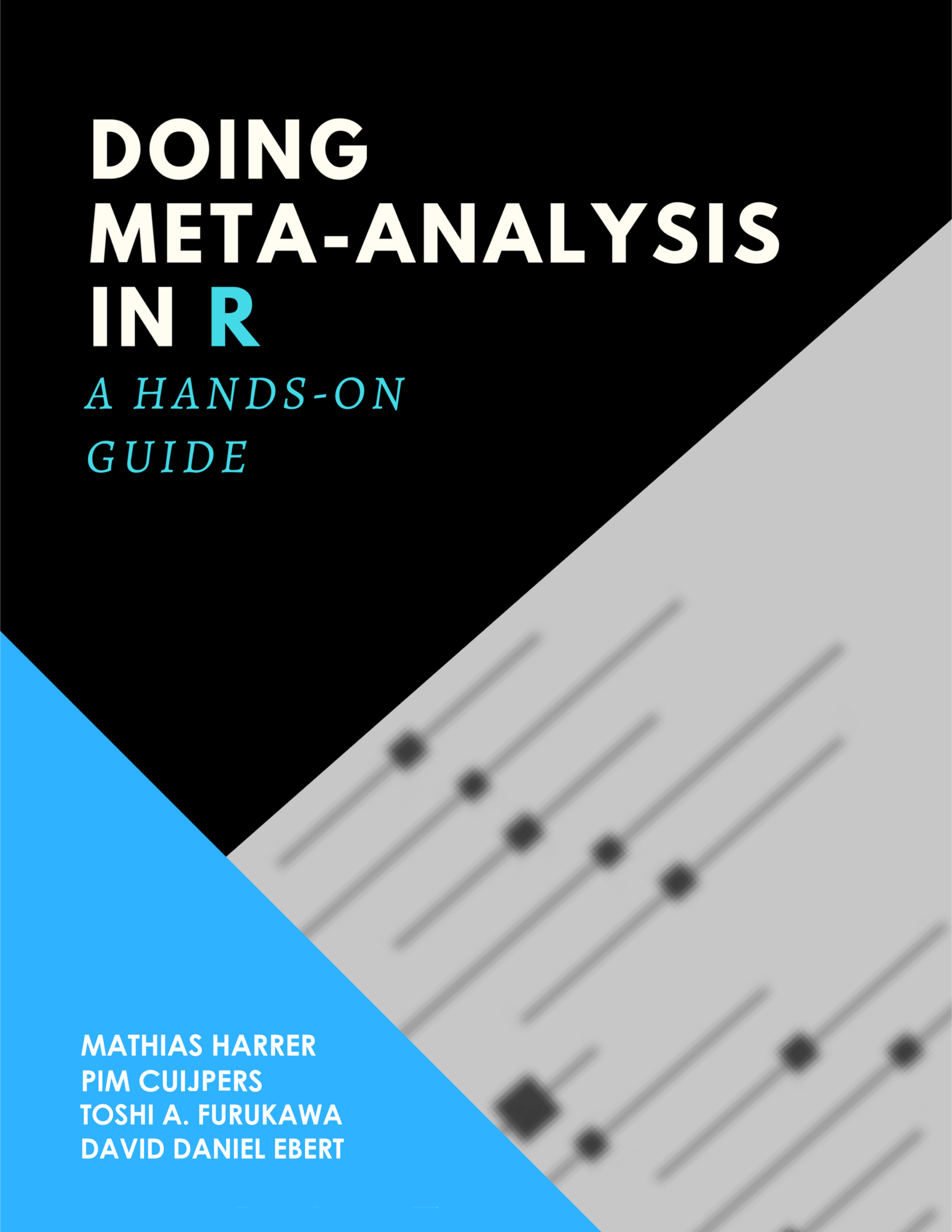
DOING META-ANALYSIS IN R

A HANDS-ON GUIDE



MATHIAS HARRER
PIM CUIJPERS
TOSHI A. FURUKAWA
DAVID DANIEL EBERT

Doing Meta-Analysis in R

A Hands-on Guide

Mathias Harrer, M.Sc.[1]
Prof. Dr. Pim Cuijpers[2]
Prof. Dr. Toshi A. Furukawa[3]
Assoc. Prof. Dr. David D. Ebert[2]

[1]*Friedrich-Alexander-University Erlangen-Nuremberg,* [2]*Vrije Universiteit Amsterdam,* [3]*Kyoto University*

Contents

Part I

Introduction & R Basics

Chapter 1

About this Guide

This guide shows you how to conduct Meta-Analyses in R from scratch. The focus of this guide is primarily on clinical outcome research in psychology. It was designed for staff and collaborators of the PROTECT Lab, which is headed by **Prof. Dr. David D. Ebert**.

The guide will show you how to:

- Get R and **RStudio** set for your Meta-Analysis.
- Get your data into R.

- **Prepare your data** for the meta-analysis.
- Introduce you to the `dmetar` *R* package we built specifically for this guide.
- Perform **fixed-effect** and **random-effects** meta-analysis using the `meta` and `metafor` packages.
- Analyse the **heterogeneity** of your results.
- Tackle heterogeneity using **subgroup analyses** and **meta-regression**.
- Check if **selective outcome reporting (publication bias)** or **p-hacking** is a present in your data
- Summarize the **risk of bias** in your data
- Calculate the **power** of a meta-analysis
- **Convert effect sizes** reported in original studies to the ones you need for your meta-analysis.
- Do advanced types of meta-analyses, such as
  - **multilevel meta-analyses**
  - **multivariate meta-analyses**
  - **meta-analytic structural equation modeling** or
  - **network meta-analyses**.

What this guide will not cover

Although this guide will provide some information on the statistics behind meta-analysis, it will not give you an **in-depth introduction** into how meta-analyses are calculated statistically.

It is also beyond the scope of this guide to advise in detail which meta-analytical strategy is suited best in which contexts, and on how the search, study inclusion and reporting of meta-analyses should be conducted. The *Cochrane Handbook for Systematic Reviews of Interventions*, however, should be a great source to find more information on these topics.

Other sources to recommend when conducting Meta-Analyses

- If you're looking for a easily digestable, hands-on introduction on how Meta-Analyses are conducted, we can recommend **Pim Cuijpers' online courses on Meta-Analysis**. The courses are freely available on YouTube. To have a look, click here.
- If you're interested in more details on how to conduct Meta-Analyses in R, you can either have a look at Wolfgang Viechtbauer's page for the `metafor` package (Link). Or you can consult a book written by Schwarzer and colleagues on the `meta` package (Schwarzer et al., 2015). + Pigott

How to get the R code for this guide

GitHub

All code behind this book is available online on **GitHub**. We have created a website containing a **download link** for all codes, and a **quick guide** on how to get the code running on your computer. The site can be found here.

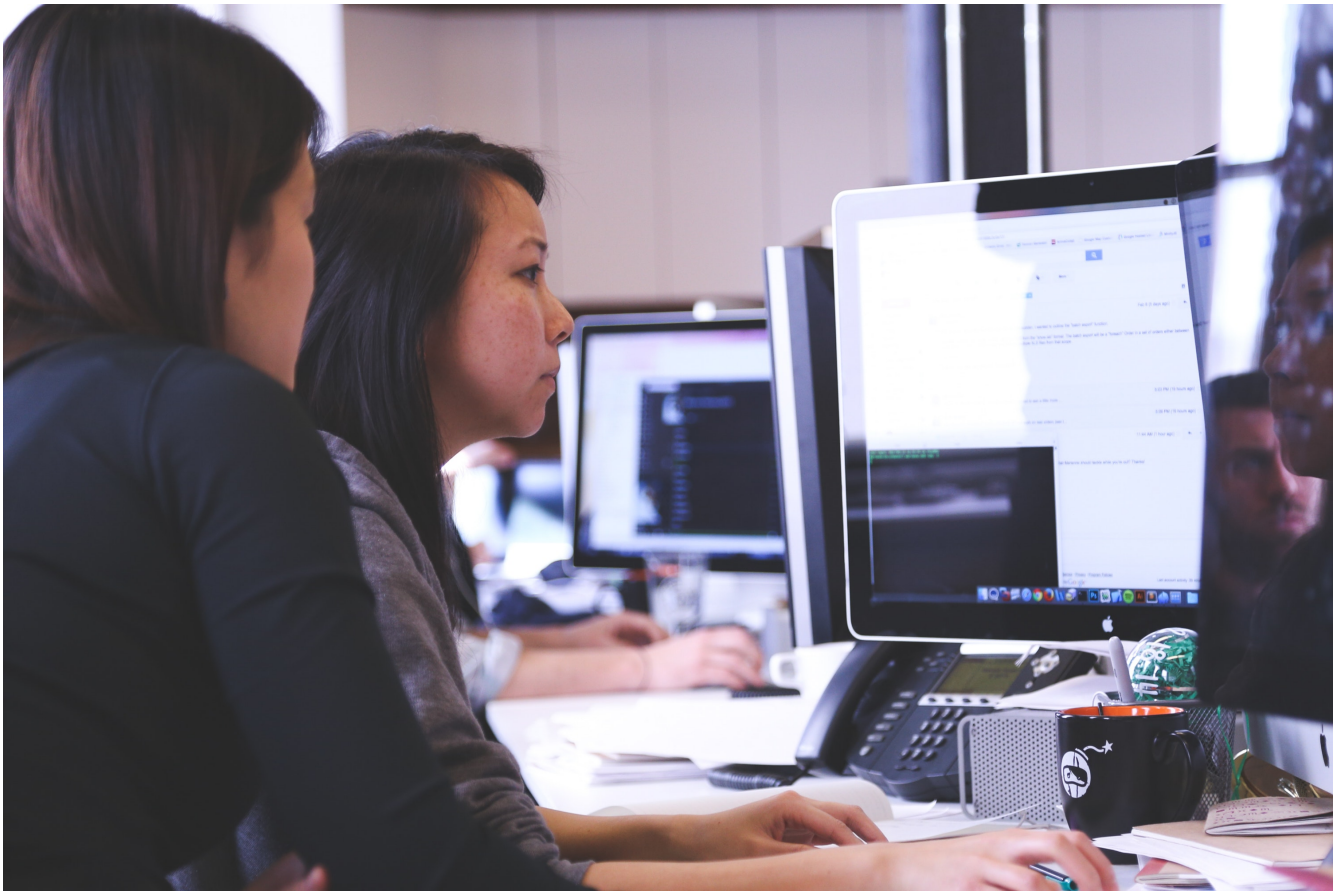
How to cite this guide

Harrer, M., Cuijpers, P., Furukawa, T.A, & Ebert, D. D. (2019). Doing Meta-Analysis in R: A Hand-on Guide. https://bookdown.org/MathiasHarrer/Doing_Meta_Analysis_in_R/.

License

Chapter 2

RStudio & Basics



Before we start with our meta-analysis, we have to download and prepare a **computer program** which allows us to use *R* for statistical analyses. Probably the best option for this at the moment is **RStudio**. This program gives us a user interface which makes it easier to handle our data, packages and output. The best part is that RStudio is **completely free** and can be downloaded anytime. In this Chapter, we'll focus on how you can install RStudio on your computer. We'll also provide some general information on *R*, and how you can get help if you get error messages. If you already have RStudio installed on your computer, and if you're an experienced *R* user already, all of this might be nothing new for you. You may **skip** this chapter then. Especially if you have **never used R before, we would like to consider this Chapter essential**, as it gives you some input on how R works, and how we can use it for our data analyses.

2.1 Getting RStudio to run on your computer

As a prerequisite for this guide, you need to have **RStudio** and a few essential **R packages** installed.

You have to follow these steps to get your RStudio version set.

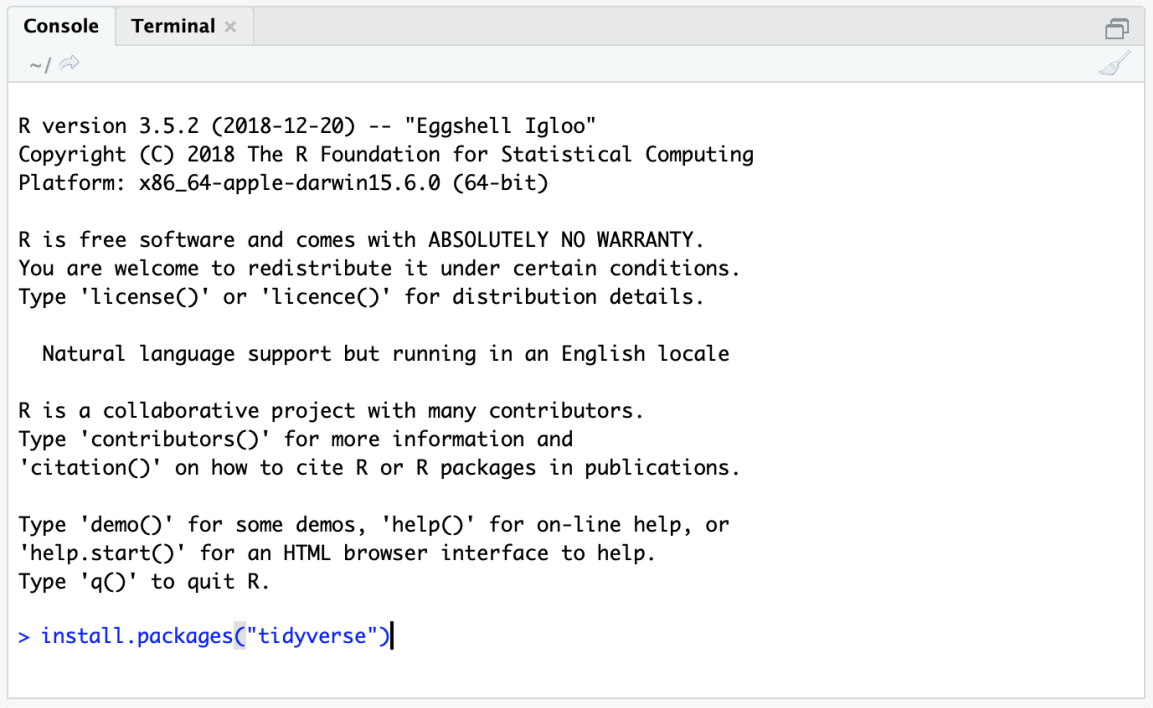
1. Download RStudio on the **RStudio** Website (Link). It's free!
2. If you do not have **R** installed yet, you will have to install the latest R Version before you can use RStudio. You can get the latest R version here. You have to choose the right *R* version depending on the operating system you use (i.e., Windows or Mac).
3. Once RStudio is running, open the **Console** on the bottom left corner of your screen.
4. We will now install a few packages using R Code. Here's an overview of the packages, and why we need them:

Package	Description
tidyverse	This is a large bundle of packages which make it easy to manipulate and visualize data in R. Functions included in the tidyverse have become very popular in the R community in recent years, and are used by many researchers, programmers and data scientists. If you want to learn more about the tidyverse, you can click on this link: https://www.tidyverse.org/.
meta	This package contains functions which make it easy to run different types of meta-analyses. We will primarily focus on this package in the guide, because it is easy to use, well documentated, and very versatile. More info on the meta package can be found here: http://www.imbi.uni-freiburg.de/lehre/lehrbuecher/meta-analysis-with-r.
metafor	The metafor package is also dedicted to conducting meta-analyses, and a true 'powerhouse' in terms of functionality. The package also provides more advanced tools, but may be a little involved for first time R users. However, because we will use this package at times in later chapters, and because metafor is used by the meta package for many applications, it is best to have it installed. The metafor package also has an excellent documentation for various meta-analysis-related topics, which can be found here: http://www.metafor-project.org/doku.php/metafor.

5. To install these packages, we use the `install.packages()` function in R. One package after another, our code should look like this:

```
install.packages("tidyverse")
install.packages("meta")
install.packages("metafor")
```

In RStudio, you simply have to type in the code displayed above into the the **Console** (usually, this is the bottom-left pane in your RStudio window) next to the little arrow (>) in the last line. Then hit **Enter** .



Don't forget to put the packages in "". Otherwise, you will get an error message. **You are now set and ready to proceed. Below, you can find some basic information on RStudio and troubleshooting**

2.1.1 Running R Code

Order to get the most out of this guide, it's helpful (but not essential) if you have some programming experience already. If you've never programmed before, you might find ***Hands-On Programming with R*** (Grolemund, 2014) to be a useful primer. There are three things you need to run the code: **R**, **RStudio**, and collection of **R packages**. Packages are the fundamental units of reproducible R code. They include reusable functions, the documentation that describes how to use them, and sample data. Gladly, once you've reached this point successfully, these three things are set already. Nevertheless, we will have to install and load a few new packages at some place in this guide, for which you can use the `install.packages()` the same way as you did before.

2.1.2 Getting Help

As you start to apply the functions described in this guide to your data you will soon find questions that the guide does not answer. This section describes a few tips on how to get help.

- If you get stuck somewhere, the best way is to start with **Google**. Google is particularly useful for error messages. If you get an error message and you have no idea what to do with it, try googling it first. The *R* community is very large, and chances are that someone else has been confused by it in the past, and there will be help somewhere on the web. (If the error message isn't in English, run `Sys.setenv(LANGUAGE = "en")` and re-run the code; you're more likely to find help for English error messages.)
- If Google doesn't help, try stackoverflow. Start by spending a little time searching for an existing answer; including [R] restricts your search to questions and answers that use R.

- Lastly, if you stumble upon an error (or typos!) in this guide's text or *R* syntax, feel free to contact **Mathias Harrer** at mathias.harrer@fau.de.

2.2 The *dmetar* package

In this guide, we want to make conducting meta-analyses as accessible and easy as possible for you as a researcher. Although there are fantastical packages like the `meta` and `metafor` package which do most of the heavy lifting, there are still some aspects of meta-analyses in the biomedical field and psychology which we consider important, but are **not easy** to do in *R* currently, particularly if you do not have a programming or statistics background. To fill this gap, we developed the `dmetar` package, which serves as the companion *R* package for this guide. The `dmetar` package has its own documentation, which can be found here. Functions of the `dmetar` package provide additional functionality for the `meta` and `metafor` packages (and a few other, more advanced packages), which we will be using frequently in this guide. Most of the functions included in the `dmetar` package and how they can improve your meta-analysis workflow will be described in detail throughout the guide.

Although highly recommended, it is **not essential** to have the `dmetar` package installed to work through the guide. For each function of the package used in the guide, we will also provide the source code, which can be used to save the function locally on your computer, and the additional *R* packages those functions rely on. However installing the `dmetar` package beforehand is **much more convenient**, because all the functions will already be pre-installed on your computer.

2.2.1 Installation of the `dmetar` package

2.2.1.1 Current R version

To install the `dmetar` package, the *R* version of your computer must be *3.5.2* or higher. If you have (re-)installed *R* recently, this will probably be the case. To check if your R version is new enough, you can paste this line of code into the Console, and then hit **Enter**.

```
R.Version()$version.string
```

This will display the current *R* version you have. If the *R* version is below *3.5.2*, you will have to update your *R* version. A tutorial on how to do this can be found here.

2.2.1.2 Devtools

If you want to install `dmetar`, one package already needs to be installed on your computer first. This package is called `devtools`. So, if `devtools` is not on your computer yet, you can install it like we did before:

```
install.packages("devtools")
```

2.2.1.3 Installation process

You can then install `dmetar` using this line of code:

```
devtools::install_github("MathiasHarrer/dmetar")
```

This will initiate the installation process. It is likely that the installation will take some time, because several other packages have to be installed along with the `dmetar` package for it to function properly.

During the installation process, the installation manager may ask you if you want to update existing *R* packages on your computer. The output may look something like this:

```
These packages have more recent versions available.
Which would you like to update?

1: All
2: CRAN packages only
3: None
4: ggpubr (0.2.2 -> 0.2.3) [CRAN]
5: zip    (2.0.3 -> 2.0.4) [CRAN]

Enter one or more numbers, or an empty line to skip updates:
```

When you get this message, it is best to first tell the installation manager that no packages should be updated. In this example, this means pasting 3 into the console and then hitting **Enter**. In the same vein, when the installation manager asks this question:

```
 There are binary versions available but the source versions are later:

 [...]

  Do you want to install from sources the package which needs compilation?
  y/n:
```

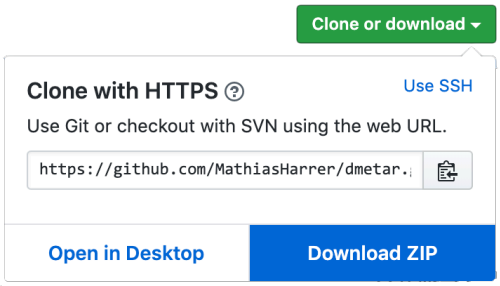
It is best to first choose n (no). If the installation fails with these strategies, run the installation again, updating all packages this time.

2.2.1.3.1 "Failed to install 'unknown package' from GitHub"

After having installed `devtools`, and then proceeding with the installation of the `dmetar` package, it is possible that the installation does not start and that you get the following error message:

```
Error: Failed to install 'unknown package' from GitHub:
  Timeout was reached: Connection timed out after 10000 milliseconds
```

This error message can have various reasons, depending on your internet connection, location, operating system, and/or *R* version (see thread for a similar issue). If you get this error message, you can try to download the `dmetar` package directly from GitHub. On the GitHub site, click on the button **Clone or download**, and then on **Download ZIP**.



After downloading, unzip the ZIP-folder, and drag the **dmetar-master** folder inside to a meaningful location on your computer. You can then install the `dmetar` package directly from your computer using the code shown below.

For Windows:

```
devtools::install("C:\\Path\\To\\The\\Folder\\dmetar-master")
```

For Mac:

```
devtools::install("~/Path/To/The/Folder/dmetar-master")
```

Where you have to replace the path shown in the above examples with the path to the **dmetar-master** folder on your computer.

After the `dmetar` package is installed successfully, you can load it from your library:

```
library(dmetar)
```

If you are having trouble installing the package, you can copy the **entire output of the installation process**, including the **code** you entered, and the **error message** you receive, and send it to **Mathias** (mathias.harrer@fau.de).

Chapter 3

Getting your data into R

This chapter will tell you about how you can **import** your effect size data in RStudio. We will also show you a few commands which make it easier to **manipulate data** directly in *R*. Data preparation can be tedious and exhausting at times, but it is the backbone of all later steps when doing meta-analyses in *R*. We therefore have to pay **close attention** to preparing the data correctly before we can proceed.

3.1 Data preparation in Excel

To conduct Meta-Analyses in *R*, you need to have your study data prepared. In the following, we describe the (preferred) way in which you should structure your dataset to facilitate the import into RStudio. We do this for two types of data: **"raw" effect size data** and **pre-calculated effect size data**. Usually, data imported into RStudio is stored in **Excel spreadsheets** first. We recommend to store your data there, because this makes it very easy to do the import.

What to (not) to do in your Excel spreadsheet

- It is very important how you **name the columns of your spreadsheet**. If you already named the columns of your sheet adequately in Excel, you can save a lot of time because your data doesn't have to be transformed in RStudio later on. "Naming" the columns of the spreadsheet simply means to write the name of the variable into the first line of the column; RStudio will automatically detect that this is the name of the column then.
- It **doesn't matter how columns are ordered in your Excel spreadsheet**. They just have to be labeled correctly.
- There's also no need to **format** the columns in any way. If you type the column name in the first line of you spreadsheet, R will automatically detect it as a column name.
- It's also important to know that the import **will distort letters like ä,ü,ö,á,é,ê, etc**. So be sure to transform them to "normal" letters before you proceed.
- Make sure that your Excel file **only contains one sheet**. If you want to import Excel files with more than one sheet, you can have a look at the `xlsx` package.
- If you have **one or several empty rows or columns which used to contain data**, make sure to delete those columns/rows completely, because RStudio could think that these columns contain data and import them also.

3.1.1 Setting the columns of the Excel spreadsheet (raw effect size data)

3.1.1.1 "Standard" effect size data (*M*, *SD*, *N*)

For a "standard" meta-analysis which uses the mean, standard deviation, and sample size from both groups in a study, the following information is needed for every study.

- The **names** of the individual studies, so that they can be easily identified later on. Usually, the first author and publication year of a study is used for this (e.g. "Ebert et al., 2018"). The names must be unique for each study.
- The **Mean** of both the Intervention and the Control group at the same assessment point.
- The **Standard Deviation** of both the Intervention and the Control group at the same assessment point.
- The **number of participants (N)** in each group of the trial.
- If you want to have a look at differences between various study subgroups later on, you also need a **subgroup code** for each study which signifies to which subgroup it belongs. For example, if a study was conducted in children, you might give it the subgroup code "children".

Here is how you should name the data columns in your EXCEL spreadheet containing your Meta-Analysis data

Column	Description
Author	This signifies the column for the study label (i.e., the first author)
Me	The Mean of the experimental/intervention group
Se	The Standard Deviation of the experimental/intervention group
Mc	The Mean of the control group
Sc	The Standard Deviation of the control group
Ne	The number of participants in the experimental/intervention group
Nc	The number of participats in the control group
Subgroup	This is the label for one of your Subgroup codes. It's not that important how you name this column, so you can give it a more informative name (e.g. population). In this column, each study should then be given an subgroup code, which should be exactly the same for each subgroup, including upper/lowercase letters. Of course, you can also include more than one subgroup column with different subgroup codings, but the column name has to be unique

3.1.1.2 Event rate data

For a meta-analysis of event rates, which uses the number of events and sample size from both groups in a study, the following information is needed for every study.

- The **names** of the individual studies, so that they can be easily identified later on. Usually, the first author and publication year of a study is used for this (e.g. "Ebert et al., 2018"). The names must be unique for each study.
- The **Number of events** of both the Intervention and the Control group at the same assessment point.
- The **number of participants (**N**)** in each group of the trial.
- If you want to have a look at differences between various study subgroups later on, you also need a **subgroup code** for each study which signifies to which subgroup it belongs. For example, if a study was conducted in children, you might give it the subgroup code "children".

Column	Description
Author	This signifies the column for the study label (i.e., the first author)
Ee	Number of events in the experimental treatment arm
Ne	Number of participants in the experimental treatment arm
Ec	Number of events in the control arm
Nc	Number of participants in the control arm
Subgroup	This is the label for one of your subgroup codes. It's not that important how you name it, so you can give it a more informative name (e.g. population). In this column, each study should then be given a subgroup code, which should be exactly the same for each subgroup, including upper/lowercase letters. Of course, you can also include more than one subgroup column with different subgroup codings, but the column name has to be unique

3.1.1.3 Incidence rate data

For a meta-analysis of incidence rates, which uses the number of events and person-time from both groups in a study, the following information is needed for every study.

- The **names** of the individual studies, so that they can be easily identified later on. Usually, the first author and publication year of a study is used for this (e.g. "Ebert et al., 2018"). The names must be unique for each study.
- The **Number of events** of both the Intervention and the Control group at the same assessment point.
- The **person-time at risk** in each group of the trial.
- If you want to have a look at differences between various study subgroups later on, you also need a **subgroup code** for each study which signifies to which subgroup it belongs. For example, if a study was conducted in children, you might give it the subgroup code "children".

Column	Description
Author	This signifies the column for the study label (i.e., the first author)
event.e	Number of events in the experimental treatment arm
time.e	The person-time at risk in the experimental treatment arm
event.c	Number of events in the control arm
time.c	The person-time at risk in the control arm
Subgroup	This is the label for one of your subgroup codes. It's not that important how you name it, so you can give it a more informative name (e.g. population). In this column, each study should then be given a subgroup code, which should be exactly the same for each subgroup, including upper/lowercase letters. Of course, you can also include more than one subgroup column with different subgroup codings, but the column name has to be unique

3.1.2 Setting the columns of the Excel spreadsheet (pre-calculated effect size data)

If you have **already calculated the effect sizes for each study on your own**, for example using *Comprehensive Meta-Analysis*, *RevMan*, or one of the **effect size calculators** we present in this guide, there's another way to prepare your data which makes things a little easier. In this case, you only have to include the following columns:

Column	Description
Author	This signifies the column for the study label (i.e., the first author)
TE	The calculated effect size of the study (either Cohen's d or Hedges' g, or some other form of effect size
seTE	The Standard Error (SE) of the calculated effect
Subgroup	This is the label for one of your Subgroup codes. It's not that important how you name it, so you can give it a more informative name (e.g. population). In this column, each study should then be given an subgroup code, which should be exactly the same for each subgroup, including upper/lowercase letters. Of course, you can also include more than one subgroup column with different subgroup codings, but the column name has to be unique

3.2 Importing the Spreadsheet into Rstudio

To get our data into R, we need to **save our data in a format and at a place where RStudio can open it**.

3.2.1 Saving the data in the right format

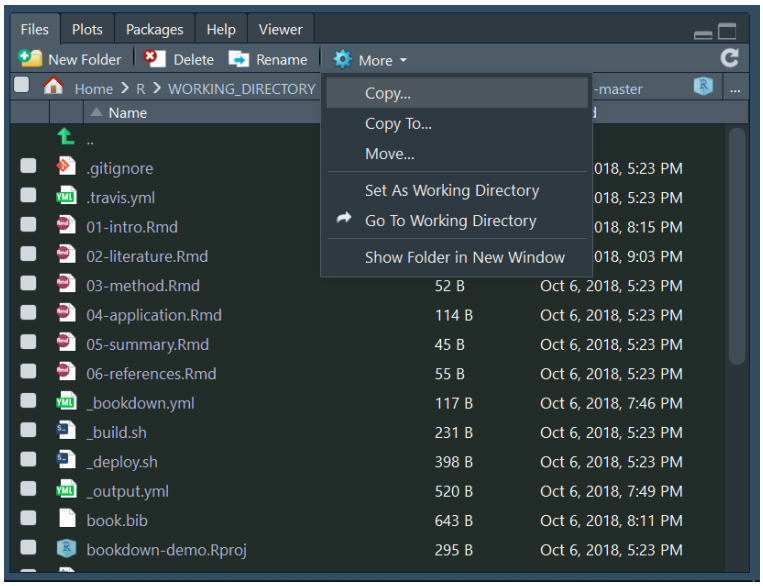
Generelly, finding the right format to import Excel can be tricky.

- If you're using a PC or Mac, it is advised to save your Excel sheet as a **comma-separated-values (.csv) file**. This can be done by clicking on "save as" and then choosing (.csv) as the output format in Excel.
- With some PCs, RStudios might not be able to open such files, or the files might be distorted. In this case, you can also try to save the sheet as a normal **.xslx** Excel-file and try if this works.

3.2.2 Saving the data in your working directory

To function properly, you have to set a working directory for RStudio first. The working directory is a **folder on your computer from which RStudio can use data, and in which output it saved.**

1. Therefore, create a folder on your computer and give it a meaningful name (e.g. "My Meta-Analysis").
2. Save your spreadsheet in the folder
3. Set the folder as your working directory. This can be done in RStudio on the **bottom-right corner of your screen**. Under the tile **"Files"**, search for the folder on your computer, and open it.
4. Once you've opened your folder, the file you just saved there should be in there.
5. Now that you've opened the folder, click on the **little gear wheel on top of the pane**

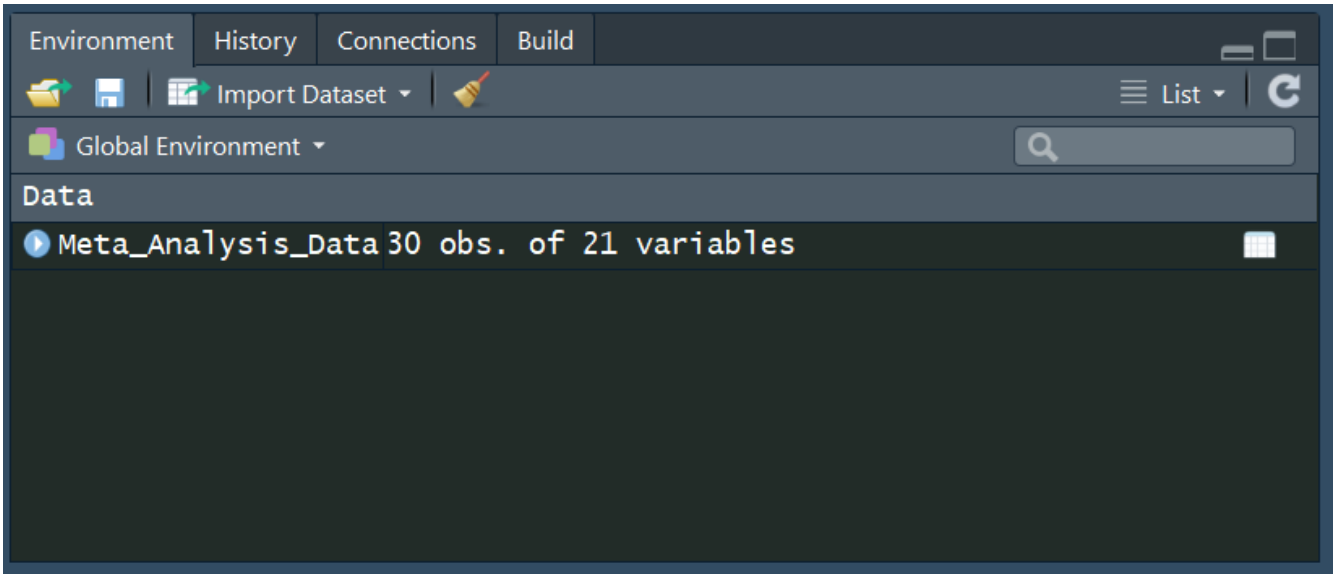


6. Then click on **"Set as working directory"**

Your file, and the working directory, are now where they should be!

3.2.3 Loading the data

1. To import the data, simply **click on the file** in the bottom-right pane. Then click on **import dataset...**
2. An **import assistant** should now pop up, which is also loading a preview of your data. This can be time-consuming sometimes, so you can skip this step if you want to, and klick straight on **"import"**



As you can see, the on the top-right pane **Environment**, your file is now listed as a data set in your RStudio environment.

3. I also want to give my data a shorter name ("madata"). To rename it, i use the following code:

```
madata <- Meta_Analysis_Data
```

This "copies" the data, and gives the copy the name `madata`.

4. Now, let's have a look at the **structure of my data** using the `str()` function

```
str(madata)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame': 18 obs. of 15 variables:
## $ Author : chr "Call et al." "Cavanagh et al." "DanitzOrsillo" "de Vibe et
al." ...
## $ TE : num 0.709 0.355 1.791 0.182 0.422 ...
## $ seTE : num 0.261 0.196 0.346 0.118 0.145 ...
## $ RoB : chr "low" "low" "high" "low" ...
## $ Control : chr "WLC" "WLC" "WLC" "no intervention" ...
## $ intervention duration: chr "short" "short" "short" "short" ...
## $ intervention type : chr "mindfulness" "mindfulness" "ACT" "mindfulness"
...
## $ population : chr "undergraduate students" "students" "undergraduate
students" "undergraduate students" ...
## $ type of students : chr "psychology" "general" "general" "general" ...
## $ prevention type : chr "selective" "universal" "universal" "universal" ...
## $ gender : chr "female" "mixed" "mixed" "mixed" ...
## $ mode of delivery : chr "group" "online" "group" "group" ...
## $ compensation : chr "none" "none" "voucher/money" "voucher/money" ...
## $ instruments : chr "DASS" "PSS" "DASS" "other" ...
## $ guidance : chr "f2f" "self-guided" "f2f" "f2f" ...
```

Although this output looks kind of messy, it's already very informative. It shows the structure of my data. In this case,

i used data for which the effect sizes were already calculated. This is why the variables **TE** and **seTE** appear. I also see plenty of other variables, which correspond to the subgroups which were coded for this dataset.

Here's a (shortened) table created for my data

Author	TE	seTE	RoB	Control	intervention duration	intervention type
Call et al.	0.7091362	0.2608202	low	WLC	short	mindfulness
Cavanagh et al.	0.3548641	0.1963624	low	WLC	short	mindfulness
DanitzOrsillo	1.7911700	0.3455692	high	WLC	short	ACT
de Vibe et al.	0.1824552	0.1177874	low	no intervention	short	mindfulness
Frazier et al.	0.4218509	0.1448128	low	information only	short	PCI
Frogeli et al.	0.6300000	0.1960000	low	no intervention	short	ACT
Gallego et al.	0.7248838	0.2246641	high	no intervention	long	mindfulness
Hazlett-Stevens & Oren	0.5286638	0.2104609	low	no intervention	long	mindfulness
Hintz et al.	0.2840000	0.1680000	low	information only	short	PCI
Kang et al.	1.2750682	0.3371997	low	no intervention	long	mindfulness
Kuhlmann et al.	0.1036082	0.1947275	low	no intervention	short	mindfulness
Lever Taylor et al.	0.3883906	0.2307689	low	WLC	long	mindfulness
Phang et al.	0.5407398	0.2443133	low	no intervention	short	mindfulness
Rasanen et al.	0.4261593	0.2579379	low	WLC	short	ACT
Ratanasiripong	0.5153969	0.3512737	high	no intervention	short	mindfulness
Shapiro et al.	1.4797260	0.3152817	low	WLC	long	mindfulness
SongLindquist	0.6125782	0.2266834	high	WLC	long	mindfulness
Warnecke et al.	0.6000000	0.2490000	low	information only	long	mindfulness

3.3 Data Manipulation

Now that we have the Meta-Analysis data in RStudio, let's do a **few manipulations with the data**. These functions might come in handy when we are conducting analyses later on. Going back to the output of the `str()` function, we see that this also gives us details on the type of column data we have stored in our data. There are different abbreviations signifying different types of data.

Abbreviation	Type	Description
num	Numerical	This is all data stored as numbers (e.g. 1.02).
chr	Character	This is all data stored as words.
log	Logical	These are variables which are binary, meaning that they signify that a condition is either TRUE or FALSE.
factor	Factor	Factors are stored as numbers, with each number signifying a different level of a variable. A possible factor of a variable might be 1 = low, 2 = medium, 3 = high.

3.3.1 Converting to factors

Let's say we have the subgroup **Risk of Bias** (in which the Risk of Bias rating is coded), and want it to be a factor with two different levels: "low" and "high".

To do this, we need the variable ROB to be a factor. However, this variable is currently stored as a character (chr). We can have a look at this variable by typing the name of our dataset, then adding the selector $ and then adding the variable we want to have a look at.

```
madata$ROB
```

```
##  [1] "high" "low"  "high" "low"  "low"  "low"  "high" "low"  "low"  "low"
## [11] "high" "low"  "low"  "low"  "high" "high" "high" "low"
```

```
str(madata$ROB)
```

```
##  chr [1:18] "high" "low" "high" "low" "low" "low" "high" "low" "low" ...
```

We can see now that ROBis indeed a **character** type variable, which contains only two words: "low" and "high". We want to convert this to a **factor** variable now, which has only two levels, low and high. To do this, we use the `factor()` function.

```
madata$ROB<-factor(madata$ROB)
madata$ROB
```

```
##  [1] high low  high low  low  low  high low  low  low  high low  low  low
## [15] high high high low
## Levels: high low
```

```
str(madata$ROB)
```

```
##  Factor w/ 2 levels "high","low": 1 2 1 2 2 2 1 2 2 2 ...
```

We now see that the variable has been **converted to a factor with the levels "high" and "low"**.

3.3.2 Converting to logicals

Now lets have a look at the **intervention type** subgroup variable. This variable is currently stores as a character chr too.

```
madata$`intervention type`
```

```
##  [1] "mindfulness" "mindfulness" "ACT"         "mindfulness" "PCI"
##  [6] "ACT"         "mindfulness" "mindfulness" "PCI"         "mindfulness"
## [11] "mindfulness" "mindfulness" "mindfulness" "ACT"         "mindfulness"
## [16] "mindfulness" "mindfulness" "mindfulness"
```

```
str(madata$`intervention type`)
```

```
##  chr [1:18] "mindfulness" "mindfulness" "ACT" "mindfulness" "PCI" ...
```

Let's say we want a variable which only contains information if a study way a mindfulness intervention or not. A logical is very well suited for this. To convert the data to logical, we use the `as.logical` function. We will create a new variable containing this information called `intervention.type.logical`. To tell R what to count as TRUE and what as FALSE, we have to define the specific intervention type using the == command.

```
intervention.type.logical<-as.logical(madata$`intervention type`=="mindfulness")
intervention.type.logical
```

```
##  [1]  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE  TRUE  TRUE
## [12]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
```

We see that R has converted the character information into trues and falses for us. To check if this was done correctly, let's compare the original and the new variable

New	Original
TRUE	mindfulness
TRUE	mindfulness
FALSE	ACT
TRUE	mindfulness
FALSE	PCI
FALSE	ACT
TRUE	mindfulness
TRUE	mindfulness
FALSE	PCI
TRUE	mindfulness
TRUE	mindfulness
TRUE	mindfulness
TRUE	mindfulness
FALSE	ACT
TRUE	mindfulness
TRUE	mindfulness
TRUE	mindfulness
TRUE	mindfulness

3.3.3 Selecting specific studies

It may often come in handy to **select certain studies for further analyses**, or to **exclude some studies in further analyses** (e.g., if they are outliers).

To do this, we can use the `filter` function in the `dplyr` package, which is part of the `tidyverse` package we installed before.

So, let's load the package first.

```
library(dplyr)
```

Let's say we want to do a Meta-Analysis with three studies in our dataset only. To do this, we need to create a new dataset containing only these studies using the `dplyr::filter()` function. The `dplyr::` part is necessary as there is more than one 'filter function in R, and we want to use to use the one of the dplyrpackage. Let's say we want to have the studies by **Cavanagh et al.**, **Frazier et al.** and **Phang et al.** stored in another dataset, so we can conduct analyses only for these studies. The R code to store these three studies in a new dataset called `madata.new` looks like this:

```
madata.new <- dplyr::filter(madata, Author %in% c("Cavanagh et al.",
                                                  "Frazier et al.",
                                                  "Phang et al."))
```

Note that the `%in%` command tells the `filter` function to search for exactly the three cases we defined in the variable `Author`. Now, let's have a look at the new data `madata.new` we just created.

Author	TE	seTE	RoB	Control	intervention duration
Cavanagh et al.	0.3548641	0.1963624	low	WLC	short
Frazier et al.	0.4218509	0.1448128	low	information only	short
Phang et al.	0.5407398	0.2443133	low	no intervention	short

Note that the function can also be used for any other type of data and variable. We can also use it to, for example, only select studies which were coded as being a mindfulness study.

```
madata.new.mf <- dplyr::filter(madata,`intervention type` %in% c("mindfulness"))
```

We can also use the `dplyr::filter()` function to exclude studies from our dataset. To do this, we only have to add `!` in front of the variable we want to use for filtering.

```
madata.new.excl <- dplyr::filter(madata,!Author %in% c("Cavanagh et al.",
                                                       "Frazier et al.",
                                                       "Phang et al."))
```

3.3.4 Changing cell values

Sometimes, even when preparing your data in Excel, you might want to **change values in RStudio once you have imported your data**.

To do this, we have to select a cell in our data frame in RStudio. This can be done by adding `[x,y]` to our dataset name, where **x** signifies the number of the **row** we want to select, and **y** signifies the number of the **column**.

To see how this works, let's select a variable using this command first:

```
madata[6,1]
```

```
## # A tibble: 1 x 1
##   Author
##   <chr>
## 1 Frogeli et al.
```

We now see the **6th study** in our dataframe, and the value of this study for **Column 1 (the author name)** is displayed. Let's say we had a typo in this name and want to have it changed. In this case, we have to give this exact cell a new value.

```
madata[6,1] <- "Frogelli et al."
```

Let's check if the name has changed.

```
madata[6,1]
```

```
## # A tibble: 1 x 1
##   Author
##   <chr>
## 1 Frogelli et al.
```

You can also use this function to change any other type of data, including numericals and logicals. Only for characters, you have to put the values you want to insert in "".

Part II

Meta-Analysis in R

Chapter 4

Pooling Effect Sizes

Now, let's get to the core of every Meta-Analysis: **pooling your effect sizes** to get one overall effect size estimate of the studies. When pooling effect sizes in Meta-Analysis, there are two approaches which we can use: the **Fixed-Effect-Model**, or the **Random-Effects-Model** (Borenstein et al., 2011). There is an extensive debate on which model fits best in which context (Fleiss, 1993), with no clear consensus in sight. Although it has been recommended to **only resort to the Random-Effects-Pooling model** in clinical psychology and the health sciences (Cuijpers, 2016), we will describe how to conduct both in *R* here. Both of these models only require an **effect size**, and a **dispersion (variance)** estimate for each study, of which the inverse is taken. This is why the methods are often called **generic inverse-variance methods**.

We will describe in-depth how to conduct meta-analyses in R with **effect sizes based on continuous outcome data** (such as standardized mean differences) first, as these are the most common ones in psychology and the health science field. Later on, we will extend this knowledge to meta-analyses with **binary outcome** data, which might be important if you're focusing on prevention trials. For these meta-analyses, we'll use the `meta` package (Schwarzer, 2007). In Section 2.1, we showed how to install the package. Now, load the package from your library to proceed.

```
library(meta)
```

4.1 Fixed-Effects-Model

4.1.1 Pre-calculated effect size data

4.1.1.1 The idea behind the fixed-effects-model

The fixed-effects-model assumes that all studies along with their effect sizes stem from a single homogeneous population (Borenstein et al., 2011). To calculate the overall effect, we therefore average all effect sizes, but give studies with greater precision a higher weight. In this case, greater precision means that the study has a larger N, which leads to a smaller **Standard Error** of its effect size estimate. For this weighting, we use the **inverse of the variance** $1/\hat{\sigma}_k^2$ of each study k. We then calculate a weighted average of all studies, our fixed effect size estimate $\hat{\theta}_F$:

$$\hat{\theta}_F = \frac{\sum_{k=1}^{K} \hat{\theta}_k/\hat{\sigma}_k^2}{\sum_{k=1}^{K} 1/\hat{\sigma}_k^2} \tag{4.1}$$

In Chapter 3.1, we have described two ways your Excel spreadsheet for your meta-analysis data can look like:

- It can either be stored as the **raw data** (including the Mean, N, and SD of every study arm)
- Or it only contains the **calculated effect sizes and the standard error (SE)**

The functions to pool the results with a fixed-effect-model **differ depending on which data format you used**, but not much. First, let's assume you already have a dataset with the **calucated effect sizes and SE** for each study. In my case, this is my `madata` dataset.

```
str(madata)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame': 18 obs. of 17 variables:
## $ Author : chr "Call et al." "Cavanagh et al." "DanitzOrsillo" "de Vibe et
al." ...
## $ TE : num 0.709 0.355 1.791 0.182 0.422 ...
## $ seTE : num 0.261 0.196 0.346 0.118 0.145 ...
## $ RoB : chr "low" "low" "high" "low" ...
## $ Control : chr "WLC" "WLC" "WLC" "no intervention" ...
## $ intervention duration: chr "short" "short" "short" "short" ...
## $ intervention type : chr "mindfulness" "mindfulness" "ACT" "mindfulness"
...
## $ population : chr "undergraduate students" "students" "undergraduate
```

```
students" "undergraduate students" ...
## $ type of students : chr "psychology" "general" "general" "general" ...
## $ prevention type : chr "selective" "universal" "universal" "universal" ...
## $ gender : chr "female" "mixed" "mixed" "mixed" ...
## $ mode of delivery : chr "group" "online" "group" "group" ...
## $ ROB streng : chr "high" "low" "high" "low" ...
## $ ROB superstreng : chr "high" "high" "high" "low" ...
## $ compensation : chr "none" "none" "voucher/money" "voucher/money" ...
## $ instruments : chr "DASS" "PSS" "DASS" "other" ...
## $ guidance : chr "f2f" "self-guided" "f2f" "f2f" ...
```

The effect sizes in this dataset are based on **continuous outcome data**. As our effect sizes are already calculated, we can use the `meta::metagen` function. For this function, we can specify loads of parameters, all of which you can accessed by typing `?metagen` in your console once the `meta` package is loaded.

Here is a table with the most important parameters for our code:

Parameter	Function
TE	This tells R to use the TE column to retrieve the effect sizes for each study
seTE	This tells R to use the seTE column to retrieve the standard error for each study
data=	After =, paste the name of your dataset here
studlab=paste()	This tells the function were the labels for each study are stored. If you named the spreadsheet columns as advised, this should be studlab=paste(Author)
comb.fixed=	Weather to use a fixed-effect-model
comb.random	Weather to use a random-effects-model
prediction=	Weather to print a prediction interval for the effect of future studies based on present evidence
sm=	The summary measure we want to calculate. We can either calculate the mean difference (MD) or Hedges' g/Cohen's d (SMD)

Let's code our first fixed-effects-model Meta-Analysis. We we will give the results of this analysis the simple name `m`.

```
m <- metagen(TE,
             seTE,
             data=madata,
             studlab=paste(Author),
             comb.fixed = TRUE,
             comb.random = FALSE,
             prediction=TRUE,
             sm="SMD")
m
```

```
##                          SMD            95%-CI %W(fixed)
## Call et al.           0.7091 [ 0.1979; 1.2203]       3.6
## Cavanagh et al.       0.3549 [-0.0300; 0.7397]       6.3
## DanitzOrsillo         1.7912 [ 1.1139; 2.4685]       2.0
## de Vibe et al.        0.1825 [-0.0484; 0.4133]      17.5
## Frazier et al.        0.4219 [ 0.1380; 0.7057]      11.6
## Frogeli et al.        0.6300 [ 0.2458; 1.0142]       6.3
## Gallego et al.        0.7249 [ 0.2846; 1.1652]       4.8
## Hazlett-Stevens & Oren 0.5287 [ 0.1162; 0.9412]       5.5
```

```
## Hintz et al.           0.2840 [-0.0453; 0.6133]       8.6
## Kang et al.            1.2751 [ 0.6142; 1.9360]       2.1
## Kuhlmann et al.        0.1036 [-0.2781; 0.4853]       6.4
## Lever Taylor et al.    0.3884 [-0.0639; 0.8407]       4.6
## Phang et al.           0.5407 [ 0.0619; 1.0196]       4.1
## Rasanen et al.         0.4262 [-0.0794; 0.9317]       3.6
## Ratanasiripong         0.5154 [-0.1731; 1.2039]       2.0
## Shapiro et al.         1.4797 [ 0.8618; 2.0977]       2.4
## SongLindquist          0.6126 [ 0.1683; 1.0569]       4.7
## Warnecke et al.        0.6000 [ 0.1120; 1.0880]       3.9
##
## Number of studies combined: k = 18
##
##                         SMD            95%-CI    z  p-value
## Fixed effect model  0.4805 [ 0.3840; 0.5771] 9.75 < 0.0001
## Prediction interval        [-0.0344; 1.1826]
##
## Quantifying heterogeneity:
## tau^2 = 0.0752; H = 1.64 [1.27; 2.11]; I^2 = 62.6% [37.9%; 77.5%]
##
## Test of heterogeneity:
##      Q d.f. p-value
##  45.50   17  0.0002
##
## Details on meta-analytical method:
## - Inverse variance method
```

We now see the results of our Meta-Analysis, including

- The **individual effect sizes** for each study, and their weight
- The total **number of included studies** (k)
- The **overall effect** (in our case, g = 0.4805) and its confidence interval and p-value
- Measures of **between-study heterogeneity**, such as tau^2 or I^2 and a Q-test of heterogeneity

Using the $ command, we can also have a look at various outputs directly. For example

```
m$lower.I2
```

Gives us the lower bound of the 95% confidence interval for I^2

```
## [1] 0.3787897
```

We can **save the results of the meta-analysis** to our working directory as a '.txt-file using this command

```
sink("results.txt")
print(m)
sink()
```

4.1.2 Raw Effect Size Data

To conduct a fixed-effects-model Meta-Analysis from **raw data** (i.e, if your data has been prepared the way we describe in Chapter 3.1.1), we have to use the `meta::metacont()` function instead. The structure of the code however, looks quite similar.

Parameter	Function
Ne	The number of participants (N) in the intervention group
Me	The Mean (M) of the intervention group
Se	The Standard Deviation (SD) of the intervention group
Nc	The number of participants (N) in the control group
Mc	The Mean (M) of the control group
Sc	The Standard Deviation (SD) of the control group
data=	After '=', paste the name of your dataset here
studlab=paste()	This tells the function were the labels for each study are stored. If you named the spreadsheet columns as advised, this should be studlab=paste(Author)
comb.fixed=	Weather to use a fixed-effects-model
comb.random	Weather to use a random-effects-model
prediction=	Weather to print a prediction interval for the effect of future studies based on present evidence
sm=	The summary measure we want to calculate. We can either calculate the mean difference (MD) or Hedges' g (SMD)

For this purpose, i will use my dataset `metacont`, which contains the raw data of all studies i want to snythesize.

```
str(metacont)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    6 obs. of  7 variables:
##  $ Author: chr  "Cavanagh" "Day" "Frazier" "Gaffney" ...
##  $ Ne    : num  50 64 90 30 77 60
##  $ Me    : num  4.5 18.3 12.5 2.34 15.21 ...
##  $ Se    : num  2.7 6.4 3.2 0.87 5.35 ...
##  $ Nc    : num  50 65 95 30 69 60
##  $ Mc    : num  5.6 20.2 15.5 3.13 20.13 ...
##  $ Sc    : num  2.6 7.6 4.4 1.23 7.43 ...
```

Now, let's code the Meta-Analysis function, this time using the `meta::metacont` function, and my `metacont` dataset. I want to name my output `m.raw` now.

```
m.raw <- metacont(Ne,
                  Me,
                  Se,
                  Nc,
                  Mc,
                  Sc,
                  data=metacont,
                  studlab=paste(Author),
                  comb.fixed = TRUE,
```

```
                  comb.random = FALSE,
                  prediction=TRUE,
                  sm="SMD")
m.raw
```

```
##              SMD             95%-CI %W(fixed)
## Cavanagh -0.4118 [-0.8081; -0.0155]      13.8
## Day      -0.2687 [-0.6154;  0.0781]      18.0
## Frazier  -0.7734 [-1.0725; -0.4743]      24.2
## Gaffney  -0.7303 [-1.2542; -0.2065]       7.9
## Greer    -0.7624 [-1.0992; -0.4256]      19.1
## Harrer   -0.1669 [-0.5254;  0.1916]      16.9
##
## Number of studies combined: k = 6
##
##                         SMD             95%-CI     z  p-value
## Fixed effect model  -0.5245 [-0.6718; -0.3773] -6.98 < 0.0001
## Prediction interval         [-1.1817;  0.1494]
##
## Quantifying heterogeneity:
## tau^2 = 0.0441; H = 1.51 [1.00; 2.38]; I^2 = 56.1% [0.0%; 82.3%]
##
## Test of heterogeneity:
##      Q d.f. p-value
##  11.39    5  0.0441
##
## Details on meta-analytical method:
## - Inverse variance method
## - Hedges' g (bias corrected standardised mean difference)
```

As you can see, all the calculated effect sizes are **negative** now, including the pooled effect. However, all studies report a positive outcome, meaning that the symptoms in the intervention group (e.g., of depression) were reduced. The negative orientation results from the fact that in **most clinical trials, lower scores indicate better outcomes** (e.g., less depression). It is no problem to report values like this: in fact, it is conventional. Some readers who are unfamiliar with meta-analysis, however, **might be confused** by this, so may consider changing the orientation of your values before you report them in your paper. We can **save the results of the meta-analysis** to our working directory as a .txt-file using this command.

```
sink("results.txt")
print(m.raw)
sink()
```

4.2 Random-Effects-Model

Previously, we showed how to perform a fixed-effect-model meta-analysis using the `metagen` and `metacont` functions. However, we can only use the fixed-effect-model when we can assume that **all included studies come from**

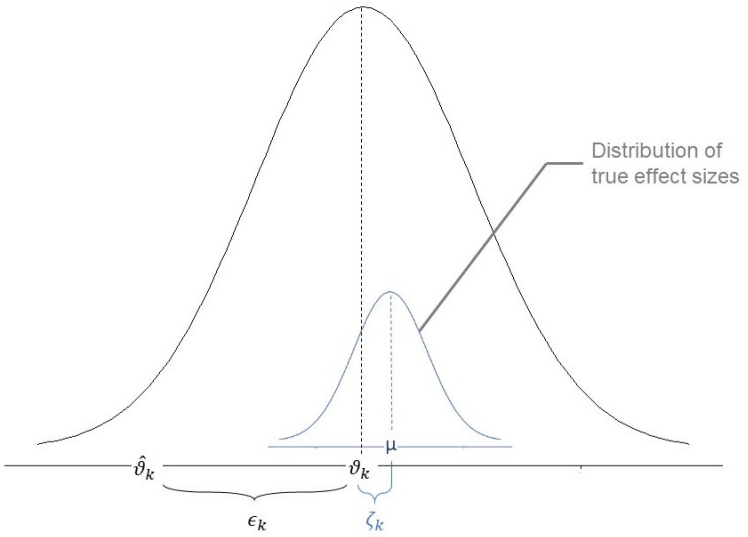


Figure 4.1: An Illustration of Parameters of the Random-Effects Model

the same population. In practice this is hardly ever the case: interventions may vary in certain characteristics, the sample used in each study might be slightly different, or its methods. In this case, we cannot assume that all studies stem from the same hypothesized "population" of studies. Same is the case once we detect **statistical heterogeneity** in our fixed-effect-model meta-analysis, as indicated by $I^2 > 0\%$. So, it is very likely that you will actually use a random-effects-model for your meta-analysis. Thankfully, there's not much more we have to think about when conducting a random-effects-model meta-analysis in *R* instead of a fixed-effect-model meta-analysis.

4.2.1 The Idea behind the Random-Effects-Model

In the Random-Effects-Model, we want to account for our assumption that the study effect estimates show more variance than when drawn from a single population (Schwarzer et al., 2015). The random-effects-model works under the so-called **assumption of exchangeability**. This means that in Random-Effects-Model Meta-Analyses, we not only assume that effects of individual studies deviate from the true intervention effect of all studies due to sampling error, but that there is another source of variance introduced by the fact that the studies do not stem from one single population, but are drawn from a "universe" of populations. We therefore assume that there is not only one true effect size, but **a distribution of true effect sizes**. We therefore want to estimate the mean of this distribution of true effect sizes. The fixed-effect-model assumes that when the observed effect size $\hat{\theta}_k$ of an individual study k deviates from the true effect size θ_F, the only reason for this is that the estimate is burdened by (sampling) error ϵ_k.

$$\hat{\theta}_k = \theta_F + \epsilon_k$$

While the random-effects-model assumes that, in addition, there is **a second source of error** ζ_k.This second source of error is introduced by the fact that even the true effect size θ_k of our study k is also only part of an over-arching distribution of true effect sizes with the mean μ (Borenstein et al., 2011).

The formula for the random-effects-model therefore looks like this:

$$\hat{\theta}_k = \mu + \epsilon_k + \zeta_k$$

When calculating a random-effects-model meta-analysis, where therefore also have to take the error ζ_k into account. To do this, we have to **estimate the variance of the distribution of true effect sizes**, which is denoted by τ^2, or *tau²*. There are several estimators for τ^2, many of which are implemented in `meta`. We will give you more details about them in the next section.

Even though it is **conventional** to use random-effects-model meta-analyses in psychological outcome research, applying this model is **not undisputed**. The random-effects-model pays **more attention to small studies** when pooling the overall effect in a meta-analysis (Schwarzer et al., 2015). Yet, small studies in particular are often fraught with **bias** (see Chapter 8.1). This is why some have argued that the fixed-effects-model should be nearly always preferred (Poole and Greenland, 1999; Furukawa et al., 2003).

4.2.2 Estimators for τ^2 in the random-effects-model

Operationally, conducting a random-effects-model meta-analysis in *R* is not so different from conducting a fixed-effects-model meta-analysis. Yet, we do have choose an estimator for τ^2. Here are the estimators implemented in `meta`, which we can choose using the `method.tau` variable in our meta-analysis code.

Code	Estimator
DL	DerSimonian-Laird
PM	Paule-Mandel
REML	Restricted Maximum-Likelihood
ML	Maximum-likelihood
HS	Hunter-Schmidt
SJ	Sidik-Jonkman
HE	Hedges
EB	Empirical Bayes

4.2.2.1 Which estimator should i use?

All of these estimators derive τ^2 using a slightly different approach, leading to somewhat different pooled effect size estimates and confidence intervals. If one of these approaches is more or less biased often depends on the context, and parameters such as the number of studies k, the number of participants n in each study, how much n varies from study to study, and how big τ^2 is. An overview paper by Veroniki and colleagues (Veroniki et al., 2016) provides an excellent summary on current evidence on which estimator might be more or less biased in which situation. The article is openly accessible, and you can read it here. Especially in medical and psychological research, the by far most often used estimator is the **DerSimonian-Laird estimator** (DerSimonian and Laird, 1986). Part of this widespread use might be attributable to the fact that programs such as *RevMan* or *Comprehensive Meta-Analysis* (older versions) only use this estimator. It is also the default option in the `meta` package in *R*. Simulation studies, however, have shown that the **Maximum-Likelihood**, **Sidik-Jonkman**, and **Empirical Bayes** estimators have better properties in estimating the between-study variance (Sidik and Jonkman, 2007; Viechtbauer, 2005).

4.2.2.2 The Hartung-Knapp-Sidik-Jonkman method

Another criticism of the **DerSimonian-Laird** method is that when estimating the variance of our pooled effect $var(\hat{\theta}_F)$, this method is prone to producing false positives (IntHout et al., 2014). This is especially the case when

the **number of studies** is small, and when there is substantial **heterogeneity** (Hartung, 1999; Hartung and Knapp, 2001b,a; Follmann and Proschan, 1999; Makambi, 2004). Unfortunately, this is very often the case in when we do meta-analyses in the medical field or in psychology. This is quite a problem, as we don't want to find pooled effects to be statistically significant when in fact they are not!

The **Hartung-Knapp-Sidik-Jonkman (HKSJ) method** was thus proposed a way to produce more robust estimates of $var(\hat{\theta}_F)$. It has been shown that this method substantially outperforms the DerSimonian-Laird method in many cases (IntHout et al., 2014). The HKSJ method can also be very easily applied in R, while other programs don't have this option yet. This is another big plus of doing a meta-analysis in *R*. The HKSJ usually leads to more **conservative** results, indicated by wider confidence intervals.

4.2.2.3 Residual concerns with the Hartung-Knapp-Sidik-Jonkman method

It should be noted, however, that the HKSJ method is not uncontroversial. Some authors argue that other (standard) pooling models should also be used **in addition** to the HKSJ as a **sensitivity analysis** (Wiksten et al., 2016). Jackson and colleagues (Jackson et al., 2017) present four residual concerns with this method, which you may take into account before selecting your meta-analytic method. The paper can be read here.

4.2.3 Pre-calculated effect size data

After all this input, you'll see that even random-effects-model meta-analyses are very easy to code in *R*. Compared to the fixed-effects-model Chapter 4.1, there's just three extra parameters we have to define. Especially, as we've described before, we have to tell *R* which **between-study-variance estimator** (τ^2) we want to use, and if we want to use the **Knapp-Hartung(-Sidik-Jonkman)** adjustment.

Here's a table of all parameters we have to define in our code to perform a random-effects-model meta-analysis with pre-calculated effect sizes:

Parameter	Function
TE	This tells R to use the TE column to retrieve the effect sizes for each study
seTE	This tells R to use the seTE column to retrieve the standard error for each study
data=	After =, paste the name of your dataset here
studlab=paste()	This tells the function were the labels for each study are stored. If you named the spreadsheet columns as advised, this should be studlab=paste(Author)
comb.fixed=	Weather to use a fixed-effects-model
comb.random=	Weather to use a random-effects-model. This has to be set to TRUE
method.tau=	Which estimator to use for the between-study variance
hakn=	Weather to use the Knapp-Hartung method
prediction=	Weather to print a prediction interval for the effect of future studies based on present evidence
sm=	The summary measure we want to calculate. We can either calculate the mean difference (MD) or Hedges' g (SMD)

I will use my `madata` dataset again to do the meta-analysis. For illustrative purposes, let's use the Sidik-Jonkman estimator (`"SJ"`) and the HKSJ method. To do this analysis, make sure that `meta` as well as `metafor` are loaded in R.

```
library(meta)
library(metafor)
```

Now, let's code our random-effects-model meta-analysis. Remember, as our effect size data are precalculated, i'll use the `meta::metagen()` function.

```
m.hksj <- metagen(TE,
                  seTE,
                  data=madata,
                  studlab=paste(Author),
                  comb.fixed = FALSE,
                  comb.random = TRUE,
                  method.tau = "SJ",
                  hakn = TRUE,
                  prediction=TRUE,
                  sm="SMD")
m.hksj
```

```
##                           SMD            95%-CI %W(random)
## Call et al.            0.7091 [ 0.1979; 1.2203]        5.2
## Cavanagh et al.        0.3549 [-0.0300; 0.7397]        6.1
## DanitzOrsillo          1.7912 [ 1.1139; 2.4685]        4.2
## de Vibe et al.         0.1825 [-0.0484; 0.4133]        7.1
## Frazier et al.         0.4219 [ 0.1380; 0.7057]        6.8
## Frogeli et al.         0.6300 [ 0.2458; 1.0142]        6.1
## Gallego et al.         0.7249 [ 0.2846; 1.1652]        5.7
## Hazlett-Stevens & Oren 0.5287 [ 0.1162; 0.9412]        5.9
## Hintz et al.           0.2840 [-0.0453; 0.6133]        6.5
## Kang et al.            1.2751 [ 0.6142; 1.9360]        4.3
## Kuhlmann et al.        0.1036 [-0.2781; 0.4853]        6.1
## Lever Taylor et al.    0.3884 [-0.0639; 0.8407]        5.6
## Phang et al.           0.5407 [ 0.0619; 1.0196]        5.4
## Rasanen et al.         0.4262 [-0.0794; 0.9317]        5.3
## Ratanasiripong         0.5154 [-0.1731; 1.2039]        4.1
## Shapiro et al.         1.4797 [ 0.8618; 2.0977]        4.5
## SongLindquist          0.6126 [ 0.1683; 1.0569]        5.7
## Warnecke et al.        0.6000 [ 0.1120; 1.0880]        5.4
##
## Number of studies combined: k = 18
##
##                          SMD            95%-CI    t  p-value
## Random effects model 0.5935 [ 0.3891; 0.7979] 6.13 < 0.0001
## Prediction interval         [-0.2084; 1.3954]
##
## Quantifying heterogeneity:
## tau^2 = 0.1337; H = 1.64 [1.27; 2.11]; I^2 = 62.6% [37.9%; 77.5%]
##
```

```
## Test of heterogeneity:
##      Q d.f. p-value
##  45.50   17  0.0002
##
## Details on meta-analytical method:
## - Inverse variance method
## - Sidik-Jonkman estimator for tau^2
## - Hartung-Knapp adjustment for random effects model
```

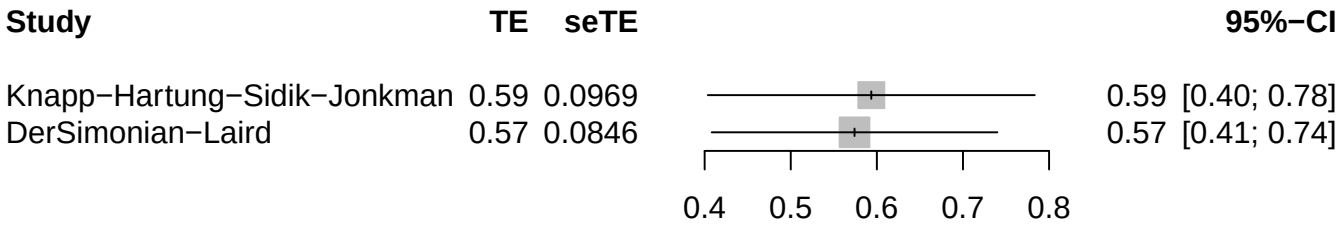
The output shows that our estimated effect is $g = 0.5935$, and the 95% confidence interval stretches from $g = 0.39$ to 0.80 (rounded). It also becomes clear that this effect is different (and larger) than the one we found in the fixed-effects-model meta-analysis in Chapter 4.1 ($g = 0.48$). Let's compare this to the output using the **DerSimonian-Laird** estimator, and when setting `hakn=FALSE`. As this estimator is the **default**, I don't have to define `method.tau` this time.

```
m.dl <- metagen(TE,
                seTE,
                data=madata,
                studlab=paste(Author),
                comb.fixed = FALSE,
                comb.random = TRUE,
                hakn = FALSE,
                prediction=TRUE,
                sm="SMD")
m.dl
```

```
##                            SMD             95%-CI %W(random)
## Call et al.             0.7091 [ 0.1979; 1.2203]        5.0
## Cavanagh et al.         0.3549 [-0.0300; 0.7397]        6.3
## DanitzOrsillo           1.7912 [ 1.1139; 2.4685]        3.7
## de Vibe et al.          0.1825 [-0.0484; 0.4133]        8.0
## Frazier et al.          0.4219 [ 0.1380; 0.7057]        7.4
## Frogeli et al.          0.6300 [ 0.2458; 1.0142]        6.3
## Gallego et al.          0.7249 [ 0.2846; 1.1652]        5.7
## Hazlett-Stevens & Oren 0.5287 [ 0.1162; 0.9412]        6.0
## Hintz et al.            0.2840 [-0.0453; 0.6133]        6.9
## Kang et al.             1.2751 [ 0.6142; 1.9360]        3.8
## Kuhlmann et al.         0.1036 [-0.2781; 0.4853]        6.3
## Lever Taylor et al.     0.3884 [-0.0639; 0.8407]        5.6
## Phang et al.            0.5407 [ 0.0619; 1.0196]        5.3
## Rasanen et al.          0.4262 [-0.0794; 0.9317]        5.1
## Ratanasiripong          0.5154 [-0.1731; 1.2039]        3.6
## Shapiro et al.          1.4797 [ 0.8618; 2.0977]        4.1
## SongLindquist           0.6126 [ 0.1683; 1.0569]        5.7
## Warnecke et al.         0.6000 [ 0.1120; 1.0880]        5.2
##
## Number of studies combined: k = 18
##
```

```
##                          SMD            95%-CI    z  p-value
## Random effects model 0.5741 [ 0.4082; 0.7399] 6.78 < 0.0001
## Prediction interval         [-0.0344; 1.1826]
##
## Quantifying heterogeneity:
## tau^2 = 0.0752; H = 1.64 [1.27; 2.11]; I^2 = 62.6% [37.9%; 77.5%]
##
## Test of heterogeneity:
##      Q d.f. p-value
##  45.50   17  0.0002
##
## Details on meta-analytical method:
## - Inverse variance method
## - DerSimonian-Laird estimator for tau^2
```

We see that the overall effect size estimate using this estimator is similar to the previous one ($g = 0.57$), but the confidence intervals **is narrower because we did not adjust them** using the HKSJ method.



4.2.4 Raw effect size data

I we use raw effect size data, such as the one stored in my `metacont` dataset, we can use the `meta::metacont()` function again. The parameters stay the same as before. I will name the meta-analysis results `m.hksj.raw` this time.

```
m.hksj.raw <- metacont(Ne,
                       Me,
                       Se,
                       Nc,
                       Mc,
                       Sc,
                       data=metacont,
                       studlab=paste(Author),
                       comb.fixed = FALSE,
                       comb.random = TRUE,
                       method.tau = "SJ",
                       hakn = TRUE,
                       prediction=TRUE,
                       sm="SMD")
m.hksj.raw
```

```
##               SMD             95%-CI %W(random)
## Cavanagh -0.4118 [-0.8081; -0.0155]       15.7
```

```
## Day      -0.2687 [-0.6154;  0.0781]       17.7
## Frazier  -0.7734 [-1.0725; -0.4743]       19.7
## Gaffney  -0.7303 [-1.2542; -0.2065]       11.7
## Greer    -0.7624 [-1.0992; -0.4256]       18.1
## Harrer   -0.1669 [-0.5254;  0.1916]       17.2
##
## Number of studies combined: k = 6
##
##                           SMD             95%-CI     t p-value
## Random effects model -0.5161 [-0.8043; -0.2280] -4.60  0.0058
## Prediction interval          [-1.1947;  0.1625]
##
## Quantifying heterogeneity:
## tau^2 = 0.0472; H = 1.51 [1.00; 2.38]; I^2 = 56.1% [0.0%; 82.3%]
##
## Test of heterogeneity:
##      Q d.f. p-value
##  11.39    5  0.0441
##
## Details on meta-analytical method:
## - Inverse variance method
## - Sidik-Jonkman estimator for tau^2
## - Hartung-Knapp adjustment for random effects model
## - Hedges' g (bias corrected standardised mean difference)
```

4.3 Binary outcomes

In some cases, researchers will have to work with **binary outcome data** (e.g., dead/alive, Depressive Disorder/no Depressive Disorder) instead of continuous outcome data. In such a case, you will probably be more interested in outcomes like the pooled **Odd's Ratio**, the **Relative Risk**, or the **Incidence Rate Ratio**. There are two common types of binary outcome data:

- **Event rate data**. This is data in which we are only dealing with the number of persons experiencing an event in each group, and the total sample size in each group. Effect sizes we can calculate from such data are the Odds Ratio, the Relative Risk or the Risk Difference, among others.
- **Incidence rate data**. Event rate data usually does not contain any information on the time span during which events did or did not occur. Given that studies often have drastically different follow-up times (e.g., 8 weeks vs. 2 years), it often makes sense to also take the time interval during which events occured into account. In epidemiology, incidence rates are often used to signify how many events occured within a standardized timeframe (e.g., one year). The corresponding effect size is the incidence rate ratio (IRR), which compares the incidence rate in the intervention group to the one in the control group.

For both event rate data and incidence rate data, there are again two options to pool effect sizes using the `meta` package:

- **The effect sizes are already calculated**. In this case, we can use the `metagen` function as we did before (see Chapter 4.1 and Chapter 4.2), with a few other specifications. We describe how to use the `metagen` function for pre-calculated binary outcome data in Chapter 4.3.3.
- **We only have the raw outcome data**. If this is the case, we will have to use the `meta::metabin()` or function

`meta::metainc()` instead. We'll show you how to do this below.

4.3.1 The Mantel-Haenszel method

In Chapter 4.1, we described the **inverse-variance** method through which effect sizes are pooled in meta-analyses based on continuous outcome data. For binary outcome data, it is common to use a slightly different approach, the **Mantel-Haenszel method** (Mantel and Haenszel, 1959; Robins et al., 1986), to pool results. The formula for this method looks like this:

$$\hat{\theta}_{MH} = \frac{\sum_{k=1}^{K} w_k \hat{\theta}_k}{\sum_{k=1}^{K} w_k} \tag{4.2}$$

This calculation comes very close the the **inverse-variance** method, but the calculation of the study weights w_k is different. This formula for the weights for different effect sizes, with a_k being the number of events in the intervention group, c_k the number of event in the control group, b_k the number of non-events in the intervention group, d_k the number of non-events in the control group and n_k being the total sample size:

Odds Ratio

$$w_k = \frac{b_k c_k}{n_k}$$

Risk Ratio

$$w_k = \frac{(a_k + b_k)c_k}{n_k}$$

Risk Difference

$$w_k = \frac{(a_k + b_k)(c_k + d_k)}{n_k}$$

For binary outcome data, it is generally recommended to use the Mantel-Haenszel method, and it is the default method in *RevMan* (Schwarzer et al., 2015). It is also the default used for binary outcome data in the `meta` package.

