## Total control group

### calculate risk differences and corresponding sampling variances (and use
### the 'slab' argument to store study labels as part of the data frame)
dat <- escalc(measure = "RD", ai = ai, n1i = n1i, ci = ci, n2i = n2i, data = dat,
              slab = paste(" ", author, year), addyi = FALSE)

### fit random-effects model (using the DL estimator)
res <- rma(yi, vi, data = dat, method = "DL")
```

```
### colors to be used in the plot
colp <- "#6b58a6"
coll <- "#a7a9ac"

### total number of studies
k <- nrow(dat)

### generate point sizes
psize <- weights(res)
psize <- 1.2 + (psize - min(psize)) / (max(psize) - min(psize))

### get the weights and format them as will be used in the forest plot
weights <- round(weights(res), 1)

### adjust the margins
par(mar = c(2.7, 3.2, 2.3, 1.3), mgp = c(3, 0, 0), tcl = 0.15)

### forest plot with extra annotations
sav <- forest(dat$yi, dat$vi, xlim = c(-3.4, 2.1), ylim = c(-0.5, k + 3), alim = c(-1,
              pch = 18, psize = psize, efac = 0, refline = NA, lty = c(1, 0), xlab = "
              ilab = cbind(paste(dat$ai, "/", dat$n1i), paste(dat$ci, "/", dat$n2i), we
              ilab.xpos = c(-1.9, -1.3, 1.2), annosym = c(" (", " to ", ")"),
              rowadj = -0.07)

### add the vertical reference line at 0
segments(0, -1, 0, k + 1.6, col = coll)

### add the vertical reference line at the pooled estimate
segments(coef(res), 0, coef(res), k, col = colp, lty = "33", lwd = 0.8)

### redraw the CI lines and points in the chosen color
segments(summary(dat)$ci.lb, k:1, summary(dat)$ci.ub, k:1, col = colp, lwd = 1.5)
points(dat$yi, k:1, pch = 18, cex = psize * 1.15, col = "white")
points(dat$yi, k:1, pch = 18, cex = psize, col = colp)

### add the summary polygon
addpoly(res, row = 0, mlab = "Total (95% CI)", efac = 2, col = colp, border = colp)

### add the horizontal line at the top
abline(h = k + 1.6, col = coll)

### redraw the x-axis in the chosen color
axis(side = 1, at = seq(-1, 1, by = 0.5), col = coll, labels = FALSE)

### now we add a bunch of text; since some of the text falls outside of the
```

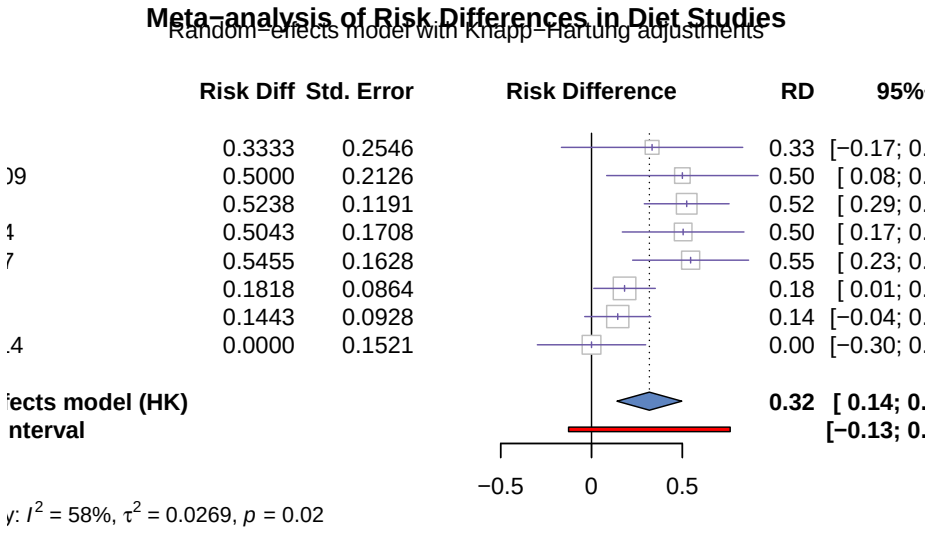
```r
### plot region, we set xpd=NA so nothing gets clipped
par(xpd = NA)

### adjust cex as used in the forest plot and use a bold font
par(cex = sav$cex, font = 2)

### add headings
text(sav$xlim[1], k + 2.5, pos = 4, "Study or\nsubgroup")
text(mean(sav$ilab.xpos[1:2]), k + 3.4, "No of events / total")
text(0, k + 2.7, "Risk difference, IV,\nrandom (95% CI)")
text(c(sav$ilab.xpos[3], sav$xlim[2] - 0.35), k + 2.7, c("Weight\n(%)", "Risk difference, IV,\nra

