Preregistration: Are people more uninformed than misinformed? A systematic review and meta analysis of experimental literature on perceived news accuracy

Anonymous

2023-04-27

```
# load required packages
library("lme4")        # model specification / estimation
library("lmerTest")    # provides p-values in the output
library("tidyverse")   # data wrangling and visualisation
library("afex")        # anova and deriving p-values from lmer
library("broom")       # extracting data from model fits
library("broom.mixed") # extracting data from mixed models
library("metafor")     # doing mata analysis
library("metaviz")     # vizualization of meta analysis
library("stringr")     # for dmetar p-curve function to work
library("poibin")      # for dmetar p-curve function to work
library("patchwork")   # put several plots together
library("ggridges")    # for plots
library("gghalves")    # for plots
library("ggbeeswarm")  # Special distribution-shaped point jittering
library("knitr")       # for tables
library("kableExtra")  # also for tables


# load required functions from dmetar package
source("variance_composition_function.R") # calculates variance composition for metafor output
source("p_curve_dmetar_function.R")       # generates p_curve and according estimates

# load plot theme
source("plot_theme.R") # extracts correlation from individual level data subset
```

Introduction

In this meta-analysis, we review experimental literature that measures perceived accuracy of both fake and true news. When judging the veracity of news, people can engage in two types of errors: judging fake news as accurate or judging true news as not accurate. Our main research question is which error is more common. In order to establish meaningful comparisons across different experimental studies, our data consists of control groups only.

Data collection

We have collected data, following the PRISMA guidelines. We have not performed any statistical analysis on the data.

We undertook a systematic review and meta-analysis of the experimental literature on perceived accuracy of news. We included peer-reviewed publications and grey literature (for example, reports, pre-prints and working papers).

Eligibility criteria. Prior to the systematic search, we set certain criteria necessary for studies to be included in our analysis. All these criteria can be found in `literature_search/inclusion_criteria.csv`. We considered relevant all document types with original data (not only published ones, but also reports, pre-prints and working papers). We only considered papers that measured the perceived accuracy of both fake and true news. Our interpretation of 'accuracy' was liberal, including not onl studies that were asking participants about how "accurate", but also those asking how "credible", "trustworthy", "reliable" or "manipulative" news items were. We also required that studies need to rely on real-world news items (i.e. present participants with fake and true news that actually circled). We excluded studies in which researchers had made up their own fake news, or manipulated properties of true news. The only exception we granted here were two studies that used artificial intelligence to generate fake news headlines taking real-world fake news headlines as input. made an exception her We included only studies with unique data, which was a relevant when different publications were using on the same data. Finally, we could only include those studies that provided us with the relevant summary statistics (means and standard deviations for both fake and true news), or publicly available data that allowed us to calculate those. In cases where we were not able to retrieve the relevant summary statistics either way, but a study was otherwise relevant, we contacted the authors. After starting the literature search, we added further criteria in order to diminish the vast number of results (see next section).

Literature search. We first conducted as Scopus search, using the following string:

> ' "false news" OR "fake news" OR "false stor*" AND "accuracy" OR "discernment" OR "credibilit*" OR "belief" OR "susceptib*" '

Given the high volume of papers (12425), we added restrictions to only include articles that were likely (i) experimental, (ii) and exposed participants to both true and false news.

> 'AND (LIMIT-TO (LANGUAGE , "English")) AND (LIMIT-TO (DOCTYPE , "ar") OR LIMIT-TO (DOCTYPE , "cp")) AND (EXCLUDE (SUBJAREA , "PHYS") OR EXCLUDE (SUBJAREA , "MATE") OR EXCLUDE (SUBJAREA , "BIOC") OR EXCLUDE (SUBJAREA , "ENER") OR EXCLUDE (SUBJAREA , "IMMU") OR EXCLUDE (SUBJAREA , "AGRI") OR EXCLUDE (SUBJAREA , "PHAR") OR EXCLUDE (SUBJAREA , "HEAL") OR EXCLUDE (SUBJAREA , "EART") OR EXCLUDE (SUBJAREA , "NURS") OR EXCLUDE (SUBJAREA , "CHEM") OR EXCLUDE (SUBJAREA , "CENG") OR EXCLUDE (SUBJAREA , "VETE") OR EXCLUDE (SUBJAREA , "DENT")) AND (EXCLUDE (SUBJAREA , "COMP") OR EXCLUDE (SUBJAREA , "ENGI") OR EXCLUDE (SUBJAREA , "MATH") OR EXCLUDE (SUBJAREA , "MEDI"))'

We excluded papers not written in English, that were not articles or conference papers, that were from disciplines that are likely irrelevant for the present search (e.g., Dentistry, Veterinary, Chemical Engineering, Chemistry, Nursing, Pharmacology, Microbiology, Materials Science, Medicine) or unlikely to use an experimental design (e.g. Computer Science, Engineering, Mathematics). After these filters were applied, we ended up with 4002 results.

Second, we conducted an advanced Google scholar search via the "Publish or Perish" software using the following string:

‘ “Fake news” | “False news”|“False stor*” “Accuracy” | “Discernment”|“Credibility”|“Belief”|“Suceptib*”, no citations, no patents’

The main advantage of this search was to identify important pre-prints or working papers that the Scopus search would have missed. The Google scholar search yielded 980 results.

All records resulting from the literature searches can be found in `literature_search/scopus.csv` and `literature_search/google.csv`.

After removing 156 duplicates from the two searches, we ended up with 4826 documents for screening. We screened records based on titles only. The vast majority of documents (4723) had irrelevant titles and were removed during that phase. Most irrelevant titles were not about false news or misinformation (e.g. “Formation of a tourist destination image: Co-occurrence analysis of destination promotion videos”), and some were about false news or misinformation but were not about belief or accuracy (e.g. “Freedom of Expression and Misinformation Laws During the COVID-19 Pandemic and the European Court of Human Rights”).

We stored the remaining 103 records in the reference management system Zotero for retrieval. Of those, we rejected a total of 72 papers that did not meet our inclusion criteria. We rejected 5 papers based on their abstract and 67 after assessment of the full text. All exclusion decisions are documented in `literature_search/excluded.csv`.

We included the remaining 31 papers from the systematic literature search.

To complement the systematic search results, we conducted forward and backward citation search through Google Scholar. We also reviewed studies that we had on our computers and papers we found scrolling through twitter (mostly unpublished manuscripts). Taken together we identified an additional 34 papers via those methods. Of these, we excluded 18 papers after full text assessment because they did not meet our inclusion criteria. For these papers, too, our exclusion decisions are documented in `literature_search/excluded.csv`. We included the remaining 16 papers.

In total, we included 47 papers in our meta analysis. We retrieved the relevant summary statistics directly from the paper for 22 papers, calculated them ourselves based on publicly available raw data for 22 papers, and got them from the authors after request for 10 papers.

Throughout this document, we use the previously generated simulation data from the `simulation_generate_data` document and models informed by the `simulation_analyze_data` document.

Anticipating another study, we have not only collected data on control, but also treatment groups (i.e. samples that received an intervention, typically intended to alter discernment). However, for this study, as stated in the introduction, we restrict our sample to *control groups only*. This ensures that we only compare effects across studies with comparable designs, hence measuring comparable concepts.

```
# load data
meta_wide <- read_csv("./data_from_simulation/meta.csv") %>%
  # control conditions only
  filter(condition == "control")
```

Measures

We have two main outcome variables:

- `accuracy` (as reported in the studies).
- `error` (as computed by us based on the accuracy ratings reported in the studies).

Our independent variable is:

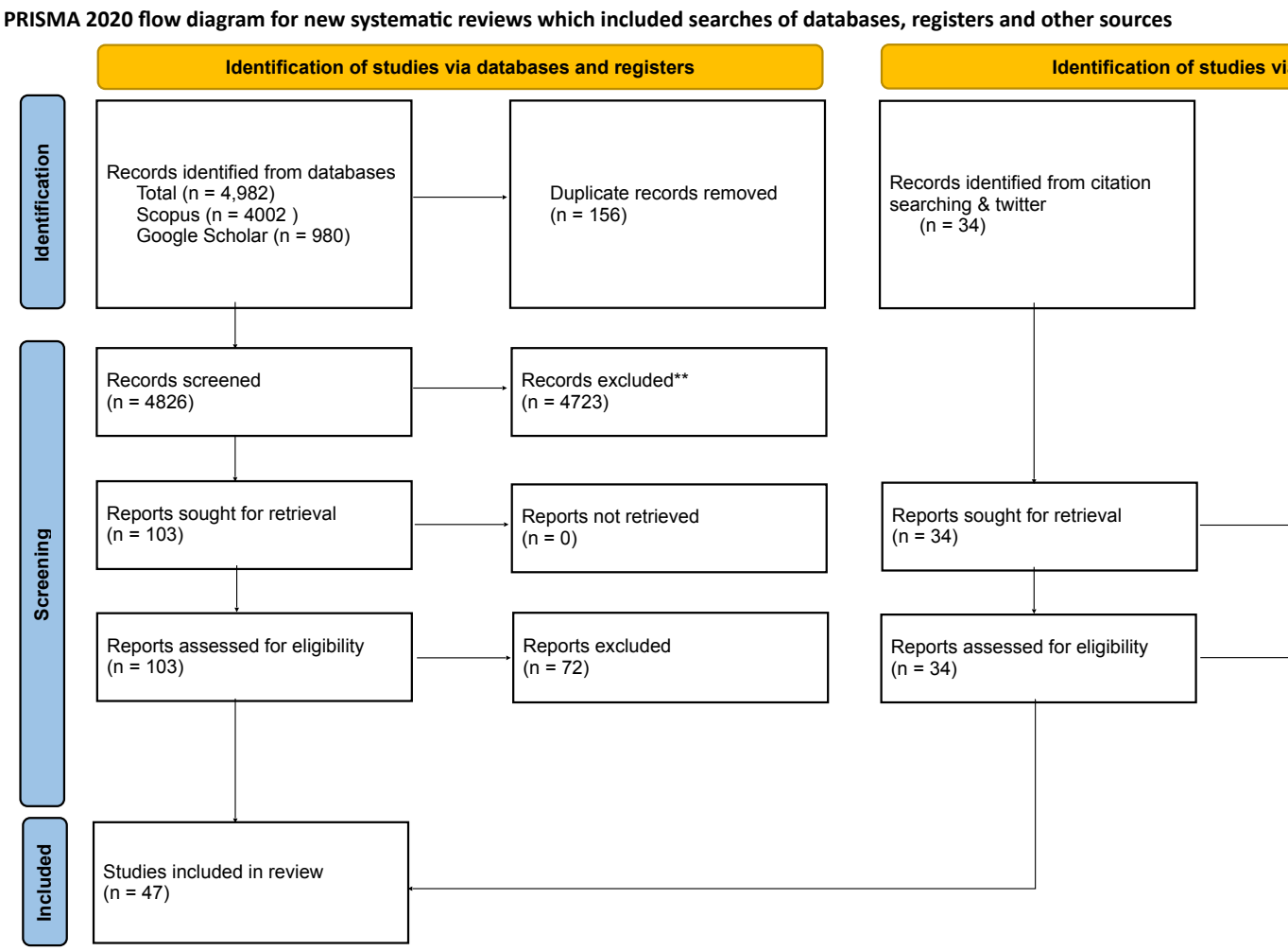


Figure 1: PRISMA flowchart

Table 1: Calculation of error variable

continuous scale	For fake news, error is defined as the distance of the accuracy value to the bottom of the scale (e.g. a
binary scale	Similar to the procedure for continuous scales, except for that there is no 'bo

- `veracity` (whether news items were true or fake)

For the meta analysis, we need to compute the effect size of `veracity` on both `accuracy` and `error`.

Accuracy

Different studies measure accuracy in different ways. The most frequent measure is a 4-point response scale (e.g. "To the best of your knowledge, how accurate is the claim in the above headline? -Not at all accurate (1) -Not very accurate (2) -Somewhat accurate (3) -Very accurate (4)").

However, some studies also use different response scales (e.g. "To the best of your knowledge, is the claim in the above headline accurate?" from 1 Extremely inaccurate to 6 Extremely accurate).

Some studies also use binary scales (e.g. "Is this news article real or fake?" Real vs. Fake). We address concerns from comparing across continuous and binary scales in the robustness checks section.

Calculate error variable

Generally, we define error as the distance of the observed average accuracy ratings per sample to the (fictive) best outcome that could have could have possibly been attained in the sample. For fake news, that means *never* responding "accurate"; for true news, that means *always* responding "accurate".

The precise way we calculate the `error` variable depends on `veracity` and `scale`. The respective calculation procedures are illustrated in the table below.

Before we calculate the error, we need to create a numeric version of the `accuracy_scale` variable.

```
# To later be able to calculate the error, we need to transform this variable
# into a numeric first.
# This is a step we will also have to do with our actual data in the analysis.

is.character(meta_wide$accuracy_scale) # TRUE
```

```
## [1] TRUE
```

```
# as.numeric() builds a numeric variable based on the original, character one.
# As a result, "binary" becomes "NA", all other numeric-style entries are
# persevered. Let's check this first.
table(as.numeric(meta_wide$accuracy_scale), useNA = "always")
```

```
## Warning in table(as.numeric(meta_wide$accuracy_scale), useNA = "always"): NAs
## introduced by coercion
```

```
##
##    4    6    7 <NA>
##   47   26   39   21
```

```r
# apply
meta_wide <- meta_wide %>%
  mutate(accuracy_scale_numeric = as.numeric(accuracy_scale))
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `accuracy_scale_numeric = as.numeric(accuracy_scale)`.
## Caused by warning:
## ! NAs introduced by coercion
```

We calculate the error as follows.

```r
# calculate `error` variable
meta_wide <- meta_wide %>%
  mutate(
    # calculate error for true news
    error_true = ifelse(accuracy_scale == "binary",
                        # for binary scales, the maximum is 1
                        1 - mean_accuracy_true,
                        # for continuous scales, the maximum is the value of
                        # `accuracy_scale_numeric`
                        accuracy_scale_numeric - mean_accuracy_true
                        ),
    # calculate error for fake news
    error_fake = ifelse(accuracy_scale == "binary",
                        # for binary scales, the minimum is 0
                        mean_accuracy_fake - 0,
                        # for continuous scales, the minimum is 1
                        mean_accuracy_fake - 1)
    )
```

Calculate effect sizes

Standardized Mean Differences (SMDs) Above we stated that we have two outcome variables, `accuracy` and `error`. But meta analyses are run on effects, not variables. In our case, these effects are the differences between `true` and `fake` news regarding `accuracy` and `error`.