4.3.2 Event rate data

For meta-analyses of event rate data, we need our data in the following format:

Column	Description
Author	This signifies the column for the study label (i.e., the first author)
Ee	Number of events in the experimental treatment arm
Ne	Number of participants in the experimental treatment arm
Ec	Number of events in the control arm
Nc	Number of participants in the control arm
Subgroup	This is the label for one of your subgroup codes. It's not that important how you name it, so you can give it a more informative name (e.g. population). In this column, each study should then be given a subgroup code, which should be exactly the same for each subgroup, including upper/lowercase letters. Of course, you can also include more than one subgroup column with different subgroup codings, but the column name has to be unique

I'll use my dataset `binarydata`, which also has this format

```
load("_data/binarydata.RData")
str(binarydata)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    11 obs. of  5 variables:
##  $ Author: chr  "Alcorta-Fleischmann" "Craemer" "Eriksson" "Jones" ...
##  $ Ee    : num  2 18 6 3 0 8 12 1 7 17 ...
##  $ Ne    : num  279 1273 1858 297 300 ...
##  $ Ec    : num  1 17 5 6 1 9 12 10 8 21 ...
##  $ Nc    : num  70 1287 1852 314 295 ...
```

The other parameters are like the ones we used in the meta-analyses with continuous outcome data, with two exceptions:

- **sm**: As we want to have a pooled effect for binary data, we have to choose another summary measure now. We can choose from **"OR"** (Odds Ratio), **"RR"** (Risk Ratio), or **RD** (Risk Difference), among other things.
- **incr**. This lets us define if and how we want **conitinuity correction** to be performed. Such a correction is necessary in cases where one of the cells in your data is zero (e.g., because no one in the intervention arm died). This can be a frequent phenomenon in some contexts, and **distorts our effect size estimates**. By default, the `metabin` function adds the value ***0.5*** in all cells were N is zero (Gart and Zweifel, 1967). This value can be changed using the `incr`-parameter (e.g., `incr=0.1`). If your trial arms are very uneven in terms of their total n, we can also use the **treatment arm continuity correction** (J. Sweeting et al., 2004). This can be done by using `incr="TACC"`.

Here's the code for a meta-analysis with raw binary data

I have decided to run a random-effect-model meta-analysis. I want the summary measure to be the Risk Ratio (RR).

```
m.bin <- metabin(Ee,
                 Ne,
                 Ec,
                 Nc,
                 data=binarydata,
                 studlab=paste(Author),
                 comb.fixed = FALSE,
```

```
                 comb.random = TRUE,
                 method.tau = "SJ",
                 hakn = TRUE,
                 prediction=TRUE,
                 incr=0.1,
                 sm="RR")
m.bin
```

```
##                          RR            95%-CI %W(random)
## Alcorta-Fleischmann 0.5018 [0.0462;  5.4551]        3.0
## Craemer             1.0705 [0.5542;  2.0676]       14.6
## Eriksson            1.1961 [0.3657;  3.9124]        8.5
## Jones               0.5286 [0.1334;  2.0945]        7.0
## Knauer              0.0894 [0.0001; 57.7924]        0.5
## Kracauer            0.9076 [0.3512;  2.3453]       10.9
## La Sala             0.9394 [0.4233;  2.0847]       12.7
## Maheux              0.0998 [0.0128;  0.7768]        3.9
## Schmidthauer        0.7241 [0.2674;  1.9609]       10.3
## van der Zee         0.8434 [0.4543;  1.5656]       15.1
## Wang                0.5519 [0.2641;  1.1534]       13.5
##
## Number of studies combined: k = 11
##
##                           RR           95%-CI     t p-value
## Random effects model 0.7420 [0.5202; 1.0582] -1.87  0.0905
## Prediction interval         [0.2305; 2.3882]
##
## Quantifying heterogeneity:
## tau^2 = 0.2417; H = 1.00 [1.00; 1.36]; I^2 = 0.0% [0.0%; 45.8%]
##
## Test of heterogeneity:
##     Q d.f. p-value
##  7.34   10  0.6929
##
## Details on meta-analytical method:
## - Mantel-Haenszel method
## - Sidik-Jonkman estimator for tau^2
## - Hartung-Knapp adjustment for random effects model
## - Continuity correction of 0.1 in studies with zero cell frequencies
```

4.3.2.1 L'Abbé Plots

So-called **L'Abbé plots** (L'Abbé et al., 1987) are a good way to visualize data based on event rates. In a L'Abbé plot, the event rate of a study's intervention group is plotted against the event rate in the control group, and the N of the study is signified by the size of the bubble in the plot. Despite the simplicity in its principles, this plot allows us to check for three important aspects of our meta-analysis with binary outcomes:

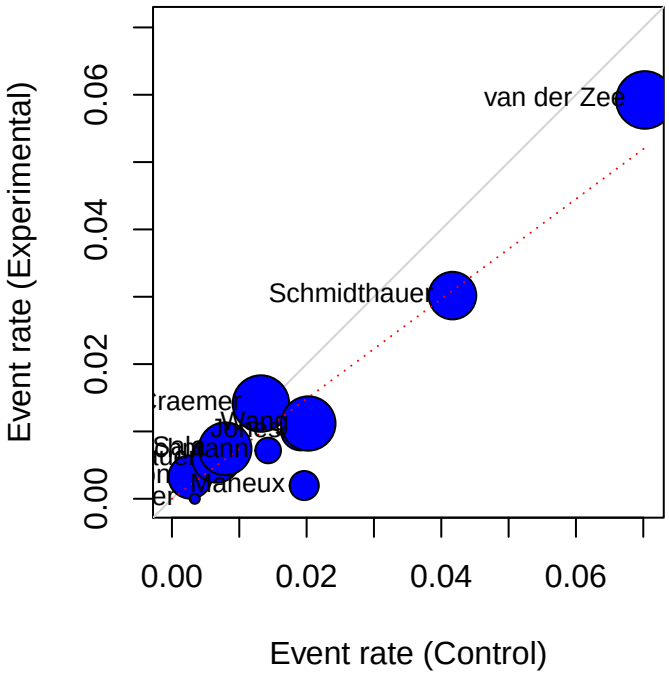
- **The overall trend of our meta-analysis**. If we are expecting a type of intervention to have a protective effect (i.e., making an adverse outcome such as death or depression onset less likely) the studies should mostly lie in the bottom-right corner of the L'Abbé plot, because the control group event rate should be higher than the intervention group event rate. If there's no effect of the intervention compared to the control group, the event rates are identical and the study is shown on the diagonal of the L'Abbé plot.
- **Heterogeneity of effect sizes**. The plot also allows us to eyeball for single studies or groups of studies which contribute to the heterogeneity of the effect we found. It could be the case, for example, that most studies lie in the bottom-right part of the plot as they report positive effects, while a few studies lie in the top-left sector indicated negative effects. Especially if such studies have a small precision (i.e., a small N of participants, indicated by small bubbles in the plot), they could have distorted our pooled effect and may contribute to the between-study heterogeneity.
- **Heterogeneity of event rates**. It may also be the case that some of the heterogeneity in our meta-analysis was introduced by the fact that the event rates "per se" are higher or lower in some studies compared to the others. The L'Abbé plot provides as with this information, as studies with higher event rates will naturally tend towards the top-right corner of the plot.

The results of the `metabin` function can be easily used to generate L'Abbé plots using the `labbe.metabin` function included in the `meta` package. We can specify the following parameters:

Parameter	Description
x	This signifies our metabin meta-analysis output
bg	The background color of the studies
col	The line color of the studies
studlab	Wether the names of the studies should be printed in the plot (TRUE/FALSE)
col.fixed	The color of the dashed line symbolizing the pooled effect of the meta-analysis, if the fixed-effect-model was used
col.random	The color of the dashed line symbolizing the pooled effect of the meta-analysis, if the random-effects-model was used

For this example, i'll use the `m.bin` output i previously generated using the `metabin` function.

```
labbe.metabin(x = m.bin,
              bg = "blue",
              studlab = TRUE,
              col.random = "red")
```



Works like a charm! We see that the **dashed red line** signifying the **pooled effect estimate** of my meta-analysis is running trough the bottom-right sector of my L'Abbé plot, meaning that the overall effect size is positive (i.e., that the intervention has a preventive effect). However, it also becomes clear that **all studies** clearly follow this trend: we see that most studies lie tightly in the bottom-left corner of the plot, meaning that these studies had **small event rates** (i.e., the event in these studies was very rare irrespective of group assignment). We also see that two of our included studies don't fall into this pattern: **Schmidthauer** and **van der Zee**. Those two studies have higher event rates, and both favor the intervention group more clearly than the others did.

4.3.3 Incidence rates

To conduct meta-analyses using incidence rate data, so-called **person-time** data has to be collected or calculated by hand. What it basically needed to calculate person-time data is the **number of events** and the **timeframe** during which they occurred. You can find a general introduction into this topic here, and this quick course by the Centers for Disease Control and Prevention gives a hands-on introduction on how person-time data is calculated. As an example, i will use my `IRR.data` dataset, in which the person-time (in my case, person-year) is stored in the `time.e` and `time.c` column for the experimental and control group, respectively. The `event.e` and `event.c` column contains the number of **events** (in my case, depression onsets) for both groups.

Author	event.e	time.e	event.c	time.c
Stice et al., 2013 (b)	6.000	362.00000	16.000	356.00000
Stice et al. 2017a (clinician-led)	7.000	77.87500	2.670	24.68902
Stice et al. 2017a (peer-led)	3.000	78.45333	2.670	24.68902
Stice et al. 2017a (online)	12.000	79.89667	2.670	24.68902
Stice et al. 2017b (non dissonance-based)	3.000	197.00000	4.500	107.50000
Stice et al. 2017b (dissonance-based)	3.000	197.00000	4.500	107.50000
Taylor et al., 2006	8.352	223.65200	13.332	220.77900
Taylor et al., 2016	22.000	160.00000	29.000	159.00000

To pool the data, we use the `metainc` function included in the `meta` package. We can set the following parameters:

Parameter	Function
event.e	The number of events in the intervention group
time.e	The person-time at risk in the intervention group
event.c	The number of events in the control group
time.c	The person-time at risk in the control group
data=	After =, paste the name of your dataset here
studlab=paste()	This tells the function were the labels for each study are stored. If you named the spreadsheet columns as advised, this should be studlab=paste(Author)
comb.fixed=	Weather to use a fixed-effects-model
comb.random=	Weather to use a random-effects-model. This has to be set to TRUE
method.tau=	Which estimator to use for the between-study variance
hakn=	Weather to use the Knapp-Hartung-Sidik-Jonkman method
prediction=	Weather to print a prediction interval for the effect of future studies based on present evidence
sm=	The summary measure we want to calculate. We want to calculate the Incidence Rate Ratio (IRR)

```
metainc(event.e,
        time.e,
        event.c,
        time.c,
        studlab = paste(Author),
        data = IRR.data,
        sm = "IRR",
        method.tau = "DL",
        comb.random = TRUE,
        comb.fixed = FALSE,
        hakn = TRUE)
```

```
##                                              IRR           95%-CI
## Stice et al., 2013 (b)                    0.3688 [0.1443; 0.9424]
## Stice et al. 2017a (clinician-led)        0.8312 [0.2030; 3.4038]
## Stice et al. 2017a (peer-led)             0.3536 [0.0680; 1.8393]
## Stice et al. 2017a (online)               1.3888 [0.3687; 5.2314]
## Stice et al. 2017b (non dissonance-based) 0.3638 [0.0844; 1.5678]
## Stice et al. 2017b (dissonance-based)     0.3638 [0.0844; 1.5678]
## Taylor et al., 2006                       0.6184 [0.2604; 1.4686]
## Taylor et al., 2016                       0.7539 [0.4332; 1.3121]
##                                             %W(random)
## Stice et al., 2013 (b)                            14.0
## Stice et al. 2017a (clinician-led)                 6.2
```

```
## Stice et al. 2017a (peer-led)                    4.5
## Stice et al. 2017a (online)                      7.0
## Stice et al. 2017b (non dissonance-based)        5.8
## Stice et al. 2017b (dissonance-based)            5.8
## Taylor et al., 2006                             16.5
## Taylor et al., 2016                             40.2
##
## Number of studies combined: k = 8
##
##                          IRR           95%-CI     t p-value
## Random effects model 0.6157 [0.4349; 0.8716] -3.30  0.0131
##
## Quantifying heterogeneity:
## tau^2 = 0; H = 1.00 [1.00; 1.44]; I^2 = 0.0% [0.0%; 51.8%]
##
## Test of heterogeneity:
##     Q d.f. p-value
##  4.71    7  0.6953
##
## Details on meta-analytical method:
## - Mantel-Haenszel method
## - DerSimonian-Laird estimator for tau^2
## - Hartung-Knapp adjustment for random effects model
```

4.3.4 Pre-calculated effect sizes

We can also synthesize pre-calculated effect sizes based on binary outcome data using the `metagen` function. However, there is an important caveat: binary outcome effect sizes are **not on a natural scale**. For effect sizes such as the Risk Ratio or Odds Ratio, our "zero" for no effect is not 0, but 1 instead, because this indicates that the event rates, or the like, are equal between groups. Furthermore, effect size differences are not directly comparable. A Risk Ratio of 0.5, for example, is not simply the "opposite" of a Risk Ratio of 1.5. While the first effect size means that the event rate has been halved, the latter does **not** mean that is has doubled. To get effect sizes on a natural scale, it is common to **log-transform** the effect sizes first before they are pooled. While this is done automatically in the `metabin` and `metainc` function, we have to do this step ourselves once we use the `metagen` function. Let's have a look at my `bin.metagen` dataset:

```
bin.metagen %>%
  kable() %>%
  kable_styling(position = "center", latex_options = c("striped")) %>%
  row_spec(0, bold = T)
```

Author	RR	lower	upper
Dorfman	0.7420459	0.5222924	1.0542602
Chieung	0.6992909	0.4827775	1.0129050
Haber	0.8270375	0.6486705	1.0544507
Xu	0.8208534	0.5268752	1.2788613
Fleischmann	0.8193257	0.5926804	1.1326420
Knapp	1.1183327	0.9411318	1.3288979
Hartung	0.9141726	0.8596176	0.9721899

This dataset contains the Risk Ratio as well as the lower and upper confidence interval of seven studies. To produce the input needed for the metagen function, we first log-transform all the effect size data.

```
bin.metagen$RR <- log(bin.metagen$RR)
bin.metagen$lower <- log(bin.metagen$lower)
bin.metagen$upper <- log(bin.metagen$upper)
```

We can calculate the **Standard Error** seTE like this:

```
bin.metagen$seTE = (bin.metagen$upper - bin.metagen$lower)/3.92
```

We now have everything we need to use the metagen function for our effect size data. To reconvert effect sizes to their original scale, we specificy sm="RR" in the function:

```
metagen(RR,
        seTE,
        studlab = Author,
        method.tau = "SJ",
        sm = "RR",
        data = bin.metagen)
```

```
##                 RR           95%-CI %W(fixed) %W(random)
## Dorfman     0.7420 [0.5223; 1.0543]       2.4        9.3
## Chieung     0.6993 [0.4828; 1.0129]       2.1        8.6
## Haber       0.8270 [0.6487; 1.0544]       4.9       14.8
## Xu          0.8209 [0.5269; 1.2789]       1.5        6.6
## Fleischmann 0.8193 [0.5927; 1.1326]       2.8       10.4
## Knapp       1.1183 [0.9411; 1.3289]       9.7       20.2
## Hartung     0.9142 [0.8596; 0.9722]      76.6       30.0
##
## Number of studies combined: k = 7
##
##                         RR           95%-CI     z p-value
## Fixed effect model   0.9137 [0.8658; 0.9643] -3.28  0.0010
## Random effects model 0.8826 [0.7772; 1.0023] -1.92  0.0544
##
## Quantifying heterogeneity:
## tau^2 = 0.0131; H = 1.29 [1.00; 1.98]; I^2 = 39.6% [0.0%; 74.6%]
##
```

```
## Test of heterogeneity:
##     Q d.f. p-value
##  9.93    6  0.1277
##
## Details on meta-analytical method:
## - Inverse variance method
## - Sidik-Jonkman estimator for tau^2
```

Please be aware that the `metagen` function can only use the generic inverse-variance method for pooling, and not the Mantel-Haenszel method. If possible, you should therefore always use the `metabin` (or `metainc`) function.

4.4 Correlations

Performing a meta-analysis of **correlations** is not too different from the methods we described before. Commonly, the **generic inverse-variance pooling** method is also used to combine correlations from different studies into one pooled correlation estimate. When pooling correlations, it is advised to perform **Fisher's z-transformation** to obtain accurate weights for each study. Luckily, we do not have to do this transformation ourselves. There is an additional function for meta-analyses of correlations included in the `meta` package, the `metacor` function, which does most of the calculations for us.

The parameters of the `metacor` function are mostly identical to the `metagen` and `metacont` function we described before (see Chapter 4.1 and 4.2), as is the statistics behind it. To use the function, we again need a dataset with the study label (`Author`), the **correlation** r (cor) reported for each study, and the **sample size** (n) for each study. Let's have a look at my `cordata` dataset, which follows this structure:

```
str(cordata)
```

```
## 'data.frame': 8 obs. of 3 variables:
## $ Author: Factor w/ 8 levels "Altman2011","Borenstein2013",..: 7 5 1 8 2 3 4
6
## $ cor : num 0.456 0.234 0.235 0.297 0.374 0.561 0.102 0.377
## $ n : num 150 67 45 80 230 110 78 45
```

We can use this dataset to calculate a meta-analysis using the `metacor` function. Like in the chapters before, we will use a **random-effects model** with the **Sidik-Jonkman** estimator for the between-study heterogeneity τ^2. In addition, we set `sm` to `"ZCOR"` to use the z-transformed correlations for the meta-analysis (as is advised). The rest of the syntax remains identical. I will call the results of the meta-analysis `m.cor`.

```
m.cor <- metacor(cor,
                 n,
                 data = cordata,
                 studlab = cordata$Author,
                 sm = "ZCOR",
                 method.tau = "SJ")
```

Let's have a look at the **output**, which has the same structure as the one of the `metagen`, `metabin` and `metainc` functions.

```
##                   COR            95%-CI %W(fixed) %W(random)
## Mantel1999     0.4560 [ 0.3191; 0.5743]      18.8       15.3
## Haenszel1987   0.2340 [-0.0066; 0.4490]       8.2       11.3
## Altman2011     0.2350 [-0.0629; 0.4944]       5.4        9.1
## Olkin1999      0.2970 [ 0.0827; 0.4851]       9.9       12.3
## Borenstein2013 0.3740 [ 0.2571; 0.4801]      29.1       16.8
## Egger2001      0.5610 [ 0.4176; 0.6771]      13.7       13.9
## Glass2007      0.1020 [-0.1233; 0.3173]       9.6       12.1
## Ioannidis2010  0.3770 [ 0.0939; 0.6037]       5.4        9.1
##
## Number of studies combined: k = 8
##
##                          COR           95%-CI     z  p-value
## Fixed effect model   0.3693 [0.3072; 0.4282] 10.83 < 0.0001
## Random effects model 0.3491 [0.2378; 0.4515]  5.85 < 0.0001
##
## Quantifying heterogeneity:
## tau^2 = 0.0186; H = 1.56 [1.06; 2.31]; I^2 = 59.0% [10.6%; 81.2%]
##
## Test of heterogeneity:
##      Q d.f. p-value
##  17.09    7  0.0168
##
## Details on meta-analytical method:
## - Inverse variance method
```

```
## - Sidik-Jonkman estimator for tau^2
## - Fisher's z transformation of correlations
```

As can be seen from the output, the pooled correlation in this dataset is $r = 0.35$ (for the random-effects model), which is significant ($p < 0.0001$). The I^2-heterogeneity in this analysis is substantial (58%), supporting the use of a random-effects model.

Chapter 5

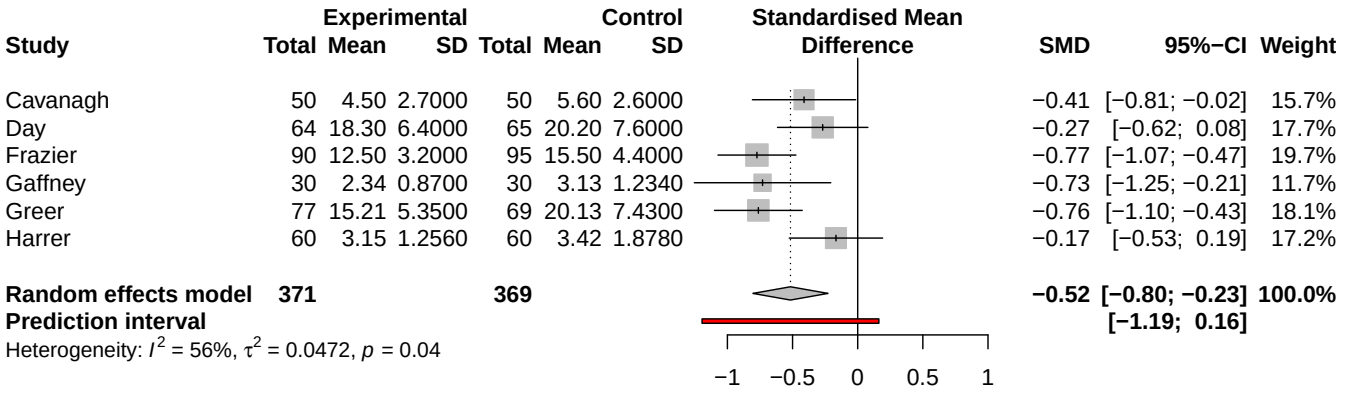
Forest Plots

Now that we created the **output of our meta-analysis** using the `metagen`, `metacont` or `metabin` functions in `meta` (see Chapter 4.1,Chapter 4.2 and Chapter 4.3), it is time to present the data in a more digestable way. **Forest Plots** are an easy way to do this, and it is conventional to report forest plots in meta-analysis publications.

5.1 Generating a Forest Plot

To produce a forest plot, we use the meta-analysis output we just created (e.g., `m`, `m.raw`) und the `meta::forest()` function. I'll use my `m.hksj.raw` output from Chapter 4.2.3 to create the forest plot

```
forest(m.hksj.raw)
```

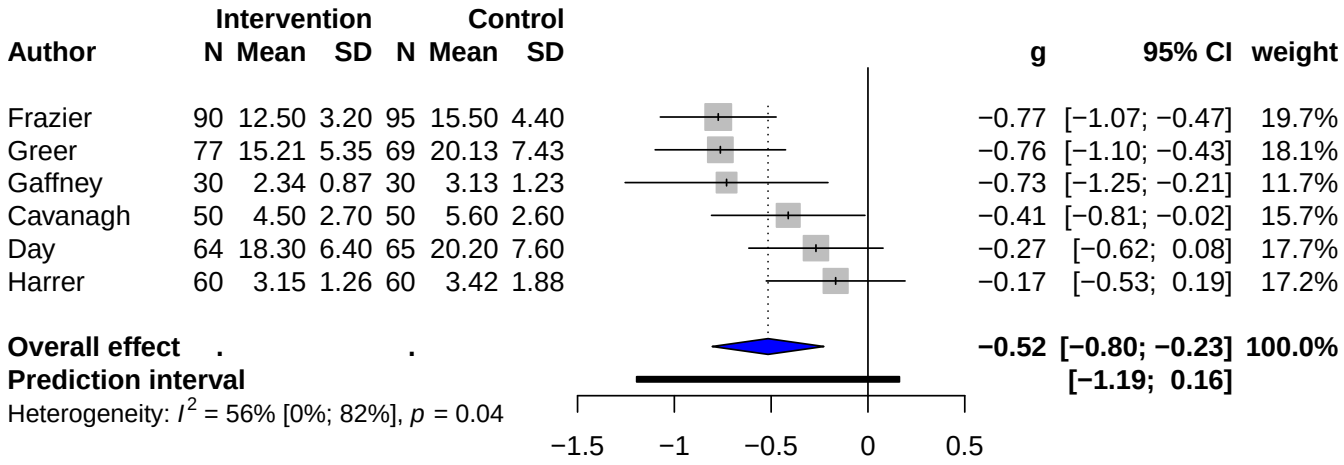


Looks good so far. We see that the function plotted a forest plot with a **diamond** (i.e. the overall effect and its confidence interval) and a **prediction interval**. There are plenty of **other parameters** within the `meta::forest` function which we can use to modify the forest plot.

Type	Parameter	Description
General	sortvar	A sorting variable. For example, you can sort the forest plot by effect size using 'TE', or by author name using 'Author'
General	studlab	This tells the function which variable should be printed as the study label. The standard is 'Author'
General	comb.fixed	Whether fixed effect estimate should be plotted. (TRUE/FALSE)
General	comb.random	Whether random effects estimate should be plotted. (TRUE/FALSE)
General	overall	Whether overall summaries should be plotted. This argument is useful in a meta-analysis with subgroups if summaries should only be plotted on group level.
General	text.fixed	A character string used in the plot to label the pooled fixed effect estimate. Has to be put in "" (e.g. "Overall effect")
General	text.random	A character string used in the plot to label the pooled random effect estimate. Has to be put in "" (e.g. "Overall effect")
General	col.fixed	Line colour (pooled fixed effect estimate). E.g. "red", "blue", or hex color code ("#2e8aff")
General	col.random	Line colour (random fixed effect estimate). E.g. "red", "blue", or hex color code ("#2e8aff")
General	prediction	Whether a prediction interval should be printed.
General	text.predict	A character string used in the plot to label the prediction interval. E.g. "Prediction Interval"
General	subgroup	A logical indicating whether subgroup results should be shown in forest plot. This argument is useful in a meta-analysis with subgroups if summaries should not be plotted on group level. (TRUE/FALSE)
General	print.subgroup.labels	A logical indicating whether subgroup label should be printed. (TRUE/FALSE)
General	study.results	Whether results for individual studies should be shown in the figure (useful to only plot subgroup results). (TRUE/FALSE)
General	xlab	A label for the x-axis on the bottom. Put in "".
General	smlab	A label for the summary measure on top. Put in "".
General	xlim	The x limits (min,max) of the plot, or the character "s" to produce symmetric forest plots. This is particularly revelant when your results deviate substantially from zero, or if you also want to have outliers depicted. (e.g. xlim=c(0,1.5) for effects from 0 to 1.5).
General	ref	The reference value to be plotted as a line in the forest plot. This is interesting if you want to compare effects to common thresholds or results from previous analyses (e.g. ref=0.5)
General	leftcols	Here you can specify all variables which should be printed on the left side of your plot. The variables have to be part of you meta-analysis output, so check with name_of_your_output$ which variables can be displayed. E.g. leftcols=c("TE","seTE").
General	rightcols	Same as leftcols, but for the right side of your plot
General	leftlabs	This specifies how the columns on the left side should be named. Always provide all labels (e.g. leftlabs=c("Author","Effect size","Standard error"))
General	rightlabs	Same as leftlabs, but for the left side of you plot
General	print.I2	Whether to print the value of the I-squared statistic.
General	print.I2.ci	Whether to print the confidence interval of the I-squared statistic.
General	squaresize	A numeric used to increase or decrease the size of squares in the forest plot. (E.g., squaresize = 1.2)
Color	col.study	The colour for individual study results and confidence limits. E.g. 'red', 'blue', or hex color code ('#2e8aff')
Color	col.inside	The colour for individual study results and confidence limits if confidence limits are completely within squares. E.g. 'red', 'blue', or hex color code ('#2e8aff')
Color	col.square	The colour for squares reflecting study's weight in the meta-analysis. E.g. 'red', 'blue', or hex color code ('#2e8aff')
Color	col.square.lines	The colour for the outer lines of squares reflecting study's weight in the meta-analysis. E.g. 'red', 'blue', or hex color code ('#2e8aff')
Color	col.diamond	The colour of diamonds representing the results for fixed effect and random effects models. E.g. 'red', 'blue', or hex color code ('#2e8aff')
Color	col.diamond.fixed	The colour of diamonds for fixed effect estimates. E.g. 'red', 'blue', or hex color code ('#2e8aff')
Color	col.diamond.random	The colour of diamonds for random effects estimates. E.g. 'red', 'blue', or hex color code ('#2e8aff')
Color	col.diamond.lines	The colour of the outer lines of diamonds representing the results for fixed effect and random effects models. E.g. 'red', 'blue', or hex color code ('#2e8aff')
Color	col.diamond.lines.fixed	The colour of the outer lines of diamond for fixed effect estimate. E.g. 'red', 'blue', or hex color code ('#2e8aff')
Color	col.diamond.lines.random	The colour of the outer lines of diamond for random effects estimate. E.g. 'red', 'blue', or hex color code ('#2e8aff')
Color	col.inside.fixed	The colour for result of fixed effect meta-analysis if confidence limit lies completely within square. E.g. 'red', 'blue', or hex color code ('#2e8aff')
Color	col.inside.random	The colour for result of random effects meta-analysis if confidence limit lies completely within square. E.g. 'red', 'blue', or hex color code ('#2e8aff')
Color	col.predict	Background colour of prediction interval. E.g. 'red', 'blue', or hex color code ('#2e8aff')
Color	col.predict.lines	Colour of outer lines of prediction interval. E.g. 'red', 'blue', or hex color code ('#2e8aff')
Color	col.label.right	The colour for label on right side of null effect. E.g. 'red', 'blue', or hex color code ('#2e8aff')
Color	col.label.left	The colour for label on left side of null effect. E.g. 'red', 'blue', or hex color code ('#2e8aff')
Digits	digits	Minimal number of significant digits for treatment effects (TE)
Digits	digits.se	Minimal number of significant digits for standard errors
Digits	digits.zval	Minimal number of significant digits for z- or t-statistic for test of overall effect
Digits	digits.tau2	Minimal number of significant digits for between-study variance
Digits	digits.pval	Minimal number of significant digits for p-value of overall treatment effect
Digits	digits.pval.Q_	Minimal number of significant digits for p-value of heterogeneity test
Digits	digits.Q_	Minimal number of significant digits for heterogeneity statistic Q_
Digits	digits.I2	Minimal number of significant digits for I-squared statistic
Digits	digits.weight	Minimal number of significant digits for weights
Digits	digits.mean	Minimal number of significant digits for the mean
Digits	digits.sd	Minimal number of significant digits for the standard deviations

This is again just an overview. For all settings, type `?meta::forest` in your **console** to see more. Let's play around with the function a little now:

```
forest(m.hksj.raw,
       sortvar=TE,
       xlim = c(-1.5,0.5),
       rightlabs = c("g","95% CI","weight"),
       leftlabs = c("Author", "N","Mean","SD","N","Mean","SD"),
       lab.e = "Intervention",
       pooled.totals = FALSE,
       smlab = "",
       text.random = "Overall effect",
       print.tau2 = FALSE,
       col.diamond = "blue",
       col.diamond.lines = "black",
       col.predict = "black",
       print.I2.ci = TRUE,
       digits.sd = 2
)
```



Looks good so far! For special **layout types**, proceed to Chapter 5.2 now.

5.2 Layout types

The `meta::forest` function also has two **Layouts** preinstalled which we can use. Those layouts can be accessed with the `layout=` parameter.

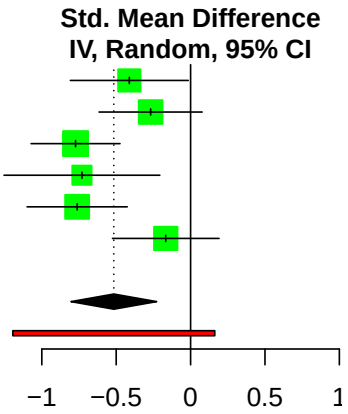
- **"RevMan5"**. This layout is used for Cochrane reviews and generated by *Review Manager 5*.
- **"JAMA"**. This layout gives you a forest plot according to the guidelines of the *Journal of the American Medical Association* as output (see details here).

The **RevMan** layout looks like this:

```
forest(m.hksj.raw,
       layout = "RevMan5",
       digits.sd = 2)
```

	Experimental			Control				Std. Mean Difference
Study	**Mean**	**SD**	**Total**	**Mean**	**SD**	**Total**	**Weight**	**IV, Random, 95% CI**
Cavanagh	4.50	2.70	50	5.60	2.60	50	15.7%	−0.41 [−0.81; −0.02]
Day	18.30	6.40	64	20.20	7.60	65	17.7%	−0.27 [−0.62; 0.08]
Frazier	12.50	3.20	90	15.50	4.40	95	19.7%	−0.77 [−1.07; −0.47]
Gaffney	2.34	0.87	30	3.13	1.23	30	11.7%	−0.73 [−1.25; −0.21]
Greer	15.21	5.35	77	20.13	7.43	69	18.1%	−0.76 [−1.10; −0.43]
Harrer	3.15	1.26	60	3.42	1.88	60	17.2%	−0.17 [−0.53; 0.19]
Total (95% CI)			**371**			**369**	**100.0%**	**−0.52 [−0.80; −0.23]**
Prediction interval								**[−1.19; 0.16]**

Heterogeneity: Tau^2 = 0.0472; Chi^2 = 11.39, df = 5 (P = 0.04); I^2 = 56%

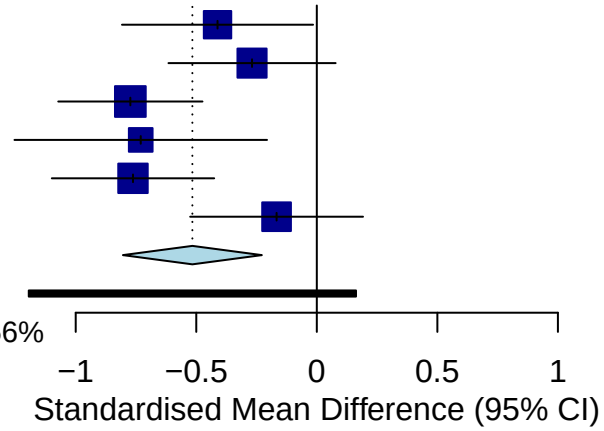


The **JAMA** layout looks like this:

```
forest(m.hksj.raw,
       layout = "JAMA",
       text.predict = "95% PI",
       col.predict = "black",
       colgap.forest.left = unit(15,"mm"))
```

Source	SMD (95% CI)
Cavanagh	−0.41 [−0.81; −0.02]
Day	−0.27 [−0.62; 0.08]
Frazier	−0.77 [−1.07; −0.47]
Gaffney	−0.73 [−1.25; −0.21]
Greer	−0.76 [−1.10; −0.43]
Harrer	−0.17 [−0.53; 0.19]
Total	−0.52 [−0.80; −0.23]
95% PI	[−1.19; 0.16]

Heterogeneity: $\chi^2_5 = 11.39$ (P = .04), I^2 = 56%



5.3 Saving the forest plots

Let's say i want to save the JAMA version of my Forest Plot now. To do this, i have to reuse the code with which i plotted my forest plot, and put it between `pdf(file='name_of_the_pdf_i_want_to_create.pdf')` and `dev.off`, both in separate lines. This saves the plot into a PDF in my Working Directory. This way, i can export the plot in different formats (you can find more details on the saving options here).

PDF

```
pdf(file='forestplot.pdf')
forest.jama<-forest(m.hksj.raw,
```

```
       layout = "JAMA",
       text.predict = "95% PI",
       col.predict = "black",
       colgap.forest.left = unit(15,"mm"))
dev.off()
```

PNG

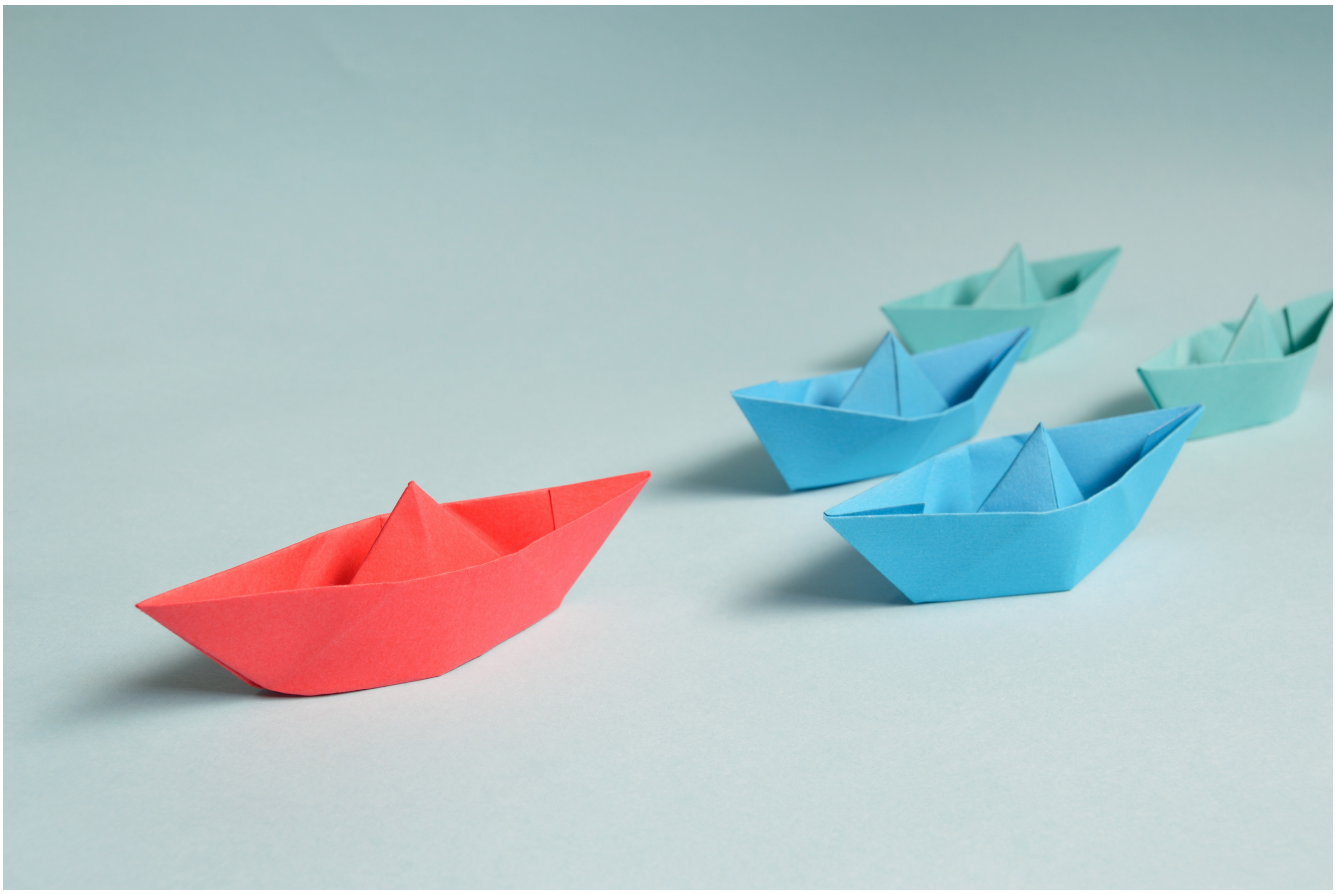
```
png(file='forestplot.png')
forest.jama<-forest(m.hksj.raw,
       layout = "JAMA",
       text.predict = "95% PI",
       col.predict = "black",
       colgap.forest.left = unit(15,"mm"))
dev.off()
```

Scalable Vector Graphic

```
svg(file='forestplot.svg')
forest.jama<-forest(m.hksj.raw,
       layout = "JAMA",
       text.predict = "95% PI",
       col.predict = "black",
       colgap.forest.left = unit(15,"mm"))
dev.off()
```

Chapter 6

Between-study Heterogeneity



By now, we have already shown you how to pool effect sizes in a meta-analysis. In meta-analytic pooling, we aim to **synthesize the effects of many different studies into one single effect**. However, this makes only sense if we aren't comparing **Apples and Oranges**. For example, it could be the case that while the overall effect we calculate in the meta-analysis is **small**, there are still a few studies which report **very high** effect sizes. Such information is lost in the aggregate effect, but it is very important to know if all studies, or interventions, yield small effect sizes, or if there are exceptions. It could also be the case that even some very **extreme effect sizes** were included in the meta-analysis, so-called **outliers**. Such outliers might have even distorted our overall effect, and it is important to know how our overall effect would have looked without them. The extent to which effect sizes vary within a meta-analysis is called **heterogeneity**. It is very important to assess heterogeneity in meta-analyses, as high heterogeneity could

be caused by the fact that there are actually two or more **subgroups** of studies present in the data, which have a different true effect. Such information could be very valuable for **research**, because this might allow us to find certain interventions or populations for which effects are lower or higher. From a statistical standpoint, high heterogeneity is also problematic. Very high heterogeneity could mean that the studies have nothing in common, and that there is no **"real" true effect behind our data**, meaning that it makes no sense to report the pooled effect at all (Borenstein et al., 2011).

6.1 The idea behind heterogeneity

Rücker and colleagues (Rücker et al., 2008) name three types of heterogeneity in meta-analyses:

1. *Clinical baseline heterogeneity*. These are differences between sample characteristics between the studies. For example, while one study might have included rather old people into their study, another might have recruited study participants who were mostly quite young.
2. *Statistical heterogeneity*. This is the statistical heterogeneity we find in our collected effect size data. Such heterogeneity migh be either important from a clinical standpoint (e.g., when we don't know if a treatment is very or only marginally effective because the effects vary much from study to study), or from statistical standpoint (because it dilutes the confidence we have in our pooled effect)
3. *Other sources of heterogeneity*, such as design-related heterogeneity.

Point 1. and 3. may be controlled for to some extent by restricting the scope of our search for studies to certain well-defined intervention types, populations, and outcomes. Point 2., on the other hand, has to be assessed once we conducted the pooling of studies. This is what this chapter focuses on.

6.2 Heterogeneity Measures

There are **three types of heterogeneity measures** which are commonly used to assess the degree of heterogeneity. In the following examples, k denotes the individual study, K denotes all studies in our meta-analysis, $\hat{\theta}_k$ is the estimated effect of k with a variance of $\hat{\sigma}_k^2$, and w_k is the individual **weight** of the study (i.e., its *inverse variance*: $w_k = \frac{1}{\hat{\sigma}_k^2}$; see infobox in Chapter 4.1.1 for more details).

1. Cochran's Q

Cochran's Q-statistic is the **difference between the observed effect sizes and the fixed-effect model estimate** of the effect size, which is then **squared, weighted and summed**.

$$Q = \sum_{k=1}^{K} w_k(\hat{\theta}_k - \frac{\sum_{k=1}^{K} w_k\hat{\theta}_k}{\sum_{k=1}^{K} w_k})^2$$

2. Higgin's & Thompson's I^2

I^2 (Higgins and Thompson, 2002) is the **percentage of variability** in the effect sizes which is not caused by sampling error. It is derived from Q:

$$I^2 = max\left\{0, \frac{Q-(K-1)}{Q}\right\}$$

3. Tau-squared

τ^2 is the between-study variance in our meta-analysis. As we show in Chapter 4.2.1, there are various proposed ways to calculate τ^2

6.2.1 Which measure should i use?

Generally, when we assess and report heterogeneity in a meta-analysis, we need a measure which is **robust, and not to easily influenced by statistical power**. **Cochran's Q** increases both when the **number of studies** (k) increases, and when the **precision** (i.e., the sample size N of a study) increases. Therefore, Q and weather it is **significant** highly depends on the size of your meta-analysis, and thus its statistical power. We should therefore not only rely on Q when assessing heterogeneity. **I^2** on the other hand, is not sensitive to changes in the number of studies in the analyses. I^2 is therefore used extensively in medical and psychological research, especially since there is a **"rule of thumb"** to interpret it (Higgins et al., 2003):

- I^2 = 25%: **low heterogeneity**
- I^2 = 50%: **moderate heterogeneity**
- I^2 = 75%: **substantial heterogeneity**

Despite its common use in the literature, I^2 not always an adequate measure for heterogeneity either, because it still heavily depends on the **precision** of the included studies (Rücker et al., 2008; Borenstein et al., 2017). As said before, I^2 is simply the amount of variability **not caused by sampling error**. If our studies become increasingly large, this sampling error tends to **zero**, while at the same time, I^2 tends to 100% simply because the single studies have greater N. Only relying on I^2 is therefore not a good option either. **Tau-squared**, on the other hand, is **insensitive** to the number of studies, **and** the precision. Yet, it is often hard to interpret how relevant our tau-squared is from a practical standpoint.

Prediction intervals (like the ones we automatically calculated in Chapter 4) are a good way to overcome this limitation (IntHout et al., 2016), as they take our between-study variance into account. Prediction intervals give us a range for which we can **expect the effects of future studies to fall** based on **our present evidence in the meta-analysis**. If our prediction interval, for example, lies completely on the positive side favoring the intervention, we can be quite confident to say that **despite varying effects, the intervention might be at least in some way beneficial in all contexts we studied in the future**. If the confidence interval includes **zero**, we can be less sure about this, although it should be noted that **broad prediction intervals are quite common, especially in medicine and psychology**.

6.3 Assessing the heterogeneity of your pooled effect size

Thankfully, once you've already pooled your effects in meta-analysis using the `metagen()`, `metabin()`, or `metacont` function, it is very easy and straightforward to retrieve the **three most common heterogeneity measures** that we described before. In Chapter 4.2.2, we already showed you how to conduct a **random-effect-model meta-analysis**. In this example, we stored our **results** in the object `m.hksj`, which we will use again here.

One way to get heterogeneity measures of my meta-analysis is to **print** the meta-analysis (in my case, `m.hksj`) output again.

```
print(m.hksj)
```

```
##                            SMD            95%-CI %W(random)
## Call et al.             0.7091 [ 0.1979; 1.2203]        5.2
## Cavanagh et al.         0.3549 [-0.0300; 0.7397]        6.1
## DanitzOrsillo           1.7912 [ 1.1139; 2.4685]        4.2
## de Vibe et al.          0.1825 [-0.0484; 0.4133]        7.1
## Frazier et al.          0.4219 [ 0.1380; 0.7057]        6.8
## Frogeli et al.          0.6300 [ 0.2458; 1.0142]        6.1
## Gallego et al.          0.7249 [ 0.2846; 1.1652]        5.7
## Hazlett-Stevens & Oren 0.5287 [ 0.1162; 0.9412]        5.9
## Hintz et al.            0.2840 [-0.0453; 0.6133]        6.5
## Kang et al.             1.2751 [ 0.6142; 1.9360]        4.3
## Kuhlmann et al.         0.1036 [-0.2781; 0.4853]        6.1
## Lever Taylor et al.     0.3884 [-0.0639; 0.8407]        5.6
## Phang et al.            0.5407 [ 0.0619; 1.0196]        5.4
## Rasanen et al.          0.4262 [-0.0794; 0.9317]        5.3
## Ratanasiripong          0.5154 [-0.1731; 1.2039]        4.1
## Shapiro et al.          1.4797 [ 0.8618; 2.0977]        4.5
## SongLindquist           0.6126 [ 0.1683; 1.0569]        5.7
## Warnecke et al.         0.6000 [ 0.1120; 1.0880]        5.4
##
## Number of studies combined: k = 18
##
##                          SMD            95%-CI    t  p-value
## Random effects model 0.5935 [ 0.3891; 0.7979] 6.13 < 0.0001
## Prediction interval         [-0.2084; 1.3954]
##
## Quantifying heterogeneity:
## tau^2 = 0.1337; H = 1.64 [1.27; 2.11]; I^2 = 62.6% [37.9%; 77.5%]
##
## Test of heterogeneity:
##      Q d.f. p-value
##  45.50   17  0.0002
##
## Details on meta-analytical method:
## - Inverse variance method
## - Sidik-Jonkman estimator for tau^2
## - Hartung-Knapp adjustment for random effects model
```

We see that this output **already provides us with all three heterogeneity measures** (and even one more, *H*, which we will not cover here).

- τ^2, as we can see from the `tau^2` output, is **0.1337**.
- I^2 is printed next to `I^2`, and has the value **62.6%**, and a 95% confidence interval rangin from 37.9% to 77.5%.

- The value of Q is displayed next to `Q` under `Test of heterogeneity:`. As we can see, the value is ***45.50***. In our case, this is highly significant ($p = 0.0002$; see `p-value`).
- The **prediction interval** can be found next to `Prediction interval`. As we can see, the 95% interval ranges from **g=-0.2084** to **1.3954**.

How can we interpret the values of this example analysis? Well, all three of our indicators suggest that **moderate to substantial heterogeneity is present in our data**. Given the **broad prediction interval**, which stretches well below zero, we also cannot be overly confident that the positive effect we found for our interventions is robust in every context. It might be very well possible that the intervention does not yield positive effects in some future scenarios; even a small negative effect might be possible based on the evidence the meta-analysis gives us. Very high effect sizes, on the other hand, are possible too.

6.3.1 When the measures are not displayed in my output

Depending on how you changed the settings of the `metagen`, `metabin`, or `metacont`, it is possible that some of the measures are not displayed in your output. That's not a big deal, because all measures are stored in the object, no matter if they are immediately displayed or not. To directly access one of the measures, we can to use `$` again (see Chapter 3.3.1). We use this **in combination with our meta-analysis output object** to define which measure we want to see.

Code	Measure
$Q_	Cochran's Q_
$pval.Q_	The p-value for Cochran's Q_
$I2	I-squared
$lower.I2	The lower bound of the I-squared 95%CI
$upper.I2	The upper bound of the I-squared 95%CI
$tau^2	Tau-squared
$lower.predict	The lower bound of the 95% prediction interval
$upper.predict	The upper bound of the 95% prediction interval

Here are a few exmaples for my `m.hksj` object. As you'll see, the output is **identical** to the one before.

```
m.hksj$Q
```

```
## [1] 45.50257
```

```
m.hksj$I2
```

```
## [1] 0.6263947
```

```
m.hksj$tau^2
```

```
## [1] 0.1337024
```

6.4 Detecting Outliers & Influential Cases

As mentioned before, **between-study heterogeneity** can also be caused by one more studies with **extreme effect sizes** which don't quite **fit in**. Especially when the **quality of these studies is low**, or the **studies are very small**, this may **distort** our pooled effect estimate, and it's a good idea to have a look on the **pooled effect again once we remove such outliers from the analysis**.

On the other hand, we also want to know **if the pooled effect estimate we found is robust**, meaning that the effect does not depend heavily on **one single study**. Therefore, we also want to know **whether there are studies which heavily push the effect of our analysis into some direction**. Such studies are called **influential cases**, and we'll devote some time to this topic in the second part of this chapter.

It should be noted that they are **many methods** to spot **outliers and influential cases**, and the methods described here are not comprehensive. If you want to read more about the underpinnings of this topic, we can recommend the paper by Wolfgang Viechtbauer and Mike Cheung (Viechtbauer and Cheung, 2010).

6.4.1 Searching for extreme effect sizes (outliers)

A common method to detect outliers directly is to define a study as an outlier if the **study's confidence interval does not overlap with the confidence interval**. This means that we define a study as an outlier when it's effect size estimate is **so extreme that we have high certainty that the study cannot be part of the "population" of effect size we determined when pooling our results** (i.e., the individual study differs significantly from the overall effect). To detect such outliers in our dataset, the `filter` function in the `dplyr` package we introduced in Chapter 3.3.3 comes in handy again. Using this function, we can search for all studies:

- for which the **upper bound of the 95% confidence interval is lower than the lower bound of the pooled effect confidence interval** (i.e., extremely small effects)
- for which the **lower bound of the 95% confidence interval is higher than the higher bound of the pooled effect confidence interval** (i.e., extremely large effects)

Here, i'll use my `m.hksj` meta-analysis output from Chapter 4.2.2 again. Let's see what the **upper and lower bound of my pooled effect confidence interval** is. As i performed a **random-effect meta-analysis in this example**, i will use the value stored under `$lower.random` and `$upper.random`. If you performed a **fixed-effect meta-analysis**, the objects would be `$lower.fixed` and `$upper.fixed`, respectively.

```
m.hksj$lower.random
```

```
## [1] 0.389147
```

```
m.hksj$upper.random
```

```
## [1] 0.7979231
```

Here we go. I now see that my **pooled effect confidence interval** stretches from $g = 0.389$ to $g = 0.798$. We can use these values to filter out outliers now. To filter out outliers **automatically**, we have prepared two **functions** for you, `spot.outliers.random` and `spot.outliers.fixed`. Both need the `dplyr` package (see Chapter 3.3.3) to function, so we need to need to have this package **installed** and **loaded into our library**.

```
library(dplyr)
```

The function we'll use in the case of my `m.hksj` dataset is `spot.outliers.random`, because we conducted a **random-effect meta-analysis to get this output object**. R doesn't know this function yet, so we have to let R learn it copying and pasting the code underneath **in its entirety** into the **console** on the bottom left pane of RStudio, and then hit **Enter**.

```
spot.outliers.random<-function(data){

    data<-data
    Author<-data$studlab
    lowerci<-data$lower
    upperci<-data$upper
    m.outliers<-data.frame(Author,lowerci,upperci)
    te.lower<-data$lower.random
    te.upper<-data$upper.random
    dplyr::filter(m.outliers,upperci < te.lower)
    dplyr::filter(m.outliers,lowerci > te.upper)

}
```

Now, the function is ready to be used. The only thing we have to tell the `spot.outliers.random` function is the **meta-analysis output** that we want to check for outliers, which is defined by `data`. In my case, this is `m.hksj`.

```
spot.outliers.random(data=m.hksj)
```

This is the output we get from the function:

```
##           Author   lowerci  upperci
## 1  DanitzOrsillo 1.1138668 2.468473
## 2 Shapiro et al. 0.8617853 2.097667
```

We see that the function has detected **two outliers**. Looking at the `lowerci` value, the lower bound of the two study's confidence intervals, we see that both have extremely high positive effects, because the lower bounds are both much higher than the **higher bound of the confidence interval of our pooled effect**, which was $g = 0.798$.

Thus, we can conduct a **sensitivity analysis** in which we **exclude these two outliers**. We can do this with the `update.meta` function in `meta`. This creates an update of our meta-analysis output `m.hksj` without the outliers.

```
m.hksj.outliers<-update.meta(m.hksj,
                             subset = Author != c("DanitzOrsillo",
                                                  "Shapiro et al."))
m.hksj.outliers
```

```
##                           SMD            95%-CI %W(random)
## Call et al.            0.7091 [ 0.1979; 1.2203]        5.0
## Cavanagh et al.        0.3549 [-0.0300; 0.7397]        6.9
## de Vibe et al.         0.1825 [-0.0484; 0.4133]       10.4
## Frazier et al.         0.4219 [ 0.1380; 0.7057]        9.1
## Frogeli et al.         0.6300 [ 0.2458; 1.0142]        7.0
## Gallego et al.         0.7249 [ 0.2846; 1.1652]        6.0
## Hazlett-Stevens & Oren 0.5287 [ 0.1162; 0.9412]        6.4
## Hintz et al.           0.2840 [-0.0453; 0.6133]        8.1
```

```
## Kang et al.            1.2751 [ 0.6142; 1.9360]        3.5
## Kuhlmann et al.        0.1036 [-0.2781; 0.4853]        7.0
## Lever Taylor et al.    0.3884 [-0.0639; 0.8407]        5.8
## Phang et al.           0.5407 [ 0.0619; 1.0196]        5.4
## Rasanen et al.         0.4262 [-0.0794; 0.9317]        5.1
## Ratanasiripong         0.5154 [-0.1731; 1.2039]        3.3
## SongLindquist          0.6126 [ 0.1683; 1.0569]        5.9
## Warnecke et al.        0.6000 [ 0.1120; 1.0880]        5.3
##
## Number of studies combined: k = 16
##
##                          SMD           95%-CI    t  p-value
## Random effects model 0.4708 [0.3406; 0.6010] 7.71 < 0.0001
## Prediction interval         [0.0426; 0.8989]
##
## Quantifying heterogeneity:
## tau^2 = 0.0361; H = 1.15 [1.00; 1.56]; I^2 = 24.8% [0.0%; 58.7%]
##
## Test of heterogeneity:
##      Q d.f. p-value
##  19.95   15  0.1739
##
## Details on meta-analytical method:
## - Inverse variance method
## - Sidik-Jonkman estimator for tau^2
## - Hartung-Knapp adjustment for random effects model
```

The entire procedure works the same if you **conducted a fixed-effect meta-analysis**. However, you need to copy and paste the code for the `spot.outliers.fixed` function then, which can be found **below**.

```
spot.outliers.fixed<-function(data){
data<-data
Author<-data$studlab
lowerci<-data$lower
upperci<-data$upper
m.outliers<-data.frame(Author,lowerci,upperci)
te.lower<-data$lower.fixed
te.upper<-data$upper.fixed
dplyr::filter(m.outliers,upperci < te.lower)
dplyr::filter(m.outliers,lowerci > te.upper)
}
```

6.5 Influence Analyses

We have now showed you how you can detect and remove **extreme effect sizes** (outliers) in your meta-analysis. As we have mentioned before, however, it is not only statistical outliers which may cause concerns regarding the robustness

of our pooled effect. It is also possible that **some studies in a meta-analysis exert a very high influence on our overall results**. For example, it could be the case that we find that an overall effect is not significant, when in fact, a highly significant effect is consistently found once we remove one particular study in our analysis. Such information is **highly important once we want to communicate the results of our meta-analysis to the public**.

Here, we present techniques which dig a little deeper than simple outlier removal. To some extent, they are based on the **Leave-One-Out**-method, in which we **recalculate the results of our meta-analysis** $K-1$ times, each times leaving out one study. This way, we can more easily detect **studies which influence the overall estimate of our meta-analysis the most**, and this lets us better assess if this **influence may distort our pooled effect** (Viechtbauer and Cheung, 2010). Thus, such analyses are called **Influence Analyses**. We have created the **function** `InfluenceAnalysis` for you, in which these analyses are conducted and results are visualized all in one. This function is part of the `dmetar` package. If you have the package installed already, you have to load it into your library first.

```
library(dmetar)
```

If you don't want to use the `dmetar` package, you can find the source code for this function here. In this case, *R* doesn't know this function yet, so we have to let *R* learn it by **copying and pasting** the code **in its entirety** into the **console** on the bottom left pane of RStudio, and then hit **Enter**. The function requires the `ggplot2`, `ggrepel`, `forcats`, `dplyr`, `grid`, `gridExtra`, `metafor`, and `meta` package to work.

The `InfluenceAnalysis` function has **several parameters** which we have to define.

Code	Description
x	An object of class meta, generated by the metabin, metagen, metacont, metacor, metainc, or metaprop function
random	Logical. Should the random-effects model be used to generate the influence diagnostics? Uses the method.tau specified in the meta object if one of 'DL', 'HE', 'SJ', 'ML', 'REML', 'EB', 'HS' or 'GENQ' (to ensure compatibility with the metafor package). Otherwise, the DerSimonian-Laird ('DL'; DerSimonian & Laird, 1986) estimator is used. FALSE by default.
subplot.heights	Concatenated array of two numerics. Specifies the heights of the first (first number) and second (second number) row of the overall results plot generated by the function. Default is c(30,18).
subplot.widths	Concatenated array of two numerics. Specifies the widths of the first (first number) and second (second number) column of the overall results plot generated by the function. Default is c(30,30).
forest.lims	Concatenated array of two numerics. Specifies the x-axis limits of the forest plots generated by the function. Use 'default' if standard settings should be used (this is the default).
return.separate.plots	Logical. Should the influence plots be returned as seperate plots in lieue of returning them in one overall plot? Additionally returns a dataframe Data containing the data used for plotting. If set to TRUE, the ouput of the function must be saved in a variable; specific plots can then be accessed by selecting the plot element and using plot().
text.scale	Positive numeric. Scaling factor for the text geoms used in the plot. Values <1 shrink the text, while values >1 increase the text size. Default is 1.

This is how the function code looks for my `m.hksj` data:

```
InfluenceAnalysis(x = m.hksj,
                  random = TRUE)
```

Now, let's have a look at the output.

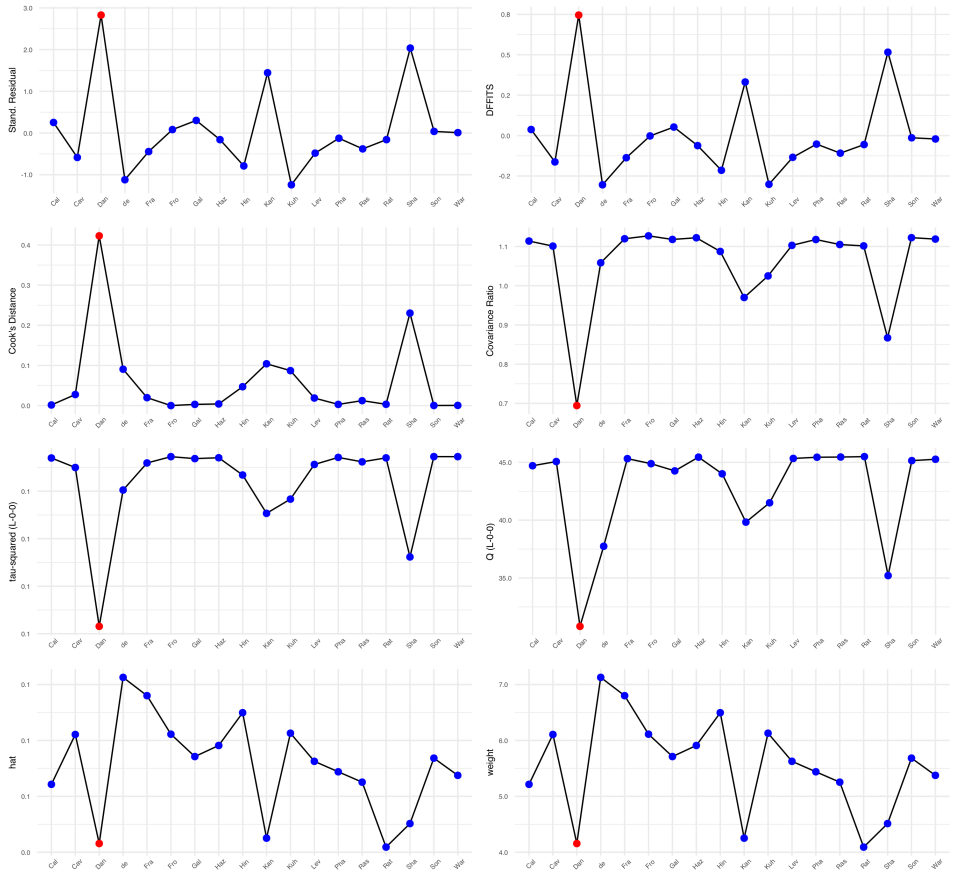


Figure 6.1: Influence Analyses

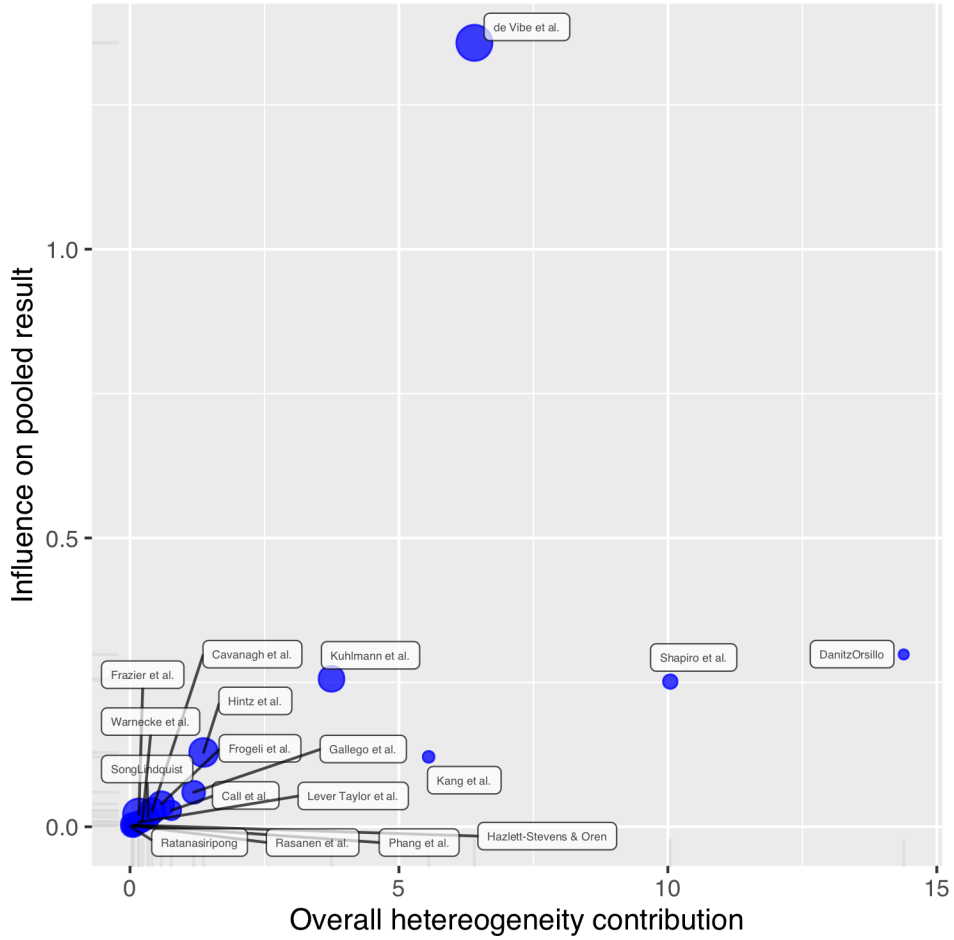


Figure 6.2: Baujat Plot

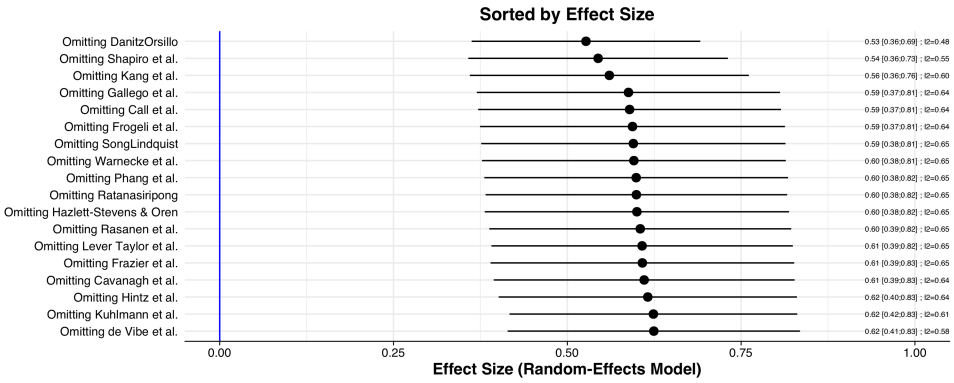


Figure 6.3: Leave-One-Out-Analyses, sorted by effect size

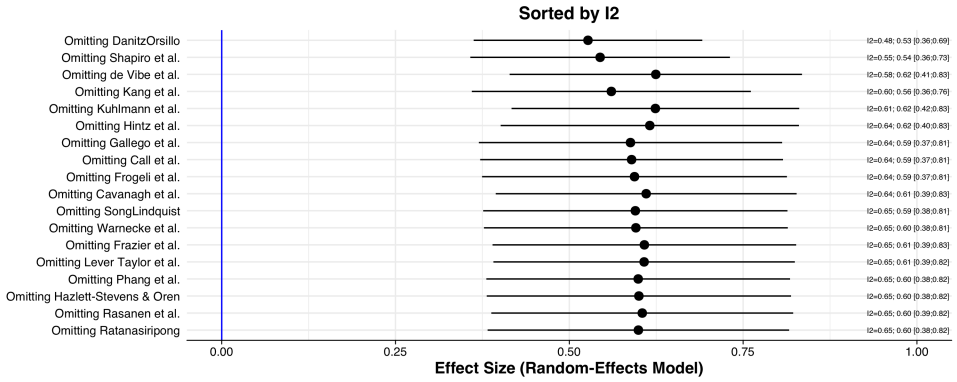


Figure 6.4: Leave-One-Out-Analyses, sorted by heterogeneity

As you can see, the `influence.analysis` function gives us various types of **plots** as output. Let's interpret them one by one.

6.5.1 Influence Analyses

In the first analysis, you can see different influence measures, for which we can see **graphs including each individual study of our meta-analysis**. This type of **influence analysis** has been proposed by Viechtbauer and Cheung (Viechtbauer and Cheung, 2010). We'll discuss the most important subplots here:

- **dffits**: The DIFFITS value of a study indicates in standard deviations how much the predicted pooled effect changes after excluding this study.
- **cook.d**: The **Cook's distance** resembles the **Mahalanobis distance** you may know from outlier detection in conventional multivariate statistics. It is the distance between the value once the study is included compared to when it is excluded.
- **cov.r**: The **covariance ratio** is the determinant of the variance-covariance matrix of the parameter estimates when the study is removed, divided by the determinant of the variance-covariance matrix of the parameter estimates when the full dataset is considered. Importantly, values of cov.r < 1 indicate that removing the study will lead to a more precise effect size estimation (i.e., less heterogeneity).

Usually, however, you don't have to dig this deep into the calculations of the individual measures. As a rule of thumb, **influential cases** are studies with **very extreme values in the graphs**. Viechtbauer and Cheung have also proposed cut-offs when to define a a study as an influential case, for example (with p being the number of model coefficients and k the number of studies):

$$DFFITS > 3 \times \sqrt{\frac{p}{k-p}}$$

$$hat > 3 \times \frac{p}{k}$$

If a case was determined being **an influential case using these cut-offs**, its value will be displayed in **red** (in our example, this is the case for study "Dan", or Danitz-Orsillo).

Please note, as Viechtbauer & Cheung emphasize, that **these cut-offs are set on somewhat arbitrary thresholds**. Therefore, you should never only have a look on the color of the study, but the general structure of the graph, and

interpret results in context. In our example, we see that while only the study by Danitz-Orsillo is defined as an influential case, there are **actually two spiked in most plots**, while the other studies all quite have the same value. Given this structure, we could also decide to define **Study "Sha"** (Shapiro et al.) as an influential case too, because its values are very extreme too.

In these analyses, we found that the studies "Danitz-Orsillo" and "Shapiro et al." might be influential. This is an interesting finding, as we **detected the same studies when only looking at statistical outliers**. This further corroborates that these two studies could maybe have distorted our pooled effect estimate, and **might cause parts of the between-group heterogeneity we found in our meta-analysis**.

6.5.2 Baujat Plot

The Baujat Plot (Baujat et al., 2002) is a diagnostic plot to detect studies **overly contributing to the heterogeneity of a meta-analysis**. The plot shows the contribution of each study to the overall heterogeneity as measured by Cochran's Q on the **horizontal axis**, and its **influence on the pooled effect size** on the vertical axis. As we want to assess heterogeneity and studies contributing to it, all studies **on the right side of the plot are important to look at**, as this means that they cause much of the heterogeneity we observe. **This is even more important when a study contributes much to the overall heterogeneity, while at the same time being not very influential concerning the overall pooled effect** (e.g., because the study had a very small sample size). Therefore, **all studies on the right side of the Baujat plot**, especially in the **lower part**, are important for us. As you might have already recognized, the only two **studies we find in those regions of the plot are the two studies we already detected before** (Danitz & Orsillo, Shapiro et al.). These studies don't have a large impact on the overall results (presumably because they are very small), but they do **add substantially to the heterogeneity we found in the meta-analysis**.

6.5.3 Leave-One-Out Analyses

In these to forest plots, we see the **pooled effect recalculated, with one study omitted each time**. There are two plots, which provide the same data, but are ordered by different values. The **first plot is ordered by heterogeneity (low to high), as measured by I^2**. We see in the plot that the lowest $I*^2$ heterogeneity is reached (as we've seen before) by omitting the studies **Danitz & Orsillo** and **Shapiro et al.**. This again corroborates our finding that these two studies were the main "culprits" for the between-study heterogeneity we found in the meta-analysis. The **second plot is ordered by effect size (low to high)**. Here, we see how the overall effect estimate changes with one study removed. Again, as the two outlying studies have very high effect sizes, we find that the overall effect is smallest when they are removed. All in all, the results of our **outlier and influence analysis** in this example point in the **same direction**. The two studies are probably **outliers which may distort the effect size estimate**, as well as its **precision**. We should therefore also conduct and report a **sensitivity analysis in which these studies are excluded**.

6.6 GOSH Plot Analysis

In the previous subchapter, we explored the robustness of our meta-analysis results using influence analyses and the leave-one-out method. An even more sophisticated way to explore the patterns of effect sizes and heterogeneity in our data are so-called **Graphic Display of Heterogeneity** (**GOSH**) plots (Olkin et al., 2012). For those plots, we fit the same meta-analysis model to **all possible subsets** of our included studies. In constrast to the leave-one-out

method, we therefore not only fit $K - 1$ models, but all 2^{k-1} possible study combinations. This means that creating GOSH plot can become quite computationally intensive when the number of included studies is large. The *R* implementation we cover here therefore only fits a maximum of 1 million randomly selected models.

Once the models are calculated, we can plot them, displaying the **pooled effect size** on the **x-axis** and the **between-study heterogeneity** at the **y-axis**. This allows us to look for specific patterns, for example subclusters with different effect sizes. This would indicate that there is in fact more than one "population" of effect sizes in our data, warranting a subgroup analysis. If the effect sizes in our sample are homogeneous, the GOSH plot should form a **symmetric distribution with one peak**. To generate GOSH plots, we can use the gosh function in the `metafor` package. If you have not installed the package yet, install it and then load it from your library.

```
library(metafor)
```

I will generate plots for my `m.hksj` meta-analysis object. Before we can generate the plot, we have to **"transform"** this object created by the `meta` package into a `metafor` meta-analysis object which can be used for the gosh function. To do this, i can simply use the data stored in `m.hksj`. The function to perform a meta-analysis in `metafor` is called `rma()`. I only have to provide the function with the effect size (TE), Standard Error (seTE), and between-study heterogeneity estimator (`method.tau`) stored in `m.hksj`. Because i want to replicate the results completely, i also use Knapp-Hartung adjustments in `rma()` by setting `test = "knha"`.

```
m.rma <- rma(yi = m.hksj$TE,
             sei = m.hksj$seTE,
             method = m.hksj$method.tau,
             test = "knha")
```

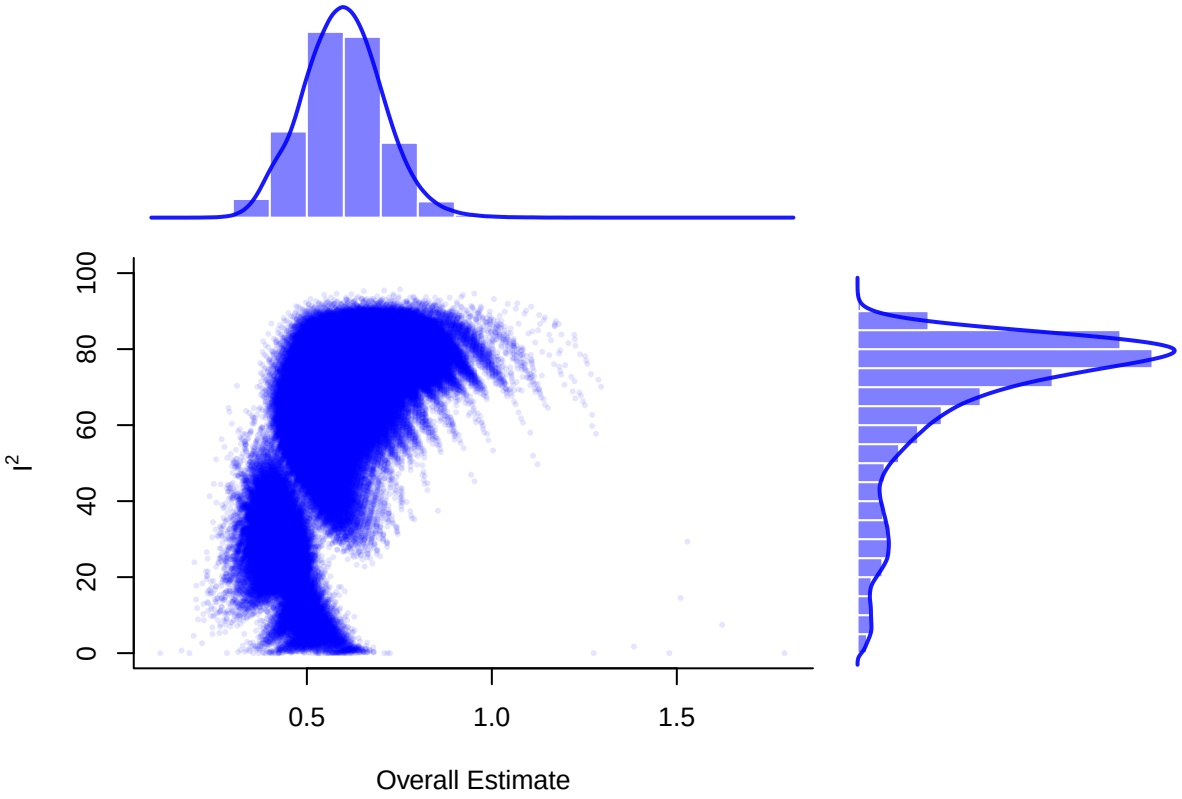
Please note that if you use the fixed-effect model, you have to set `method = "FE"`.

