Meta-analysis cheat sheet

Joe Hilgard

Meta-analysis is the process of taking an effect size and its standard error from every study in the literature.

Rules of meta-analysis:

1. All effect sizes must be relevant and competent tests of the hypothesis.
2. All effect sizes must involve a single degree of freedom in the numerator (e.g., a two-sample t-test or the interaction term from a 2×2 ANOVA).
   - If you have more than one degree of freedom in the numerator, make the appropriate contrast(s) that have one degree of freedom.
3. Each row is one effect size.
   - Multiple contrasts within one study have the same study name, same outcome name, different contrast name.
   - Multiple outcomes within one study have the same study name, same contrast name, different outcome name.

Your R tools:

The functions you will use come in packages known as "libraries". Libraries do not come standard with R – they are developed and shared by other R users. The libraries you need are metafor, compute.es, and my homemade package "hilgard" which has a function for pooling standard deviations.

You install the official libraries like this:

```r
# Commenting this out b/c of etiquette, uncomment if you intend to run it
# install.packages(c('metafor', 'compute.es'))
```

The hilgard package is unofficial and has to be installed via devtools::install_github().

```r
# install.packages('devtools')
# library(devtools)
# install_github("Joe-Hilgard/hilgard")
```

You load all the packages like this:

```r
library(metafor)
library(compute.es)
library(hilgard)
```

Two-sample t-tests

In a two sample t-test, the means of two groups are compared. This is the simplest case you will deal with.

> Participants in the treatment group (n = 51, M = 7.4, SD = 2.3) gave higher ratings than participants in the control group (n = 48, M = 5.7, SD = 1.7), t(97) = 3.96, p < .001.

Compute.es has the functions fes() and tes(), useful for converting a test statistic into an effect size. Using this, we can get the effect size directly from the t-statistic:

```r
tes(3.96, 51, 48)
```

```
## Mean Differences ES:
##
##  d [ 95 %CI] = 0.8 [ 0.38 , 1.21 ]
##   var(d) = 0.04
##   p-value(d) = 0
##   U3(d) = 78.71 %
##   CLES(d) = 71.33 %
##   Cliff's Delta = 0.43
##
##  g [ 95 %CI] = 0.79 [ 0.38 , 1.2 ]
##   var(g) = 0.04
##   p-value(g) = 0
##   U3(g) = 78.53 %
##   CLES(g) = 71.18 %
##
##  Correlation ES:
##
##  r [ 95 %CI] = 0.37 [ 0.19 , 0.53 ]
##   var(r) = 0.01
##   p-value(r) = 0
##
##  z [ 95 %CI] = 0.39 [ 0.19 , 0.59 ]
##   var(z) = 0.01
##   p-value(z) = 0
##
##  Odds Ratio ES:
##
##  OR [ 95 %CI] = 4.24 [ 2 , 8.99 ]
##   p-value(OR) = 0
##
##  Log OR [ 95 %CI] = 1.44 [ 0.69 , 2.2 ]
##   var(lOR) = 0.14
##   p-value(Log OR) = 0
##
##  Other:
##
##  NNT = 3.55
##  Total N = 99
```

You can reduce the amount of output by adding the argument "verbose = FALSE" and then retrieving d and var.d manually.

```r
es1 <- tes(3.96, 51, 48, verbose = F)
# using the $ operator
es1$d; es1$var.d
```

```
## [1] 0.8
```

```
## [1] 0.04
```

```r
# using indexing
es1[, c("d", "var.d")]
```

```
##      d var.d
## 1 0.8  0.04
```

Note also that it can be a good idea to ask for more digits to be printed so as to avoid rounding error, especially when the sample size is large.

```r
es2 <- tes(16.7, 5000, 5000, verbose = F)
es2$d; es2$var.d
```

```
## [1] 0.33
```

```
## [1] 0
```

```r
# var.d = 0 will screw everything up because that implies a sample size of Infinity
es2 <- tes(16.7, 5000, 5000, verbose = F, dig = 8)
es2$d; es2$var.d
```

```
## [1] 0.334
```

```
## [1] 0.00040558
```

```r
# that's more like it. .0004 is small, but it's not zero.
```

2x2 ANOVA

ANOVA is just a series of simultaneous t-statistics. Imagine we are testing two factors. One has levels A and B, the other has levels 1 and 2. Fully crossed, they look like this:

.	A	B
1	A1	B1
2	A2	B2

Three effects are usually reported: A main effect of A vs B, a main effect of 1 vs 2, and an interaction of the two factors.