Essentially, we take the means (of `accuracy` and `error`, respectively) for `true` and `fake` news and calculate their difference, i.e. the 'mean difference' (`MD`).

$$\text{MD} = \bar{x}_{true} - \bar{x}_{fake}$$

However, we calculate *standardized* mean differences (`SMDs`). This allows to compare the differences across the different scales of measurement that different studies use. SMDs report differences measured in units of standard deviations (`SD`).

$$\text{SMD} = \frac{MD}{SD_{\text{pooled}}}$$

where SD_{pooled} is the average of SD_{true} and SD_{fake}

$$SD_{\text{pooled}} = \sqrt{\frac{SD_{true}^2 + SD_{fake}^2}{2}}$$

Standard Errors Ultimately, we want to compare across all the SMDs. In meta analyses, more weight is given to those studies that make estimates based on a larger sample size, i.e. those studies with smaller standard errors (`SEs`).

The SE for a SMD is calculated as:

$$SE_{\text{SMD}_{\text{between}}} = \sqrt{\frac{n_1 + n_2}{n_1 n_2} + \frac{\text{SMD}^2}{2(n_1 + n_2)}}$$

Where n_1 and n_2 are the sample sizes of group 1 (e.g. fake news) and group 2 (e.g. true news), and with SMD being the calculated (between-group) standardized mean difference.

Thus, to calculate SE's, we first need a sample size. In our data we collect both the number of participants (`n_subj`) and the number news items each participant saw (`n_news`). We define sample size as the number of observations (`n_observations`) within a study, where `n_observations = n * n_news`.

```
# add `n_observations` variable
meta_wide <- meta_wide %>%
  mutate(n_observations = n_subj*n_news)
```

Second, we need to account for the fact that our measures for fake and true news are *not independent* of each other. That's because most of the studies use a *within participant* design regarding `veracity`. In other words, in most studies the same participant would rate both `true` and `fake` news.

The within participants SMD (or SMD_{within}) is also referred to as a *standardized mean change using change score standardization* in the literature (Gibbons, Hedeker, and Davis 1993).

The *within* aspect does in fact not change the way the SMD is calculated (see above); but it does change the way its SE is calculated.

$$SE_{\text{SMD}_{\text{within}}} = \sqrt{\frac{2(1 - cor_{\text{true/fake}})}{n} + \frac{\text{SMD}^2_{\text{within}}}{2n}}$$

where $cor_{\text{true/fake}}$ is the correlation for fake and true news for a given observation (i.e. a sample).

So, for each observation (samples) in our data, we need an estimate of the correlation between fake and true news ratings. Unfortunately, this correlation is generally not reported. We can only obtain it for a subset of studies for which we collected the summary statistics ourselves, based on the raw data.

What we can do, however, is make a guess for an average correlation. This approach is in line with the Cochrane recommendations for crossover trials (Higgins et al. 2019).

We base this guess on those studies that we have raw, participant-level data for. We calculate an average by-participant correlation across all samples and take the average. The procedure is documented in the `compute_correlation.R` script.

```
# get average correlation for samples from the raw data subset
average_correlation <- read_csv("correlations_by_sample.csv") %>%
  summarise(mean_r = mean(r)) %>% pull(mean_r)
```

```
## Rows: 22 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): paper_id, condition, scale
## dbl (2): sample_id, r
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# add a column to the data frame that identifies the correlation
meta_wide <- meta_wide %>%
  mutate(cor = average_correlation)
```

With this correlation, we can calculate the standardized mean differences *accounting for dependence* due to the within participant design.

Some studies have also tested veracity *between* participants, i.e. participants saw only either fake or true news. For these, we have to calculate effect sizes assuming *independence*. The effect size we'll calculate in this case is a *standardized mean difference* (Hedges 1981).

In our simulated data, we assumed *within* participant designs only.

```
# create an indicator variable called `design` for the simulated data
# (in this case a constant with only the level 'within')
# in the real world data, we already have this variable coded
meta_wide <- meta_wide %>%
  mutate(design = "within")
```

For all effect size calculations, we rely on the `metafor` package (Viechtbauer 2010)

```
# Within participant studies

# "SMCC" for the standardized mean change using change score standardization
# use escalc function
within_participants <- escalc(measure="SMCC",
                          # diff = true (m1i) - fake (m2i)
                          m2i= mean_accuracy_fake,
                          sd2i=sd_accuracy_fake,
                          ni=n_observations,
                          m1i=mean_accuracy_true,
                          sd1i=sd_accuracy_true,
                          data = meta_wide %>%
                            # restrict to within studies only
                            filter(design == "within"),
                          ri = cor)


# this is just for demonstration purposes with simulated data to
accuracy_effect <- within_participants # for our actual data, we'll use the code
# below instead

# Between participant studies.
# we use the escalc function from the metafor package
# standardized mean difference (SMD)
# between_participants <- escalc(measure="SMD",
#                              # diff = true (m1i) - fake (m2i)
```

```
#                               m2i= mean_accuracy_fake,
#                               sd2i=sd_accuracy_fake, n2i=n_observations,
#                               m1i=mean_accuracy_true, sd1i=sd_accuracy_true,
#                               n1i=n_observations,
#                               data = meta_wide %>%
#                               # restrict to between studies only
#                               filter(design == "between"))

# Re-unite both designs in a single data frame
# accuracy_effect <- rbind(within_participants %>% select(-r), between_participants)
```

Accuracy effect sizes

Error effect sizes Note that we obtain error - for both fake and true news - by subtracting a constant from accuracy ratings (a distinct one for true and fake news respectively). The error therefore has the same standard deviation as accuracy.

```
# Within participant studies

# "SMCC" for the standardized mean change using change score standardization
# use escalc function
within_participants <- escalc(measure="SMCC",
                              # diff = true (m1i) - fake (m2i)
                              m2i= error_fake,
                              sd2i=sd_accuracy_fake,
                              ni=n_observations,
                              m1i=error_true,
                              sd1i=sd_accuracy_true,
                              data = meta_wide %>%
                                # restrict to within studies only
                                filter(design == "within"),
                              ri = cor)


# this is just for demonstration purposes with simulated data to
error_effect <- within_participants # for our actual data, we'll use the code
# below instead

# Between participant studies.
# we use the escalc function from the metafor package
# standardized mean difference (SMD)
# between_participants <- escalc(measure="SMD",
#                               # diff = true (m1i) - fake (m2i)
#                               m2i= error_fake,
#                               sd2i=sd_accuracy_fake, n2i=n_observations,
#                               m1i=error_true, sd1i=sd_accuracy_true,
#                               n1i=n_observations,
#                               data = meta_wide %>%
#                               # restrict to between studies only
#                               filter(design == "between"))

# Re-unite both designs in a single data frame
# error_effect <- rbind(within_participants %>% select(-r), between_participants)
```

H1: People rate true news as more accurate than fake news

Some of the observations (samples) are *dependent* - either because the same sample was repeatedly tested or because we had several measures (e.g. when we split observations according to covariates, such as news type).

To account for this dependency, we use a) multi-level random effect model and b) calculate robust standard errors.

a) multi-level random effect model Precisely, we use a three-level model with two nested grouping variables. We assume that

- participants (*level 1*; unobserved since we collected summary data)
- are nested within observations, i.e. single lines in our meta data frame, i.e. individual effect sizes (*level 2* ; defined by `observation_id`)
- are nested within samples (*level 3*; defined by `sample_id`).

```
# Multilevel random effect model for accuracy
model_accuracy <-  metafor::rma.mv(yi, vi, random = ~ 1 | unique_sample_id / observation_id, data=accura
model_accuracy
```

```
##
## Multivariate Meta-Analysis Model (k = 133; method: REML)
##
## Variance Components:
##
##             estim    sqrt  nlvls  fixed                          factor
## sigma^2.1  0.1119  0.3345     74     no                 unique_sample_id
## sigma^2.2  0.0000  0.0050    133     no  unique_sample_id/observation_id
##
## Test for Heterogeneity:
## Q(df = 132) = 106053.3468, p-val < .0001
##
## Model Results:
##
## estimate      se     zval    pval   ci.lb   ci.ub
##   1.0758  0.0389  27.6301  <.0001  0.9995  1.1521  ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

b) robust standard errors When several observations are from the same sample, we expect their sampling errors to be correlated. This, however, is not yet captured by our three-level model. We therefore additionally calculate robust standard errors clustered at the sample level.

```
# robust standard errors clustered at the sample level
robust_model_accuracy <- robust(model_accuracy,
                                cluster = accuracy_effect$unique_sample_id)
robust_model_accuracy
```

```
##
## Multivariate Meta-Analysis Model (k = 133; method: REML)
```

```
##
## Variance Components:
##
##             estim    sqrt  nlvls  fixed                           factor
## sigma^2.1  0.1119  0.3345     74     no                unique_sample_id
## sigma^2.2  0.0000  0.0050    133     no  unique_sample_id/observation_id
##
## Test for Heterogeneity:
## Q(df = 132) = 106053.3468, p-val < .0001
##
## Number of estimates:   133
## Number of clusters:    74
## Estimates per cluster: 1-4 (mean: 1.80, median: 1)
##
## Model Results:
##
## estimate      se[1]     tval[1]  df[1]    pval[1]   ci.lb[1]   ci.ub[1]
##   1.0758  0.0389   27.6301   73   <.0001   0.9982   1.1534   ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## 1) results based on cluster-robust inference (var-cov estimator: CR1,
##    approx t-test and confidence interval, df: residual method)
```

Interpreting the results. We start from the top of the output table with the `Variance Components`. We see random-effects variances calculated for each level of our model. The first one, `sigma^2.1`, shows the *level 3* between-cluster variance. In our example, this is equivalent to the between-sample heterogeneity variance (corresponding to 'tau^2' in a conventional meta-analysis, since clusters represent samples in our model).

The second variance component `sigma^2.2` shows the variance within clusters (*level 2*), i.e. within-sample variation of observations in our example.

In the `nlvls` column, we see the number of groups on each level.

Under `Model Results`, we see the `estimate` of our pooled effect of `veracity`.

Distribution of variance. The variance components are hard to interpret in the raw metafor output. What we are interested in is the distribution of variance over the three levels of our model. Harrer et al. (2021) write that, "In conventional meta-analyses, I2 represents the amount of variation not attributable to sampling error. In three-level models, this heterogeneity variance is split into two parts: one attributable to true effect size differences within clusters, and the other to between-cluster variation. Thus, there are two I2 values, quantifying the percentage of total variation associated with either level 2 or level 3."

Here, we rely on a their function to calculate this variance decomposition.

```
variance_composition <- var.comp(model_accuracy)
variance_composition$results %>%
    mutate_if(is.numeric, round, 3)
```

```
##         % of total variance    I2
## Level 1               0.154   ---
## Level 2               0.022  0.02
## Level 3              99.823 99.82
```

The sampling error variance on *level 1* (the individual observation level) is very, very small. The value of I2 *Level 2*, i.e. the amount of heterogeneity variance within samples, is estimated to be 0. That is coherent

with the way we generated our data: All systematic differences for observations stem from *between* sample differences. Accordingly, the bulk share falls to I2 *level 3*, the between-sample heterogeneity.

Overall, this indicates that there is substantial between-study heterogeneity on the third level.

However, I2 is simply the percentage of variability not caused by sampling error. If our studies become increasingly large, the sampling error (level 1) tends to zero, while at the same time, I2 (level 2 and 3 combined) tends to 100%.

Considering this limitation of I2, another useful way to interpret the variance component estimates (i.e `sigma^2.1` and `sigma^2.2`in our model output) is to calculate prediction intervals. Prediction intervals give us a range into which we can expect the effects of future studies to fall based on present evidence. For example, if our prediction interval does not include 0 and lies completely on the "positive" side, we would expect any future study to find people rating true news as more accurate than fake news (under the assumption that the study context is similar).

```
predict_model_accuracy <- predict(robust_model_accuracy)
predict_model_accuracy
```

```
##
##    pred     se  ci.lb  ci.ub  pi.lb  pi.ub
##  1.0758 0.0389 0.9982 1.1534 0.4045 1.7471
```

H2: People are better at rating false news as false than true news as true

We proceed just as for H1, except for that this time our dependent variable is `error`.

```
# Multilevel random effect model for error
model_error <-  metafor::rma.mv(yi, vi, random = ~ 1 | unique_sample_id / observation_id, data=error_ef
model_error
```

```
##
## Multivariate Meta-Analysis Model (k = 133; method: REML)
##
## Variance Components:
##
##             estim    sqrt  nlvls  fixed                          factor
## sigma^2.1  0.4821  0.6943     74     no                 unique_sample_id
## sigma^2.2  0.0001  0.0099    133     no  unique_sample_id/observation_id
##
## Test for Heterogeneity:
## Q(df = 132) = 387689.0212, p-val < .0001
##
## Model Results:
##
## estimate      se    zval    pval   ci.lb   ci.ub
##   0.3264  0.0807  4.0425  <.0001  0.1681  0.4846  ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# accuracy: robust standard errors clustered at the sample level
robust_model_error <- robust(model_error,
                             cluster = error_effect$unique_sample_id)
robust_model_error
```

```
##
## Multivariate Meta-Analysis Model (k = 133; method: REML)
##
## Variance Components:
##
##             estim    sqrt  nlvls  fixed                          factor
## sigma^2.1  0.4821  0.6943     74     no                 unique_sample_id
## sigma^2.2  0.0001  0.0099    133     no  unique_sample_id/observation_id
##
## Test for Heterogeneity:
## Q(df = 132) = 387689.0212, p-val < .0001
##
## Number of estimates:   133
## Number of clusters:    74
## Estimates per cluster: 1-4 (mean: 1.80, median: 1)
##
## Model Results:
##
## estimate      se¹    tval¹  df¹    pval¹   ci.lb¹   ci.ub¹
##   0.3264  0.0807   4.0425   73   0.0001   0.1655   0.4873   ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## 1) results based on cluster-robust inference (var-cov estimator: CR1,
##    approx t-test and confidence interval, df: residual method)
```

```
variance_composition <- var.comp(model_error)
variance_composition$results %>%
    mutate_if(is.numeric, round, 3)
```

```
##         % of total variance    I2
## Level 1               0.028   ---
## Level 2               0.020  0.02
## Level 3              99.952 99.95
```

```
predict_model_error <- predict(robust_model_error)
predict_model_error
```

```
##
##    pred     se  ci.lb  ci.ub   pi.lb  pi.ub
##  0.3264 0.0807 0.1655 0.4873 -1.0669 1.7196
```

Research questions: Moderator effects

We want to know to which extent features of research design moderate the effect of `veracity` on both `accuracy` and `error`.

We consider the following moderator variables:

- `country_grouped`: Country where sample has been tested. The vast majority of studies has been done in the US. We will coded this variable as binary (levels: `US`, `non-US`).

- `political_concordance`: Binary variable with two levels (contained in `news_family` variable; levels: `politically_concordant` and `politically_discordant`).
- `news_family_grouped`: We will cluster the different news families in three levels: `political` (including concordant vs. discordant), `covid` and `other` (historical; environment/health/science/ /military). The baseline for the regression will be `political`.
- `news_format_grouped`: which format was the news presented in (levels: `headline`, `headline_picture`, `headline_picture_lede`).
- `news_source`: whether news items were accompanied by a news source or not (levels: `TRUE`, `FALSE`)
- `accuracy_scale_grouped`: Indicates the the number of options in answering questions (levels: `binary` for binary response scales, `4`, `6`, `7`, `other` for all other scales). Note that we use 4-point scale as reference in the regression.

Note that the selection of the moderators is guided not only by the theoretical or practical relevance of these moderators, but also their distribution in our dataset. For instance, we exclude moderators with very low variance (e.g. only 3 studies were conducted in person and not online, we thus do not test the moderating effect of online vs offline). The same goes for the grouping levels for each moderator - we tried to build coherent categories as long as there seemed to be enough observations per level.

We run a separate meta regression for each of these moderators. The resulting estimates are the ones we will interpret. We will also run a model that includes all moderator variables as covariates (but we will not interpret the resulting estimates).