### add 'Favours control'/'Favours experimental' text below the x-axis
text(c(-1, 1), -2.5, c("Favors control", "Favors experimental"), pos = c(4, 2), offset = -0.3)

### use a non-bold font for the rest of the text
par(cex = sav$cex, font = 1)

### add text with heterogeneity statistics
text(sav$xlim[1], -1, pos = 4, bquote(paste("Test for heterogeneity: ",
                                            tau^2, "=", .(round(res$tau2, digits = 2)), "; ",
                                            I^2, "=", .(round(res$I2, digits = 2)), "%")))

### add text with overall estimate
text(sav$xlim[1], -2.5, pos = 4, bquote(paste("Overall effect: ",
                                              riskdiff == .(round(coef(res), digits = 2)))))

### add titles
text(sav$xlim[1], k + 4.5, pos = 4, "Risk difference between low and very low carbohydrate diets'
```



4.0.3 Example 2: Forest plot with meta

```
library(meta)
# Perform meta-analysis using the metagen function from `meta`
# Perform meta-analysis using the metagen function from `meta`
meta_analysis <- metagen(
  TE = dat$yi,  # Use the calculated effect sizes
  seTE = sqrt(dat$vi),  # Standard errors
  studlab = paste(dat$author, dat$year),
  data = dat,
  sm = "RD",  # Risk Difference as the summary measure
  fixed = FALSE,  # Using a random-effects model
  random = TRUE,
  method.tau = "REML",  # Restricted maximum likelihood estimator
  hakn = TRUE,  # Knapp-Hartung adjustment
  title = "Risk Difference between Diet Groups"
)

# Create a forest plot with custom options and store the output
forest_plot <- meta::forest(
  meta_analysis,
  prediction = TRUE,  # Add a prediction interval
  print.tau2 = TRUE,  # Print the tau-squared statistic in the plot
```

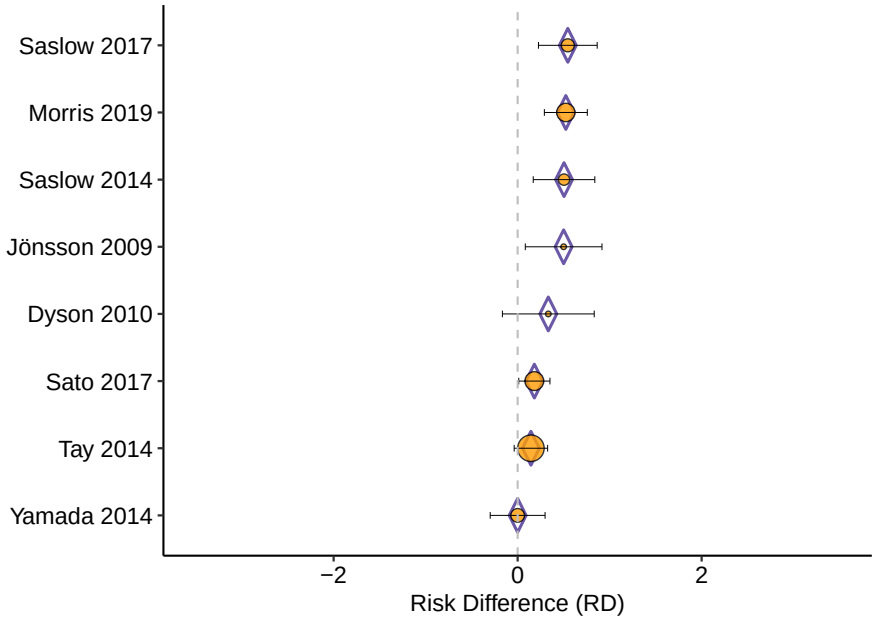
```
  leftlabs = c("Study", "Risk Diff", "Std. Error"),  # Custom labels
  lab.e = "Favours Low Carb",  # Label for experimental group
  lab.c = "Favours Control",  # Label for control group
  col.study = "#6b58a6",  # Study point color
  col.diamond = "#5e84c0",  # Color of the summary diamond
  col.square = "white"  # Color for individual study estimates
)

library(grid)
grid.text("Meta-analysis of Risk Differences in Diet Studies",
          x = 0.5, y = unit(0.95, "npc"),
          gp = gpar(fontsize = 14, fontface = "bold"))
grid.text("Random-effects model with Knapp-Hartung adjustments",
          x = 0.5, y = unit(0.93, "npc"),
          gp = gpar(fontsize = 12))
```



4.0.4 Example 3: Forest plot with ggplot