Each of these is just a contrast between the averages of two groups.

The main effect of A vs B:

.	A	B
1	+1	-1
2	+1	-1

The main effect of 1 vs 2:

.	A	B
1	+1	+1
2	-1	-1

The interaction of the two factors:

.	A	B
1	+1	-1

.	A	B
2	-1	+1

Usually the text will report the F- or t-test associated with the effect you're interested in. But sometimes it won't. See below under "Effect size from summary statistics" for how to compute an effect size from scratch.

Main effects

An F-statistic with one denominator degree of freedom is just a squared t-statistic. You can use fes() on F-statistics.

> There was a significant main effect of treatment, F(1, 38) = 19.14, p < .001

```
# Via f-test
f1 <- fes(19.14, 20, 20, verbose = F)
f1$d; f1$var.d
```

```
## [1] 1.38
```

```
## [1] 0.12
```

```
# Via t-test
f2 <- tes(sqrt(19.14), 20, 20, verbose = F)
f2$d; f2$var.d
```

```
## [1] 1.38
```

```
## [1] 0.12
```

```
# same result
```

Interaction test

Remember that the interaction test is just the contrast

.	A	B
1	+1	-1
2	-1	+1

So, the interaction test is comparing (A1 + B2) against (A2 + B1).

Usually you can just use tes or fes.

> As expected, the MS X Target interaction was significant, F(1, 29) = 4.25, p = .048

```
t3 <- fes(4.25, 31/2, 31/2, verbose = F)
t3$d; t3$var.d
```

```
## [1] 0.74
```

```
## [1] 0.14
```

Sometimes authors do not report the proper test of the interaction, instead reporting the two "simple slopes". In these cases you will need to calculate the effect size from the summary statistics.

Effect size from summary statistics

Sometimes it is necessary to calculate the effect from the summary statistics.

> Thinking of death lead to eating more Big Macs (n = 30, M = 3.2, SD = 2) than thinking of pain (n = 31, M = 0.8, SD = 1.2).

We can use the escalc function for this. Just put in the ns, means, and SDs, like so:

```r
escalc("SMD",
       n1i = 31, m1i = 3.2, sd1i = 2,
       n2i = 31, m2i = 0.9, sd2i = 1.2)
```

```
##       yi     vi
## 1 1.3771 0.0798
```

Sometimes, to make an interaction term, you will need to combine groups:

> Among participants primed by mortality salience, more stones were thrown at a heretic (n = 20, M = 12.4, SD = 4.1) than at a saint (n = 25, M = 6.8, SD = 3.7). Among participants not primed, there was no difference between stones thrown at heretics (n = 23, M = 10.7, SD = 3.9) and stones thrown at saints (n = 21, M = 9.1, SD = 3.3).

We need to combine the MS-heretic and no-prime-saint condition and compare that against the combination of the MS-saint and no-prime-heretic condition.

Combine the means using weighted.mean(), weighting the means by their sample sizes:

```r
m1 <- weighted.mean(x = c(12.4, 9.1),
              w = c(20, 21))
m2 <- weighted.mean(x = c(6.8, 10.7),
                    w = c(25, 23))
```

Combine the SDs using pool.sd(), weighting by the sample sizes

```r
sd1 <- pool.sd(sds = c(4.1, 3.3),
               ns = c(20, 21))
sd2 <- pool.sd(sds = c(3.7, 3.9),
               ns = c(25, 23))
```

Now you can use those combined ns, ms, and sds to get the effect size:

```r
escalc("SMD",
       m1i = m1, sd1i = sd1, n1i = 20+21,
       m2i = m2, sd2i = sd2, n2i = 25+23)
```

```
##       yi     vi
## 1 0.5384 0.0469
```