We can then use this object to generate the GOSH plot object. Depending on the number of studies in your analysis, this can take some time, up to a few hours. I save it as `dat.gosh`.

```
dat.gosh <- gosh(m.rma)
```

I can then display the the GOSH plot by pluggin the `dat.gosh` object into the `plot()` function.

```
plot(dat.gosh, alpha= 0.1, col = "blue")
```



Interestingly, we see that there are **two patterns** in our data: one which lower effect sizes and lower heterogeneity, and one with higher effects, but considerable between-study heterogeneity: it seems like there are **two subclusters in our data**.

Now that we know the **effect size-heterogeneity** patterns in our data, the really important question of course is: which studies cause those patterns, and may thus belong to which subcluster? To answer this question we have developed the `gosh.diagnostics` function. This function uses **three clustering (also known as supervised machine learning) algorithms** to detect clusters in the GOSH plot data and determine which studies contribute the most to them automatically. The function is part of the `dmetar` package. If you have the package installed already, you have to load it into your library first.

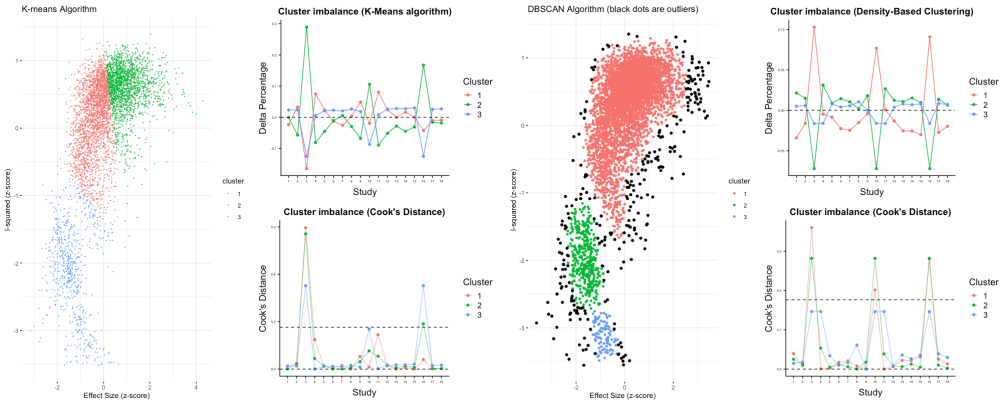
```
library(dmetar)
```

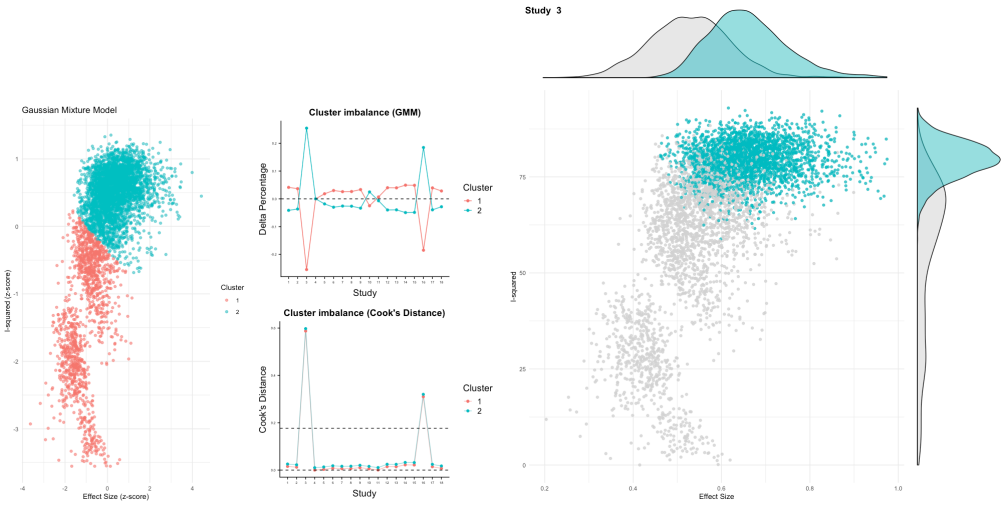
If you don't want to use the `dmetar` package, you can find the source code for this function here. In this case, *R* doesn't know this function yet, so we have to let *R* learn it by **copying and pasting** the code **in its entirety** into the **console** in the bottom left pane of RStudio, and then hit **Enter**. The function then requires the `dplyr`, `cluster`, `mvtnorm`, `factoextra`, `fpc`, `cowplot`, `reshape2`, and `flexmix` package installed and loaded in your library to function properly. We will use the default settings for the function here, but many more are available (see documentation). We simply plug the `dat.gosh` object into the function. Again, as the there is a lot of data to process, the function may take need some time before it is finished.

```
gosh.diagnostics(dat.gosh)
```

```
 Perform Clustering...
[=============================================================] DONE

 Number of k-means clusters used: 3
```

```
Number of DBSCAN clusters detected: 3

Number of GMM clusters detected: 2

Identification of potential outliers
--------------------------------

Studies identified as potential outliers:

- K-means: 3 16

- DBSCAN: 3 10 16

- Gaussian Mixture Model: 3 16
```





We see that the **three algorithms**, k-means, DBSCAN and the Gaussian Mixture Model, have detected **three studies** which might potentially contribute to the **cluster imbalance**: study 3, study 10, and study 16. These studies seem to nearly fully account for the second high-effect size, high-heterogeneity cluster we found.

Let us see what happens if we rerun the meta-analysis, this time removing these three studies.

```
metagen(TE,
        seTE,
        data=madata,
        studlab=paste(Author),
        comb.fixed = FALSE,
        comb.random = TRUE,
        method.tau = "SJ",
        hakn = TRUE,
        prediction=TRUE,
        sm="SMD",
        exclude = c(3, 10, 16))
```

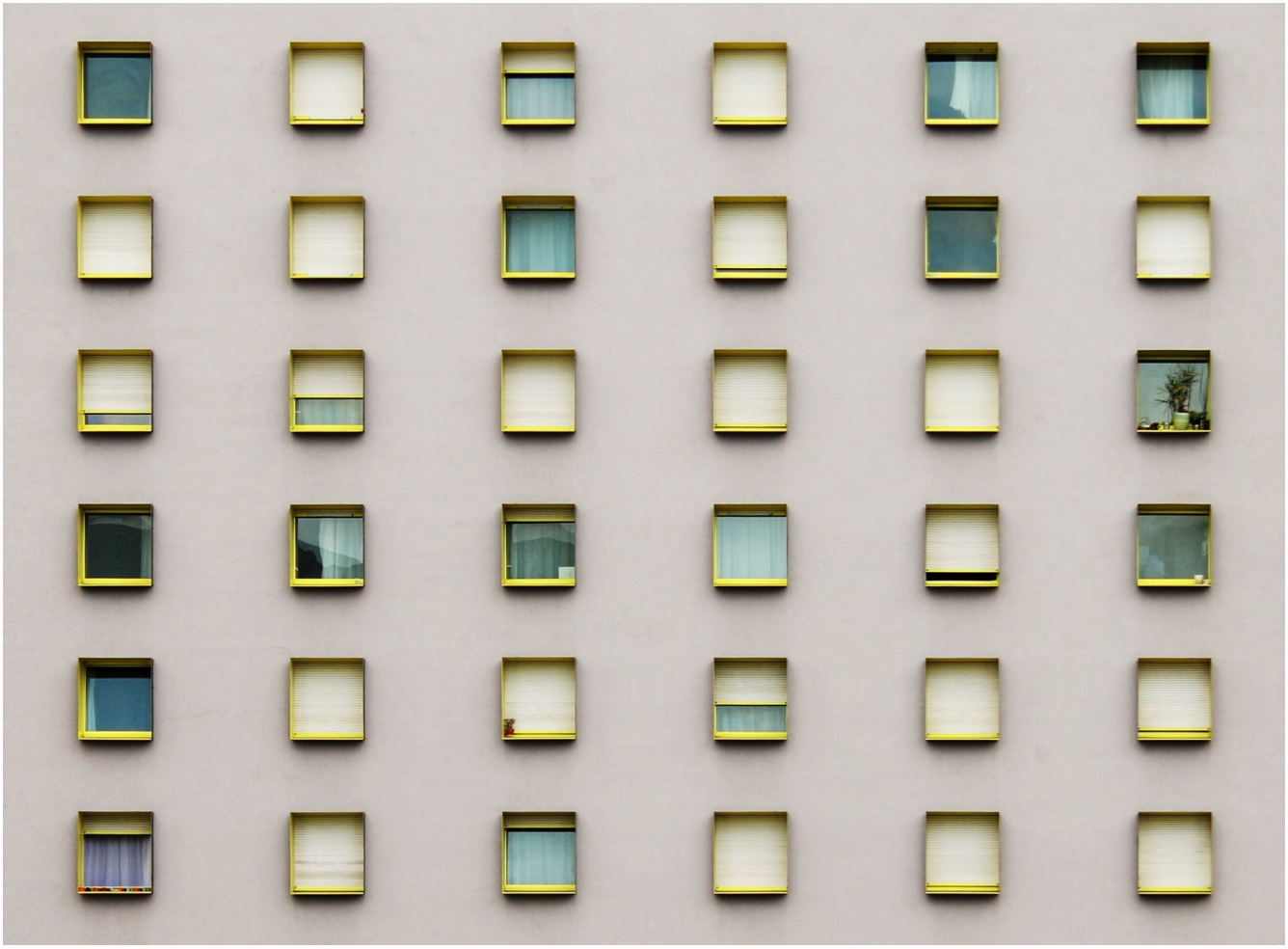
```
##                          SMD            95%-CI %W(random) exclude
## Call et al.           0.7091 [ 0.1979; 1.2203]        4.6
## Cavanagh et al.       0.3549 [-0.0300; 0.7397]        7.1
## DanitzOrsillo         1.7912 [ 1.1139; 2.4685]        0.0       *
## de Vibe et al.        0.1825 [-0.0484; 0.4133]       13.2
## Frazier et al.        0.4219 [ 0.1380; 0.7057]       10.6
## Frogeli et al.        0.6300 [ 0.2458; 1.0142]        7.2
## Gallego et al.        0.7249 [ 0.2846; 1.1652]        5.8
## Hazlett-Stevens & Oren 0.5287 [ 0.1162; 0.9412]        6.5
## Hintz et al.          0.2840 [-0.0453; 0.6133]        8.8
## Kang et al.           1.2751 [ 0.6142; 1.9360]        0.0       *
## Kuhlmann et al.       0.1036 [-0.2781; 0.4853]        7.2
## Lever Taylor et al.   0.3884 [-0.0639; 0.8407]        5.6
## Phang et al.          0.5407 [ 0.0619; 1.0196]        5.1
## Rasanen et al.        0.4262 [-0.0794; 0.9317]        4.7
## Ratanasiripong        0.5154 [-0.1731; 1.2039]        2.8
```

```
## Shapiro et al.         1.4797 [ 0.8618; 2.0977]        0.0       *
## SongLindquist          0.6126 [ 0.1683; 1.0569]        5.8
## Warnecke et al.        0.6000 [ 0.1120; 1.0880]        5.0
##
## Number of studies combined: k = 15
##
##                          SMD           95%-CI    t  p-value
## Random effects model 0.4299 [0.3234; 0.5364] 8.66 < 0.0001
## Prediction interval         [0.1428; 0.7169]
##
## Quantifying heterogeneity:
## tau^2 = 0.0152; H = 1.00 [1.00; 1.44]; I^2 = 0.0% [0.0%; 51.8%]
##
## Test of heterogeneity:
##      Q d.f. p-value
##  13.48   14  0.4890
##
## Details on meta-analytical method:
## - Inverse variance method
## - Sidik-Jonkman estimator for tau^2
## - Hartung-Knapp adjustment for random effects model
```

We see that the between-study heterogeneity has **dropped to** 0%, indicating that the studies we are now analyzing **stem from one homogeneous (sub)population**, the lower cluster in our GOSH plot.

Chapter 7

Subgroup Analyses



In Chapter 6, we discussed in depth why **between-study heterogeneity** is such an important issue in interpreting the results of our meta-analysis, and how we can **explore sources of heterogeneity** using outlier and influence analyses. Another source of between-study heterogeneity making our effect size estimate less precise could be that **there are slight differences in the study design or intervention components between the studies**. For example, in a meta-analysis on the effects of **cognitive behavioral therapy** (CBT) for **depression** in **university students**, it could be the case that some studies delivered the intervention in a **group setting**, while others delivered the therapy to each student

individually. In the same example, it is also possible that studies used different **criteria** to determine if a student suffers from **depression** (e.g. they either used the *ICD-10* or the *DSM-5* diagnostic manual). Many other differences of this sort are possible, and it seems plausible that such study differences may also be associated with differences in the overall effect. In **subgroup analyses**, we therefore have a look at different **subgroups within the studies of our meta-analysis** and try to determine of the **differ between these subgroups**.

7.1 The idea behind subgroup analyses

Basically, a every subgroup analysis consists of **two parts**: (1) **pooling the effect of each subgroup**, and (2) **comparing the effects of the subgroups** (Borenstein and Higgins, 2013).

7.1.1 1. Pooling the effect of each subgroup

This point it rather straightforward, as the same criteria as the ones for a **simple meta-analysis without subgroups** (see Chapter 4 and Chapter 4.2) apply here.

- If you assume that **all studies in subgroup** stem from the same population, and all have **one shared true effect**, you may use the **fixed-effect-model**. As we mention in Chapter 4, many **doubt** that this assumption is ever **true in psychological** and **medical research**, even when we partition our studies into subgroups.
- The alternative, therefore, is to use a **random-effect-model** which assumes that the studies within a subgroup are drawn from a **universe** of populations follwing its own distribution, for which we want to estimate the **mean**.

7.1.2 2. Comparing the effects of the subgroups

After we calculated the pooled effect for each subgroup, **we can compare the size of the effects of each subgroup**. However, to know if this difference is in fact singnificant and/or meaningful, we have to calculate the **Standard Error of the differences between subgroup effect sizes** SE_{diff}, to calculate **confidence intervals** and conduct **significance tests**. There are **two ways to calculate** SE_{diff}, and both based on different assumptions.

- **Fixed-effects (plural) model**: The fixed-effects-model for subgroup comparisons is appropriate when **we are only interested in the subgroups at hand** (Borenstein and Higgins, 2013). This is the case when **the subgroups we chose to examine** were not randomly "chosen", but represent fixed levels of a characteristic we want to examine. Gender is such a characteristic, as its two subgroups **female** and **male** were not randomly chosen, but are the two subgroups that gender (in its classical conception) has. Same does also apply, for example, if we were to examine if studies in patients with **clinical depression** versus **subclinical depression** yield different effects. Borenstein and Higgins (@ Borenstein and Higgins, 2013) argue that the **fixed-effects (plural) model** may be the **only plausible model** for most analysis in **medical research, prevention, and other fields**.

As this model assumes that **no further sampling error is introduced at the subgroup level** (because subgroups were not randomly sampled, but are fixed), SE_{diff} only depends on the *variance within the subgroups* A and B, V_A and V_B.

$$V_{Diff} = V_A + V_B$$

The fixed-effects (plural) model can be used to test differences in the pooled effects between subgroups, while the pooling **within the subgroups is still conducted using a random-effects-model**. Such a combination is sometimes called a **mixed-effects-model**. We'll show you how to use this model in R in the next chapter.

- **Random-effects-model**: The random-effects-model for between-subgroup-effects is appropriate when the **subgroups we use were randomly sampled from a population of subgroups**. Such an example would be if we were interested if the effect of an intervention **varies by region** by looking at studies from 5 different countries (e.g., Netherlands, USA, Australia, China, Argentina). These variable "region" has many different potential subgroups (countries), from which we randomly selected five means that this has introduced a **new sampling error**, for which we have to control for using the **random-effects-model** for between-subgroup-comparisons.

The (simplified) formula for the estimation of V_{Diff} using this model therefore looks like this:

$$V_{Diff} = V_A + V_B + \frac{\hat{T}_G^2}{m}$$

Where $\hat{T}_G^2$ is the **estimated variance between the subgroups**, and m is the **number of subgroups**.

Be aware that subgroup analyses should **always be based on an informed, *a priori* decision** which subgroup differences within the study might be **practically relevant**, and would lead to information gain on relevant **research questions** in your field of research. It is also **good practice** to specify your subgroup analyses **before you do the analysis**, and list them in **the registration of your analysis**. It is also important to keep in mind that **the capabilites of subgroup analyses to detect meaningful differences between studies is often limited**. Subgroup analyses also need **sufficient power**, so it makes no sense to compare two or more subgroups when your entire number of studies in the meta-analysis is smaller than $k = 10$ (Higgins and Thompson, 2004).

7.2 Subgroup Analyses using the Mixed-Effects-Model

To conduct subgroup analyses using the **Mixed-Effects-Model** (random-effects-model within subgroups, fixed-effects-model between subgroups), you can use the `subgroup.analysis.mixed.effects` function we prepared for you. This function is part of the `dmetar` package. If you have the package installed already, you have to load it into your library first.

```
library(dmetar)
```

If you don't want to use the `dmetar` package, you can find the source code for this function here. In this case, *R* doesn't know this function yet, so we have to let *R* learn it by **copying and pasting** the code **in its entirety** into the **console** on the bottom left pane of RStudio, and then hit **Enter**. The function requires the `meta` package to work.

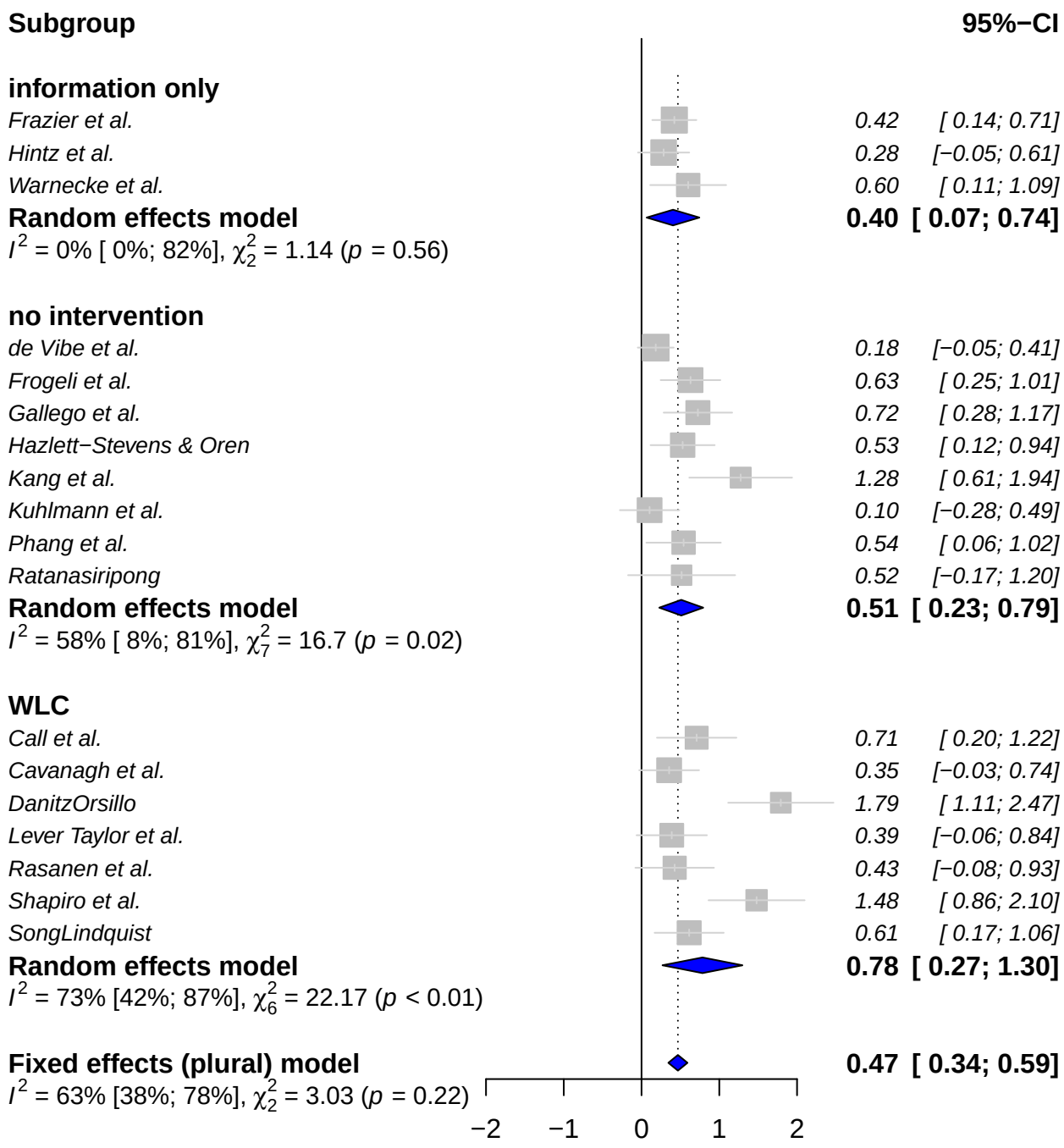
For the `subgroup.analysis.mixed.effects` function, the following parameters have to be set:

Code	Description
x	An object of class meta, generated by the metabin, metagen, metacont, metacor, metainc, or metaprop function.
subgroups	A character vector of the same length as the number of studies within the meta-analysis, with a unique code for the subgroup each study belongs to. Must have the same order as the studies in the meta object.
exclude	Single string or concatenated array of strings. The name(s) of the subgroup levels to be excluded from the subgroup analysis. If 'none' (default), all subgroup levels are used for the analysis.
plot	Logical. Should a forest plot for the mixed-effect subgroup analysis be generated? Calls the forest.meta function internally. TRUE by default.

In my `madata` dataset, which i used previously to generate my meta-analysis output `m.hksj`, i stored the subgroup variable `Control`. This variable specifies **which control group type was employed in which study**. There are **three subgroups**: `WLC` (waitlist control), no `intervention` and `information only`.

The function to do a subgroup analysis using the mixed-effects-model with these parameters looks like this.

```
subgroup.analysis.mixed.effects(x = m.hksj,
                                subgroups = madata$Control)
```




```
## Subgroup Results:
## --------------
##                   k        TE       seTE  LLCI  ULCI            p         Q
## information only  3 0.4048358 0.07875304 0.250 0.559 2.739011e-07  1.144426
## no intervention   8 0.5107720 0.11945084 0.277 0.745 1.902797e-05 16.704467
## WLC               7 0.7836032 0.20993337 0.372 1.195 1.894924e-04 22.167163
##                     I2 I2.lower I2.upper
## information only  0.00     0.00     0.82
## no intervention   0.58     0.08     0.81
## WLC               0.73     0.42     0.87
##
```

```
## Test for subgroup differences (mixed/fixed-effects (plural) model):
## --------------
##                        Q df         p
## Between groups 3.031562  2 0.2196365
##
## - Total number of studies included in subgroup analysis:  18
## - Tau estimator used for within-group pooling:  SJ
```

The results of the subgroup analysis are displayed under Subgroup Results. We also see that, while the **pooled effects of the subgroups differ quite substantially** (g = 0.41-0.78), this difference is **not statistically significant**. This can be seen under Test for subgroup differences (mixed/fixed-effects (plural) model) in the Between groups row. We can see that $Q = 3.03$ and $p = 0.2196$. This information can be reported in our meta-analysis paper.

7.3 Subgroup Analyses using the Random-Effects-Model

Now, let's assume i want to **know if intervention effects in my meta-analysis differ by region**. I use a **random-effects-model** and the selected coutries Argentina, Australia, China, and the Netherlands.

Again, i use the m.hksj meta-analysis output object. I can perform a random-effects-model for between-subgroup-differences using the update.meta function. For this function, we have to **set two parameters**.

Code	Description
byvar	Here, we specify the variable in which the subgroup of each study is stored
comb.random	Weather we want to use a random-effects-model for between-subgroup-differences. In this case, we have to set comb.random = TRUE

```
region.subgroup <- update.meta(m.hksj,
                               byvar=region,
                               comb.random = TRUE,
                               comb.fixed = FALSE)
region.subgroup
```

```
##                                      95%-CI %W(random)      region
## Call et al.            0.7091 [ 0.1979; 1.2203]        5.2 Netherlands
## Cavanagh et al.        0.3549 [-0.0300; 0.7397]        6.1 Netherlands
## DanitzOrsillo          1.7912 [ 1.1139; 2.4685]        4.2 Netherlands
## de Vibe et al.         0.1825 [-0.0484; 0.4133]        7.1         USA
## Frazier et al.         0.4219 [ 0.1380; 0.7057]        6.8         USA
## Frogeli et al.         0.6300 [ 0.2458; 1.0142]        6.1         USA
## Gallego et al.         0.7249 [ 0.2846; 1.1652]        5.7         USA
## Hazlett-Stevens & Oren 0.5287 [ 0.1162; 0.9412]        5.9   Argentina
## Hintz et al.           0.2840 [-0.0453; 0.6133]        6.5   Argentina
## Kang et al.            1.2751 [ 0.6142; 1.9360]        4.3   Argentina
## Kuhlmann et al.        0.1036 [-0.2781; 0.4853]        6.1   Australia
## Lever Taylor et al.    0.3884 [-0.0639; 0.8407]        5.6   Australia
```

```
## Phang et al.           0.5407 [ 0.0619; 1.0196]        5.4   Australia
## Rasanen et al.         0.4262 [-0.0794; 0.9317]        5.3       China
## Ratanasiripong         0.5154 [-0.1731; 1.2039]        4.1       China
## Shapiro et al.         1.4797 [ 0.8618; 2.0977]        4.5       China
## SongLindquist          0.6126 [ 0.1683; 1.0569]        5.7       China
## Warnecke et al.        0.6000 [ 0.1120; 1.0880]        5.4       China
##
## Number of studies combined: k = 18
##
##                                        95%-CI    t  p-value
## Random effects model 0.5935 [0.3891; 0.7979] 6.13 < 0.0001
##
## Quantifying heterogeneity:
## tau^2 = 0.1337; H = 1.64 [1.27; 2.11]; I^2 = 62.6% [37.9%; 77.5%]
##
## Quantifying residual heterogeneity:
## H = 1.68 [1.27; 2.24]; I^2 = 64.7% [37.6%; 80.0%]
##
## Test of heterogeneity:
##      Q d.f. p-value
##  45.50   17  0.0002
##
## Results for subgroups (random effects model):
##                         k                   95%-CI     Q  tau^2   I^2
## region = Netherlands    3 0.9142 [-0.9150; 2.7433] 13.06 0.4508 84.7%
## region = USA            4 0.4456 [ 0.0600; 0.8312]  6.87 0.0357 56.3%
## region = Argentina      3 0.6371 [-0.5837; 1.8580]  6.95 0.1826 71.2%
## region = Australia      3 0.3194 [-0.2427; 0.8815]  2.13 0.0204  6.1%
## region = China          5 0.7098 [ 0.2018; 1.2177]  7.81 0.1110 48.8%
##
## Test for subgroup differences (random effects model):
##                      Q d.f. p-value
## Between groups   4.52    4  0.3405
##
## Details on meta-analytical method:
## - Inverse variance method
## - Sidik-Jonkman estimator for tau^2
## - Hartung-Knapp adjustment for random effects model
```

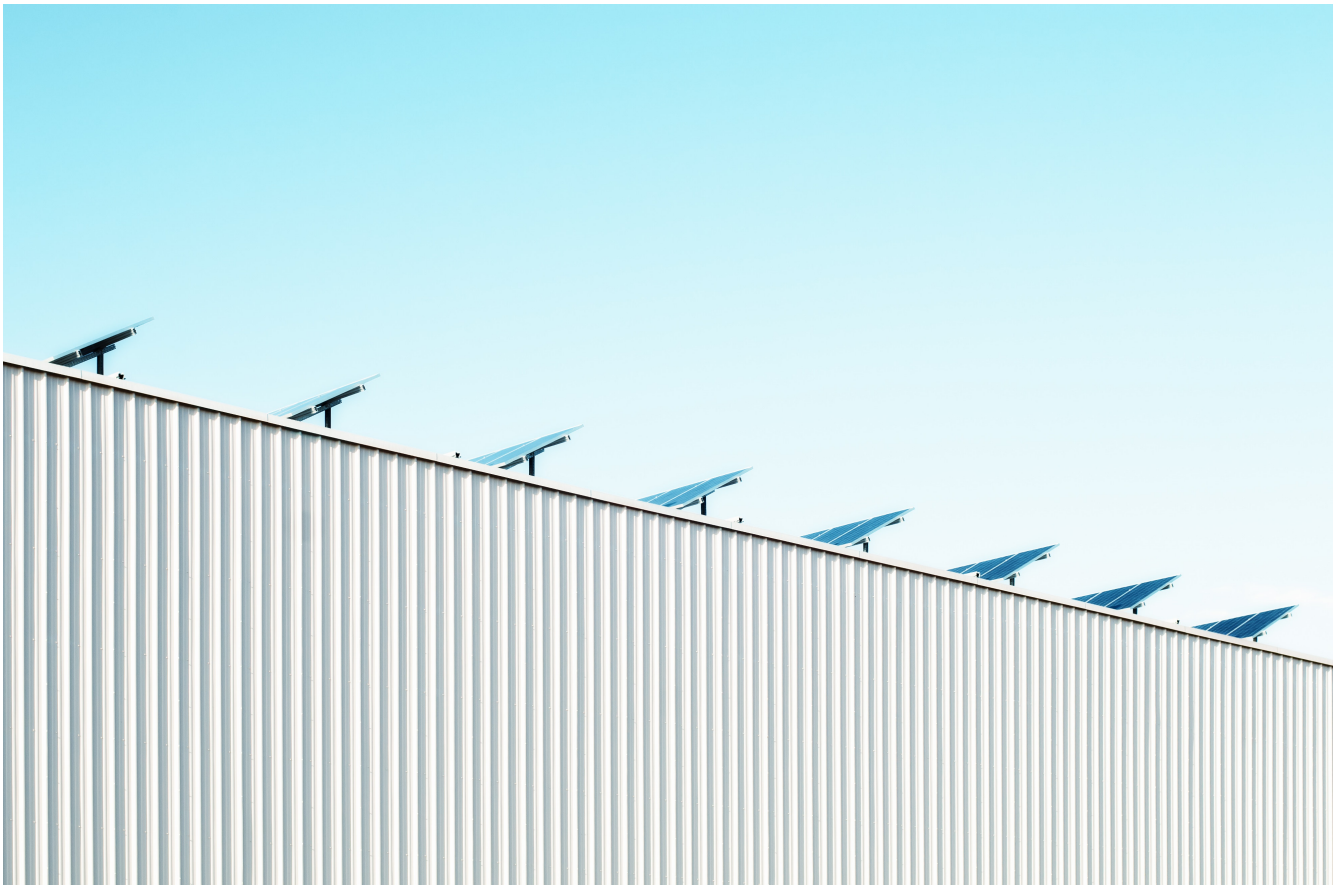
Here, we get the **pooled effect for each subgroup** (country). Under `Test for subgroup differences (random effects model)`, we can see the **test for subgroup differences using the random-effects-model**, which is **not significant** ($Q = 4.52$,$p = 0.3405$). This means that we did not find differences in the overall effect between different regions, represented by the country in which the study was conducted.

7.3.1 Using a fixed-effect-model for within-subgroup-pooling and a fixed-effects-model for between-subgroup-differences

To use a fixed-effect-model in combination with a fixed-effects-model, we can also use the `update.meta` function again. The procedure is the same as the one we described before, but we have to set `comb.random` as `FALSE` and `comb.fixed` as `TRUE`.

Chapter 8

Meta-Regression



Conceptually, **Meta-Regression** does not differ much from a **subgroup analysis**. In fact, subgroup analyses with more than two groups are nothing more than a meta-regression with categorial covariates. However, meta-regression does also allow us to use **continuous data** as covariates and check weather values of this variable are associated with effect size.

8.1 The idea behind meta-regression

You may have already performed regressions in regular data where participants or patients are the **unit of analysis**. In typical Meta-Analyses, we do not have the individual data for each participant available, but only the **aggregated effects**, which is why we have to perform meta-regressions with covariates at the **study level**. This also means that while we conduct analyses on participant samples much larger than usual in single studies, it is still very likely that **we don't have enough studies for a meta-regression to be sensible**. In Chapter 7, we told you that subgroup analyses make no sense when k<10. For **meta-regression**, Borenstein and colleages (Borenstein et al., 2011) recommend that **each covariate should at least contain ten studies**, although this should not be seen as clear rule. In a conventional regression, we want to estimate a parameter y using a covariate x_i with n regression coefficients β. A standard regression equation therefore looks like this:

$$y = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n$$

In a meta-regression, we want to estimate the **effect size** θ for different values of the covariate(s), so our regression looks like this:

$$\hat{\theta}_k = \theta + \beta_1 x_{1k} + ... + \beta_n x_{nk} + \epsilon_k + \zeta_k$$

You might have seen that when estimating the effect size θ_k of a study k in our regression model, there are two **extra terms in the equation**, ϵ_k and ζ_k. The same terms can also be found in the equation for the random-effects-model in Chapter 4.2. The two terms signify two types of **independent errors** which cause our regression prediction to be **imperfect**. The first one, ϵ_k, is the sampling error through which the effect size of the study deviates from its "true" effect. The second one, ζ_k, denotes that even the true effect size of the study is only sampled from **an overarching distribution of effect sizes** (see the Chapter on the Random-Effects-Model for more details). In a **fixed-effect-model**, we assume that all studies actually share the **same true effect size** and that the **between-study heterogeneity** $\tau^2 = 0$. In this case, we do not consider ζ_k in our equation, but only ϵ_k.

As the equation above includes **fixed effects** (the β coefficients) as well as **random effects** (ζ_k), the model used in meta-regression is often called **a mixed-effects-model**. Mathematically, this model is identical to the **mixed-effects-model** we described in Chapter 7 where we explained how **subgroup analyses** work. Indeed **subgroup analyses with more than two subgroups** are nothing else than a **meta-regression** with a **categorical predictor**. For meta-regression, these subgroups are then **dummy-coded**, e.g.

$$D_k = \begin{cases} 0 : ACT \\ 1 : CBT \end{cases}$$

$$\hat{\theta}_k = \theta + \beta x_k + D_k \gamma + \epsilon_k + \zeta_k$$

In this case, we assume the same **regression line**, which is simply "shifted" **up or down for the different subgroups** D_k.

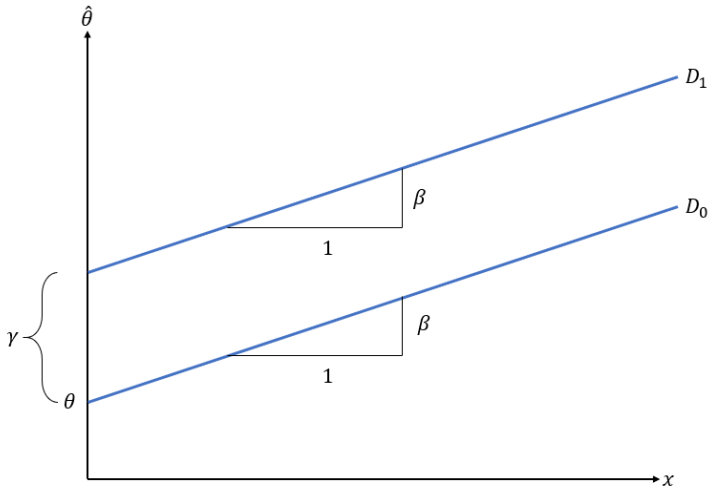


Figure 8.1: Visualisation of a Meta-Regression with dummy-coded categorial predictors

8.2 Assessing the fit of a regression model

To evaluate the **statistical significance of a predictor**, we a **t-test** of its β-weight is performed.

$$t = \frac{\beta}{SE_\beta}$$

Which provides a p-value telling us if a variable significantly predicts effect size differences in our regression model. If we fit a regression model, our aim is to find a model **which explains as much as possible of the current variability in effect sizes** we find in our data. In conventional regression, R^2 is commonly used to quantify the **goodness of fit** of our model in percent (0-100%). As this measure is commonly used, and many researchers know how to to interpret it, we can also calculate a R^2 analog for meta-regression using this formula:

$$R_2 = \frac{\hat{\tau}^2_{REM} - \hat{\tau}^2_{MEM}}{\hat{\tau}^2_{REM}}$$

Where $\hat{\tau}^2_{REM}$ is the estimated total heterogenetiy based on the random-effects-model and $\hat{\tau}^2_{REM}$ the total heterogeneity of our mixed-effects regression model.

8.3 Calculating Meta-Regressions in R

Meta-regressions can be conducted in R using the `metareg` function in `meta`. To show the similarity between subgroup analysis and `meta-regression` with categorical predictors, i'll first conduct a meta-regression with my variable "Control" as predictor again.

```
metareg(m.hksj,Control)
```

```
##
## Mixed-Effects Model (k = 18; tau^2 estimator: SJ)
##
## tau^2 (estimated amount of residual heterogeneity):     0.1343 (SE = 0.0536)
## tau (square root of estimated tau^2 value):             0.3665
## I^2 (residual heterogeneity / unaccounted variability): 73.92%
## H^2 (unaccounted variability / sampling variability):   3.84
## R^2 (amount of heterogeneity accounted for):            0.00%
##
## Test for Residual Heterogeneity:
## QE(df = 15) = 40.0161, p-val = 0.0005
##
## Test of Moderators (coefficients 2:3):
## F(df1 = 2, df2 = 15) = 0.9467, p-val = 0.4100
##
## Model Results:
##
##                         estimate      se    tval    pval    ci.lb   ci.ub
## intrcpt                   0.4252  0.2250  1.8899  0.0782  -0.0543  0.9048
## Controlno intervention    0.1003  0.2678  0.3744  0.7134  -0.4706  0.6711
## ControlWLC                0.3380  0.2765  1.2224  0.2404  -0.2514  0.9274
##
## intrcpt                 .
## Controlno intervention
## ControlWLC
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see in the output that the `metareg` function uses the values of "Control" (i.e, the three different types of control groups) as a **moderator**. It takes **"information only"** as a dummy-coded *reference group*, and **"no intervention"** and **"WLC"** as dummy-coded **predictors**. Under `Test of Moderators`, we can see that control groups are not significantly associated with effect size differences $F_{2,15} = 0.947$, $p = 0.41$. Our regression model does not explain any of the variability in our effect size data ($R^2 = 0\%$). Below `Model Results`, we can also see the β-values (`estimate`) of both predictors, and their significance level `pval`. As we can see, both predictors were not significant.

8.3.1 Continuous variables

Let's assume i want to check if the **publication year** is associated with effect size. I have stored the variable `pub_year`, containing the publication year of every study in my dataset, and conducted the meta-analysis with it. I stored my meta-analysis output in the `m.pubyear` output. **Now, i can use this predictor in a meta-regression.**

```
output.metareg<-metareg(m.pubyear,pub_year)
output.metareg
```

```
##
## Mixed-Effects Model (k = 18; tau^2 estimator: DL)
```

```
##
## tau^2 (estimated amount of residual heterogeneity):     0.0831 (SE = 0.0488)
## tau (square root of estimated tau^2 value):             0.2883
## I^2 (residual heterogeneity / unaccounted variability): 64.69%
## H^2 (unaccounted variability / sampling variability):   2.83
## R^2 (amount of heterogeneity accounted for):            0.00%
##
## Test for Residual Heterogeneity:
## QE(df = 16) = 45.3076, p-val = 0.0001
##
## Test of Moderators (coefficient 2):
## QM(df = 1) = 0.0054, p-val = 0.9412
##
## Model Results:
##
##           estimate       se     zval    pval     ci.lb    ci.ub
## intrcpt    -1.4580  27.6151  -0.0528  0.9579  -55.5825  52.6666
## pub_year    0.0010   0.0137   0.0737  0.9412   -0.0259   0.0280
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As you can see from the output, pub_year was now included as a **predictor**, but it is not significantly associated with the effect size ($p = 0.9412$).

8.4 Plotting regressions

To plot our meta-regression output, we can use the `bubble` function in `meta`. Here a few parameters we can specify for this function.

Parameter	Function
xlim	The x-axis limit of the plot. Must be specified as, e.g., 'xlim=c(0,1)'
ylim	The y-axis limit of the plot. Must be specified as, e.g., 'ylim=c(0,1)'
xlab	The label for the x axis
ylab	The label for the y axis
col	The color of the individual studies
lwd	The line width of the regression line
col.line	The color of the regression line
studlab	If the labels for each study should be printed within the plot (TRUE/FALSE)

```
bubble(output.metareg,
       xlab = "Publication Year",
       col.line = "blue",
       studlab = TRUE)
```

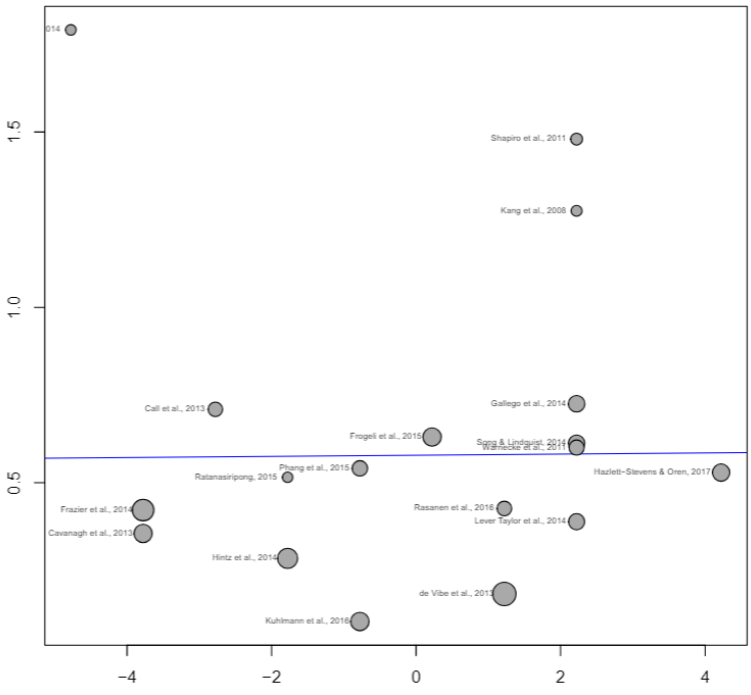


Figure 8.2: A finished bubble plot

8.5 Multiple Meta-Regression

In the beginning of Chapter 8, we already introduced the basic idea behind meta-regression. We showed that, in general, a meta-regression has the following form:

$$\hat{\theta}_k = \theta + \beta_1 x_{1k} + ... + \beta_n x_{nk} + \epsilon_k + \zeta_k$$

Where $\hat{\theta}_k$ is the estimate of our effect size of a study k and β are the regression coefficients. There are two parameters in this formula which cause our effect size estimate to deviate from the "true" overall effect size: ϵ_k, the sampling error, and ζ_k, which denotes that the true effect size of the study is only sampled from an **overarching distribution of effect sizes**. As we explained before, if $\zeta_k > 0$, the model we describe with this formula is a **random-effects meta-regression model**, also known as a *mixed-effects model*.

8.5.1 The idea behind multiple meta-regression

Previously, we only considered the scenario in which we use **one predictor** $\beta_1 x_1$ in our meta-regression (e.g., we want to check if the effect size a study reports depends on the year it was published). But imagine another scenario. Let's assume that we have made the experience in our previous research that the effect sizes we find in a research field for a certain study depends on how prestigious the journal in which the study was published is (e.g., higher effects are reported in journals with a high reputation). On the other hand, it also seems quite logical that journals with a good reputation publish studies of high quality, which may also be associated with higher effect sizes. So to check

if journal reputation is indeed associated with higher effect sizes, we have to make sure that this relationship is not **confounded** by the fact that prestigious journals publish studies of higher quality, which may be the true reason we find this relationship between journal reputation and effect size. This means we have to **control** for study quality (e.g., rated on a scale from 0 to 10) when we examine the relationship between journal prestige (e.g., operationalized as the impact factor of a journal) and effect size.

This, and many other possible scenarios can be dealt with using **multiple meta-regression**. In multiple meta-regression we use several predictors (variables) to predict (differences in) effect sizes. When we look back at the **general meta-regression** formula we defined before, we actually see that the formula already provides us with this feature through the $\beta_n x_{nk}$ part. Here, the parameter n denotes that we can include n more predictors/variables into our meta-regression, making it a multiple meta-regression. While, statistically speaking, the n parameter gives us the possibility to include as many predictors into our multiple meta-regression model as we wish to, things are a little more tricky in reality. In the following, we will discuss a few important pitfalls in multiple meta-regression and how we can build multiple meta-regression models which are robust and trustworthy. But first, let's cover another important feature of multiple meta-regression: **interactions**.

8.5.2 Interactions

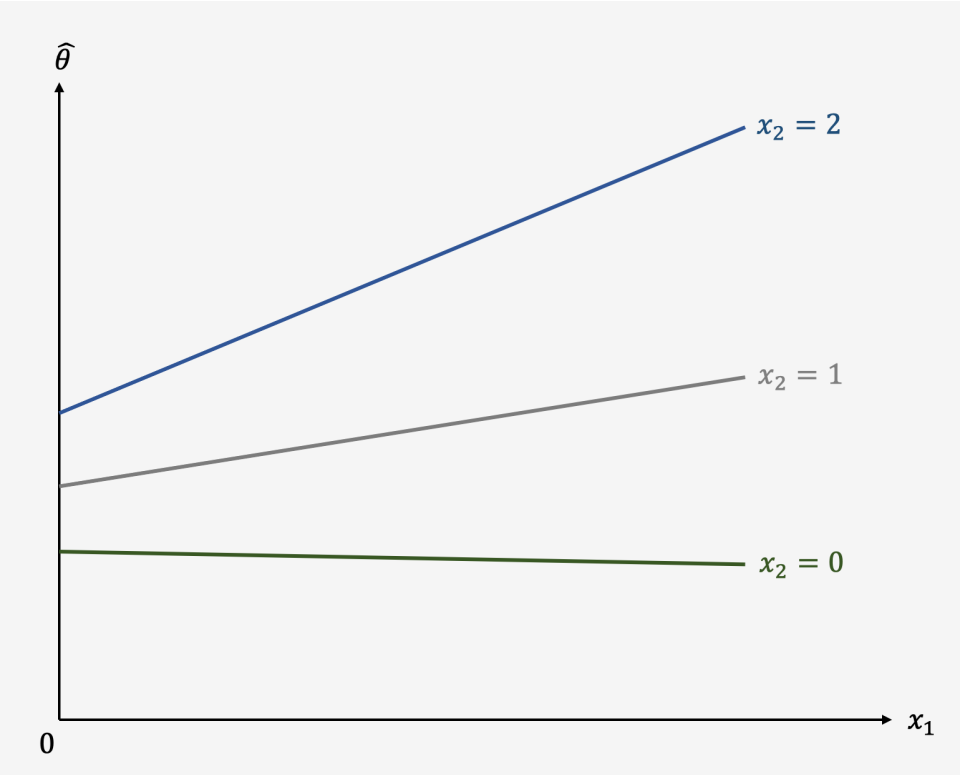
So far, in our multiple meta-regression model, we only considered the case where we have multiple predictor variables $x_1, x_2, ...x_n$, and along with their predictor estimates β_n, **add them together** to calculate our estimate of the true effect size $\hat{\theta}_k$ for each study k. In multiple meta-regression models, however, we can not only model such **additive relationships**. We can also model so-called **interactions**. Interactions mean that the **relationship** between one **predictor variable** (e.g., x_1) and the **estimated effect size** is different for different values of another predictor variable (e.g. x_2). Imagine a scenario where we want to model two predictors and their relationship to the effect size: the **publication year** (x_1) of a study and the **quality** (x_2) of a study, which we rate like this:

- 0: bad
- 1: moderate
- 2: good

As we described before, we can now imagine a meta-regression model in which we combine these two predictors x_1 and x_2 and assume an additive relationship. We can do this by simply adding them:

$$\hat{\theta}_k = \theta + \beta_1 x_{1k} + \beta_2 x_{2k} + \epsilon_k + \zeta_k$$

Let's assume that, overall, higher publication year (x_1) is associated with higher effect sizes (i.e., reported effect sizes have risen over the years). We could now ask ourselves if this positive relationship **varies** depending on the quality of the studies (x_2). For example, maybe this rise in effect sizes was strongest for high-quality studies, while effect sizes stayed mostly the same over the years for studies of lower quality? We can visualize our assumed relationship between effect size ($\hat{\theta}_k$), publication year (x_1) and study quality (x_2) the following way:



To examine such questions, we can add an **interaction term** to our meta-regression model. This interaction term lets predictions of x_1 vary for different values of x_2 (and vice versa). We can denote this additional interactional relationship in our model by introducing a third predictor, β_3, which captures this interaction $x_{1k}x_{2k}$ we want to test in our model:

$$\hat{\theta}_k = \theta + \beta_1 x_{1k} + \beta_2 x_{2k} + \beta_3 x_{1k} x_{2k} + \epsilon_k + \zeta_k$$

8.5.3 Common pitfalls of multiple meta-regression models

As we have mentioned before, multiple meta-regression, while **very useful when applied properly**, comes with **certain caveats** we have to know and consider when fitting a model. Indeed, some argue that (multiple) meta-regression is **often improperly used and interpreted in practice**, leading to a low validity of many meta-regression models (Higgins and Thompson, 2004). Thus, there are some points we have to keep in mind when fitting multiple meta-regression models, which we will describe in the following.

8.5.3.1 Overfitting: seeing a signal when there is none

To better understand the risks of (multiple) meta-regression models, we have to understand the concept of **overfitting**. Overfitting occurs when we build a statistical model which fits the data **too closely**. In essence, this means that we build a statistical model which can predict the data at hand very well, but performs bad at predicting future data it has never seen before. This happens if our model assumes that some variation in our data **stems from a true "signal" in our data, when in fact we only model random noise** (Iniesta et al., 2016). As a result, our statistical model produces **false positive results**: it sees relationships where there are none.

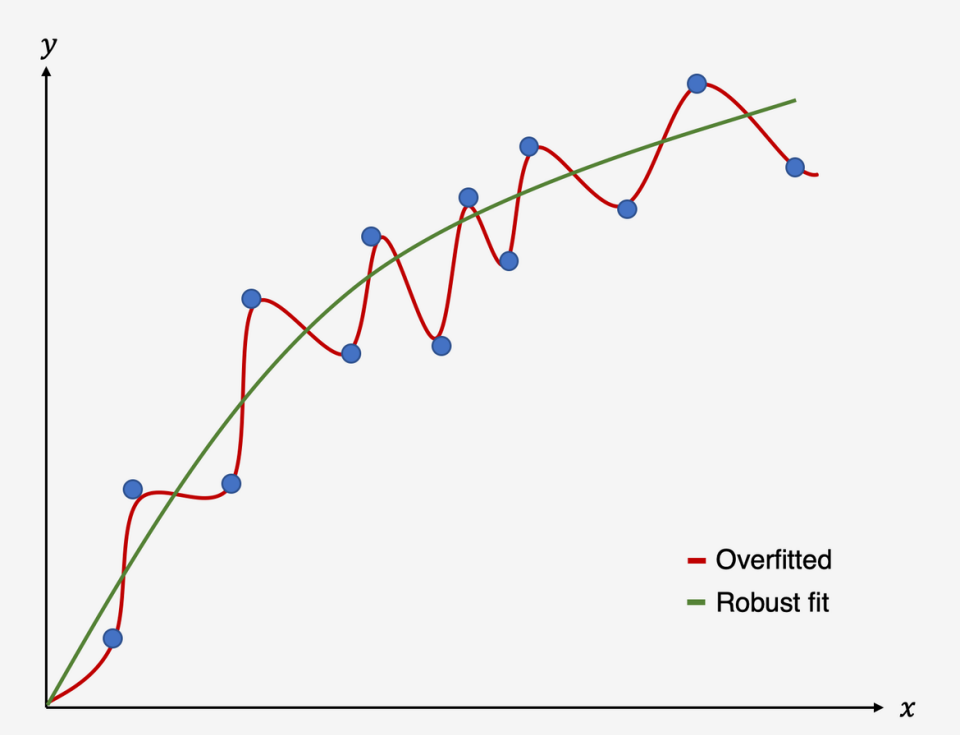


Figure 8.3: Illustration of an overfitted model vs. model with a good fit

Regression methods, which usually utilize minimization or maximization procedures such as Ordinary Least Squares or Maximum Likelihood estimatation, can be prone to overfitting (Gigerenzer, 2004, 2008). Unfortunately, the risk of building a **non-robust model, which produces false-positive results**, is **even higher** once we go from conventional regression to **meta-regression** (Higgins and Thompson, 2004). There are several reasons for this:

1. In Meta-Regression, our **sample of data is mostly small**, as we can only use the synthesized data of all analyzed studies k.
2. As our meta-analysis aims to be a **comprehensive overview of all evidence**, we have no **additional data** on which we can "test" how well our regression model can predict high or low effect sizes.
3. In meta-regressions, we have to deal with the potential presence of effect size heterogeneity (see Chapter 6). Imagine a case in which we have to studies with different effect sizes and non-overlapping confidence intervals. Every variable which has different values for the different studies might be a potential explanation for effect size difference we find, while it seems straightforward that **most of such explanations are spurious** (Higgins and Thompson, 2004).
4. Meta-regression as such, and multiple meta-regression in particular, make it very easy to **"play around" with predictors**. We can test numerous meta-regression models, include many more predictors or remove them in an attempt to explain the heterogeneity in our data. Such an approach is of course tempting, and often found in practice, because we, as meta-analysts, want to find a significant model which explains why effect sizes differ (Higgins et al., 2002). However, such behavior has been shown to massively **increase the risk of spurious findings** (Higgins and Thompson, 2004), because we can change parts of our model indefinitely until we find a significant model, which is then very likely to be overfitted (i.e., it mostly models statistical noise).

Some guidelines have been proposed to avoid an excessive false positive rate when building meta-regression models:

- Minimize the number of investigated predictors. In multiple meta-regression, this translates to the concept of **parsimony**, or simplicity: when evaluating the fit of a meta-regression model, we prefer models which achieve

a good fit with less predictors. Estimators such as the *Akaike* and *Bayesian information criterion* can help with such decisions. We will get to this topic again below.
- Predictor selection should be based on **predefined scientific or theoretical questions** we want to answer in our meta-regression.
- When the number of studies is low (which is very likely to be the case), and we want to compute the significance of a predictor, we should use the Knapp-Hartung adjustment to obtain more reliable estimates (Higgins et al., 2002).
- We can use **permutation** to assess the robustness of our model in resampled data. We will describe the details of this method later.

8.5.3.2 Multicollinearity

Multicollinearity means that one or more predictor in our regression model can be (linearly) **predicted from another predictor** in our model with relatively high accuracy (Mansfield and Helms, 1982). This basically means that we have two or more predictors in our model which are **highly correlated**. Most of the dangers of multicollinearity are associated with the problem of **overfitting** which we described above. High collinearity can cause our predictor coefficient estimates b to behave erratically, and change considerably with minor changes in our data. It also limits the size of the explained variance by the model, in our case the R^2 analog.

Multicollinearity in meta-regression is common (Berlin and Antman, 1992). Although multiple regression can handle lower degrees of collinearity, we should **check** and, if necessary, **control for very highly correlated predictors**. There is no consolidated yes-no-rule for the presence of multicollinearity. A crude, but often effective way is to check for very high correlations (i.e., $r \geq 0.8$) before fitting the model. Multicollinearity can then be reduced by either (1) removing one of the close-to-redundant predictors, or (2) trying to combine the predictors into one single predictor.

8.5.4 Model building methods

When building a multiple regression model, there are **different approaches through which we can select and insert predictors** into our model. Here, we discuss the most important ones along with their strengths and weaknesses:

- **Forced entry.** In forced entry methods, all relevant predictors are forced into the regression model simultaneously. In most functions in R, this is the default setting. Although this is a generally recommended procedure (Field et al., 2012), keep in mind that all predictors to use in forced entry should still be based on a predefined, theory-led decision.
- **Hierarchical.** Hierchical multiple regression means including predictors into our regression model stepwise, but based on a clearly defined scientific rationale. First, only predictors which have been associated with effect size differences in previous research are included in the order of their importance. After this step, novel predictors can be added to explore if these variables explain heterogeneity which has not yet been captured by the known predictors.
- **Stepwise.** Stepwise methods mean that variables/predictors are added to the model one after another. At first glance, this sounds a lot like hierarchical regression, but there is a crucial difference: stepwise regression methods select predictors based on a *statistical* criterion. In a procedure called **forward selection**, the variable explaining the largest amount of variability in the data is used as the first predictor. This processes is then repeated for the remaining variables, each time selecting the variable which explains most of the *remaining* (unexplained) variability in the data as the next predictor, which is then added to the model. Furthermore, there is a procedure called **backward selection**, in which all variables are used as predictors in the model first,

and then removed successively based on a predefined statistical criterion. There is an extensive literature discouraging the usage of stepwise methods (Whittingham et al., 2006; Chatfield, 1995), and if we recall the common pitfalls of multiple regression models we presented above, it becomes clear that these methods have high risk of producing spurious findings. Nevertheless, stepwise methods are still frequently used in practice, making it important to know that these procedures exist. If we use stepwise methods ourselves, however, it is advised to primarily do so in an exploratory fashion and keep the limitations of this procedure in mind.

- **Multimodel inference.** Multi-model methods differ from stepwise methods as they do not try to successively build the "best" one model explaining most of the variance. Instead, in this procedure, *all* possible combinations of a predifined selection of predictors are modeled, and evaluated using a criterion such as Akaike's Information Criterion, which rewards simpler models. This enables a full eximination of all possible models, and how they perform. A common finding using this procedure is that there are many different kinds of predictor combinations within a model which lead to a good fit. In multimodel inference, the estimated coefficients of predictors can then be synthesized across all possible models to infer how important certain predictors are overall.

8.5.5 Using `metafor` to compute Multiple Meta-Regressions

After all this input, it is now **time to fit our first multiple regression models**. To do this, we will not use the `meta` package we have used so far. Instead, we will use the `metafor` package (Viechtbauer et al., 2010). This package provides a vast array of advanced functionalities for meta-analysis, along with great documentation. We will also use this package in a later chapter where we focus on **Multilevel Meta-Analysis**, so it is worth getting accustomed to the inner workings of this package. So, to begin, let's make sure we have `metafor` installed and loaded in our library.

```
library(metafor)
```

For our multiple meta-regression examples, I will use my `mvreg.data` dataset. This is a "toy" dataset, which we simulated for illustrative purposes. **Let's have a look at the structure of my data:**

```
## 'data.frame':    36 obs. of  6 variables:
##  $ yi        : num  0.0944 0.0998 0.1693 0.1751 0.273 ...
##  $ sei       : num  0.196 0.192 0.119 0.116 0.165 ...
##  $ reputation: num  -11 0 -11 4 -10 -9 -8 -8 -8 0 ...
##  $ quality   : num  6 9 5 9 2 10 6 3 10 3 ...
##  $ pubyear   : num  -0.855 -0.753 -0.66 -0.563 -0.431 ...
##  $ continent : chr  "North America" "Europe" "North America" "Europe" ...
```

We see that there are 6 variables in our dataset. The `yi` and `sei` columns store the **effect size** and **standard error** of a particular study. Thus, these columns correspond to the TE and seTE columns we used before. We have named these variables this way because this is the standard notation that `metafor` uses: `yi` corresponds to the effect size y_i we want to predict in our meta-regression, while `sei` is SE_i, the standard error. To designate the variance of an effect size, `metafor` uses `vi`, or v_i in mathematical notation, which we do not need here because `yi` and `sei` contain all the information we need.

The other four variables we have in our dataset are potential predictors for our meta-regression. We want to check if `reputation`, the (mean-centered) impact score of the journal the study was published in, `quality`, the quality of the study rated from 0 to 10, `pubyear`, the (standardized) publication year, and `continent`, the continent in which the study was performed, are associated with different effect sizes.

For `continent`, note that we store information as a predictor with 2 labels: `Europe` and `North America`, meaning

that this predictor is a **dummy variable**. Always remember that such dummy variables have to be converted from a chr to a factor vector before we can proceed.

```
mvreg.data$continent = factor(mvreg.data$continent)
```

8.5.5.1 Checking for multicollinearity

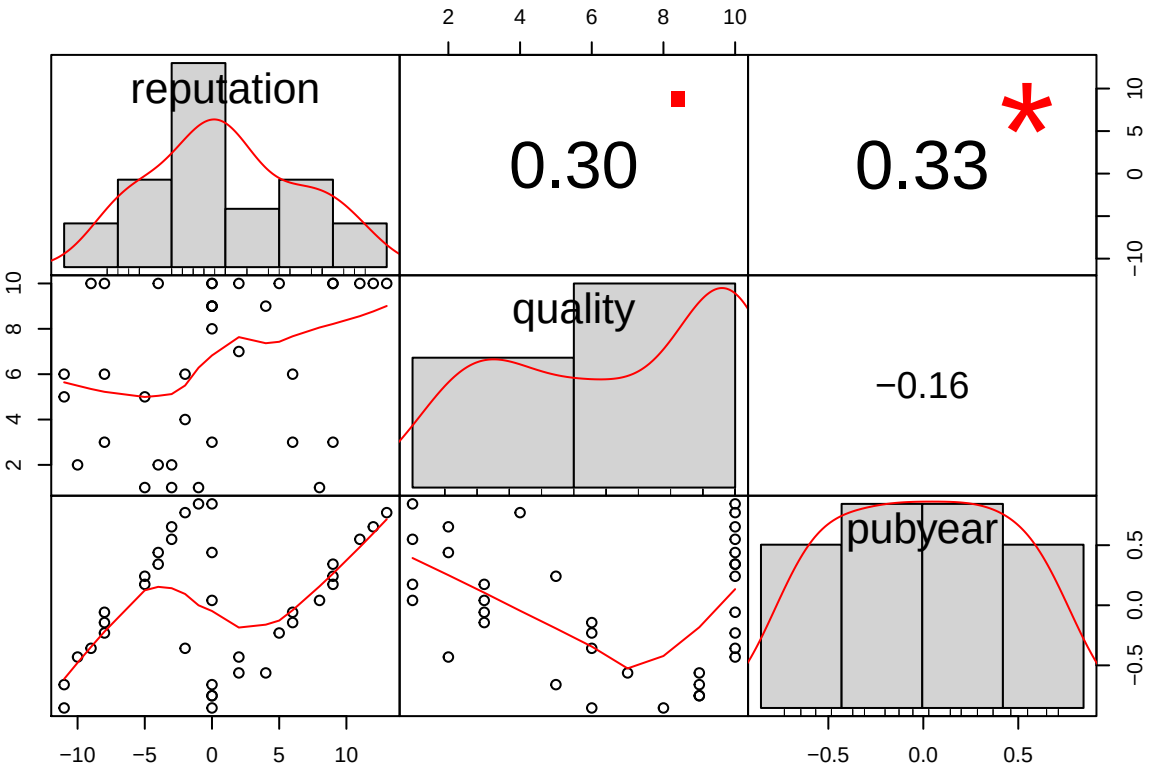
As we mentioned before, we have to check for multicollinearity of our predictors to make sure that our meta-regression model coefficient estimates are robust. A quick way to check for high intercorrelation is to calculate a **intercorrelation matrix** for all continuous variables with the following code:

```
cor(mvreg.data[,3:5])
```

```
##             reputation    quality    pubyear
## reputation  1.0000000  0.3015694  0.3346594
## quality     0.3015694  1.0000000 -0.1551123
## pubyear     0.3346594 -0.1551123  1.0000000
```

The `PerformanceAnalytics` package provides the `chart.Correlation` function which we can use to visualize the intercorrelations. Make sure to install the `PerformanceAnalytics` package first, and then use this code:

```
library(PerformanceAnalytics)
chart.Correlation(mvreg.data[,3:5])
```



We see that our variables are indeed correlated, but probably not to a degree that would warrant excluding one of them.

8.5.5.2 Fitting a meta-regression model without interaction terms

Now, let's fit our first meta-regression model, for now without assuming any interactions between predictors. Previously, we wanted to explore if a high **reputation** of the journal a study was published predicts higher effect sizes. Let's also assume we already know very well from previous literature that the **quality** of a study is also associated with differences in effects. We can now perform a hierarchical regression where we first include our known predictor `quality`, and then check if `reputation` explains heterogeneity beyond that while taking `quality` into account. To do this, we use the *rma* function in `metafor` to perform a **random-effects meta-regression**. This function has countless parameters, but we will only concentrate on the following (to see all parameters, type *?rma* into your console):

Parameter	Function
yi	The column of our dataset in which the effect sizes for each study are stored.
sei	The column of our dataset in which the standard error of the effect sizes for each study is stored.
data	The dataset with our meta-analysis data.
method	The estimator for tau-squared we want to use. For meta-regression models, it is advised to use the maximum likelihood ('ML') or restricted maximum likelihood ('REML') estimator.
mods	This parameter defines our meta-regression model. First, we specify our model with '~' (a tilde). Then, we add the predictors we want to include, separating them with '+' (e.g., 'variable1 + variable2'). Interactions between two variables are denoted by 'variable1*variable2'.
test	The test we want to apply for our regression coefficients. We can choose from 'z' (default) and 'knha' (Knapp-Hartung method).

First, let's perform a meta-regression using only `quality` as the predictor along with Knapp-Hartung adjustments. **We save the results as `model1`, and then inspect the output.**

```
model1 <- rma(yi=yi,
              sei=sei,
              data=mvreg.data,
              method = "ML",
              mods = ~ quality,
              test="knha")
model1
```

```
##
## Mixed-Effects Model (k = 36; tau^2 estimator: ML)
##
## tau^2 (estimated amount of residual heterogeneity):     0.0667 (SE = 0.0275)
## tau (square root of estimated tau^2 value):             0.2583
## I^2 (residual heterogeneity / unaccounted variability): 60.04%
## H^2 (unaccounted variability / sampling variability):   2.50
## R^2 (amount of heterogeneity accounted for):            7.37%
##
## Test for Residual Heterogeneity:
## QE(df = 34) = 88.6130, p-val < .0001
##
## Test of Moderators (coefficient 2):
## F(df1 = 1, df2 = 34) = 3.5330, p-val = 0.0688
```

```
##
## Model Results:
##
##          estimate      se    tval    pval    ci.lb   ci.ub
## intrcpt    0.3429  0.1354  2.5318  0.0161   0.0677  0.6181  *
## quality    0.0356  0.0189  1.8796  0.0688  -0.0029  0.0740  .
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the output, we can see the results for our predictor `quality` under `Model Results` and the values for our regression intercept (`intrcpt`). We see that the p value for our predictor is non-significant $p = 0.0688$, but only marginally so. Under `Test of Moderators (coefficient(s) 2)`, we can see the overall test results for our regression model ($F_{1,34} = 3.68, p = 0.0688$). Because we only included one predictor, the p-value reported there is identical to the one we saw before. In total, our model explains $R^2 = 2.23\%$ of the heterogeneity in our data, which we can see next to the `R^2 (amount of heterogeneity accounted for)` line in our output.

Now, let's check if we can improve our model by including `reputation` as predictor. We add `+ reputation` in our argument to `mods` and save the output to `model2`.

```
model2 <- rma(yi=yi,
              sei=sei,
              data=mvreg.data,
              method = "ML",
              mods = ~ quality + reputation,
              test="knha")
model2
```

```
##
## Mixed-Effects Model (k = 36; tau^2 estimator: ML)
##
## tau^2 (estimated amount of residual heterogeneity):     0.0238 (SE = 0.0161)
## tau (square root of estimated tau^2 value):             0.1543
## I^2 (residual heterogeneity / unaccounted variability): 34.62%
## H^2 (unaccounted variability / sampling variability):   1.53
## R^2 (amount of heterogeneity accounted for):            66.95%
##
## Test for Residual Heterogeneity:
## QE(df = 33) = 58.3042, p-val = 0.0042
##
## Test of Moderators (coefficients 2:3):
## F(df1 = 2, df2 = 33) = 12.2476, p-val = 0.0001
##
## Model Results:
##
##             estimate      se    tval    pval    ci.lb   ci.ub
## intrcpt       0.5005  0.1090  4.5927  <.0001   0.2788  0.7222  ***
## quality       0.0110  0.0151  0.7312  0.4698  -0.0197  0.0417
## reputation    0.0343  0.0075  4.5435  <.0001   0.0189  0.0496  ***
```

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

We now see that a new line appears under `Model Results` displaying the results for our `reputation` predictor. We see that a coefficient of $b = 0.0343$ was calculated for `reputation`. We also see that this predictor is highly significant ($p < 0.0001$) and positively associated with the effect sizes. We also see that the meta-regression model itself is significant ($F_{2,33} = 12.25, p < 0.0001$), explaining $R^2 = 66.95\%$ of the heterogeneity. This means that journal reputation is **associated with higher effect sizes**, even when controlling for study quality.

But is `model2` indeed better than our first model, `model1`? To assess this, we can use the *anova* function, feeding it with the two models we want to compare.

```
anova(model1, model2)
```

```
##
##         df     AIC     BIC    AICc   logLik     LRT   pval      QE  tau^2
## Full     4 19.4816 25.8157 20.7720  -5.7408                58.3042 0.0238
## Reduced  3 36.9808 41.7314 37.7308 -15.4904 19.4992 <.0001 88.6130 0.0667
##              R^2
## Full
## Reduced 64.3197%
```

This function performs a model test and provides us with several statistics to assess if `model2` has a better fit than `model1`. Here we compare our `full` model, `model2`, which includes both `quality` and `reputation`, with our Reduced model, which only uses `quality` as a predictor. The *anova* function performs a **Likelihood Ratio Test**, the results of which we can see in the LRT column. The test is highly significant ($\chi^2_1 = 19.50, p < 0.001$), which means that that our full model indeed provides a better fit. Another important statistic is reported in the `AICc` column. This provides us with the *Akaike's Information Criterion*, corrected for small samples. As we mentioned before, AICc penalizes complex models with more predictors to avoid overfitting. It is important to note that **lower values of AIC(c) mean that a model performs better**. Interestingly, in our output, we see that the full model ($AICc = 20.77$) has a better AIC value than our reduced model ($AICc = 37.73$), despite having more predictors. All of this suggests that our multiple regression **does indeed provide a good fit** to our data.

8.5.5.3 Modeling interaction terms

Let's say we want to model an **interaction hypothesis** with our additional predictors `pubyear` (publication year) and `continent`, where we assume that the relationship between publication year and effect size differs for Europe and North America. To model this in our `rma` function, we have to **connect our predictors** with * in the `mods` parameter. Because we do not want to compare the models directly using the *anova* function, we specify the τ^2 estimator to be `"REML"` (restricted maximum likelihood) this time:

```
interaction.model <- rma(yi=yi,
                         sei=sei,
                         data=mvreg.data,
                         method = "REML",
                         mods = ~ pubyear*continent,
                         test="knha")
interaction.model
```

```
##
## Mixed-Effects Model (k = 36; tau^2 estimator: REML)
##
## tau^2 (estimated amount of residual heterogeneity):     0 (SE = 0.0098)
## tau (square root of estimated tau^2 value):             0
## I^2 (residual heterogeneity / unaccounted variability): 0.00%
## H^2 (unaccounted variability / sampling variability):   1.00
## R^2 (amount of heterogeneity accounted for):            100.00%
##
## Test for Residual Heterogeneity:
## QE(df = 32) = 24.8408, p-val = 0.8124
##
## Test of Moderators (coefficients 2:4):
## F(df1 = 3, df2 = 32) = 28.7778, p-val < .0001
##
## Model Results:
##
##                                estimate      se    tval    pval    ci.lb
## intrcpt                          0.3892  0.0421  9.2472  <.0001   0.3035
## pubyear                          0.1683  0.0834  2.0184  0.0520  -0.0015
## continentNorth America           0.3986  0.0658  6.0539  <.0001   0.2645
## pubyear:continentNorth America   0.6323  0.1271  4.9754  <.0001   0.3734
##                                 ci.ub
## intrcpt                        0.4750  ***
## pubyear                        0.3380    .
## continentNorth America         0.5327  ***
## pubyear:continentNorth America 0.8911  ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The last line, `pubyear:continentNorth America`, is the coefficient for our interaction term. Note that `metafor` automatically **includes not only the interaction term**, but also both `pubyear` and `contintent` as **"normal" lower-order predictors** (as one should do). Also note that, as `continent` is a factor, `rma` detected that this is a **dummy predictor**, and used our category `Europe` as the $D = 0$ dummy against which the `North America` category is compared. We see that our interaction term `pubyear:continentNorth America` has a positive coefficient ($b = 0.6323$), and that it is highly significant ($p < 0.0001$), meaning that assumed interaction effect might in fact exist: there is an increase in effect sizes in recent years, but it is stronger for studies conducted in North America. We also see that model we fitted explains $R^2 = 100\%$ of our heterogeneity. This is because our data was simulated for illustrative purposes. In practice, you will hardly ever explain all of the heterogeneity in your data (in fact, one should rather be concerned if one finds such results in real-life data, as this might mean we have overfitted our model).

8.5.5.4 Testing the robustness of our model through a permutation test

8.5.5.4.1 The idea behind permutation

Permutation is a **mathematical operation** in which we take a **set** containing numbers or objects, and iteratively draw

elements from this set to put them in a sequential order. When we already have a ordered set of numbers, this equals a process in which we **rearrange**, or **shuffle**, the order of our data. As an example, imagine we have a set S containing **three numbers**: $S = \{1, 2, 3\}$. One possible permutation of this set is $(2, 1, 3)$; another is $(3, 2, 1)$. We see that the permuted results both contain **all three numbers from before**, but in a different order. Permutation can also be used to perform **permutation tests**, which is a specific form of **resampling methods**. Broadly speaking, resampling methods are used to validate the **robustness** of a statistical model by providing it with (slightly) different data sampled from the same data source or generative process (Good, 2013). This is a way to better **assess if the coeffients in our model indeed capture a true pattern underlying our data**, or if we overfitted our model, thereby falsely assuming patterns in our data when they are in fact statistical noise. Permutation tests do not require us to have a spare "test" dataset on which we can evaluate how our meta-regression actually performs in predicting effect sizes. For this reason, among others, permutation tests have been **recommended** to assess the robustness of our meta-regression models (Higgins and Thompson, 2004). We will not go too much into the **details of how a permutation test is performed for meta-regression** models, but the most important part is that we **re-calculate** the **p-values** of our overall meta-regression model and its coefficients based on the test statitics obtained across all possible, or many randomly selected, permutations of our original dataset (or, to be more precise, our meta-regression model matrix X_{ij}). The crucial indicator here is **how often the test statistic we obtain from in our permuted data is equal or greater than our original test statistic**. For example, if our test statistic is greater or equal to the original test statistic in 50 of 1000 permuted datasets, we get a p-value of $p = 0.05$ (Viechtbauer et al., 2015).