```
# Load necessary libraries
library(ggplot2)
library(dplyr)
library(ggpubr)
```

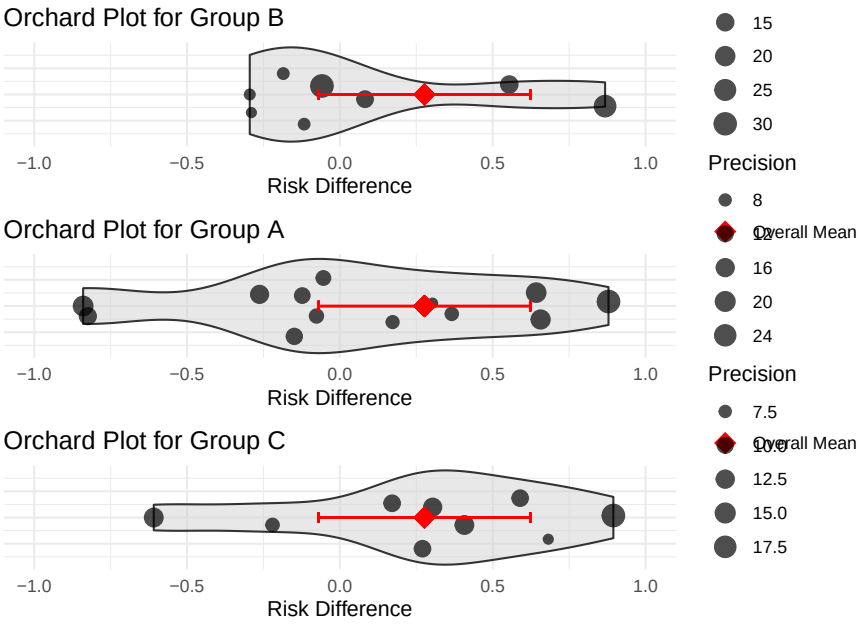
```r
library(ggstar)

# Prepare the data: Calculate estimates, confidence intervals, and other metrics
dat <- dat %>%
  mutate(
    estimate = yi,  # Estimated effect size
    conf.low = yi - 1.96 * sqrt(vi),  # Lower bound of the confidence interval
    conf.high = yi + 1.96 * sqrt(vi),  # Upper bound of the confidence interval
    Sub_Cat_intervention = paste(author, year)  # Combine author and year for labels
  )

# Create a ggplot forest plot
p <- ggplot(dat, aes(estimate, reorder(Sub_Cat_intervention, estimate))) +
  # Add stars for effect sizes
  geom_star(starshape = 12, size = 4, starstroke = 1.1, color = "#6b58a6") +
  # Add points for individual study estimates
  geom_point(aes(size = n1i), shape = 21, alpha = 0.8, fill = "#ff9d02",
             position = position_dodge(width = 0.6)) +
  # Add error bars using confidence intervals
  geom_errorbar(aes(xmin = conf.low, xmax = conf.high), color = "black",
                size = 0.2, width = 0.1) +
  # Add a vertical line at x = 0
  geom_vline(xintercept = 0, linetype = 2, color = "gray") +
  # Apply ggpubr theme
  theme_pubr() +
  # Set x-axis limits
  xlim(c(-3.5, 3.5)) +  # Adjust limits based on your data
  # Remove legend
  theme(legend.position = "none") +
  # Set axis labels
  labs(y = "", x = "Risk Difference (RD)")

# Display the plot
print(p)
```



4.0.5 Example 4: Orchard plots

```r
# Load required libraries
library(ggplot2)
library(dplyr)
library(metafor)
library(ggbeeswarm)

# Create a larger randomized dataset with valid constraints
set.seed(42)  # For reproducibility
n_studies <- 30  # Total number of studies

dat <- data.frame(
  author = paste("Study", 1:n_studies),
  year = sample(2000:2020, n_studies, replace = TRUE),
  n1i = sample(10:100, n_studies, replace = TRUE),  # Total in experimental group
  n2i = sample(10:100, n_studies, replace = TRUE)   # Total in control group
)

# Randomly generate events ensuring that events do not exceed group sizes
dat$ai <- sapply(1:n_studies, function(i) sample(0:dat$n1i[i], 1))  # Events in experimental grou
dat$ci <- sapply(1:n_studies, function(i) sample(0:dat$n2i[i], 1))  # Events in control group
```