```
# separate models per moderator
moderator_models <- list(

  accuracy_country <-  robust(metafor::rma.mv(yi, vi,
                                              mods = ~country_grouped,
                                              random = ~ 1 | unique_sample_id /
                                                observation_id, data=accuracy_effect),
                              cluster = accuracy_effect$unique_sample_id
  ),

  accuracy_news_family <-  robust(metafor::rma.mv(yi, vi,
                                                  mods = ~news_family_grouped,
                                                  random = ~ 1 | unique_sample_id /
                                                    observation_id, data=accuracy_effect),
                                  cluster = accuracy_effect$unique_sample_id
  ),

  accuracy_news_format <-  robust(metafor::rma.mv(yi, vi,
                                                  mods = ~news_format,
                                                  random = ~ 1 | unique_sample_id /
                                                    observation_id, data=accuracy_effect),
                                  cluster = accuracy_effect$unique_sample_id
  ),

  accuracy_news_source <-  robust(metafor::rma.mv(yi, vi,
                                                  mods = ~news_source,
                                                  random = ~ 1 | unique_sample_id /
                                                    observation_id, data=accuracy_effect),
                                  cluster = accuracy_effect$unique_sample_id
  ),

  accuracy_all <- robust(metafor::rma.mv(yi, vi,
```

Table 2: Moderator effects

	(1)	(2)	(3)	(4)	(5)
intercept	1.072	1.079	1.072	1.092	1.091
	(0.040)	(0.039)	(0.039)	(0.044)	(0.046)
country_groupedUS	0.007				0.006
	(0.004)				(0.006)
news_family_groupedother		−0.006			−0.006
		(0.006)			(0.006)
news_family_groupedpolitical		−0.004			−0.012
		(0.004)			(0.006)
news_formatheadline_picture			0.006		0.012
			(0.003)		(0.007)
news_formatheadline_picture_lede			0.007		0.009
			(0.006)		(0.009)
news_sourceTRUE				0.002	−0.003
				(0.005)	(0.007)
Num.Obs.	131	130	130	92	87
AIC	−230.7	−227.0	−228.0	−103.3	−97.9
BIC	−219.2	−212.7	−213.7	−93.2	−75.7

```
                                        mods = ~country_grouped + news_family_grouped +
                                          news_format + news_source,
                                        random = ~ 1 | unique_sample_id /
                                          observation_id, data=accuracy_effect),
                          cluster = accuracy_effect$unique_sample_id
  )

)
```

```
## Warning: Rows with NAs omitted from model fitting.

## Warning: Rows with NAs omitted from model fitting.
```

```
modelsummary::modelsummary(moderator_models,
                           title = 'Moderator effects')
```

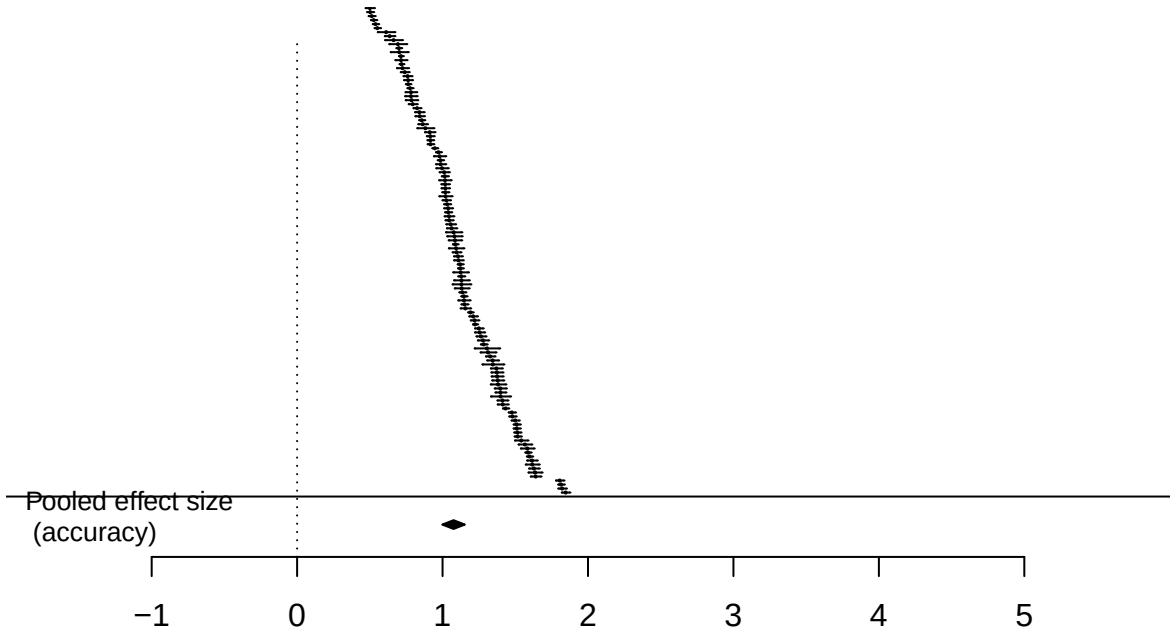
Graphs

H1 and H2

Caterpillar plot, i.e. a forest plot with many estimates which are ordered by their magnitude.

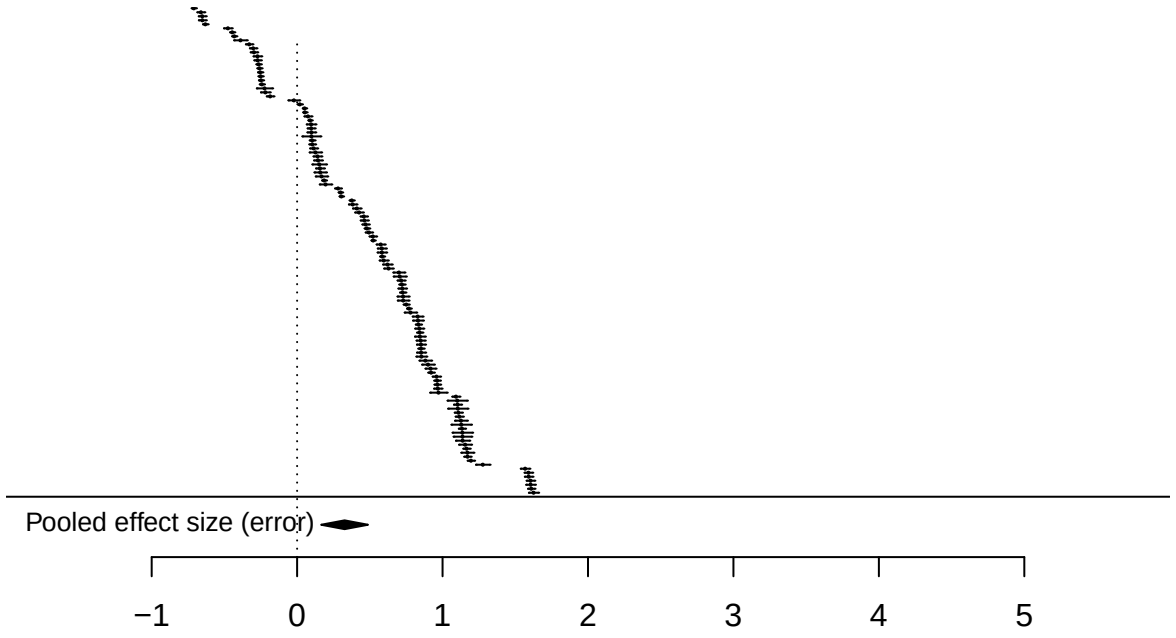
```
### create plot for accuracy
forest.rma(robust_model_accuracy,
       xlim = c(-2, 6),        ### adjust horizontal plot region limits
       at = c(-1, 0, 1, 2, 3, 4, 5),
       order="obs",             ### order by size of yi
       slab=NA, annotate=FALSE, ### remove study labels and annotations
```

```
       efac=c(0, 1),             ### remove vertical bars at end of CIs
       pch=19,                   ### changing point symbol to filled circle
       col="gray40",             ### change color of points/CIs
       psize=1,                  ### increase point size
       cex.lab=0.5, cex.axis=1,   ### increase size of x-axis title/labels
       lty=c("solid", "dotted", "blank"),  ### remove horizontal line at top of plot
       mlab = "BLASt",
       ylim = c(-10, 117),
       addfit = FALSE,
       xlab = "")
addpoly(robust_model_accuracy, mlab="Pooled effect size \n (accuracy)", cex = 0.8, row = -7)
abline(h=0)
```




```
### create plot for error
forest.rma(robust_model_error,
       xlim = c(-2, 6),         ### adjust horizontal plot region limits
       at = c(-1, 0, 1, 2, 3, 4, 5),
       order="obs",              ### order by size of yi
       slab=NA, annotate=FALSE, ### remove study labels and annotations
       efac=c(0, 1),             ### remove vertical bars at end of CIs
       pch=19,                   ### changing point symbol to filled circle
       col="gray40",             ### change color of points/CIs
       psize=1,                  ### increase point size
       cex.lab=0.5, cex.axis=1,   ### increase size of x-axis title/labels
       lty=c("solid", "dotted", "blank"),  ### remove horizontal line at top of plot
```

```
       mlab = "BLASt",
       ylim = c(-10, 117),
       addfit = FALSE,
       xlab = "")
addpoly(robust_model_error, mlab="Pooled effect size (error)", cex = 0.8, row = -7)
abline(h=0)
```



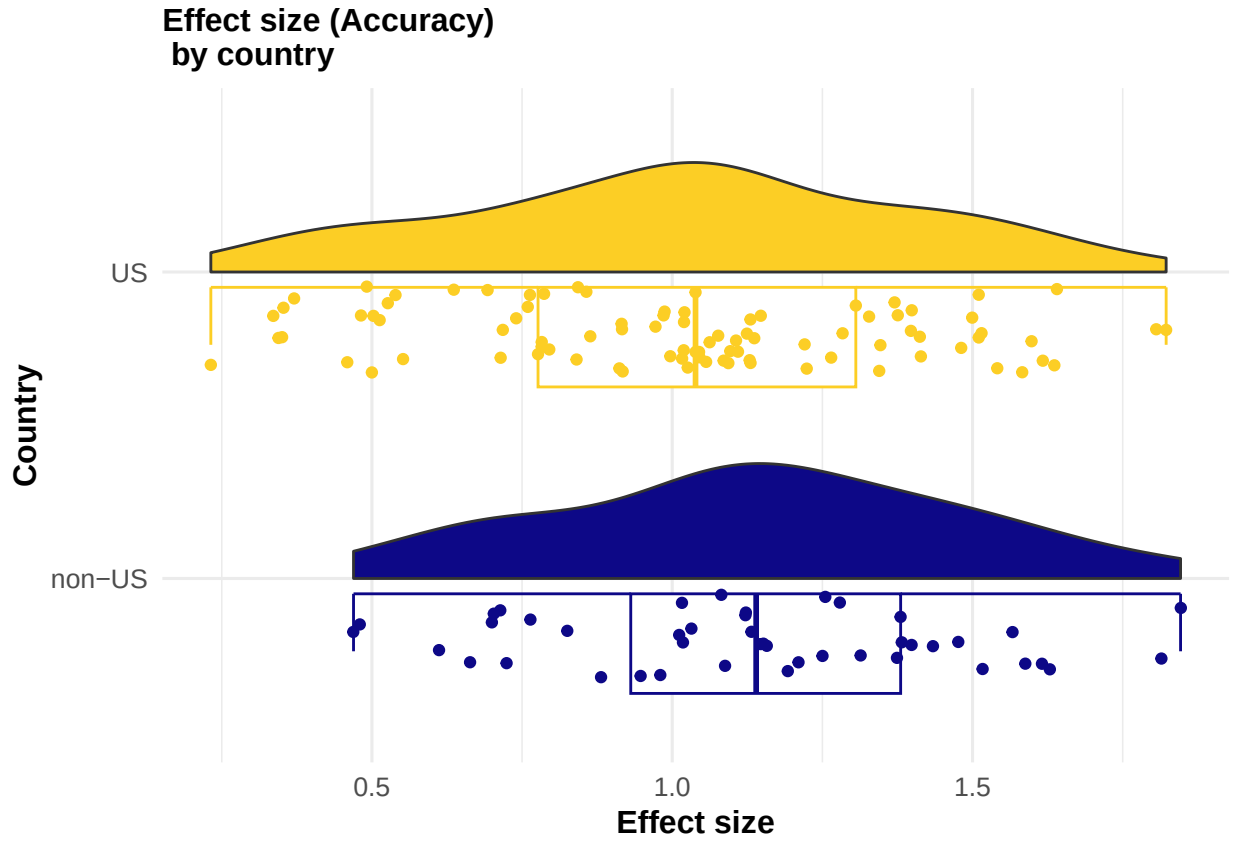
RQ: Moderators

Make several plots to visualize how the different moderator variables affect the effect size.