To **perform a permutation test** on our meta-regression model, we can use `metafors` in-built `permutest` function. As an example, i will now recalculate the results of my `model2` model we fitted above. We only have to supply the `permutest` function with the *rma* object, but **be aware that the permutation test can be computationally intensive, especially for large datasets, so the function might need some time to run**.

```
permutest(model2)
```

```
##
## Test of Moderators (coefficients 2:3):
## F(df1 = 2, df2 = 33) = 12.2476, p-val* = 0.0010
##
## Model Results:
##
##             estimate      se    tval   pval*    ci.lb   ci.ub
## intrcpt       0.5005  0.1090  4.5927  0.1810   0.2788  0.7222
## quality       0.0110  0.0151  0.7312  0.4290  -0.0197  0.0417
## reputation    0.0343  0.0075  4.5435  0.0010   0.0189  0.0496  ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We again see our **familiar output** including the **results for all predictors**. Looking at the `pval*` column, we see that our p-value for the `reputation` predictor has decreased from $p < 0.0001$ to $p^* = 0.001$. The p-value of our overall model has decreased from $p = 0.0001$ to $p^* = 0.001$. However, as both of these p-values are still highly significant, this indicates that our model might indeed capture a real pattern underlying our data. It has been **recommended** to always use this **permutation test on our meta-regression model before we report a meta-regression model** to be significant in our research (Higgins and Thompson, 2004).

8.5.5.4.2 Permutation tests in small datasets

Please note that when the **number of studies** k included in our model is **small**, conventionally used tresholds for statistical significance (i.e., p < 0.05) **cannot be reached**. For meta-regression models, a permutation test using `permutest` will only be able to reach statistical significance if $k \geq 5$. More details on the `permutest` function can be found here.

8.5.5.5 Multimodel Inference

Previously, we already described that one can also try to **model all possible predictor combinations** in a procedure called **multimodel inference** to examine which possible predictor combinations provide the best fit, and which predictors are the most important ones overall. To perform multimodel inference, you can use the `multimodel.inference` function we prepared for you. This function is part of the `dmetar` package. If you have the package installed already, you have to load it into your library first.

```
library(dmetar)
```

If you don't want to use the `dmetar` package, you can find the source code for this function here. In this case, *R* doesn't know this function yet, so we have to let *R* learn it by **copying and pasting** the code **in its entirety** into the **console** on the bottom left pane of RStudio, and then hit **Enter**. The function requires the `MuMIn`, `ggplot2` and `metafor` package to work.

For this function, the following parameters need to be specified:

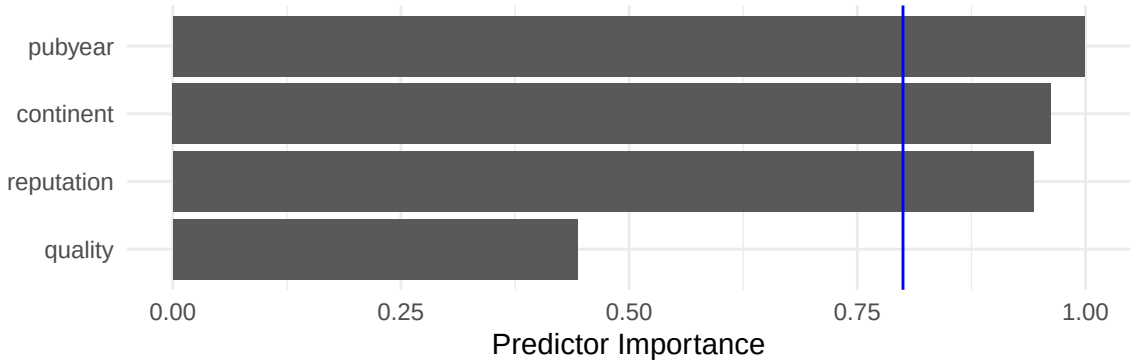
Parameter	Function
TE	The precalculated effect size for each study. Must be supplied as the name of the effect size column in the dataset (in quotation marks; e.g. TE = 'effectsize').
seTE	The precalculated standard error for each study. Must be supplied as the name of the standard error column in the dataset (in quotation marks; e.g. seTE = 'se').
data	A data.frame containing columns for the effect size, standard error and meta-regression predictors of each study/effect.
predictors	A concentenated array of characters specifying the predictors to be used for multimodel inference. Names of the predictors must be identical to the names of the column names of the data.frame supplied to data.
method	Meta-analysis model to use for pooling effect sizes. Use 'FE' for the fixed-effect model. Different random-effect models are available: 'DL', 'HE', 'SJ', 'ML', 'REML', 'EB', 'HS', 'GENQ'. If 'FE' is used, the test argument is automatically set to 'z', as the Knapp-Hartung method is not meant to be used with fixed-effect models. Default is 'REML', and it is strongly advised to remain with this option to use a standard (mixed-effects) meta-regression model.
test	Method to use to compute test statistics and confidence intervals. Default is 'knha' which uses the Knapp-Hartung (Knapp & Hartung, 2003) adjustment method. 'Conventional' Wald-type tests and CIs are calculated by setting this argument to 'z'. When method='FE', this argument is set to 'z' automatically as the Knapp-Hartung method was not meant to be used with fixed-effect models.
eval.criterion	Evaluation criterion to sort the multiple models by. Can be either 'AICc' (default; corrected Akaike's Information Criterion), 'AIC' (Akaike's Information Criterion) or 'BIC' (Bayesian Information Criterion).
interaction	If set to FALSE (default), no interactions between predictors are considered. Setting this parameter to TRUE means that all interactions are modeled.

Now, let's perform multimodelling for all possible predictor combinations using all predictors in the `mvreg.data` dataset, excluding all interaction terms. Be aware that this can take some time, especially if the number of predictors is large.

```
multimodel.inference(TE = "yi",
                     seTE = "sei",
                     data = mvreg.data,
                     predictors = c("pubyear", "quality", "reputation", "continent"),
                     interaction = FALSE)
```

```
##
  |
  |                                                                      |   0%
  |
  |====                                                                  |   6%
  |
  |========                                                              |  12%
  |
  |============                                                          |  19%
  |
  |================                                                      |  25%
  |
  |====================                                                  |  31%
  |
  |========================                                              |  38%
  |
  |============================                                          |  44%
  |
  |================================                                      |  50%
  |
  |====================================                                  |  56%
  |
  |========================================                              |  62%
  |
  |============================================                          |  69%
  |
  |================================================                      |  75%
  |
  |====================================================                  |  81%
  |
  |============================================================          |  88%
  |
  |================================================================      |  94%
##
##
## Multimodel Inference: Final Results
## -------------------------
##
```

```
##  - Number of fitted models: 16
##  - Full formula: ~ pubyear + quality + reputation + continent
##  - Coefficient significance test: knha
##  - Interactions modeled: no
##  - Evaluation criterion: AICc
##
##
## Best 5 Models
## --------------------------
##
##
## Global model call: metafor::rma(yi = TE, sei = seTE, mods = form, data = glm.data,
##     method = method, test = test)
## ---
## Model selection table
##    (Intrc) cntnn  pubyr   qulty   rpttn df logLik AICc delta weight
## 12       +     + 0.3533         0.02160  5  2.981  6.0  0.00  0.536
## 16       +     + 0.4028 0.02210 0.01754  6  4.071  6.8  0.72  0.375
## 8        +     + 0.4948 0.03574          5  0.646 10.7  4.67  0.052
## 11       +       0.2957         0.02725  4 -1.750 12.8  6.75  0.018
## 15       +       0.3547 0.02666 0.02296  5 -0.395 12.8  6.75  0.018
## Models ranked by AICc(x)
##
##
## Multimodel Inference Coefficients
## --------------------------
##
##
##              Estimate  Std. Error   z value  Pr(>|z|)
## intrcpt    0.38614661 0.106983583 3.6094006 0.0003069
## continent1 0.24743836 0.083113174 2.9771256 0.0029096
## pubyear    0.37816796 0.083045572 4.5537402 0.0000053
## reputation 0.01899347 0.007420427 2.5596198 0.0104787
## quality    0.01060060 0.014321158 0.7402055 0.4591753
##
##
## Predictor Importance
## --------------------------
##
##
##        model importance
## 1    pubyear  0.9988339
## 2  continent  0.9621839
## 3 reputation  0.9428750
## 4    quality  0.4432826
```



Let's go through the output one after another:

- `Multimodel Inference: Final Results`. This part of the output provides us with details about the fitted models. We see that the total number of $2^4 = 16$ possible models have been fitted. We also see that the function used the corrected AIC (`aicc`) to compare the models.
- `Best 5 Models`. Displayed here are the **five models with the lowest AICc**, sorted from low to high. Predictors are displayed as columns of the table, and models as rows. A number (weight) or + sign (for categorical predictors) indicates that a predictor/interaction term was used for the model, while empty cells indicate that the predictor was omitted in this model. We see that `TE ~ 1 + continent + pubyear + reputation` shows the best fit ($AICc = 6.0$). But we also see that the other models including other predictors which are presented here come very close to this value. Thus, it is hard to say which model really is the "best" model. Nevertheless, we see that all of the best five models contain the predictor `pubyear`, suggesting that this variable might be particularly important.
- `Multimodel Inference Coefficients`. Here, we can see the **coefficients of all predictors** we used for our models **aggregated** over all models in which they appear. We see that the coefficient `Estimate` is largest for `pubyear` ($b = 0.378$), which corroborates our finding from before. Approximate confidence intervals can be obtained by subtracting and adding the value stored in `Std.Error`, multiplied by 1.96, from/to `Estimate`.
- **Model-averaged importance of terms plot**. In the plot, we see the averaged importance of each predictor across all models displayed as a bar chart. We again see that `pubyear` is the most important predictor, followed by `reputation`, `continent`, and `quality`.

This example should make clear that Multimodel Inference can be a useful way to obtain a **comprehensive look on which predictors are more or less important** for predicting differences in effect sizes. Despite avoiding some of the problems of stepwise regression methods, it should be noted that this method should still be rather seen as **exploratory**, and may be used when we have no prior knowledge on how our predictors are related to effect sizes in the research field we meta-analyze. If you decide to build a meta-regression model using the information from multimodel inference, it is **essential to report that you have used this procedure before building the model**, because your model will then not be based on an apriori hypothesis, but on statistical properties.

The output of the `mreg.multimodel.inference` function displayed after running the function is **only a small part of all results returned by the function**. If we want to access all results of the function, we can save the results to an object (e.g. `results <- multimodel.inference(...)`), and then open them using the 'dollar' operator. The following results are returned, among other things:

- `all.models`: the AICc statistics for all fitted models, ordered from best to worst.
- `top5.models`: the top 5 models by AICc we already saw in the output of the function before.
- `multimodel.coef`: the averaged multimodel coefficient estimates we already saw before.
- `predictor.importance`: The predictor importance table we already saw before.
- `formula`: The full formula used to conduct the multimodel analysis.

Chapter 9

Publication Bias

In the last chapters, we have showed you how to **pool effects** in meta-analysis, choose the **right pooling model**, assess the **heterogeneity** of your effect estimate, and determine sources of heterogeneity through **outlier, influence, and subgroup analyses**. Nevertheless, even the most thoroughly conducted meta-analysis can only work with the **study material at hand**. An issue commonly discussed in research, however, is the **file-drawer** or **publication bias** problem, which states that a study with high effect sizes **is more likely to be published** than a study with a low effect size (Rothstein et al., 2006). Such **missing studies** with low effect sizes, it is assumed, thus never get published and therefore cannot be integrated into our meta-analysis. This leads to **publication bias**, as the pooled effect we estimate in our meta-analysis might be higher than the **true effect size** because we did not consider the missing studies with lower effects due to the simple fact that they were never published. Although this practice is gradually changing

(Nelson et al., 2018), weather a study is published or not heavily depends on the **statistical significance** ($p < 0.05$) of its results (Dickersin, 2005). For any sample size, a result is more likely to become **statistically significant** if its **effect size is high**. This is particularly true for **small studies**, for which very large effect sizes are needed to attain a statisitcally significant result. In the following chapter, we will describe the **idea behind statistical models for publication bias** in further depth. We termed these concepts and methods **small-study effect methods**, as it is small studies that they mostly focus on. These methods assume that publication bias is primarily driven by **effect size** and because researchers **immediately put every study in the file drawer once the results are not significant**.

Recently, it has been argued that these **assumptions may not be true**, and that publication bias is mostly caused by **significance levels** and **p-hacking** (Simonsohn et al., 2014a). An alternative method called **P-Curve** has therefore been suggested to examine **publication bias**. We will present this method in the last chapter of this section.

9.1 Which method should i use for my meta-analysis?

While recent research suggests that the conventional **small-study effect methods** may have substantial **limitations**, and that **p-Curve** may be able to estimate the true effect with less bias (Simonsohn et al., 2014a, 2015, 2014b), please note that both methods are based on different **theoretical assumptions** about the origin of publication bias. As we cannot ultimately decide which assumption is the **"true"** one in specific research fields, and, in practice **the true effect is unkown when doing meta-analysis**, we argue that you may use **both methods** and compare results as **sensitivity analyses** (Harrer et al., 2019).

P-curve was developed with **full-blown experimental psychological research in mind**, in which researchers often have **high degrees of "researcher freedom"** (Simmons et al., 2011) in deleting outliers and performing statistical test on their data.

We argue that this looks slightly different for **clinical psychology** and the medical field, where researchers conduct **randomized controlled trials** whith a clear **primary outcome**: the difference between the control and the intervention group after the treatment. While it is also true for **medicine and clinical psychology that statistical significance plays an important role**, the **effect size** of an intervention is often of greater interest, as **treatments are often compared in terms of their treatment effects** in this field. Furthermore, best practice for randomized controlled trials is to perform **intention-to-treat** analyses, in which all collected data in a trial has to be considered, giving researchers less space to "play around" with their data and perform p-hacking. While we certainly do not want to insinuate that **outcome research in clinical psychology** is free from p-hacking and bad data analysis practices, this should be seen as a **caveat** that the assumptions of the small-study effects methods may be more adequate for clinical psychology than other fields within psychology, especially when **the risk of bias for each study is also taken into account*. Facing this uncertainty, we think that conducting both analyses and reporting them in our research paper may be the most adequate approach until meta-scientific research gives us more certainty about which **assumption actually best reflects the field of clinical psychology**.

9.2 Small-study effect methods

The **small-study effect methods** we present here have been conventional for many years. Thus various methods to assess and control for publication bias have been developed, but we will only focus on the most important ones here.

9.2.1 The model behind small-study effects methods

According to Borenstein et al. (Borenstein et al., 2011). The model behind the most common small-study effects methods has these core **assumptions**:

1. Because they involve large commitment of ressources and time, **large studies are likely to get published**, weather the results are significant or not
2. Moderately sized studies are at **greater risk of missing**, but with a moderate sample size even moderately sized effects are likely to become significant, which means that only some studies will be missing
3. Small studies are **at greatest risk** for being non-significant, and thus being missing. Only small studies with a very large effect size become significant, and will be found in the published literature.

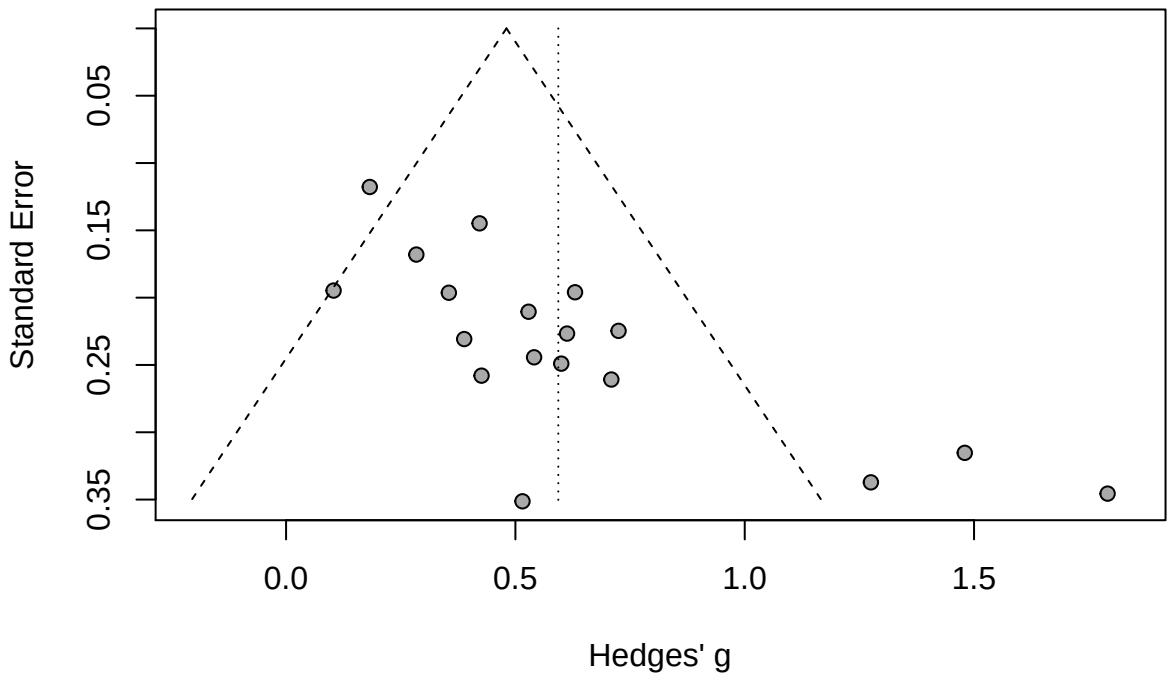
In accordance with these assumptions, the methods we present here particularly focus **on small studies with small effect sizes, and wheather they are missing**.

9.2.2 Funnel plots

The best way to visualize whether **small studies with small effect sizes are missing** is through **funnel plots**.

We can generate a funnel plot for our `m.hksj` meta-analysis output using the `funnel()` function in `meta`.

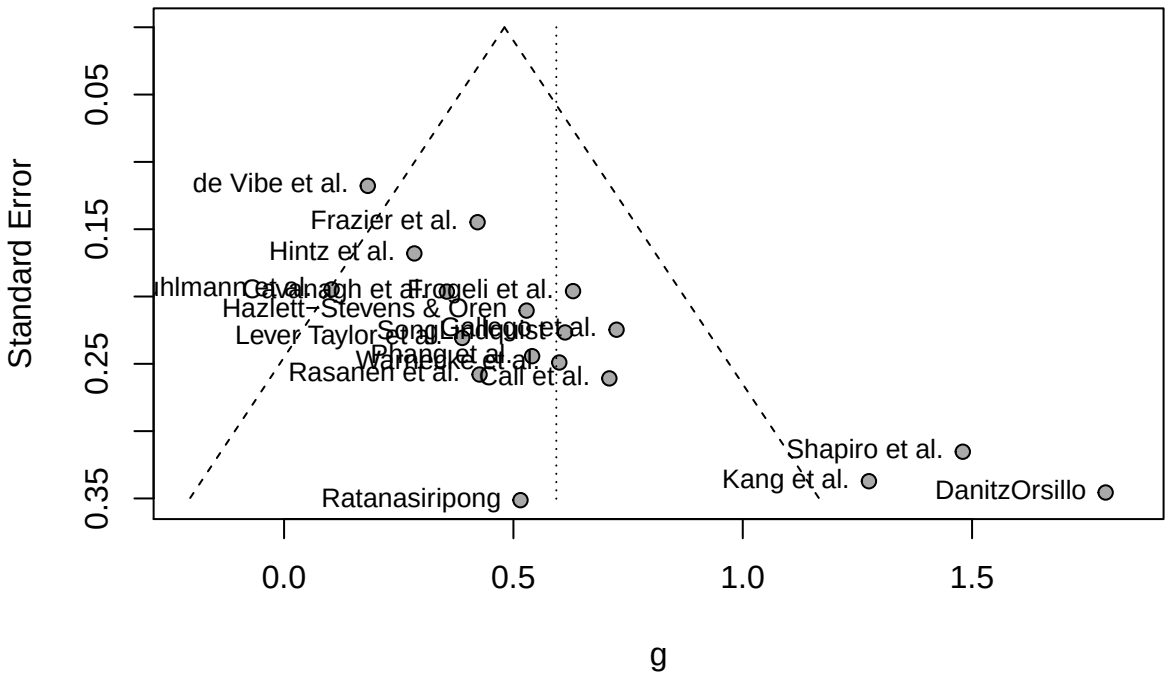
```
funnel(m.hksj,xlab = "Hedges' g")
```



The **funnel plot** basically consists of a **funnel** and two **axes**: the y-axis showing the **Standard Error** SE of each study, with larger studies (which thus have a smaller SE) plotted **on top of the y-axis**; and the x-axis showing the **effect size** of each study. Given our assumptions, and in the case when there is **no publication bias**, all studies would lie **symmetrically around our pooled effect size (the striped line)** within the form of the funnel. When **publication bias is present**, we would assume that the funnel would look asymmetrical, because only the small studies with a large effect size very published, **while small studies without a significant, large effect would be missing**. We see from the plot that in the case of our meta-anlysis `m.hksj`, the latter is probably true. We see that the plot is highly asymmetrical,

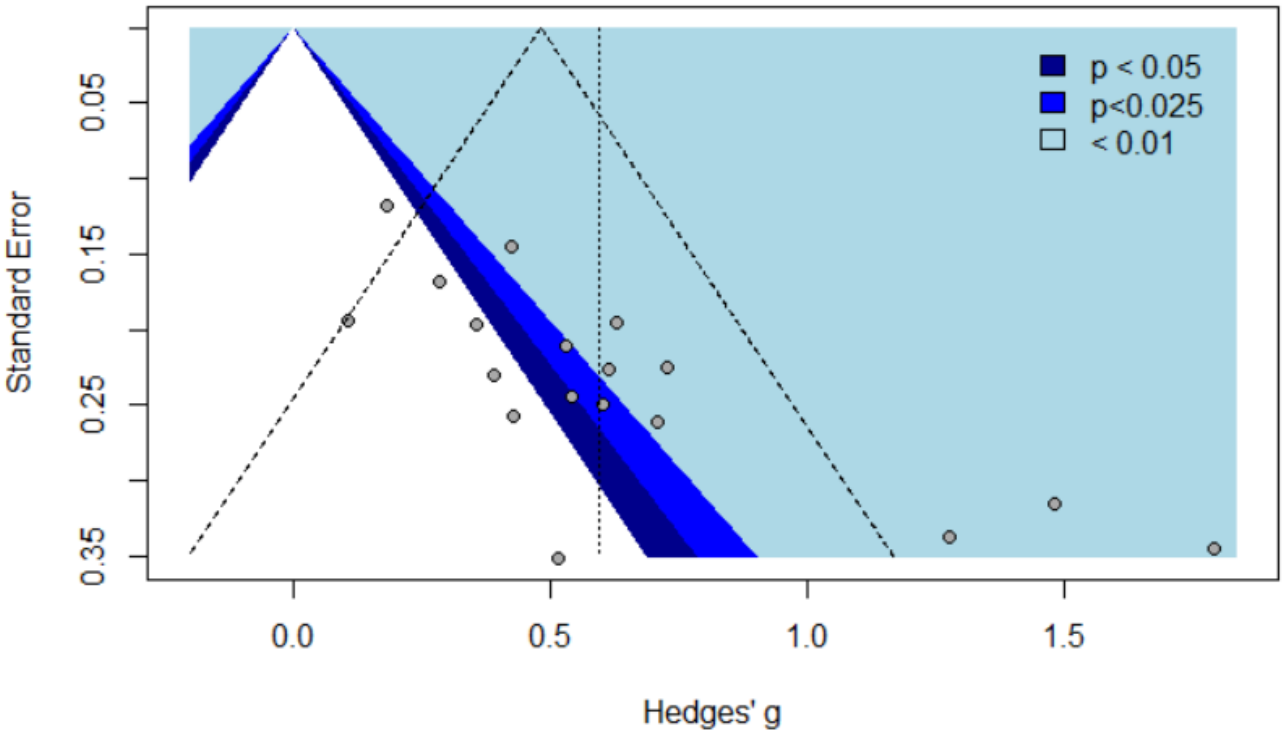
with exactly the small studies with low effect size missing in the bottom-left corner of our plot. We can also display the name of each study using the `studlab` parameter.

```
funnel(m.hksj,xlab = "g",studlab = TRUE)
```



Here, we see that asymmetry is primarily driven by **three studies with high effects, but a small study sample** in the bottom right corner. Interestingly, two of these studies are the ones we also detected in our outlier and influence analyses. An even better way to inspect the funnel plot is through **contour-enhanced funnel plots**, which help to distinguish publication bias from other forms of asymmetry (Peters et al., 2008). Contour-enhanced funnels include colors signifying the significance level into which the effects size of each study falls. We can plot such funnels using this code:

```
funnel(m.hksj, xlab="Hedges' g",
       contour = c(.95,.975,.99),
       col.contour=c("darkblue","blue","lightblue"))+
legend(1.4, 0, c("p < 0.05", "p<0.025", "< 0.01"),bty = "n",
       fill=c("darkblue","blue","lightblue"))
```



We can see in the plot that while **some studies have statistically significant effect sizes** (blue background), other do not (white background). We also see a trend that while the moderately sized studies are partly significant and non-significant, with slightly more significant studies, the asymmetry is much larger for small studies. This gives us a hint that publication bias might indeed be present in our analysis.

9.2.3 Testing for funnel plot asymmetry using Egger's test

Egger's test of the intercept (Egger et al., 1997) quantifies the funnel plot asymmetry and performs a statistical test. We have prepared a function called `eggers.test` for you, which allows you to perform Egger's test of the intercept in *R*. The function is a wrapper for the `metabias` function in `meta`. The `eggers.test function is part if the` [dmetar'](#dmetar) package. If you have the package installed already, you have to load it into your library first.

```
library(dmetar)
```

If you don't want to use the `dmetar` package, you can find the source code for this function here. In this case, *R* doesn't know this function yet, so we have to let *R* learn it by **copying and pasting** the code **in its entirety** into the **console** in the bottom left pane of RStudio, and then hit **Enter** . The function requires the `meta` package to work.

Now we can use the `eggers.test` function. We only have to specify our meta-analysis output `m.hksj` as the parameter x the function should use.

```
eggers.test(x = m.hksj)
```

```
##              Intercept ConfidenceInterval     t       p
## Egger's test     4.111        2.347-5.875 4.677 0.00025
```

The function returns the `intercept` along with its confidence interval. We can see that the `p-value` of Egger's test

is **significant** ($p < 0.05$), which means that there is substanital asymmetry in the Funnel plot. This asymmetry could have been caused by publication bias.

9.2.4 Duval & Tweedie's trim-and-fill procedure

Duval & Tweedie's trim-and-fill procedure (Duval and Tweedie, 2000) is also based the funnel plot and its symmetry/asymmetry. When **Egger's test is significant**, we can use this method to estimate what **the actaul effect size would be had the "missing" small studies been published**. The procedure **imputes** missing studies into the funnel plot until symmetry is reached again.

9.2.4.1 The trim-and-fill procedure includes the following five steps** (Schwarzer et al., 2015)

1. Estimating the number of studies in the outlying (right) part of the funnel plot
2. Removing (trimming) these effect sizes and pooling the results with the remaining effect sizes
3. This pooled effect is then taken as the center of all effect sizes
4. For each trimmed/removed study, a additional study is imputed, mirroring the effect of the study on the left side of the funnel plot
5. Pooling the results with the imputed studies and the trimmed studies included

The **trim-and-fill-procedure** can be performed using the `trimfill` function in `meta`, and specifying our meta-analysis output.

```
trimfill(m.hksj)
```

```
##                              SMD             95%-CI %W(random)
## Call et al.                0.7091 [ 0.1979;  1.2203]        3.8
## Cavanagh et al.            0.3549 [-0.0300;  0.7397]        4.1
## DanitzOrsillo              1.7912 [ 1.1139;  2.4685]        3.3
## de Vibe et al.             0.1825 [-0.0484;  0.4133]        4.4
## Frazier et al.             0.4219 [ 0.1380;  0.7057]        4.3
## Frogeli et al.             0.6300 [ 0.2458;  1.0142]        4.1
## Gallego et al.             0.7249 [ 0.2846;  1.1652]        4.0
## Hazlett-Stevens & Oren     0.5287 [ 0.1162;  0.9412]        4.0
## Hintz et al.               0.2840 [-0.0453;  0.6133]        4.2
## Kang et al.                1.2751 [ 0.6142;  1.9360]        3.4
## Kuhlmann et al.            0.1036 [-0.2781;  0.4853]        4.1
## Lever Taylor et al.        0.3884 [-0.0639;  0.8407]        3.9
## Phang et al.               0.5407 [ 0.0619;  1.0196]        3.9
## Rasanen et al.             0.4262 [-0.0794;  0.9317]        3.8
## Ratanasiripong             0.5154 [-0.1731;  1.2039]        3.3
## Shapiro et al.             1.4797 [ 0.8618;  2.0977]        3.5
## SongLindquist              0.6126 [ 0.1683;  1.0569]        4.0
## Warnecke et al.            0.6000 [ 0.1120;  1.0880]        3.9
## Filled: Warnecke et al.    0.0520 [-0.4360;  0.5401]        3.9
## Filled: SongLindquist      0.0395 [-0.4048;  0.4837]        4.0
## Filled: Frogeli et al.     0.0220 [-0.3621;  0.4062]        4.1
```

```
## Filled: Call et al.     -0.0571 [-0.5683;  0.4541]        3.8
## Filled: Gallego et al.  -0.0729 [-0.5132;  0.3675]        4.0
## Filled: Kang et al.     -0.6230 [-1.2839;  0.0379]        3.4
## Filled: Shapiro et al.  -0.8277 [-1.4456; -0.2098]        3.5
## Filled: DanitzOrsillo   -1.1391 [-1.8164; -0.4618]        3.3
##
## Number of studies combined: k = 26 (with 8 added studies)
##
##                          SMD            95%-CI    t p-value
## Random effects model 0.3431 [ 0.0994; 0.5868] 2.90  0.0077
## Prediction interval         [-0.8463; 1.5326]
##
## Quantifying heterogeneity:
## tau^2 = 0.3181; H = 2.05 [1.70; 2.47]; I^2 = 76.2% [65.4%; 83.7%]
##
## Test of heterogeneity:
##       Q d.f.  p-value
##  105.15   25 < 0.0001
##
## Details on meta-analytical method:
## - Inverse variance method
## - Sidik-Jonkman estimator for tau^2
## - Hartung-Knapp adjustment for random effects model
## - Trim-and-fill method to adjust for funnel plot asymmetry
```

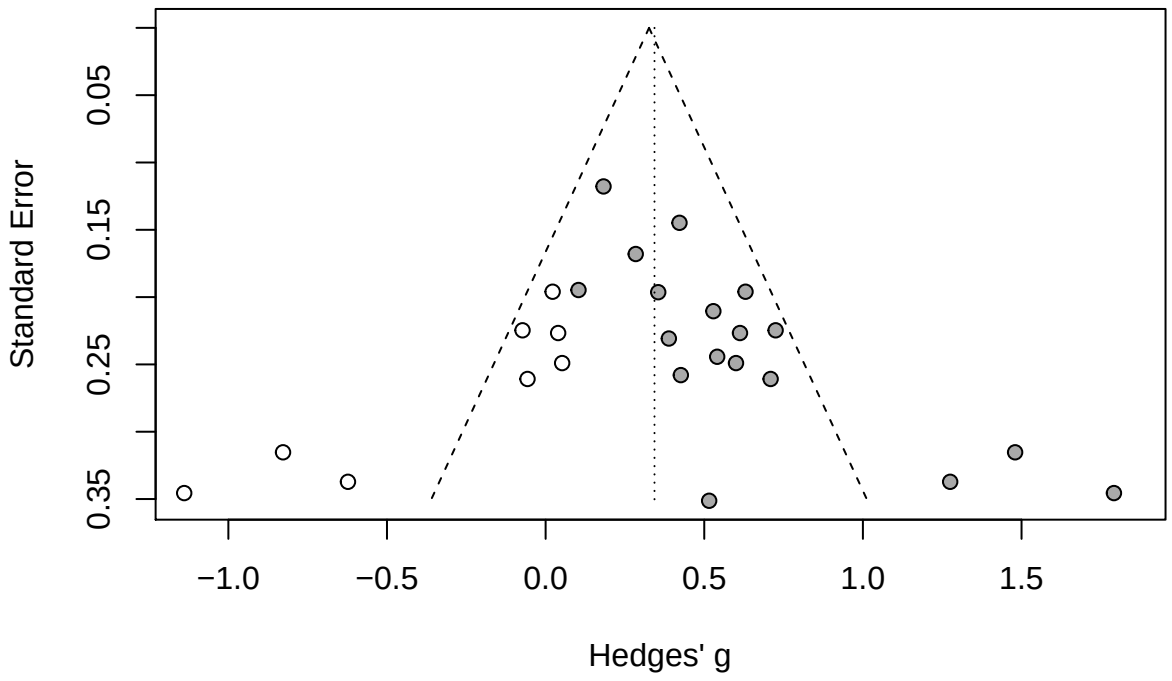
We see that the procedure identified and trimmed **eight studies** (`with 8 added studies`). The overall effect estimated by the procedure is $g = 0.34$. Let's compare this to our initial results.

```
m.hksj$TE.random
```

```
## [1] 0.593535
```

The initial pooled effect size was $g = 0.59$, which is substantially larger than the bias-corrected effect. In our case, if we assume that **publication bias** was a problem in the analyses, the **trim-and-fill procedure** lets us assume that our initial results were **overestimated** due to publication bias, and the "true" effect when controlling for selective publication might be $g = 0.34$ rather than $g = 0.59$. If we store the results of the `trimfill` function in an **object**, we can also create **funnel plots including the imputed studies**.

```
m.hksj.trimfill<-trimfill(m.hksj)
funnel(m.hksj.trimfill,xlab = "Hedges' g")
```



9.3 *P*-Curve

In the last chapter, we showed you how you can apply **Egger's test of the intercept**, **Duval & Tweedie's trim and fill procedure**, and inspect **Funnel plots** in *R*.

As we have mentioned before, recent research has shown that the assumptions of the small-effect study methods may be inaccurate in many cases. The **Duval & Tweedie trim-and-fill procedure** in particular has been shown to be prone to providing **inaccurate effect size estimates** (Simonsohn et al., 2014b).

P**-curve Analysis** has been proposed as an alternative way to assess publication bias and estimate the true effect behind our collected data. P-Curve assumes that publication bias is not primarily generated because researchers do not publish non-significant results, but because the "play" around with their data (e.g., selectively removing outliers, choosing different outcomes, controlling for different variables) until a non-significant finding becomes significant. This (bad) practice is called p**-hacking**, and has been shown to be very frequent among researchers (Head et al., 2015).

9.3.1 Performing a *p*-curve analysis

To conduct a p-curve analysis, you can use the `pcurve` function we prepared for you. This function is part if the `dmetar` package. If you have the package installed already, you have to load it into your library first.

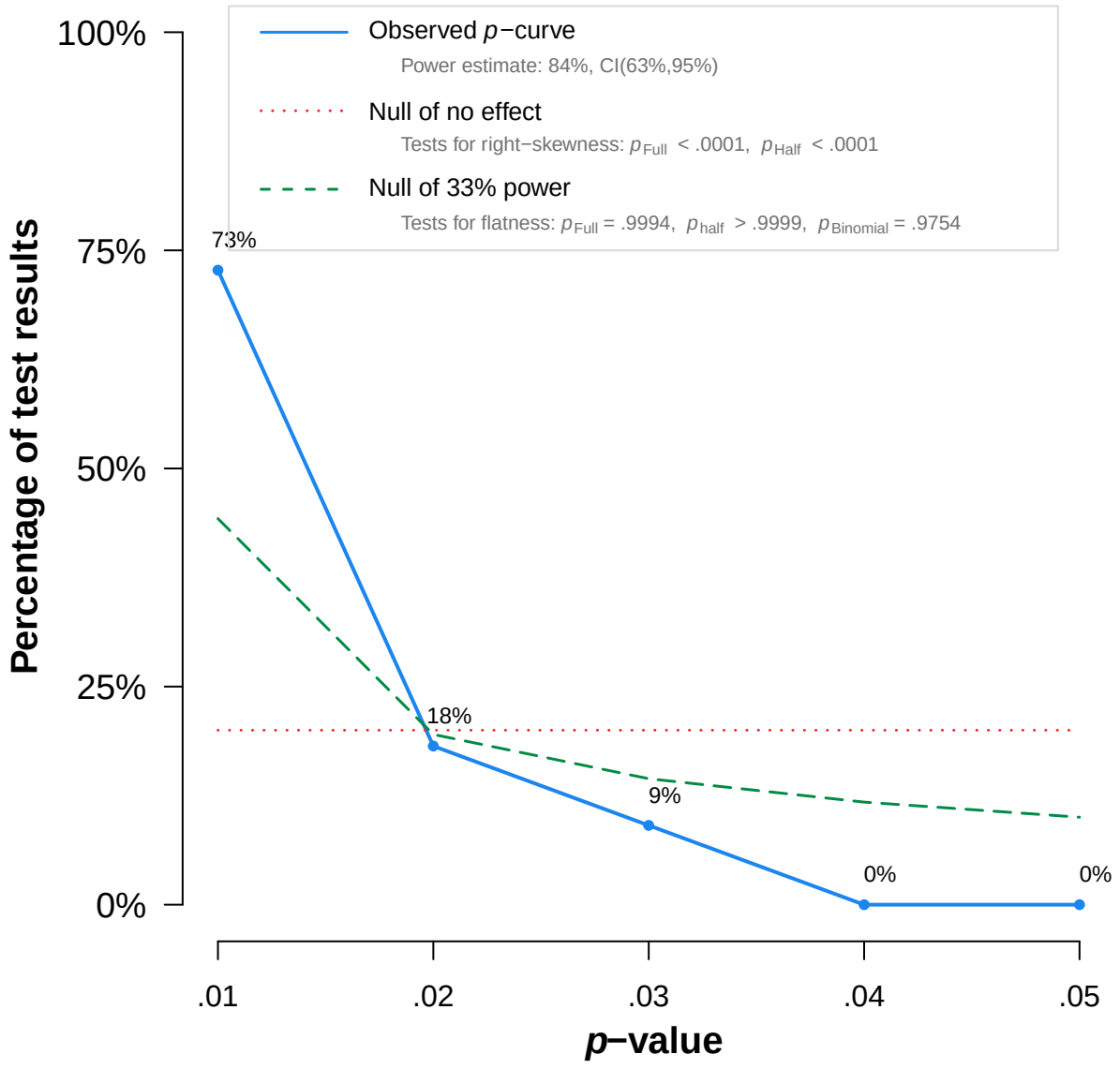
```
library(dmetar)
```

If you don't want to use the `dmetar` package, you can find the source code for this function here. In this case, *R* doesn't know this function yet, so we have to let *R* learn it by **copying and pasting** the code **in its entirety** into the **console** on the bottom left pane of RStudio, and then hit **Enter** . The function requires the `stringr` and `poibin` package to work. **For this function, the following parameters need to be specified:**

Parameter	Function
x	The meta-analysis results object generated by meta functions.
effect.estimation	Logical. Should the true effect size underlying the p-curve be estimated? If set to TRUE, a vector containing the total sample size for each study must be provided for N. FALSE by default.
N	A numeric vector of same length as the number of effect sizes included in x specifiying the total sample size N corresponding to each effect. Only needed if effect.estimation = TRUE.
dmin	If effect.estimation = TRUE: lower limit for the effect size (d) space in which the true effect size should be searched. Must be greater or equal to 0. Default is 0.
dmax	If effect.estimation = TRUE: upper limit for the effect size (d) space in which the true effect size should be searched. Must be greater than 0. Default is 1.

First, let's use the `pcurve` function with `effect.estimation` set to FALSE. As this is the default, we only have to plug the `m.hksj` meta-analysis object into the function to generate the p-curve.

```
pcurve(m.hksj)
```



Note: The observed p−curve includes 11 statistically significant (p < .05) results, of which 10 are p < .025. There were 7 additional results entered but excluded from p−curve because they were p > .05.

```
## P-curve analysis
##  ----------------------
## - Total number of provided studies: k = 18
## - Total number of p<0.05 studies included into the analysis: k = 11 (61.11%)
## - Total number of studies with p<0.025: k = 10 (55.56%)
##
## Results
##  ----------------------
##                     pBinomial  zFull pFull  zHalf pHalf
## Right-skewness test     0.006 -5.943 0.000 -4.982     0
## Flatness test           0.975  3.260 0.999  5.158     1
## Note: p-values of 0 or 1 correspond to p<0.001 and p>0.999, respectively.
## Power Estimate: 84% (62.7%-94.6%)
##
## Evidential value
##  ----------------------
## - Evidential value present: yes
## - Evidential value absent/inadequate: no
##
```

The function produces a large output, so let us go through it one by one:

- **P-Curve plot**. The figure shows the p-curve for your results (in blue). On the bottom, you can also find the number of effect sizes with $p < 0.05$ which were included in the analysis. Results of the Right-Skewness and Flatness test are also displayed (see Results).
- **P-curve analysis**. This section of the output provides general details about the studies used for the p-curve analysis, such as the number of studies in the meta-analysis (in total), the number of significant effect sizes used for the p-curve analysis, and number of studies with effect sizes for which $p < 0.025$ (the so-called "half-curve").
- **Results**. This section displays results of the Right-Skewness test and the Flatness test. The Right-Skewness tests analyzes if the p-curve resulting from your data is significantly right-skewed, which would indicate that there is a "true" effect behind your data. The flatness test analyzes if the p-curve is flat, which could indicate that the power is insufficient, or that there is no "true" effect behind your data. For both tests, result are reported for the full p-curve (all values for which $p < 0.05$) and for the half p-curve (all values for which $p < 0.025$).
- **Power Estimate**. This line displays the estimated power of the studies in your analysis, and the confidence interval.
- **Evidential value**. In terms of interpretation, this section is the most important one of the output. It shows if P-curve estimates that evidential value (a "true" effect) is present in the analysis or not. This interpretation is done automatically based on the values of the Flatness and Right-Skewness test (you can read the documentation of the function for more information on how this is done). There are two types of information provided: (i) if evidential value is present, or (ii) if it is absent or inadequate. This may look a little peculiar at first, because we would expect a simple yes/no interpretation concerning the evidential value of our analysis. However, it is possible that both `Evidential value present` and `Evidential value absent/inadequate` result in a no decision. This basically means that while the presence of evidential value could not be ascertained, it could also not be verified that evidential value is indeed absent or inadequate (for example because a very small effect exists).

For our `m.hksj` object, we see that 11 studies were included into the analysis, of which 10 had a p-value lower than 0.025. We also see that the Power of the analysis was 84% (95%CI: 62.7%-94.6%). We are provided with the interpretation that **evidential value is present**, and that evidential value is **not absent or inadequate**. This means that P-curve estimates that there is a "true" effect size behind our findings, and that the results are not the product of publication bias and p-hacking alone.

9.3.2 Estimating the "true" effect

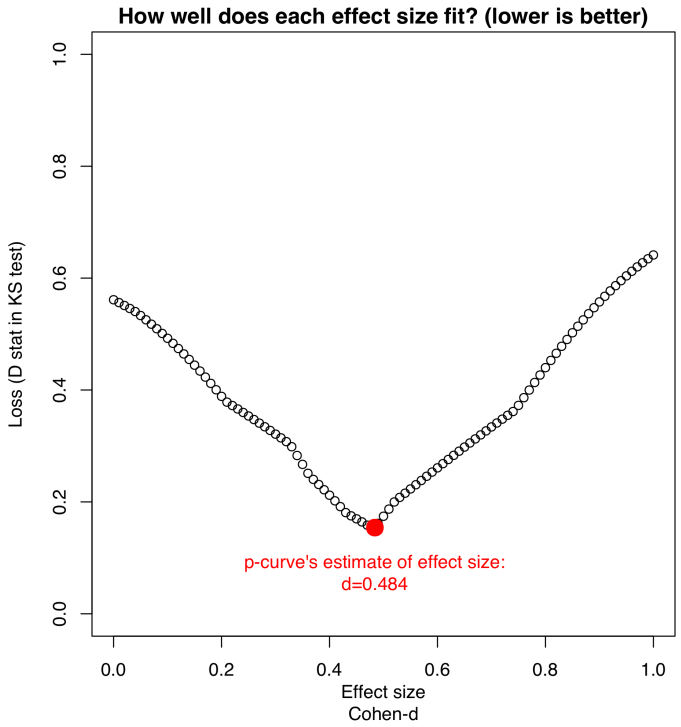
The `pcurve` function also allows you to estimate P-curve's estimate of the "true" effect size underlying your data (much like the Duval & Tweedie trim-and-fill procedure we described before). However, there is one important information we need to do this: we need to know the **total sample size of each study**. Thankfully, this information is usually reported in most research publications. Let's assume i have stored a variable called `N.m.hksj` in *R*, which contains the total sample size for each study contained in `m.hksj` (in the same order as the studies in `m.hksj`). Let's have a look at `N.m.hksj` first.

```
N.m.hksj
```

```
##  [1] 105 161  60  37 141  82  97  61 200  79 124  25 166  59 201  95 166
## [18] 144
```

With this information, we can extend the `pcurve` call from before with some additional arguments. First, we have to set `effect.estimation` to TRUE to estimate the effect size. Second, we have to provide the function with the study sample sizes `N.m.hksj`. Lastly, we can specify the range of effect sizes in which the function should search for the true effect (expressed as Cohen's d) through `dmin` and `dmax`. We will search for the effect between $d = 0.0$ and $d = 1.0$. The function returns the same output as before, and one additional plot:

```
pcurve(m.hksj, effect.estimation = TRUE, N = N.m.hksj, dmin = 0, dmax = 1)
```

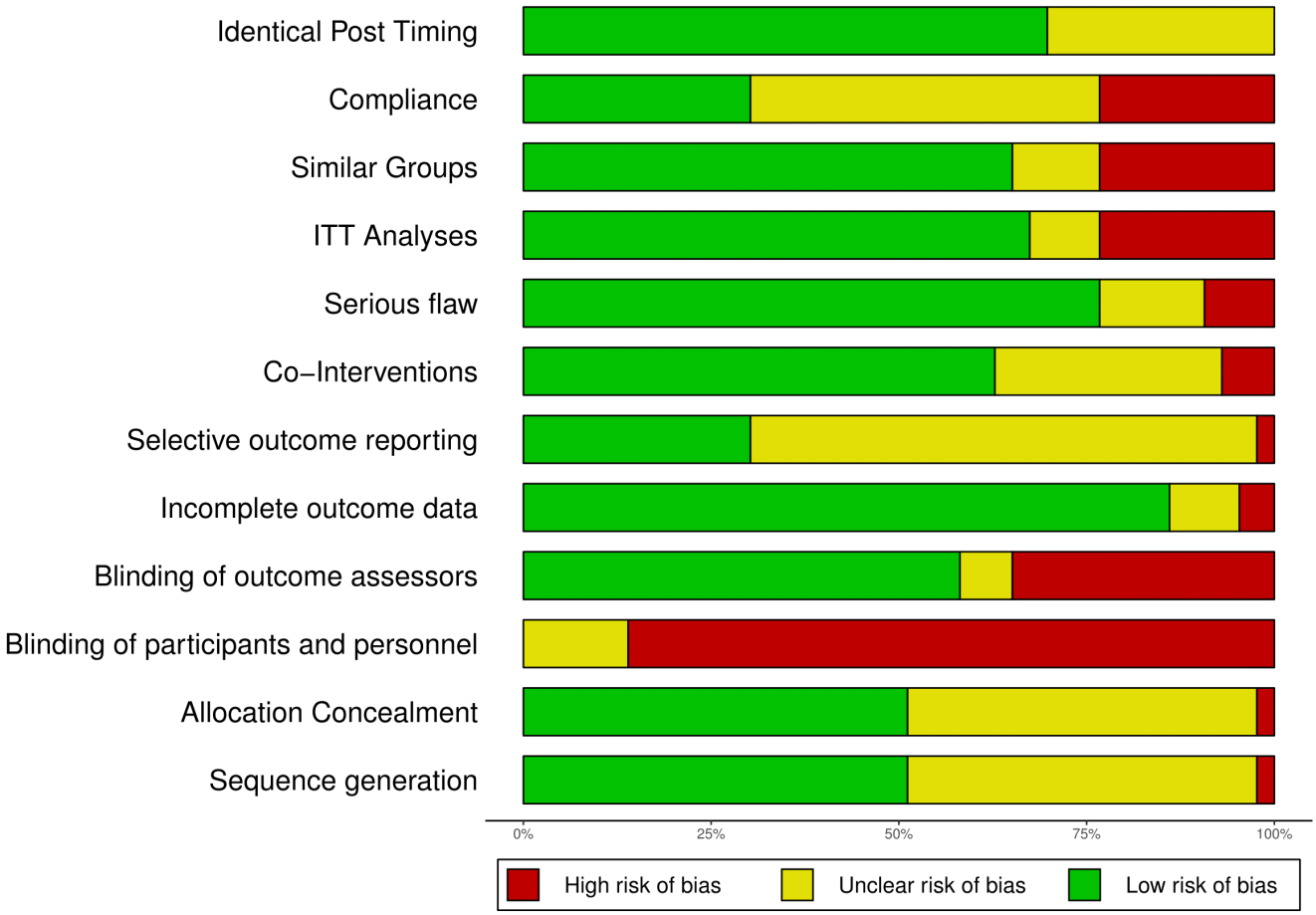


As can be seen in the plot, the function provides an effect estimate of $d = 0.48$. This effect mirrors the one we found for `m.hksj` when using the fixed-effect model, while the effect size for the random-effects model ($g = 0.59$) was somewhat higher.

It should be noted that this chapter should only be seen as **an introduction into P-curve**, and should not be seen as comprehensive. Simonsohn et al. (Simonsohn et al., 2015) also stress that P-Curve should only be used for **outcome data which was actually of interest for the authors of the specific article**, because those are the one's likely to get p-hacked. They also ask meta-researchers to provide a detailed table in which the reported results of each outcome data used in the P-curve is documented (a guide can be found here). It has also been shown that P-Curve's effect estimate are **not robust** when the heterogeneity of a meta-analyis is high ($I^2 > 50\%$). Van Aert et al. (van Aert et al., 2016) propose not to determine the "true" effect using P-Curve when heterogeneity is high (defined as $I^2 > 50\%$). The `pcurve` function therefore automatically prints a warning message at the end of the output if a true effect is estimated and heterogeneity is considerable. A poosible **solution** for this problem might be to reduce the overall heterogeneity using outlier removal, or to p-curve results in more homogeneous subgroups.

Chapter 10

Risk of Bias Summary



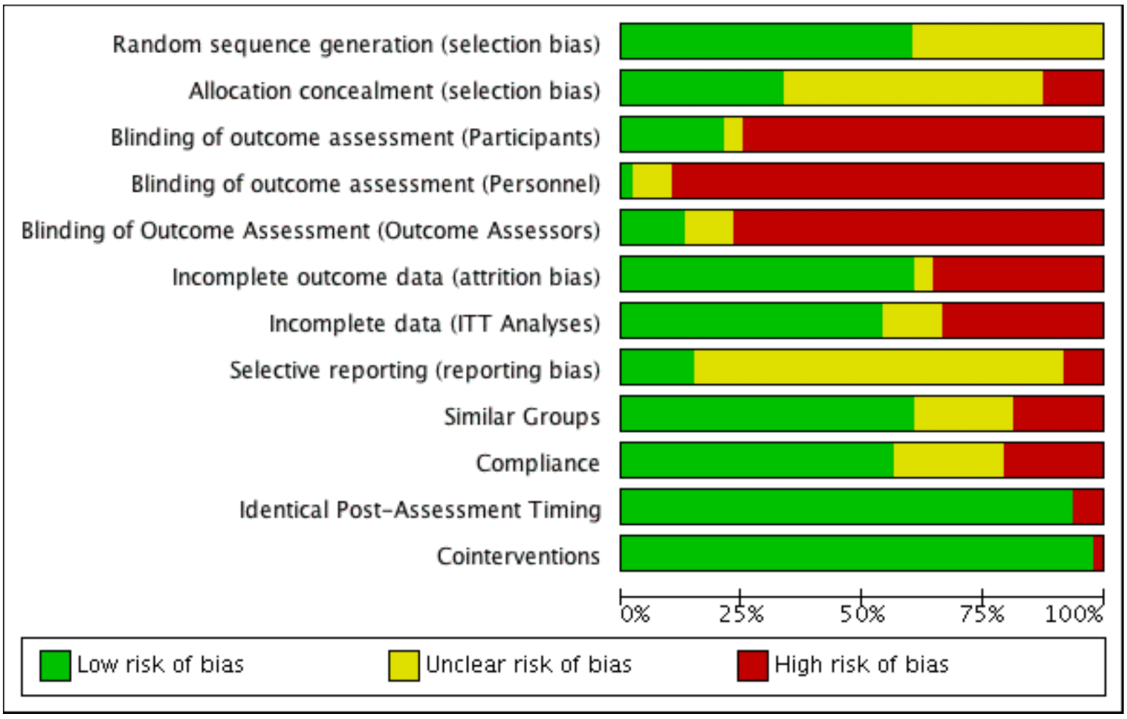
In this Chapter, we will describe how to create a RevMan style Risk of Bias summary in *R*.

10.1 Preparing your Risk of Bias data

In many Meta-Analyses, you will also want to have a look at the quality of included studies using the **RevMan Risk of Bias assessment tool**. However, many researchers don't use **RevMan** to conduct Meta-Analyses, one has to put

some extra effort into inserting all study quality data by hand into RevMan, which is extremely time-consuming. Furthermore, the quality of the Risk of Bias (RoB) summary figures in RevMan are of suboptimal picture quality. Many journals will require figures with better quality, or figures saved in another format (such as **.svg** or **.pdf**).

Figure 2. Risk of bias summary: authors' judgements broken down for each risk of bias criterion across all included studies



To avoid all of this, you can easily plot the **RoB Summary in RStudio yourself**. To do this, we again have to prepare an Excel sheet in which we store our RoB data. In Chapter 3.1.1, we already described how the preparation and import of Excel sheets into RStudio works in general. For this data sheet, you need to follow a few guidelines:

- Name the first column **Author**, and put all the study names in this column (e.g., Frogeli et al.)
- Give the other **columns a name signifying the RoB criterion** you assessed. Do this for all criteria you want to have included in your plot. **Important**: when naming your column, do not use spaces between words, but use underscores or points instead (e.g. allocation_concealment)
- In these columns, you have to describe if the study received a **High**, **Low**, or **Unclear** risk of bias rating. Use exactly these codes for your data, including upper and lowercase (*R* is case sensitive).
- Do **not** store any other information in your data (e.g., commentaries on your RoB decision).

10.2 Plotting the summary

To plot the summary, we have to import our dataset first. We describe how to do this in Chapter 3.2. I simply called my dataset rob.

Let's have a look at the structure of the data first:

```
str(rob)
```

```
## 'data.frame': 43 obs. of 13 variables:
## $ Sequence_Generation : chr "Low" "Low" "Low" "Low" ...
## $ Allocation_Concealment_ : chr "Low" "Low" "Low" "Low" ...
## $ Blinding.of.Participants.and.Personnel: chr "High" "High" "High" "High"
...
## $ Blinding.of.Outcome.Assessors : chr "Low" "Low" "Low" "Low" ...
## $ Incomplete.Outcome.Data : chr "Low" "Low" "Low" "Low" ...
## $ Selective_Outcome_Reporting : chr "Low" "Unclear" "Low" "Low" ...
## $ Cointerventions : chr "Unclear" "Low" "Low" "Low" ...
## $ Serious_Flaw : chr "Low" "Low" "Low" "Low" ...
## $ Intention_to_treat_Analyses : chr "Low" "Low" "Low" "Low" ...
## $ Similar_Groups : chr "Unclear" "Low" "Low" "Low" ...
## $ Compliance : chr "Low" "Unclear" "High" "High" ...
## $ Identical_Post_Timing : chr "Low" "Low" "Low" "Low" ...
## $ Author : chr "BiesheuvelLeliefeld 2017" "Bockting 2005 & 2010 & 2015"
"Bockting 2018" "Bockting 2018" ...
```

We can see that we have the data imported in RStudio now, with ratings for every criterion in each column. Whether you named your columns differently or used less or more criteria is not important. To plot the risk of bias summary, we have prepared a function called `rob.summary` for you. The `rob.summary` function is part of the `dmetar` package. If you have the package installed already, you have to load it into your library first.

```
library(dmetar)
```

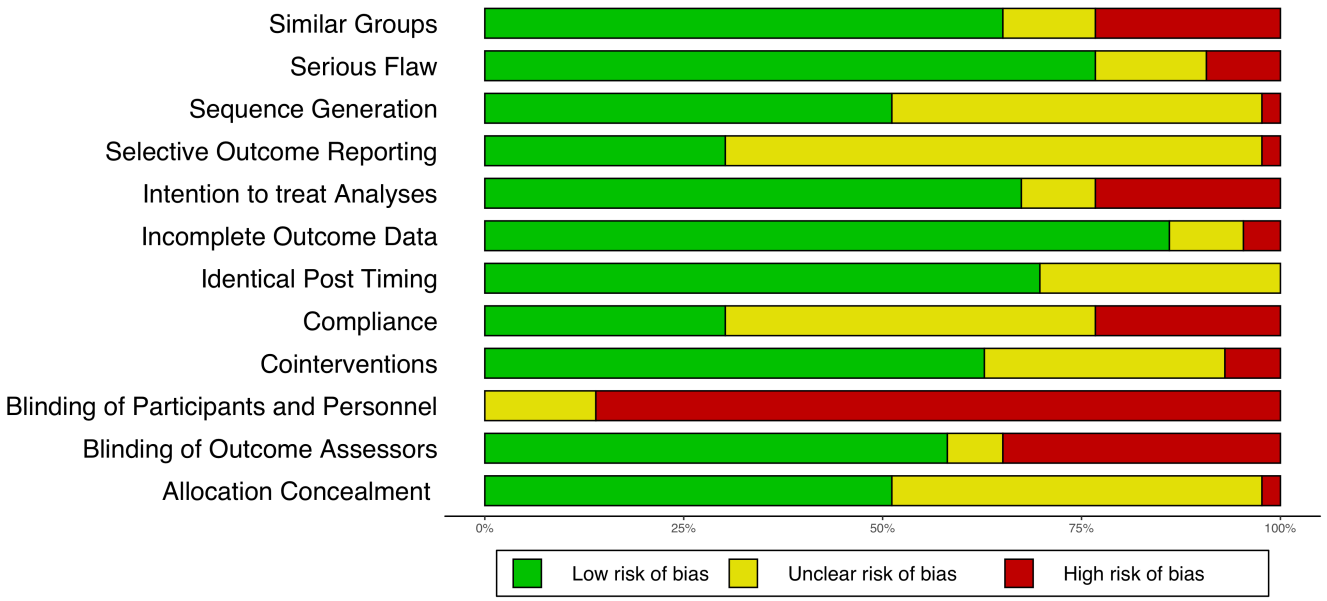
If you don't want to use the `dmetar` package, you can find the source code for this function here. In this case, *R* doesn't know this function yet, so we have to let *R* learn it by **copying and pasting** the code **in its entirety** into the **console** in the bottom left pane of RStudio, and then hit **Enter**. The function then requires the `tidyr` and `ggplot2` package to work.

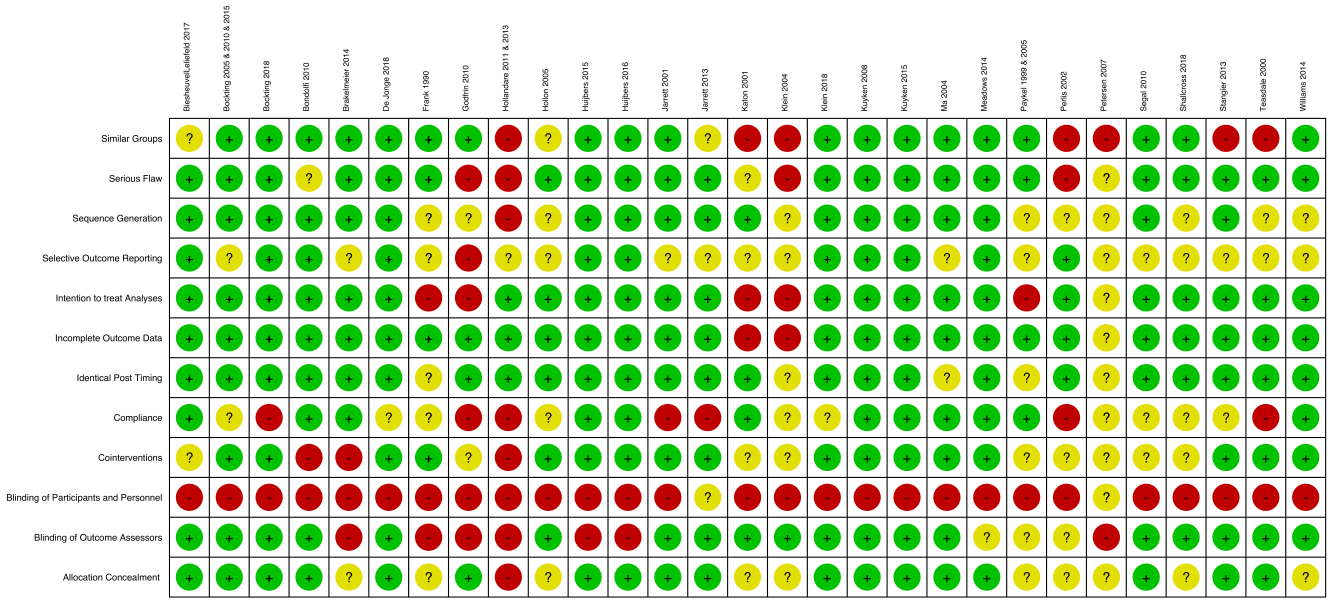
The function has the following parameters:

Parameter	Function
data	A data.frame containing a column for each risk of bias criterion, where rows represent each individual studies. The risk of bias assessment for each criterion for each study must be coded as a character string. Up to four codes can be used, referring to low risk of bias, unclear risk of bias, high risk of bias, or missing information. The string used to specify the categories must be specified in name.high, name.unclear, name.low and/or name.missing, unless defaults for those parameters are used.
name.high	Character specifying how the 'high risk of bias' category was coded in data (e.g., name.high = 'high'). Default is 'High'.
name.unclear	Character specifying how the 'unclear risk of bias' category was coded in data (e.g., name.unclear = 'unclear'). Default is 'Unclear'.
name.low	Character specifying how the 'low risk of bias' category was coded in data (e.g., name.low = 'low'). Default is 'Low'.
studies	A vector of the same length as the number of rows in data specifying the study labels for the risk of bias ratings. Only has to be specified when table = TRUE.
name.missing	Character specifying how missing information was coded in data (e.g., name.missing = 'missing'). Default is 'Missing'. All ratings, including missing information, must be coded as strings, so using NA in data to signify missing information is not valid.
table	Should an additional RevMan style risk of bias table be produced? If set to TRUE, studies must be specified. FALSE by default.

If you named the columns and Risk of Bias levels as suggested before, you only have to provide the `rob.summary` function with your dataset and the study labels in `studies`. As we want to produce both the summary plot and the *RevMan* Risk of Bias table, we specify `table` as TRUE (this is only possible if `studies` is specified).

```
rob.summary(rob, studies = rob$Author, table = TRUE)
```





Looks good so far. As you can see, the `rob.summary` also detected words separated by "_" or "." in the column names and automatically cleaned the output from those symbols.

10.3 Saving the Summary Plot

I want to save the plot as a **PDF** file in my working directory. To do this, define the name of the file as `rob_summary.pdf`, and save it at the same time in the correct size and orientation using this code:

```
pdf(file='rob_summary.pdf', width = 13, height = 6)
rob.summary(rob); dev.off()
```

I can also save the plot as **PNG** file using this command.

