# Not-so-standardized effect-sizes

Why use standardized effect sizes?

Ian Hussey

06 Mai, 2024

## Contents

**Why standardize?** 1

**Violating assumptions of standardized effect sizes** 2

**Cohen's d** 5

**Standardized effect sizes require estimating multiple parameters** 6

**Session info** 9

```r
# dependencies ----
library(tidyr)
library(dplyr)
library(forcats)
library(readr)
library(purrr)
library(ggplot2)
library(effsize)
library(sn)
library(knitr)
library(kableExtra)

# set the seed ----
# for the pseudo random number generator to make results reproducible
set.seed(123)
```

## Why standardize?

They have different possible ranges, different population means ($\mu$), and different population SDs ($\sigma$).

Even if had perfect that a given therapy has a (population) efficacy of lowering BDI-II depression scores by 6 points, without knowing a lot about the relationships between the BDI-II and other scores, we know little about how many points the same therapy would affect depression scores on the MADRS or the HAM-D.

(surprisingly, very little work is ever done to collect information on the relationship between different scores so that we could know this)

Imagine three different published RCTs, each of which studied the efficacy of the same form of cognitive behavioral therapy for depression:

- RCT 1 found that it lowered depression scores on the BDI-II by 6 points on average

- RCT 2 found that it lowered depression scores on the MADRS by 8 points on average
- RCT 3 found that it lowered depression scores on the HAM-D by 4 points on average

What is the efficacy of the intervention for depression scores on the PHQ-9? This is impossible to answer without knowing a lot about the details of the different scales (e.g., their min/max scores), the distribution of each scale's scores in the population (eg population $\mu$ and $\sigma$), and the relationship between different depression scales in the population. A one-point-change on one scale likely has a very different meaning to a one-point-change on another scale.

What is the efficacy of the intervention for depression *in general*? This too is impossible to answer as there is no common scale between them.

'Standardized' effect sizes are useful here as they provide common units. Instead of points on the self-report scale (i.e., sum scores), which differ between scales, standardized effect sizes generally use Standard Deviations as their units. For example, Cohen's d = 0.2 means that there are 0.2 Standard Deviations of difference between the two groups.

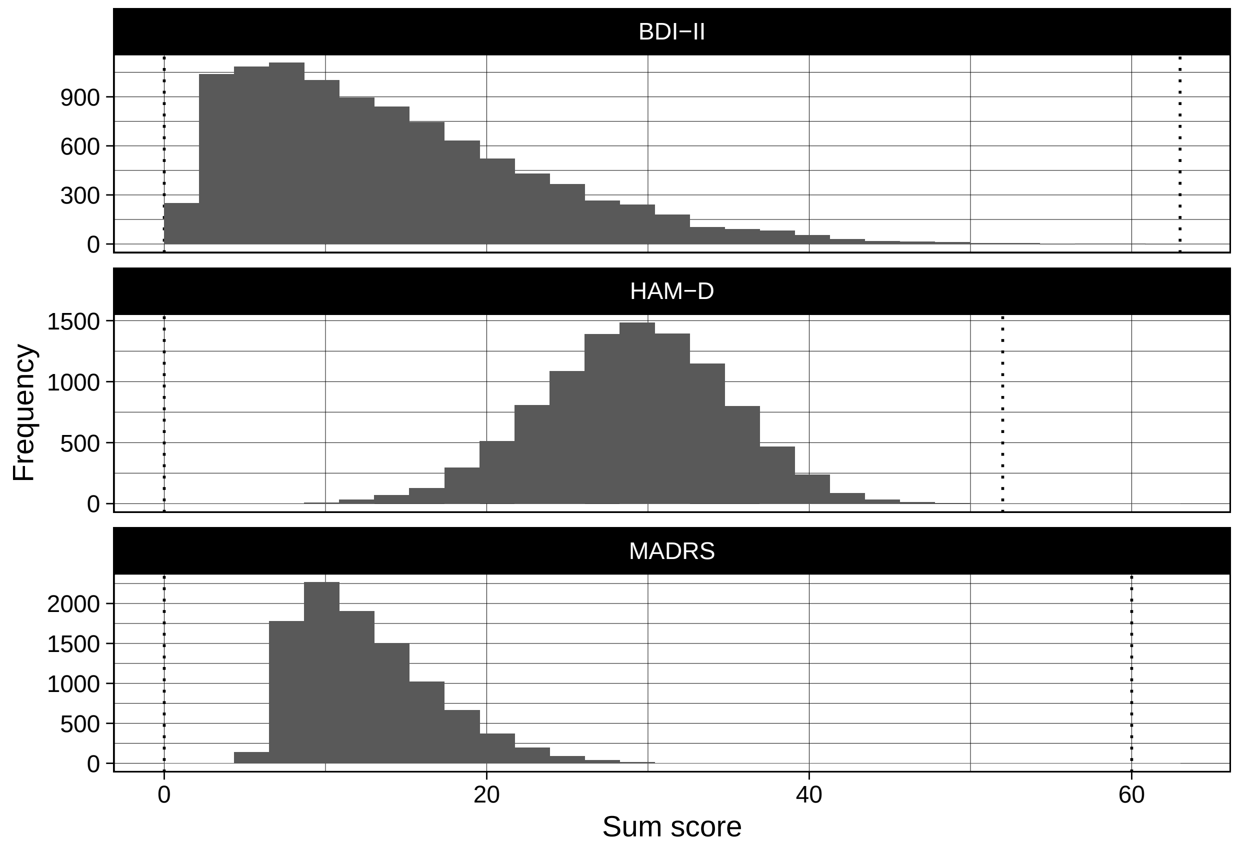
In principle, standardized effect sizes are extremely useful as they allow us to draw comparisons between studies using very different outcome measures, or indeed to synthesise results between such studies (i.e., meta-analysis).