```
# Effect by country
# Same plot type for `news_source` and `pretested`
ggplot(accuracy_effect,
       aes(x = country_grouped, y = yi)) +
  geom_half_boxplot(aes(color = country_grouped), side = "l", size = 0.5, nudge = 0.05,
                    outlier.shape = NA) +
  geom_half_violin(aes(fill = country_grouped), side = "r") +
  geom_half_point(aes(color = country_grouped), side = "l",
                  transformation_params = list(height = 0, width = 0.1, seed = 1)) +
  # colors
  scale_color_viridis_d(option = "plasma", end = 0.9)+
  scale_fill_viridis_d(option = "plasma", end = 0.9) +
  # labels and scales
```

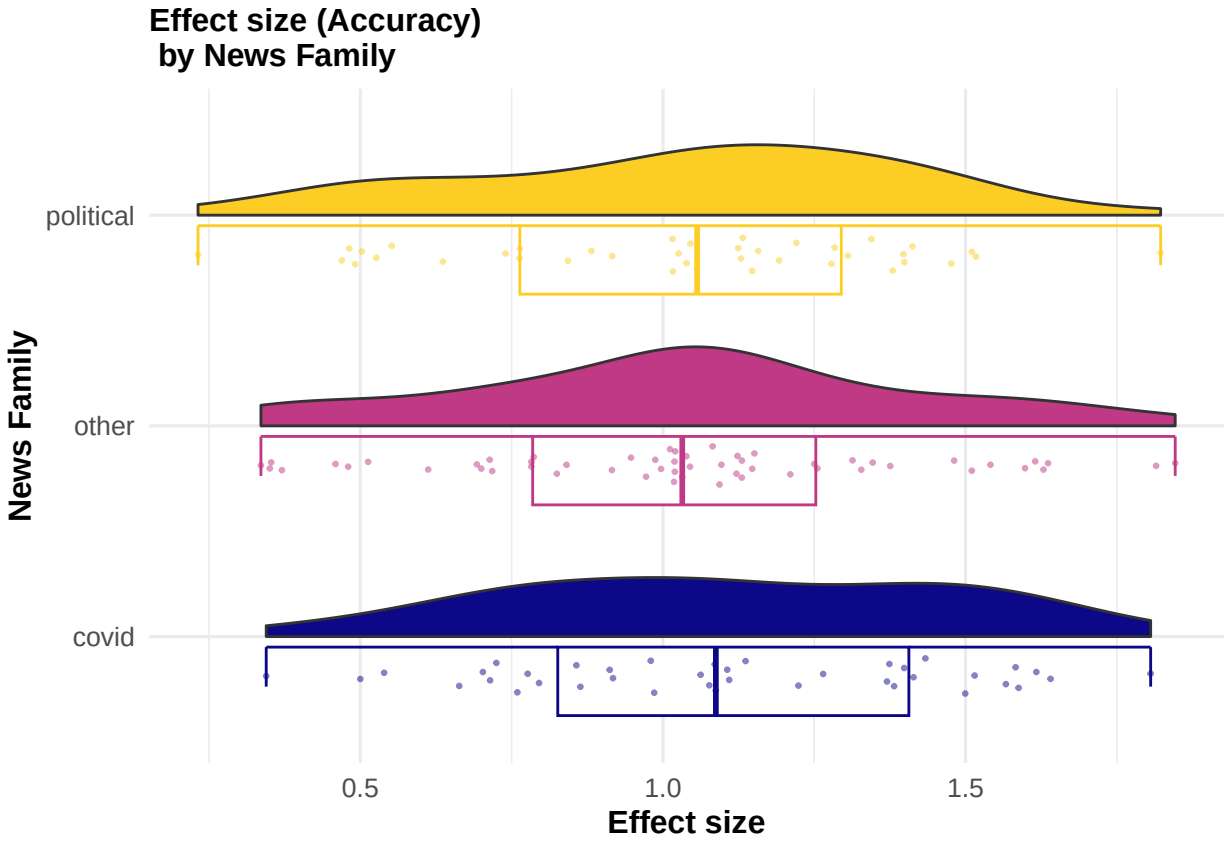
```
  guides(color = "none", fill = "none") +
  labs(x = "Country", y = "Effect size",
       title = "Effect size (Accuracy) \n by country") +
  plot_theme +
  coord_flip()
```

```
## Warning in geom_half_point(aes(color = country_grouped), side = "l",
## transformation_params = list(height = 0, : Ignoring unknown parameters:
## 'transformation_params'
```




```
# Effect by news_family
# same for `news_format`
ggplot(accuracy_effect,
       aes(x = news_family_grouped, y = yi)) +
  geom_half_boxplot(aes(color = news_family_grouped), side = "l", size = 0.5, nudge = 0.05,
                    outlier.shape = NA) +
  geom_half_violin(aes(fill = news_family_grouped), side = "r") +
  geom_half_point(aes(color = news_family_grouped),
                  transformation = position_quasirandom(width = 0.1),
                  side = "l", size = 0.5, alpha = 0.5) +
  # colors
  scale_color_viridis_d(option = "plasma", end = 0.9)+
  scale_fill_viridis_d(option = "plasma", end = 0.9) +
  # labels and scales
```

```
  guides(color = "none", fill = "none") +
  labs(x = "News Family", y = "Effect size",
       title = "Effect size (Accuracy) \n by News Family") +
  plot_theme +
  coord_flip()
```



Robustness checks

When using *standardized* effect measures, accounting for dependency by imputing correlations impacts on the magnitude of the effect estimate itself. That is because the standard deviation is not only used to calculate the standard error (as with *non* standardized effects), but also for the effect itself (difference between true and fake measured in units of standard deviations). Imputed correlations should therefore be accompanied by sensitivity analysis, i.e. check how sensitive the effect estimate is to different imputed correlation values (Higgins et al. 2019).

```
# load individual level subset data and average correlation
individual_level_subset <- read_csv("individual_level_subset.csv")

# load correlations
correlations <- read_csv("correlations_by_sample.csv") %>% pull(r)
```

Imputed average correlation sensitivity analysis

We create two functions. The first one, `robustness_correlation_values` takes a set of correlations as input. It returns the average effect of the mixed meta model for each correlation value.

```
# a function that calculates effects for different correlations,
# for either accuracy or error, and returns the main average effect for the mixed meta
# model for each correlation value in a data frame
robustness_correlation_values <- function(correlations, effect) {

  if (effect == "accuracy") {

    results <- correlations %>%
      # make a loop that for each correlation adds the main effect size in the
      # mixed meta model to a data frame
      map_df(function(x) {

        # Keep track
        print(paste0("Currently estimating model for corrrelation value ", x))

        # Step 1: Add correlation to data frame
        meta_wide <- meta_wide %>%
          mutate(cor = x)

        # Step 2: Calculate effect sizes

        # Within participant studies
        # "SMCC" for the standardized mean change using change score standardization
        # use escalc function
        within_participants <- escalc(measure="SMCC",
                                      # diff = true (m1i) - fake (m2i)
                                      m2i= mean_accuracy_fake,
                                      sd2i=sd_accuracy_fake,
                                      ni=n_observations,
                                      m1i=mean_accuracy_true,
                                      sd1i=sd_accuracy_true,
                                      data = meta_wide %>%
                                        # restrict to within studies only
                                        filter(design == "within"),
                                      ri = cor)

        # this is just for demonstration purposes with simulated data to
        accuracy_effect <- within_participants # for our actual data, we'll use the code
        # below instead

        # Between participant studies.
        # we use the escalc function from the metafor package
        # standardized mean difference (SMD)
        # between_participants <- escalc(measure="SMD",
        #                                # diff = true (m1i) - fake (m2i)
        #                                m2i= mean_accuracy_fake,
        #                                sd2i=sd_accuracy_fake, n2i=ni=n_observations,
        #                                m1i=mean_accuracy_true, sd1i=sd_accuracy_true,
        #                                n1i=ni=n_observations,
```

```
      #                               data = meta_wide %>%
      #                               # restrict to between studies only
      #                               filter(design == "between"))

      # Re-unite both designs in a single data frame
      # accuracy_effect <- rbind(within_participants %>% select(-r), between_participants)

      # Step 3: calculate model
      # Multilevel random effect model for accuracy
      model_accuracy <-  metafor::rma.mv(yi, vi,
                                         random = ~ 1 | unique_sample_id / observation_id,
                                         data=accuracy_effect)


      # robust standard errors clustered at the sample level
      robust_model_accuracy <- robust(model_accuracy,
                                      cluster = accuracy_effect$unique_sample_id)

      # Step 4: return estimate of interest
      return(tidy(robust_model_accuracy) %>%
               mutate(imputed_correlation = x,
                      effect = "accuracy") %>%
               mutate_if(is.numeric, round, 5)
      )
    }
    )
}

else {

  results <- correlations %>%
    # make a loop that for each correlation adds the main effect size in the
    # mixed meta model to a data frame
    map_df(function(x) {

      # Keep track
      print(paste0("Currently estimating model for corrrelation value ", x))

      # Step 1: Add correlation to data frame
      meta_wide <- meta_wide %>%
        mutate(cor = x)

      # Step 2: Calculate effect sizes


      # Within participant studies

      # "SMCC" for the standardized mean change using change score standardization
      # use escalc function
      within_participants <- escalc(measure="SMCC",
                                    # diff = true (m1i) - fake (m2i)
                                    m2i= error_fake,
                                    sd2i=sd_accuracy_fake,
```

```
                                    ni=n_observations,
                                    m1i=error_true,
                                    sd1i=sd_accuracy_true,
                                    data = meta_wide %>%
                                      # restrict to within studies only
                                      filter(design == "within"),
                                    ri = cor)


        # this is just for demonstration purposes with simulated data to
        error_effect <- within_participants # for our actual data, we'll use the code
        # below instead

        # Between participant studies.
        # we use the escalc function from the metafor package
        # standardized mean difference (SMD)
        # between_participants <- escalc(measure="SMD",
        #                                # diff = true (m1i) - fake (m2i)
        #                                m2i= error_fake,
        #                                sd2i=sd_accuracy_fake, n2i=n_observations,
        #                                m1i=error_true, sd1i=sd_accuracy_true,
        #                                n1i=n_observations,
        #                                data = meta_wide %>%
        #                                # restrict to between studies only
        #                                filter(design == "between"))

        # Re-unite both designs in a single data frame
        # error_effect <- rbind(within_participants %>% select(-r), between_participants)

        # Step 3: calculate model
        # Multilevel random effect model for accuracy
        model_error <-  metafor::rma.mv(yi, vi,
                                        random = ~ 1 | unique_sample_id / observation_id,
                                        data=error_effect)


        # robust standard errors clustered at the sample level
        robust_model_error <- robust(model_error,
                                     cluster = error_effect$unique_sample_id)

        # Step 4: return estimate of interest
        return(tidy(robust_model_error) %>%
                 mutate(imputed_correlation = x,
                        effect = "error") %>%
                 mutate_if(is.numeric, round, 5)
        )
      }
      )
  }

}
```

The second one, `robustness_independence`, calculates returns the effect assuming independence between fake and true news measures.

```r
# a function that returns the main average effect assuming *independence*
robustness_independence <- function(effect = "accuracy") {

  if (effect == "accuracy") {
    # Calculate effect assuming *independence*
    independence_effect <- escalc(measure="SMD",
                                  # diff = true (m1i) - fake (m2i)
                                  m2i= mean_accuracy_fake,
                                  sd2i=sd_accuracy_fake, n2i=n_observations,
                                  m1i=mean_accuracy_true, sd1i=sd_accuracy_true,
                                  n1i=n_observations,
                                  data = meta_wide)
    # Multilevel random effect model
    model_independence <-  metafor::rma.mv(yi, vi,
                                           random = ~ 1 | unique_sample_id / observation_id,
                                           data=independence_effect)


    # robust standard errors clustered at the sample level
    robust_model_independence <- robust(model_independence,
                                        cluster = independence_effect$unique_sample_id)

    # Step 4: return estimate of interest
    return(tidy(robust_model_independence) %>%
             mutate_if(is.numeric, round, 5))

  }

  if (effect == "error") {
    # Calculate effect assuming *independence*
    independence_effect <- escalc(measure="SMD",
                                  # diff = true (m1i) - fake (m2i)
                                  m2i= error_fake,
                                  sd2i=sd_accuracy_fake, n2i=n_observations,
                                  m1i=error_true, sd1i=sd_accuracy_true,
                                  n1i=n_observations,
                                  data = meta_wide)
    # Multilevel random effect model
    model_independence <-  metafor::rma.mv(yi, vi,
                                           random = ~ 1 | unique_sample_id / observation_id,
                                           data=independence_effect)


    # robust standard errors clustered at the sample level
    robust_model_independence <- robust(model_independence,
                                        cluster = independence_effect$unique_sample_id)

    # Step 4: return estimate of interest
    return(tidy(robust_model_independence) %>%
             mutate_if(is.numeric, round, 5))

  }
```

```
}
```

We generate the different estimates using the above functions. We compare those estimates to our main estimate from imputing the average correlation. We do that descriptively, by plotting.