```
png(file='rob_summary.png', width = 900, height = 595)
rob.summary(rob); dev.off()
```

Or as a **Scalable Vector Graphic** (.svg) with this command.

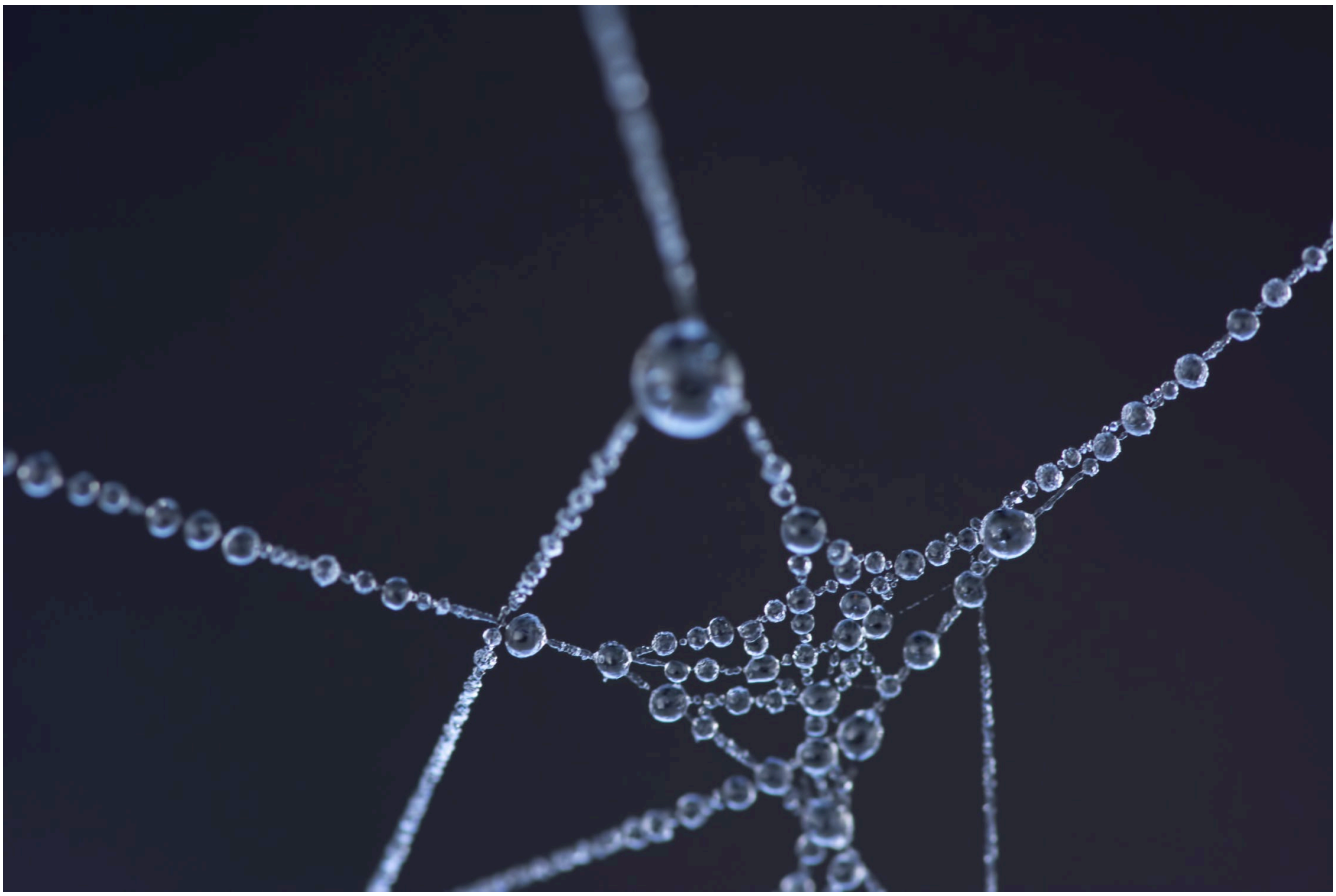
```
svg(file='rob_summary.svg', width = 13, height = 6)
rob.summary(rob); dev.off()
```

Part III

Advanced Topics

Chapter 11

Network Meta-Analysis



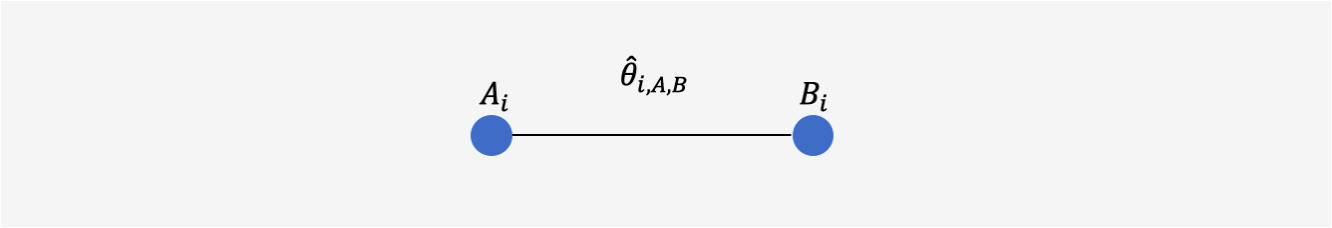
Often when performing a meta-analysis on the effectiveness of certain interventions, we are less interested in the question if **one particular intervention is effective** (e.g., because it is quite well established that the intervention is efficacious), but weather **one intervention is more or less effective than another type of intervention for some condition or population**. Yet, once we are interested in **head-to-head** comparisons between two treatments, we often face the problem that **only very few, if any, randomized controlled trials have compared the effects of two interventions directly**. This makes it very hard, if not impossible to conduct conventional meta-analyses to answer questions on the comparative effects of two or more interventions for one indication or outcome (e.g., different types of psychotherapy for major depression).

Nevertheless, while direct comparisons between two or more interventions may not exist, it is often the case that such interventions were evaluated in separate randomized controlled trials using the same **control group** (e.g., wait-list control groups, or placebos). This means that we do have **indirect comparisons** of the effects of different interventions, because they were compared to the same control condition. **Multiple-treatments meta-analysis (MTM)** is an extension of conventional meta-analysis which allows us to **incorporate such indirect comparisons**, and thus the simultaneous analysis of several interventions.

These meta-analysis methods are also referred to as **network meta-analyses** (Dias et al., 2013) or **mixed-treatment comparison** meta-analyses (van Valkenhoef et al., 2012), because such methods allow for **multiple direct and indirect intervention comparisons to be integrated into our analysis**, which can be formalized as a **"network"** of comparisons. Network Meta-Analysis is a "hot" research topic, and in the last decade, its methodology has been increasingly picked up by applied researchers in the medical field (Schwarzer et al., 2015; Efthimiou et al., 2016). However, Network Meta-Analysis also comes with additional **challenges and potential pitfalls**, particularly in terms of heterogeneity or network **inconsistency** (Salanti et al., 2014; Schwarzer et al., 2015). Therefore, it is very important to first discuss the core components and assumptions of network meta-analytical models before we proceed. The underpinnings of network meta-analysis can be a little abstract at times; we will therefore go through the essential parts in small steps to get a better understanding of the idea behind network meta-analysis models.

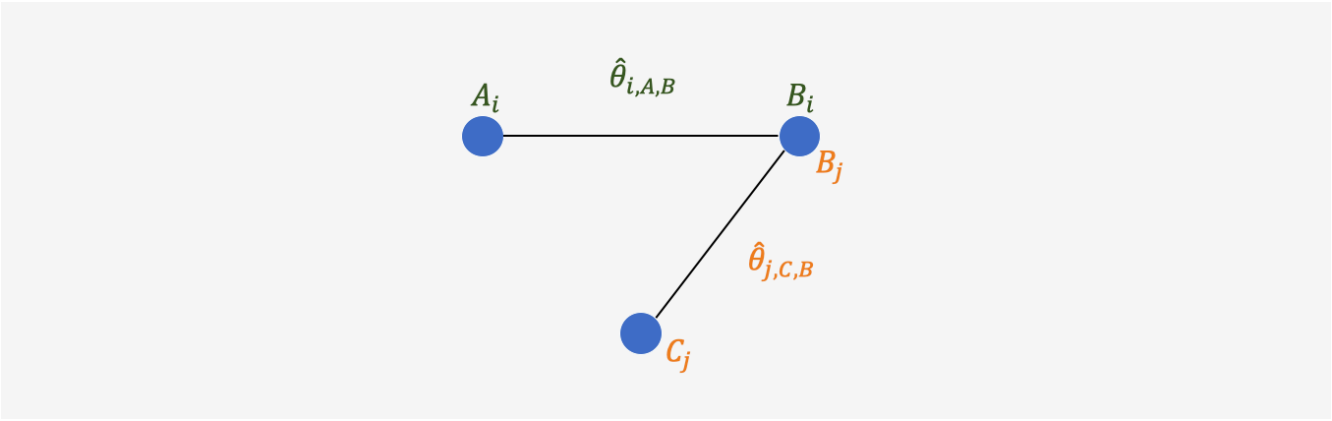
11.1 The idea behind network meta-analysis

First, we have to understand what meta-analysts **mean** when they talk about a "network" of treatments. Let us first consider a simple **pairwise comparison** between two conditions. The example we present here is no different from the kind of data we discussed in Chapter 4, where we showed how to perform a conventional meta-analysis. Let us assume we have a randomized controlled trial i, which **compared the effect of one treatment A** (e.g., Cognitive Behavioral Therapy for depression) to **another condition B** (e.g., a waitlist control group). We can illustrate this comparison in a graphical way like this:



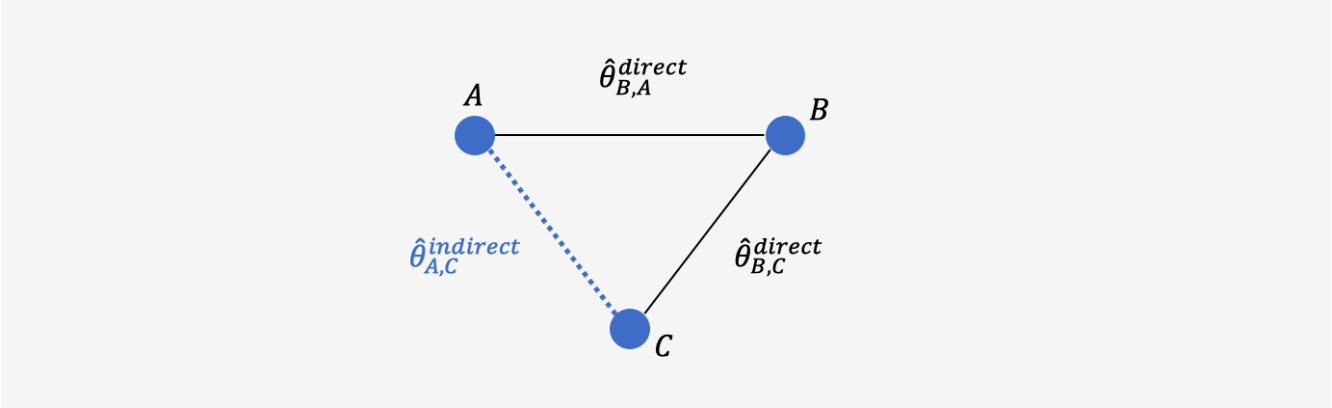
The form in which the treatment comparison is displayed here is called a **graph**. Graphs are structures used to model **how different objects relate to each other**, and there is an entire subfield of mathematics devoted to it: **Graph theory**. The graph has two core components. First, the two blue points (so-called **nodes**), which represent the **two conditions** A **and** B in trial i. The second component is the **line** connecting the two nodes, which is called an **edge**. This edge represents how A and B relate to each other. In our case, the interpretation of this line is quite straightforward: we can describe the relationship between A and B as the **effect size** $\hat{\theta}_{i,A,B}$ we observe for the comparison between A and B. This effect size can be expressed through different metrics, such as the **standardized mean difference (SMD)**, **Hedges' g**, **Odds Ratio**, **Incidence Rate Ratio**, and so forth, depending on the context.

Now let us proceed by assuming that we also have **data of another study** j. In this trial, the condition B (which we imagined to be a waitlist control group) was also included. But instead of using treatment A, like in the first study, this study **used another treatment** C (e.g., psychodynamic therapy), which was compared to B. We can add this information to our graph:



This creates our first small network. We can see now that we have **two effect size estimates** in this network: $\hat{\theta}_{i,A,B}$, comparing A to B, and $\hat{\theta}_{j,C,B}$, the comparison between C and B. Since we took both of these effect size estimates from **actual comparisons which were made in "real" randomized trials**, we call such information **direct evidence**. Thus, we can formalize these effect sizes as $\hat{\theta}^{direct}_{B,A}$ and $\hat{\theta}^{direct}_{B,C}$. Notice how B comes first in this notation because we determine this condition to be our **reference condition**, since both effect size estimates contain this condition.

In this first graph, all nodes (conditions) are either **directly or indirectly connected**. The B condition (our waitlist control group), is directly connected to all other nodes, i.e., it takes only one "step" on the graph to get from B to all the other nodes A and C: $B \rightarrow A, B \rightarrow C$. A and C both have only one direct connection, and they both connect to B: $A \rightarrow B$ and $C \rightarrow B$. However, there is an **indirect connection** between A and C, where B serves as the **link**, or **"brige"** between the two conditions: $A \rightarrow B \rightarrow C$. This indirect connection means that we have **indirect evidence** for the relationship between A and C, which we can infer from the information the entire network provides us with:



Using the information from our direct evidence, we can therefore calculate the **indirect evidence** $\hat{\theta}^{indirect}_{A,C}$, the effect size between A **and** C (e.g., Cognitive-Behavioral Therapy and Psychodynamic Therapy) like this:

$$\hat{\theta}^{indirect}_{A,C} = \hat{\theta}^{direct}_{B,A} - \hat{\theta}^{direct}_{B,C} \tag{1}$$

This is a crucial component of network meta-analytical models. This equation effectively lets us calculate an **estimate of the effect size of a comparison, even if the two conditions were never directly compared in an RCT**. Network Meta-Analysis in general means that we can combine both direct and indirect evidence in one model to estimate the effect sizes resulting from several treatment comparisons. This also means that even if there is direct evidence for a specific comparison (e.g. $A - B$), we can also **add information from indirect evidence** to further "fortify" our model and

make our effect sizes estimations **even more precise** (thus the name **mixed-treatment comparison meta-analysis**). The example we gave you here should illustrate the great strength of network meta-analytical models:

- They allow us to pool **all available information** in a set of connected studies in one analysis. Imagine how we would usually deal in pairwise meta-analysis with trials comparing different treatments to, say, a placebo. We would have to pool each comparison (e.g. treatment A compared to a placebo, treatment B compared to a placebo, treatment A compared to treatment B) in a seperate meta-analysis.
- They allow us to incorporate **indirect evidence** in a network, which we have to discard in conventional meta-analysis. Usually in pairwise meta-analysis, we can only pool direct evidence from comparisons which were actually conducted and reported in randomized controlled trials.
- If all assumptions are met, and results are conclusive enough, network meta-analyses allow us to draw cogent conclusions concerning **which type of treatment may be more or less preferable** for a specific target population under study.

Of course all of this sounds intruiging, but there are some important limitations we have to consider here. First, we should look at how the **variance** of the indirect effect size estimate is calculated:

$$V_{A,C}^{indirect} = V_{B,A}^{direct} + V_{B,C}^{direct} \tag{2}$$

As you can see, to calculate the variance of the indirect comparison, we actually **add up** the variance of the direct comparisons. This basically means that the effect size estimated from indirect evidence will always have a greater variance, and thus a lower precision than direct evidence (Dias et al., 2018). This, of course, makes quite a lot of sense because we have a higher confidence in effect size estimates which were **actually observed** (because researchers actually performed a study using this comparison), and thus give it a higher weight, compared to effect size estimates derived from indirect evidence. Furthermore, an essential point is that equation (1) only holds if a **core assumption of network meta-analysis** is met: the assumption of **transitivity**, or statistically speaking, **network consistency** (Efthimiou et al., 2016). We will explain what this means in the following, and why this assumption is important.

11.1.1 Transitivity and Consistency

Although network meta-analysis is certainly a valuable extension of the meta-analytical arsenal, the validity of this method has not remained uncontested. Most of the **criticism** of network meta-analysis revolves around, as you might have guessed, the **use of indirect evidence**, especially when direct evidence for a comparison is actually available (Edwards et al., 2009; Ioannidis, 2006). The key issue addressed here is that while participants in a randomized controlled trial (which we use as direct evidence in network meta-analysis) are randomly allocated to one of the treatment conditions (e.g., A and B), the **treatment conditions themselves** ($A, B, ...$) **were not randomly selected in the trials** included in our network (Edwards et al., 2009). This is of course quite logical, since we have not, for example, forced all researchers to determine which conditions they compare in their trial through, for example, a dice roll, before they were allowed to roll out their study. However, the fact that the selected treatment comparisons in our study pool will hardly ever follow a random pattern across trials does **not constitute a problem** for network meta-analytical models *per se* (Dias et al., 2018). In fact, what is required for equations (1) and (2) to hold is the following: **the selection, or non-selection, of a specific comparison in a specific trial must be unrelated to the true (relative) effect size of that comparison** (Dias et al., 2013). This statement is very abstract, so let us elaborate on it a little.

This requirement is derived from the **transitivity assumption** of network meta-analyses. There is disagreement about whether this is an additional assumption of network meta-analysis, or simply an extension of the assumptions of

standard pairwise meta-analysis; this disagreement may also be partly caused by an inconsistent usage of terms in the literature (Dias et al., 2018; Efthimiou et al., 2016; Song et al., 2009; Lu and Ades, 2009). The transitivity assumption's core tenet is that we can **combine direct evidence** (e.g. from the comparisons $A - B$ and $C - B$) to create indirect evidence about a related comparison (e.g. $A - C$), as we have already expressed in formula (1) above (Efthimiou et al., 2016).

The assumption of transitivity also relates to, or is derived from, the **exchangeability assumption** we described in our Chapter about the random-effects model. This assumption presupposes that an effect size $\hat{\theta}_i$ of a comparison i is randomly drawn from an "overarching" distribution of true effect sizes, the mean of which can be estimated. Translating this assumption to our scenario, we can think about **network meta-analytical models as consisting of a set of K trials which each contain all possible M treatment comparisons** (e.g. $A - B$,$A - C$,$B - C$, and so forth), but that **some of the treatment comparisons have been "deleted", and are thus "missing"" in some trials**; the reason for this of course being that studies in practice do not assess *all* possible treatment options for a specific condition, but only two or three (Dias et al., 2018). The key assumption here is that the relative effect of a comparison, e.g. $A - B$ is *exchangeable* between trials, no matter if a trial actually assessed this comparison or if this comparison is "missing". The assumption of exchangeability thus basically means that the effect size $\hat{\theta}$ of a specific comparison (e.g. $A - B$) must stem from a random draw from the same overarching distribution of effect sizes, no matter if this effect size is derived through direct or indirect evidence.

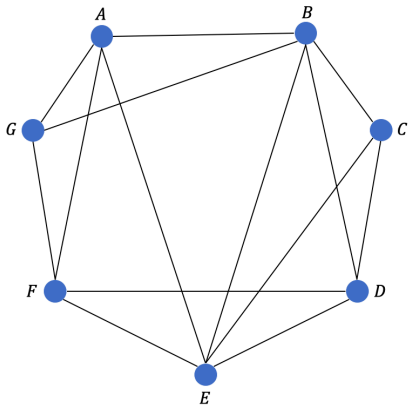
The assumption of transitivity may be violated when **covariates**, or **effect modifiers** (such as the age of the studied populations, or the treatment intensity) are not evenly distributed across trials reporting data on, for example, $A-B$ and $C-B$ comparisons (Song et al., 2009). Transitivity as such can not be tested statistically, but the risk for violating this assumption may be **attenuated by only including studies for which the population, methodology and studied target condition is as similar as possible** (Salanti et al., 2014).

The statistical manifestation of transitivity has been referred to as **consistency** (Efthimiou et al., 2016; Cipriani et al., 2013). Consistency means that the **direct evidence** in a network for the effect size between two treatments (e.g. A and B) **does not differ from the indirect evidence calculated for that same comparison** (Schwarzer et al., 2015):

$$\theta_{A,B}^{indirect} = \theta_{A,B}^{direct}$$

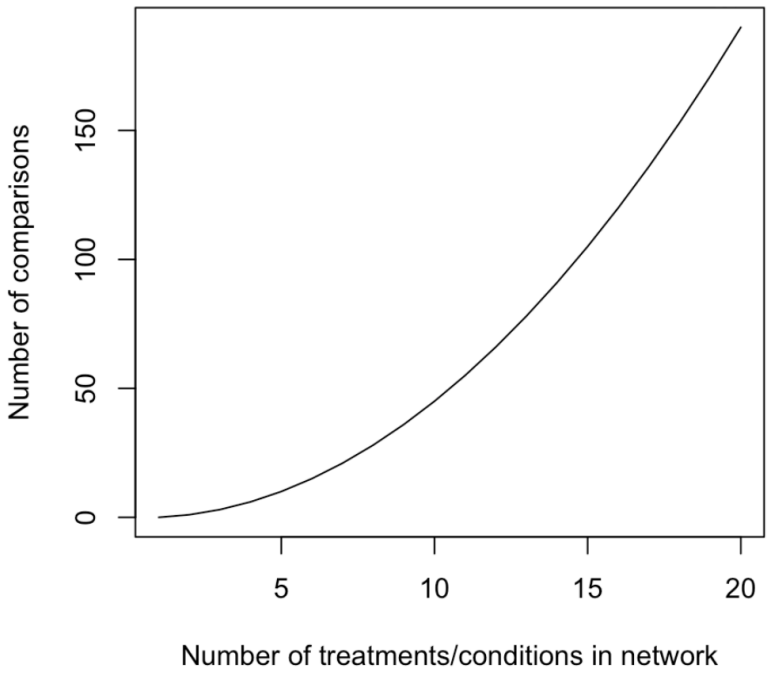
Several methods have been proposed to evaluate inconsistency in network meta-analysis models, including **net heat plots** (Krahn et al., 2013) and **node splitting** (Dias et al., 2010). We will describe these methods in further detail in the following two subchapters where we explain how to perform a network meta-analysis in R.

Above, we described the **basic theory and assumptions of network meta-analysis models**. We illustrated these properties using a simple network with three nodes and edges. In practice, however, the **number of treatments** we as meta-analysts want to include in a network meta-analysis **may be much higher**, resulting in much more **complex network**, which may look more like this:



However, with an increasing number of treatments S in our network, the number of (direct and indirect) pairwise comparisons C we have to estimate skyrockets (Dias et al., 2018):

$$C = S\frac{(S-1)}{2}$$



We therefore need a **computational model** which allows us to efficiently pool all available network data in a coherent and internally consistent manner. Several statistical approaches have been developed for network meta-analysis (Efthimiou et al., 2016). In the following subchapters, we will discuss two major approaches, a **frequentist** as well as a **bayesian hierarchical model**, and how they can be implemented in R.

11.1.2 Which modeling approach should i use?

The good message is that while network meta-analysis models may differ in their statistical approach, they should produce **the same results** when the sample size is large (Shim et al., 2019), and none of them is more or less valid

than the other. You can therefore safely choose one or the other approach, depending on which one you find **more intuitive**, or depending on the **functionality** of the package which implements it (Efthimiou et al., 2016). One asset of frequentist models is that this approach is very common, and used for most applications in the statistical world. This means that many people might understand the results this method produces more easily. The frequentist network meta-analysis package `netmeta`, which we will present in the following, however, does not yet provide a straightforward way to conduct meta-regression, while this is possible using the bayesian approach.

In practice, a useful strategy may also be to choose one approach as the main analysis, and then perform the other approach as a **sensitivity analysis** (e.g. Cipriani et al., 2018). This makes it possible to compare where the two methods come to the same conclusion (which may indicate that these specific results are robust), and where they diverge.

11.2 Frequentist Network Meta-Analysis

In the following, we will describe how to **perform a network meta-analysis** using the `netmeta` package (Schwarzer et al., 2015; Rücker et al., 2015). This package allows to estimate network meta-analysis models within a **frequentist framework**, with its statistical approach derived from graph theoretical methods developed for electrical networks (Rücker, 2012).

11.2.1 The frequentist interpretation of probability

Frequentism is a common theoretical approach to interpret the probability of an event E. This approach defines the probability of E in terms of how *frequently* E occurs if we repeat some process (e.g., an experiment) many, many times (Aronow and Miller, 2019).

Frequentist ideas are at the core of many statistical procedures quantitative researchers use on a day to day basis, such as **significance testing**, calculating **confidence intervals** or interpreting **p-values**.

11.2.2 The Network Meta-Analysis Model

Let us now describe how the **network meta-analysis model** underlying the `netmeta` package is **formulated**. Let us assume that we have **collected effect size data** from K trials. We can then check all our K trials and then **count the total number of treatment comparisons** (e.g. some antidepressant vs. placebo) which are included in these trials. This number of pairwise comparisons is denoted by m. We then calculate the effect size $\hat{\theta}$ for each of the pairwise comparisons, and then put them together in a **vector**. You can think of a vector as an **object containing a collection of numbers**, like a `numeric` column in an R dataframe is basically a collection of numbers (see Chapter *3.3*). In mathematical notation, vectors are usually **written in bold**. We therefore denote the vector containing all pairwise effect sizes $(\hat{\theta}_1, \hat{\theta}_2, ..., \hat{\theta}_m)$ as $\hat{\boldsymbol{\theta}}$, and the vector containing the respective standard errors as $\mathbf{s} = s_1, s_2, ..., s_m$. Our model formula then looks like this:

$$\hat{\boldsymbol{\theta}} = \mathbf{X}\theta^{treatment} + \epsilon$$

Using this equation, we can basically imagine that our effect size vector $\hat{\boldsymbol{\theta}}$ was **"generated" by the right side of the formula**, our model. The first part, $\mathbf{X}$, is the $m \times n$ **design matrix**, in which the columns represent the different treatments n, and the rows the comparisons between treatments, which are signified by 1 and -1. The parameter ϵ is a vector of sampling errors ϵ_k, which are assumed to be randomly drawn from a gaussian normal distribution with mean 0 and variance s_i^2:

$$\epsilon_k \sim \mathcal{N}(0, s_i^2)$$

The really important part of the formula, and what we actually want to estimate, is the vector $\theta^{treatment}$. This vector contains the **"real" effects of the n treatments abstracted from the treatment comparisons we have in our network**, and in the end allows us to compare the effects of different treatments directly.

To exemplify this (see Schwarzer et al., 2015), let us assume we have $K = 5$ studies, each containing **one distinct treatment comparison**: $A-B$, $A-C$, $A-D$, $B-C$, and $B-D$. We then have a vector of (observed) treatment comparisons $\hat{\boldsymbol{\theta}} = (\hat{\theta}_{1,A,B}, \hat{\theta}_{2,A,C}, \hat{\theta}_{3,A,D}, \hat{\theta}_{4,B,C}, \hat{\theta}_{5,B,D})^\top$ and our vector containing the four (unknown) "true" effects for each treatment included in our network, $\theta^{treatment} = (\theta_A, \theta_B, \theta_C, \theta_D)^\top$. If we **plug** these parameters into our **model formula**, we get the following equation:

$$\hat{\boldsymbol{\theta}} = \mathbf{X}\theta^{treatment} + \epsilon$$

$$\begin{pmatrix} \hat{\theta}_{1,A,B} \\ \hat{\theta}_{2,A,C} \\ \hat{\theta}_{3,A,D} \\ \hat{\theta}_{4,B,C} \\ \hat{\theta}_{5,B,D} \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} \begin{pmatrix} \theta_A \\ \theta_B \\ \theta_C \\ \theta_D \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \end{pmatrix}$$

The **important part** to mention here is that in its current form, the model formula is problematic from a **mathematical point of view**. Right now, the model is **overparameterized**, meaning that too many parameters in our model have to be estimated based on the data we have. This has to do with the design matrix $\mathbf{X}$ not being of **full rank**. A matrix is not of full rank whenever the **rows** of the matrix are not all **independent**. Because we are dealing with a network

of comparisons it is clear that not all comparisons (rows) will be independent from each other in our matrix; in our example, the row for comparison $B - C$ is a **linear combination** of comparisons $A - B$ and $A - C$, and thus not independent. The fact that the $\mathbf{X}$ matrix is not of full rank, however, means that it is **not invertible**, which makes it impossible to directly estimate $\theta^{treatment}$ using a weighted least squares approach. While we have at best $n - 1$ *independent* treatment comparisons, our model at the same time **has to estimate the "true" effect for each of the** n **treatments** in $\theta^{treatment}$.

This is where the **graph theoretical approach** of the `netmeta` package comes in with a solution. We will spare you the mathematical details of this approach, particularly given that the `netmeta` package will do the heavy lifting for us anyway. Let us only mention that this approach involves contructing a so-called **Moore-Penrose pseudoinverse matrix** which then allows for calculating the fitted values of our network model using a weighted least squares approach. The procedure also takes care of **multiarm studies which contribute more than one pairwise comparison** (i.e., studies in which more than two treatments were compared with each other). Since such comparisons are correlated when coming from the same study (see Chapter 13.9), the standard errors of multiarm study comparisons are increased artificially. A random-effects model can also be incorporated into this approach by adding the **estimated heterogeneity** $\hat{\tau}^2$ to the variance of each comparison i: $s_i^2 + \hat{\tau}^2$. In the `netmeta` package, τ^2 is estimated using an adaptation of the **DerSimonian-Laird** estimator (Jackson et al., 2013). Values of I^2 are also calculated, which represent the amount of **inconsistency** in our network. The model allows to calculate an equivalent of **Cochran's** Q (see Chapter 6), Q_{total}, measuring the heterogeneity in the network. This can be used to calculate an I^2 equivalent for our network model:

$$I^2 = max(\frac{Q_{total} - df}{Q_{total}}, 0)$$

Where the **degrees of freedom** in our network (df) are:

$$df = (\sum_{k=1}^{K} p_k - 1) - (n - 1)$$

with K being the number of studies, p the number of arms in each study k, and n the total number of treatments in our entire network.

11.2.3 Performing a Network Meta-Analysis using the `netmeta` package

After all this statistical input, it is now time that we start working with the `netmeta` package. As always, we first **install the package** and then **load it into our library**.

```
install.packages("netmeta")
library(netmeta)
```

11.2.3.1 Data Preprocessing & Exploration

For our first network meta-analysis, we will use an **original dataset** reported by Senn et al. (2013). This dataset contains effect size data of randomized controlled trials comparing different medications for **diabetes**. The effect size of all comparisons denotes the **mean difference** (MD) in diabetic patients' **HbA1c value** at post-test. This value represents the concentration of glucose in the blood, which diabetic medication aims to decrease. We will use this

dataset because it is already **preinstalled** in the `netmeta` package, with all columns labeled correctly. To work with the dataset, we only need to **load it into our global environment** using the `data()` function. The dataset is then available as `Senn2013`, which we then rename to `data` for our convenience. Here is how you do this:

```
data(Senn2013)
data <- Senn2013
```

Now, let us have a look at the `data`.

TE	seTE	treat1.long	treat2.long	treat1	treat2	studlab
-1.90	0.1414	Metformin	Placebo	metf	plac	DeFronzo1995
-0.82	0.0992	Metformin	Placebo	metf	plac	Lewin2007
-0.20	0.3579	Metformin	Acarbose	metf	acar	Willms1999
-1.34	0.1435	Rosiglitazone	Placebo	rosi	plac	Davidson2007
-1.10	0.1141	Rosiglitazone	Placebo	rosi	plac	Wolffenbuttel1999
-1.30	0.1268	Pioglitazone	Placebo	piog	plac	Kipnes2001
-0.77	0.1078	Rosiglitazone	Placebo	rosi	plac	Kerenyi2004
0.16	0.0849	Pioglitazone	Metformin	piog	metf	Hanefeld2004
0.10	0.1831	Pioglitazone	Rosiglitazone	piog	rosi	Derosa2004
-1.30	0.1014	Rosiglitazone	Placebo	rosi	plac	Baksi2004
-1.09	0.2263	Rosiglitazone	Placebo	rosi	plac	Rosenstock2008
-1.50	0.1624	Rosiglitazone	Placebo	rosi	plac	Zhu2003
-0.14	0.2239	Rosiglitazone	Metformin	rosi	metf	Yang2003
-1.20	0.1436	Rosiglitazone	Sulfonylurea	rosi	sulf	Vongthavaravat2002
-0.40	0.1549	Acarbose	Sulfonylurea	acar	sulf	Oyama2008
-0.80	0.1432	Acarbose	Placebo	acar	plac	Costa1997
-0.57	0.1291	Sitagliptin	Placebo	sita	plac	Hermansen2007
-0.70	0.1273	Vildagliptin	Placebo	vild	plac	Garber2008
-0.37	0.1184	Metformin	Sulfonylurea	metf	sulf	Alex1998
-0.74	0.1839	Miglitol	Placebo	migl	plac	Johnston1994
-1.41	0.2235	Miglitol	Placebo	migl	plac	Johnston1998a
0.00	0.2339	Rosiglitazone	Metformin	rosi	metf	Kim2007
-0.68	0.2828	Miglitol	Placebo	migl	plac	Johnston1998b
-0.40	0.4356	Metformin	Placebo	metf	plac	Gonzalez-Ortiz2004
-0.23	0.3467	Benfluorex	Placebo	benf	plac	Stucci1996
-1.01	0.1366	Benfluorex	Placebo	benf	plac	Moulin2006
-1.20	0.3758	Metformin	Placebo	metf	plac	Willms1999
-1.00	0.4669	Acarbose	Placebo	acar	plac	Willms1999

We see that the data has 28 rows, representing the treatment comparisons, and **seven columns**, some of which may already seem familiar to you if you have worked yourself through previous chapters.

- The first column, **TE**, contains the **effect size** of each comparison, and **seTE** contains the respective **standard error**. In case you do not have precalculated effect size data for each comparison, you can first use the `metacont` (Chapter 4.1.2) or `metabin` function (Chapter 4.3), and then extract the calculated effect sizes from the meta-analysis object you created using the `$TE` and `$seTE` selector. Another, more flexible approach may be to use the **effect size calculators** we describe in Chapter 13.

- **treat1.long**, **treat2.long**, **treat1** and **treat2** represent the **two treatments being compared**. Variables **treat1** and **treat2** simply simply contain a shortened name of the original treatment name, and are thus redundant.
- The **studlab** column contains the **unique study label**, signifying in which study the specific treatment comparison was made. We can easily check if we have **multiarm studies** contributing more than one comparison by using the `summary()` function.

```
summary(data$studlab)
```

```
##           Alex1998          Baksi2004          Costa1997
##                  1                  1                  1
##       Davidson2007       DeFronzo1995        Derosa2004
##                  1                  1                  1
##         Garber2008 Gonzalez-Ortiz2004       Hanefeld2004
##                  1                  1                  1
##      Hermansen2007       Johnston1994      Johnston1998a
##                  1                  1                  1
##      Johnston1998b        Kerenyi2004           Kim2007
##                  1                  1                  1
##         Kipnes2001          Lewin2007        Moulin2006
##                  1                  1                  1
##          Oyama2008     Rosenstock2008        Stucci1996
##                  1                  1                  1
## Vongthavaravat2002         Willms1999  Wolffenbuttel1999
##                  1                  3                  1
##           Yang2003            Zhu2003
##                  1                  1
```

We see that all studies **only contribute one comparison**, except for `Willms1999`, which **contributes three**. For all later steps, it is essential that you (1) include the **studlab** column in your dataset, (2) each individual study gets a unique label/name in the column, and (3) studies which contribute 2+ comparisons are named exactly the same across comparisons.

11.2.3.2 Fitting the Model

We can now proceed by fitting **our first network meta-analysis model** using the `netmeta` function. The function has many, many parameters, and the most important ones are described below.

Code	Description
TE	The name of the column in our dataset containing the effect sizes for each comparison
seTE	The name of the column in our dataset containing the standard error of the effect size for each comparison
treat1	The column in our dataset containing the name of the first treatment in a comparison
treat2	The column in our dataset containing the name of the second treatment in a comparison
studlab	The column in our dataset containing the name of the study a comparison was extracted from. Although this argument is per se optional, we recommend to always specify it, because this is the only way to let the function know if multiarm trials are part of our network
data	The dataset containing all our network data

(continued)

Code	Description
sm	The summary measure underlying our TE column. This can be specified as 'RD' (Risk Difference), 'RR' (Risk Ratio), 'OR' (Odds Ratio), 'HR' (hazard ratio), 'MD' (mean difference), 'SMD' (standardized mean difference), etc.
comb.fixed	Whether a fixed-effects network meta-analysis should be conducted (TRUE/FALSE)
comb.random	Whether a random-effects network meta-analysis should be conducted (TRUE/FALSE)
reference.group	This lets us specify which treatment should be taken as a reference treatment (e.g. reference.group = 'placebo') for all other treatments
tol.multiarm	The effect sizes for comparisons from multi-arm studies are, by design, consistent. Sometimes however, original papers may report slightly deviating results for each comparison, which may result in a violation of consistency. This argument lets us specify a tolerance threshold (a numeric value) for the inconsistency of effect sizes and their variances allowed in our network
details.chkmultiarm	Wether we want to print out effect estimates of multiarm comparisons with inconsistent effect sizes.
sep.trts	The character trough which compared treatments are seperated, e.g. ' vs. '

I will save the the result of the function the object `m.netmeta`. Let's look at the **results of our first model**, for now assuming a **fixed-effects model**.

```
m.netmeta <- netmeta(TE = TE,
                     seTE = seTE,
                     treat1 = treat1,
                     treat2 = treat2,
                     studlab = paste(data$studlab),
                     data = data,
                     sm = "MD",
                     comb.fixed = TRUE,
                     comb.random = FALSE,
                     reference.group = "plac",
                     details.chkmultiarm = TRUE,
                     sep.trts = " vs ")
m.netmeta
```

```
## Original data (with adjusted standard errors for multi-arm studies):
##
##                    treat1 treat2      TE   seTE seTE.adj narms multiarm
## DeFronzo1995         metf   plac -1.9000 0.1414   0.1414     2
## Lewin2007            metf   plac -0.8200 0.0992   0.0992     2
## Willms1999           acar   metf  0.2000 0.3579   0.3884     3        *
## Davidson2007         plac   rosi  1.3400 0.1435   0.1435     2
## Wolffenbuttel1999    plac   rosi  1.1000 0.1141   0.1141     2
## Kipnes2001           piog   plac -1.3000 0.1268   0.1268     2
## Kerenyi2004          plac   rosi  0.7700 0.1078   0.1078     2
## Hanefeld2004         metf   piog -0.1600 0.0849   0.0849     2
## Derosa2004           piog   rosi  0.1000 0.1831   0.1831     2
## Baksi2004            plac   rosi  1.3000 0.1014   0.1014     2
```

```
## Rosenstock2008       plac   rosi  1.0900 0.2263   0.2263     2
## Zhu2003              plac   rosi  1.5000 0.1624   0.1624     2
## Yang2003             metf   rosi  0.1400 0.2239   0.2239     2
## Vongthavaravat2002   rosi   sulf -1.2000 0.1436   0.1436     2
## Oyama2008            acar   sulf -0.4000 0.1549   0.1549     2
## Costa1997            acar   plac -0.8000 0.1432   0.1432     2
## Hermansen2007        plac   sita  0.5700 0.1291   0.1291     2
## Garber2008           plac   vild  0.7000 0.1273   0.1273     2
## Alex1998             metf   sulf -0.3700 0.1184   0.1184     2
## Johnston1994         migl   plac -0.7400 0.1839   0.1839     2
## Johnston1998a        migl   plac -1.4100 0.2235   0.2235     2
## Kim2007              metf   rosi -0.0000 0.2339   0.2339     2
## Johnston1998b        migl   plac -0.6800 0.2828   0.2828     2
## Gonzalez-Ortiz2004   metf   plac -0.4000 0.4356   0.4356     2
## Stucci1996           benf   plac -0.2300 0.3467   0.3467     2
## Moulin2006           benf   plac -1.0100 0.1366   0.1366     2
## Willms1999           metf   plac -1.2000 0.3758   0.4125     3        *
## Willms1999           acar   plac -1.0000 0.4669   0.8242     3        *
##
## Number of treatment arms (by study):
##                     narms
## Alex1998                2
## Baksi2004               2
## Costa1997               2
## Davidson2007            2
## DeFronzo1995            2
## Derosa2004              2
## Garber2008              2
## Gonzalez-Ortiz2004      2
## Hanefeld2004            2
## Hermansen2007           2
## Johnston1994            2
## Johnston1998a           2
## Johnston1998b           2
## Kerenyi2004             2
## Kim2007                 2
## Kipnes2001              2
## Lewin2007               2
## Moulin2006              2
## Oyama2008               2
## Rosenstock2008          2
## Stucci1996              2
## Vongthavaravat2002      2
## Willms1999              3
## Wolffenbuttel1999       2
## Yang2003                2
## Zhu2003                 2
```

```
##
## Results (fixed effect model):
##
##                    treat1 treat2      MD             95%-CI     Q leverage
## DeFronzo1995         metf   plac -1.1141 [-1.2309; -0.9973] 30.89     0.18
## Lewin2007            metf   plac -1.1141 [-1.2309; -0.9973]  8.79     0.36
## Willms1999           acar   metf  0.2867 [ 0.0622;  0.5113]  0.05     0.09
## Davidson2007         plac   rosi  1.2018 [ 1.1084;  1.2953]  0.93     0.11
## Wolffenbuttel1999    plac   rosi  1.2018 [ 1.1084;  1.2953]  0.80     0.17
## Kipnes2001           piog   plac -1.0664 [-1.2151; -0.9178]  3.39     0.36
## Kerenyi2004          plac   rosi  1.2018 [ 1.1084;  1.2953] 16.05     0.20
## Hanefeld2004         metf   piog -0.0477 [-0.1845;  0.0891]  1.75     0.68
## Derosa2004           piog   rosi  0.1354 [-0.0249;  0.2957]  0.04     0.20
## Baksi2004            plac   rosi  1.2018 [ 1.1084;  1.2953]  0.94     0.22
## Rosenstock2008       plac   rosi  1.2018 [ 1.1084;  1.2953]  0.24     0.04
## Zhu2003              plac   rosi  1.2018 [ 1.1084;  1.2953]  3.37     0.09
## Yang2003             metf   rosi  0.0877 [-0.0449;  0.2203]  0.05     0.09
## Vongthavaravat2002   rosi   sulf -0.7623 [-0.9427; -0.5820]  9.29     0.41
## Oyama2008            acar   sulf -0.3879 [-0.6095; -0.1662]  0.01     0.53
## Costa1997            acar   plac -0.8274 [-1.0401; -0.6147]  0.04     0.57
## Hermansen2007        plac   sita  0.5700 [ 0.3170;  0.8230]  0.00     1.00
## Garber2008           plac   vild  0.7000 [ 0.4505;  0.9495]  0.00     1.00
## Alex1998             metf   sulf -0.6746 [-0.8482; -0.5011]  6.62     0.56
## Johnston1994         migl   plac -0.9439 [-1.1927; -0.6952]  1.23     0.48
## Johnston1998a        migl   plac -0.9439 [-1.1927; -0.6952]  4.35     0.32
## Kim2007              metf   rosi  0.0877 [-0.0449;  0.2203]  0.14     0.08
## Johnston1998b        migl   plac -0.9439 [-1.1927; -0.6952]  0.87     0.20
## Gonzalez-Ortiz2004   metf   plac -1.1141 [-1.2309; -0.9973]  2.69     0.02
## Stucci1996           benf   plac -0.9052 [-1.1543; -0.6561]  3.79     0.13
## Moulin2006           benf   plac -0.9052 [-1.1543; -0.6561]  0.59     0.87
## Willms1999           metf   plac -1.1141 [-1.2309; -0.9973]  0.04     0.02
## Willms1999           acar   plac -0.8274 [-1.0401; -0.6147]  0.04     0.02
##
## Number of studies: k = 26
## Number of treatments: n = 10
## Number of pairwise comparisons: m = 28
## Number of designs: d = 15
##
## Fixed effects model
##
## Treatment estimate (sm = 'MD', comparison: other treatments vs 'plac'):
##           MD             95%-CI
## acar -0.8274 [-1.0401; -0.6147]
## benf -0.9052 [-1.1543; -0.6561]
## metf -1.1141 [-1.2309; -0.9973]
## migl -0.9439 [-1.1927; -0.6952]
## piog -1.0664 [-1.2151; -0.9178]
```

```
## plac       .                  .
## rosi -1.2018 [-1.2953; -1.1084]
## sita -0.5700 [-0.8230; -0.3170]
## sulf -0.4395 [-0.6188; -0.2602]
## vild -0.7000 [-0.9495; -0.4505]
##
## Quantifying heterogeneity / inconsistency:
## tau^2 = 0.1087; I^2 = 81.4%
##
## Tests of heterogeneity (within designs) and inconsistency (between designs):
##                       Q d.f.  p-value
## Total             96.99   18 < 0.0001
## Within designs    74.46   11 < 0.0001
## Between designs   22.53    7   0.0021
```

There is plenty to see in this output, so let us go through it one by one.

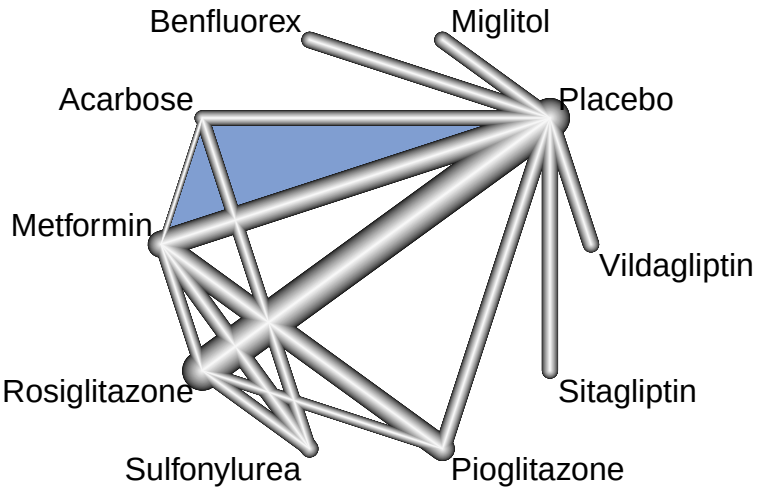
- The first thing we see are the calculated effect sizes for each comparison, with an asterisk signifying multiarm studies, for which the standard error had to be corrected.
- Next, we see an overview of the number of treatment arms in each included study. Again, it is the study `Willms2003` which stands out here because it contains three treatment arms, and thus multiple comparisons.
- The next table shows us the fitted values for each comparison in our network meta-analysis model. The `Q` column in this table is usually very interesting, because it tells us which comparison may contribute substantially to the overall inconsistency in our network. For example, we see that the `Q` value of `DeFronzo1995` is rather high, with $Q = 30.89$.
- We then get to the core of our network model: the `Treatment estimates`. As specified, the effects of all treatments are displayed in comparison to the placebo condition, which is why there is no effect shown for `plac`.
- We also see that the `heterogeneity/inconsistency` in our network model is very high, with $I^2 = 81.4\%$. This means that a random-effects model may be warranted, and that we should rerun the function setting `comb.random` to `TRUE`.
- The last part of the output (`Tests of heterogeneity`) breaks down the total heterogeneity in our network into heterogeneity attributable to within and between-design variation, respectively. The heterogeneity between treatment designs reflects the actual inconsistency in our network, and is highly significant ($p = 0.0021$). The ("conventional") within-designs heterogeneity is also highly significant. The information provided here is yet another sign that the random-effects model may be indicated for our network meta-analysis model.

11.2.3.3 Further examination of the network model

11.2.3.3.1 The Network Graph

Now that we have created our network meta-analysis model, we can proceed and plot our **network graph**. This can be done using te `netgraph()` function. This function has many parameters, which you can look up by typing `?netgraph()` into your console. Most of those arguments, however, have very sensible default values, so we do not have to specify much here. As a first step, we feed the function with our `netmeta` object `m.netmeta`. We can also specify the **order** in which the treatment nodes appear using the `seq` argument.

```
netgraph(m.netmeta,
         seq = c("plac", "migl", "benf", "acar", "metf",
                 "rosi", "sulf", "piog", "sita", "vild"))
```



This network graph transports several kinds of information.

- First, we see the overall **structure** of comparisons in our network, allowing us to understand which treatments were compared with each other in the original data.
- Second, we can see that the edges have a **different thickness**, which corresponds to **how often** we find this specific comparison in our network. We see that Rosiglitazone has been compared to Placebo in many, many trials
- We also see the one **multiarm** trial in our network, which is represented by the **blue triangle** in our network. This is the study `Willms2003`, which compared Meformin, Acarbose and Placebo.

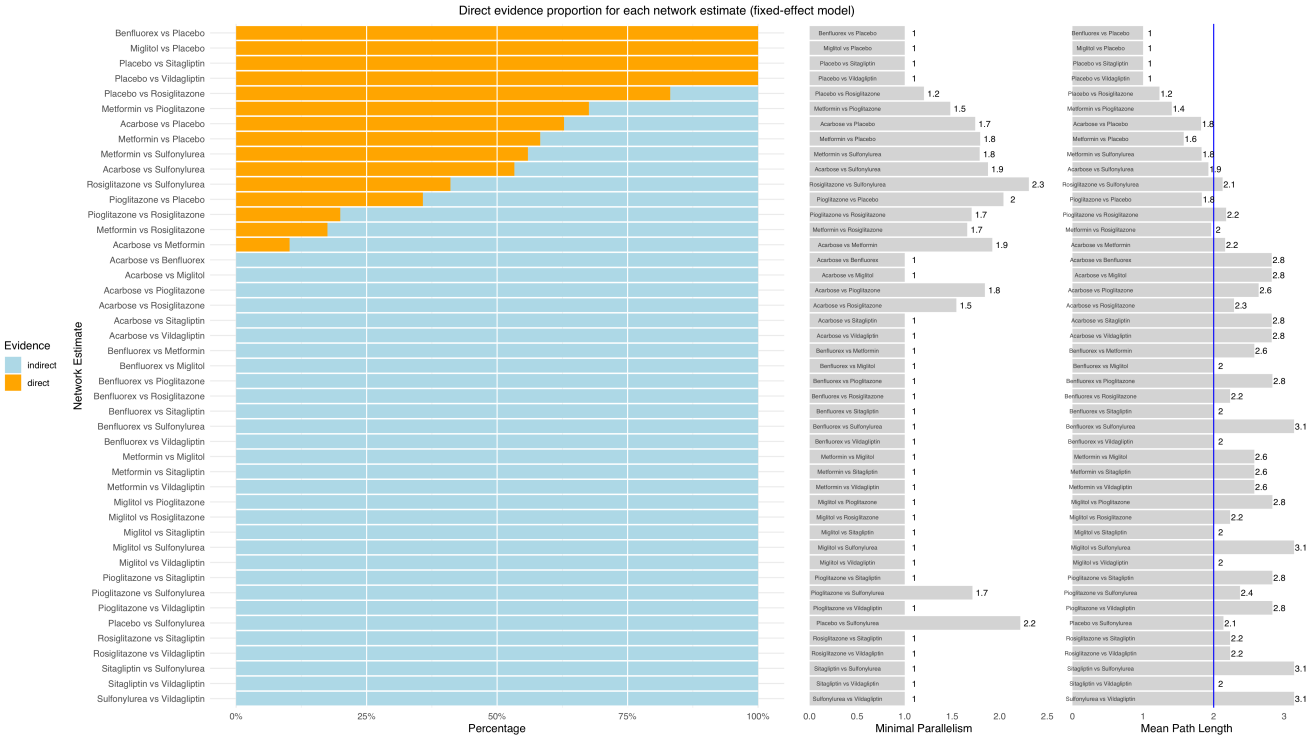
11.2.3.3.2 Visualising direct and indirect evidence

As a next step, we can also turn our attention to the **direct and indirect evidence** in our network by looking at the **proportion** of direct and indirect contributing to each comparison. We have prepared a function for this purpose for you, which is called `direct.evidence.plot()`. The `direct.evidence.plot` function is part of the `dmetar` package. If you have the package installed already, you have to load it into your library first.

```
library(dmetar)
```

If you don't want to use the `dmetar` package, you can find the source code for this function here. In this case, *R* doesn't know this function yet, so we have to let *R* learn it by **copying and pasting** the code **in its entirety** into the **console** in the bottom left pane of RStudio, and then hit **Enter** . The function then requires the `ggplot2`, `gridExtra`, and `scales` package to work. The function provides you with two outputs: a **plot** showing the percentage of direct and indirect evidence used for each estimated comparison, and a corresponding data frame containing the underlying data. The only thing the `direct.evidence.plot()` function needs as input is an object created by the `netmeta()` function.

```
direct.evidence.plot(x = m.netmeta)
```



As we can see, there are many estimates in our network model which had to be inferred by indirect evidence alone. The plot also provides us with two additional metrics: the **Minimal Parallelism** and the **Mean Path Length** of each comparison. According to König et al. (2013), lower values of minimal parallelism and $MeanPathLength > 2$ means that results for a specific comparison should be interpreted with caution.

11.2.3.3.3 Producing a matrix containing all treatment comparison estimates

Next, we can have a look at the **estimates** of our network for **all possible treatment combinations**. To do this, we can use the result **matrices** saved in our `netmeta` results object under `$TE.fixed` (if we use the fixed-effects model) or `$TE.random` (if we use the random-effects model). We will make a few preprocessing steps to make the matrix more readable. First, we extract the matrix from our `m.netmeta` object, and **round** the numbers in the matrix to **three digits**.

```
result.matrix <- m.netmeta$TE.fixed
result.matrix <- round(result.matrix, 3)
```

Given that one "triangle" in our matrix will **hold redundant information**, we replace the lower triangle with an empty value using this code:

```
result.matrix[lower.tri(result.matrix, diag = FALSE)] <- "."
```

We then get the following results:

	Acarbose	Benfluorex	Metformin	Miglitol	Pioglitazone	Placebo	Rosiglitazone	Sitagliptin	Sulfonylurea	Vildagliptin
Acarbose	0	0.078	0.287	0.117	0.239	-0.827	0.374	-0.257	-0.388	-0.127
Benfluorex	.	0	0.209	0.039	0.161	-0.905	0.297	-0.335	-0.466	-0.205
Metformin	.	.	0	-0.17	-0.048	-1.114	0.088	-0.544	-0.675	-0.414
Miglitol	.	.	.	0	0.123	-0.944	0.258	-0.374	-0.504	-0.244
Pioglitazone	.	.	.	.	0	-1.066	0.135	-0.496	-0.627	-0.366
Placebo	.	.	.	.	.	0	1.202	0.57	0.439	0.7
Rosiglitazone	.	.	.	.	.	.	0	-0.632	-0.762	-0.502
Sitagliptin	.	.	.	.	.	.	.	0	-0.131	0.13
Sulfonylurea	.	.	.	.	.	.	.	.	0	0.261
Vildagliptin	.	.	.	.	.	.	.	.	.	0

If we want to report these results in our research paper, a good idea might be to also include the **confidence intervals** for each comparison. These can be obtained the same way as above using the `$lower.fixed` and `$upper.fixed` (or `$lower.random` and `$upper.random`) matrices for the lower and upper confidence interval.

An extremely convenient way to export all estimated effect sizes is provided by the `netleague()` function. This function creates a matrix similar to the one above. Yet, in the matrix produced by this function, the **upper triangle** will display only the pooled effect sizes of the **direct comparisons** available in our network, like one would attain them if we had performed a conventional meta-analysis for each comparison. Because we do not have direct evidence for all comparisons, some fields in the upper triangle will remain empty. The **lower triangle** then contains the **network meta-analysis effect sizes** for each comparison. This matrix can be easily exported into a .csv file, which we can then use to report our network meta-analysis results in a table (for example in Microsoft WORD). The big plus of this function is that effect size estimates and confidence intervals will be displayed together in each cell; we only have too tell the function how the brackets for the confidence intervals should look like, and how many digits we want our estimates to have behind the comma. If i want to save the fixed-effects model results in a .csv (EXCEL) document called **"netleague.csv"**, i can use the following code. As always, we also need to feed the `netleague` function with our `netmeta` object.

```
netleague <- netleague(m.netmeta, bracket = "(", digits=2)
write.csv(netleague$fixed, "netleague.csv")
```

11.2.3.3.4 The treatment ranking

The most interesting question we may want to answer in a network meta-analysis is of course: **which intervention works the best?**. The `netrank()` function implemented in `netmeta` allows us to generate such a **ranking** of treatments from most to least beneficial. The `netrank()` function is again based on a frequentist treatment ranking method using **P-scores**. These P-scores measure the certainty that one treatment is better than another treatment, averaged over all competing treatments. The P-score has been shown to be equivalent to the **SUCRA** score (Rücker and Schwarzer, 2015), which we will describe in the chapter on bayesian network meta-analysis. The function needs our `netmeta` object as input. Additionally, we should specify the `small.values` parameter, which defines if smaller effect sizes in a comparison indicate a beneficial (`"good"`) or harmful (`"bad"`) effect. Let us have a look at the output for our example:

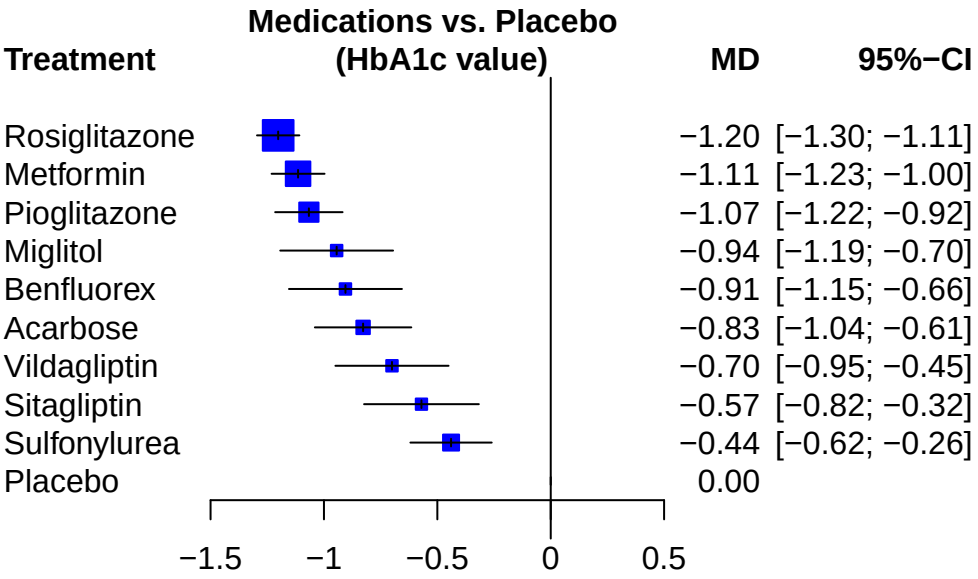
```
netrank(m.netmeta, small.values = "good")
```

```
##               P-score
## Rosiglitazone  0.9789
## Metformin      0.8513
## Pioglitazone   0.7686
```

```
## Miglitol       0.6200
## Benfluorex     0.5727
## Acarbose       0.4792
## Vildagliptin   0.3512
## Sitagliptin    0.2386
## Sulfonylurea   0.1395
## Placebo        0.0000
```

We see that the Rosiglitazone treatment has the highest P-score, indicating that this treatment may be particularly helpful. Conversely, Placebo has a P-score of zero, which seems to go along with our intuition that a placebo will not likely be the best treatment decision. Nevertheless, it should be noted that one should **never automatically conclude** that one treatment is the best by solely because it has the highest score (Mbuagbaw et al., 2017). A good way to also visualize the **uncertainty** in our network is to produce **network forest plots** with the **"weakest"** treatment as comparison. This can be done using the `forest` function, wich works very similar to the one of the `meta` package described in Chapter 5. We can specify the reference group for the forest plot using the `reference.group` argument.

```
forest(m.netmeta,
       reference.group = "placebo",
       sortvar = TE,
       xlim=c(-1.5,0.5),
       col.square = "blue",
       smlab = "Medications vs. Placebo \n (HbA1c value)")
```



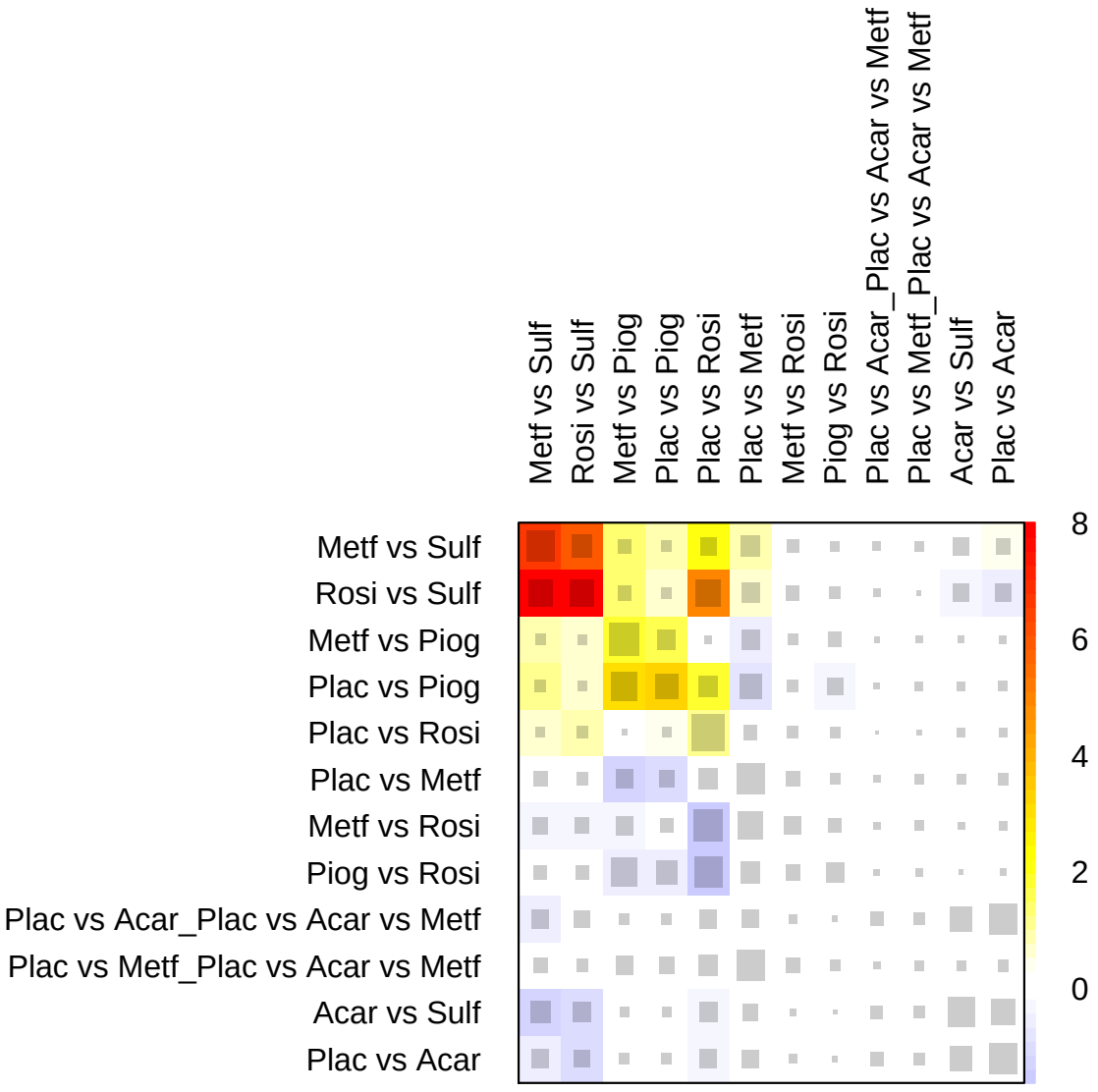
We now see that the results are not as clear as they seemed before; we see that there are indeed **several high-performing treatments** with overlapping confidence intervals. This means that we cannot come to a clear-cut decision which treatment is in fact the best, but instead see that there are indeed several treatments for which we can see a high effectiveness compared to placebo.

11.2.4 Evaluating the validity of our results

11.2.4.1 The Net Heat Plot

The `netmeta` package has an in-built function, `netheat()`, which allows us to produce a **net heat plot**. This net heat plot is very helpful to evaluate the inconsistency in our network model, and what contributes to it. We simply plug our `netmeta` object into the `netheat()` function to generate such a plot. The argument `nchar.trts` can be used to shorten our comparison labels to a specific number of characters. Let us try the function out now.

```
netheat(m.netmeta, nchar.trts = 4)
```



The function generates a quadratic matrix in which each element in a row is compared to all other elements in the columns. It is important here to mention that the rows and columns signify specific **designs**, not all m treatment comparisons in our network. Thus, we also have rows and columns for the multiarm study `Willms1999`, which had a design comparing **"Plac"**, **"Metf"** and **Acar**. Treatment comparison with only one kind of evidence (i.e. indirect or indirect evidence) are omitted in this plot, because we are interested in **cases of inconsistency** between direct and indirect evidence. Beyond that, the net heat plot has two important features (Schwarzer et al., 2015):

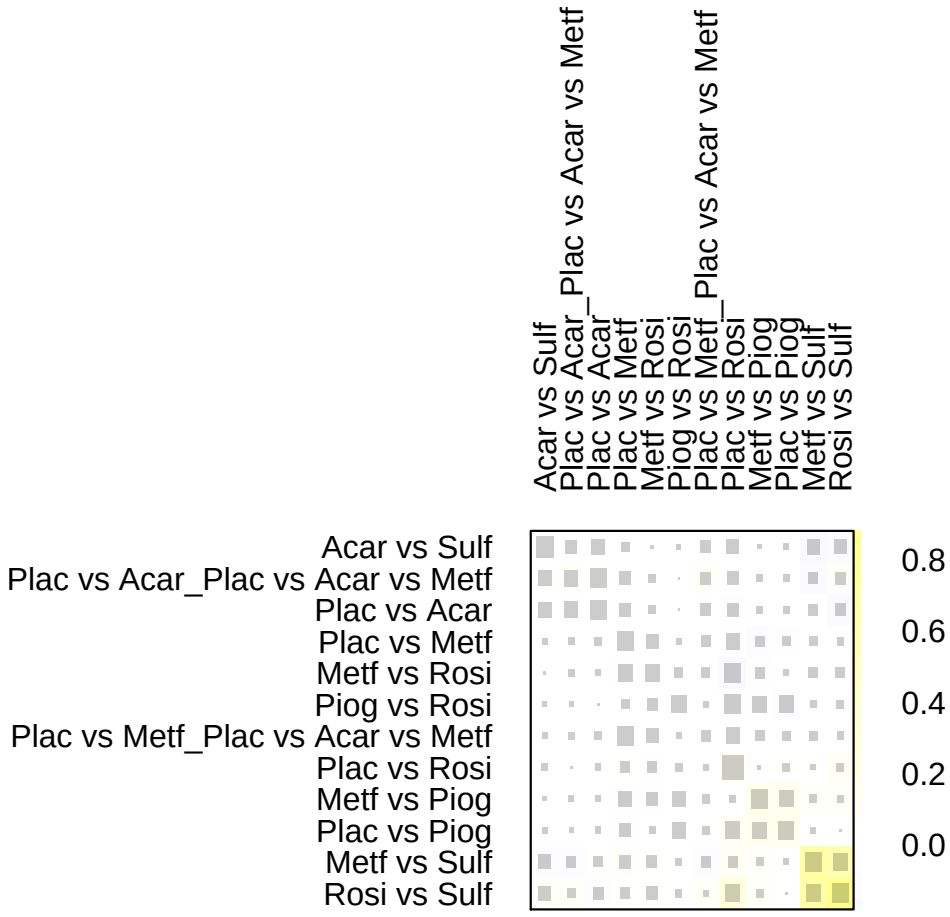
1. **Grey boxes**. The grey boxes for each comparison of designs signify how **important** a treatment comparison is

for the estimation of another treatment comparison. The bigger the box, the more important a comparison is. An easy way to analyse this is to gow through the rows of the plot one after another, and then to check for each row in which columns the grey boxes are the largest. A common finding is that the boxes are large in a row where the row comparison and the column comparison intersect, meaning that direct evidence was used. For example, a particularly big grey box can be seen at the intersection of the "Plac vs Rosi2 row and the"Plac vs Rosi" column.

2. **Colored backgrounds**. The colored backgrounds, ranging from blue to red, signify the inconsistency of the comparison in a row attributable to the design in a column. Inconsistent fields are displayed in the upper-left corner in red. For example, in the row for "Rosi vs Sulf", we see that the entry in column "Metf vs Sulf" is displayed in red. This means that the evidence contributed by "Metf vs Sulf" for the estimation of "Metf vs Sulf" is inconsistent with the other evidence.

We can now remind ourselves that these results are based on the **fixed-effects model**, which we used for our network analysis to begin with. From what we have seen so far, we can conclude that the fixed-effect model is not justified, because there is too much unexpected heterogeneity. We can thus check how the net heat plot changes when we assume a random-effects model by changing the `random` argument of the `netheat` function to TRUE. We see that this results in a substantial decrease of inconsistency in our network.

```
netheat(m.netmeta,
        nchar.trts = 4,
        random = TRUE)
```



11.2.4.2 Net splitting

Another method to check for consistency in our network is **net splitting**, also known as **node splitting**. This method splits our network estimates into the contribution of direct and indirect evidence, which allows us to control for inconsistency in specific comparisons in our network. To generate a net split and compare the results, we only have to plug our `netmeta` object into the `netsplit` function.

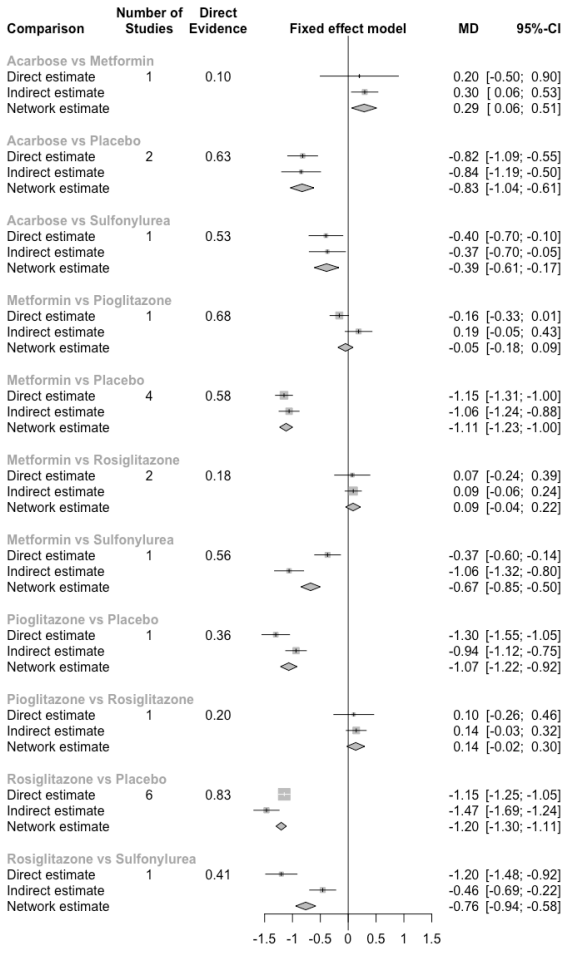
```
netsplit(m.netmeta)
```

```
## Back-calculation method to split direct and indirect evidence
##
## Fixed effect model:
##
##                     comparison k prop     nma  direct  indir.    Diff     z  p-value
##         Acarbose vs Benfluorex 0    0  0.0778       .  0.0778       .     .        .
##          Acarbose vs Metformin 1 0.10  0.2867  0.2000  0.2966 -0.0966 -0.26   0.7981
##           Acarbose vs Miglitol 0    0  0.1166       .  0.1166       .     .        .
##       Acarbose vs Pioglitazone 0    0  0.2391       .  0.2391       .     .        .
##            Acarbose vs Placebo 2 0.63 -0.8274 -0.8172 -0.8446  0.0274  0.12   0.9030
##      Acarbose vs Rosiglitazone 0    0  0.3745       .  0.3745       .     .        .
##        Acarbose vs Sitagliptin 0    0 -0.2574       . -0.2574       .     .        .
##       Acarbose vs Sulfonylurea 1 0.53 -0.3879 -0.4000 -0.3740 -0.0260 -0.11   0.9088
##       Acarbose vs Vildagliptin 0    0 -0.1274       . -0.1274       .     .        .
##        Benfluorex vs Metformin 0    0  0.2089       .  0.2089       .     .        .
##         Benfluorex vs Miglitol 0    0  0.0387       .  0.0387       .     .        .
##     Benfluorex vs Pioglitazone 0    0  0.1612       .  0.1612       .     .        .
##          Benfluorex vs Placebo 2 1.00 -0.9052 -0.9052       .       .     .        .
##    Benfluorex vs Rosiglitazone 0    0  0.2967       .  0.2967       .     .        .
##      Benfluorex vs Sitagliptin 0    0 -0.3352       . -0.3352       .     .        .
##     Benfluorex vs Sulfonylurea 0    0 -0.4657       . -0.4657       .     .        .
##     Benfluorex vs Vildagliptin 0    0 -0.2052       . -0.2052       .     .        .
##          Metformin vs Miglitol 0    0 -0.1702       . -0.1702       .     .        .
##      Metformin vs Pioglitazone 1 0.68 -0.0477 -0.1600  0.1866 -0.3466 -2.32   0.0201
##           Metformin vs Placebo 4 0.58 -1.1141 -1.1523 -1.0608 -0.0915 -0.76   0.4489
##     Metformin vs Rosiglitazone 2 0.18  0.0877  0.0731  0.0908 -0.0178 -0.10   0.9204
##       Metformin vs Sitagliptin 0    0 -0.5441       . -0.5441       .     .        .
##      Metformin vs Sulfonylurea 1 0.56 -0.6746 -0.3700 -1.0611  0.6911  3.88   0.0001
##      Metformin vs Vildagliptin 0    0 -0.4141       . -0.4141       .     .        .
##       Miglitol vs Pioglitazone 0    0  0.1225       .  0.1225       .     .        .
##            Miglitol vs Placebo 3 1.00 -0.9439 -0.9439       .       .     .        .
##      Miglitol vs Rosiglitazone 0    0  0.2579       .  0.2579       .     .        .
##        Miglitol vs Sitagliptin 0    0 -0.3739       . -0.3739       .     .        .
##       Miglitol vs Sulfonylurea 0    0 -0.5044       . -0.5044       .     .        .
##       Miglitol vs Vildagliptin 0    0 -0.2439       . -0.2439       .     .        .
##        Pioglitazone vs Placebo 1 0.36 -1.0664 -1.3000 -0.9363 -0.3637 -2.30   0.0215
##  Pioglitazone vs Rosiglitazone 1 0.20  0.1354  0.1000  0.1442 -0.0442 -0.22   0.8289
##    Pioglitazone vs Sitagliptin 0    0 -0.4964       . -0.4964       .     .        .
```

```
##   Pioglitazone vs Sulfonylurea 0    0 -0.6269       . -0.6269       .     .        .
##   Pioglitazone vs Vildagliptin 0    0 -0.3664       . -0.3664       .     .        .
##       Rosiglitazone vs Placebo 6 0.83 -1.2018 -1.1483 -1.4665  0.3182  2.50   0.0125
##         Sitagliptin vs Placebo 1 1.00 -0.5700 -0.5700       .       .     .        .
##        Sulfonylurea vs Placebo 0    0 -0.4395       . -0.4395       .     .        .
##        Vildagliptin vs Placebo 1 1.00 -0.7000 -0.7000       .       .     .        .
##   Rosiglitazone vs Sitagliptin 0    0 -0.6318       . -0.6318       .     .        .
##  Rosiglitazone vs Sulfonylurea 1 0.41 -0.7623 -1.2000 -0.4575 -0.7425 -3.97 < 0.0001
##  Rosiglitazone vs Vildagliptin 0    0 -0.5018       . -0.5018       .     .        .
##    Sitagliptin vs Sulfonylurea 0    0 -0.1305       . -0.1305       .     .        .
##    Sitagliptin vs Vildagliptin 0    0  0.1300       .  0.1300       .     .        .
##   Sulfonylurea vs Vildagliptin 0    0  0.2605       .  0.2605       .     .        .
##
## Legend:
##  comparison - Treatment comparison
##  k          - Number of studies providing direct evidence
##  prop       - Direct evidence proportion
##  nma        - Estimated treatment effect (MD) in network meta-analysis
##  direct     - Estimated treatment effect (MD) derived from direct evidence
##  indir.     - Estimated treatment effect (MD) derived from indirect evidence
##  Diff       - Difference between direct and indirect treatment estimates
##  z          - z-value of test for disagreement (direct versus indirect)
##  p-value    - p-value of test for disagreement (direct versus indirect)
```

The important information here is in the p-value column. If the value in this column is $p < 0.05$, there is a **significant disagreement** (inconsistency) between the direct and indirect estimate. We see in the output that there are indeed a few comparisons showing significant inconsistency between direct and indirect evidence when using the fixed-effects model. A good way to visualize the netsplit results is through a **forest plot** displaying all comparisons for which there is both direct and indirect evidence.