```r
# Calculate risk differences
dat <- escalc(measure = "RD", ai = ai, n1i = n1i, ci = ci, n2i = n2i, data = dat)

# Fit a random-effects model and create a subgroup variable
dat$subgroup <- sample(c("Group A", "Group B", "Group C"), n_studies, replace = TRUE)

# Create an empty plot list to store individual subgroup plots
plot_list <- list()

# Loop over each subgroup to create the overall mean for each
for (group in unique(dat$subgroup)) {
  subgroup_data <- dat[dat$subgroup == group, ]
  res <- rma(yi, vi, data = subgroup_data, method = "DL")  # Fit model for subgroup

  # Create the orchard plot for each subgroup
  p <- ggplot(subgroup_data, aes(x = yi, y = 1)) +
    geom_violin(fill = "lightgray", alpha = 0.5) +
    geom_quasirandom(aes(size = 1/sqrt(vi)), alpha = 0.7) +  # Use inverse of SE for s
    geom_point(aes(x = res$b, y = 1, color = "Overall Mean"), size = 5, shape = 18) +
    geom_errorbarh(aes(xmin = res$ci.lb, xmax = res$ci.ub, y = 1), height = 0.1, color
    labs(x = "Risk Difference", y = "", title = paste("Orchard Plot for", group)) +
    scale_color_manual(name = "", values = c("Overall Mean" = "red")) +
    theme_minimal() +
    theme(axis.text.y = element_blank(), axis.ticks.y = element_blank()) +
    xlim(c(-1, 1)) +
    scale_size(range = c(2, 5), name = "Precision")

  # Store the plot in the list
  plot_list[[group]] <- p
}

# Display all orchard plots for each subgroup
library(gridExtra)
do.call(grid.arrange, c(plot_list, ncol = 1))  # Arrange plots in a single column
```


Orchard Plot for Group B
−1.0
−0.5
0.0
0.5
1.0
Risk Difference
Orchard Plot for Group A
−1.0
−0.5
0.0
0.5
1.0
Risk Difference
Orchard Plot for Group C
−1.0
−0.5
0.0
0.5
1.0
Risk Difference
15
20
25
30
Precision
8
12
Overall Mean
16
20
24
Precision
7.5
10.0
Overall Mean
12.5
15.0
17.5

Chapter 5

Analysis and Visualization of Publication Bias

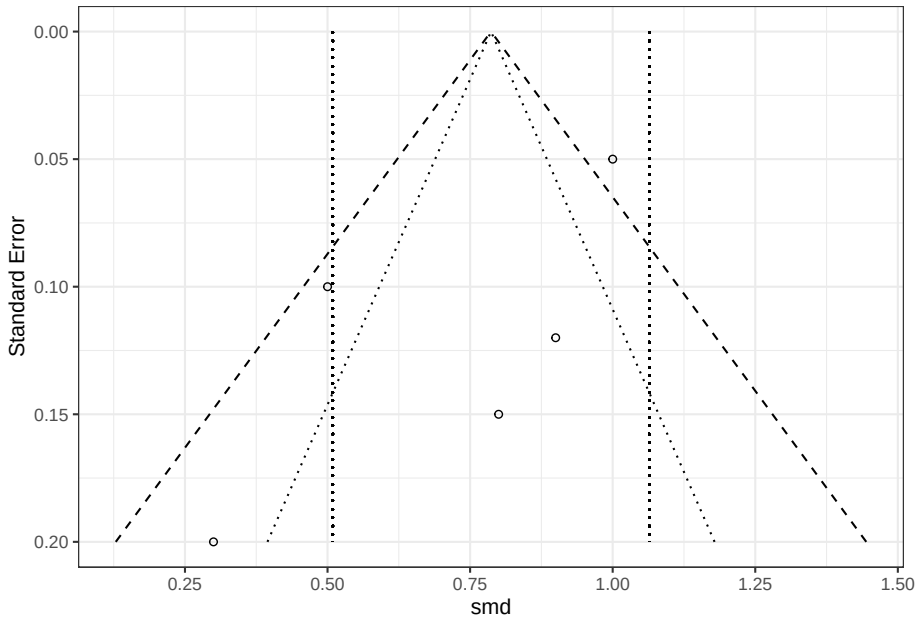
Publication bias is a significant concern in meta-analysis, as it can distort the overall effect estimate and lead to misleading conclusions. In this section, we will discuss various methods for detecting and visualizing publication bias, including funnel plots, the *trim and fill* method, the Egger test, and Robust Bayesian Meta-Analysis.

5.0.1 Funnel Plots

A funnel plot is a scatter plot that helps visualize the relationship between the effect size and the precision of individual studies in a meta-analysis. It is an essential tool for identifying potential publication bias.