```
# write a function that returns plots to compare model results from different imputations of correlatio
# (and a version assuming independence)
robustness_plot <- function(effect){


  # calculate effect as function of correlation value for error
  effect_by_correlation <- robustness_correlation_values(correlations, effect = effect)

  # calculate effect assuming independence for error
  effect_independence <- robustness_independence(effect = effect)

  # get data and main model results from environment
  main_model_name <- paste0("robust_model_", effect)
  data_name <- paste0(effect, "_effect")

  main_model <- get(main_model_name)
  data <- get(data_name)

  # get tidy version of main model (to be used as reference)
  main_model <-  tidy(main_model) %>%
    mutate_if(is.numeric, round, 5) %>%
    # add the value of the correlation
    mutate(imputed_correlation = mean(data$cor))


# plot distribution of standard error
SE_plot <- ggplot(effect_by_correlation,
                  aes(x = imputed_correlation, y = std.error,
                      fill = "a", color = "a")) +
geom_point(alpha = 0.6) +
  # add point with average correlation effect
  geom_point(data = main_model, aes(x = imputed_correlation, y = std.error),
             color = "red", fill = "red")  +
  geom_text(data = main_model, aes(x = imputed_correlation, y = 0.9*std.error,
                                   label =  paste0("SE assuming \n avearage correlation (", round(
                                     imputed_correlation, digits = 2), ")", "\n",
                                     "= ", round(std.error, digits = 4))),
            color = 'red', nudge_x = 0.08,
            # since we didn't pre-compute means, ggplot would print
            # mean(distance_by_walk) for each observation which makes the text
            # super bold - so we tell it to check for overlap and remove it
            check_overlap = T
            ) +
  # add h_line with effect assuming independence
  geom_hline(data = effect_independence, aes(yintercept = std.error),
             linetype='dotted',
             color = 'darkorange') +
  geom_text(data = effect_independence, aes(y = 0.9*std.error, x = 0,
```

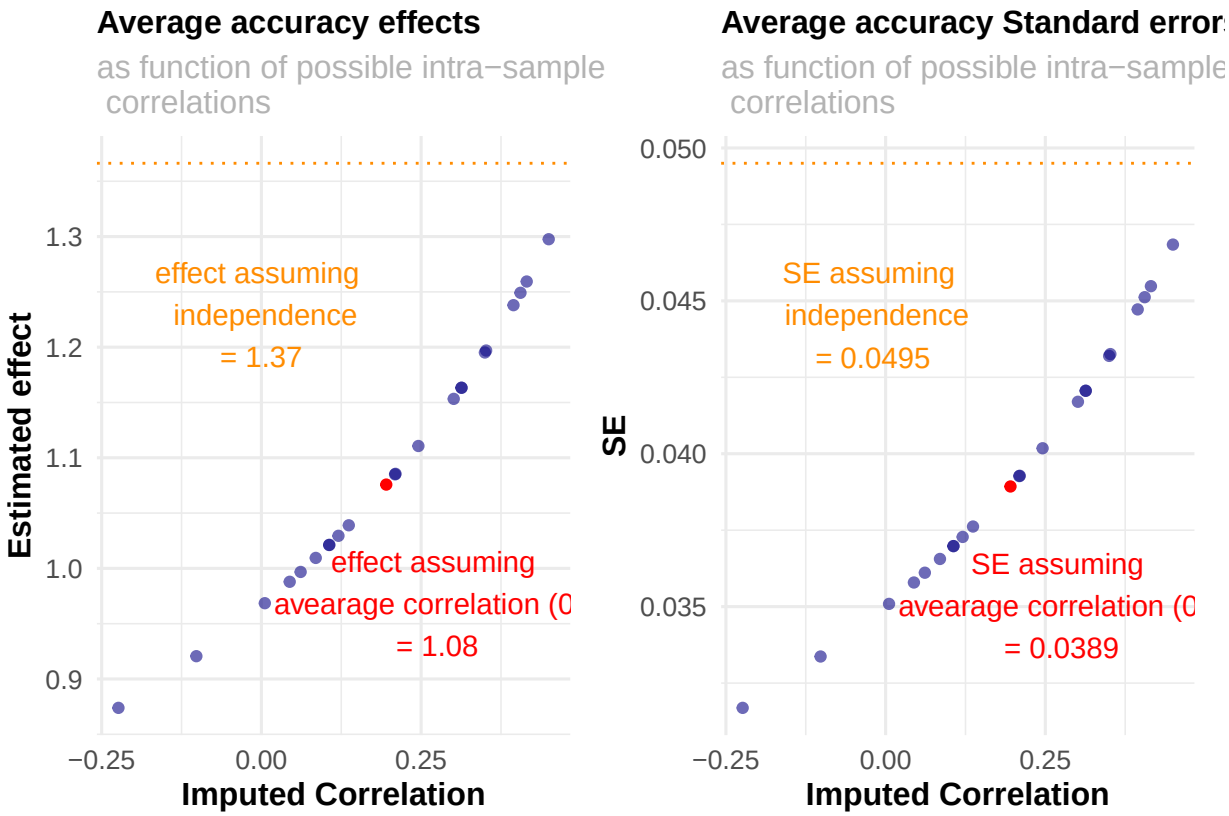
```
                                                label =  paste0("SE assuming \n independence",
                                                                "\n", "= ",
                                                                round(std.error, digits = 4))),
            color = 'darkorange', nudge_x = -0.02,
            check_overlap = T
  ) +
  # colors
  scale_color_viridis_d(option = "plasma") +
  scale_fill_viridis_d(option = "plasma") +
  # labels and scales
  guides(color = "none", fill = "none") +
    labs(x = "Imputed Correlation", y = "SE",
       title = paste0("Average ", effect, " Standard errors"),
       subtitle = "as function of possible intra-sample \n correlations") +
  plot_theme

# scatter plot effect by correlation
effect_plot <- ggplot(effect_by_correlation,
                      aes(x = imputed_correlation, y = estimate,
                                                fill = "a", color = "a")) +
  geom_point(alpha = 0.6) +
  # add point with average correlation effect
  geom_point(data = main_model, aes(x = imputed_correlation, y = estimate),
             color = "red", fill = "red")  +
  geom_text(data = main_model, aes(x = imputed_correlation, y = 0.9*estimate,
                                   label =  paste0("effect assuming \n avearage correlation (", round
                                     imputed_correlation, digits = 2), ")", "\n",
                                     "= ", round(estimate, digits = 2))),
            color = 'red', nudge_x = 0.08,
            # since we didn't pre-compute means, ggplot would print
            # mean(distance_by_walk) for each observation which makes the text
            # super bold - so we tell it to check for overlap and remove it
            check_overlap = T
            ) +
  # add h_line with effect assuming independence
  geom_hline(data = effect_independence, aes(yintercept = estimate),
             linetype='dotted',
             color = 'darkorange') +
  geom_text(data = effect_independence, aes(y = 0.9*estimate, x = 0,
                                                label =  paste0("effect assuming \n independence",
                                                                "\n", "= ",
                                                                round(estimate, digits = 2))),
            color = 'darkorange',
            check_overlap = T
  ) +
  # colors
  scale_color_viridis_d(option = "plasma") +
  scale_fill_viridis_d(option = "plasma") +
  # labels and scales
  guides(color = "none", fill = "none") +
    labs(x = "Imputed Correlation", y = "Estimated effect",
       title = paste0("Average ", effect, " effects"),
       subtitle = "as function of possible intra-sample \n correlations") +
```

```
  plot_theme

return(list(effect_plot, SE_plot))
}
```

```
# make plots for accuracy
accuracy_plots <- robustness_plot("accuracy")
```

```
## [1] "Currently estimating model for corrrelation value -0.101790176305029"
## [1] "Currently estimating model for corrrelation value 0.351769048012383"
## [1] "Currently estimating model for corrrelation value 0.449710681299835"
## [1] "Currently estimating model for corrrelation value 0.41525641951147"
## [1] "Currently estimating model for corrrelation value 0.349815229643699"
## [1] "Currently estimating model for corrrelation value 0.245731931964752"
## [1] "Currently estimating model for corrrelation value 0.313188575913319"
## [1] "Currently estimating model for corrrelation value 0.0443020561106972"
## [1] "Currently estimating model for corrrelation value 0.0053581399301534"
## [1] "Currently estimating model for corrrelation value 0.394562668160262"
## [1] "Currently estimating model for corrrelation value -0.223727217130069"
## [1] "Currently estimating model for corrrelation value 0.0614167007089909"
## [1] "Currently estimating model for corrrelation value 0.1205795029782"
## [1] "Currently estimating model for corrrelation value 0.136791113188052"
## [1] "Currently estimating model for corrrelation value 0.405510561632877"
## [1] "Currently estimating model for corrrelation value 0.301029705603367"
## [1] "Currently estimating model for corrrelation value 0.313253547469207"
## [1] "Currently estimating model for corrrelation value 0.106066326665599"
## [1] "Currently estimating model for corrrelation value 0.209938621992656"
## [1] "Currently estimating model for corrrelation value 0.106287557564847"
## [1] "Currently estimating model for corrrelation value 0.0850875334075016"
## [1] "Currently estimating model for corrrelation value 0.209217063993221"
```

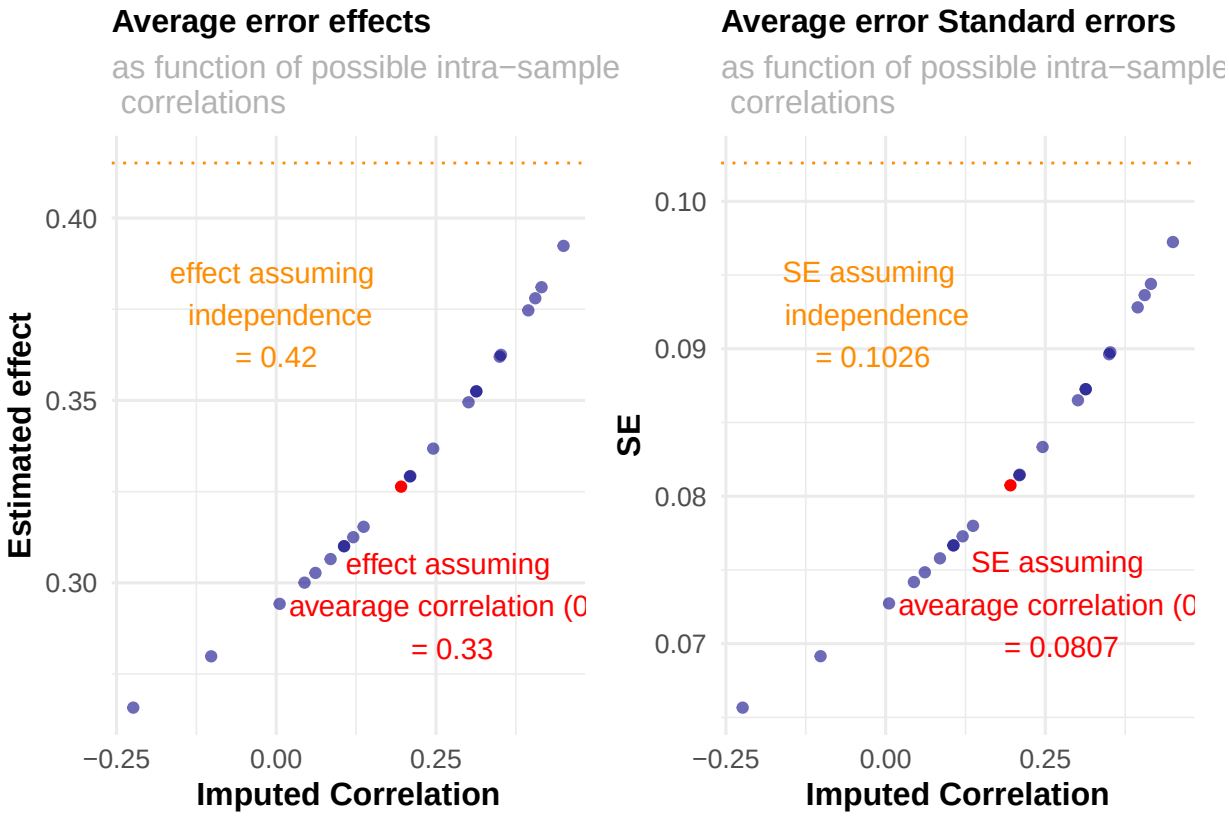
```
accuracy_plots[[1]] + accuracy_plots[[2]]
```




```
# make plots for error
error_plots <- robustness_plot("error")
```

```
## [1] "Currently estimating model for corrrelation value -0.101790176305029"
## [1] "Currently estimating model for corrrelation value 0.351769048012383"
## [1] "Currently estimating model for corrrelation value 0.449710681299835"
## [1] "Currently estimating model for corrrelation value 0.41525641951147"
## [1] "Currently estimating model for corrrelation value 0.349815229643699"
## [1] "Currently estimating model for corrrelation value 0.245731931964752"
## [1] "Currently estimating model for corrrelation value 0.313188575913319"
## [1] "Currently estimating model for corrrelation value 0.0443020561106972"
## [1] "Currently estimating model for corrrelation value 0.0053581399301534"
## [1] "Currently estimating model for corrrelation value 0.394562668160262"
## [1] "Currently estimating model for corrrelation value -0.223727217130069"
## [1] "Currently estimating model for corrrelation value 0.0614167007089909"
## [1] "Currently estimating model for corrrelation value 0.1205795029782"
## [1] "Currently estimating model for corrrelation value 0.136791113188052"
## [1] "Currently estimating model for corrrelation value 0.405510561632877"
## [1] "Currently estimating model for corrrelation value 0.301029705603367"
## [1] "Currently estimating model for corrrelation value 0.313253547469207"
## [1] "Currently estimating model for corrrelation value 0.106066326665599"
## [1] "Currently estimating model for corrrelation value 0.209938621992656"
## [1] "Currently estimating model for corrrelation value 0.106287557564847"
## [1] "Currently estimating model for corrrelation value 0.0850875334075016"
## [1] "Currently estimating model for corrrelation value 0.209217063993221"
```

```
error_plots[[1]] + error_plots[[2]]
```



Compare to subset of (non-binary) individual-level data

We want to compare our main meta model to two other models:

- a reduced meta model with only those (non-binary) studies that we also have individual-level data on
- a mixed model on the (non-binary) individual-level data with standardized accuracy measures

To be able to compare the effect size from the linear model to the meta model, we need to *standardize* our accuracy measures first. Our standardization process is:

- within each sample, calculate the standard deviation of accuracy ratings (fake and true news combined)
- for each sample, divide accuracy ratings by the respective standard deviation

```
# load data
individual_level_subset <- read_csv("individual_level_subset.csv")

# compute standardized accuracy and error measures
individual_level_subset <- individual_level_subset %>%
  # remove binary scales and treatment conditions
  filter(scale != "binary" & condition == "control") %>%
  # remove NA's for accuracy
```

```
  drop_na(accuracy) %>%
  # identify unique samples
  mutate(unique_sample_id = paste(paper_id, sample_id, sep = "_"),
         # make an ungrouped versions just for comparison
         ungrouped_std_acc = accuracy/sd(accuracy),
         ungrouped_std_err = error/sd(accuracy)
         ) %>%
  # group by unique samples
  group_by(unique_sample_id) %>%
  # calculate standardized accuracy measure
  mutate(accuracy_std = accuracy/sd(accuracy),
         error_std = error/sd(accuracy)) %>%
  # ungroup
  ungroup()
```

Next, we reduce our meta data to only those stuies that we have raw data on.