```
forest(netsplit(m.netmeta))
```



11.2.4.3 Comparison-adjusted Funnel Plots

Assessing the publication bias of a network meta-analysis in its aggregated form is difficult. Analyzing so-called **comparison-adjusted funnel plot** (see Chapter 9.1 for the general idea behind funnel plots) has been proposed to evaluate the risk of publication bias under specific circumstances (Salanti et al., 2014). Comparison-adjusted funnel plots allow to assess potential publication bias if we have an *a priori* hypothesis concerning which mechanism may underlie publication bias.

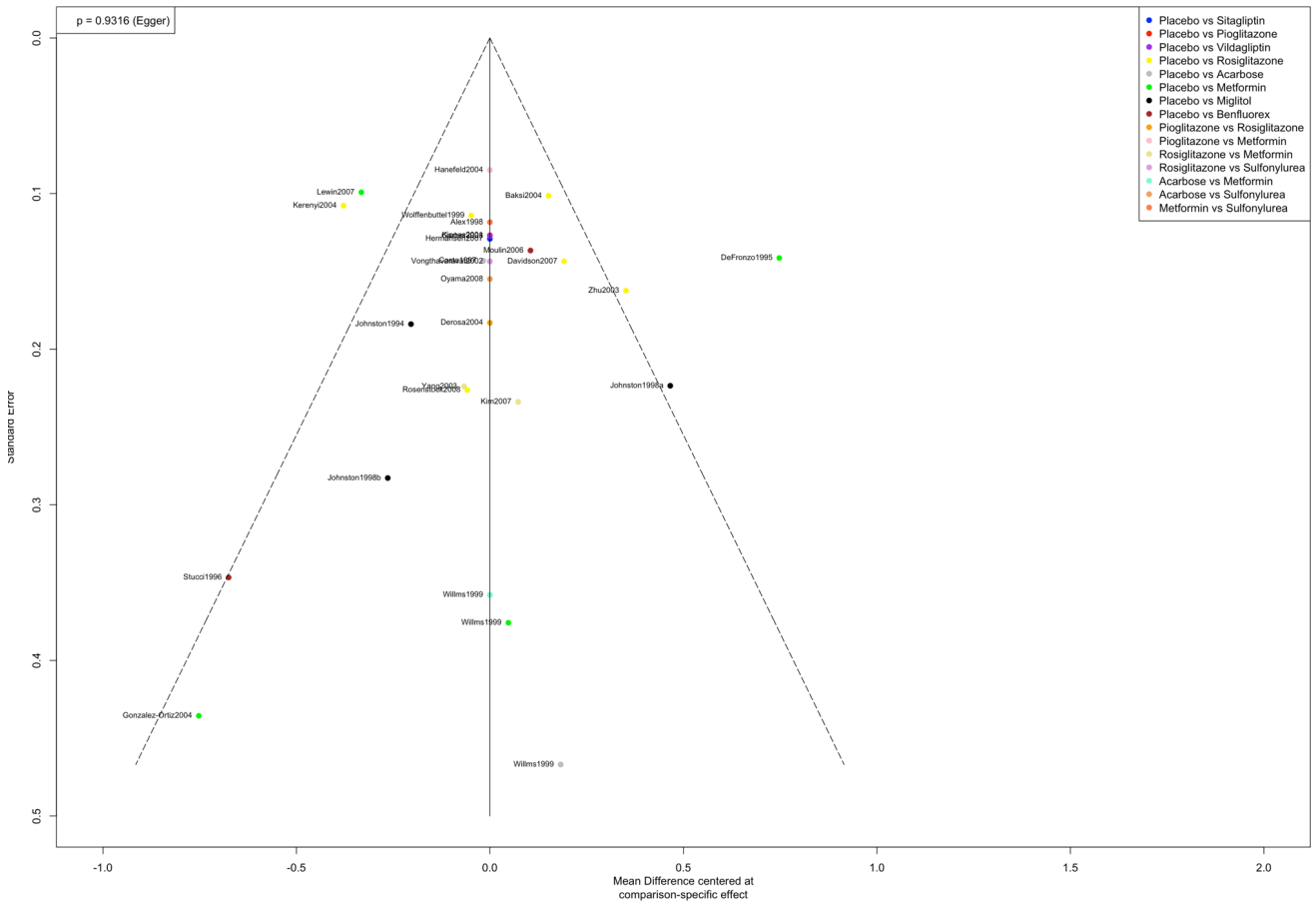
For example, we may have the hypothesis that publication bias exists in our data because studies which find that a **new form of treatment is superior** to an already known treatment have a **higher chance** of getting published, even if they have a small sample size, and thus a larger standard error of their effect size estimate. This is a pretty logical assumption, because it seems sensible that scientific journals may be particularly interested in publishing "novel" findings, such as that there might be a better treatment format for some kind of condition. The `funnel()` function in `netmeta` makes it easy to generate comparison-adjusted funnel plots to test such hypotheses. Here are the most important parameters of the function we may want to specify:

Code	Description
x	Here, we specify the name of our netmeta object.

(continued)

Code	Description
order	This paramter specifies the order of the hypothesized publication bias mechanism. We simply have to concenate ('c()') the names of all treatments in our network here, and order them according to our hypothesis. For example, if we want to test for publication bias favoring 'newer' treatments, we would insert the names of all treatments here, starting from the oldest treatment and ending with the most novel type of treatment or intervention.
pch	This lets us specify the symbol used for the studies in the funnel plot. Setting this to '19' gives us simple dots.
col	Using this parameter, we can specifiy a concatenated selection of colors we want to use to distinguish the different comparison. The number of colors we specify here must be the same as the number of comparisons in our funnel plot. A complete list of possible color that R can use for plotting can be found here: http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf
linreg	If we set this to TRUE, Egger's test of the intercept for funnel plot asymmetry will be conducted, and its p-value will be displayed in the plot.
xlim, ylim	These parameters let us specify the scope of the x and y axis
studlab	If we set this to TRUE, study labels will be printed in the plot along with each point
cex.studlab	This parameter lets us control the size of the study label. A value smaller than 1 means that the study labels will become smaller than by default

```
funnel <- funnel(m.netmeta,
                 order = c("placebo", "sitagliptin", "pioglitazone",
                           "vildagliptin", "rosiglitazone",
                           "acarbose", "metformin", "miglitol",
                           "benfluorex", "sulfonylurea"),
                 pch = 19,
                 col = c("blue", "red", "purple", "yellow", "grey",
                         "green", "black", "brown", "orange", "pink",
                         "khaki", "plum", "aquamarine", "sandybrown", "coral"),
                 linreg = TRUE,
                 xlim = c(-1, 2),
                 ylim = c(0.5, 0),
                 studlab = TRUE,
                 cex.studlab = 0.7)
```



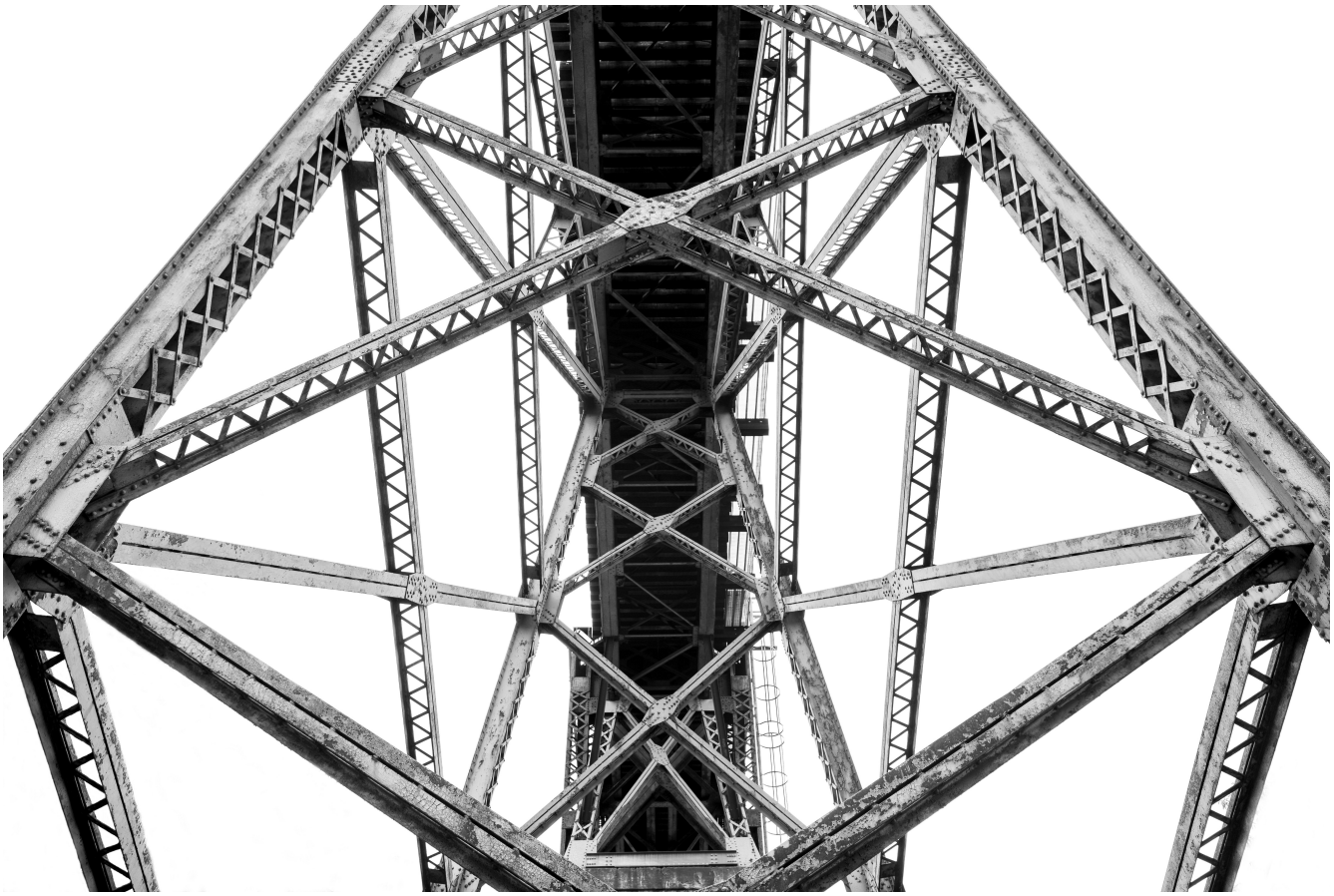
If our hypothesis is true, we would expect that studies with a **small sample**, and thus a higher **standard error** would by asymmetrically distributed around the zero line in our funnel plot. This is because we would expect that small studies comparing a novel treatment to an older one, yet finding that the new treatment is not better, are less likely to get published. In our plot, and from the p-value for Egger's Test ($p = 0.93$), however, we see that such funnel asymmetry is **not present**. Therefore, we cannot say that publication bias is present in our network because of "innovative" treatments with favorable trial effect being more likely to get published.

11.2.5 Summary

This has been a long chapter, and there have been many things we have covered until now. We have showed you the core assumptions behind the statistical modeling approach of the `netmeta()` package, described how to fit your network meta-analysis model, visualize and interpret the results, and evaluate the validity of your findings. We would like to stress that it is crucially important in network meta-analysis not base your clinical decision-making on one single test or metric, but to **explore your model and its results with open eyes**, checking the patterns you find for their consistency and robustness, and to assess the uncertainty concerning some of its parameters.

In the next subchapter, we will address how to perform network meta-analysis using a bayesian hierarchical framework. Although the statistical procedures behind this approach vary considerably from what is described here at times, both methods essentially try to answer the same question. Process-wise, the analysis "pipeline" is also very similar most of the times. Time to go bayesian!

11.3 Bayesian Network Meta-Analysis



In the following, we will describe how to perform a network meta-analysis based on a **bayesian hierarchical** framework. The R package we will use to do this is the `gemtc` package (van Valkenhoef et al., 2012). But first, let us consider the idea behind bayesian in inference in general, and the bayesian hierarchical model for network meta-analysis in particular.

11.3.1 Bayesian Inference

Besides the frequentist framework, bayesian inference is another **important strand of inference statistics**. Although the frequentist approach is arguably used more often in most research fields, bayesian statistics is actually older, and despite being increasingly picked up by researchers in recent years (Marsman et al., 2017), has never really been "gone" (McGrayne, 2011). The central idea of bayesian statistics revolves around **Bayes' Theorem**, first formulated by **Thomas Bayes** (1701-1761). Bayesian statistics differs from frequentism because it also incorporates "subjective" **prior knowledge** into statistical inference. In particular, Bayes' theorem allows us to estimate the probability of an event A, given that we already know that another event B has occurred, or is the case. This is a **conditional probability**, which can be expressed like this in mathematical notation: $P(A|B)$. Bayes' theorem then looks like this:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

The two probabilities in the upper part of the fraction to the right have their own names. First, the $P(B|A)$ part

is (essentially) known as the **likelihood**; in our case, this is the probability of event B (which has already occurred), given that A is in fact the case, or occurs (Etz, 2018). $P(A)$ is the **"prior"** probability of A occurring. $P(A|B)$, lastly, is the **"posterior"** probability. Given that $P(B)$ is a fixed constant, the formula above is often shortened to this form:

$$P(A|B) \propto P(B|A) \times P(A)$$

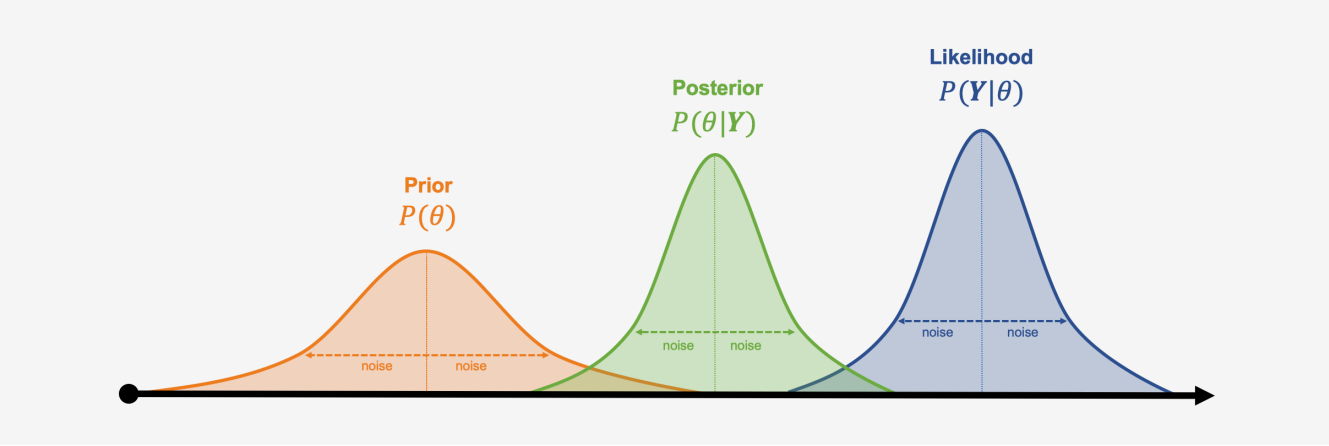
Where the $\propto$ sign means that now that we have discarded the denominator of the fraction, the probability on the left remains at least **"proportional"** to the part on the right with changing values (Shim et al., 2019). We may understand Bayes' Theorem better if we think of the formula as a **process**, beginning on the right side of the equation. We simply combine the **prior** information we have on the probability of A occuring with the **likelihood** of B given that A actually occurs to produce our **posterior**, or **adapted**, probability of A. The crucial point here is that we can produce a **"better"**, i.e. posterior, estimate of A's probability when we take our previous knowledge of the context into account. This context knowledge is our **prior** knowledge on the probability of A.

Bayes' Theorem is often explained in the way we did above, with A and B standing for specific events. The example may become less abstract however, if we take into account that A and B can also represent **distributions** of a variable. Let us now assume that $P(A)$ follows in fact a distribution. This distribution is characterized by a set of parameters which we denote by θ. For example, if our distribution follows a gaussian normal distribution, it can be described by its mean μ and variance σ^2. The θ parameters is what we actually want to estimate. Now, let us also assume that, instead of B, we have collected **actual data** through which we want to estimate θ. We store our observed data in a vector $\mathbf{Y}$; of course, our observed data also follows a distribution, $P(\mathbf{Y})$. The formula now looks like this:

$$P(\theta|\mathbf{Y}) \propto P(\mathbf{Y}|\theta) \times P(\theta)$$

In this equation, we now have $P(\theta)$, the **prior** distribution of θ, which we simply define *a priori*, either based on our previous knowledge, or even our intuition on what the θ values may look like. Together with the **likelihood distribution** $P(\mathbf{Y}|\theta)$ of our data given the true parameters θ, we can estimate the **posterior distribution** $P(\theta|\mathbf{Y})$, which represents what we **think the θ parameters look like if we take both the observed data and our prior knowledge into account**. As can be seen in the formula, the **posterior** is still a distribution, not an estimated "true" value, meaning that even the results of bayesian inference are still **probabilistic**, and subjective in the sense that they represent our **belief** in the actual parameter values. Bayesian inference thus also means that we do not calculate confidence intervals around our estimates, but **credibility intervals** (CrI).

Here is a visualisation of the three distributions we described above, and how they might look like in a specific example:



A big asset of bayesian approaches is that the distributions do not have to follow a **bell curve distribution** like the ones in our visualization; any other kind of (more complex) distribution can be modeled. A disadvantage of bayesian inference, however, is that generating the (joint) distribution parameters from our collected data can be very computationally intensive; special **Markov Chain Monte Carlo** simulation procedures, such as the **Gibbs sampling** algorithm have thus been developed to generate posterior distributions. Markov Chain Monte Carlo is also used in the `gemtc` package to build our bayesian network meta-analysis model (van Valkenhoef et al., 2012).

11.3.2 The Network Meta-Analysis Model

We will now formulate the **bayesian hierarchical** model underlying the `gemtc` package. We will start by defining the model for a conventional **pairwise meta-analysis**. This definition of the meta-analysis model is equivalent with the one provided in Chapter 4.2, where we discuss the random-effects model. What we will describe is simply the **bayesian** way to conceptualize meta-analysis, and we use this other formulation so that you can more easily read up on the statistical background of bayesian network meta-analysis in other literature. On the other hand, the bayesian definition of pairwise meta-analysis is also highly important because it is directly applicable to network meta-analyses, without any further extension (Dias et al., 2013). The model used in the `gemtc` package is also called a **bayesian hierarchical model** (Efthimiou et al., 2016). There is nothing majorly complicated about the word **hierarchical** here; indeed we already described in Chapter 12 that even the most simple meta-analysis model presupposes a hierarchical, or **"multilevel"** structure.

11.3.2.1 The fixed-effect model

Suppose that we want to conduct a conventional pairwise meta-analysis. We have included K studies, and have collect an effect size Y_k for each of our studies $k = 1, ..., K$. We can then define the **fixed-effect model** like this:

$$Y_k \sim \mathcal{N}(\delta, s_k^2)$$

This formula expresses the **likelihood** (the $P(\mathbf{Y}|\theta)$ part from above) of our effect sizes, assuming that they follow a normal distribution. We assume that each effect size is a simply a draw from the distribution of the **one true effect size** δ, which has the variance s_k^2. In the fixed-effect model, we assume that there is **one true effect** underlying all the effect sizes we observe, so δ stays the same for different studies k and their corresponding observed effect sizes Y_k. The interesting part in the bayesian model is that while the true effect δ is unkown, we can still define a **prior** distribution defining how we think δ may look like. For example, we could assume a prior following a normal distribution, $\delta \sim \mathcal{N}(0, s^2)$, were we specify s^2. The `gemtc` package by default uses so-called **uninformative priors**, which are prior distributions with a very large variance. This is done so that our prior "beliefs" do not have a big impact on the posterior results, and we only let the actual observed data "speak".

11.3.2.2 The random-effects model

We can easily extend this formula for the **random-effects model**:

$$Y_k \sim \mathcal{N}(\delta_k, s_k^2)$$

Not much has changed here, except that now using the random-effects model, we do not assume that each study is an estimator of the one true effect size δ, but that there are **study-specific** true effects δ_k estimated by each effect size Y_k. Furthermore, these study-specific true effects are again part of an **overarching distribution** of true effect sizes, which is defined by its mean d and variance τ^2, our **between-study heterogeneity**.

$$\delta_k \sim \mathcal{N}(d, \tau^2)$$

In the bayesian model, we also give a (uninformative) prior distribution to both d and τ^2.

11.3.2.3 Application to network meta-analysis

Now that we have covered how a bayesian meta-analysis model can be formulated for pairwise meta-analysis, we can start to apply this model to the network meta-analysis context. The random-effect model formula above can be directly used to achieve this - we only have to **conceptualize** our model parameters a little differently. Now that we also want to make clear that comparisons in network meta-analysis can consist of **different treatments** being compared, we can denote a single effect size we have obtained from one study k as Y_{kab}, which signifies that for this effect size in study k, treatment a was compared to treatment b. If we apply this new notation to the entire formula, we get this:

$$Y_{kab} \sim \mathcal{N}(\delta_{kab}, s_k^2)$$
$$\delta_{kab} \sim \mathcal{N}(d_{ab}, \tau^2)$$

We see that the formulae still follow the same concept as before, but now we assume that our study-specific "true" effect for the comparison of a and b is assumed to be part of an overarching distribution of true effect sizes with mean d_{ab}. This mean true effect size d_{ab} of the $a - b$ comparison is in fact **the result of subtracting** $d_{1a} - d_{1b}$, where d_{1a} is the effect of treatment a compared to some predefined **reference treatment** 1, and d_{1b} is the effect of treatment b compared to the same reference treatment. In the bayesian model, these effects compared to the reference group are also given a prior distribution.

As we have already mentioned in the previous chapter on frequentist network meta-analysis, combining **multiarm studies** into our network model is problematic because these treatment arms are correlated. In the bayesian framework, this issue is solved by assuming that the effects of the i treatments contained in the $i + 1$ treatment arms of some multiarm study k stem from a **multivariate** normal distribution:

$$\begin{pmatrix} \delta_{kab1} \\ \delta_{kab1} \\ \delta_{kab3} \\ \vdots \\ \delta_{kabi} \end{pmatrix} = \mathcal{N}\left(\begin{pmatrix} d_{ab1} \\ d_{ab1} \\ d_{ab3} \\ \vdots \\ d_{abi} \end{pmatrix}, \begin{pmatrix} \tau^2 & \tau^2/2 & \tau^2/2 & \cdots & \tau^2/2 \\ \tau^2/2 & \tau^2 & \tau^2/2 & \cdots & \tau^2/2 \\ \tau^2/2 & \tau^2/2 & \tau^2 & \cdots & \tau^2/2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \tau^2/2 & \tau^2/2 & \tau^2/2 & \cdots & \tau^2 \end{pmatrix} \right) \tag{11.1}$$

11.3.3 Performing a Network Meta-Analysis using the `gemtc` package

Now, let us start using the `gemtc` package to perform our first **bayesian network meta-analysis**. As always, we have to first **install the package**, and then load it from our library.

```
install.packages("gemtc")
library(gemtc)
```

The gemtc also requires the rjags package, which is used for the **Gibbs sampling** procedure we described above, to function properly. However, before we install and load this package, we also have to install a software called **JAGS** (short for "Just Another Gibbs Sampler") on our computer. The software is available for both Windows and Mac/Unix, and you can download it for free here.

After this is done, we can also install and load the rjags package.

```
install.packages("rjags")
library(rjags)
```

11.3.3.1 Data preprocessing

Now, let us restructure our data so that functions of the gemtc package can process it. Like in the chapter on frequentist network meta-analysis, we will use the data by Senn2013 comparing different medications for diabetic patients for our analysis.

However, we had to tweak the structure of our data little to conform with gemtcs requirements. The original dataset contains the **Mean Difference** (TE) between two treatments, with each **row representing one comparison**. However, for the gemtc package, we need to reshape our data so that **each row represents a single treatment arm**. Furthermore, we have to specify which treatment arm was the **reference group** for the effect size calculated for a comparison by giving it the label **"NA"**. Here is how our reshaped dataset looks like, which we again called *data*:

study	treatment	diff	std.err
Alex1998	Metformin	-0.37	0.1184
Alex1998	Sulfonylurea	NA	NA
Baksi2004	Rosiglitazone	-1.30	0.1014
Baksi2004	Placebo	NA	NA
Costa1997	Acarbose	-0.80	0.1432
Costa1997	Placebo	NA	NA
Davidson2007	Rosiglitazone	-1.34	0.1435
Davidson2007	Placebo	NA	NA
DeFronzo1995	Metformin	-1.90	0.1414
DeFronzo1995	Placebo	NA	NA
Derosa2004	Pioglitazone	0.10	0.1831
Derosa2004	Rosiglitazone	NA	NA
Garber2008	Vildagliptin	-0.70	0.1273
Garber2008	Placebo	NA	NA
Gonzalez-Ortiz2004	Metformin	-0.40	0.4356
Gonzalez-Ortiz2004	Placebo	NA	NA
Hanefeld2004	Pioglitazone	0.16	0.0849
Hanefeld2004	Metformin	NA	NA
Hermansen2007	Sitagliptin	-0.57	0.1291
Hermansen2007	Placebo	NA	NA
Johnston1994	Miglitol	-0.74	0.1839
Johnston1994	Placebo	NA	NA
Johnston1998a	Miglitol	-1.41	0.2235
Johnston1998a	Placebo	NA	NA
Johnston1998b	Miglitol	-0.68	0.2828
Johnston1998b	Placebo	NA	NA
Kerenyi2004	Rosiglitazone	-0.77	0.1078

(continued)

study	treatment	diff	std.err
Kerenyi2004	Placebo	NA	NA
Kim2007	Rosiglitazone	0.00	0.2339
Kim2007	Metformin	NA	NA
Kipnes2001	Pioglitazone	-1.30	0.1268
Kipnes2001	Placebo	NA	NA
Lewin2007	Metformin	-0.82	0.0992
Lewin2007	Placebo	NA	NA
Moulin2006	Benfluorex	-1.01	0.1366
Moulin2006	Placebo	NA	NA
Oyama2008	Acarbose	-0.40	0.1549
Oyama2008	Sulfonylurea	NA	NA
Rosenstock2008	Rosiglitazone	-1.09	0.2263
Rosenstock2008	Placebo	NA	NA
Stucci1996	Benfluorex	-0.23	0.3467
Stucci1996	Placebo	NA	NA
Vongthavaravat2002	Rosiglitazone	-1.20	0.1436
Vongthavaravat2002	Sulfonylurea	NA	NA
Willms1999	Metformin	-1.20	0.3758
Willms1999	Acarbose	-1.00	0.4669
Willms1999	Placebo	NA	0.4002
Wolffenbuttel1999	Rosiglitazone	-1.10	0.1141
Wolffenbuttel1999	Placebo	NA	NA
Yang2003	Rosiglitazone	-0.14	0.2239
Yang2003	Metformin	NA	NA
Zhu2003	Rosiglitazone	-1.50	0.1624
Zhu2003	Placebo	NA	NA

The `gemtc` package also requires that the **columns** of our dataframe are labeled correctly. If we have precalculated effect sizes for continuous outcomes (such as the **Mean Difference** or **Standardized Mean Difference**), our dataset has to contain these columns:

- **study**. This columns contains the (unique) label for each study included in our network, equivalent to the "studlab" column used in `netmeta`.
- **treatment**. This column contains the label for the treatment.
- **diff**. This column contains the effect size (e.g., the mean difference calculated for a comparison). Importantly, the diff column contains "NA", a missing, in the row of the reference treatment of a specific study. The row of the treatment to which the reference treatment was compared to then holds the actual effect size calculated for this comparison. Also keep in mind that the reference category is defined study-wise, not comparison-wise. This means that in multiarm studies, we still have only one reference treatment to which the other treatments are compared.
- **std.err**. This column contains the standard error of the effect sizes, and follows the same pattern.

Please note that **other data formats** are also possible, for example for binary outcome or other raw effect size data. The way your dataset needs to be structured in such cases is described here and here.

11.3.3.2 Network graph

Now that we have our data ready, we feed it to the `mtc.network()` function to generate an element of class `mtc.network`, which we can use for later modeling steps. Because we are using **precalculated effect size data**, we have to specify our dataset with the `data.re` parameter. For raw effect size data (e.g. Mean, SD, N), we would have used the `data.ab` argument. We save the result as `network`.

```
network <- mtc.network(data.re = data)
```

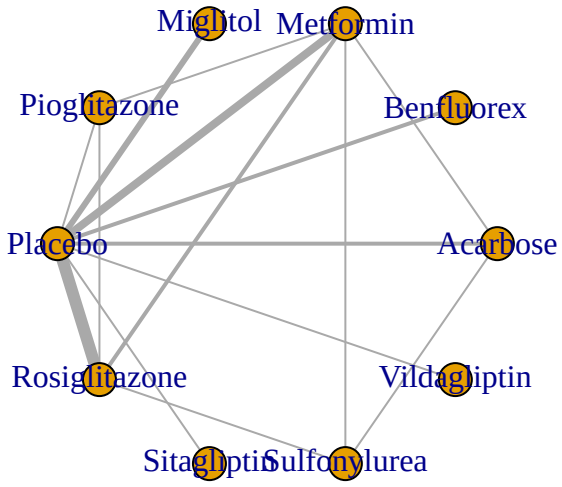
Plugging the resulting `network` object into the `summary()` function already provides us with some interesting information about the structure of our network.

```
summary(network)
```

```
## $Description
## [1] "MTC dataset: Network"
##
## $`Studies per treatment`
##      Acarbose    Benfluorex     Metformin      Miglitol  Pioglitazone
##             3             2             8             3             3
##       Placebo Rosiglitazone   Sitagliptin  Sulfonylurea  Vildagliptin
##            19            10             1             3             1
##
## $`Number of n-arm studies`
## 2-arm 3-arm
##    25     1
##
## $`Studies per treatment comparison`
##               t1            t2 nr
## 1       Acarbose     Metformin  1
## 2       Acarbose       Placebo  2
## 3       Acarbose  Sulfonylurea  1
## 4     Benfluorex       Placebo  2
## 5      Metformin  Pioglitazone  1
## 6      Metformin       Placebo  4
## 7      Metformin Rosiglitazone  2
## 8      Metformin  Sulfonylurea  1
## 9       Miglitol       Placebo  3
## 10  Pioglitazone       Placebo  1
## 11  Pioglitazone Rosiglitazone  1
## 12       Placebo Rosiglitazone  6
## 13       Placebo   Sitagliptin  1
## 14       Placebo  Vildagliptin  1
## 15 Rosiglitazone  Sulfonylurea  1
```

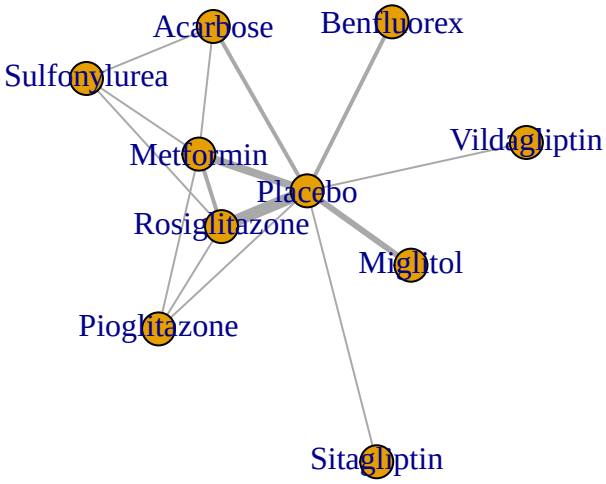
We can also use the `plot()` function to generate our network plot. Like the network generated by the `netmeta` package, the network's edge thickness corresponds with the number of studies we have for that comparison.

```
plot(network)
```



As an alternative, we can also check if we can create a better visualisation of our network using the **Fruchterman-Reingold** algorithm. This algorithm comes with some inherent **randomness**, meaning that we can rerun the code below several times to get different results (make sure to have the `igraph` package installed and loaded for this to work).

```
plot(network, layout=layout.fruchterman.reingold)
```



11.3.3.3 Model compilation

Using our `mtc.network()` object `network`, we can now start to specify and compile our model. The great thing about the *gemtc* package is that is **automates** most parts of the bayesian inference process, for example by choosing adequate prior distributions for all parameters in our model. Thus, there are only a few parameters we have to specify when compiling our model using the `mtc.model()` function. First, we have to specify the `mtc.network()` object we have created before. Furthermore, we have to specify if we want to use a **random- or fixed effects model** using the `linearModel` argument. Given that our previous frequentist analysis indicated substantial heterogeneity and inconsistency, we will set `linearModel = "random"`. Lastly, we also have to specify the **number of Markov chains** we want to use. A value between 3 and 4 is sensible here, and we will chosse `n.chain = 4`. We call our compiled model `model`.

```
model <- mtc.model(network,
                   linearModel = "random",
```

```
                    n.chain = 4)
```

11.3.3.4 Markov Chain Monte Carlo simulation

Now we come to the crucial part of our analysis: the **Markov Chain Monte Carlo** simulation, which will allows to estimate the posterior distributions of our parameters, and thus generate the results of our network meta-analysis. There are two important desiderata we want to achieve for our model using this procedure:

1. We want that the **first few runs** of the Markov Chain Monte Carlo simulations, which will likely produce inadequate results, do not have a large impact on the whole simulation results.
2. We want the Markov Chain Monte Carlo process to **run long enough** for us to reach **accurate estimates** of our parameters (i.e., to reach convergence).

To address these points, we split the number of times the Markov Chain Monte Carlo algorithm iterates over our data to infer the model results into **two phases**: first, we define the number of **burn-in** iterations (`n.adapt`), the results of which are then discarded. For the following phase, we can specify the number of **actual simulation iterations** (`n.iter`), which we use for the model inference. Given that we often simulate many, many iterations, we can also specify the `thin` argument, which allows us to only extract the values of every ith iteration, which makes our simulation less computationally intensive. The simulation can be performed using the `mtc.run()` function. In our example, we will perform two seperate simulations with different settings to compare which one works the best. We have to provide the function with our compiled `model` object, and we have to specify the parameters we described above. We conduct a first simulation with less iterations, and then a second simulation in which we increase the number of iterations. We save these objects as `mcmc1` and `mcmc2`. Depending on the size of your network, the simulation may take some time to finish.

```
mcmc1 <- mtc.run(model, n.adapt = 50, n.iter = 1000, thin = 10)
mcmc2 <- mtc.run(model, n.adapt = 5000, n.iter = 100000, thin = 10)
```

11.3.3.5 Assessing the convergence

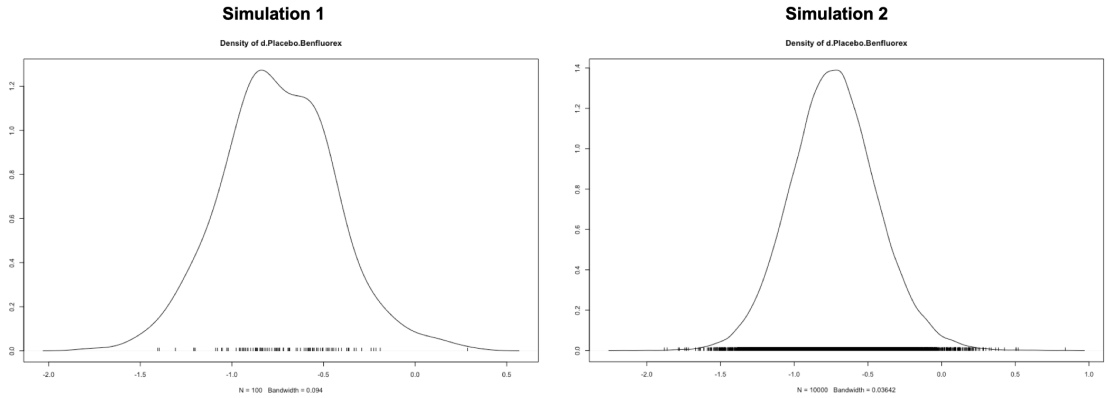
To see if our simulations have resulted in the **convergence** of the algorithm, and to check which settings are preferable, we can evaluate some of the outputs of our `mcmc1` and `mcmc2` objects. A good start is to use the `plot()` function. This provides us with a time-series plot for each treatment comparison over all iterations. For now, we only focus on the comparison **Placebo vs. Benfluorex**.

Here is the time series plot for `mcmc1`.

And this is the time series plot for `mcmc2`.

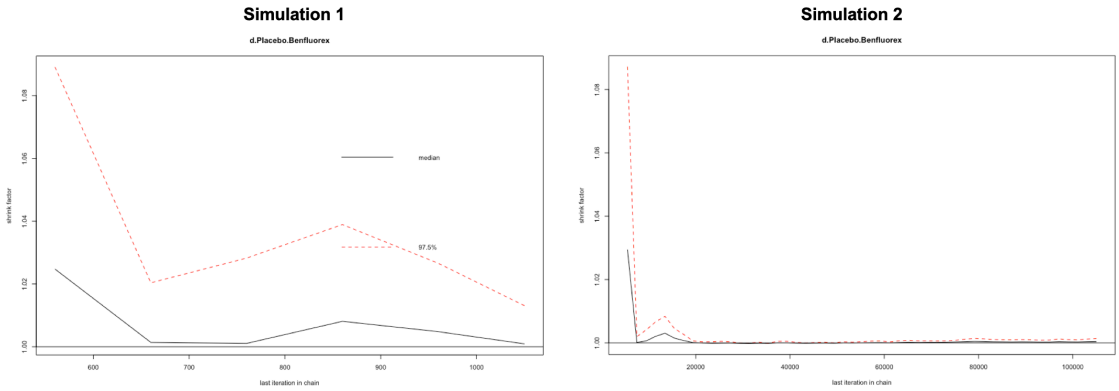
When comparing earlier iterations to later iterations in both plots, we see that there is much more **discontinuity**, and that the moving averages of the four different chains (the four solid lines) **differ in their course** when moving from the first half to the second half of the plot. The plot for our second simulation, on the other hand, is **much smoother**, which delivers a first indication that the settings of `mcmc2` are **more adequate**.

We can continue with our convergence evaluation by looking at the **density plots** of our posterior effect size estimate. We see that while the distribution resulting from `mcmc1` still diverges considerably from a smooth normal distribution, the result of `mcmc2` looks much more like this.



Another highly helpful method to assess convergence is the **Gelman-Rubin-Brooks** plot. This plot shows the so-called **Potential Scale Reduction Factor** (PSRF), which compares the variation within each chain in simulation to the variation between chains, and how it develops over time. In case of convergence, the PRSF should gradually shrink down to zero with increasing numbers of interations, and should at least be below 1.05 in the end. To produce this plot, we simply have to plug in the `mtc.run()` object into the `gelman.plot()` function. Here is there result for both simulations.

```
gelman.plot(mcmc1)
gelman.plot(mcmc2)
```



We can also directly access the **overall PSRF** of our model results using this code:

```
gelman.diag(mcmc1)$mpsrf
```

```
## [1] 1.055129
```

```
gelman.diag(mcmc2)$mpsrf
```

```
## [1] 1.000689
```

We see that the overall PRSF is only below the threshold for our second simulation `mcmc2`, indicating that we should use the results obtained by this simulation.

11.3.4 Assessing inconsistency: the nodesplit method

Like the netmeta package, the gemtc package also provides us with a way to evaluate the consistency of our network model, using the nodesplit method. The idea behind the procedure we apply here equals the one of the **net splitting** method we described before. To perform a nodesplit analysis, we use the `mtc.nodesplit()` function, using the same settings as used for mcmc2. We save the result as `nodesplit`.

Be aware that the nodesplit computation may take a long time, even up to hours, depending on the complexity of your network.

```
nodesplit <- mtc.nodesplit(network,
                           linearModel = "random",
                           n.adapt = 5000,
                           n.iter = 100000,
                           thin = 10)
```

The best way to interpret the results of our nodesplit model is to plug it into the summary function.

```
summary(nodesplit)
```

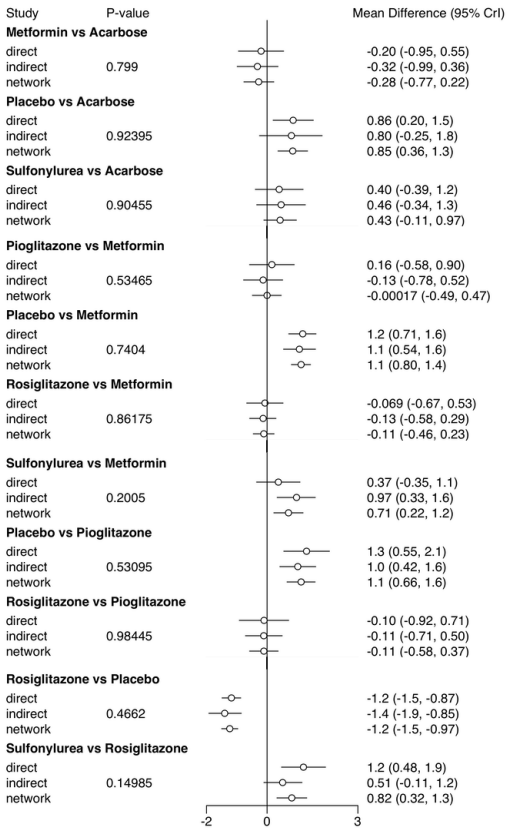
```
## Node-splitting analysis of inconsistency
## ========================================
##
##    comparison                   p.value CrI
## 1  d.Acarbose.Metformin         0.79900
## 2  -> direct                            -0.20 (-0.95, 0.55)
## 3  -> indirect                          -0.32 (-0.99, 0.36)
## 4  -> network                           -0.28 (-0.77, 0.22)
## 5  d.Acarbose.Placebo           0.92395
## 6  -> direct                            0.86 (0.20, 1.5)
## 7  -> indirect                          0.80 (-0.25, 1.8)
## 8  -> network                           0.85 (0.36, 1.3)
## 9  d.Acarbose.Sulfonylurea      0.90455
## 10 -> direct                            0.40 (-0.39, 1.2)
## 11 -> indirect                          0.46 (-0.34, 1.3)
## 12 -> network                           0.43 (-0.11, 0.97)
## 13 d.Metformin.Pioglitazone     0.53465
## 14 -> direct                            0.16 (-0.58, 0.90)
## 15 -> indirect                          -0.13 (-0.78, 0.52)
## 16 -> network                           -0.00017 (-0.49, 0.47)
## 17 d.Metformin.Placebo          0.74040
## 18 -> direct                            1.2 (0.71, 1.6)
## 19 -> indirect                          1.1 (0.54, 1.6)
## 20 -> network                           1.1 (0.80, 1.4)
## 21 d.Metformin.Rosiglitazone    0.86175
## 22 -> direct                            -0.069 (-0.67, 0.53)
## 23 -> indirect                          -0.13 (-0.58, 0.29)
## 24 -> network                           -0.11 (-0.46, 0.23)
## 25 d.Metformin.Sulfonylurea     0.20050
```

```
## 26 -> direct                            0.37 (-0.35, 1.1)
## 27 -> indirect                          0.97 (0.33, 1.6)
## 28 -> network                           0.71 (0.22, 1.2)
## 29 d.Pioglitazone.Placebo       0.53095
## 30 -> direct                            1.3 (0.55, 2.1)
## 31 -> indirect                          1.0 (0.42, 1.6)
## 32 -> network                           1.1 (0.66, 1.6)
## 33 d.Pioglitazone.Rosiglitazone 0.98445
## 34 -> direct                            -0.10 (-0.92, 0.71)
## 35 -> indirect                          -0.11 (-0.71, 0.50)
## 36 -> network                           -0.11 (-0.58, 0.37)
## 37 d.Placebo.Rosiglitazone      0.46620
## 38 -> direct                            -1.2 (-1.5, -0.87)
## 39 -> indirect                          -1.4 (-1.9, -0.85)
## 40 -> network                           -1.2 (-1.5, -0.97)
## 41 d.Rosiglitazone.Sulfonylurea 0.14985
## 42 -> direct                            1.2 (0.48, 1.9)
## 43 -> indirect                          0.51 (-0.11, 1.2)
## 44 -> network                           0.82 (0.32, 1.3)
```

The function output shows us the results for the effects of different comparisons when using only **direct**, only **indirect** and **all available** evidence. Different estimates using direct and indirect evidence would suggest the presence of **inconsistency**. We can control for this by looking at the **P-value** column. One or more comparisons with $p < 0.05$ would suggest that we have a problematic amount of inconsistency in our network. By looking at the output we see that **this is not the case**, thus indicating that the model we built is trustworthy.

We can also plug the summary into the `plot()` function to generate a forest plot of the information above.

```
plot(summary(nodesplit))
```

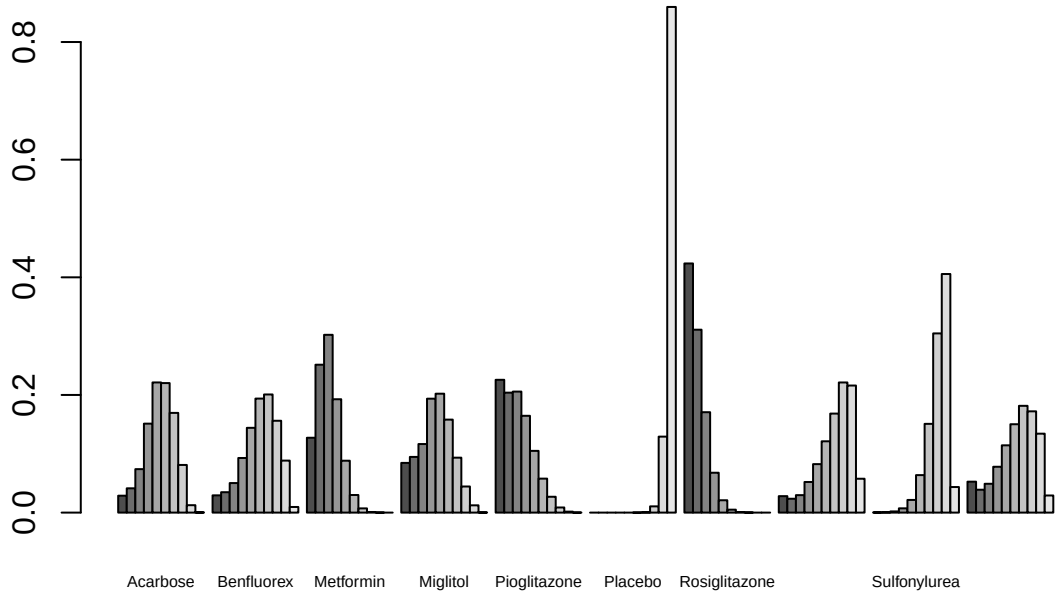


11.3.5 Generating the network meta-analysis results

Now that we produced our network meta-analysis model, and have convinced ourselves of its appropriateness, it is time to finally **produce the results of our analysis**, which we can report in our research paper.

As said before, the main question we may want to answer in network meta-analysis is: **which treatment performs the best?** To answer this question, we might first use the `rank.probability()` function, which calculates the probability for each treatment to be the best, second best, third best, and so forth, option. We only have to plug our `mcmc2` object into this function, and additionally specify the `preferredDirection`. If smaller (i.e. negative) effect sizes indicate better outcomes, we set `preferredDirection=-1`, otherwise we use `preferredDirection=1`. We save the result of the function as `rank`, and then plot the results in a so-called **rankogram**.

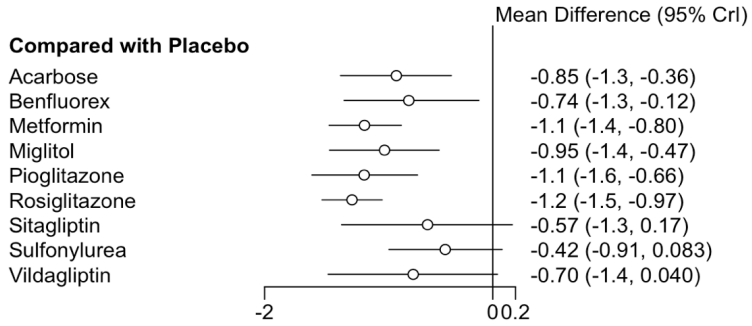
```
rank <- rank.probability(mcmc2, preferredDirection = -1)
plot(rank, beside=TRUE, cex.names=0.5)
```



According to this plot, we see that **Rosiglitazone** may probably be the best treatment option under study, given that its first bar (signifying the first rank) is the largest. This finding is in agreement with the results from the frequentist analysis, where we found the same pattern.

Additionally, we can also produce a forest plot of our results using the `forest()` function. To do this, we first have to put our results object into the `relative.effect()` function and have to specify `t1`, the reference treatment. Then, we plug this into the `forest()` function to generate the plot.

```
forest(relative.effect(mcmc2, t1 = "Placebo"))
```



In the chapter on frequentist network meta-analysis, we already covered the **P-score** as a metric to evaluate which treatment in a network is likely to be the most efficacious. An equivalent to the **P-score** is the **Surface Under the Cumulative RAnking** (SUCRA) score, which can be calculated like this (Salanti et al., 2011):

$$SUCRA_j = \frac{\sum_{b=1}^{a-1} cum_{jb}}{a-1}$$

Where j is some treatment, a are all competing treatments, b are the $b = 1, 2, ...a-1$ best treatments, and cum represents the cumulative probability of a treatment being among the b best treatments. To do this, we prepared the

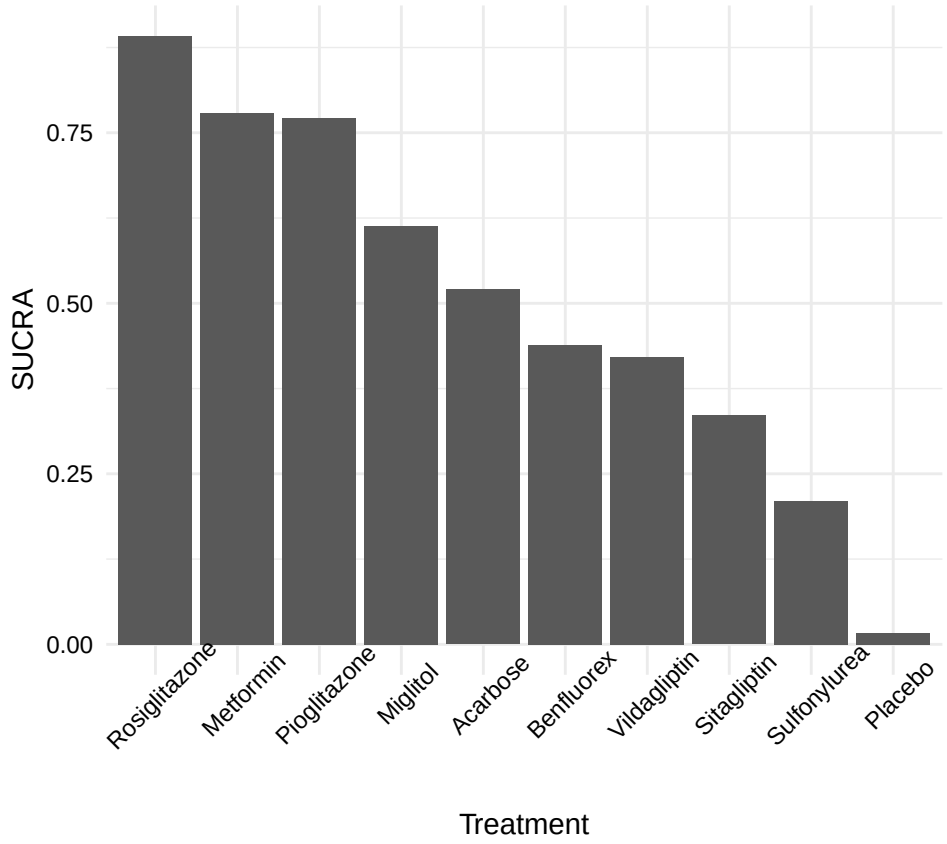


Figure 11.1: SUCRA Score

sucra function for you. The sucra function is part of the dmetar package. If you have the package installed already, you have to load it into your library first.

```
library(dmetar)
```

If you don't want to use the dmetar package, you can find the source code for this function here. In this case, *R* doesn't know this function yet, so we have to let *R* learn it by **copying and pasting** the code **in its entirety** into the **console** in the bottom left pane of RStudio, and then hit **Enter**. The function then requires the ggplot2 and forcats package to work.

The function only needs a rank.probability() object as input, and we need to specify if lower values indicate better outcomes in our context by specifying lower.is.better. Let us see what results we get.

```
rank.probability <- rank.probability(mcmc2)
sucra(rank.probability, lower.is.better = TRUE)
```

```
##                    SUCRA
## Rosiglitazone 0.89206111
## Metformin     0.77920000
## Pioglitazone  0.77183056
## Miglitol      0.61334167
## Acarbose      0.52037500
## Benfluorex    0.43883611
## Vildagliptin  0.42119722
```

```
## Sitagliptin   0.33563889
## Sulfonylurea  0.21059167
## Placebo       0.01692778
```

Looking at the SUCRA values of each treatment, we again see that **Rosiglitazone** may be the best treatment, followed by, with some distance, **Metformin** and **Pioglitazone**. A last thing we also often want to report in our research paper is the effect size estimate for **each treatment comparison**, based on our model. We can easily export such a matrix using the `relative.effect.table()` function. We can save the results of this function in an object, such as `result`, which we can then save to .csv file for later usage. The `relative.effect.table()` automatically creates a treatment comparison matrix containing the **estimated effect**, as well as the **credibility intervals** for each comparison. Here is how we can do this:

```
results <- relative.effect.table(mcmc2)
save(results, file = "results.csv")
```

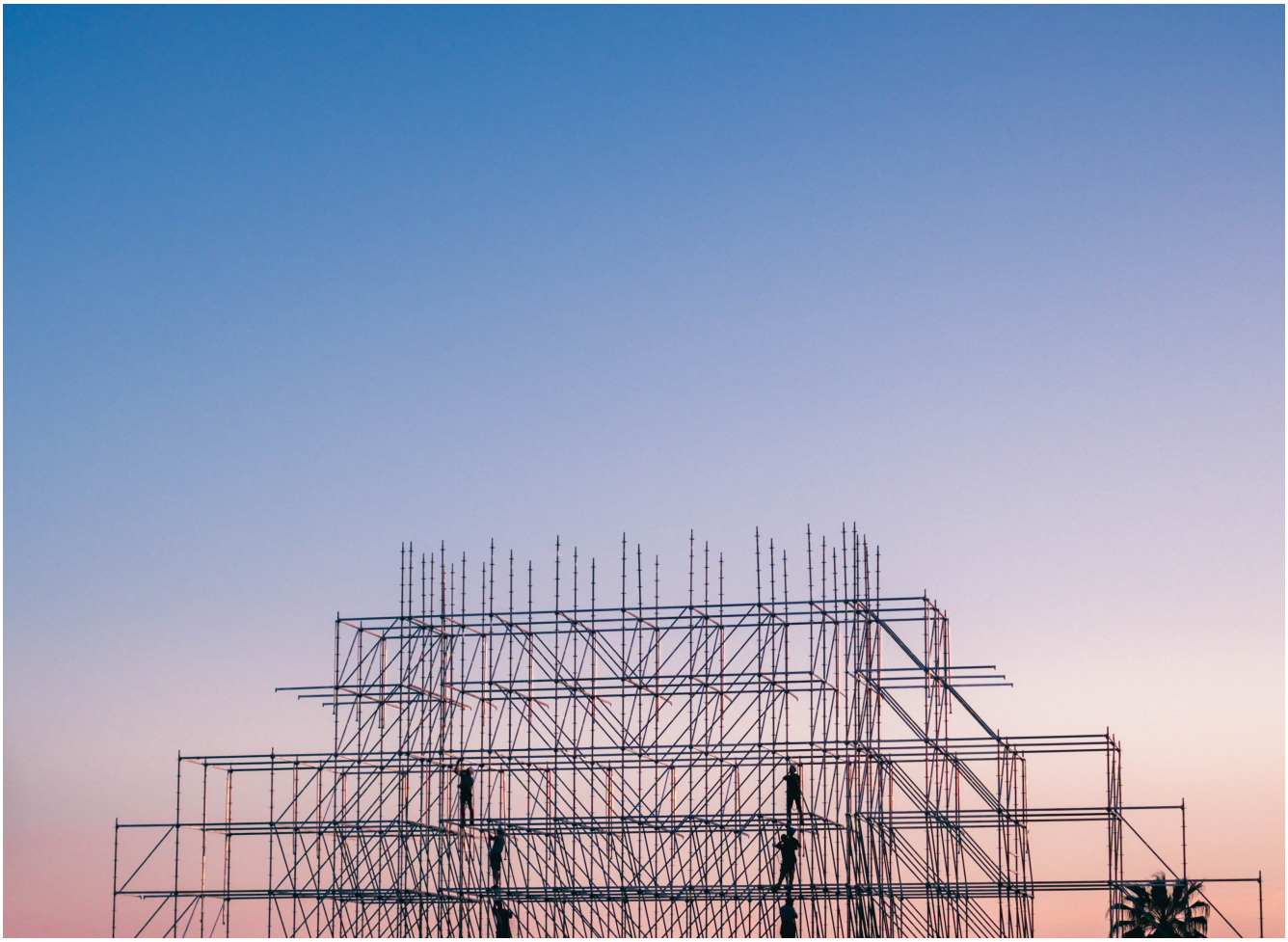
11.3.6 Summary

This is the **end of our short introduction** into network meta-analysis and the `netmeta` and `gemtc` package. We have described the general idea behind network meta-analysis, the assumptions and caveats associated with it, two different statistical approaches through which network meta-analysis can be conducted, and how they are implemented in R. Still, we would like to stress that what we covered here is **still only a rough overview** of this metholodological approach. There are many other techniques and procedures helpful in generating robust evidence for decision-making from network meta-analysis, and we recommend to get acquainted with them. Here are two rather easily digestable **literature recommendations** you may find helpful for this endeavor:

- Efthimiou, O., Debray, T. P., van Valkenhoef, G., Trelle, S., Panayidou, K., Moons, K. G., ... & GetReal Methods Review Group. (2016). GetReal in network meta analysis: a review of the methodology. *Research synthesis methods*, *7*(3), 236-263. Link
- Dias, S., Ades, A. E., Welton, N. J., Jansen, J. P., & Sutton, A. J. (2018). *Network meta-analysis for decision-making*. Wiley & Sons. Link

Chapter 12

“Multilevel” Meta-Analysis



This chapter deals with the topic of pooling effect sizes in **“multilevel” meta-analyses**. You probably wonder why we put the word “multilevel” into **quotation marks**. There is an increasing number of studies describing themselves as “multilevel” meta-analyses, which insinuates that a “multilevel” meta-analysis would be something special or

extraordinary compared to "standard" meta-analyses. This, however, is not true, as every meta-analytic model presupposes a multilevel structure within the data to pool results (Pastor and Lazowski, 2018). This means that basically, once you've reached this chapter in the guide, you will have already fitted a multilevel meta-analytic model to data several times (maybe even without knowing). When people talk about multilevel meta-analysis, what they often think of are **three-level meta-analytic models**, which are indeed somewhat different to the fixed-effect and random-effects models we presented before.

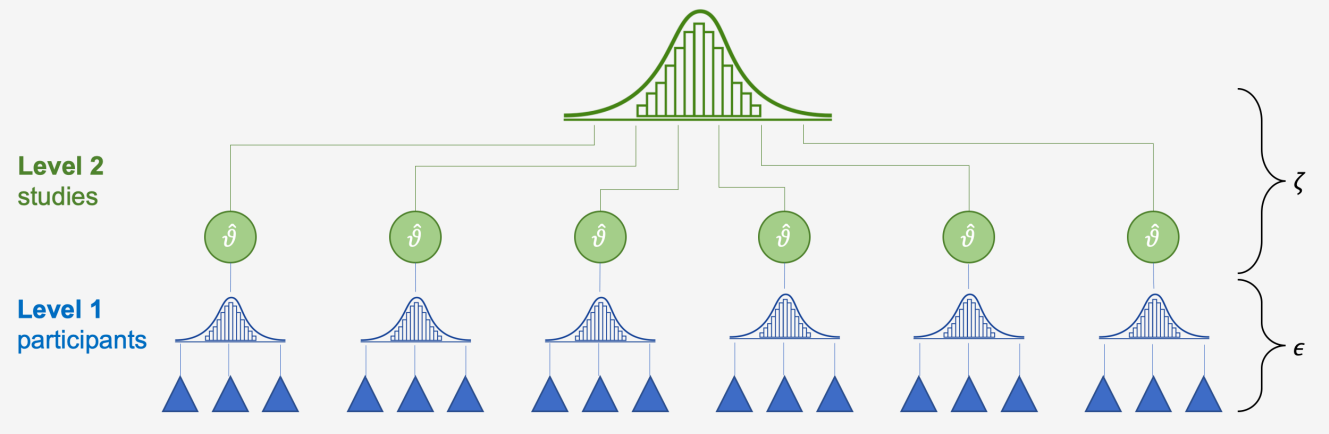
12.1 The multilevel nature of Meta-Analysis

To see why meta-analysis is by nature multileveled, let us go back to the formula for the random-effects model we discussed in Chapter 4.2:

$$\hat{\theta}_k = \mu + \epsilon_k + \zeta_k$$

We discussed that the terms ϵ_k and ζ_k are introduced in a random-effects model because we assume that there are two sources of variability. The first one is caused by the **sampling error** (ϵ_k) of individual studies, which causes their effect size estimates to deviate from the true effect size θ_k. The second one, ζ_k is the between-study heterogeneity caused by the fact that the true effect size of a study k itself is again only part of an **overarching distribution of true effect sizes**, from which the individual true effect size θ_k was drawn. Our aim in the random-effects model is therefore to estimate the mean of this distribution of true effect sizes, μ.

The two error terms correspond with the two levels within our meta-analysis data: the **"participant" level** (level 1) and the **"study" level** (level 2). The figure below symbolizes this structure. In the lower level (level 1), we have the participants (and/or patients). These participants are, of course, part of a number of larger units, the studies (level 2), from which we retrieved the data for our meta-analysis. While the data on level 1 usually already reaches us, the meta-analysts, in a "pooled" form (e.g., the authors of the paper provide us with the mean and standard deviation of their studied sample instead of the raw data), pooling on level 2, the study level, has to be performed in meta-analysis. In the tradition of multilevel modeling, such data is called **nested data**: in most meta-analyses, one can say that **participants are "nested" within studies**.



Thus, if we split the formula from above into the components corresponding to the two levels, we get the following formulae:

Level 1 (participants) model:

$$\hat{\theta}_k = \theta_k + \epsilon_k \tag{12.1}$$

Level 2 (studies) model:

$$\theta_k = \mu + \zeta_k \tag{12.2}$$

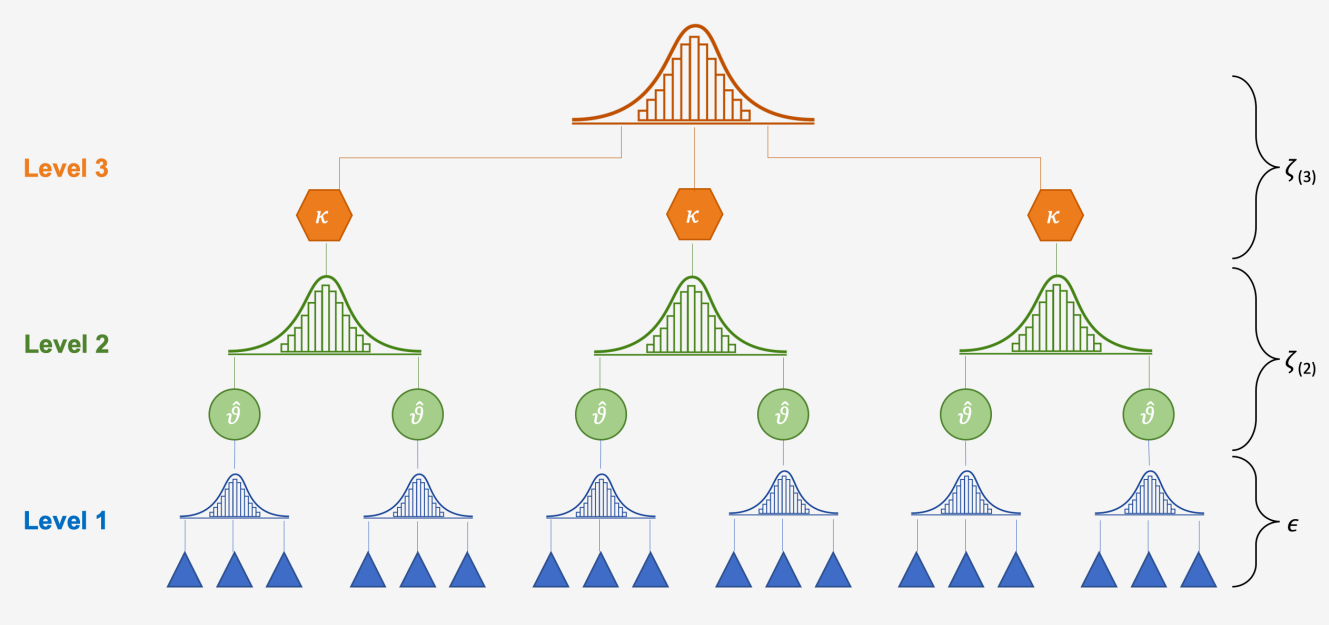
You might have already detected that we can substitute θ_k in the first equation with its definition in the second equation. What we then get is **exactly the generic formula for the meta-analytic model from before** (even a fixed-effects model can be defined as having a multilevel structure, yet with ζ_k being zero). Thus, it becomes evident that the way we define a meta-analytic model already has multilevel properties "built-in" given that we assume that participants are nested within studies in our data.

12.2 Three-level meta-analytic models

It has become clear that meta-analysis naturally possesses a multilevel structure. This allows us to expand this structure even further to better reflect our data. **Three-level models** (Cheung, 2014; Assink and Wibbelink, 2016) are a good way to do this. **Statistical independence** is one of the core assumptions of meta-analytic pooling (Hedges, 2009). If there is a dependency between effect sizes (i.e., the effect sizes are correlated), this may artificially reduce heterogeneity and thus lead to false-positive results. Dependence may stem from different sources (Cheung, 2014):

1. **Dependence introduced by the authors of the individual studies**. For example, scientists conducting the study might have collected data from multiple sites, or they might have compared multiple interventions to one single control group, or they might have used different questionnaires to measure an outcome of interest (e.g., they used the BDI as well as the PHQ-9 to assess depression in patients). For all of these scenarios, we can assume that some kind of dependency is introduced within the reported data.
2. **Dependence introduced by the meta-analyst herself**. For example, a meta-analysis could focus on a specific psychological mechanism and include studies which were conducted in different cultural regions of the world. It seems reasonable that the reported results are more similar when studies were conducted in the same cultural context.

We can take such dependencies into account by integrating a **third layer** into the structure of our meta-analysis model. For example, one could model that different questionnaires are nested within studies. Or one could create a model in which studies are nested within cultural regions. This creates a three-level meta-analytic model, as illustrated by the figure below.



The generic mathematical formulae for a three-level meta-analytic model look like this:

Level 1 model

$$\hat{\theta}_{ij} = \theta_{ij} + \epsilon_{ij}$$

Level 2 model

$$\theta_{ij} = \kappa_j + \zeta_{(2)ij}$$

Level 3 model

$$\kappa_j = \beta_0 + \zeta_{(3)j}$$

Where θ_{ij} is the **true effect size**, $\hat{\theta}_{ij}$ its estimator in the ith effect size in cluster j, κ_j the **average effect size** in j and β_0 the average population effect. Again, we can piece these formulae together and get this:

$$\hat{\theta}_{ij} = \beta_0 + \zeta_{(2)ij} + \zeta_{(3)j} + \epsilon_{ij}$$

The `metafor` package is particularly well suited for fitting various three-level models in meta-analyses. When setting up meta-analytic models, we have previously used the `meta` function primarly, because we think that this package's code is a little less technical. `metafor`, however, is also fairly easy to use once you prepared the data correctly. We'll get to this in the next chapter.

12.3 Fitting a three-level model

12.3.1 Data preparation

Before we start with fitting the model, we need the `metafor` package to be loaded from our library.

```
library(metafor)
```

For this chapter, i will use the `mlm.data` dataset. It has already been prepared for model fitting using `metafor`. Let's have a peek at the dataset:

```
mlm.data
```

```
## # A tibble: 80 x 6
##     ID_1  ID_2     y       v `peer-review` dissertation
##    <dbl> <int> <dbl>   <dbl>         <dbl>        <dbl>
##  1     1     1 -0.01 0.0740              1            0
##  2     1     2  0.11 0.0398              1            0
##  3     1     3  0.2  0.0481              1            0
##  4     1     4  0.3  0.0239              1            0
##  5     1     5  0.12 0.0331              1            0
##  6     1     6  0.02 0.0886              1            0
##  7     1     7  0.13 0.0115              1            0
##  8     1     8  0.18 0.00761             1            0
##  9     1     9  0.18 0.00649             1            0
## 10     2    10  0.44 0.332               1            0
## # ... with 70 more rows
```

We see that the dataset has 80 rows and 5 columns (i.e., 5 variables). The first four are the most important ones:

- **ID_1** and **ID_2**. In these two columns, we stored the identifiers corresponding to the individual effect sizes and clusters on different levels of our multilevel model. We have left these columns quite generic to make clear that such multilevel data can accommodate various types of data structures. For example, one can think of a scenario where ID_2 corresponds with the different effect sizes we find within a study (e.g., different effect sizes for different measures of depression). In this example, the effect sizes in ID_2 would then be nested within the larger clusters defined in ID_1, the studies. Another possible example would be that ID_2 in fact symbolizes the unique identifier for single studies, which are then nested in larger clusters (e.g., cultural regions) shown in ID_1.
- **y** and **v**. In these columns, the effect size data is stored. The format is similar to the one we know from the `metagen` function in `meta`. The column `y` contains the effect size (d, g, r, z, or other continuous effect size metrics). The `v` column contains the sampling variance which can be obtained by **squaring the standard error**.

There are two additional columns which contain data needed for **subgroup analyses**. To conduct subgroup analyses with `metafor`, however, we need to store the data a little differently than in Chapter 7. This time, we do not use a factor vector containing the subgroup levels as strings, but **recode all subgroup levels** as individuals columns in the data. In our case, we are interested if the publication status has a moderating effect on the pooled results. We want to compare studies published in peer-review journals to dissertations. To do this, we need to create **two variables/columns for each subgroup level** (peer-review, dissertation). In each column we provide a code if the effect size belongs to this subgroup, defined by `1 = yes` and `0 = no`. Of course, the subgroup codings now have to be

mutually exclusive, as we cannot calculate subgroup effects if a study is both part of the "peer-review" group and the "dissertation" group.

12.3.2 Model fitting

We are now prepared to fit our model. I will save the model to the object `full.model`. For model fitting, we use the `rma.mv` function in `metafor`. The function has plenty of parameters one can specify (type `?metafor::rma.mv` in the console to see them all). For now, we will only focus on the really essential ones.

Code	Description
y	The variable in our dataset containing the effect size data
v	The variable in our dataset containing the sampling variance data corresponding to each y
random	The model specifications for our multilevel model. We describe this in more detail below
tdist	This is an old friend in disguise, the Knapp-Hartung adjustment for our confidence intervals (if set to TRUE)
data	The dataframe containing our data
method	The tau-squared estimator. Our options here are limited, and it is advised to use the restricted maximum likelihood ('REML') method, which is the default

Setting up the formula

Under the `random` parameter, we feed `rma.mv` with the formula for our model. As we actually have two formulas in three-level models, we have to bundle them together in a `list()`. The notation for `rma.mv` is very similar to other packages specialized for mixed-effects models such as `lme4` (Bates et al., 2015).

Within the `list()`, formulae are separated with commas. Within the formula, we first define the random effects variance as `~ 1`, followed by a **vertical bar** `|`. Behind the vertical bar, we assign this random effect to a **grouping variable** such as studies, measures, regions and so forth. We only have to define the formulae for **level 2** and **level 3**, as the sampling error in level 1 is assumed to be known. We type in the formulae in the order of our multilevel model. As in our example, ID_2 is nested in ID_1, we first define level 2 as `~ 1 | ID_2` and then level 3 as `~ 1 | ID_1`. Now, let's put it all together:

```
full.model<-rma.mv(y,
                   v,
                   random = list(~ 1 | ID_2,
                                 ~ 1 | ID_1),
                   tdist = TRUE,
                   data = mlm.data,
                   method = "REML")
```

To see the results of our model, we can use the `summary()` function.

```
summary(full.model)
```

```
##
## Multivariate Meta-Analysis Model (k = 80; method: REML)
##
##   logLik  Deviance       AIC       BIC      AICc
```

```
## -44.6962   89.3924   95.3924  102.5007   95.7124
##
## Variance Components:
##
##             estim    sqrt  nlvls  fixed  factor
## sigma^2.1  0.0561  0.2368     80     no    ID_2
## sigma^2.2  0.1575  0.3968     14     no    ID_1
##
## Test for Heterogeneity:
## Q(df = 79) = 624.8063, p-val < .0001
##
## Model Results:
##
## estimate      se    tval    pval   ci.lb   ci.ub
##   0.4345  0.1173  3.7058  0.0004  0.2011  0.6679  ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Let's go through the output one by one. First, let's have a look at the `Variance Components`. Here, we see the variance calculated for each level of our model, σ_1^2 (level 2) and σ_2^2 (level 3). Under `nlvls`, we see the number of levels (i.e., subgroups) on each level. For `ID_2`, this is 80 because each effect size has its own ID.

Under `Model Results`, we see the `estimate` of our pooled effect, which is 0.4345. As our `y` column contained effect sizes expressed as Hedges' g, the effect is therefore $g = 0.4545$, with a confidence interval of $0.20 - 0.67$.

Under `Test for Heterogeneity`, we see that the variance of effect sizes within our data overall was highly significant ($p < 0.0001$). This however, is not very informative as we are interested how variance can be attributed to the different levels in our model.

12.3.3 Distribution of total variance

As mentioned before, what we're actually interested in is the **distribution of variance over the three levels of our model**. Cheung (2014) provides a formula to estimate the sampling variance, which is needed to calculate the variance proportions for each level. We have have prepared a function called `mlm.variance.distribution`, which implements this formula. The function is part of the `dmetar` package. If you have the package installed already, you have to load it into your library first.

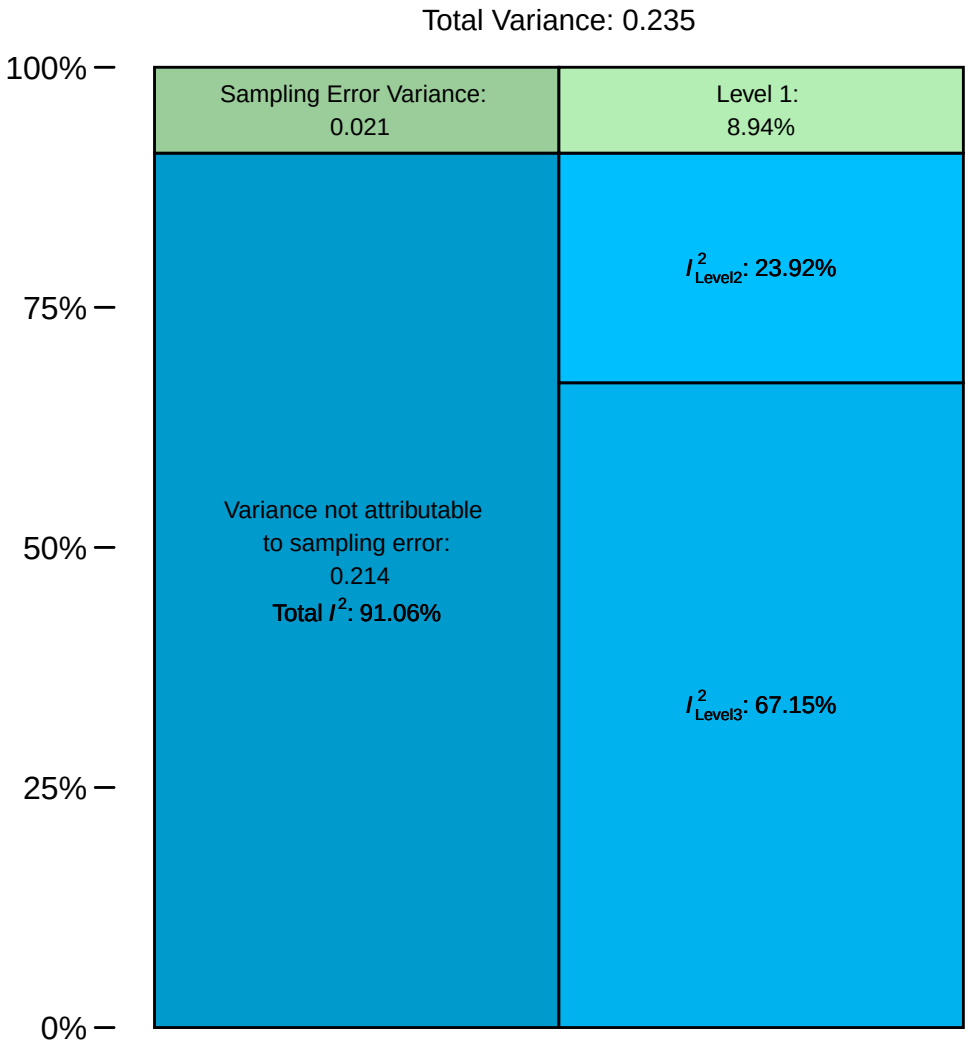
```
library(dmetar)
```

If you don't want to use the `dmetar` package, you can find the source code for this function here. In this case, *R* doesn't know this function yet, so we have to let *R* learn it by **copying and pasting** the code **in its entirety** into the **console** in the bottom left pane of RStudio, and then hit **Enter** . The function then requires the `ggplot2` package to work.

We only have to specify one parameter: `x`, the model we just fitted using `rma.mv`. Let's have a look:

```
mlm.variance.distribution(x = full.model)
```

```
##         % of total variance    I2
## Level 1            8.937849   ---
## Level 2           23.916209 23.92
## Level 3           67.145942 67.15
## Total I2: 91.06%
```



From the outputs, we see that about 8.94% of the overall variance can be attributed to level 1, 23.92% to level 2, and as much as 67.15% to level 3. The variance proportions at level 2 and 3 are equivalent to the between-study heterogeneity I^2 we covered in Chapter 6. However, in the context of three-level meta-analysis models, there are three types of I^2 values: I^2_{Level2}, which describes the amount of within-cluster heterogeneity, I^2_{Level3}, representing the amount of between-cluster heterogeneity, and I^2_{total}, which is the total amount of heterogeneity not attributable to sampling error. I^2_{total} is simply the sum of the I^2 values at level 2 and level 3.

12.3.4 Comparing the fit

Of course, when fitting a three-level model, we are interested if this model actually **captures the variability in our data better than a two-level model**. `metafor` allows us to easily do this by comparing models **in which one of the levels is removed**.

To do this, we can use the rma.mv function again, but this time, we hold the variance component σ^2 of one level constant. This can be done by specifying the sigma2 parameter. The parameter can be specified by providing a vector telling the function which level to hold constant, with the generic version being c(level 2, level3). So, if we want to hold level 2 constant while leaving the rest as is, we use sigma2 = c(0,NA), and vice verse for the third level.

Model without level 2

```
model.l2.removed<-rma.mv(y,
                  v,
                  random = list(~ 1 | ID_2,
                                ~ 1 | ID_1),
                  tdist = TRUE,
                  data = mlm.data,
                  method = "REML",
                  sigma2 = c(0, NA))
summary(model.l2.removed)
```

```
##
## Multivariate Meta-Analysis Model (k = 80; method: REML)
##
##   logLik  Deviance       AIC       BIC      AICc
## -59.8748  119.7495  123.7495  128.4884  123.9074
##
## Variance Components:
##
##             estim    sqrt  nlvls  fixed  factor
## sigma^2.1  0.0000  0.0000     80    yes    ID_2
## sigma^2.2  0.1620  0.4025     14     no    ID_1
##
## Test for Heterogeneity:
## Q(df = 79) = 624.8063, p-val < .0001
##
## Model Results:
##
## estimate      se    tval    pval   ci.lb   ci.ub
##   0.4217  0.1130  3.7337  0.0004  0.1969  0.6465  ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see that the Model Results have changed, but does the second level in fact capture a significant amount of variability in the data? To check, we can use the anova function to compare the full.model to the model.l2.removed.