```
# Load necessary libraries
library(metafor)
library(ggplot2)

# Create a data frame with example data
data <- data.frame(
  study = c("Study1", "Study2", "Study3", "Study4", "Study5"),
  smd = c(0.5, 0.8, 0.3, 1.0, 0.9),
  var_smd = c(0.1, 0.15, 0.2, 0.05, 0.12)
)

# Conduct a random-effects meta-analysis
res <- rma(yi = smd, vi = var_smd, data = data)
```

```r
estimate <- res$b  # Replace with your meta-analytic estimate if needed
se <- res$se        # Replace with your standard error if needed

# Define a sequence for standard errors (SE) to create CI regions
se.seq <- seq(0, max(data$var_smd), 0.001)

# Calculate 95% and 99% confidence intervals for the region
ll95 <- estimate - (1.96 * se.seq)
ul95 <- estimate + (1.96 * se.seq)
ll99 <- estimate - (3.29 * se.seq)
ul99 <- estimate + (3.29 * se.seq)

# Compute confidence intervals for the mean meta-analytic estimate
meanll95 <- estimate - (1.96 * se)
meanul95 <- estimate + (1.96 * se)

# Store all calculated values in a single data frame for easy plotting
dfCI <- data.frame(ll95, ul95, ll99, ul99, se.seq, estimate, meanll95, meanul95)


# Draw the enhanced funnel plot with shaded CI regions and lines
funnel_plot <- ggplot(aes(x = var_smd, y = smd), data = data)+
  # Add shaded areas for 95% and 99% CI regions
  geom_point(shape = 1) +
  xlab('Standard Error') + ylab('smd')+
  geom_line(aes(x = se.seq, y = ll95), linetype = 'dotted', data = dfCI) +
  geom_line(aes(x = se.seq, y = ul95), linetype = 'dotted', data = dfCI) +
  geom_line(aes(x = se.seq, y = ll99), linetype = 'dashed', data = dfCI) +
  geom_line(aes(x = se.seq, y = ul99), linetype = 'dashed', data = dfCI) +
  geom_segment(aes(x = min(se.seq), y = meanll95, xend = max(se.seq), yend = meanll95)
  geom_segment(aes(x = min(se.seq), y = meanul95, xend = max(se.seq), yend = meanul95)
  scale_x_reverse()+
  scale_y_continuous(breaks=seq(-1.25,2,0.25))+
  coord_flip()+
  theme_bw()