```
# These steps are to identify those studies that we have raw data on in the
# meta data and subset that data accordingly.

# Step 1: extract `unique_sample_id` for samples in raw data
raw_data_samples <- individual_level_subset %>%
  summarize(unique(unique_sample_id)) %>%
  pull()
```

```
## Warning: Returning more (or less) than 1 row per ‘summarise()‘ group was deprecated in
## dplyr 1.1.0.
## i Please use ‘reframe()‘ instead.
## i When switching from ‘summarise()‘ to ‘reframe()‘, remember that ‘reframe()‘
##   always returns an ungrouped data frame and adjust accordingly.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
# Note that in our simulated data, paper_ids are just numbers
# We thus have to include some intermediate steps that we would not include
# in the actual analysis here.

# Intermediate steps:
# In the following steps, we make sure to have three matching cases between our
# raw data and our simulated meta data.

# a) check which id's correspond to binary studies in meta data
continuous_samples <- accuracy_effect %>% filter(accuracy_scale != "binary") %>%
  summarize(unique(unique_sample_id)) %>% pull()
```

```
## Warning: Returning more (or less) than 1 row per ‘summarise()‘ group was deprecated in
## dplyr 1.1.0.
## i Please use ‘reframe()‘ instead.
## i When switching from ‘summarise()‘ to ‘reframe()‘, remember that ‘reframe()‘
##   always returns an ungrouped data frame and adjust accordingly.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```r
# b) rename the first three of these id's with names that fit the raw data
rename_ids <- function(data) {
  results <- data %>%
    mutate(unique_sample_id = case_when(
      unique_sample_id ==  all_of(continuous_samples)[1] ~ all_of(raw_data_samples)[1],
      unique_sample_id == all_of(continuous_samples)[2] ~ all_of(raw_data_samples)[2],
      unique_sample_id == all_of(continuous_samples)[3] ~ all_of(raw_data_samples)[3],
      TRUE ~ as.character(unique_sample_id)
    )
    )
}

accuracy_effect <- rename_ids(accuracy_effect)
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `unique_sample_id = case_when(...)`.
## Caused by warning:
## ! Using `all_of()` outside of a selecting function was deprecated in tidyselect
##   1.2.0.
## i See details at
##   <https://tidyselect.r-lib.org/reference/faq-selection-context.html>
```

```r
error_effect <- rename_ids(error_effect)

# Step 2: filter the meta data frame
# We want to reduce our data to continuous measure samples that we have raw data on
reduce_samples <- function(data) {

    results <- data %>%
      filter(unique_sample_id %in% all_of(raw_data_samples))
}
# for accuracy
reduced_accuracy_effect <- reduce_samples(accuracy_effect)
# for error
reduced_error_effect <- reduce_samples(error_effect)
```

```r
# run linear mixed model with random slope and intercept for subject id nested
# in sample id
# accuracy
individual_level_accuracy <- lmer(accuracy_std ~ 1 + veracity +
                                    (1 + veracity | unique_sample_id / id),
                data = individual_level_subset) %>%
  tidy(conf.int = TRUE) %>%
  mutate(model = "individual level",
         outcome = "accuracy") %>%
  # select only effect of interest
  filter(term == "veracitytrue")

# error
individual_level_error <- lmer(error_std ~ 1 + veracity +
                                    (1 + veracity | unique_sample_id / id),
                data = individual_level_subset) %>%
  tidy(conf.int = TRUE) %>%
```

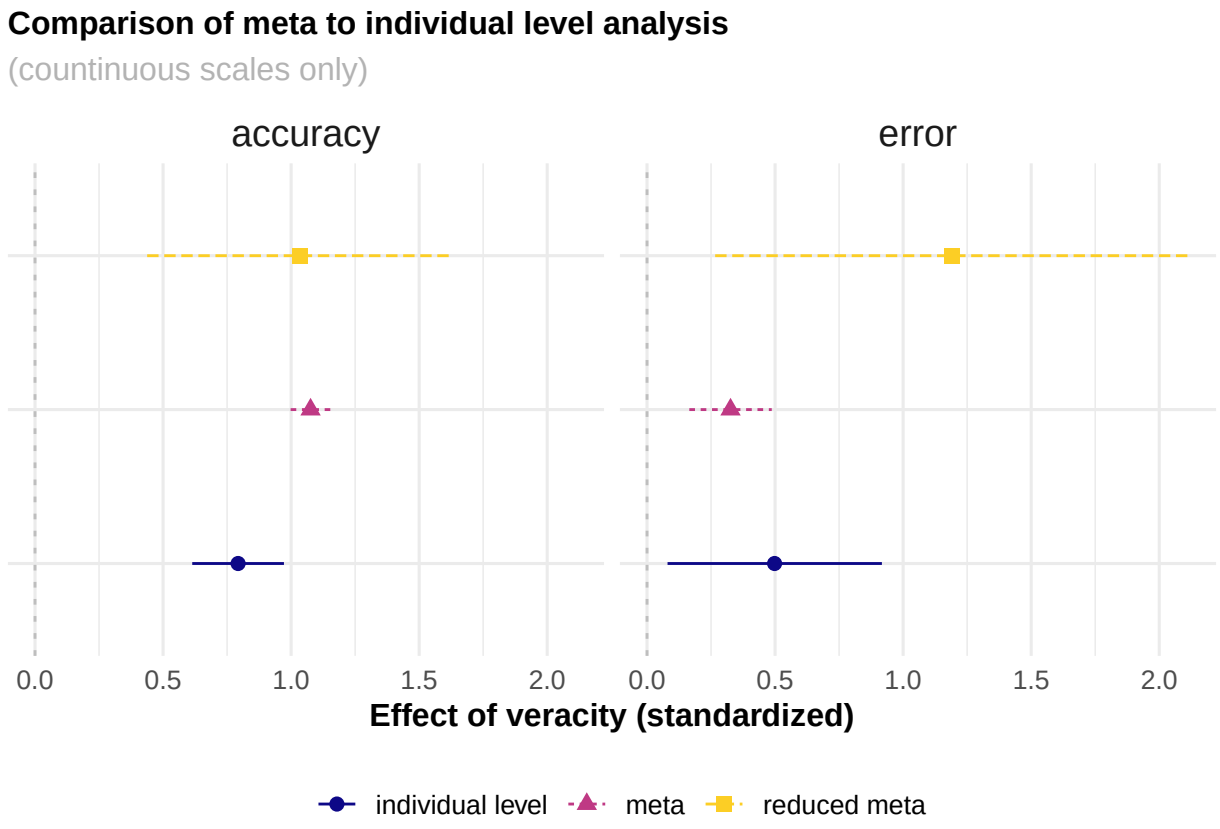
```
  mutate(model = "individual level",
         outcome = "error") %>%
  # select only effect of interest
  filter(term == "veracitytrue")
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.00698083 (tol = 0.002, component 1)
```

```
# We then run the meta model on reduced data
reduced_accuracy <- robust(metafor::rma.mv(yi, vi,
                                   random = ~ 1 | unique_sample_id /
                                     observation_id, data = reduced_accuracy_effect ),
                    cluster = reduced_accuracy_effect$unique_sample_id
) %>%
  tidy(conf.int = TRUE) %>%
  mutate(model = "reduced meta",
         outcome = "accuracy")

reduced_error <- robust(metafor::rma.mv(yi, vi,
                                   random = ~ 1 | unique_sample_id /
                                     observation_id, data = reduced_error_effect ),
                    cluster = reduced_error_effect$unique_sample_id
) %>%
  tidy(conf.int = TRUE) %>%
  mutate(model = "reduced meta",
         outcome = "error")

# We store the results of all analyses in a single data frame
comparison <- bind_rows(
  # models individual level data
  individual_level_accuracy,
  individual_level_error,
  # models on reduced data
  reduced_accuracy,
  reduced_error,
  # main models
  robust_model_accuracy %>%
    tidy(conf.int = TRUE) %>%
    mutate(model = "meta",
           outcome = "accuracy"),
  robust_model_error %>%
    tidy(conf.int = TRUE) %>%
    mutate(model = "meta",
           outcome = "error")) %>%
  # give a nicer name to the estimate
  mutate(term = "veracity (baseline: false")

# plot results
ggplot(comparison  %>%
         # round values to fewer digits
         mutate_if(is.numeric, round, 3),
       aes(x = estimate, y = model, shape = model,
           linetype = model)
```

```
      ) +
  geom_vline(xintercept = 0,
             linewidth = 0.5, linetype = "24", color = "grey") +
  geom_pointrange(aes(xmin = conf.low, xmax = conf.high, color = model),
                  position = position_dodge(width = -0.6)) +
  labs(title = "Comparison of meta to individual level analysis",
       subtitle = "(countinuous scales only)",
       x = "Effect of veracity (standardized)", y = NULL, linetype = NULL,
       shape = NULL, color=NULL) +
  # colors
  scale_color_viridis_d(option = "plasma", end = 0.9) +
  plot_theme +
    theme(legend.position = "bottom",
          axis.text.y=element_blank(),
          strip.text = element_text(size = 14)) +
  facet_wrap(~outcome)
```

```
## Warning: `position_dodge()` requires non-overlapping x intervals
## `position_dodge()` requires non-overlapping x intervals
```



Binary vs. continuous outcomes

Some of the studies included in our review measure perceived accuracy on a continuous scale, others on a binary (or dichotomous) scale.

This is not problematic per se - there are statistical methods to to be able to compare effects on both scales (Higgins et al. 2019). This requires, however, appropriate summary statistics for both scales. For continuous measures, that would be means and standard deviations, for binary measures those would be odds or risk ratios, for example.

The problem we face is that authors do not provide these appropriate summary statistics for binary scales. Instead, they report means and standard deviations, just as they do for continuous outcomes.

Nevertheless, we decide to include studies with binary response scales in the main analysis. We will include a series of robustness checks to see how this decision affects our results.

```
# Start with making a new variable distinguishing between binary and
# continuous scales.
# This step is only necessary with our simulated data.

add_binary_continuous_scales <- function(data) {

  results <- data %>%
    mutate(scale_binary_continuous = ifelse(accuracy_scale == "binary", "binary",
                                            "continuous")
    )
}

accuracy_effect <- add_binary_continuous_scales(accuracy_effect)
error_effect <- add_binary_continuous_scales(error_effect)
```

1. Compare non-binary effect sizes to binary effect sizes

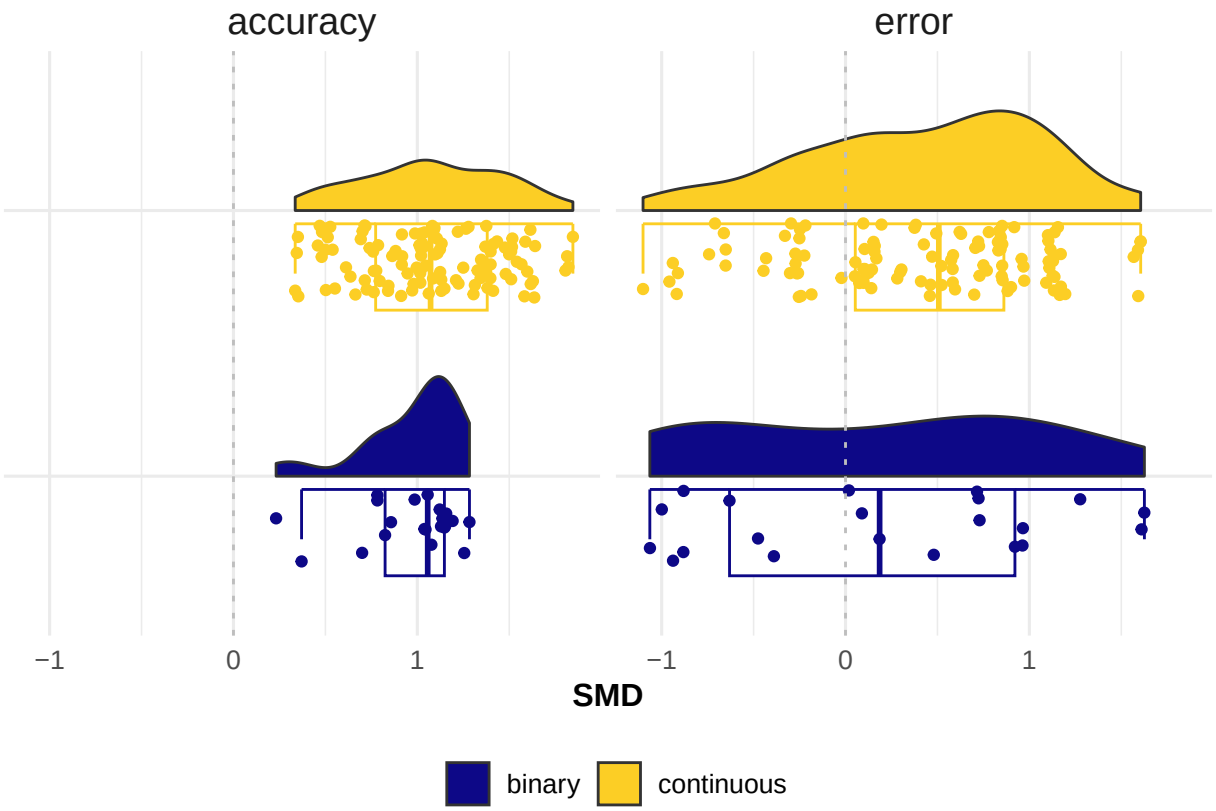
Visual.