## Violating assumptions of standardized effect sizes

In reality, the distribution of scores on different scales differ in lots of ways, including which probability distribution they follow and their parameter values. For example, even if all depression scales' scores followed skew-normal distributions, they could differ in their means (location), SDs (scale), and skew - and also their bounding (e.g., min/max possible scores). For example, imagine these scores on the BDI-II, MADRS, and HAM-D (note that only min/max scores are realistic, other parameters are chosen merely for illustration):

```r
N <- 10000

generated_data <-
  bind_rows(
    tibble(measure = "BDI-II",
           score = rsn(n = N,
                        xi = 2,  # location
                        omega = 15, # scale
                        alpha = 16),
           max_score = 63), # skew
    tibble(measure = "HAM-D",
           score = rsn(n = N,
                        xi = 33,  # location
                        omega = 7, # scale
                        alpha = -1),
           max_score = 52), # skew
    tibble(measure = "MADRS",
           score = rsn(n = N,
                        xi = 7,  # location
                        omega = 7, # scale
                        alpha = 9),
           max_score = 60) # skew
  ) |>
  mutate(score = case_when(score < 0 ~ 0,
                           score > max_score ~ max_score,
                           TRUE ~ score))
```

```r
ggplot(generated_data, aes(score)) +
  geom_vline(aes(xintercept = 0), linetype = "dotted") +
  geom_vline(aes(xintercept = max_score), linetype = "dotted") +
  geom_histogram(boundary = 0) +
  facet_wrap(~ measure, ncol = 1, scales = "free_y") +
  theme_linedraw() +
  ylab("Frequency") +
  xlab("Sum score")
```