# Display the enhanced funnel plot
funnel_plot
```



5.0.1.1 Understanding Funnel Plots

- **Structure**: In an ideal scenario, a funnel plot resembles an inverted funnel or a pyramid. This shape reflects that larger studies tend to be more precise (higher up on the plot), while smaller studies have more variability (scattered at the bottom).
- **Axes**:
  - The **x-axis** represents the estimated effect size for each study (e.g., risk ratios, odds ratios plotted on a logarithmic scale, or mean differences).
  - The **y-axis** indicates the study precision, often measured by the standard error. Larger studies with greater precision are displayed at the top.

In a bias-free scenario, 95% of studies would be expected to lie within the dashed lines representing the 95% confidence interval.

Interpreting Funnel Plots and Reasons for Asymmetry

When analyzing a funnel plot, look for symmetry. A symmetrical plot suggests that publication bias is unlikely, while an asymmetrical plot may indicate bias.

Asymmetry in a funnel plot can arise from several factors:

- **Non-reporting Bias**: Studies with non-significant results may be less likely to be published.
- **Methodological Quality**: Studies with poor design may report exaggerated effect sizes.
- **True Heterogeneity**: Variability in patient populations can lead to biased estimates, especially in smaller studies.
- **Artefactual Correlation**: Correlations between effect estimates and their standard errors can create false asymmetry.
- **Chance**: Random variation is more likely in analyses with fewer studies.

Methods such as the Precision Effect Test (PET) and Precision Effect Estimation with Standard Error (PEESE) have been developed to help researchers better assess and adjust for publication bias.

- **PET** evaluates the relationship between effect sizes and their standard errors, positing that smaller studies (often unpublished) will display larger effect sizes due to the likelihood of publication. By fitting a regression model where effect sizes are regressed on their standard errors, researchers can determine whether a systematic bias exists. If the regression shows a significant slope, it suggests that publication bias is present, as smaller studies are showing disproportionately large effects.
- **PEESE** builds on the insights from PET by providing a more nuanced approach to estimate the true effect size. It uses both the effect sizes and their standard errors, allowing for a more accurate adjustment of the effect size that accounts for the potential bias introduced by smaller studies. PEESE calculates an adjusted effect size by modeling the observed effect sizes and their variances, yielding a refined estimate that is less susceptible to the biases highlighted in PET.

5.0.2 Trim and Fill Method

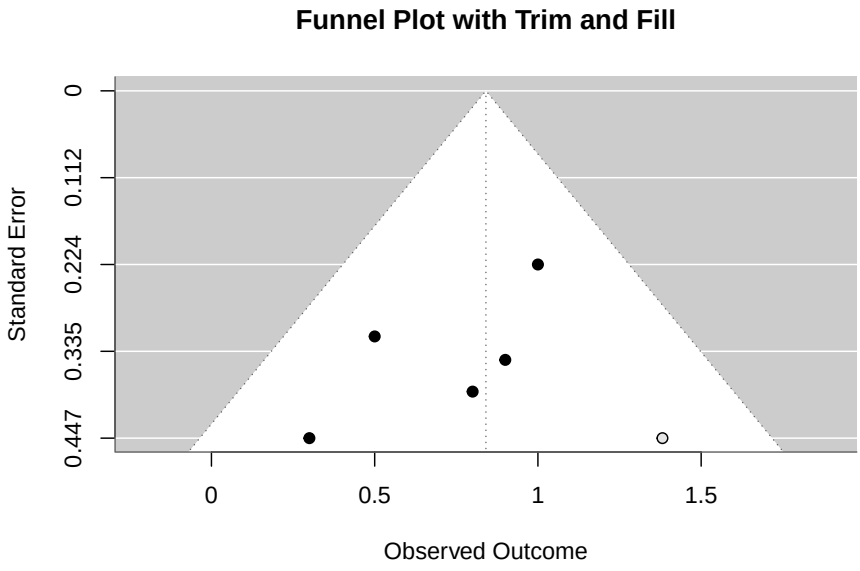
The *trim and fill* method adjusts for publication bias by estimating the number of studies that would need to be added to achieve symmetry. This method imputes missing studies to provide a more accurate effect size.

5.0.2.1 Example of Using Trim and Fill

```r
# Perform trim and fill analysis
trimmed_res <- trimfill(res)

# Summary of the results
summary(trimmed_res)
```

```
##
## Estimated number of missing studies on the right side: 1 (SE = 1.7009)
##
## Random-Effects Model (k = 6; tau^2 estimator: REML)
##
##   logLik  deviance       AIC       BIC      AICc
##  -1.7515    3.5031    7.5031    6.7219   13.5031
##
## tau^2 (estimated amount of total heterogeneity): 0.0000 (SE = 0.0673)
## tau (square root of estimated tau^2 value):      0.0022
## I^2 (total heterogeneity / total variability):   0.00%
## H^2 (total variability / sampling variability):  1.00
##
## Test for Heterogeneity:
## Q(df = 5) = 4.6326, p-val = 0.4623
##
## Model Results:
##
## estimate      se    zval    pval   ci.lb   ci.ub
##   0.8407  0.1348  6.2349  <.0001  0.5764  1.1050  ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Create a funnel plot with imputed studies
funnel(trimmed_res, main = "Funnel Plot with Trim and Fill")
```



In this example, the `trimfill` function from the `metafor` package adjusts the meta-analysis results for publication bias.

5.0.3 Egger Test

The Egger test statistically assesses funnel plot asymmetry. A significant result indicates potential publication bias.

5.0.3.1 Conducting the Egger Test

```
# Conduct the Egger test
egger_test <- regtest(res)

# Display the test results
print(egger_test)
```

```
##
## Regression Test for Funnel Plot Asymmetry
##
## Model:     mixed-effects meta-regression model
## Predictor: standard error
##
```

```
## Test for Funnel Plot Asymmetry: z = -1.2521, p = 0.2105
## Limit Estimate (as sei -> 0):   b =  1.4949 (CI: 0.3521, 2.6376)
```

5.0.4 Rosenthal's Fail-Safe N

Rosenthal's Fail-Safe N calculates the number of unpublished studies with null results needed to invalidate the overall effect. A high Fail-Safe N suggests that publication bias is unlikely to significantly impact the findings.

5.0.4.1 Example of Calculating Fail-Safe N