```
data <- bind_rows(accuracy_effect %>% mutate(outcome = "accuracy"),
                  error_effect %>% mutate(outcome = "error"))

ggplot(data,
       aes(x = scale_binary_continuous, y = yi)) +
    geom_half_boxplot(aes(color = scale_binary_continuous), side = "l", size = 0.5, nudge = 0.05,
                      outlier.shape = NA) +
    geom_half_violin(aes(fill = scale_binary_continuous), side = "r") +
    geom_half_point(aes(color = scale_binary_continuous), side = "l",
                    transformation_params = list(height = 0, width = 0.1, seed = 1)) +
    # add line of 0
    geom_hline(yintercept = 0,
               linewidth = 0.5, linetype = "24", color = "grey") +
    # colors
    scale_color_viridis_d(option = "plasma", end = 0.9)+
    scale_fill_viridis_d(option = "plasma", end = 0.9) +
    # labels and scales
    labs(x = NULL, y = "SMD", fill = NULL) +
  guides(color = FALSE) +
    plot_theme +
    coord_flip() +
  plot_theme +
    theme(legend.position = "bottom",
          axis.text.y=element_blank(),
```

```
          strip.text = element_text(size = 14)) +
    facet_wrap(~outcome)
```

```
## Warning in geom_half_point(aes(color = scale_binary_continuous), side = "l", :
## Ignoring unknown parameters: ‘transformation_params‘
```

```
## Warning: The ‘<scale>‘ argument of ‘guides()‘ cannot be ‘FALSE‘. Use "none" instead as
## of ggplot2 3.3.4.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



Calculating a meta-regression.

```
# function that calculates a meta regression and takes a data frame
# as input
scale_comparison_meta_regression <- function (data) {

  results <- robust(metafor::rma.mv(yi, vi,
                                    mods = ~scale_binary_continuous,
                                    random = ~ 1 | unique_sample_id /
                                      observation_id, data = data),
                    cluster = data$unique_sample_id
  ) %>%
    tidy(conf.int = TRUE) %>%
```

```
    mutate(model = "Original SMDs (all data)") %>%
    # give a nicer name to the estimate
    mutate(term = ifelse(term == "scale_binary_continuouscontinuous",
                  "effect of continuous scale (baseline binary)", term)
    )

  return(results)
}
```

```
# calculate the model for accuracy
scales_comparison_all_data_accuracy <- scale_comparison_meta_regression(accuracy_effect)
# for error
scales_comparison_all_data_error <- scale_comparison_meta_regression(error_effect)
```

The problem with this first robustness check is that if we find a difference, we do not know how much of it is a genuine effect of binary vs. continuous scales on participants accuracy ratings, and how much of it is due to the fact of comparing them based on biasing summary statistics. We will qualitatively explore this issue in several follow-up steps:

2. Compare to recommended procedure in Higgins et al. (2019)

a) Among the binary studies, we retain only those that we have raw data. We will later on run the same model as in robustness check 1.

```
# Step 1: extract `unique_sample_id` for samples with binary outcomes that
# raw data is available for

# load data (again, since we previously did some modifications)
individual_level_subset <- read_csv("individual_level_subset.csv")

binary_samples_raw_data <- individual_level_subset %>%
  filter(scale == "binary") %>%
  # (note: do not pick `paper_id` because within a paper, some samples might have
  # been measured on binary, others on a continuous scale)
  summarize(unique(unique_sample_id)) %>%
  pull()
```

```
## Warning: Returning more (or less) than 1 row per ‘summarise()‘ group was deprecated in
## dplyr 1.1.0.
## i Please use ‘reframe()‘ instead.
## i When switching from ‘summarise()‘ to ‘reframe()‘, remember that ‘reframe()‘
##   always returns an ungrouped data frame and adjust accordingly.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
# Note that in our simulated data, paper_ids are just numbers
# We thus have to include some intermediate steps that we would not include
# in the actual analysis here.

# Intermediate steps:
# In the following steps, we make sure to have three matching cases between our
```

```r
# raw data and our simulated meta data.

# a) check which id's correspond to binary studies in meta data
binary_samples <- accuracy_effect %>% filter(accuracy_scale == "binary") %>%
  summarize(unique(unique_sample_id)) %>% pull()
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## i Please use `reframe()` instead.
## i When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```r
# b) rename the first three of these id's with names that fit the raw data
rename_ids <- function(data) {
  results <- data %>%
    mutate(unique_sample_id = case_when(
      unique_sample_id ==  all_of(binary_samples)[1] ~ all_of(binary_samples_raw_data)[1],
      unique_sample_id == all_of(binary_samples)[2] ~ all_of(binary_samples_raw_data)[2],
      unique_sample_id == all_of(binary_samples)[3] ~ all_of(binary_samples_raw_data)[3],
      TRUE ~ as.character(unique_sample_id)
    )
    )
}

accuracy_effect <- rename_ids(accuracy_effect)
error_effect <- rename_ids(error_effect)

# Step 2: filter the meta data frame
# We want to reduce our data to continuous measure samples AND only those binary
# measure samples that we have raw data on

reduce_binary_samples <- function(data) {

    results <- data %>%
      filter(accuracy_scale != "binary" | unique_sample_id %in% all_of(binary_samples_raw_data))
}
# for accuracy
reduced_accuracy_effect <- reduce_binary_samples(accuracy_effect)
# for error
reduced_error_effect <- reduce_binary_samples(error_effect)
```

b) For this set of binary studies, we then calculate the odds ratios from the raw data.

A general overview of summary statistics for binary outcomes can be found here (Higgins et al. (2019)).

Here, we will calculate the odds ratios, for accuracy and error, respectively.

Odds. The 'odds' refer to the ratio of the probability that a particular event will occur to the probability that it will not occur, and can be any number between zero and infinity (Higgins et al. (2019)). It is commonly expressed as a ratio of two integers. For example, in a clinical context, 1 out of 100 patients might die; then the odds of dying are `0.01`, or `1:100`.

Table 3: Binary outcome possibilities

	Accuracy ratings	
	Rated as accurate	**Rated as not accurate**
True	Sum of responses rating true news as 'accurate' (Accurate_true)	Sum of responses rating true news as 'Not accu
Fake	Sum of responses rating fake news as accurate' (Accurate_fake)	Sum of responses rating fake news as 'NOT ac

In our case, the odds are `Accurate:NotAccurate`. For example, if 10 participants made ratings on 10 news each, we'd have 100 ratings. Of these ratings, 60 said 'accurate', 40 said 'not accurate', then the odds are `60:40` or `1.5`.

The odds **ratio** (OR) is the ratio of the Odds of `accuracy` ratings between `true` and `fake` news. It is calculated as

$$OR_{Accuracy} = \frac{(Accurate_{true}/NotAccurate_{true})}{(Accurate_{fake}/NotAccurate_{fake})}$$

Interpretation. If the OR is `1`, participants were just as likely to rate items as 'accurate' when looking at `true` news as they were when looking at `fake` news. If the OR is `> 1`, then participants rated true news as more accurate than fake news. An OR of `2` means that participants were twice as likely to rate true news as accurate compared to fake news.

For binary variables, `error = 1 - accuracy` for `true` news, and `error = accuracy` for `fake_news`.

The odds ratio (OR) for `error` is thus calculated as

$$OR_{Error} = \frac{(NotAccurate_{true}/Accurate_{true})}{(Accurate_{fake}/NotAccurate_{fake})} = \frac{\frac{1}{(NotAccurate_{true}/Accurate_{true})}}{(Accurate_{fake}/NotAccurate_{fake})} = \frac{1}{OR_{Accuracy}} \quad (1)$$

To be able to calculate `OR_Accuracy` and `OR_Error`, we need to count "not accurate", i.e. *correct* responses for `fake` news. Taking our `accuracy` variable, we can calculate a `NO_accuracy` variable as `1-accuracy`.

Below, we calculate the odds ratio (OR) for both accuracy and error. We also express the OR on a logarithmic scale, also referred to as "log odds ratio"(logOR), by additionally calculating `log(OR)`.

$$log(OR) = log_e(OR)$$

If the log odds ratio is positive, it indicates that the odds of `accuracy` are higher for `true` news compared to `fake` news. To interpret the magnitude of that difference we have to look at the odds ratio. The reason we use the log odds ratios is that it allows to model odds ratios as a linear function in regression models. This property helps us transforming ORs to SMDs later.

```
# Calculate Odds ratios for error and accuracy

# Demonstration
individual_level_subset %>%
  filter(scale == "binary") %>%
  group_by(unique_sample_id, veracity) %>%
  summarize(
    mean_error = mean(error, na.rm = TRUE),
    mean_accuracy = mean(accuracy, na.rm = TRUE),
    sum_accuracy = sum(accuracy),
    sum_NO_accuracy = sum(1-accuracy)
```

```
  ) %>%
  pivot_wider(names_from = veracity,
              values_from = c(mean_error, mean_accuracy,
                              sum_accuracy, sum_NO_accuracy)) %>%
  group_by(unique_sample_id, mean_accuracy_true, mean_accuracy_fake,
           mean_error_true, mean_error_fake) %>%
  summarise(
    # for accuracy
    OR_accuracy = (sum_accuracy_true/sum_NO_accuracy_true) /
      (sum_accuracy_fake/sum_NO_accuracy_fake),
    # for error
    OR_error = (sum_NO_accuracy_true/sum_accuracy_true) /
      (sum_accuracy_fake/sum_NO_accuracy_fake)
  ) %>%
  mutate(across(starts_with("OR"), ~log(.), .names = "{.col}_log"))
```

```
## `summarise()` has grouped output by 'unique_sample_id'. You can override using
## the `.groups` argument.
## `summarise()` has grouped output by 'unique_sample_id', 'mean_accuracy_true',
## 'mean_accuracy_fake', 'mean_error_true'. You can override using the `.groups`
## argument.
```

```
## # A tibble: 7 x 9
## # Groups:   unique_sample_id, mean_accuracy_true, mean_accuracy_fake,
## #   mean_error_true [7]
##   unique_sample_id mean_accuracy_true mean_accuracy_fake mean_error_true
##   <chr>                         <dbl>              <dbl>           <dbl>
## 1 Bago_2020_1                   0.746              0.307           0.254
## 2 Bago_2020_2                   0.654              0.277           0.346
## 3 Bago_2022_1                   0.677              0.342           0.323
## 4 Bago_2022_2                   0.674              0.286           0.326
## 5 Bago_2022_3                   0.666              0.265           0.334
## 6 Bago_2022_4                   0.756              0.167           0.244
## 7 Sultan_2022_1                 0.665              0.248           0.335
## # i 5 more variables: mean_error_fake <dbl>, OR_accuracy <dbl>, OR_error <dbl>,
## #   OR_accuracy_log <dbl>, OR_error_log <dbl>
```

We will do the same thing we did above 'by hand' using the the `metafor` package and its `escalc()` function in R that we used before (Viechtbauer 2010). The advantage of doing so is that `escalc()` will automatically calculate variances, too.

```
# make a data frame that `escalc` can use to calculate logOR
odds_ratios <- individual_level_subset %>%
  filter(scale == "binary") %>%
  group_by(unique_sample_id, veracity) %>%
  summarize(
    sum_accuracy = sum(accuracy),
    sum_NO_accuracy = sum(1-accuracy)
    ) %>%
  pivot_wider(names_from = veracity, values_from = c(sum_accuracy, sum_NO_accuracy))
```

```
## `summarise()` has grouped output by 'unique_sample_id'. You can override using
## the `.groups` argument.
```

```r
# calculate logOR using `escalc` for accuracy
odds_ratios_accuracy <- escalc(measure="OR",
       # true / fake
       ai= sum_accuracy_true,
       bi=sum_NO_accuracy_true,
       ci=sum_accuracy_fake,
       di=sum_NO_accuracy_fake,
       data = odds_ratios )

# calculate logOR using `escalc` for error
odds_ratios_error <- escalc(measure="OR",
       # true / fake
       bi= sum_accuracy_true,
       ai=sum_NO_accuracy_true,
       ci=sum_accuracy_fake,
       di=sum_NO_accuracy_fake,
       data = odds_ratios )
```

c) We then transform these odds ratios (ORs) into standardized mean differences (SMDs) (this requires some assumptions about underlying distributions of binary outcomes, see Higgins et al. (2019)). Here is a direct link to the relevant chapter online.

The transformation formula we use is originally from Chinn (2000).

$$SMD = \frac{\sqrt{3}}{\pi} ln(OR)$$

The standard error of the log odds ratio can be converted to the standard error (SE) of a SMD by multiplying by the same constant ($\frac{\sqrt{3}}{\pi} = 0.5513$) (Higgins et al. (2019)).

$$SE_{SMD} = SE_{ln(OR)} \cdot \frac{\sqrt{3}}{\pi}$$

Since the `metafor` package works with variances, we will do the same transformation on the variance (`vi`) instead of the SE.

```r
# calculate SMDs from ORs

# function that takes the data `odds_ratios...` data sets as input
calculate_SMD_from_OR <- function(data) {

  data %>%
    mutate(SMD_from_OR = sqrt(3) / pi * yi,
           Var_from_OR = sqrt(3) / pi * vi) %>%
    select(unique_sample_id, SMD_from_OR, Var_from_OR)
}


# for accuracy
SMD_from_OR_accuracy <- calculate_SMD_from_OR(odds_ratios_accuracy)
# for error
SMD_from_OR_error <- calculate_SMD_from_OR(odds_ratios_error)
```

d) We, again, run the the same meta-regression as in 2a), but using the SMDs obtained from the ORs now. We compare the estimates of the model using SMDs based on means/SDs (2a) and the model using SMds based on ORs.