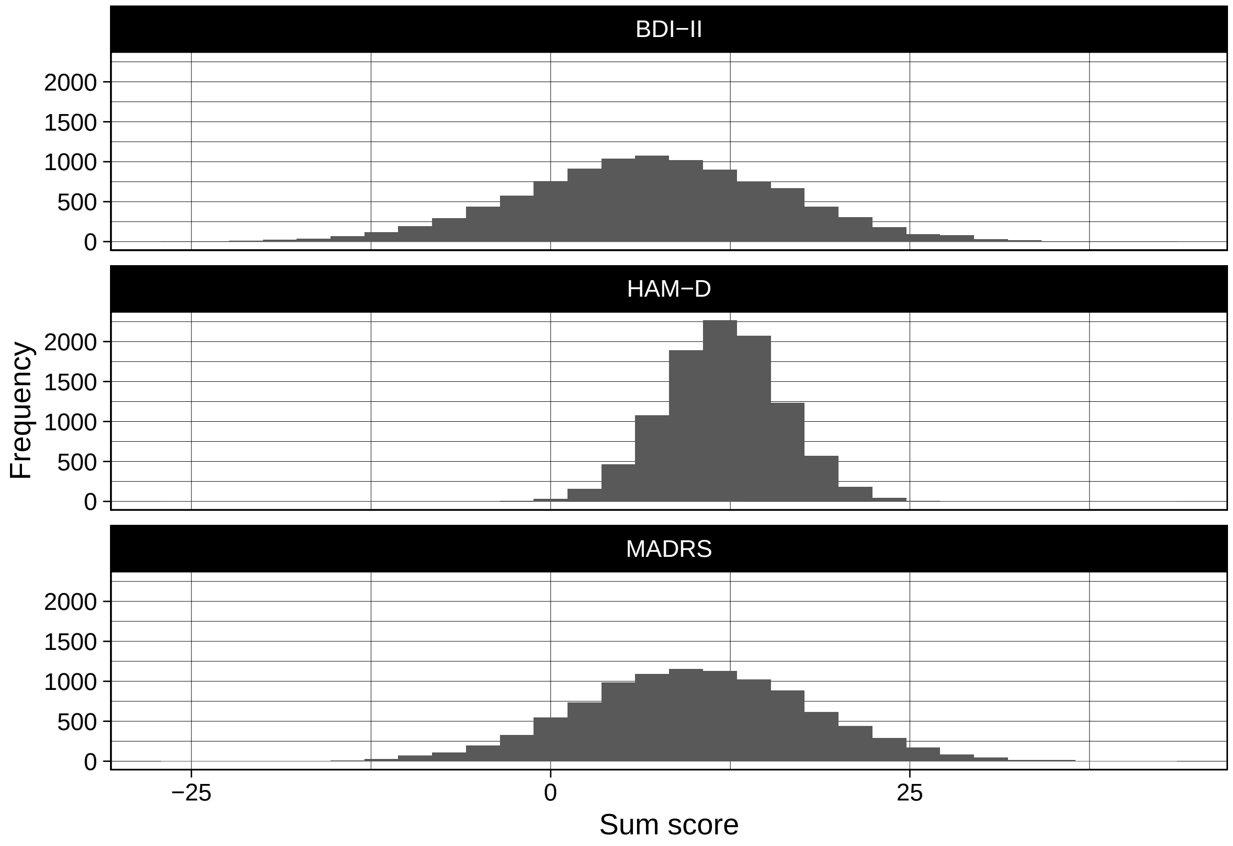


Unfortunately, we already run into a problem with standardized effect sizes like Cohen's d (and indeed Pearson's r correlations): they assume continuous normal data. If we were strictly adhering to only using statistical methods when their assumptions are met, then we shouldn't use standardized effect sizes for most psychological research. As usual, we often ignore these violations of assumptions. As we have seen from previous lessons, it can matter less when each data set violates the assumptions in similar ways (e.g., skew).

Nonetheless, remember that violation of assumptions is the first thing that can cause problems for the use of standardized effect sizes.

For the moment, let's pretend like these scales produce continuous normal data that only differ in their population location ($\mu$) and scale ($\sigma$):