```r
# Calculate the Fail-Safe N
failsafe_n <- fsn(res)
print(failsafe_n)
```

```
##
## Fail-safe N Calculation Using the General Approach
##
## Average Effect Size:         0.7867 (with file drawer: 0.2950)
## Amount of Heterogeneity:     0.0000 (with file drawer: 0.1641)
## Observed Significance Level: <.0001 (with file drawer: 0.0493)
## Target Significance Level:   0.05
##
## Fail-safe N: 8
```

5.0.5 Robust Bayesian Meta-Analysis (RoBMA)

Robust Bayesian Meta-Analysis (RoBMA) is an advanced method designed to address the limitations of traditional meta-analysis by incorporating information on publication bias, heterogeneity, and effect sizes in a single model. It relies on Bayesian principles to estimate the probability of the presence of a real effect while considering the potential for biases and study variability. RoBMA extends the traditional meta-analytic framework by integrating multiple models that account for different sources of uncertainty. It not only provides point estimates of effect sizes but also outputs the probability that publication bias and heterogeneity are influencing the results. This allows for a more transparent and reliable evaluation of the evidence base. Specifically, RoBMA offers:

- **Probabilistic conclusions**: Instead of simple yes/no results, RoBMA provides probabilities for the presence of effects, heterogeneity, and bias.

- **Robust model averaging**: It combines estimates across different models, reducing sensitivity to outliers and small study effects.
- **Enhanced diagnostics**: RoBMA provides visual tools to detect and correct for bias, making it easier to understand and communicate findings.

5.0.5.1 Setting Up and Running a RoBMA Model in R

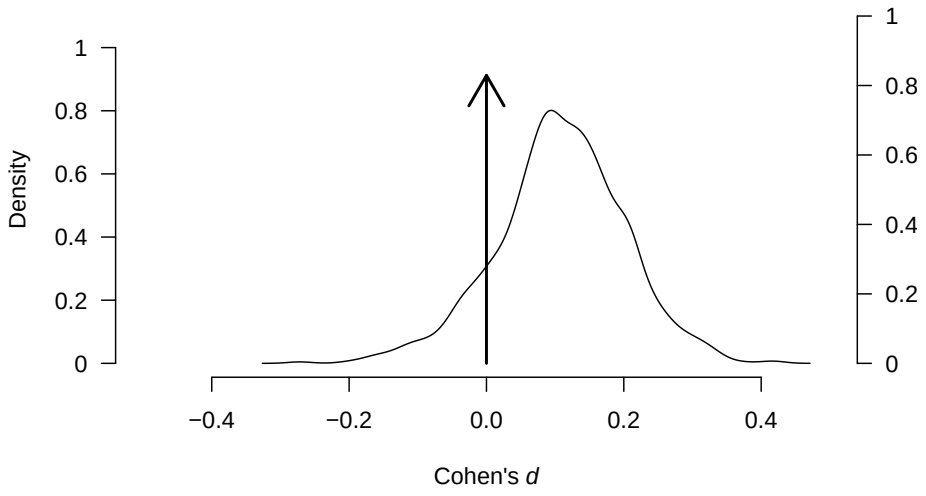
The implementation of RoBMA in R is straightforward using the `RoBMA` package. Below is a basic example using simulated effect sizes (`d`) and standard errors (`se`):

```r
# Load the necessary libraries
library(RoBMA)

# Ensure your dataset 'dat' is structured correctly
# Use the yi and vi columns from 'dat'
# Fit the Bayesian meta-analysis model
fit <- RoBMA(
  d = dat$yi,                       # Effect sizes
  se = sqrt(dat$vi),                # Standard errors from variances
  study_names = as.character(dat$paper),        # Unique study identifiers
  seed = 1,                          # Seed for reproducibility
  chains = 2,
  sample = 500,
  burnin = 200,parallel = TRUE)

# Summarize the results
summary(fit)
```

```
## Call:
## RoBMA(d = dat$yi, se = sqrt(dat$vi), study_names = as.character(dat$paper),
##     chains = 2, sample = 500, burnin = 200, parallel = TRUE,
##     seed = 1)
##
## Robust Bayesian meta-analysis
## Components summary:
##               Models Prior prob. Post. prob. Inclusion BF
## Effect         18/36       0.500       0.171        0.206
## Heterogeneity  18/36       0.500       1.000          Inf
## Bias           32/36       0.500       0.359        0.560
##
## Model-averaged estimates:
##                     Mean Median 0.025 0.975
## mu                 0.018  0.000 0.000 0.204
```