```
anova(full.model, model.l2.removed)
```

```
##
##         df      AIC      BIC     AICc   logLik     LRT   pval       QE
## Full     3  95.3924 102.5007  95.7124 -44.6962                624.8063
```

```
## Reduced  2 123.7495 128.4884 123.9074 -59.8748 30.3571 <.0001 624.8063
```

We see that the `Full` model compared to the Reduced model does indeed have a better fit, with the **Akaike Information Criterion** (`AIC`) and the **Bayesian Information Criterion** (`BIC`) being lower for this model. The difference is significant ($p < 0.0001$), suggesting that it was a good idea to include this level into our analysis. Now, let's do the same when holding level *3* constant.

Model without level *3*

```
model.l3.removed<-rma.mv(y,
                  v,
                  random = list(~ 1 | ID_2,
                                ~ 1 | ID_1),
                  tdist = TRUE,
                  data = mlm.data,
                  method = "REML",
                  sigma2 = c(NA, 0))
summary(model.l3.removed)
```

```
##
## Multivariate Meta-Analysis Model (k = 80; method: REML)
##
##   logLik  Deviance       AIC       BIC      AICc
## -75.4151  150.8302  154.8302  159.5691  154.9881
##
## Variance Components:
##
##             estim    sqrt  nlvls  fixed  factor
## sigma^2.1  0.3119  0.5585     80     no    ID_2
## sigma^2.2  0.0000  0.0000     14    yes    ID_1
##
## Test for Heterogeneity:
## Q(df = 79) = 624.8063, p-val < .0001
##
## Model Results:
##
## estimate      se    tval    pval   ci.lb   ci.ub
##   0.6049  0.0691  8.7539  <.0001  0.4674  0.7424  ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(full.model, model.l2.removed)
```

```
##
##         df      AIC      BIC     AICc   logLik     LRT   pval       QE
## Full     3  95.3924 102.5007  95.7124 -44.6962                624.8063
## Reduced  2 123.7495 128.4884 123.9074 -59.8748 30.3571 <.0001 624.8063
```

We see the same pattern here, suggesting that overall, our three-level model is very useful.

12.4 Subgroup Analyses in Three-Level Models

Once our **three-level meta-analytic model** is set, it is easy to also check for moderators of the overall effect, i.e. conduct subgroup analyses. To do this, we can use the `mods` parameter in `rma.mv`. The subgroup/moderator can be specified by putting a tilde (~) in front of it. In our case, we want to check if dissertations report higher or lower effect sizes, so we use `mods = ~ dissertation`. Here's the full code:

```
model.mods<-rma.mv(y,
                   v,
                   random = list(~ 1 | ID_2,
                                 ~ 1 | ID_1),
                   tdist = TRUE,
                   data = mlm.data,
                   method = "REML",
                   mods = ~ dissertation)
summary(model.mods)
```

```
##
## Multivariate Meta-Analysis Model (k = 80; method: REML)
##
##   logLik  Deviance       AIC       BIC      AICc
## -43.8184   87.6367   95.6367  105.0636   96.1847
##
## Variance Components:
##
##             estim    sqrt  nlvls  fixed  factor
## sigma^2.1  0.0581  0.2410     80     no    ID_2
## sigma^2.2  0.1594  0.3992     14     no    ID_1
##
## Test for Residual Heterogeneity:
## QE(df = 78) = 428.5167, p-val < .0001
##
## Test of Moderators (coefficient 2):
## F(df1 = 1, df2 = 78) = 0.0226, p-val = 0.8808
##
## Model Results:
##
##               estimate      se    tval    pval    ci.lb   ci.ub
## intrcpt         0.4297  0.1229  3.4972  0.0008   0.1851  0.6744  ***
## dissertation    0.0339  0.2255  0.1504  0.8808  -0.4150  0.4828
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

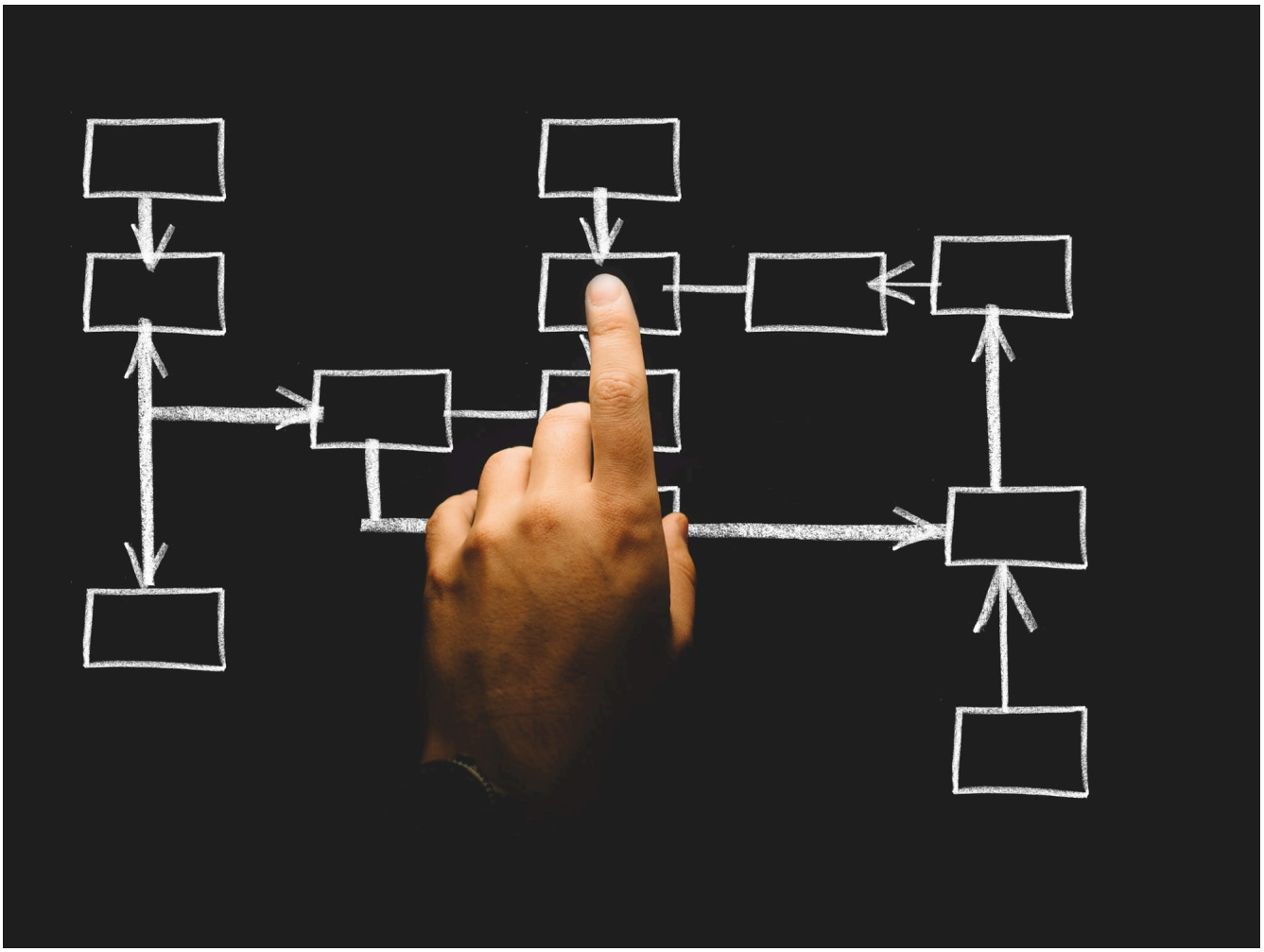
The first important output is the `Test of Moderators`. We see that $F_{1,78} = 0.0226$ with $p = 0.8808$, meaning that there is no significant difference between subgroups. The `Model Results` are printed within a meta-regression framework, so we **cannot simply extract the estimates directly to attain the summary effect size within subgroups**. The first row, the `intrcpt` (intercept) is the value of g when dissertation = 0, i.e., the value for peer-review articles. The

predictor `dissertation`'s estimate is expressed as a **slope**, meaning that we have to add its value to the intercept to get the actual summary effect. This is:

$$g_{dissertation} = \beta_0 + \beta_{dissertation} = 0.4297 + 0.0339 = 0.4636$$

Chapter 13

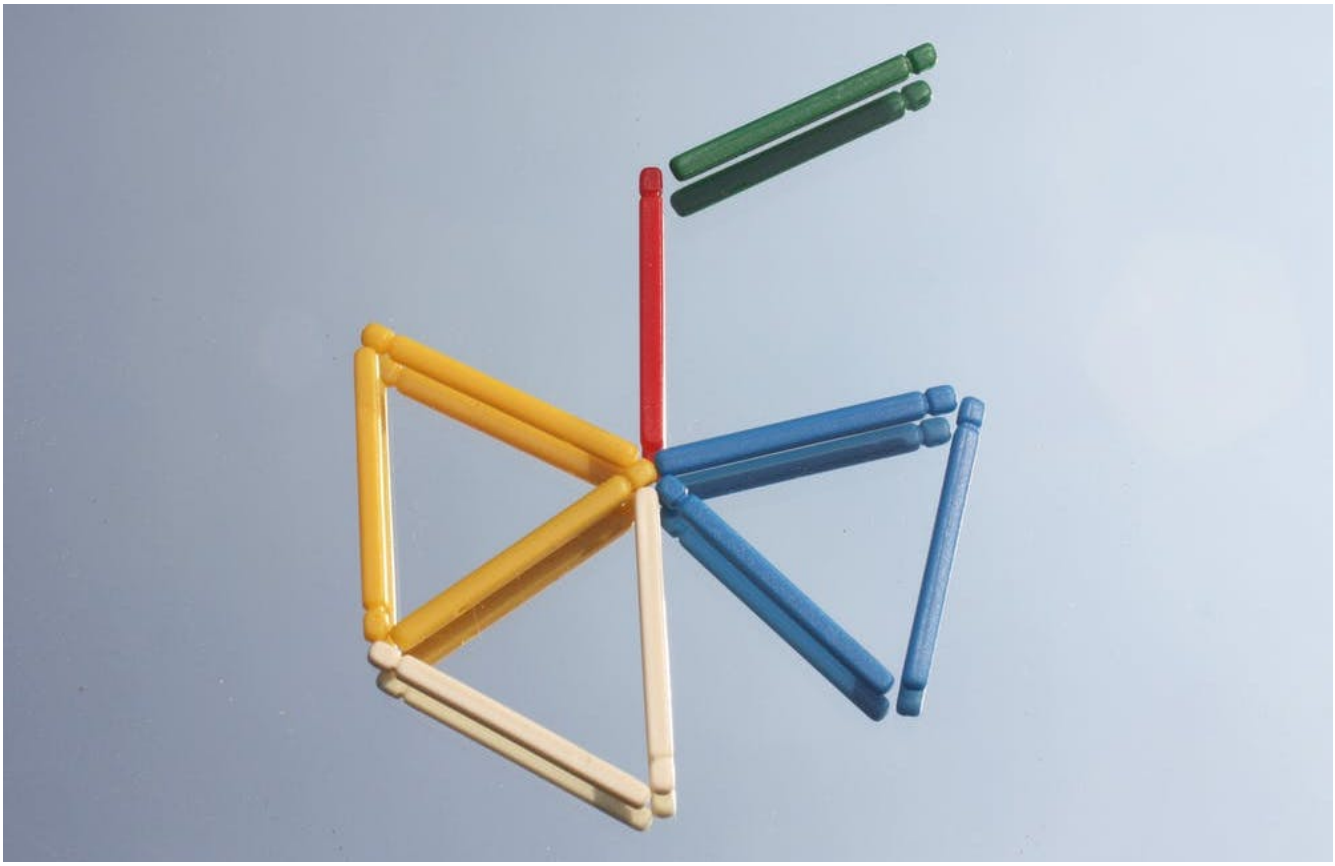
Structural Equation Modeling Meta-Analysis



In the past Chapter on Three-Level Meta-Analysis Models, we were able to generalize our conceptual knowledge on meta-analyses by showing that meta-analysis models have an **inherent multilevel structure**, which can be used to extend conventional meta-analysis models to three-level models. A peculiar thing about statistical methods is that they are often put into seperate "boxes", and treated as unrelated in research and practice, when in fact they are not. For many social scientists, for example, it is often surprising to find out that Analysis of Variance (ANOVA) and dummy-coded Regression are doing essentially the same thing (Montgomery, 2001). This often happens because

two methods are traditionally used in different contexts, and taught as separate entities. It might thus have been only fairly recently that researchers recognized that **multilevel models** are simply a special form of a **Structural Equation Model**, or **SEM** (Bauer, 2003; Mehta and Neale, 2005). As said before, every meta-analysis model is in itself a multilevel model, so this association also has exciting implications for meta-analysts: **we can treat our meta-analysis as a structural equation model**, in which the pooled effect size we want to estimate is the **latent** (or unobserved) variable (Cheung, 2015b,a). This does not *only* mean that we can model previous types of meta-analyses we presented before from a SEM perspective, but also allows us to use structural equation modeling to test more complex models meta-analytically. This is a great advantage; for example, using this approach, we can test **mediation** models, **factor analytic** models, or perform **multivariate meta-analyses** based on effect size data obtained from several independent studies. This is a great way to evaluate if certain models or theories in the literature are actually correct if we use all available evidence, if the theory's assumptions are not backed by the evidence, or, even more interestingly, if the theory does only apply to a subgroup of individuals or entities.

13.1 The Idea behind Meta-Analytic SEM



13.1.1 What is Structural Equation Modeling?

Structural Equation Modeling (SEM) is a statistical technique to model and test hypotheses about the relationship of **manifest** (observed) and (usually) **latent** (unobserved or unobservable) variables (Kline, 2015). Latent variables, as said, are either not observed or observable (personality, for example, is a latent construct which can only be measured indirectly, for example through different items in a questionnaire). In SEM, the assumed relationship between the

manifest and latent variables (the "structure") can be modeled using the manifest, measured variables while taking their measurement error into account. SEM analysis is somewhat different to "conventional" statistical tests (e.g., t-tests). Usually, for example in a t-test, the researcher tests against a null hypthesis, such as $H_0 : \mu_1 = \mu_2$ (where μ_1 and μ_1 are the means of two groups). The researchers "aims" to *reject* the null hypothesis to conclude that the two groups differ. In SEM, a specific model is proposed beforehand instead, and the researcher "aims" to *accept* this model if the **goodness of fit** is sufficient (Cheung, 2015a).

13.1.2 Model Specification

Usually, SEM are specified and represented mathematically as a series of **matrices**. You can think of a matrix as a simple table containing rows and columns, much like a `data.frame` object in *R* (in fact, most `data.frame`s can be easily converted to a matrix using the `as.matrix()` function). Visually, SEM can be represented as **path diagrams**. Such path diagrams are often very intuitive, so we will start specifying a SEM using this approach first, and then move on to the matrix notation.

13.1.2.1 Path Diagrams

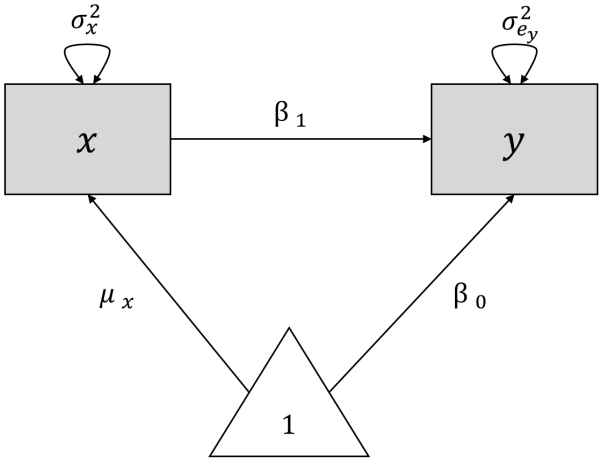
Path diagrams represent the mathematical model of our SEM graphically. There is no full consensus on how path diagrams should be drawn, yet there are a few conventions. Here are the main components of path diagrams, and how they are represented.

Symbol	Name	Description
□	Rectangle	Manifest/observed variables
○	Circle	Latent/unobserved variables
△	Triangle	Intercept (fixed vector of 1s)
→	Arrow	Prediction. The variable at the start of the arrow predicts the variable at the end of the arrow: Predictor → Target.
↔	Double Arrow	(Co-)Variance. If a double arrow connects two variables (rectangles/circles), it signifies the covariance/correlation between the two variables. If a double arrow forms a loop on one single variable, it signifies the variance of the variable.

As an illustration, let us create path diagram for a simple linear (non-meta-analytic) regression model, in which we want to predict y with x. The model formula looks like this:

$$y_i = \beta_0 + \beta_1 x_i + e_i$$

In this model, x_i and y_i are the observed variables. There are no unobserved variables. The parameter μ_x is the population mean of x, while the population mean of y is the regression intercept, β_0. The variance of our observed data for x is σ_x^2. Provided that x is not a perfect predictor of y, there will be some amount of error variance $\sigma_{e_y}^2$ in our predictions associated with y. There are two regression coefficients: β_0, the intercept, and β_1, the slope coefficient of x. **Using these components, we can build a graphical model for a simple univariate linear regression:**



We can also use this graphical model as a starting point to reassemble the regression model equation. From the model, we can infer that y is influenced by two components: $x \times \beta_1$ and $1 \times \beta_0$. If we add these two parts together, we again arrive at the formula for $\hat{y}$ from before.

13.1.2.2 Matrix Representation

There are several common ways to represent SEM mathematically through matrices (Jöreskog and Sörbom, 2006; Muthén and Muthén, 2012; McArdle and McDonald, 1984). Here, we will focus on the **Reticular Action Model** formulation, or **RAM** (McArdle and McDonald, 1984), because this formulation is used by the `metaSEM` package we will be introducing later on. In the RAM, four matrices are specified: $\mathbf{F}$, $\mathbf{A}$, $\mathbf{S}$ and $\mathbf{M}$. Because the $\mathbf{M}$ matrix is not necessary to fit the meta-analytic SEM we will present later, we omit it here (see Cheung (2015a) for a more extensive introduction). We will now specify the remaining three matrices for our linear regression from before. The three matrices all have the same number of rows and columns, corresponding with the (manifest and latent) variables we have in our model: x and y. The generic structure in our regression example for all matrices therefore looks like this:

$$\begin{array}{c} \\ x \\ y \end{array} \begin{array}{c} \begin{array}{cc} x & y \end{array} \\ \begin{bmatrix} i_{x,x} & i_{x,y} \\ i_{y,x} & i_{y,y} \end{bmatrix} \end{array}$$

The $\mathbf{A}$ Matrix: Single Arrows

The $\mathbf{A}$ matrix represents the asymmetrical (single) arrows in our path model. The way to fill this matrix is to search for the matrix **column entry** of the variable in which the arrow starts (x), and then for the matrix **row entry** of the variable in which the arrow ends (y). We put the value of our arrow, β_1, where the selected column and row intersect in the matrix ($i_{y,x}$). Given that there are no more paths between the variables in our matrix, we fill all other fields with 0. The $\mathbf{A}$ matrix for our example therefore looks like this:

$$A = \begin{array}{c} \\ x \\ y \end{array} \begin{array}{c} \begin{array}{cc} x & y \end{array} \\ \begin{bmatrix} 0 & 0 \\ \beta_1 & 0 \end{bmatrix} \end{array}$$

The $\mathbf{S}$ Matrix: Double Arrows/Variances

The $\mathbf{S}$ matrix represents the (co)variances we want to estimate for the included variables. For x, our predictor, we want to estimate the variance σ^2_x. For our estimated regression target $\hat{y}$, we want to know the error variance $\sigma^2_{e_y}$. We therefore specify $\mathbf{S}$ like this:

$$S = \begin{array}{c} \\ x \\ y \end{array} \begin{array}{c} \begin{array}{cc} x & y \end{array} \\ \begin{bmatrix} \sigma^2_x & 0 \\ 0 & \sigma^2_{e_y} \end{bmatrix} \end{array}$$

The $\mathbf{F}$ Matrix: Observed Variables

The $\mathbf{F}$ matrix allows us to specify the observed (vs. latent) variables in our model. To specify that a variable has been observed, we simply insert 1 in the respective diagonal field of the matrix. Given that both x and y are observed in our model, we put 1 in all diagonal fields of the matrix:

$$F = \begin{array}{c} \\ x \\ y \end{array} \begin{array}{c} \begin{array}{cc} x & y \end{array} \\ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{array}$$

Once these matrices are set, it is possible to estimate the parameters in our SEM, and to derive how good the specified model fits the data using matrix algebra and **Maximum-Likelihood** estimation. We omit how these steps are performed here. If you are interested in understanding the details behind this step, you can have a look at Cheung (2015a), McArdle and McDonald (1984), or this blog post.

13.1.3 Meta-Analysis from a SEM perspective

We will now combine our knowledge about meta-analysis models and SEM to formulate the random/fixed-effect model as a structural equation model (Cheung, 2008). To begin, let us return to the formula of the **random-effects model** first. Previously, we already described that the random-effects model follows a multilevel structure, which looks like this:

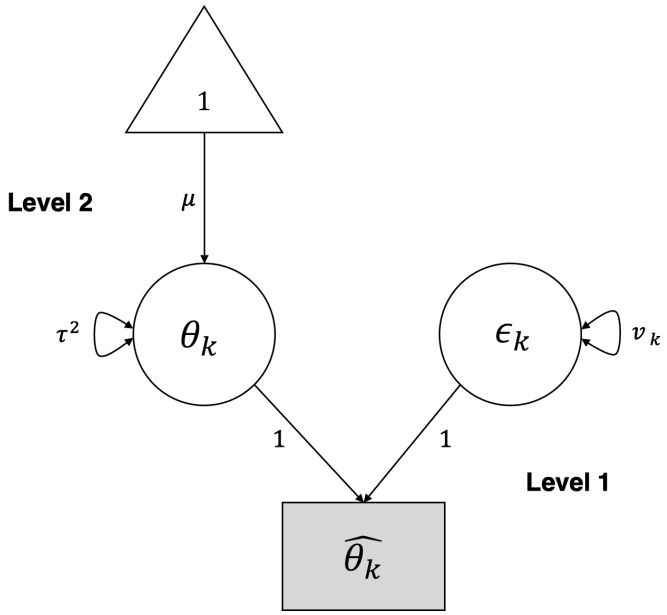
Level 1:

$$\hat{\theta}_k = \theta_k + \epsilon_k$$

Level 2:

$$\theta_k = \mu + \zeta_k$$

On the first level, we assume that the effect size $\hat{\theta}_k$ we observe for some study k in our meta-analysis is an estimator for the true effect size of k, θ_k. The observed effect size $\hat{\theta}_k$ deviates from the true effect size θ_k because of the sampling error ϵ_k, the variance of which is $Var(\epsilon_k) = v_i$. In a random-effects model, we assume that even the true effect size for k is only drawn from a "super-population" of true effect sizes at level 2. The mean of this "super-population" μ is what we want to estimate as the pooled effect in a random-effects model, along with the variance of the "super-population", $Var(\zeta_k) = \tau^2$: the between-study heterogeneity. The fixed-effect model is only a special case of the random-effects model where we assume that τ^2 is zero. It is quite straightforward to represent this model as a SEM graph if we use the variables on level 1 as **latent variables** to "explain" how the effect sizes we observed came into being (Cheung, 2015a):



In this graphical model, it becomes clear that the observed effect size $\hat{\theta}_k$ in the kth study is "influenced" by two arms: by the sampling error ϵ_k with variance v_k, and the true effect size θ_k with variance τ^2.

13.1.4 The Two-Stage Meta-Analytic SEM approach

Above, we defined the (random-effects) meta-analysis model from a SEM perspective. Although this is interesting from a theoretical standpoint, the model above is still not more or less capable than the meta-analysis techniques we learned before: it simply pools effect sizes assuming a random-effects model. To really use the versatility of meta-analytic SEM, an approach involving two steps is required (Tang and Cheung, 2016; Cheung, 2015a). In **Two-Stage Structural Equation Modeling** (**TSSEM**), we first **pool the effect sizes** from each study (usually correlations between variables we want to use for modeling). This allows for evaluating the heterogeneity of the pooled effect sizes, and if a random-effects model or subgroup analyses should be used. Using the maximum-likelihood-based approach used

by the metaSEM package we will introduce in the following, even studies with **missing data** can be included. In the second step, **weighted least squares** (WLS) estimation is used to fit the structural equation model we specified. The function for the specified model $\rho(\theta)$ is (Cheung and Chan, 2009; Cheung, 2015a):

$$F_{WLS}(\theta) = (\mathbf{r} - \rho(\theta))^{\top}\mathbf{V}^{-1}(\mathbf{r} - \rho(\theta))$$

Where $\mathbf{r}$ is a transformation of the pooled correlation matrix. The important part in this formula is $\mathbf{V}^{-1}$, the matrix containing the covariances of $\mathbf{r}$, from which the inverse is taken. This approach is quite similar to the inverse-variance principle (see Chapter 4) traditionally used in meta-analysis. It gives effects with lower variance (i.e., greater precision/N) a larger weight in the estimation process. This is also a good way to account for the uncertainty in our estimates which may have been introduced by missing data. Importantly, the formula for this second step is the same whether we assume a random or fixed-effect model, because the between study-heterogeneity, if existant, is already taken care of in step 1.

13.2 Multivariate Meta-Analysis

Now it is time to delve into our first worked example of a meta-analytic SEM using *R*. We will begin by using the SEM-based approach for **multivariate meta-analysis**, which has not been covered before. In multivariate meta-analyses, each study contributes more than just one effect size at the same time. This may be helpful in cases where we are studying a problem or condition for which there are several main outcomes, not just one. For example, for some type of treatment, there could be two types of outcomes which are deemed as important in the literature, and are thus assessed in most studies (Schwarzer et al., 2015). In multivariate meta-analyses, we can estimate the effect sizes for both outcomes simultaneously in one model. Taking the correlation between the two outcomes into account, we can also determine if **studies with a high effect size on one outcome also have higher effect sizes on the other outcome**, or if there is no, or a negative relationship.

It is of note here that multivariate meta-analysis can also be performed outside a SEM framework (Schwarzer et al., 2015). Nevertheless, as an introduction, we will to show you how multivariate meta-analysis can be performed from a SEM perspective. In this and the following examples, we will work with `metaSEM`, a magnificent package for meta-analytic SEM developed by Mike Cheung (Cheung, 2015c). To begin, as always, install the `metaSEM` package and load it from your library.

```
library(metaSEM)
```

For my multivariate meta-analysis, i will use the *dat.da* dataset. This is a fictitious dataset (adapted from the wvs94a data in `metaSEM`) containing $k = 37$ studies examining the effects of psychotherapy on symptoms of Depression and Anxiety, expressed as the Standardized Mean Difference (SMD). The dataset can be downloaded here. Let's have a look at the data first.

Author	Depression	Anxiety	Depression_var	Anxiety_var	Covariance
Aguilar et al. (2002)	0.5491483	0.5709870	0.0039844	0.0041291	0.0018822
Bailey et al. (1995)	0.2729480	0.4298583	0.0042041	0.0042093	0.0010175
Baldwin et al. (1996)	0.4433041	0.1844816	0.0039062	0.0039441	0.0015389
Bourne et al. (1994)	0.6201276	0.5928386	0.0040589	0.0041520	0.0012331
Brennan et al. (2004)	0.5728169	0.3782552	0.0048733	0.0051134	0.0015471
Carpenter et al. (1997)	0.5494935	0.4670941	0.0043259	0.0044725	0.0016574
Coleman et al. (1996)	0.5120421	0.4309947	0.0030348	0.0030783	0.0015312
Cunningham et al. (2003)	0.5215392	0.5161226	0.0014564	0.0015132	0.0004050
Davis et al. (1992)	0.4668021	0.6286930	0.0016345	0.0016932	0.0005978
Diaz et al. (1997)	0.5293446	0.4607610	0.0019213	0.0019438	0.0010211
Diaz et al. (2007)	0.4789913	0.4863009	0.0040430	0.0041583	0.0014075
Evans et al. (2002)	0.5547111	0.4584910	0.0040097	0.0040110	0.0013359
Figueroa et al. (1998)	0.3793597	0.2974129	0.0032409	0.0032986	0.0010344
Fisher et al. (2000)	0.5091362	0.6086920	0.0021402	0.0022283	0.0006351
Goodwin et al. (1998)	0.4523533	0.3478796	0.0040448	0.0041384	0.0013152
Graves et al. (1996)	0.6104927	0.6238620	0.0039071	0.0039521	0.0011870
Hall et al. (2008)	0.5480115	0.2956450	0.0032744	0.0032750	0.0007267
Hawkins et al. (2006)	0.2995766	0.5129779	0.0039187	0.0039035	-0.0000098
Jones et al. (2003)	0.5985606	0.4107096	0.0028876	0.0028936	0.0009337
Klein et al. (2009)	0.4530782	0.3264904	0.0023158	0.0023291	0.0011682
Lewis et al. (2000)	0.4751269	0.5014020	0.0043074	0.0043173	0.0015122
Mack et al. (2001)	0.5120740	0.5688421	0.0134946	0.0136358	0.0045629
Martinez et al. (2001)	0.5342118	0.4347309	0.0027238	0.0027464	0.0011858
Maxwell et al. (1991)	0.4044607	0.4043562	0.0069526	0.0070016	0.0017507
McDaniel et al. (1995)	0.6780380	0.5386545	0.0034308	0.0034670	0.0011828
McGee et al. (1988)	0.5216277	0.4057494	0.0022297	0.0022409	0.0009981
Morrison et al. (1996)	0.5139628	0.4693981	0.0040549	0.0041275	0.0010106
Payne et al. (2004)	0.5659932	0.6211478	0.0026812	0.0027109	0.0009760
Price et al. (1999)	0.5963063	0.4624809	0.0009759	0.0010004	0.0004464
Rodgers et al. (1998)	0.4831117	0.3621403	0.0040413	0.0041181	0.0017576
Stanton et al. (2003)	0.3715120	0.3740849	0.0041318	0.0044545	0.0012427
Tait et al. (1997)	0.5119991	0.3535861	0.0014935	0.0015032	0.0007319
Thomson et al. (1998)	0.6422394	0.6211067	0.0020051	0.0020739	0.0007342

(continued)

Author	Depression	Anxiety	Depression_var	Anxiety_var	Covariance
Vasquez et al. (2007)	0.3259087	0.4332810	0.0057361	0.0057322	0.0021250
Wallace et al. (1999)	0.3595851	0.3694884	0.0026719	0.0026886	0.0012311
Whittaker et al. (2008)	0.5677016	0.5137125	0.0041935	0.0042120	0.0019380
Zimmermann et al. (2001)	0.4519358	0.4344271	0.0040080	0.0040071	0.0016929

```
dat.da
```

As we can see, the dataset contains the effect sizes on both Depression and Anxiety, along with the variances v_k for both effect sizes. There is also a column called `Covariance` in which the covariance between Depression and Anxiety reported in each study is stored.

A common problem is that the **covariance or correlation** between two outcomes is **not reported in original studies**. If this is the case, we have to estimate the covariance using a reasonable assumption on the correlation between the outcomes. Let us assume that we would not know the covariance in each study yet. How can we estimate it? A good way is to look for previous literature reporting the correlation between the two outcomes, optimally in the same kind of context we are synthesizing. Let us say we found in the literature that Depression and Anxiety are highly correlated in post-tests of psychotherapy trials, with $r_{D,A} \approx 0.6$. We could then approximate the covariance for each study k by using $cov_k = SD_{D_k} \times SD_{A_k} \times r_{D,A}$. We can do this in *R* like this:

```
estimated.cov <- sqrt(dat.da$Depression_var) * sqrt(dat.da$Anxiety_var) * 0.6
```

Where we take the square root because $SD = \sqrt{Var}$.

13.2.1 Specifying the Model

To specify the model for the multivariate meta-analysis, we do not have to follow the TSSEM procedure programmatically, nor do we have to specify the RAM matrices. For such a relatively simple model, we can use the `meta` function in `metaSEM` to pool the studies, specify and fit the model all at once. **Here are the most important parameters we have to specify for the `meta()` function.**

Parameter	Description
y	The columns of our dataset which contain the effect size data. In a multivariate meta-analysis, we have to combine the effect size columns we want to include using cbind().
v	The columns of our dataset which contain the variance for the effect size. In a multivariate meta-analysis, we have to combine the variance columns we want to include using cbind(). We also have to include the column including the covariance between the effect sizes. The structure of the cbind() function is cbind(variance_1, covariance, variance_2)
data	The dataset in which the effect sizes and their variances are stored.

I will save my meta-analysis model as `m.mv`. My code then looks like this:

```
m.mv <- meta(y = cbind(Depression, Anxiety),
             v = cbind(Depression_var, Covariance, Anxiety_var),
             data = dat.da)
```

To get the results, i have to plug the `m.mv` object into the `summary()` function.

```
summary(m.mv)
```

```
Call:
meta(y = cbind(Depression, Anxiety), v = cbind(Depression_var,
    Covariance, Anxiety_var), data = dat.da)

95% confidence intervals: z statistic approximation
Coefficients:
              Estimate   Std.Error      lbound      ubound z value  Pr(>|z|)
Intercept1  0.50321391  0.01466266  0.47447563  0.53195220 34.3194 < 2.2e-16 ***
Intercept2  0.45553567  0.01748527  0.42126518  0.48980617 26.0525 < 2.2e-16 ***
Tau2_1_1    0.00459325  0.00186286  0.00094210  0.00824439  2.4657  0.013675 *
Tau2_2_1    0.00287809  0.00167184 -0.00039865  0.00615483  1.7215  0.085158 .
Tau2_2_2    0.00783290  0.00254639  0.00284206  0.01282374  3.0761  0.002097 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Q statistic on the homogeneity of effect sizes: 223.8427
Degrees of freedom of the Q statistic: 72
P value of the Q statistic: 0

Heterogeneity indices (based on the estimated Tau2):
                             Estimate
Intercept1: I2 (Q statistic)   0.6079
Intercept2: I2 (Q statistic)   0.7220

Number of studies (or clusters): 37
Number of observed statistics: 74
Number of estimated parameters: 5
Degrees of freedom: 69
-2 log likelihood: -140.7221
OpenMx status1: 0 ("0" or "1": The optimization is considered fine.
Other values may indicate problems.)
```

13.2.2 Evaluating the Output

Given that the SEM model is fitted using the **Maximum-Likelihood** algorithm, the first thing we always do is check the OpenMx `status` right at the end of the output. Maximum-Likelihood is an **optimization procedure**, in which parameters are changed iteratively until the optimal solution for the data at hand is found. However, especially with more complex models, it can happen that this optimum is not reached even after many iterations; the Maximum-Likelihood algorithm will then stop and output the parameters values it has approximated so far. However, those values for our model components will then **be very likely** to be **incorrect** and should **not be trusted**. The OpenMx `status` for my model is `0`, which indicates that the Maximum-Likelihood estimation worked fine. However, if the status would have been other than `0` or 1, it would have been necessary for me to rerun the fitting process with this code:

```
rerun(m.mv)
```

In the output, the two pooled effect sizes are shown as `Intercept1` and `Intercept2`. The effect sizes are numbered be the order in which we inserted them into our call to `meta()`. We can see that the pooled effect sizes are $SMD_D = 0.50$ for Depression and $SMD_A = 0.46$. Both effect sizes are significant. Under `Heterogeneity indices`, we can also see the values of I^2 (see Chapter 6), which are $I^2_D = 61\%$ and $I^2_A = 72\%$, indicating substantial between-study heterogeneity.

The direct estimates of the between study heterogeneity τ^2 are also provided. We see that there are not only two estimates, but three. To understand what this means, we can extract the `"random"` values from the `m.mv` object first.

```
tau.coefs <- coef(m.mv, select = "random")
```

Then, we can use the `vec2symMat()` function on the coefficients to create a matrix. We then give the matrix rows and columns the names of our variables, Depression and Anxiety

```
tc.mat <- metaSEM::vec2symMat(tau.coefs)
dimnames(tc.mat)[[1]] <- dimnames(tc.mat)[[2]] <- c("Depression", "Anxiety")

tc.mat
```

```
##             Depression     Anxiety
## Depression 0.004593246 0.002878092
## Anxiety    0.002878092 0.007832899
```

We now understand better what the three τ^2 values mean: they represent the **between study "variance"/heterogeneity** at the diagonal of the matrix; in the other two fields, they are the estimate for the covariance between Depression and Anxiety. Given that the covariance is just an unstandardized version of the **correlation**, we can also calculate the correlations using the `cov2cor()` function.

```
cov2cor(tc.mat)
```

```
##            Depression   Anxiety
## Depression  1.0000000 0.4798257
## Anxiety     0.4798257 1.0000000
```

We see that, quite logically, the correlations in the diagonal elements of the matrix are 1. The correlation between effect sizes on Depression and effect sizes on Anxiety is $r_{D,A} = 0.48$. This is an interesting finding: is shows that there is a **positive association between a treatment's effect on Depression and its effect on Anxiety**. Treatments which have high effects on Depression are very likely to have high effects on Anxiety too. It is of note that the **confidence intervals** presented in the output are so-called **Wald-type** intervals. Such Wald-type intervals have been found to be prone to inacurateness, especially in small samples (DiCiccio and Efron, 1996). It may thus be valuable to constuct confidence intervals in another way, using so-called **likelihood-based** confidence intervals. We can get these CIs by rerunning the `meta()` function and additionally specifying `intervals.type = "LB"`

```
m.mv <- meta(y = cbind(Depression, Anxiety),
             v = cbind(Depression_var, Covariance, Anxiety_var),
             data = dat.da,
             intervals.type = "LB")
```

We have already seen that the output for our `m.mv` contains estimates of the **between-study heterogeneity** τ^2. If

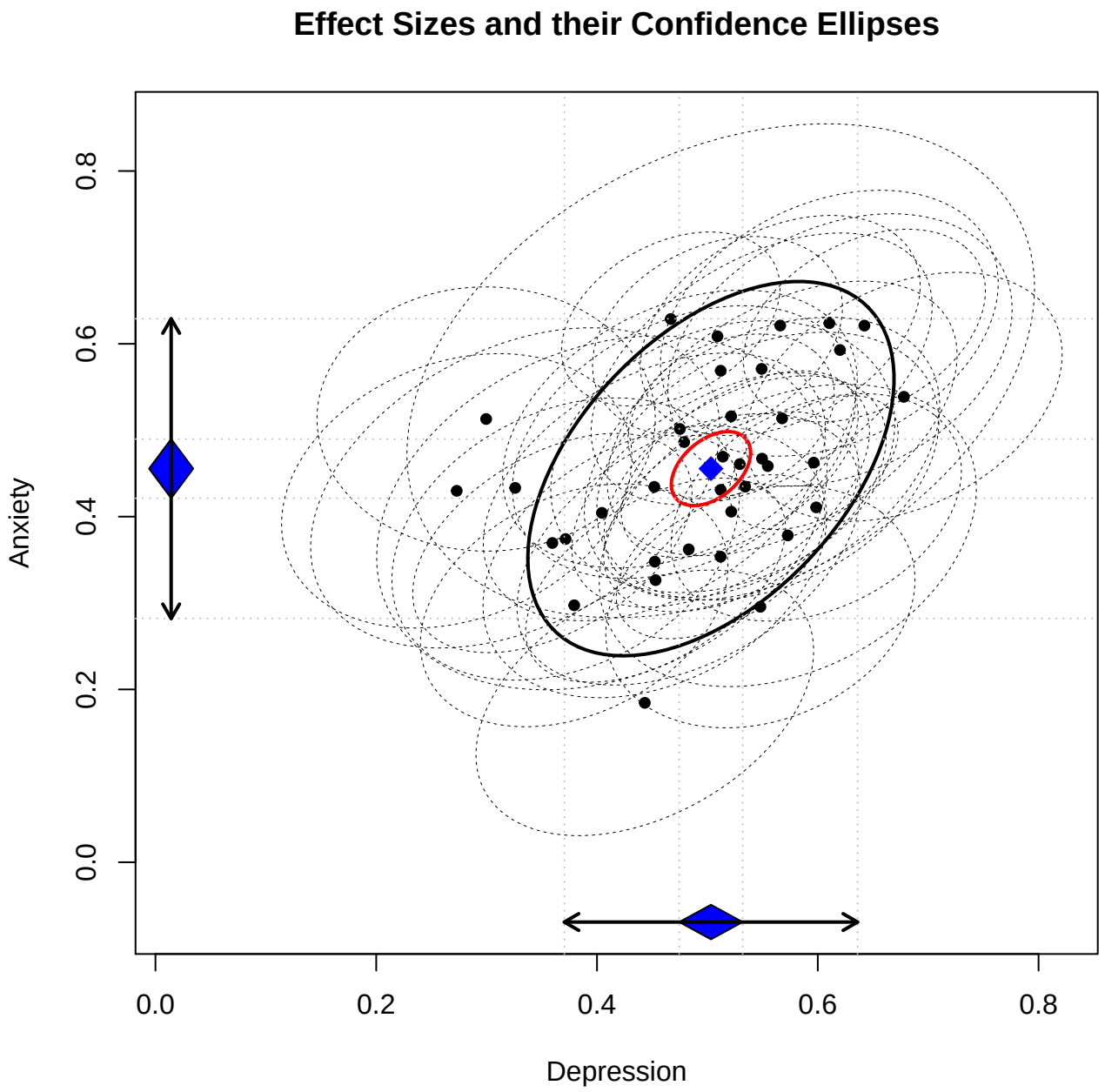
we recall our knowledge of the meta-analysis model, we can therefore conclude that the model we just fitted is a **random-effects model**. The `meta()` function uses a random-effects model automatically, because the fixed-effects model is only a special case of the random-effects model in which we set $\tau^2 = 0$. From the I^2 in our output, we can conclude that the random-effects model is indeed indicated; however, if we wanted to fit a fixed-effect model, we can redo the analysis, this time adding the parameter `RE.constraints = matrix(0, nrow=2, ncol=2)`, which creates a matrix of zeros of the same structure as the `tc.mat` matrix, constraining our `tau2` values to zero:

```
m.mv <- meta(y = cbind(Depression, Anxiety),
             v = cbind(Depression_var, Covariance, Anxiety_var),
             data = dat.da,
             RE.constraints = matrix(0, nrow=2, ncol=2))
```

13.2.3 Plotting the Model

To plot the multivariate meta-analysis model, we can simply use the `plot()` function. I will make some additional specifications here to change the appearance of the plot slightly. If you want to see all styling options, you can paste `?metaSEM::plot.meta()` into the Console and then hit Enter.

```
plot(m.mv,
     axis.labels = c("Depression", "Anxiety"),
     randeff.ellipse.col = "black",
     univariate.arrows.col = "black",
     univariate.polygon.col = "blue")
```



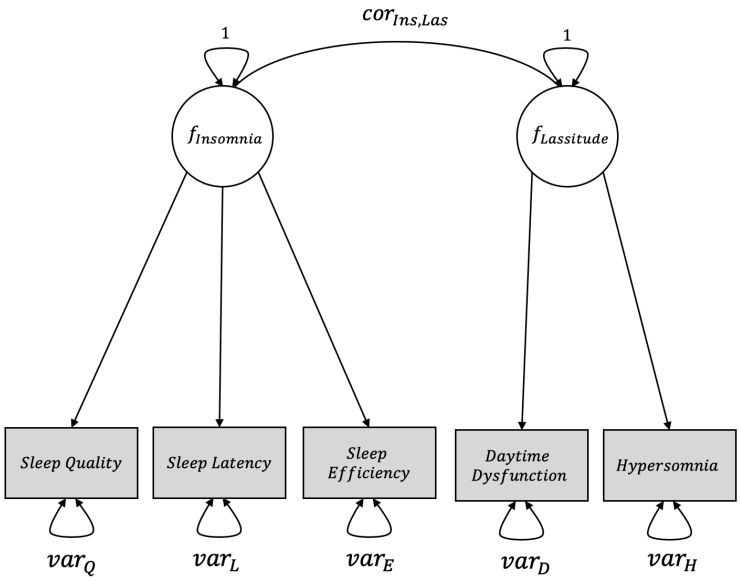
Let us go through the plot one by one:

- We see that the plot has two axes: the x-axis, displaying the **effect size on Depression**, and the y-axis displaying the **effect size on Anxiety**.
- We also see the pooled effect and its 95% CI for both outcomes, symbolized by the blue diamond and the black arrows.
- In the middle of the plot, the pooled effect for both variables is represented by another blue diamond.
- The red ellipse represents the **95% confidence interval* ellipse** for out pooled effect sizes.
- The black ellipse represents the space in which we expect 95% of all studies to fall based on the random-effects model.
- The black dots represent the **individual studies**, and the dashed lines are the 95% CIs of the individual studies.

13.3 Confirmatory Factor Analysis

Confirmatory Factor Analysis (**CFA**) is a popular SEM method in which one specifies how observed variables relate to assumed latent variables (Thompson, 2004). CFA is often used to evaluate the psychometric properties of questionnaires or other assessments. It allows researchers to determine if the variables they assess indeed measure one or more latent variables, and how these latent variables relate to each other. For frequently used questionnaires, there are often many studies which report the correlations between the different items of the questionnaires. Such data can be used for meta-analytic SEM, which allows us to **evaluate which latent variable (or factor) structure** is the **most appropriate** based on all available evidence.

In this example, let us assume that we want to confirm the latent factor structure of a **questionnaire for sleep problems**. The questionnaire is assumed to measure two distinct latent variables, which together characterize sleep problems: **insomnia** and **lassitude** (in fact Koffel and Watson (2009) argue that sleep complaints do indeed follow this structure). For our questionnaire, we have created a fictitious dataset, `dat.cfa` (based on the `Digman97` data in the `metaSEM` package) which contains 14 independent studies which report the correlations between the different symptoms of sleep complaints that our questionnaire measures: **sleep quality**, **sleep latency**, **sleep efficiency**, **daytime dysfunction** and **hypersomnia** (i.e., sleeping too much; the dataset can be downloaded here). We assume that the first three symptoms are related, because they all measure insomnia as a latent variable, whereas daytime dysfunction and hypersomnia are related because they are symptoms of the lassitude factor. **The proposed structure represented as a graphical model looks like this** (please note that the labels are somewhat idiosyncratic to make identifying the relevant components of the model easier later on):



13.3.1 Setting up our data

Let us first have a look at the data we want to use for model fitting. The `dat.cfa` dataset i will use here has a **special structure**: it is a so-called `list`, containing (1) another `list` of matrices and (2) a vector. A list is a very versatile *R* object, which allows to bind together different objects in one single big object. Lists can be accessed like `data.frames` by using the $ operator. By using the `names()` function, we can see the names of the objects in the list.

```
names(dat.cfa)
```

```
## [1] "data" "n"
```

We see that the list contains two elements, our `data` and the n (sample size) of each study. The `data` object is itself a list, so we can get the names of its contents using the `names()` function, or can display single contents of it through the $ operator.

```
names(dat.cfa$data)
```

```
##  [1] "Coleman et al. (2003)"  "Salazar et al. (2008)"
##  [3] "Newman et al. (2016)"   "Delacruz et al. (2009)"
##  [5] "Wyatt et al. (2002)"    "Pacheco et al. (2016)"
##  [7] "Riggs et al. (2015)"    "Guerrero et al. (2010)"
##  [9] "Brewer et al. (2007)"   "Bridges et al. (1999)"
## [11] "Esparza et al. (2019)"  "Charles et al. (1999)"
## [13] "Cooke et al. (2000)"    "Ponce et al. (2017)"
```

```
dat.cfa$data$`Coleman et al. (2003)`
```

```
##           Quality Latency Efficiency DTDysf HypSomnia
## Quality      1.00    0.62       0.41  -0.48      0.00
## Latency      0.62    1.00       0.59  -0.10      0.35
```

```
## Efficiency    0.41    0.59       1.00   0.27      0.41
## DTDysf       -0.48   -0.10       0.27   1.00      0.37
## HypSomnia     0.00    0.35       0.41   0.37      1.00
```

We see that the `data` list contains 14 elements for each of the 14 included studies. A closer look at the `Coleman et al. (2003)` study reveals that the data are stored as **correlation matrices** for our five observed variables. The Coleman et al. (2003) study contains correlation data for all variable combinations; however we can also allow for some studies to have missings (coded as NA) on some of the fields **because the meta-analytic SEM approach can handle missing data** at least to some extent.

Before we proceed, let us quickly show how you can construct such a list yourself. Let us assume that you have the correlation matrices stored in separate `data.frames`, which you imported into R (Chapter 3). The important part here is that the **column names**, and their **order** should already be **the same** in all of the `data.frames`. Let us say i have two data frames containing the correlation data, `df1` and `df2`, which look like this:

```
df1
```

```
##   Quality Latency Efficiency DTDysf HypSomnia
## 1    1.00    0.62       0.41  -0.48      0.00
## 2    0.62    1.00       0.59  -0.10      0.35
## 3    0.41    0.59       1.00   0.27      0.41
## 4   -0.48   -0.10       0.27   1.00      0.37
## 5    0.00    0.35       0.41   0.37      1.00
```

```
df2
```

```
##   Quality Latency Efficiency DTDysf HypSomnia
## 1    1.00    0.39       0.53  -0.30     -0.05
## 2    0.39    1.00       0.59   0.07      0.44
## 3    0.53    0.59       1.00   0.09      0.22
## 4   -0.30    0.07       0.09   1.00      0.45
## 5   -0.05    0.44       0.22   0.45      1.00
```

I can transform these `data.frames` into a matrix using the `as.matrix()` function. Because we want the rows and columns to contain the names of the variables, i have to specify the **dimension names** (`dimnames`) of the matrices. The first dimension represents the rows, the second the columns. I can do this in *R* like this:

```
# Convert to matrices
mat1 <- as.matrix(df1)
mat2 <- as.matrix(df2)

# Set the dimension names
dimnames(mat1)[[1]] <- dimnames(mat1)[[2]] <- c("Quality", "Latency", "Efficiency",
                                                "DTDysf", "HypSomnia")
dimnames(mat2)[[1]] <- dimnames(mat2)[[2]] <- c("Quality", "Latency", "Efficiency",
                                                "DTDysf", "HypSomnia")

# Print the matrices
mat1
##            Quality Latency Efficiency DTDysf HypSomnia
## Quality       1.00    0.62       0.41  -0.48      0.00
```

```
## Latency       0.62    1.00       0.59  -0.10      0.35
## Efficiency    0.41    0.59       1.00   0.27      0.41
## DTDysf       -0.48   -0.10       0.27   1.00      0.37
## HypSomnia     0.00    0.35       0.41   0.37      1.00
mat2
##            Quality Latency Efficiency DTDysf HypSomnia
## Quality       1.00    0.39       0.53  -0.30     -0.05
## Latency       0.39    1.00       0.59   0.07      0.44
## Efficiency    0.53    0.59       1.00   0.09      0.22
## DTDysf       -0.30    0.07       0.09   1.00      0.45
## HypSomnia    -0.05    0.44       0.22   0.45      1.00
```

We can then bind these matrices together in a `list`, and then give the list elements a name using the `names()` function.

```
matrix.list <- list(mat1, mat2)
names(matrix.list) <- c("Study1", "Study2")
```

To do the modeling, we also need the total sample size N of each study. We only have to create a numeric vector which contains the sample sizes in the **same order** as the objects in our list. We can then create one big list, containing both the list with our correlation matrix data, and the sample sizes for each study, again using the `list()` function.

```
n <- c(205, 830)
all.data <- list(matrix.list, n)
```

That is it! We can now proceed to specifying our model for the `dat.cfa` data.

13.3.2 Model Specification

To specify our CFA model, we will have to use the **RAM specification** and the **TSSEM** procedure we mentioned before (Chapter 13.1). The `metaSEM` package directly follows the TSSEM approach; in fact, it even has two different functions for the two stages, `tssem1` and `tssem2`. The first pools our correlation matrices across all studies, and the second fits the proposed model to the data.

13.3.2.1 Stage 1

At the first stage, we pool the correlation matrices using the `tssem1()` function. There are 3-4 important parameters we have to specify in the function.

Parameter	Description
Cov	A list of correlation matrices we want to pool. Please note that all correlation matrices in the list need to have an identical structure.
n	A numeric vector containing the sample sizes of each study, in the same order as the matrices contained in Cov.
method	This specifies if we want to use a fixed-effect model ('FEM') or random-effects model ('REM').
RE.type	When a random-effects model is used, this specifies which of the random-effects should be estimated. The default is 'Symm', which estimates all tau-squared values, including the covariances. When set to 'Diag', only the diagonal elements of the random-effects matrix are estimated. This means that we assume that the random effects are independent. Although using 'Diag' is a simplification of our model, it is often preferable, because less parameters have to be estimated. This particularly makes sense when the number of variables is high and/or the number of studies is low.

For this step, i will assume a random-effects model, and use `RE.type = "Diag"`. I will save the model as `cfa1`, and then call the `summary()` function on it to retrieve the output. Here is the code for that:

```
cfa1 <- tssem1(dat.cfa$data,
               dat.cfa$n,
               method="REM",
               RE.type = "Diag")

summary(cfa1)
```

```
Call:
meta(y = ES, v = acovR, RE.constraints = Diag(paste0(RE.startvalues,
    "*Tau2_", 1:no.es, "_", 1:no.es)), RE.lbound = RE.lbound,
    I2 = I2, model.name = model.name, suppressWarnings = TRUE,
    silent = silent, run = run)

95% confidence intervals: z statistic approximation
Coefficients:
               Estimate   Std.Error      lbound      ubound z value  Pr(>|z|)
Intercept1   0.38971904  0.05429257  0.28330757  0.49613052  7.1781 7.068e-13 ***
Intercept2   0.43265876  0.04145136  0.35141559  0.51390194 10.4377 < 2.2e-16 ***
Intercept3   0.04945626  0.06071079 -0.06953470  0.16844722  0.8146   0.41529
Intercept4   0.09603706  0.04478711  0.00825593  0.18381818  2.1443   0.03201 *
Intercept5   0.42724237  0.03911621  0.35057601  0.50390873 10.9224 < 2.2e-16 ***
Intercept6   0.11929310  0.04106204  0.03881299  0.19977321  2.9052   0.00367 **
Intercept7   0.19292421  0.04757961  0.09966988  0.28617853  4.0548 5.018e-05 ***
Intercept8   0.22690159  0.03240893  0.16338126  0.29042192  7.0012 2.538e-12 ***
Intercept9   0.18105563  0.04258855  0.09758361  0.26452765  4.2513 2.126e-05 ***
Intercept10  0.43614968  0.03205961  0.37331400  0.49898536 13.6043 < 2.2e-16 ***
Tau2_1_1     0.03648373  0.01513055  0.00682838  0.06613907  2.4113   0.01590 *
Tau2_2_2     0.01963097  0.00859789  0.00277942  0.03648253  2.2832   0.02242 *
Tau2_3_3     0.04571437  0.01952285  0.00745030  0.08397845  2.3416   0.01920 *
```

```
Tau2_4_4     0.02236122  0.00995083  0.00285794  0.04186449  2.2472   0.02463 *
Tau2_5_5     0.01729073  0.00796404  0.00168149  0.03289996  2.1711   0.02992 *
Tau2_6_6     0.01815482  0.00895896  0.00059557  0.03571407  2.0264   0.04272 *
Tau2_7_7     0.02604880  0.01130265  0.00389601  0.04820160  2.3047   0.02119 *
Tau2_8_8     0.00988761  0.00513713 -0.00018098  0.01995620  1.9247   0.05426 .
Tau2_9_9     0.01988243  0.00895052  0.00233973  0.03742514  2.2214   0.02633 *
Tau2_10_10   0.01049222  0.00501578  0.00066148  0.02032297  2.0918   0.03645 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Q statistic on the homogeneity of effect sizes: 1220.333
Degrees of freedom of the Q statistic: 130
P value of the Q statistic: 0

Heterogeneity indices (based on the estimated Tau2):
                             Estimate
Intercept1: I2 (Q statistic)   0.9326
Intercept2: I2 (Q statistic)   0.8872
Intercept3: I2 (Q statistic)   0.9325
Intercept4: I2 (Q statistic)   0.8703
Intercept5: I2 (Q statistic)   0.8797
Intercept6: I2 (Q statistic)   0.8480
Intercept7: I2 (Q statistic)   0.8887
Intercept8: I2 (Q statistic)   0.7669
Intercept9: I2 (Q statistic)   0.8590
Intercept10: I2 (Q statistic)  0.8193

Number of studies (or clusters): 14
Number of observed statistics: 140
Number of estimated parameters: 20
Degrees of freedom: 120
-2 log likelihood: -112.4196
OpenMx status1: 0 ("0" or "1": The optimization is considered fine.
Other values may indicate problems.)
```

A look at the OpenMx `status1` shows the the model estimates are trustworthy. To make the results more easily digestable, we can extract the **fixed effects** (our estimated pooled correlations) using the `coef()` function. We then make a symmetrical matrix out of the coefficients using `vec2symMat()`, and add the dimension names for easier interpretation.

```
# Extract the fixed coefficients (correlations)
fixed.coefs <- coef(cfa1, "fixed")

# Make a symmetric matrix
fc.mat <- vec2symMat(fixed.coefs, diag = FALSE)

# Label rows and columns
dimnames(fc.mat)[[1]] <- dimnames(fc.mat)[[2]] <- c("Quality", "Latency", "Efficiency",
```

```
                                                    "DTDysf", "HypSomnia")

fc.mat
```

```
##                Quality   Latency Efficiency     DTDysf  HypSomnia
## Quality    1.00000000 0.3897190  0.4326588 0.04945626 0.09603706
## Latency    0.38971904 1.0000000  0.4272424 0.11929310 0.19292421
## Efficiency 0.43265876 0.4272424  1.0000000 0.22690159 0.18105563
## DTDysf     0.04945626 0.1192931  0.2269016 1.00000000 0.43614968
## HypSomnia  0.09603706 0.1929242  0.1810556 0.43614968 1.00000000
```

We can now see the **pooled correlation matrix** for our our variables. Looking back the model output, we also see that all correlation coefficients are **significant** ($p < 0.05$), except one: the **correlation between sleep quality and daytime dysfunction** was not significant. From the perspective of our model, this makes sense, because we expect these variables to load on different factors. We also find that the I^2 values for the different estimates are very, very large ($76\% - 93\%$). We may therefore also have a look if a model fitted in different subclusters of studies might reduce the amount of heterogeneity we find, and if this translates to different SEM fits on stage 2.

13.3.2.2 Stage 2

After pooling the correlation matrices, it is now time to determine if our proposed factor model does indeed fit the data. To specify our model, we have to resort to the **RAM** formulation this time, and specify the $\mathbf{A}$, $\mathbf{S}$ and $\mathbf{F}$ matrices. To fill the fields, it is often easier to construct an empty matrix first. In the rows and columns, the matrices will not only contain the observed variables, but also the **latent variables** we want to estimate, `f_Insomnia` and `f_Lassitude`. Here is how we can create a zero matrix as a blueprint:

```
# Create vector of column/row names
dims <- c("Quality", "Latency", "Efficiency", "DTDysf",
          "HypSomnia", "f_Insomnia", "f_Lassitude")

# Create 7x7 matrix of zeros
mat <- matrix(rep(0, 7*7), nrow = 7, ncol = 7)

# Label the rows and columns
dimnames(mat)[[1]] <- dimnames(mat)[[2]] <- dims

mat
```

```
##             Quality Latency Efficiency DTDysf HypSomnia f_Insomnia
## Quality           0       0          0      0         0          0
## Latency           0       0          0      0         0          0
## Efficiency        0       0          0      0         0          0
## DTDysf            0       0          0      0         0          0
## HypSomnia         0       0          0      0         0          0
## f_Insomnia        0       0          0      0         0          0
## f_Lassitude       0       0          0      0         0          0
##             f_Lassitude
```

```
## Quality              0
## Latency              0
## Efficiency           0
## DTDysf               0
## HypSomnia            0
## f_Insomnia           0
## f_Lassitude          0
```

$\mathbf{A}$ Matrix

In the $\mathbf{A}$ matrix, we specify the asymmetrical (i.e. single) arrows in our model. The logic, again, is that the arrow starts at the **column variable** and ends where the column meets with the entry of the **row variable**. All other fields which do not represent arrows are filled with `0`. We specify that an arrow has to be "estimated" by providing a character string. This character string starts with a **starting value** for the optimiztion procedure (usually somewhere between 0.1 and 0.3) followed by *. After the * symbol, we specify a **label** for the value. If two fields in the $\mathbf{A}$ matrix have the same label, this means that we assume that the fields have the **same value**. We assume a starting value of 0.3 for all estimated arrows, and label the fields according to the graph from above. We can do this like this:

```
A <- matrix(c(0, 0, 0, 0, 0, "0.3*Ins_Q", 0          ,
              0, 0, 0, 0, 0, "0.3*Ins_L", 0          ,
              0, 0, 0, 0, 0, "0.3*Ins_E", 0          ,
              0, 0, 0, 0, 0, 0          , "0.3*Las_D",
              0, 0, 0, 0, 0, 0          , "0.3*Las_H",
              0, 0, 0, 0, 0, 0          , 0          ,
              0, 0, 0, 0, 0, 0          , 0
              ), nrow = 7, ncol = 7, byrow=TRUE)

# Label columns and rows
dimnames(A)[[1]] <- dimnames(A)[[2]] <- dims

A
```

```
##             Quality Latency Efficiency DTDysf HypSomnia f_Insomnia
## Quality     "0"     "0"     "0"        "0"    "0"       "0.3*Ins_Q"
## Latency     "0"     "0"     "0"        "0"    "0"       "0.3*Ins_L"
## Efficiency  "0"     "0"     "0"        "0"    "0"       "0.3*Ins_E"
## DTDysf      "0"     "0"     "0"        "0"    "0"       "0"
## HypSomnia   "0"     "0"     "0"        "0"    "0"       "0"
## f_Insomnia  "0"     "0"     "0"        "0"    "0"       "0"
## f_Lassitude "0"     "0"     "0"        "0"    "0"       "0"
##             f_Lassitude
## Quality     "0"
## Latency     "0"
## Efficiency  "0"
## DTDysf      "0.3*Las_D"
## HypSomnia   "0.3*Las_H"
## f_Insomnia  "0"
## f_Lassitude "0"
```

Looks good so far. The last step is to plug the A matrix into the `as.mxMatrix()` function to make it usable for the later step.

```
A <- as.mxMatrix(A)
```

S Matrix

In the S matrix, we specify the variances we want to estimate. We want to estimate the variance for all observed variables, as well as the correlation between our latent variables. We set the correlation of our latent variables with themselves to be 1. We use a starting value of 0.2 for the variances in the observed variables, and 0.3 for the correlations. Here is how we construct the matrix:

```
# Make a diagonal matrix for the variances
Vars <- Diag(c("0.2*var_Q", "0.2*var_L", "0.2*var_E", "0.2*var_D", "0.2*var_H"))

# Make the matrix for the latent variables
Cors <- matrix(c(1, "0.3*cor_InsLas",
                 "0.3*cor_InsLas", 1),
               nrow=2, ncol=2)

# Combine
S <- bdiagMat(list(Vars, Cors))

# Label columns and rows
dimnames(S)[[1]] <- dimnames(S)[[2]] <- dims

S
```

```
##             Quality     Latency     Efficiency  DTDysf      HypSomnia
## Quality     "0.2*var_Q" "0"         "0"         "0"         "0"
## Latency     "0"         "0.2*var_L" "0"         "0"         "0"
## Efficiency  "0"         "0"         "0.2*var_E" "0"         "0"
## DTDysf      "0"         "0"         "0"         "0.2*var_D" "0"
## HypSomnia   "0"         "0"         "0"         "0"         "0.2*var_H"
## f_Insomnia  "0"         "0"         "0"         "0"         "0"
## f_Lassitude "0"         "0"         "0"         "0"         "0"
##             f_Insomnia       f_Lassitude
## Quality     "0"              "0"
## Latency     "0"              "0"
## Efficiency  "0"              "0"
## DTDysf      "0"              "0"
## HypSomnia   "0"              "0"
## f_Insomnia  "1"              "0.3*cor_InsLas"
## f_Lassitude "0.3*cor_InsLas" "1"
```

And again, we transform the matrix using `as.mxMatrix()`.

```
S <- as.mxMatrix(S)
```

$\mathbf{F}$ Matrix

The $\mathbf{F}$ matrix is quite easily specified: in the diagonal elements of **observed variables**, we fill in 1. Elsewhere we specify 0. In the $\mathbf{F}$ matrix, we **only select the rows in which at least on element is not zero**.

```
# Construct diagonal matrix
F <- Diag(c(1, 1, 1, 1, 1, 0, 0))

# Only select non-null rows
F <- F[1:5,]

# Specify row and column labels
dimnames(F)[[1]] <- dims[1:5]
dimnames(F)[[2]] <- dims

F
```

```
##            Quality Latency Efficiency DTDysf HypSomnia f_Insomnia
## Quality          1       0          0      0         0          0
## Latency          0       1          0      0         0          0
## Efficiency       0       0          1      0         0          0
## DTDysf           0       0          0      1         0          0
## HypSomnia        0       0          0      0         1          0
##            f_Lassitude
## Quality              0
## Latency              0
## Efficiency           0
## DTDysf               0
## HypSomnia            0
```

```
F <- as.mxMatrix(F)
```

Model fitting

Now, it is time to fit our proposed model to the data. To do this, we use the `tssem2()` function. We only have to provide the **stage 1 model** `cfa1`, the three matrices, and specify `diag.constraints=FALSE`, because we are not fitting a mediation model. I save the resulting object as `cfa2` and then access it using `summary()`.

```
cfa2 <- tssem2(cfa1,
               Amatrix = A,
               Smatrix = S,
               Fmatrix = F,
```

```
               diag.constraints = FALSE)

summary(cfa2)
```

```
##
## Call:
## wls(Cov = pooledS, aCov = aCov, n = tssem1.obj$total.n, Amatrix = Amatrix,
##     Smatrix = Smatrix, Fmatrix = Fmatrix, diag.constraints = diag.constraints,
##     cor.analysis = cor.analysis, intervals.type = intervals.type,
##     mx.algebras = mx.algebras, model.name = model.name, suppressWarnings = suppressWarnings,
##     silent = silent, run = run)
##
## 95% confidence intervals: z statistic approximation
## Coefficients:
##            Estimate Std.Error   lbound   ubound z value  Pr(>|z|)
## Las_D      0.679955  0.075723 0.531541 0.828370  8.9795 < 2.2e-16 ***
## Ins_E      0.760455  0.061964 0.639009 0.881901 12.2726 < 2.2e-16 ***
## Las_H      0.641842  0.072459 0.499825 0.783859  8.8580 < 2.2e-16 ***
## Ins_L      0.590630  0.052649 0.487439 0.693821 11.2182 < 2.2e-16 ***
## Ins_Q      0.569435  0.052425 0.466684 0.672187 10.8619 < 2.2e-16 ***
## cor_InsLas 0.377691  0.047402 0.284785 0.470596  7.9679 1.554e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Goodness-of-fit indices:
##                                                Value
## Sample size                                 4496.0000
## Chi-square of target model                     7.8204
## DF of target model                             4.0000
## p value of target model                        0.0984
## Number of constraints imposed on "Smatrix"     0.0000
## DF manually adjusted                           0.0000
## Chi-square of independence model             501.6766
## DF of independence model                      10.0000
## RMSEA                                          0.0146
## RMSEA lower 95% CI                             0.0000
## RMSEA upper 95% CI                             0.0297
## SRMR                                           0.0436
## TLI                                            0.9806
## CFI                                            0.9922
## AIC                                           -0.1796
## BIC                                          -25.8234
## OpenMx status1: 0 ("0" or "1": The optimization is considered fine.
## Other values indicate problems.)
```

We again see that the OpenMx `status1` is `0`, meaning that the optimization worked fine. We get the estimates for the different paths between the latent variables and the observed symptoms, such as 0.68 for $Lassitude \rightarrow$

$Daytime Dysfunction$ (Las_D). The important part, however, it to check **how well the assumed model fits our data**. We can have a look at the second, third and fourth row of the Goodness-of-fit indices, where we see that the **goodness of fit** test is $\chi^2_{4,4496} = 7.82$, which is **not significant** $p = 0.098$. On the other hand, we see that the **Root Mean Square Error of Approximation** (**RMSEA**; Steiger and Lind (1980)) value is $RMSEA = 0.0146$. As a rule of thumb, a model can be considered to fit the data well when the $RMSEA$ is close to 0.05 and smaller than 0.10. The indices for this model are therefore somewhat **conflicting**, but generally indicate that the model may **not fit all of the data closely**. A potential way to explore this would be to conduct a **subgroup analysis**, in which the model is fitted to subgroups of studies which share some kind of characteristic (e.g. age group, country of origin, etc.) to see if there might be subclusters in which the model fits better than others.

Please be aware that a common problem in SEM studies is that researchers often only focus on their **one proposed model**, and if it fits the data well. If it is found that the model has a close fit for the data, it is often directly assumed that the data prove the underlying structure or theory. This is **problematic**, because for the same data, more than one model can achieve a good fit at the same time. It is therefore necessary to also check for **alternative model hypotheses** and structures. If the alternative model fits the data well too, it becomes less clear if our proposed structure really is the "correct" one.

13.3.2.3 Plotting the Model

After we fit the model, metaSEM makes it quite easy for us to visualize it graphically. To do this, we first have to install and load the semPlot package (Epskamp, 2019).

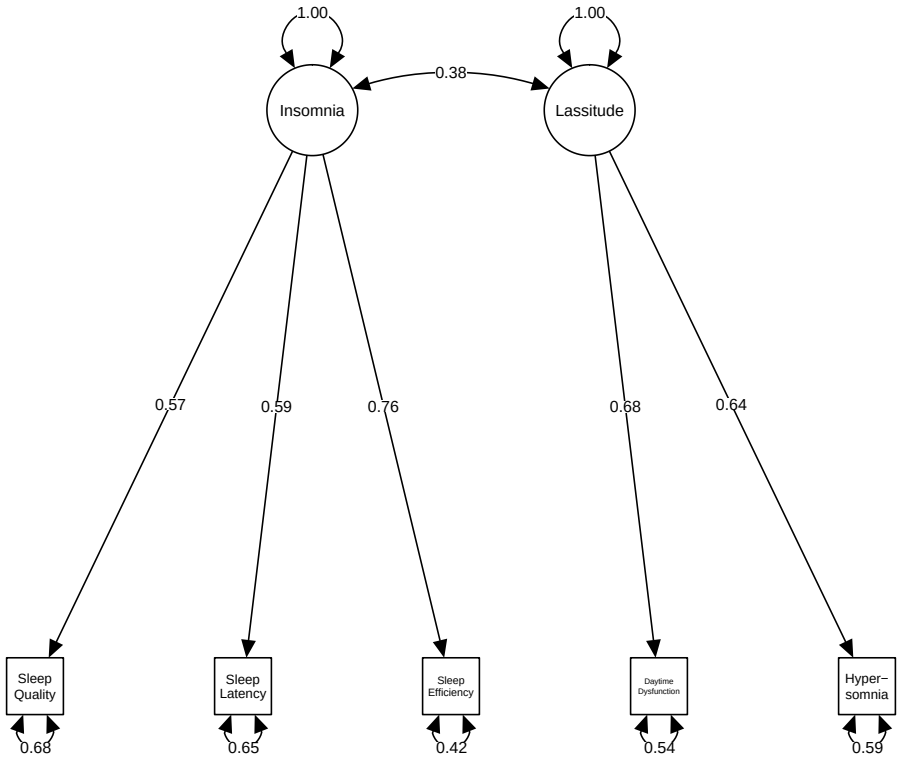
```
library(semPlot)
```

To plot the model, we have to **convert** it into a format that semPlot can use. We can use the meta2semPlot() function to do this.

```
cfa.plot <- meta2semPlot(cfa2)
```

We can then use the semPaths() function in semPlot to generate the plot. This function has many parameters, which you can access by typing ?semPaths into the Console, and then hitting Enter. Here is how my code looks like, and the resulting plot:

```
# Create Plot labels (left to right, bottom to top)
labels <- c("Sleep\nQuality","Sleep\nLatency","Sleep\nEfficiency","Daytime\nDysfunction",
            "Hyper-\nsomnia","Insomnia", "Lassitude")

# Plot
semPaths(cfa.plot,
         whatLabels = "est",
         edge.color = "black",
         nodeLabels = labels)
```


1.00
1.00
0.38
Insomnia
Lassitude
0.57
0.59
0.76
0.68
0.64
Sleep Quality
Sleep Latency
Sleep Efficiency
Daytime Dysfunction
Hyper−somnia
0.68
0.65
0.42
0.54
0.59

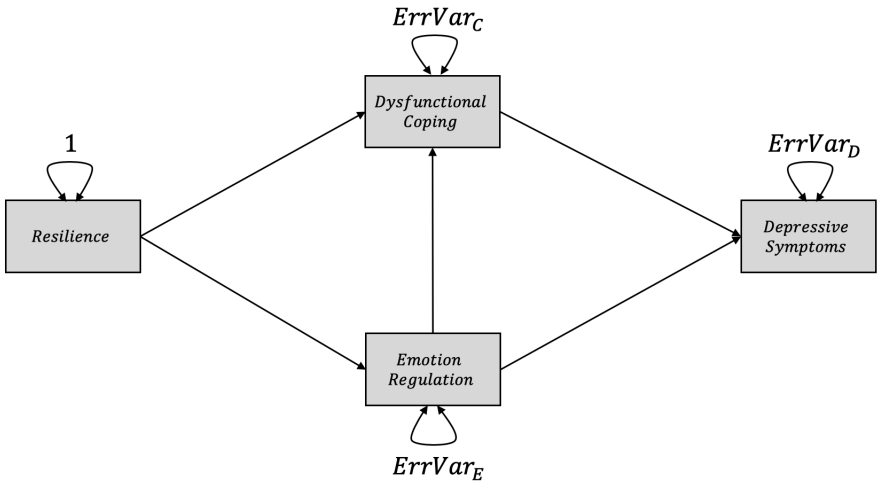
13.4 Mediation

In mediation models (Baron and Kenny, 1986), we want to examine if a **direct effect** from one variable to another is **mediated** by an **intervening** or **mediator variable**. If a total mediation exists, we assume that a variable X has an effect on a variable Y only because X influences a mediating variable Z, which itself affects Y. Mediation is used in many fields, especially when we are interested in the **mechanisms** behind some relationship between two variables. However, it is important to mention that mediation models should always be based on a **theoretically** and **logically sound rationale** why the some variable Z is mediating the relationship between X and Y.

Using meta-analytic SEM, we can synthesize evidence from original studies to determine if a proposed mediation is indeed backed by all available evidence. In the following, we will show you an example of how this can be done in *R* using the `metaSEM` package.

13.4.1 Model Specification

For this example, let us assume we want to disentangle why there is a relationship between (low) **psychological resilience** (see Fletcher and Sarkar (2013) for a definition of this concept) and elevated levels of **depressive symptoms**. Based on the literature, we assume that there are two mediating variables: **emotion regulation** capabilities and **dysfunctional coping styles**. Both mediating variables are influenced by resilience, while low emotion regulation capabilities also influence dysfunctional coping. Both emotion regulation and coping style then influence the level of depressive symptoms a person experiences. One may represent the proposed model graphically like this (again using idiosyncratic notation to facilitate the model specification later on):



For this example, we ill use the fictitious `dat.med` dataset, which was adapted from the `Hunter83` dataset in `metaSEM`. The data can be downloaded here. The dataset is again a list containing (1) 14 correlation matrices for resilience, emotion regulation, dysfunctional coping and depressive symptoms extracted from 14 independent studies and (2) the N of each study (see previous chapter for more details on the dataset structure required). Let us have a look at the matrix for the fictitious Devegvar et al. (1992) study:

```
dat.med$data$`Devegvar et al. (1992)`
```

```
##            Resilience EmotReg Coping Depression
## Resilience         NA      NA     NA         NA
## EmotReg            NA    1.00   0.72       0.05
## Coping             NA    0.72   1.00       0.32
## Depression         NA    0.05   0.32       1.00
```

We see that this study has some **missings**, because the variable **Resilience** was not assessed. To see the overall missing data pattern, we can use the `pattern.na()` function.

```
pattern.na(dat.med$data)
```

```
##            Resilience EmotReg Coping Depression
## Resilience          1       3      3          2
## EmotReg             3       2      4          3
## Coping              3       4      2          3
## Depression          2       3      3          1
```

We see that the correlation $r_{EmotReg,Coping}$ has the most missings in our data, with four studies not providing data for it. Given that we have $k = 14$ studies overall, this amount of missing data may be acceptable. However, we have to check if the matrices are **positive definite**, because this is a requirement for the later processing steps. We can do this with the `is.pd()` function.

```
is.pd(dat.med$data)
```

```
##   Guttman et al. (2003) McCaffrey et al. (2002)  Loescher et al. (1997)
##                    TRUE                    TRUE                    TRUE
##  O'Malley et al. (1999)       Hay et al. (1999)   Twiraga et al. (2014)
```

```
##                    TRUE                    TRUE                    TRUE
##    Wanzer et al. (1994)    Arthur et al. (1991)   Frondel et al. (1999)
##                    TRUE                    TRUE                    TRUE
##      Mill et al. (2001)      Ilan et al. (2002) Severence et al. (1996)
##                    TRUE                    TRUE                    TRUE
##  Devegvar et al. (1992)   Matloff et al. (2008)
##                    TRUE                    TRUE
```

We get TRUE for all studies, so everything is fine and we can continue.

13.4.2 Stage 1

Now let us proceed to **pooling the correlation matrices in the first step**. For a more detailed description of this step, please refer to the previous subchapter. This time, we use a fixed-effects model.

```
med1 <- tssem1(dat.med$data,
               dat.med$n,
               method = "FEM")

summary(med1)
```

```
##
## Call:
## tssem1FEM(Cov = Cov, n = n, cor.analysis = cor.analysis, model.name = model.name,
##     cluster = cluster, suppressWarnings = suppressWarnings, silent = silent,
##     run = run)
##
## Coefficients:
##        Estimate Std.Error z value  Pr(>|z|)
## S[1,2] 0.510487  0.012702  40.188 < 2.2e-16 ***
## S[1,3] 0.427086  0.014082  30.329 < 2.2e-16 ***
## S[1,4] 0.207713  0.015931  13.038 < 2.2e-16 ***
## S[2,3] 0.522965  0.013111  39.888 < 2.2e-16 ***
## S[2,4] 0.284562  0.015769  18.046 < 2.2e-16 ***
## S[3,4] 0.243256  0.016266  14.954 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Goodness-of-fit indices:
##                                      Value
## Sample size                       3975.0000
## Chi-square of target model         264.3980
## DF of target model                  60.0000
## p value of target model              0.0000
## Chi-square of independence model 2777.2830
## DF of independence model           66.0000
## RMSEA                               0.1096
```

```
## RMSEA lower 95% CI                  0.0964
## RMSEA upper 95% CI                  0.1234
## SRMR                                0.0918
## TLI                                 0.9171
## CFI                                 0.9246
## AIC                               144.3980
## BIC                              -232.8688
## OpenMx status1: 0 ("0" or "1": The optimization is considered fine.
## Other values may indicate problems.)
```

The optimization status is `0`, so the estimates are trustworthy. If you do not get a status of `0` or 1, plug the model into the `rerun()` function to try fitting it again.

13.4.3 Stage 2

Now that we have the pooled correlation matrix available in `med1`, we can proceed by specifying our proposed mediation model. Again, we specify the $\mathbf{A}$ and $\mathbf{S}$ matrices. The $\mathbf{F}$ is not specified, because all of the variables in are model are **observed** (i.e., there are no latent variables). We will omit the details behind the matrix specification here; for more details please refer to the first subchapter for the general structure of the matrices, and the last subchapter on how the matrices are specified in *R*.