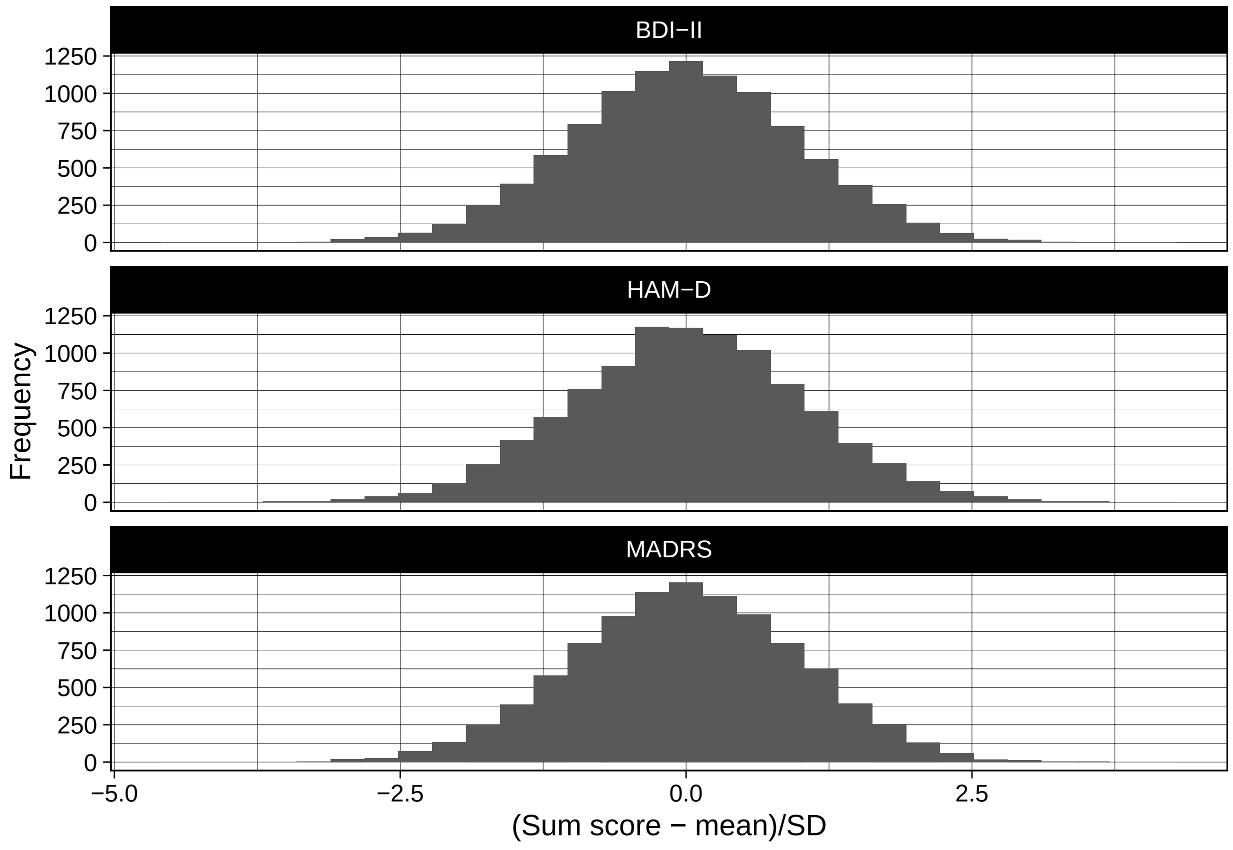
```r
generated_data <-
  bind_rows(
    tibble(measure = "BDI-II",
           score = rnorm(n = N, mean = 7, sd = 9),
           max_score = 63),
    tibble(measure = "HAM-D",
            score = rnorm(n = N, mean = 12, sd = 4),
            max_score = 52),
    tibble(measure = "MADRS",
            score = rnorm(n = N, mean = 10, sd = 8),
            max_score = 60)
  )

ggplot(generated_data, aes(score)) +
  geom_histogram() +
  facet_wrap(~ measure, ncol = 1) +
  theme_linedraw() +
  ylab("Frequency") +
  xlab("Sum score")
```



A one-point change on the BDI-II still means something very different to a one-point change on the MADRS or HAM-D.