Note that you don't have to run escalc yourself. It is enough to put the means, sds, and ns into the Excel spreadsheet. From there I can run escalc on the whole spreadsheet at once.

```r
spreadsheet <- data.frame("m1" = c(10, 25, 50),
                          "m2" = c(10, 20, 40),
                          "sd1" = c(1, 1, 5),
                          "sd2" = c(1, 3, 5),
                          "n1" = c(30, 30, 300),
                          "n2" = c(40, 20, 300))
spreadsheet
```

```
##   m1 m2 sd1 sd2  n1  n2
## 1 10 10   1   1  30  40
## 2 25 20   1   3  30  20
## 3 50 40   5   5 300 300
```

```r
with(spreadsheet,
     escalc("SMD", m1i = m1, m2i = m2, sd1i = sd1, sd2i = sd2, n1i = n1, n2i = n2))
```

```
##       yi     vi
## 1 0.0000 0.0583
## 2 2.4110 0.1415
## 3 1.9975 0.0100
```

Checking effect sizes from summary statistics

Once you have d and var.d, you can make your own t-statistic like this:

$t = \frac{d}{\sqrt{(var.d)}}$

You can then check if that t-value gives you a p-value in the right ballpark with pt():

```r
t <- abs(.538/sqrt(.047))
2*pt(t, df = 80, lower.tail = F)
```

```
## [1] 0.01517431
```

Complicated designs

Sometimes you will deal with a more complicated design with more than two groups. You might have several treatment conditions or multiple control conditions. The authors may not report the particular contrast you are interested in.

Determine which cells belong in the contrast, and which do not. Determine which cells need to be averaged together, and which need to be excluded. Then use the techniques shown above to make those averages and create the effect size from those averages.

Example

Let's work through an example together.

Renkema, Stapel, & van Yperen, 2008. https://pure.uvt.nl/ws/files/1025738/Gowiththe.pdf (We realize that many of Stapel's papers have been retracted. This one has not, so we treat it as real data in our meta-analysis and in this example.)

In Study 3, the authors report a 3 (Mortality salience: Death essay, TV essay, dental pain essay) $\times$ 2 (Rating: high-low or low-high) ANOVA.

The authors report an F-test with two degrees of freedom for the interaction. > As expected, we did find an interaction effect of mortality salience and likeability on the rating of the drawings F(2, 84) = 3.21, p < .05.

Remember that we cannot use an F-test with more than 1 degree of freedom in the numerator. We only want the 2 (Mortality: Death essay, pain essay) $\times$ 2 (Rating: high-low or low-high) interaction, ignoring the TV essay condition.

Looking at their Table 1, we get the following means and SDs.

.	Death	TV	Pain
Liked	7.2 (0.5)	5.6 (0.4)	5.7 (0.5)
Disliked	5.2 (0.5)	5.6 (0.4)	5.8 (0.5)

They mention a total N of 90, so we will assume $90/(3 \times 2) = 15$ subjects per cell.

To get the Mortality Salience × Pain interaction, we need to combine the cells appropriately by hand using weighted.mean(), pool.sd(), then finally, escalc().

First, we focus on the cells we need:

.	Death	Pain
Liked	7.2 (0.5)	5.7 (0.5)
Disliked	5.2 (0.5)	5.8 (0.5)

We remember that we are interested in the interaction term, which means we must combine the diagonal cells:

.	Death	Pain
Liked	+1	-1
Disliked	+1	-1

We do that using weighted.mean for the means and pool.sd for the sds:

```r
# Means combined by weighted.mean
m1 <- weighted.mean(c(7.2, 5.8), c(15, 15))
m2 <- weighted.mean(c(5.2, 5.7), c(15, 15))
# SDs combined by pool.sd
sd1 <- pool.sd(c(0.5, 0.5), c(15, 15))
sd2 <- pool.sd(c(0.5, 0.5), c(15, 15))
```

With the combined means, sds, and ns, we are ready to plug everything into escalc:

```r
escalc("SMD",
       n1i = 15+15, m1i = m1, sd1i = sd1,
       n2i = 15+15, m2i = m2, sd2i = sd2)
```

```
##       yi     vi
## 1 2.0727 0.1025
```

Here we see that this yields d = 2.10 with var.d = .10, equal to sd.d = 0.32. Let's check the p-value.

```r
t <- 2.10/sqrt(.10)
2*pt(t, 58, lower.tail = F)
```

```
## [1] 1.172872e-08
```

$p < .0000001$! Surely the effect is not this significant.