$\mathbf{A}$ Matrix

We use starting values of 0.2.

```
# Build matrix
A <- matrix(c(0             , 0             , 0             , 0,
              "0.2*Res_EmR", 0             , 0             , 0,
              "0.2*Res_Cop", "0.2*EmR_Cop", 0             , 0,
              0             , "0.2*EmR_Dep", "0.2*Cop_Dep", 0),
              ncol = 4, nrow=4, byrow=TRUE)

# Set column and row labels
dimnames(A)[[1]] <- dimnames(A)[[2]] <- c("Resilience", "EmotReg",
                                          "Coping", "Depression")

A
```

```
##            Resilience    EmotReg       Coping        Depression
## Resilience "0"           "0"           "0"           "0"
## EmotReg    "0.2*Res_EmR" "0"           "0"           "0"
## Coping     "0.2*Res_Cop" "0.2*EmR_Cop" "0"           "0"
## Depression "0"           "0.2*EmR_Dep" "0.2*Cop_Dep" "0"
```

```
A <- as.mxMatrix(A)
```

S Matrix

We use starting values of 0.1.

```
# Build matrix
S <- Diag(c(1, "0.1*ErrVarE", "0.1*ErrVarC", "0.1*ErrVarD"))

# Set column and row labels
dimnames(S)[[1]] <- dimnames(S)[[2]] <- c("Resilience", "EmotReg",
                                          "Coping", "Depression")

S
```

```
##            Resilience EmotReg       Coping         Depression
## Resilience "1"        "0"           "0"            "0"
## EmotReg    "0"        "0.1*ErrVarE" "0"            "0"
## Coping     "0"        "0"           "0.1*ErrVarC"  "0"
## Depression "0"        "0"           "0"            "0.1*ErrVarD"
```

```
S <- as.mxMatrix(S)
```

Model Fitting

We can now proceed to fitting the model. In a mediation model, we also want to **estimate the indirect effect** from resilience to depression, taking all mediation paths into account. To do this, we can simply add all the mediation paths together. In our model, this would look like this:

$$\beta_{indirect_{Res-Dep}} = (\beta_{Res-Cop} \times \beta_{Cop-Dep}) + (\beta_{Res-EmR} \times \beta_{EmR-Cop} \times \beta_{Cop-Dep}) + (\beta_{Res-EmR} \times \beta_{Emr-Dep})$$

We can define this function in our model so that it provides us with 95% confidence intervals around the indirect effect if we use likelihood-based intervals. We can define this function as a `list` containing an `mxAlgebra` object in *R*. Here is the code, using the labels we defined in the $\mathbf{A}$ matrix above:

```
list(indirectEffect = mxAlgebra(Res_Cop*Cop_Dep + Res_EmR*EmR_Cop*Cop_Dep + Res_EmR*EmR_Dep,
                                name="indirectEffect"))
```

We can use this code as the argument for the `mx.algebra` parameter in our call to `tssem2()`. Because this is a mediation model, we also have to specify `diag.constraints = TRUE`. Here is the code:

```
med2 <- tssem2(med1,
               Amatrix = A,
               Smatrix = S,
               intervals.type = "LB",
               diag.constraints = TRUE,
```

```
                mx.algebras = list(indirectEffect = mxAlgebra(Res_Cop*Cop_Dep +
                                                               Res_EmR*EmR_Cop*Cop_Dep +
                                                               Res_EmR*EmR_Dep,
                                                              name="indirectEffect")))

# Rerun
med2 <- rerun(med2)

summary(med2)
```

```
Call:
wls(Cov = coef.tssem1FEM(tssem1.obj), aCov = vcov.tssem1FEM(tssem1.obj),
    n = sum(tssem1.obj$n), Amatrix = Amatrix, Smatrix = Smatrix,
    Fmatrix = Fmatrix, diag.constraints = diag.constraints,
    cor.analysis = tssem1.obj$cor.analysis,
    intervals.type = intervals.type, mx.algebras = mx.algebras,
    model.name = model.name, suppressWarnings = suppressWarnings,
    silent = silent, run = run)

95% confidence intervals: Likelihood-based statistic
Coefficients:
        Estimate Std.Error   lbound   ubound z value Pr(>|z|)
EmR_Cop 0.411161        NA 0.377782 0.444676      NA       NA
Res_Cop 0.217913        NA 0.183661 0.252118      NA       NA
Cop_Dep 0.131068        NA 0.091942 0.170226      NA       NA
EmR_Dep 0.218838        NA 0.180424 0.257309      NA       NA
Res_EmR 0.513365        NA 0.488406 0.538309      NA       NA
ErrVarC 0.691468        NA 0.664472 0.717400      NA       NA
ErrVarD 0.904928        NA 0.885399 0.922669      NA       NA
ErrVarE 0.736457        NA 0.710223 0.761460      NA       NA

mxAlgebras objects (and their 95% likelihood-based CIs):
                        lbound  Estimate    ubound
indirectEffect[1,1] 0.1498335 0.1685701 0.1878938

Goodness-of-fit indices:
                                                Value
Sample size                                3975.0000
Chi-square of target model                    9.4087
DF of target model                            1.0000
p value of target model                       0.0022
Number of constraints imposed on "Smatrix"    3.0000
DF manually adjusted                          0.0000
Chi-square of independence model           2697.6496
DF of independence model                      6.0000
RMSEA                                         0.0460
```

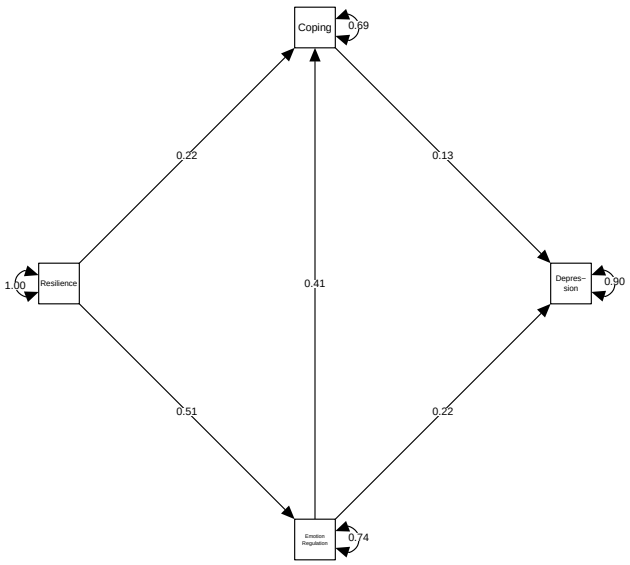
```
RMSEA lower 95% CI                                0.0226
RMSEA upper 95% CI                                0.0748
SRMR                                              0.0161
TLI                                               0.9813
CFI                                               0.9969
AIC                                               7.4087
BIC                                               1.1209
OpenMx status1: 0 ("0" or "1": The optimization is considered fine.
Other values indicate problems.)
```

Because we told the `tssem2()` function to use **likelihood-based** intervals, the Wald-type p-values are not displayed. We see that the proposed model fits the data closely, with $\chi^2_{1,3975} = 9.4, p = 0.002$) and the $RMSEA = 0.046$ being close to 0.05. Please note however, that we used the **fixed-effect model** in stage 1 to pool the correlations, which may not be appropriate if the between-study heterogeneity is substantial. The estimate of the indirect effect from resilience to depression assuming our moderation model is 0.17, which is significant ($95\% CI : 0.15 - 0.19$).

13.4.4 Plotting the Model

Again, we can plot our model using the `semPaths()` function in the `semPlot` package.

```
# Convert to semPlot
sem.plot <- meta2semPlot(med2)

# Create Labels (left to right, bottom to top)
labels <- c("Resilience","Emotion\nRegulation","Coping","Depres-\nsion")

# Plot
semPaths(sem.plot,
         whatLabels = "est",
         edge.color = "black",
         layout="tree2",
         rotation=2,
         nodeLabels = labels)
```

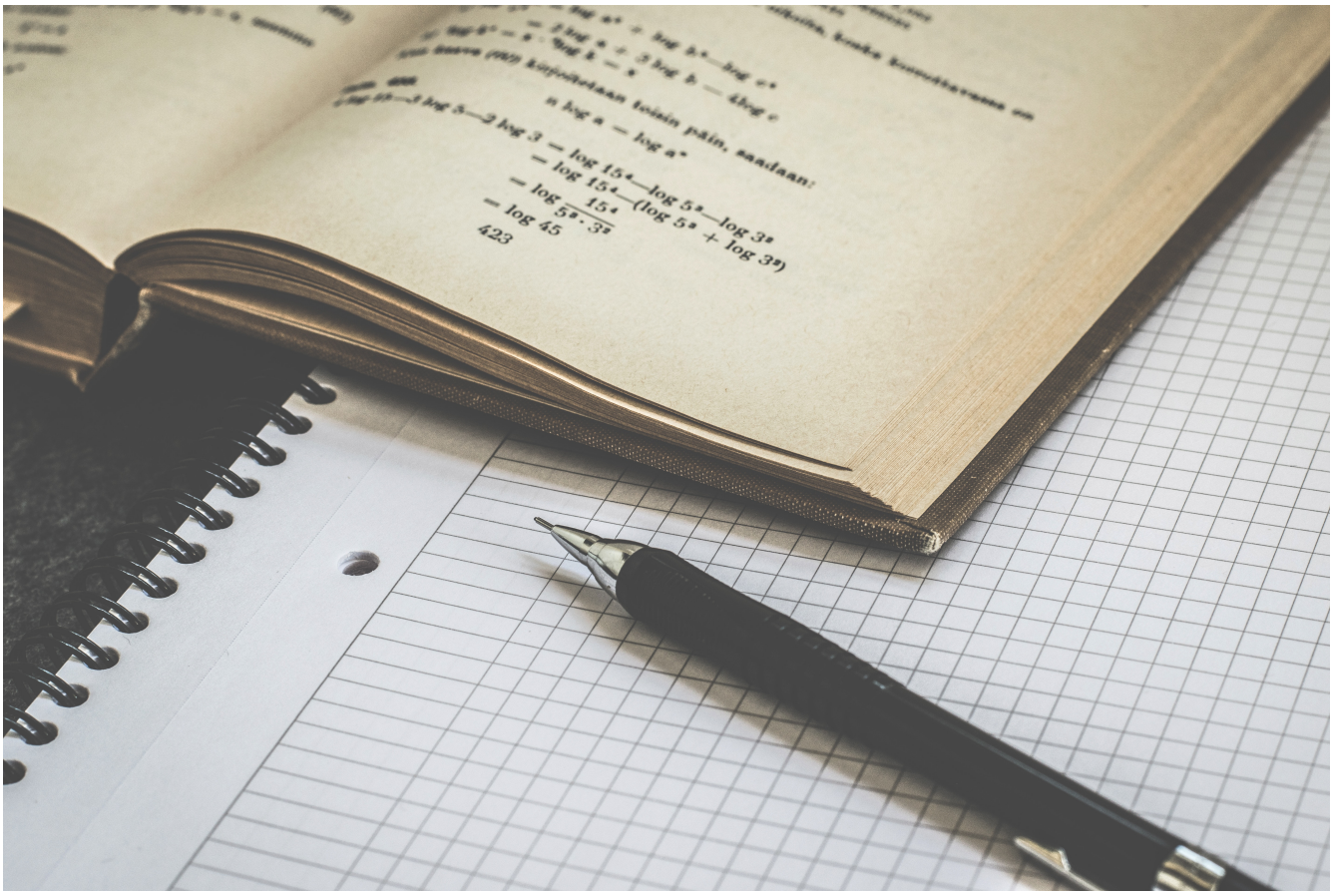

Coping
0.69
0.22
0.13
1.00
Resilience
0.41
Depres–
sion
0.90
0.51
0.22
Emotion
Regulation
0.74

Part IV

Helpful Tools

Chapter 14

Effect size calculators



Although the `meta` package can calculate all **individual effect sizes for every study** if we use the `metabin` or `metacont` function, a frequent scenario is that **some papers do not report the effect size data in the right format**. Especially older articles may often only report results of t**-tests**, **ANOVAs**, or χ^2**-tests**. If enough data is reported, we can also use **such outcome formats to calculate effect sizes**. This way, we can calculate the **effect size (e.g., Hedges' g)** and the **Standard Error (SE)**, which we can then use in a meta-analysis with **pre-calculated effect sizes** using the `metagen` function (see Chapter 4.1.1).

Hedges' g

When dealing with **continuous outcome data**, it is conventional to calculate the **Standardized Mean Difference** (SMD) as an outcome for each study, and as your **summary measure** (Borenstein et al., 2011). A common format to to calculate the SMD in single trials is **Cohen's d** (Cohen, 1988). Yet, this summary measure has been **shown to have a slight bias in small studies, for which it overestimates the effect** (Hedges, 1981). **Hedges *g*** is a similar summary measure, but it **controls for this bias**. It uses a slightly different formula to calculate the pooled variance s_{pooled}, $s*_{pooled}$. The transformation from *d* to *g* is often performed using the formula by Hedges and Olkin (Hedges and Olkin, 1985).

$$g \simeq d \times (1 - \frac{3}{4(n_1 + n_2) - 9})$$

Hedges' g is **commonly used in meta-analysis**, and it's the standard output format in **RevMan**. Therefore, we highly recommend that you also use this measure in you meta-analysis. In `meta`'s `metabin` and `metacont` function, Hedges' g is automatically calculated for each study if we set `sm="SMD"`. If you use the `metgen` function, however, you should calculate Hedges' g for each study yourself first. To calculate the effect sizes, we will use Daniel Lüdecke's extremely helpful `esc` package (Lüdecke, 2018). So, please **install this package first** using the `install.packages("esc")` command, and then load it in you library.

```
library(esc)
```

Here's an overview of all calculators covered in this guide

1. Calculating Hedges' *g* from the Mean and SD
2. Calculating Hedges' *g* from a regression coefficient
3. Calculating an Odd's Ratio from *Chi-square*
4. Calculating Hedges' *g* from a one-way ANOVA
5. Calculating Hedges' *g* from the Mean and SE
6. Calculating Hedges' *g* from a correlation
7. Calculating Hedges' *g* from an independent t-test
8. Calculating Hedges' *g* from Cohen's *d*
9. Calculating effect sizes for studies with multiple comparisons
10. Calculating the *Number Needed to Treat* (NNT) from an effect size
11. Calculating the Standard Error from a *p*-value and effect size

14.1 Hedges' g from the Mean and SD

To calculate Hedges' *g* from the *Mean*, *Standard Deviation*, and n_{group} of both trial arms, we can use the `esc_mean_sd` function with the following parameters.

- `grp1m`: The **mean** of the **first group** (e.g., the intervention).

- `grp1sd`: The **standard deviation** of the **first group**.
- `grp1n`: The **sample size** of the **first group**.
- `grp2m`: The **mean** of the **second group**.
- `grp2sd`: The **standard deviation** of the **second group**.
- `grp2n`: The **sample size** of the **second group**.
- `totalsd`: The **full sample standard deviation**, if the standard deviation for each trial arm is not reported
- `es.type`: the **effect measure** we want to calculate. In our case this is `"g"`. But we could also calculate Cohen's *d* using `"d"`.

Here's an example

```
esc_mean_sd(grp1m = 10.3, grp1sd = 2.5, grp1n = 60,
grp2m = 12.3, grp2sd = 3.1, grp2n = 56, es.type = "g")
```

```
##
## Effect Size Calculation for Meta Analysis
##
##      Conversion: mean and sd to effect size Hedges' g
##     Effect Size:  -0.7082
##  Standard Error:   0.1916
##        Variance:   0.0367
##        Lower CI:  -1.0837
##        Upper CI:  -0.3326
##          Weight:  27.2374
```

14.2 Hedges' *g* from a regression coefficient

14.2.1 Unstandardized regression coefficients

It is also possible to calculate **Hedges' *g*** from an unstandardized or standardized regression coeffiecent (Lipsey and Wilson, 2001).

For **unstardardized coefficients**, we can use the `esc_B` function with the following parameters:

- `b`: unstandardized coefficient b (the "treatment" predictor).
- `sdy`: the standard deviation of the dependent variable y (i.e., the outcome).
- `grp1n`: the number of participants in the first group.
- `grp2n`: the number of participants in the second group.
- `es.type`: the **effect measure** we want to calculate. In our case this is `"g"`. But we could also calculate Cohen's *d* using `"d"`.

Here's an example

```
esc_B(b=3.3,sdy=5,grp1n = 100,grp2n = 150,es.type = "g")
```

```
##
## Effect Size Calculation for Meta Analysis
##
##      Conversion: unstandardized regression coefficient to effect size Hedges' g
##     Effect Size:   0.6941
##  Standard Error:   0.1328
##        Variance:   0.0176
##        Lower CI:   0.4338
##        Upper CI:   0.9544
##          Weight:  56.7018
```

14.2.2 Standardized regression coefficents

Here, we can use the `esc_beta` function with the follwing parameters:

- `beta`: standardized coefficient β (the "treatment" predictor).
- `sdy`: the standard deviation of the dependent variable y (i.e., the outcome).
- `grp1n`: the number of participants in the first group.
- `grp2n`: the number of participants in the second group.
- `es.type`: the **effect measure** we want to calculate. In our case this is `"g"`. But we could also calculate Cohen's *d* using `"d"`.

Here's an example

```
esc_beta(beta=0.7, sdy=3, grp1n=100, grp2n=150, es.type = "g")
```

```
##
## Effect Size Calculation for Meta Analysis
##
##      Conversion: standardized regression coefficient to effect size Hedges' g
##     Effect Size:   1.9868
##  Standard Error:   0.1569
##        Variance:   0.0246
##        Lower CI:   1.6793
##        Upper CI:   2.2942
##          Weight:  40.6353
```

14.3 Odds Ratio from *Chi-square*

To calculate the **Odds Ratio** (or any other kind of effect size measure) from χ^2 using the `esc_chisq` function with the following paramters:

- `chisq`: The value of Chi-squared (or only p)
- `p`: the chi squared p or phi value (or only `chisq`)
- `totaln`: total sample size
- `es.type`: the summary measure (in our case, `"cox.or"`)

Here's an example

```
esc_chisq(chisq=9.9,totaln=100,es.type="cox.or")
```

```
##
## Effect Size Calculation for Meta Analysis
##
##      Conversion: chi-squared-value to effect size Cox odds ratios
##     Effect Size:   2.9858
##  Standard Error:   0.3478
##        Variance:   0.1210
##        Lower CI:   1.5101
##        Upper CI:   5.9036
##          Weight:   8.2667
```

14.4 Hedges' *g* from a one-way ANOVA

We can also derive the SMD from the F-value of a **one-way ANOVA with two groups**. Such ANOVAs can be detected if you look for the **degrees of freedom** (df) underneath of F. In a one-way ANOVA with two groups, the degrees of freedom should always start with 1 (e.g. $F_{1,147} = 5.31$). The formula for this transformation looks like this (Cohen, 1992; Rosnow and Rosenthal, 1996; Rosnow et al., 2000):

$$d = \sqrt{F(\frac{n_t + n_c}{n_t n_c})(\frac{n_t + n_c}{n_t + n_c - 2})}$$

To calculate **Hedges' g** from F-values, we can use the `esc_f` function with the following parameters:

- `f`: *F*-value of the ANOVA
- `grp1n`: Number of participants in group 1
- `grp2n`: Number of participants in group 2
- `totaln`: The total number of participants (if the *n* for each group is not reported)
- `es.type`: the **effect measure** we want to calculate. In our case this is `"g"`. But we could also calculate Cohen's *d* using `"d"`.

Here's an example

```
esc_f(f=5.04,grp1n = 519,grp2n = 528,es.type = "g")
```

```
##
## Effect Size Calculation for Meta Analysis
##
##      Conversion: F-value (one-way-Anova) to effect size Hedges' g
##     Effect Size:   0.1387
##  Standard Error:   0.0619
##        Variance:   0.0038
##        Lower CI:   0.0174
##        Upper CI:   0.2600
##          Weight: 261.1022
```

14.5 Hedges' *g* from the Mean and SE

When calculating **Hedges' g** from the **Mean** and **Standard Error**, we simply make use of the fact that the Standard error is not much more than the **Standard Deviation** when the sample size is taken into account (Thalheimer and Cook, 2002):

$$SD = SE\sqrt{n_c}$$

We can calculate **Hedges' g** using the `esc_mean` function with the following parameters:

- `grp1m`: The mean of the first group.
- `grp1se`: The standard error of the first group.
- `grp1n`: The sample size of the first group.
- `grp2m`: The mean of the second group.
- `grp2se`: The standard error of the second group.
- `grp2n`: The sample size of the second group.
- `es.type`: the **effect measure** we want to calculate. In our case this is `"g"`. But we could also calculate Cohen's *d* using `"d"`.

Here's an example

```
esc_mean_se(grp1m = 8.5, grp1se = 1.5, grp1n = 50,
  grp2m = 11, grp2se = 1.8, grp2n = 60, es.type = "g")
```

```
##
## Effect Size Calculation for Meta Analysis
##
##      Conversion: mean and se to effect size Hedges' g
##     Effect Size:  -0.1998
##  Standard Error:   0.1920
```

```
##         Variance:   0.0369
##         Lower CI:  -0.5760
##         Upper CI:   0.1765
##           Weight:  27.1366
```

14.6 Hedges' *g* from a correlation

For **equally sized groups** ($n_1 = n_2$), we can use the following formula to derive the SMD from the pointbiserial **correlation** (Rosenthal, 1984).

$$r_{pb} = \frac{d}{\sqrt{d^2 + 4}}$$

And this formula for **unequally sized groups** (Aaron et al., 1998):

$$r_{pb} = \frac{d}{\sqrt{d^2 + \frac{(N^2 - 2 \times N)}{n_1 n_2}}}$$

To convert r_{pb} to **Hedges' g**, we can use the `esc_rpb` function with the following parameters:

- `r`: The *r*-value. Either *r* or its *p*-value must be given.
- `p`: The *p*-value of the correlation. Either *r* or its *p*-value must be given.
- `grp1n`: The sample size of group 1.
- `grp2n`: The sample size of group 2.
- `totaln`: Total sample size, if `grp1n` and `grp2n` are not given.
- `es.type`: the **effect measure** we want to calculate. In our case this is `"g"`. But we could also calculate Cohen's *d* using `"d"`.

```
esc_rpb(r = 0.25, grp1n = 99, grp2n = 120, es.type = "g")
```

```
##
## Effect Size Calculation for Meta Analysis
##
##      Conversion: point-biserial r to effect size Hedges' g
##     Effect Size:   0.5170
##  Standard Error:   0.1380
##        Variance:   0.0190
##        Lower CI:   0.2465
##        Upper CI:   0.7875
##          Weight:  52.4967
```

14.7 Hedges' *g* from an independent t-test

The SMD can also be derived from an **independent t-test value** with the following formula (Thalheimer and Cook, 2002):

$$d = \frac{t(n_1 + n_2)}{\sqrt{(n_1 + n_2 - 2)(n_1 n_2)}}$$

We can calculate **Hedges' g** from a **t-test** using the `esc_t` function with the following paramters:

- `t`: The t-value of the t-test. Either *t* or its *p*-value must be given.
- `p`: The *p*-value of the t-test. Either *t* or its *p*-value must be given.
- `grp1n`: The sample size of group 1.
- `grp2n`: The sample size of group 2.
- `totaln`: Total sample size, if `grp1n` and `grp2n` are not given.
- `es.type`: the **effect measure** we want to calculate. In our case this is `"g"`. But we could also calculate Cohen's *d* using `"d"`.

Here's an example

```
esc_t(t = 3.3, grp1n = 100, grp2n = 150,es.type="g")
```

```
##
## Effect Size Calculation for Meta Analysis
##
##      Conversion: t-value to effect size Hedges' g
##     Effect Size:   0.4247
##  Standard Error:   0.1305
##        Variance:   0.0170
##        Lower CI:   0.1690
##        Upper CI:   0.6805
##          Weight:  58.7211
```

14.8 Hedges' *g* from Cohen's *d*

We can also directly correct **Cohen's *d*** and thus generate **Hedges' g** using the formula by Hedges and Olkin (Hedges and Olkin, 1985):

$$g \simeq d \times (1 - \frac{3}{4(n_1 + n_2) - 9})$$

This can be done in R using the `hedges_g` function with the following parameters:

- `d`: The value of **Cohen's d**
- `totaln`: the total *N* in the study

```
hedges_g(d = 0.75, totaln = 50)
```

```
## [1] 0.7382199
```

14.9 Multiple comparisons

Many randomized-controlled trials do not only include a single **intervention** and **control group**, but compare the effect of **two or more interventions** to a control group. It might be tempting in such a scenario to **simply include all the comparisons between the intervention groups and control within a study into one meta-analysis**. Yet, researchers should abstain from this practice, as this would mean that the control group is used twice for the meta-analysis, thus **"double-counting"** the participants in the control group. This results in a **unit-of-analysis** error, as the effect size are correlated, and thus not independent, but are treated as if they would stem from independent samples.

There are two ways to deal with this:

- Splitting the N of the control group: One method to control for the unit-of-analysis error to some extent would be to **split** the number of participants in the control group between the two intervention groups. So, if your control group has $N = 50$ participants, you could divide the control group into two control groups with he same mean and standard deviation, and $N = 25$ participants each. After this preparation step, you could calculate the effect sizes for each intervention arm. As this procedure only partially removes the unit of analysis error, it is not generally recommended. A big plus of this procedure, however, is that it makes investigations of hetereogeneity between study arms possible.
- Another option would be to **synthesize the results of the intervention arms** to obtain one single comparison to the control group. Despite its practical limitations (sometimes, this would mean synthesizing the results from extremely different types of interventions), this procedure does get rid of the unit-of-analysis error problem, and is thus recommended from a statistical standpoint. The following calculations will deal with this option.

To synthesize the **pooled effect size data** (pooled Mean, Standard Deviation and N), we have to use the following

formula:

$$N_{pooled} = N_1 + N_2$$

$$M_{pooled} = \frac{N_1 M_1 + N_2 M_2}{N_1 + N_2}$$

$$SD_{pooled} = \sqrt{\frac{(N_1 - 1)SD_1^2 + (N_2 - 1)SD_2^2 + \frac{N_1 N_2}{N_1 + N_2}(M_1^2 + M_2^2 - 2M_1 M_2)}{N_1 + N_2 - 1}}$$

As these formulae are quite lengthy, we prepared the function `pool.groups` for you, which does the pooling for you automatically. The function is part of the `dmetar` package. If you have the package installed already, you have to load it into your library first.

```
library(dmetar)
```

If you don't want to use the `dmetar` package, you can find the source code for this function here. In this case, *R* doesn't know this function yet, so we have to let *R* learn it by **copying and pasting** the code **in its entirety** into the **console** in the bottom left pane of RStudio, and then hit **Enter** .

To use this function, we have to specifiy the following parameters:

- `n1`: The N in the first group
- `n2`: The N in the second group
- `m1`: The Mean of the first group
- `m2`: The Mean of the second group
- `sd1`: The Standard Deviation of the first group
- `sd2`: The Standard Deviation of the second grop

Here's an example

```
pool.groups(n1=50,
            n2=50,
            m1=3.5,
            m2=4,
            sd1=3,
            sd2=3.8)
```

```
##   Mpooled SDpooled Npooled
## 1    3.75 3.415369     100
```

What should i do when an study has more than two intervention groups

If a study has more than one two intervention groups you want to synthesize (e.g. four arms, with three distinct intervention arms), you can **pool the effect size data for the first two interventions**, and then **synthesize the pooled data you calculated with the data from the third group**. This is fairly straightforward if you save the output from `pool.groups` as an object, and then use the $ operator:

First, pool the **first** and **second intervention group**. I will save the output as `res`.

```
res <- pool.groups(n1 = 50,
                   n2 = 50,
                   m1 = 3.5,
                   m2 = 4,
                   sd1 = 3,
                   sd2 = 3.8)
```

Then, use the pooled data saved in `res` and **pool it with the data from the third group**, using the $ operator to access the different values saved in `res`.

```
pool.groups(n1 = res$Npooled,
            n2 = 60,
            m1 = res$Mpooled,
            m2 = 4.1,
            sd1=res$SDpooled,
            sd2 = 3.8)
```

```
##   Mpooled SDpooled Npooled
## 1 3.88125 3.556696     160
```

14.10 Number Needed to Treat (*NNT*)

Effect sizes such as Cohen's d or Hedges' g are often difficult to interpret from a clinical standpoint. What exactly does an effect of $g = 0.35$ mean for patients, medical professionals, or other stakeholders? A good way to facilitate the understanding of your results is to calculate the **Number Needed to Treat**, or NNT, which signifies how many additional patients must be treated with an intervention or treatment to avoid one additional negative event (e.g., relapse) or tp achieve one additional positive event (e.g., symptom remission, response). There are two ways to calculate the NNT from effect size data such as d or g:

- The method by **Kraemer and Kupfer (2006)**, calculates NNT from the Area Under the Curve (AUC) defined as the probability that a patient in the treatment has an outcome preferable to one in the control. This method allows to calculate the NNT directly from d or g without any extra variables.
- The method by **Furukawa** calculates the NNT from d using a reasonable estimate of CER, in most contexts the assumed response rate in the control group.

Furukawa's method has been shown to be superior in predicting the NNT compared to the Kraemer & Kupfer method (Furukawa and Leucht, 2011). If reasonable assumptions can be made concerning the CER, Furukawa's method should therefore be preferred. When **event data** is used, the CER and EER (experimental group event rate) is calculated first, and the standard definition of the NTT, $\frac{1}{EER-CER}$, can be applied.

14.10.1 The NNT function

To calculate the NNT using these three method, we prepared the NNT function for you. The function is part if the `dmetar` package. If you have the package installed already, you have to load it into your library first.

```
library(dmetar)
```

If you don't want to use the `dmetar` package, you can find the source code for this function here. In this case, *R* doesn't know this function yet, so we have to let *R* learn it by **copying and pasting** the code **in its entirety** into the **console** in the bottom left pane of RStudio, and then hit **Enter**.

Kraemer & Kupfer method

To use the Kraemer & Kupfer method, we only have to provide the function with an effect size (d or g).

```
NNT(d = 0.245)
```

```
## Kraemer & Kupfer's method used.
```

```
## [1] 7.270711
```

Furukawa's method

Once we supply a CER value additionally, Furukawa's method is used automatically.

```
NNT(d = 0.245, CER = 0.35)
```

```
## Furukawa's method used.
```

```
## [1] 10.61533
```

Binary event data

We can also provide binary event data to calculate the NNT directly. To do this, we only need the event rate in the experimental group `event.e` and the total sample in the experimental group, `n.e`, and the same information for the control group in `event.c` and `n.c`.

```
NNT(event.e = 10, event.c = 20, n.e = 200, n.c = 200)
```

```
## [1] 20
```

You can read more about this function in the documentation.

14.11 Standard Error from *p*-value

When extracting effect sizes from published articles, it can sometimes happen that a study only reports the effect size (e.g., Cohen's d), its p-value, but nothing more. To pool results in a meta-analysis (for example using the `metagen` function, see Chapter 4), however, we need some metric for the dispersion of the effect size, preferably the **Standard**

Error (SE). Standard errors can be calculated from Effect sizes such as Cohen's d or the Risk Ratio (RR) using the formula by Altman and Bland (2011). We have prepared a function called `se.from.p` for you which implements this formula. The function is part of the `dmetar` package. If you have the package installed already, you have to load it into your library first.

```
library(dmetar)
```

If you don't want to use the `dmetar` package, you can find the source code for this function here. In this case, *R* doesn't know this function yet, so we have to let *R* learn it by **copying and pasting** the code **in its entirety** into the **console** in the bottom left pane of RStudio, and then hit **Enter** . The function then requires the `esc` package to work.

Effect sizes based on a difference

Assuming we have a study with $N = 75$ participants reporting an effect size of $d = 0.71$ with $p = 0.013$, we can calculate the standard error like this:

```
se.from.p(effect.size = 0.71,
          p = 0.013,
          N = 75,
          effect.size.type= "difference")
```

```
##   EffectSize StandardError StandardDeviation      LLCI     ULCI
## 1       0.71     0.2860464          2.477234 0.1493491 1.270651
```

Effect sizes based on a ratio

Assuming we have a study with $N = 200$ participants reporting an effect size of $OR = 0.91$ with $p = 0.05$, we can calculate the standard error like this:

```
se.from.p(effect.size = 0.91,
          p = 0.05,
          N = 200,
          effect.size.type= "ratio")
```

```
##   logEffectSize logStandardError logStandardDeviation    logLLCI
## 1   -0.09431068       0.04820028            0.6816549 -0.1887832
##        logULCI EffectSize     LLCI     ULCI
## 1 0.0001618765       0.91 0.827966 1.000162
```

As you can see from the output, the function automatically calculates the log-transformed effect size and standard error, which is needed for usage in the `metagen` function (see Chapter 4.3.3).

Chapter 15

Power Analysis

A big asset (and probably one of the reasons why meta-analysis can be helpful in practical research) of meta-analyses is that **they allow for large data to be combined** to attain a more precise pooled effect. **Lack of statistical power**, however, may still play an important role, even in meta-analysis. This is particularly true for the clinical field, where it is often the case that only **few studies are available for synthesis**. The median number of included studies in the *Cochrane Database for Systematic Reviews*, for example, is six (Borenstein et al., 2011). This is even more grave once we consider that (1) many meta-analysts will also want to perform **subgroup analyses and meta-regression**, for which even more power is required, and (2) many meta-analyses have **high heterogeneity**, which reduces our precision, and thus our power.

Power is directly related to the **Type II error level** (β) we defined: $Power = 1 - \beta$. It is common practice to set our **Type I error level** (α) to $\alpha = 0.05$, and thus to assume that the **Type I error is four times as grave as the Type II error** (i.e., falsely finding an effect while there is no effect in reality is four times as bad as not finding an

effect while there is one in reality). The **Type II error** is therefore set at $\beta = 0.20$, and the power should thus be $1 - \beta = 1 - 0.20 = 80\%$.

What assumptions should i make for my meta-analysis?

While researchers conducting primary studies can **plan the size of their sample based on the effect size they want to find**, the situation is a little different in meta-analysis, where we can only work with the published material. However, we have some **control over the number of studies we want to include in our meta-analysis** (e.g., through more leniently or strictly defined inclusion criteria). Therefore, we can change our power to some extent by including more or less studies into the meta-analysis. There are **four things we have to make assumptions about when assessing the power of our meta-analysis a priori**.

- The **number of included or includable studies** k
- The **overall size of the studies we want to include** (are the studies in the field rather small or large?)
- The **effect size we want to determine**. This is particularly important, as we have to make assumptions about how **big an effect size has to be to still be clinically meaningful**. One study calculated that for interventions against depression, even effects as small as $SMD = 0.24$ may still be meaningful for patients (Cuijpers et al., 2014). If we want to study **negative effects of an intervention** (e.g., death or symptom deterioration), even **very small effect sizes are extremely important and should be detected**.
- The **heterogeneity** of our studies' effect sizes, as this also affects the precision of our meta-analysis, and thus its potential to find significant effects.

Besides these parameters, it is also important to think about other analyses, such as the **subgroup analyses** we want to conduct. How many studies are there for each subgroup, and what effects do we want to find in the subgroups? This is particularly important if we **hypothesize that an intervention is not effective in a subgroup of patients**, because we do not want to falsely find a treatment to be ineffective simply because the power was insufficient.

Post-hoc power tests: the abuse of power

Please note that power analyses should always be conducted **a priori**, meaning *before* you perform the meta-analysis. Power analyses conducted *after* an analysis ("post hoc") are fundamentally flawed (Hoenig and Heisey, 2001), as they suffer from the so-called **"power approach paradox"**, in which an analysis yielding no significant effect is thought to show more evidence that the null hypothesis is true when the p-value is smaller, since then, the power to detect a true effect would be higher.

15.1 Fixed-Effect Model

To determine the **power** of a meta-analysis **under the fixed-effect model**, we have to assume the **true value of a distribution when the alternative hypothesis is correct** (i.e., when there is an effect). For power analysis in a conventional study, this distribution is Z. Follwing Borenstein et al. (Borenstein et al., 2011), we will call the true value λ here to make clear that we are dealing with a meta-analysis, and not a primary study. λ is defined as:

$$\lambda = \frac{\delta}{\sqrt{V_\delta}}$$

Where δ is the **true effect size** and V_δ its variance.

V_δ can be calculated for meta-analysis using the fixed-effect model with this formula:

$$V_\delta = \frac{\frac{n_1+n_2}{n_1 x n_2} + \frac{d^2}{2(n_1+n_2)}}{k}$$

Where k are all the included studies, and n_1 and n_2 are the **average sample sizes in each trial arm we assume across our studies**.

Assuming a normal distribution and using λ, we can calculate the Power:

$$Power = 1 - \beta$$
$$Power = 1 - \Phi(c_\alpha - \lambda) + \Phi(-c_\alpha - \lambda)$$

Where c_α is the critical value of a Z-distribution. Φ is the **standard normal density function**, which we we need to calcuate the power using this equation:

$$\Phi(Z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}$$

Luckily, you don't have too think about these statistical details too much, as we have prepared a **function** for you with which you can easily conduct a **power analysis** using the fixed-effect model yourself. The function is called `power.analysis`. The function is part of the `dmetar` package. If you have the package installed already, you have to load it into your library first.

```
library(dmetar)
```

If you don't want to use the `dmetar` package, you can find the source code for this function here. In this case, *R* doesn't know this function yet, so we have to let *R* learn it by **copying and pasting** the code **in its entirety** into the **console** in the bottom left pane of RStudio, and then hit **Enter**. The function then requires the `ggplot2` package to work.

For this function, we have to specify the following parameters:

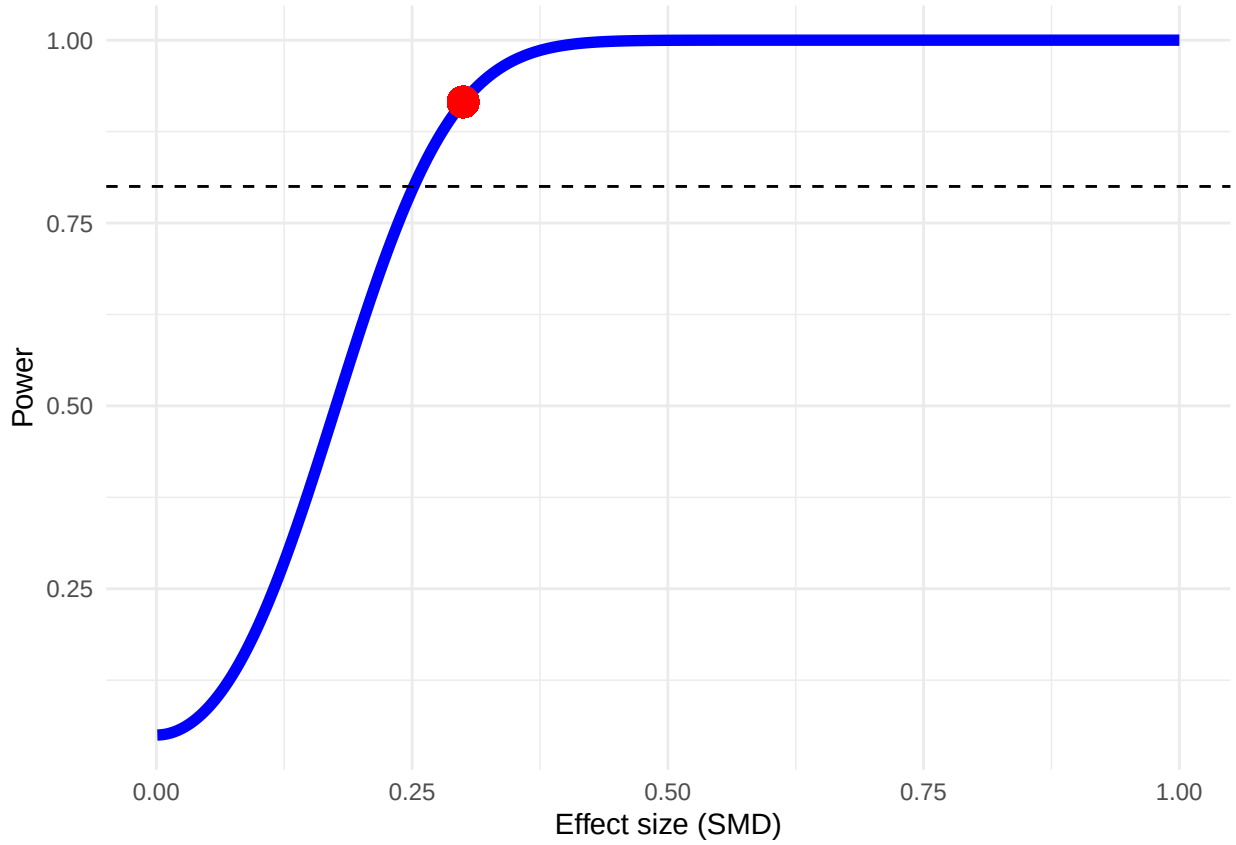
Parameter	Description
d	The hypothesized, or plausible overall effect size of a treatment/intervention under study compared to control, expressed as the standardized mean difference (SMD). Effect sizes must be positive numerics (i.e., expressed as positive effect sizes).
OR	The hypthesized, or plausible overall effect size of a treatment/intervention under study compared to control, expressed as the Odds Ratio (OR). If both d and OR are specified, results will only be computed for the value of d.
k	The expected number of studies to be included in the meta-analysis.
n1	The expected, or plausible mean sample size of the treatment group in the studies to be included in the meta-analysis.
n2	The expected, or plausible mean sample size of the control group in the studies to be included in the meta-analysis.
p	The alpha level to be used for the power computation. Default is alpha=0.05
heterogeneity	Which level of between-study heterogeneity to assume for the meta-analysis. Can be either 'fixed' for no heterogeneity/a fixed-effect model, 'low' for low heterogeneity, 'moderate' for moderate-sized heterogeneity or 'high' for high levels of heterogeneity. Default is 'fixed'.

Now, let's give an example. I assume that an effect of $d = 0.30$ is likely and meaningful for the field of my meta-analysis. I also assume that on average, the studies in my analysis will be rather small, with 25 participants in each trial arm, and that there will be 10 studies in my analysis. I will set the α-level to 0.05, as is convention.

```
power.analysis.(d=0.30,
                k=10,
                n1=25,
                n2=25,
                p=0.05)
```

The output of the function is:

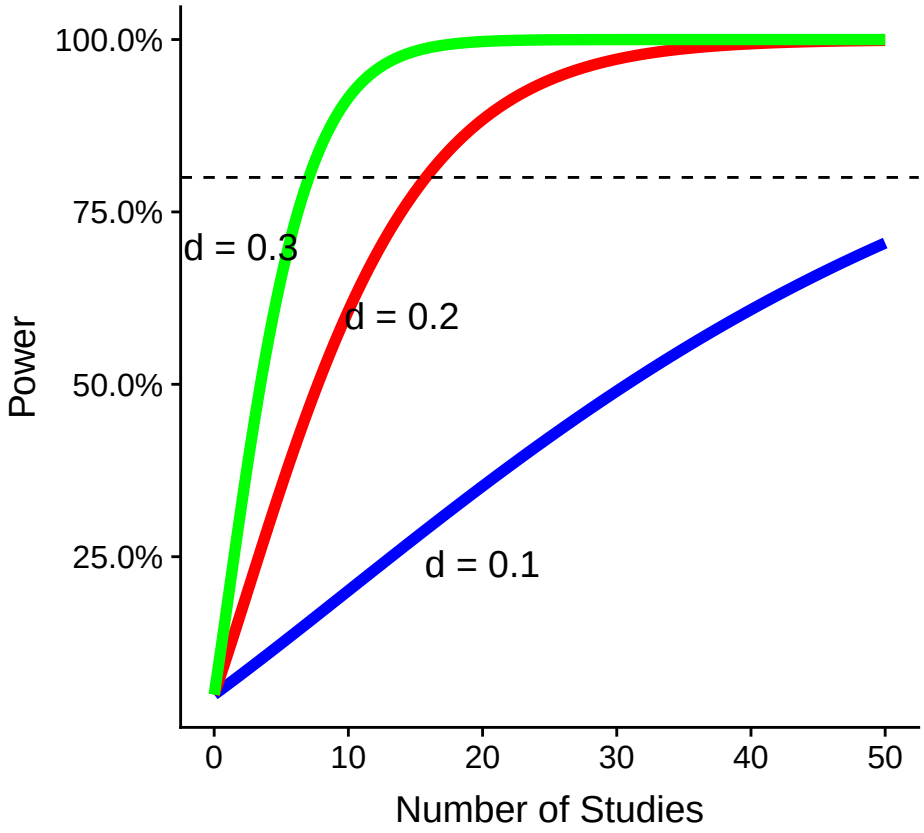
```
## Fixed-effect model used.
```




```
## Power:
```

```
## [1] 0.9155008
```

Meaning that my power is 92%. This is more than the desired 80%, so given that my assumptions are remotely true, my meta-analysis will have **sufficient power using the fixed-effect model to detect a clinically relevant effect if it exists**. So, if i assume an effect of $d = 0.30$ in this example, i am lucky. If we play around with the effect size a little, however, while holding the other parameters constant, this can look very different.



As you can see from this plot, sufficient power (see the **dashed line**) is soon reached for $d = 0.30$, even if only few studies are included. If i assume a smaller effect size of $d = 0.10$, however, **even 50 studies will not be sufficient to find a true effect**.

15.2 Random-Effects Model

For power analyses under the **random-effects model**, the formula to calculate the variance of my true mean effect looks slightly different:

$$V_\delta^* = \frac{V_Y + \tau^2}{k}$$

We see that again, tau^2 has to be included to take the **between-study heterogeneity into account** (see Chapter 4.2 for more details). However, i **do not know the between-study heterogeneity of my analysis** before i perform it, so what value should i assume? According to Hedges and Pigott (Hedges and Pigott, 2004), the follwing formulae may be used to calculate the power in the random-effect model assuming **small, moderate or high heterogeneity**:

Small heterogeneity:

$$V_\delta^* = 1.33 \times \frac{V_Y}{k}$$

Moderate heterogeneity:

$$V_\delta^* = 1.67 \times \frac{V_Y}{k}$$

Large heterogeneity:

$$V_\delta^* = 2 \times \frac{V_Y}{k}$$

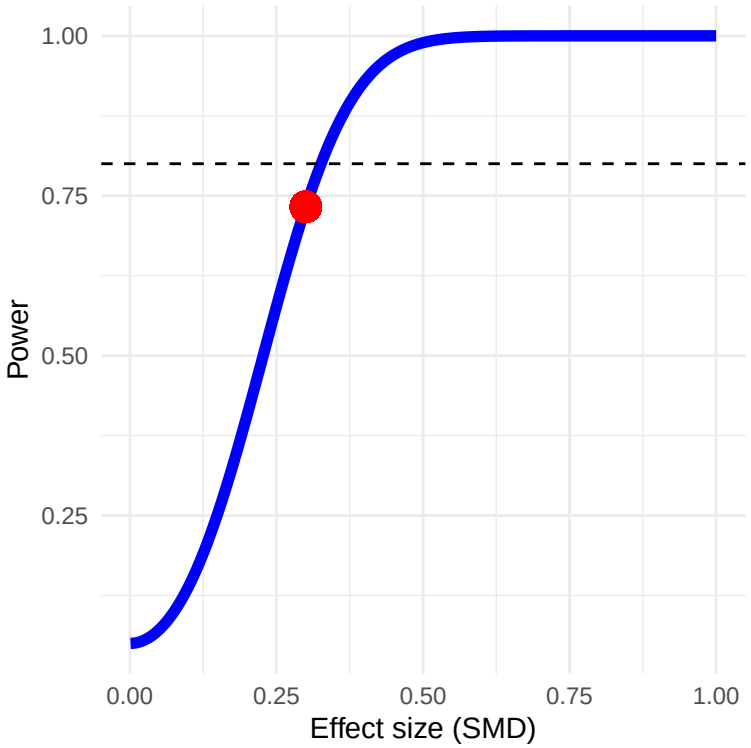
Again, you don't have to worry about the statistical details here. We have put the entire calculations into the power.analysis function, which can already introduced before.

I will assume the same parameters used for the fixed-effect model power analysis, but this time i will also have to specify the heterogeneity in the function, which can take the values "low", "moderate" and "high". I will choose "moderate" for this example.

```
power.analysis(d=0.30,
               k=10,
               n1=25,
               n2=25,
               p=0.05,
               heterogeneity = "moderate")
```

The output i get is:

```
## Random-effects model used (moderate heterogeneity assumed).
```




```
## Power:

## [1] 0.7327163
```

Interestingly, we see that this value is 73%, which is smaller than the value of 91% which was calculated using the **fixed-effect model**. The value is also below 80%, meaning that i would not have optimal power to find the desired effect of $d = 0.30$ to be statistically significant if it exists. This has to do with the **larger heterogeneity** i assume in this simulation, which decreases the precision of my effect size estimate, and thus increases my need for statistical power. **The graph below visualizes this relationship:**

15.3 Subgroup Analyses

Often when conducting a test for subgroup differences within a meta-analysis, we might be interested which effect size difference is needed between the two subgroups for the difference to become significant given the data material at hand. To evaluate this, a **power analysis for subgroup differences** may be helpful.

We have created a function called `power.analysis.subgroup`, which calculates the power of a subgroup difference test (using only two subgroups) by implementing the formulae described by Hedges and Pigott (2001). The function is part of the `dmetar` package. If you have the package installed already, you have to load it into your library first.

```
library(dmetar)
```

If you don't want to use the `dmetar` package, you can find the source code for this function here. In this case, *R* doesn't know this function yet, so we have to let *R* learn it by **copying and pasting** the code **in its entirety** into the **console** in the bottom left pane of RStudio, and then hit **Enter** . The function then requires the `ggplot2` package to work.

Let's assume we have a subgroup analysis in which the pooled effect size of subgroup 1 is $g = 0.30$, with a standard

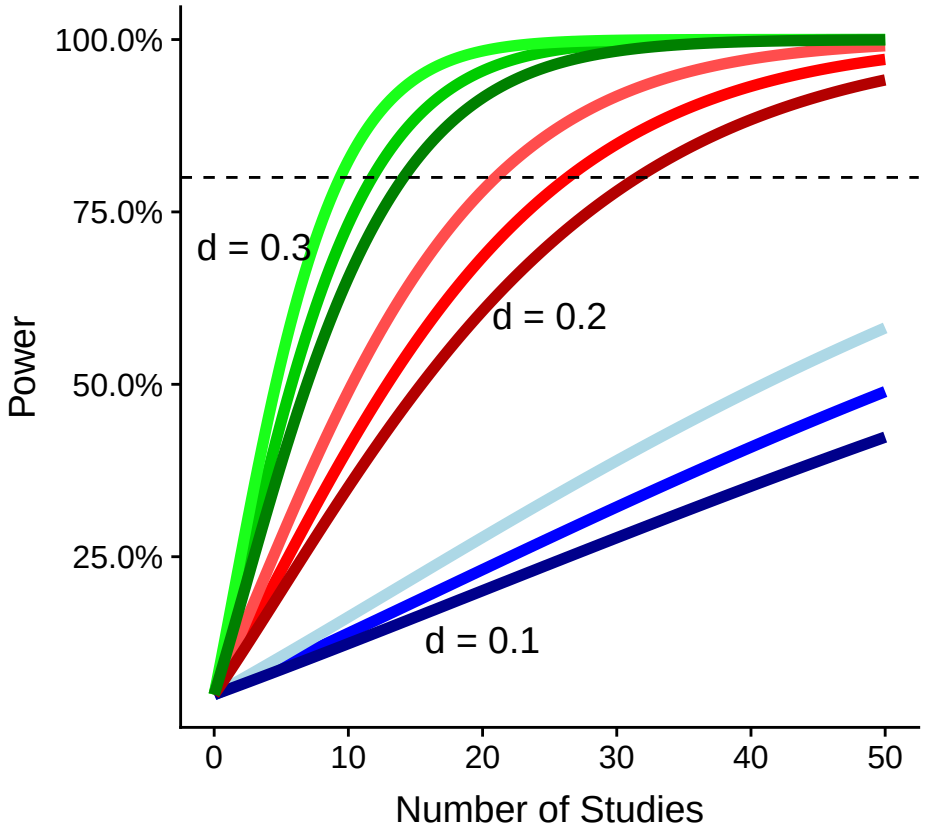
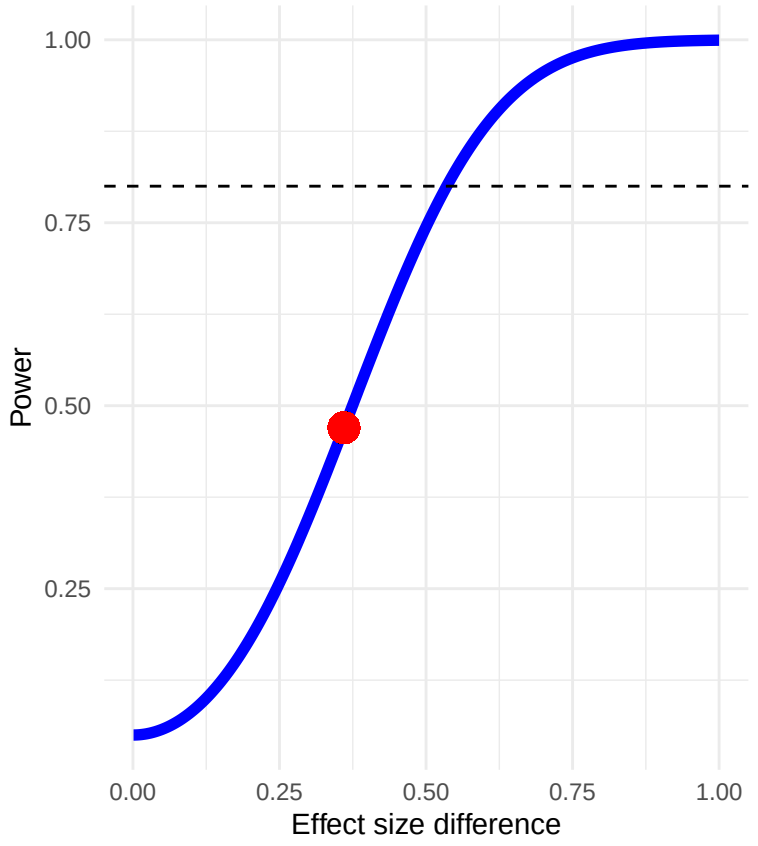


Figure 15.1: Power in the random-effects-model. Darker colors indicate higher heterogeneity

error of 0.13, and an effect size of $g = 0.66$ for subgroup 2, with a standard error of 0.14. We can then use the function like this:

```
power.analysis.subgroup(TE1=0.30,
                        TE2=0.66,
                        seTE1=0.13,
                        seTE2=0.14)
```




```
## Minimum effect size difference needed for sufficient power: 0.536 (input: 0.36)
## Power for subgroup difference test (two-tailed):
```

```
## [1] 0.4698997
```

From the output, we can see that the power for our subgroup test (47%) is **not sufficient**. We also see that the effect size difference must be at least 0.536, leaving all other parameters the same, to reach sufficient power.

15.4 Power Calculator Tool

If you're feeling lazy, or if you want to quickly **check for the power of your meta-analysis under varying assumptions**, you might need a tool which makes it easier for you to calculate the power without having to run the R functions we described before each time.

We therefore built a online **Power Calculator Tool**, which you can find online. The calculations are based on the formulae and functions we described in the previous chapters.

View the tool

Datasets

Here are the **links** to the datasets used in this guide. You can download the data set on the GitHuB page, drag it into your working directory folder, and then **click** on it in the file viewer in RStudio to import them into your environment.

- `bin.metagen`: link
- `binarydata`: link
- `dat.gosh`: link
- `dat.cfa`[1]: link
- `dat.da`[1]: link
- `dat.med`[1]: link
- `IRR.data`: link
- `madata`: link
- `metacont`: link
- `mlm.data`: link
- `mvreg.data`: link
- `rob`: link
- `Senn2013`(reshaped for `gemtc`)[2]: link

[1]Adapted from datasets contained in the `metaSEM` package (Cheung, 2015b).

[2]Adapted from the `Senn2013` dataset contained in the `meta` package (Schwarzer, 2007).

Bibliography

Aaron, B., Kromrey, J. D., and Ferron, J. (1998). *Equating r-based and d-based effect size indices: problems with a commonly recommended formula*. ERIC Clearinghouse.

Altman, D. G. and Bland, J. M. (2011). How to obtain the confidence interval from a p value. *BMJ*, *343*:d2090.

Aronow, P. M. and Miller, B. T. (2019). *Foundations of agnostic statistics*. Cambridge University Press.

Assink, M. and Wibbelink, C. J. (2016). Fitting three-level meta-analytic models in r: A step-by-step tutorial. *The Quantitative Methods for Psychology*, 12(3):154–174.

Baron, R. M. and Kenny, D. A. (1986). The moderator–mediator variable distinction in social psychological research: Conceptual, strategic, and statistical considerations. *Journal of personality and social psychology*, 51(6):1173.

Bates, D., Mächler, M., Bolker, B., and Walker, S. (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1):1–48.

Bauer, D. J. (2003). Estimating multilevel linear models as structural equation models. *Journal of Educational and Behavioral Statistics*, 28(2):135–167.

Baujat, B., Mahé, C., Pignon, J.-P., and Hill, C. (2002). A graphical method for exploring heterogeneity in meta-analyses: application to a meta-analysis of 65 trials. *Statistics in medicine*, 21(18):2641–2652.

Berlin, J. A. and Antman, E. M. (1992). Advantages and limitations of meta-analytic regressions of clinical trials data. *Controlled Clinical Trials*, 13(5):422.

Borenstein, M., Hedges, L. V., Higgins, J. P., and Rothstein, H. R. (2011). *Introduction to meta-analysis*. John Wiley & Sons.

Borenstein, M. and Higgins, J. P. (2013). Meta-analysis and subgroups. *Prevention Science*, 14(2):134–143.

Borenstein, M., Higgins, J. P., Hedges, L. V., and Rothstein, H. R. (2017). Basics of meta-analysis: I2 is not an absolute measure of heterogeneity. *Research synthesis methods*, 8(1):5–18.

Chatfield, C. (1995). Model uncertainty, data mining and statistical inference. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 158(3):419–444.

Cheung, M. W. and Chan, W. (2009). A two-stage approach to synthesizing covariance matrices in meta-analytic structural equation modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 16(1):28–53.

Cheung, M. W.-L. (2008). A model for integrating fixed-, random-, and mixed-effects meta-analyses into structural equation modeling. *Psychological methods*, 13(3):182.

Cheung, M. W.-L. (2014). Modeling dependent effect sizes with three-level meta-analyses: a structural equation modeling approach. *Psychological Methods*, 19(2):211.

Cheung, M. W.-L. (2015a). *Meta-analysis: A structural equation modeling approach*. John Wiley & Sons.

Cheung, M. W.-L. (2015b). metasem: An r package for meta-analysis using structural equation modeling. *Frontiers in Psychology*, 5:1521.

Cheung, M. W.-L. (2015c). metaSEM: An r package for meta-analysis using structural equation modeling. *Frontiers in Psychology*, 5(1521).

Cipriani, A., Furukawa, T. A., Salanti, G., Chaimani, A., Atkinson, L. Z., Ogawa, Y., Leucht, S., Ruhe, H. G., Turner, E. H., Higgins, J. P., et al. (2018). Comparative efficacy and acceptability of 21 antidepressant drugs for the acute treatment of adults with major depressive disorder: a systematic review and network meta-analysis. *Focus*, 16(4):420–429.

Cipriani, A., Higgins, J. P., Geddes, J. R., and Salanti, G. (2013). Conceptual and technical challenges in network meta-analysis. *Annals of internal medicine*, 159(2):130–137.

Cohen, J. (1988). Statistical power analysis for the behavioral sciences. 1988, hillsdale, nj: L. *Lawrence Earlbaum Associates*, 2.

Cohen, J. (1992). A power primer. *Psychological bulletin*, 112(1):155.

Cuijpers, P. (2016). Meta-analyses in mental health research. a practical guide. *Amsterdam, the Netherlands: Pim Cuijpers Uitgeverij*.

Cuijpers, P., Turner, E. H., Koole, S. L., Van Dijke, A., and Smit, F. (2014). What is the threshold for a clinically relevant effect? the case of major depressive disorders. *Depression and anxiety*, 31(5):374–378.

DerSimonian, R. and Laird, N. (1986). Meta-analysis in clinical trials. *Controlled clinical trials*, 7(3):177–188.

Dias, S., Ades, A., Welton, N. J., Jansen, J. P., and Sutton, A. J. (2018). *Network Meta-Analysis for Decision-Making*. Wiley.

Dias, S., Sutton, A. J., Ades, A., and Welton, N. J. (2013). Evidence synthesis for decision making 2: a generalized linear modeling framework for pairwise and network meta-analysis of randomized controlled trials. *Medical Decision Making*, 33(5):607–617.

Dias, S., Welton, N., Caldwell, D., and Ades, A. (2010). Checking consistency in mixed treatment comparison meta-analysis. *Statistics in medicine*, 29(7-8):932–944.

DiCiccio, T. J. and Efron, B. (1996). Bootstrap confidence intervals. *Statistical science*, pages 189–212.

Dickersin, K. (2005). Publication bias: Recognizing the problem, understanding its origins and scope, and preventing harm. *Publication bias in meta-analysis: Prevention, assessment and adjustments*, pages 11–33.

Duval, S. and Tweedie, R. (2000). Trim and fill: a simple funnel-plot–based method of testing and adjusting for publication bias in meta-analysis. *Biometrics*, 56(2):455–463.

Edwards, S., Clarke, M., Wordsworth, S., and Borrill, J. (2009). Indirect comparisons of treatments based on systematic reviews of randomised controlled trials. *International journal of clinical practice*, 63(6):841–854.

Efthimiou, O., Debray, T. P., van Valkenhoef, G., Trelle, S., Panayidou, K., Moons, K. G., Reitsma, J. B., Shang, A., Salanti, G., and Group, G. M. R. (2016). Getreal in network meta-analysis: a review of the methodology. *Research synthesis methods*, *7*(3):236–263.

Egger, M., Smith, G. D., Schneider, M., and Minder, C. (1997). Bias in meta-analysis detected by a simple, graphical test. *Bmj*, *315*(7109):629–634.

Epskamp, S. (2019). *semPlot: Path Diagrams and Visual Analysis of Various SEM Packages' Output*. R package version 1.1.2.

Etz, A. (2018). Introduction to the concept of likelihood and its applications. *Advances in Methods and Practices in Psychological Science*, 1(1):60–69.

Field, A., Miles, J., and Field, Z. (2012). *Discovering statistics using R*. Sage publications.

Fleiss, J. (1993). Review papers: The statistical basis of meta-analysis. *Statistical methods in medical research*, 2(2):121–145.

Fletcher, D. and Sarkar, M. (2013). Psychological resilience: A review and critique of definitions, concepts, and theory. *European psychologist*, 18(1):12.

Follmann, D. A. and Proschan, M. A. (1999). Valid inference in random effects meta-analysis. *Biometrics*, *55*(3):732–737.

Furukawa, T. A. and Leucht, S. (2011). How to obtain nnt from cohen's d: comparison of two methods. *PloS one*, 6(4):e19070.

Furukawa, T. A., McGuire, H., and Barbui, C. (2003). Low dosage tricyclic antidepressants for depression. *Cochrane database of systematic reviews*.

Gart, J. J. and Zweifel, J. R. (1967). On the bias of various estimators of the logit and its variance with application to quantal bioassay. *Biometrika*, pages 181–187.

Gigerenzer, G. (2004). Mindless statistics. *The Journal of Socio-Economics*, *33*(5):587–606.

Gigerenzer, G. (2008). *Rationality for mortals: How people cope with uncertainty*. Oxford University Press.

Good, P. (2013). *Permutation tests: a practical guide to resampling methods for testing hypotheses*. Springer Science & Business Media.

Grolemund, G. (2014). *Hands-On Programming with R: Write Your Own Functions and Simulations*. O'Reilly.

Harrer, M., Adam, S. H., Baumeister, H., Karyotaki, E., Cuijper, P., Bruffaerts, R., Auerbach, R. P., Kessler, R. C., Berking, M., and Ebert, D. D. (2019). Internet interventions for mental health in university students: a systematic review and meta-analysis. *Journal of Methods in Psychiatric Research*.

Hartung, J. (1999). An alternative method for meta-analysis. *Biometrical Journal: Journal of Mathematical Methods in Biosciences*, 41(8):901–916.

Hartung, J. and Knapp, G. (2001a). On tests of the overall treatment effect in meta-analysis with normally distributed responses. *Statistics in medicine*, 20(12):1771–1782.

Hartung, J. and Knapp, G. (2001b). A refined method for the meta-analysis of controlled clinical trials with binary outcome. *Statistics in medicine*, 20(24):3875–3889.

Head, M. L., Holman, L., Lanfear, R., Kahn, A. T., and Jennions, M. D. (2015). The extent and consequences of p-hacking in science. *PLoS biology*, *13*(3):e1002106.

Hedges, L. and Olkin, I. (1985). Statistical models for meta-analysis.

Hedges, L. V. (1981). Distribution theory for glass's estimator of effect size and related estimators. *journal of Educational Statistics*, *6*(2):107–128.

Hedges, L. V. (2009). Statistical considerations. *The handbook of research synthesis and meta-analysis*, pages 38–47.

Hedges, L. V. and Pigott, T. D. (2001). The power of statistical tests in meta-analysis. *Psychological methods*, *6*(3):203.

Hedges, L. V. and Pigott, T. D. (2004). The power of statistical tests for moderators in meta-analysis. *Psychological methods*, *9*(4):426.

Higgins, J., Thompson, S., Deeks, J., and Altman, D. (2002). Statistical heterogeneity in systematic reviews of clinical trials: a critical appraisal of guidelines and practice. *Journal of health services research & policy*, *7*(1):51–61.

Higgins, J. P. and Thompson, S. G. (2002). Quantifying heterogeneity in a meta-analysis. *Statistics in medicine*, *21*(11):1539–1558.

Higgins, J. P. and Thompson, S. G. (2004). Controlling the risk of spurious findings from meta-regression. *Statistics in medicine*, *23*(11):1663–1682.

Higgins, J. P., Thompson, S. G., Deeks, J. J., and Altman, D. G. (2003). Measuring inconsistency in meta-analyses. *BMJ: British Medical Journal*, *327*(7414):557.

Hoenig, J. M. and Heisey, D. M. (2001). The abuse of power: the pervasive fallacy of power calculations for data analysis. *The American Statistician*, *55*(1):19–24.

Iniesta, R., Stahl, D., and McGuffin, P. (2016). Machine learning, statistical learning and the future of biological research in psychiatry. *Psychological medicine*, *46*(12):2455–2465.

IntHout, J., Ioannidis, J. P., and Borm, G. F. (2014). The hartung-knapp-sidik-jonkman method for random effects meta-analysis is straightforward and considerably outperforms the standard dersimonian-laird method. *BMC medical research methodology*, *14*(1):25.

IntHout, J., Ioannidis, J. P., Rovers, M. M., and Goeman, J. J. (2016). Plea for routinely presenting prediction intervals in meta-analysis. *BMJ open*, *6*(7):e010247.

Ioannidis, J. P. (2006). Indirect comparisons: the mesh and mess of clinical trials. *The Lancet*, *368*(9546):1470–1472.

J. Sweeting, M., J. Sutton, A., and C. Lambert, P. (2004). What to add to nothing? use and avoidance of continuity corrections in meta-analysis of sparse data. *Statistics in medicine*, *23*(9):1351–1375.

Jackson, D., Law, M., Rücker, G., and Schwarzer, G. (2017). The hartung-knapp modification for random-effects meta-analysis: A useful refinement but are there any residual concerns? *Statistics in medicine*, *36*(25):3923–3934.

Jackson, D., White, I. R., and Riley, R. D. (2013). A matrix-based method of moments for fitting the multivariate random effects model for meta-analysis and meta-regression. *Biometrical Journal*, *55*(2):231–245.

Jöreskog, K. G. and Sörbom, D. (2006). Lisrel 8.80. *Chicago: Scientific Software International*.

Kline, R. B. (2015). *Principles and practice of structural equation modeling*. Guilford publications.

Koffel, E. and Watson, D. (2009). The two-factor structure of sleep complaints and its relation to depression and anxiety. *Journal of abnormal psychology*, 118(1):183.

König, J., Krahn, U., and Binder, H. (2013). Visualizing the flow of evidence in network meta-analysis and characterizing mixed treatment comparisons. *Statistics in Medicine*, *32*(30):5414–5429.

Krahn, U., Binder, H., and König, J. (2013). A graphical tool for locating inconsistency in network meta-analyses. *BMC medical research methodology*, *13*(1):35.

L'Abbé, K. A., Detsky, A. S., and O'Rourke, K. (1987). Meta-analysis in clinical research. *Annals of Internal Medicine*, 107(2):224–233.

Lipsey, M. W. and Wilson, D. B. (2001). *Practical meta-analysis*. Sage Publications, Inc.

Lu, G. and Ades, A. (2009). Modeling between-trial variance structure in mixed treatment comparisons. *Biostatistics*, 10(4):792–805.

Lüdecke, D. (2018). *Effect Size Computation for Meta Analysis*. R package version 0.4.1.

Makambi, K. H. (2004). The effect of the heterogeneity variance estimator on some tests of treatment efficacy. *Journal of biopharmaceutical statistics*, 14(2):439–449.

Mansfield, E. R. and Helms, B. P. (1982). Detecting multicollinearity. *The American Statistician*, *36*(3a):158–160.

Mantel, N. and Haenszel, W. (1959). Statistical aspects of the analysis of data from retrospective studies of disease. *Journal of the national cancer institute*, *22*(4):719–748.

Marsman, M., Schönbrodt, F. D., Morey, R. D., Yao, Y., Gelman, A., and Wagenmakers, E.-J. (2017). A bayesian bird's eye view of 'replications of important results in social psychology'. *Royal Society Open Science*, *4*(1):160426.

Mbuagbaw, L., Rochwerg, B., Jaeschke, R., Heels-Andsell, D., Alhazzani, W., Thabane, L., and Guyatt, G. H. (2017). Approaches to interpreting and choosing the best treatments in network meta-analyses. *Systematic reviews*, *6*(1):79.

McArdle, J. J. and McDonald, R. P. (1984). Some algebraic properties of the reticular action model for moment structures. *British Journal of Mathematical and Statistical Psychology*, *37*(2):234–251.

McGrayne, S. B. (2011). *The theory that would not die: how Bayes' rule cracked the enigma code, hunted down Russian submarines, & emerged triumphant from two centuries of controversy*. Yale University Press.

Mehta, P. D. and Neale, M. C. (2005). People are variables too: Multilevel structural equations modeling. *Psychological methods*, 10(3):259.

Montgomery, D. C. (2001). Design and analysis of experiments, 2001. *New York: John Wiley & Sons*, pages *394–395*.

Muthén, L. K. and Muthén, B. O. (2012). Mplus: statistical analysis with latent variables–user's guide.

Nelson, L. D., Simmons, J., and Simonsohn, U. (2018). Psychology's renaissance. *Annual review of psychology*, *69*.

Olkin, I., Dahabreh, I. J., and Trikalinos, T. A. (2012). Gosh–a graphical display of study heterogeneity. *Research synthesis methods*, *3*(3):214–223.

Pastor, D. A. and Lazowski, R. A. (2018). On the multilevel nature of meta-analysis: a tutorial, comparison of software programs, and discussion of analytic choices. *Multivariate behavioral research*, 53(1):74–89.

Peters, J. L., Sutton, A. J., Jones, D. R., Abrams, K. R., and Rushton, L. (2008). Contour-enhanced meta-analysis funnel plots help distinguish publication bias from other causes of asymmetry. *Journal of clinical epidemiology*, 61(10):991–996.

Poole, C. and Greenland, S. (1999). Random-effects meta-analyses are not always conservative. *American Journal of Epidemiology*, 150(5):469–475.

Robins, J., Greenland, S., and Breslow, N. E. (1986). A general estimator for the variance of the mantel-haenszel odds ratio. In *American journal of epidemiology*. Citeseer.

Rosenthal, R. (1984). Meta-analysis procedure for social research.

Rosnow, R. L. and Rosenthal, R. (1996). Computing contrasts, effect sizes, and counternulls on other people's published data: General procedures for research consumers. *Psychological Methods*, 1(4):331.

Rosnow, R. L., Rosenthal, R., and Rubin, D. B. (2000). Contrasts and correlations in effect-size estimation. *Psychological science*, 11(6):446–453.

Rothstein, H. R., Sutton, A. J., and Borenstein, M. (2006). *Publication bias in meta-analysis: Prevention, assessment and adjustments*. John Wiley & Sons.

Rücker, G. (2012). Network meta-analysis, electrical networks and graph theory. *Research synthesis methods*, 3(4):312–324.

Rücker, G. and Schwarzer, G. (2015). Ranking treatments in frequentist network meta-analysis works without resampling methods. *BMC medical research methodology*, 15(1):58.

Rücker, G., Schwarzer, G., Carpenter, J. R., and Schumacher, M. (2008). Undue reliance on i 2 in assessing heterogeneity may mislead. *BMC medical research methodology*, 8(1):79.

Rücker, G., Schwarzer, G., Krahn, U., König, J., and Schwarzer, M. G. (2015). Package 'netmeta'. *Network Meta-Analysis using Frequentist Methods (Version 0.7-0)*.

Salanti, G., Ades, A., and Ioannidis, J. P. (2011). Graphical methods and numerical summaries for presenting results from multiple-treatment meta-analysis: an overview and tutorial. *Journal of clinical epidemiology*, 64(2):163–171.

Salanti, G., Del Giovane, C., Chaimani, A., Caldwell, D. M., and Higgins, J. P. (2014). Evaluating the quality of evidence from a network meta-analysis. *PloS one*, 9(7):e99682.

Schwarzer, G. (2007). meta: An r package for meta-analysis. *R news*, 7(3):40–45.

Schwarzer, G., Carpenter, J. R., and Rücker, G. (2015). *Meta-analysis with R*. Springer.

Senn, S., Gavini, F., Magrez, D., and Scheen, A. (2013). Issues in performing a network meta-analysis. *Statistical Methods in Medical Research*, 22(2):169–189.

Shim, S. R., Kim, S.-J., Lee, J., and Rücker, G. (2019). Network meta-analysis: application and practice using r software. *Epidemiology and health*, 41:e2019013.

Sidik, K. and Jonkman, J. N. (2007). A comparison of heterogeneity variance estimators in combining results of studies. *Statistics in medicine*, 26(9):1964–1981.

Simmons, J. P., Nelson, L. D., and Simonsohn, U. (2011). False-positive psychology: Undisclosed flexibility in data collection and analysis allows presenting anything as significant. *Psychological science*, 22(11):1359–1366.

Simonsohn, U., Nelson, L. D., and Simmons, J. P. (2014a). P-curve: a key to the file-drawer. *Journal of Experimental Psychology: General*, 143(2):534.

Simonsohn, U., Nelson, L. D., and Simmons, J. P. (2014b). p-curve and effect size: Correcting for publication bias using only significant results. *Perspectives on Psychological Science*, 9(6):666–681.

Simonsohn, U., Simmons, J. P., and Nelson, L. D. (2015). Better p-curves: Making p-curve analysis more robust to errors, fraud, and ambitious p-hacking, a reply to ulrich and miller (2015).

Song, F., Loke, Y. K., Walsh, T., Glenny, A.-M., Eastwood, A. J., and Altman, D. G. (2009). Methodological problems in the use of indirect comparisons for evaluating healthcare interventions: survey of published systematic reviews. *Bmj*, 338:b1147.

Steiger, J. H. and Lind, J. C. (1980). Statistically based tests for the number of common factors, paper presented at the annual meeting of the psychometric society. *Iowa City, IA*.

Tang, R. W. and Cheung, M. W.-L. (2016). Testing ib theories with meta-analytic structural equation modeling: The tssem approach and the univariate-r approach. *Review of International Business and Strategy*, 26(4):472–492.

Thalheimer, W. and Cook, S. (2002). How to calculate effect sizes from published research: A simplified methodology. *Work-Learning Research*, 1.

Thompson, B. (2004). *Exploratory and confirmatory factor analysis: Understanding concepts and applications*. American Psychological Association.

van Aert, R. C., Wicherts, J. M., and van Assen, M. A. (2016). Conducting meta-analyses based on p values: Reservations and recommendations for applying p-uniform and p-curve. *Perspectives on Psychological Science*, 11(5):713–729.

van Valkenhoef, G., Lu, G., de Brock, B., Hillege, H., Ades, A., and Welton, N. J. (2012). Automating network meta-analysis. *Research synthesis methods*, 3(4):285–299.

Veroniki, A. A., Jackson, D., Viechtbauer, W., Bender, R., Bowden, J., Knapp, G., Kuss, O., Higgins, J. P., Langan, D., and Salanti, G. (2016). Methods to estimate the between-study variance and its uncertainty in meta-analysis. *Research synthesis methods*, 7(1):55–79.

Viechtbauer, W. (2005). Bias and efficiency of meta-analytic variance estimators in the random-effects model. *Journal of Educational and Behavioral Statistics*, 30(3):261–293.

Viechtbauer, W. and Cheung, M. W.-L. (2010). Outlier and influence diagnostics for meta-analysis. *Research synthesis methods*, 1(2):112–125.

Viechtbauer, W. et al. (2010). Conducting meta-analyses in r with the metafor package. *J Stat Softw*, 36(3):1–48.

Viechtbauer, W., López-López, J. A., Sánchez-Meca, J., and Marín-Martínez, F. (2015). A comparison of procedures to test for moderators in mixed-effects meta-regression models. *Psychological methods*, 20(3):360.

Whittingham, M. J., Stephens, P. A., Bradbury, R. B., and Freckleton, R. P. (2006). Why do we still use stepwise modelling in ecology and behaviour? *Journal of animal ecology*, *75*(5):1182–1189.

Wiksten, A., Rücker, G., and Schwarzer, G. (2016). Hartung–knapp method is not always conservative compared with fixed-effect meta-analysis. *Statistics in medicine*, *35*(15):2503–2515.