```
## tau                0.479  0.472 0.373 0.628
## omega[0,0.025]     1.000  1.000 1.000 1.000
## omega[0.025,0.05] 0.992  1.000 0.873 1.000
## omega[0.05,0.5]    0.969  1.000 0.572 1.000
## omega[0.5,0.95]    0.965  1.000 0.529 1.000
## omega[0.95,0.975] 0.967  1.000 0.541 1.000
## omega[0.975,1]     0.969  1.000 0.541 1.000
## PET                0.099  0.000 0.000 1.183
## PEESE              0.500  0.000 0.000 6.212
## The estimates are summarized on the Cohen's d scale (priors were specified on the Cohen's d sc
## (Estimated publication weights omega correspond to one-sided p-values.)
```

```r
# Plot the results for the parameter of interest
plot(fit, parameter = "mu", xlim = c(-0.5, 0.5))  # Adjust xlim as necessary
```



5.0.5.2 Key Outputs and Interpretation

1. **Model Components and Inclusion Probabilities**:

   RoBMA assesses three primary components: Effect, Heterogeneity, and Bias. The *Inclusion Probability* indicates how likely each component is part of the model:

   - **Effect**: Probability that a true non-zero effect exists.

- **Heterogeneity**: Probability that effect sizes vary significantly between studies.
- **Bias**: Probability that results are influenced by publication bias.

For example, if `Bias = 98%`, this suggests a very high likelihood that publication bias is present and impacting the results.

2. **Model-Averaged Estimates**:

   RoBMA reports *model-averaged* estimates for the average effect size (`mu`), between-study variance (`tau`), and other model parameters. This averaging reduces the impact of individual extreme results by incorporating information from all candidate models.

   - `mu`: The central tendency of the effect size (e.g., Cohen's d).
   - `tau`: The estimated heterogeneity, reflecting how much the effect sizes vary between studies.
   - **Bias Weights (`omega`)**: These weights correspond to the probability of study inclusion based on their p-values, directly indicating the severity of publication bias.

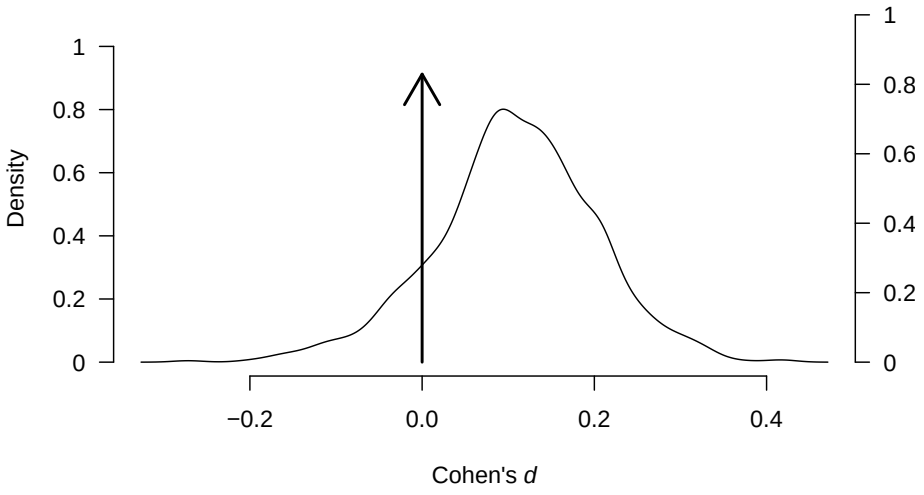
3. **Posterior Model Probabilities**:

   Each model in RoBMA has a posterior probability, which tells you how well that specific combination of effect, heterogeneity, and bias explains the observed data. This helps in identifying the most plausible scenarios.

5.0.5.3 Visualizing Publication Bias with RoBMA

RoBMA provides tools to visually inspect and diagnose publication bias using enhanced funnel plots. To generate a funnel plot that shows bias-corrected effects, you can use:

```
# Create an enhanced funnel plot with bias adjustment
plot(fit, type = "funnel", show_legend = TRUE)
```



The enhanced funnel plot compares the original effect sizes with those adjusted for potential biases. Asymmetry or clustering patterns in the plot indicate possible biases, while the adjusted estimates provide a more accurate picture of the underlying effect.

5.0.5.4 Advantages of Using RoBMA

- **Comprehensive Bias Assessment**: RoBMA integrates several models to check for the presence and impact of publication bias.
- **Flexible Prior Specification**: Allows for the use of both informative and non-informative priors, adapting to various research scenarios.
- **Model-Averaging for Stability**: By combining different models, RoBMA reduces the chance of overfitting and provides more robust conclusions.

5.0.5.5 Limitations to Consider

- **Computational Complexity**: Running RoBMA with large datasets or complex prior structures can be resource-intensive.
- **Advanced Interpretation**: Understanding and explaining RoBMA results may require familiarity with Bayesian concepts and model averaging.