Data for a single sample can be standardized by taking each participant's score, deducting the mean score (the sample estimate of $\mu$), and then dividing by the SD of scores (the sample estimate of $\sigma$). Now, all scales have a mean of 0 and an SD of 1. A one-point change on any scale has the same interpretation: a one-standard deviation change on that scale's scores:

```
generated_data <-
  bind_rows(
    tibble(measure = "BDI-II",
            score = rnorm(n = N, mean = 0, sd = 1),
            max_score = 63),
    tibble(measure = "HAM-D",
           score = rnorm(n = N, mean = 0, sd = 1),
           max_score = 52),
    tibble(measure = "MADRS",
           score = rnorm(n = N, mean = 0, sd = 1),
           max_score = 60)
  )

ggplot(generated_data, aes(score)) +
  geom_histogram() +
  facet_wrap(~ measure, ncol = 1) +
  theme_linedraw() +
  ylab("Frequency") +
  xlab("(Sum score - mean)/SD")
```



## Cohen's d

Cohen's d is an extension of this logic to two samples. In its original form, Cohen specified it as: $d = \frac{M_{group1} - M_{group2}}{SD_{pooled}}$

How $SD_{pooled}$ is determined is a little more complex, but think of it as the pooling of two SD values ($SD_{group1}$ and $SD_{group2}$). It's not quite their average, but conceptually its close enough.

There's an excellent interactive visualization of Cohen's d here which I encourage you to play around with.

In another RMarkdown document, we'll come back to different versions and implementations of Cohen's d.

## Standardized effect sizes require estimating multiple parameters

Cohen's d (usually) involves having to create a sample estimate of the means in each group. Researchers are usually more interested in differences between means.

But it also involves having to estimate the SDs. This can be a little a little confusing the first time you encounter it: we often intuitively think of SD as the amount of noise around the signal we're interested in (the mean). We are somewhat more used to thinking about the fact that estimated means have error round them: the standard error of the mean (SEM) is used to calculate confidence intervals around means, and the SEM is actually just the SD of the mean (as opposed to normal SD, which is SD of the data).

We are relatively less familiar with thinking about the fact that estimates of standard deviation also are estimated with error, e.g., the standard error of the SD, which is the SD of the SD. Confused yet?

We can understand this more easily with a simulation. We generate data for a single sample with a population mean ($\mu$) = 0 and population SD ($\sigma$) = 1.

Across lots of iterations, we can see that the average sample mean is close to the population mean ($\mu$), and the average sample SD is close to the population ($\sigma$):

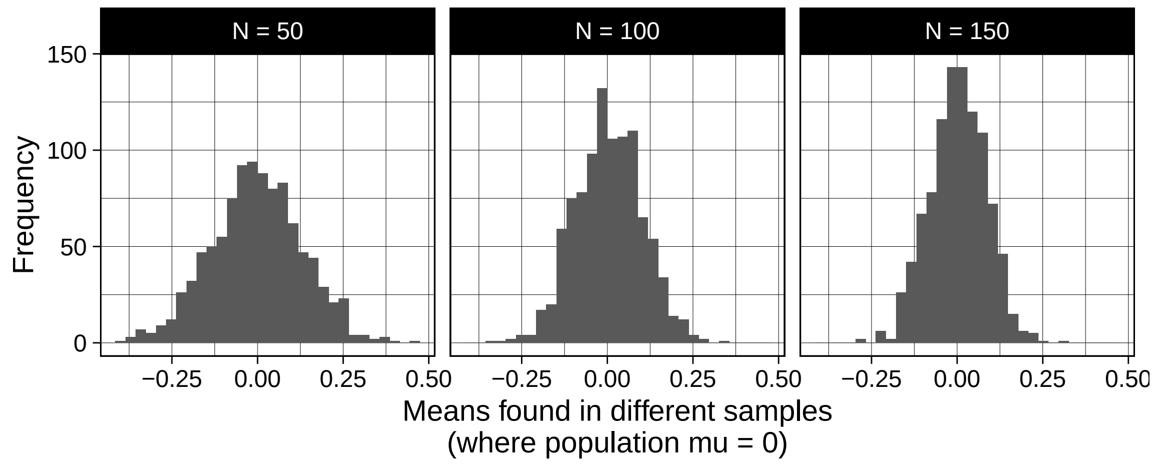
```r
# set the seed ----
# for the pseudo random number generator to make results reproducible
set.seed(123)


# define data generating function ----
generate_data <- function(n,
                          mean,
                          sd) {

  data <- tibble(score = rnorm(n = n, mean = mean, sd = sd))

  return(data)
}


# define data analysis function ----
analyse_data <- function(data) {

  res <- data |>
    summarize(sample_mean = mean(score),
              sample_sd = sd(score))

  return(res)
}


# define experiment parameters ----
experiment_parameters_grid <- expand_grid(
  n = c(50, 100, 150),
  mean = 0,
  sd = 1,
  iteration = 1:1000
)


# run simulation ----
```

```
simulation <-
  # using the experiment parameters
  experiment_parameters_grid |>

  # generate data using the data generating function and the parameters relevant to data generation
  mutate(generated_data = pmap(list(n,
                                    mean,
                                    sd),
                               generate_data)) |>

  # apply the analysis function to the generated data using the parameters relevant to analysis
  mutate(analysis_results = pmap(list(generated_data),
                                 analyse_data))


# summarise simulation results over the iterations ----
simulation_summary <- simulation |>
  unnest(analysis_results)

simulation_summary |>
  group_by(n) |>
  summarize(average_sample_means = mean(sample_mean),
            average_sample_sds = mean(sample_sd)) |>
  mutate_if(is.numeric, janitor::round_half_up, digits = 2) |>
  kable() |>
  kable_classic(full_width = FALSE)
```