Perhaps the authors have confused standard deviation for standard error. Let's talk about how to fix that.

Standard deviation and standard error

Authors sometimes confuse standard deviation (SD) with standard error (SE). Standard error is always much smaller than the SD, so confusing the two can lead to misreporting the effect size as much, much bigger than it should be.

The relationship between the two is $SE = \frac{SD}{\sqrt{n}}$. SD does not shrink with sample size; SE does. Think about it like this: If you measure the heights of ten people or ten thousand people, that does not change the fact that people have different heights. But if you measure the heights of ten people, your sampling error is big, and if you measure the heights of ten thousand people, your sampling error is small.

If we have reason to believe that the SE was reported as the SD, we can use algebra to solve for the SD. This gives us $SD = SE \times \sqrt{n}$.

Returning to our example from Renkema et al. (2008), let's use the SEs and ns to get the actual SD.

```
# Remember, SD = SE * sqrt(n)
# Study reported cell sizes of 15, cell SEs of 0.5
0.5*sqrt(15)
```

```
## [1] 1.936492
```

```
# Let's use this 1.94 in pool.sd
sd1 <- pool.sd(c(1.94, 1.94), c(15, 15))
sd2 <- pool.sd(c(1.94, 1.94), c(15, 15))
# Now try escalc again with this new value
escalc("SMD",
       n1i = 15+15, m1i = m1, sd1i = sd1,
       n2i = 15+15, m2i = m2, sd2i = sd2)
```

```
##       yi     vi
## 1 0.5342 0.0690
```

This looks like a much closer match to the *p*-value they reported.

Extracting means and SDs from a figure

Sometimes the manuscript doesn't give you the numbers you need, but there is a figure. There exist a number of software tools for extracting means and SDs/SEs from figures. Try metaDigitize or webPlotDigitizer.

Backsolving for SDs or SEs from test statistics

Sometimes authors will report just a table of means with one ANOVA test besides the one we're interested in (e.g., the interaction term, when we're interested in a main or simple effect).

I think there's a way to backsolve for SDs / SEs in this circumstance, but I can't remember it.

Chi-square tests

Sometimes an article's outcome is dichotomous. In this case, authors will probably report a chi-square test instead of the usual F-test or t-test. We can still turn these into our usual statistics, however, using compute.es::chies(). Just remember that the chi-square should have only one degree of freedom.

```r
chi1 <- chies(chi.sq = 10.76,
      n = 72,
      verbose = F)
chi1$d; chi1$var.d
```

```
## [1] 0.83
```

```
## [1] 0.07
```

You can also make the effect size from the cell counts with metafor::escalc. Just put in the cell frequencies for the 2x2 table as requested. See ?escalc, subheading "Measures for Dichotomous Variables."

```r
escalc("SMD",
       ai = 20, bi = 17,
       ci = 7, di = 28)
```

```
## Error in escalc.default("SMD", ai = 20, bi = 17, ci = 7, di = 28): Cannot compute outcomes. Check tha
##   information is specified via the appropriate arguments.
```

R doesn't seem to like it when we ask it to calculate the SMD directly. We have to ask it to transform the odds ratio into a *d* through either the probit, normal, or logistic distribution.

```r
escalc("PBIT",
       ai = 20, bi = 17,
       ci = 7, di = 28)
```

```
##       yi     vi
## 1 0.9434 0.1009
```

```r
escalc("OR2DN",
       ai = 20, bi = 17,
       ci = 7, di = 28)
```

```
##       yi     vi
## 1 0.9387 0.1056
```

```r
escalc("OR2DL",
       ai = 20, bi = 17,
       ci = 7, di = 28)
```

```
##       yi     vi
## 1 0.8539 0.0874
```

Thankfully these don't vary *too* much, but I would like to know more about the assumptions involved. Let's pick one and stick to it for now. How about… "OR2DN".