| n | average_sample_means | average_sample_sds |
|---:|---:|---:|
| 50 | 0 | 1 |
| 100 | 0 | 1 |
| 150 | 0 | 1 |

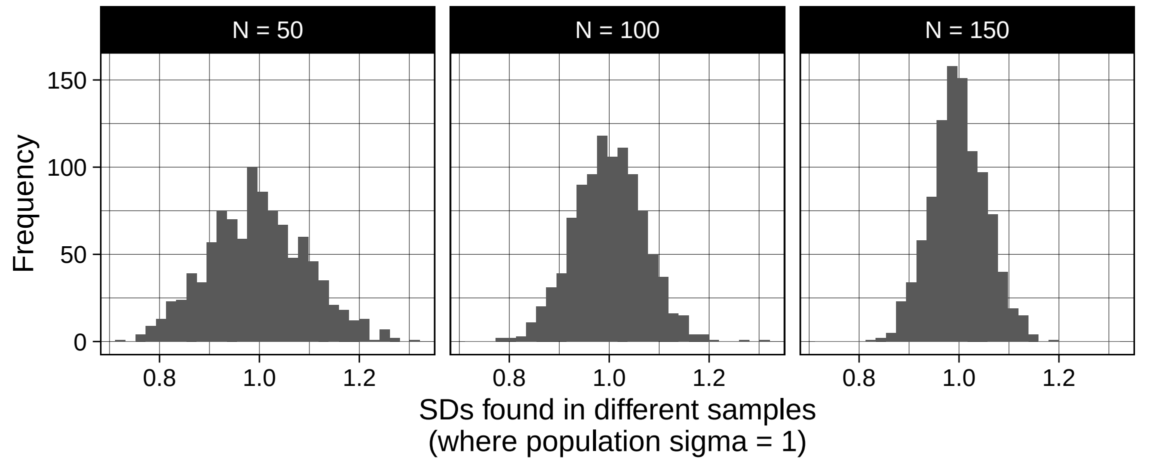
But the estimated means in individual samples (i.e., individual iterations) vary around this true value ($\mu = 0$). The smaller the sample size, the more deviation there is from the population value:

```
simulation_summary |>
  mutate(n_string = paste("N =", n),
         n_string = fct_relevel(n_string, "N = 50", "N = 100", "N = 150")) |>
  ggplot(aes(sample_mean)) +
  geom_histogram(boundary = 0) +
  theme_linedraw() +
  ylab("Frequency") +
  xlab("Means found in different samples\n(where population mu = 0)") +
  facet_wrap(~ n_string)
```



The same applies to the estimated SDs in individual samples (i.e., individual iterations), which also vary around this true value ($\sigma = 1$). The smaller the sample size, the more deviation there is from the population value:

```r
simulation_summary |>
  mutate(n_string = paste("N =", n),
         n_string = fct_relevel(n_string, "N = 50", "N = 100", "N = 150")) |>
  ggplot(aes(sample_sd)) +
  geom_histogram(boundary = 0) +
  theme_linedraw() +
  ylab("Frequency") +
  xlab("SDs found in different samples\n(where population sigma = 1)") +
  facet_wrap(~ n_string)
```



We can summarize this across the iterations too. The previous table reported the average sample mean (mean of the means) and average sample SD (mean of the SD). We can also add the SD of the sample means, and the SD of the sample SDs. Notice that these SDs - these deviations from the population values - get smaller as the sample sizes get larger.

```r
simulation_summary |>
  group_by(n) |>
  summarize(average_sample_means = mean(sample_mean),
            average_sample_sds = mean(sample_sd),
            sd_of_sample_means = sd(sample_mean),
          sd_of_sample_sds = sd(sample_sd)) |>
  mutate_if(is.numeric, janitor::round_half_up, digits = 2) |>
  kable() |>
  kable_classic(full_width = FALSE)
```

| n | average_sample_means | average_sample_sds | sd_of_sample_means | sd_of_sample_sds |
|---:|---:|---:|---:|---:|
| 50 | 0 | 1 | 0.13 | 0.10 |
| 100 | 0 | 1 | 0.10 | 0.07 |
| 150 | 0 | 1 | 0.08 | 0.06 |

Why does this matter? Two reasons:

1. It's useful to broaden our understanding of the concepts of uncertainty around estimates. In a previous lesson we learned that tests of assumptions are themselves just hypothesis tests applied to other parameters (e.g., differences in SDs instead of means). This is a related idea, that all sample estimates have uncertainty, not just means and standardized effect sizes.

2. Because Cohen's d involves at least four estimates (two means and two SDs), there are additional ways for it to be biased. The more parameters we have to estimate, the more things we can get wrong. We'll return to this idea in another RMarkdown document.

## Session info

```
sessionInfo()
```

```
## R version 4.3.2 (2023-10-31 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 11 x64 (build 22631)
##
## Matrix products: default
##
##
## locale:
## [1] LC_COLLATE=German_Switzerland.utf8  LC_CTYPE=German_Switzerland.utf8
## [3] LC_MONETARY=German_Switzerland.utf8 LC_NUMERIC=C
## [5] LC_TIME=German_Switzerland.utf8
##
## time zone: Europe/Zurich
## tzcode source: internal
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] kableExtra_1.4.0 knitr_1.46       sn_2.1.1         effsize_0.8.1
##  [5] ggplot2_3.5.1    purrr_1.0.2      readr_2.1.5      forcats_1.0.0
##  [9] dplyr_1.1.4      tidyr_1.3.1
##
## loaded via a namespace (and not attached):
##  [1] utf8_1.2.4          generics_0.1.3      xml2_1.3.6
##  [4] stringi_1.8.3       hms_1.1.3           digest_0.6.35
##  [7] magrittr_2.0.3      timechange_0.3.0    evaluate_0.23
## [10] grid_4.3.2          fastmap_1.1.1       fansi_1.0.6
## [13] viridisLite_0.4.2   scales_1.3.0        numDeriv_2016.8-1.1
## [16] mnormt_2.1.1        cli_3.6.2           rlang_1.1.3
## [19] munsell_0.5.1       withr_3.0.0         yaml_2.3.8
## [22] tools_4.3.2         tzdb_0.4.0          colorspace_2.1-0
## [25] vctrs_0.6.5         R6_2.5.1            lubridate_1.9.3
## [28] lifecycle_1.0.4     snakecase_0.11.1    stringr_1.5.1
## [31] janitor_2.2.0       pkgconfig_2.0.3     pillar_1.9.0
## [34] gtable_0.3.5        glue_1.7.0          systemfonts_1.0.6
## [37] xfun_0.43           tibble_3.2.1        tidyselect_1.2.1
## [40] highr_0.10          rstudioapi_0.16.0   farver_2.1.1
## [43] htmltools_0.5.8.1   rmarkdown_2.26      svglite_2.1.3
## [46] labeling_0.4.3      compiler_4.3.2
```