First, we re-integrate the SMDs we obtained from the ORs into the original `reduced_{accuracy/error}_effect` data frame.

```
# function to merge data with original SMD and the new SMDs from ORs
merge_SMD_SMD_from_OR <- function (data_SMD, data_SMD_from_OR) {

  left_join(data_SMD, data_SMD_from_OR, by="unique_sample_id") %>%
    # rename yi (the previous SMD) to SMD and vi (variance) to Var
    rename(SMD = yi,
           Var = vi
    ) %>%
    mutate(
      # `SMD_from_OR_accuracy for now takes NA fro non-binary scales
      # make it take the value of SMD for all non-binary scales
      SMD_from_OR = ifelse(accuracy_scale == "binary", SMD_from_OR, SMD),
      # same for variance
      Var_from_OR = ifelse(accuracy_scale == "binary", Var_from_OR, Var)
    ) %>%
    select(unique_sample_id, observation_id, scale_binary_continuous, starts_with(c("SMD", "Var")))

}
# apply function
accuracy_SMD_OR <- merge_SMD_SMD_from_OR(reduced_accuracy_effect, SMD_from_OR_accuracy)
error_SMD_OR <- merge_SMD_SMD_from_OR(reduced_error_effect, SMD_from_OR_error)
```

```
# We then run the same analysis on the SMDs (same analysis as before) and on the SMDs from ORs.
# We store the results of those analyses in a single data frame

compare_results <- function(data) {

  # SMD model
  SMD_model <- robust(metafor::rma.mv(yi = SMD, V = Var,
                                      mods = ~scale_binary_continuous,
                                      random = ~ 1 | unique_sample_id /
                                        observation_id, data = data),
                      cluster = data$unique_sample_id
  ) %>%
    tidy(conf.int = TRUE) %>%
    mutate(model = "Original SMDs (reduced data)")

  # SMD imputed from OR model
  SMD_from_OR_model <- robust(metafor::rma.mv(yi = SMD_from_OR, V = Var_from_OR,
                                      mods = ~scale_binary_continuous,
                                      random = ~ 1 | unique_sample_id /
                                        observation_id, data = data),
                      cluster = data$unique_sample_id
  ) %>%
    tidy(conf.int = TRUE) %>%
    mutate(model = "SMD imputed from OR (reduced data)")
```

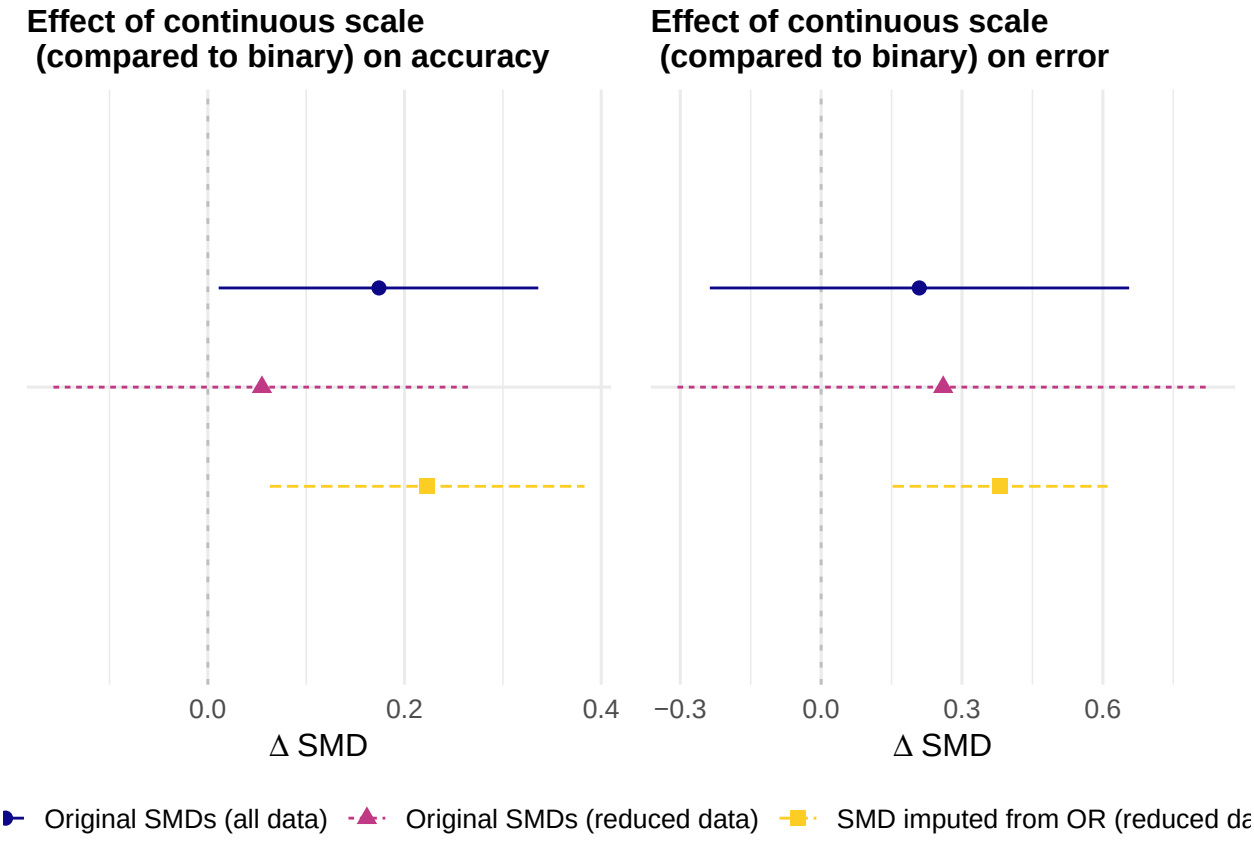
```
  # combine models
  results <- rbind(SMD_model, SMD_from_OR_model) %>%
    # give a nicer name to the estimate
    mutate(term = ifelse(term == "scale_binary_continuouscontinuous",
                  "effect of continuous scale (baseline binary)", term)
    )

  return(results)

}

# calculate models on reduced data and add full data model
# accuracy
comparison_models_accuracy <- rbind(
  compare_results(accuracy_SMD_OR),
  scales_comparison_all_data_accuracy
  )
# error
comparison_models_error <- rbind(
  compare_results(error_SMD_OR),
  scales_comparison_all_data_error)
```

Plot the differences for the reduced models.

```
# function
plot_scale_comparison <- function(data, name){

  ggplot(data  %>%
         # look only at coefficient of scale
         filter(term == "effect of continuous scale (baseline binary)") %>%
         # round values to fewer digits
         mutate_if(is.numeric, round, 3),
       aes(x = estimate, y = term, shape = model,
           linetype = model)
       ) +
  geom_vline(xintercept = 0,
             linewidth = 0.5, linetype = "24", color = "grey") +
  geom_pointrange(aes(xmin = conf.low, xmax = conf.high, color = model),
                  position = position_dodge(width = -0.6)) +
  labs(title = paste0("Effect of continuous scale \n (compared to binary) on ", name),
       x = expression(Delta~SMD), y = NULL, linetype = NULL,
       shape = NULL, color=NULL) +
  # colors
  scale_color_viridis_d(option = "plasma", end = 0.9) +
  plot_theme +
    theme(legend.position = "bottom",
          axis.text.y=element_blank())

}

accuracy <- plot_scale_comparison(comparison_models_accuracy, "accuracy")
error <- plot_scale_comparison(comparison_models_error, "error")
```

```
ggpubr::ggarrange(accuracy, error, common.legend = TRUE, legend="bottom")
```

```
## Warning: `position_dodge()` requires non-overlapping x intervals
## `position_dodge()` requires non-overlapping x intervals
## `position_dodge()` requires non-overlapping x intervals
```

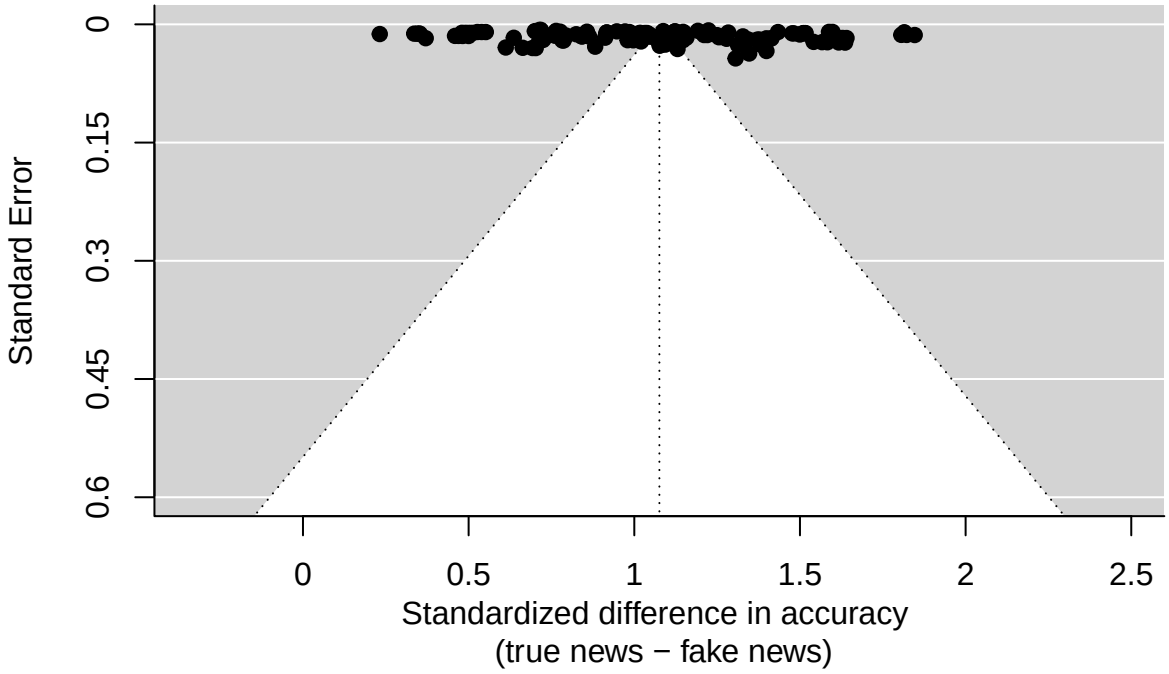


Publication bias

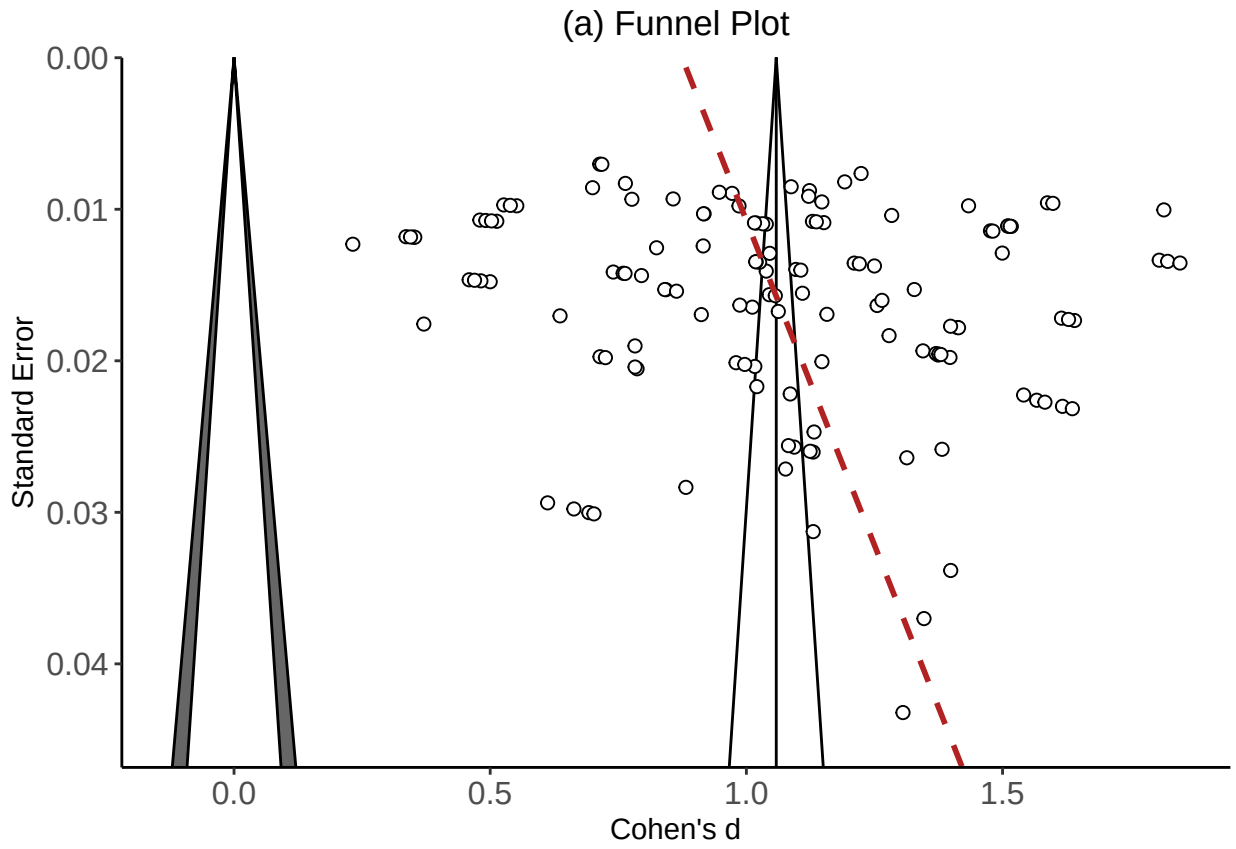
We will include some standard procedures to detect publication bias. However, we do not expect any publication bias regarding our variables of interest. That's because they are hardly ever the main outcome of most of the studies. Researchers usually test factors that alter accuracy discernment, and not the state of accuracy discernment in the control group. No studies set out to compare `error` as we define it here.

Visual

```
funnel <- funnel(model_accuracy, ylim = c(0, 0.60), xlab = "Standardized difference in accuracy \n (tru
```




```
# Create funnel plot.
funnel_raw <- viz_funnel(model_accuracy, egger = TRUE,
                         xlab = "Cohen's d", text_size = 5,
                         contours_col = "Greys")

# Convert funnel plot into ggplot object so it can be ggarranged.
funnel <- funnel_raw + ggtitle("(a) Funnel Plot") + theme_classic() +
  theme(plot.title = element_text(hjust = 0.5),
        axis.text.x = element_text(size=12),
        axis.text.y = element_text(size=12))

funnel
```



Formal

Interpreting the funnel plot just by looking at it clearly also has its limitations. There is no explicit rule when our results are "too asymmetric". To quantify asymmetry, we can run Egger's regression test (Egger et al. 1997).

```
# classical Egger test of asymmetry using linear regression
accuracy_effect %>%
  mutate(y = yi/sqrt(vi), x = 1/sqrt(vi)) %>%
  lm(y ~ x, data = .) %>%
  tidy() %>%
  mutate_if(is.numeric, round, digits = 4)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)   11.7      7.21        1.62   0.108
## 2 x              0.874    0.0947      9.24   0
```

Interpretation. The outcome in the Egger's regression test are the observed effect sizes in our meta-analysis, divided by their standard error. The resulting values are equivalent to z-scores. These scores tell us directly if an effect size is significant; $z \geq 1.96$ or $z \leq -1.96$, we know that the effect is significant ($p < 0.05$). This outcome is regressed on the inverse of its standard error, which is equivalent to precision (Harrer et al. 2021).

In every linear regression model, the intercept represents the value of y when all other predictors are zero. The predictor in our model is the precision of a study, so the intercept shows the expected z-score when the precision is zero (i.e. when the standard error of a study is infinitely large).

Given a precision of 0, or an infinitely large standard error, we expect a z-score scattered around 0. However, when the funnel plot is asymmetric, for example due to publication bias, we expect that small studies with very high effect sizes will be considerably over-represented in our data, leading to a surprisingly high number of low-precision studies with high z-values. Due to this distortion, the predicted value of y for zero precision will be considerably larger than zero, resulting in a significant intercept.

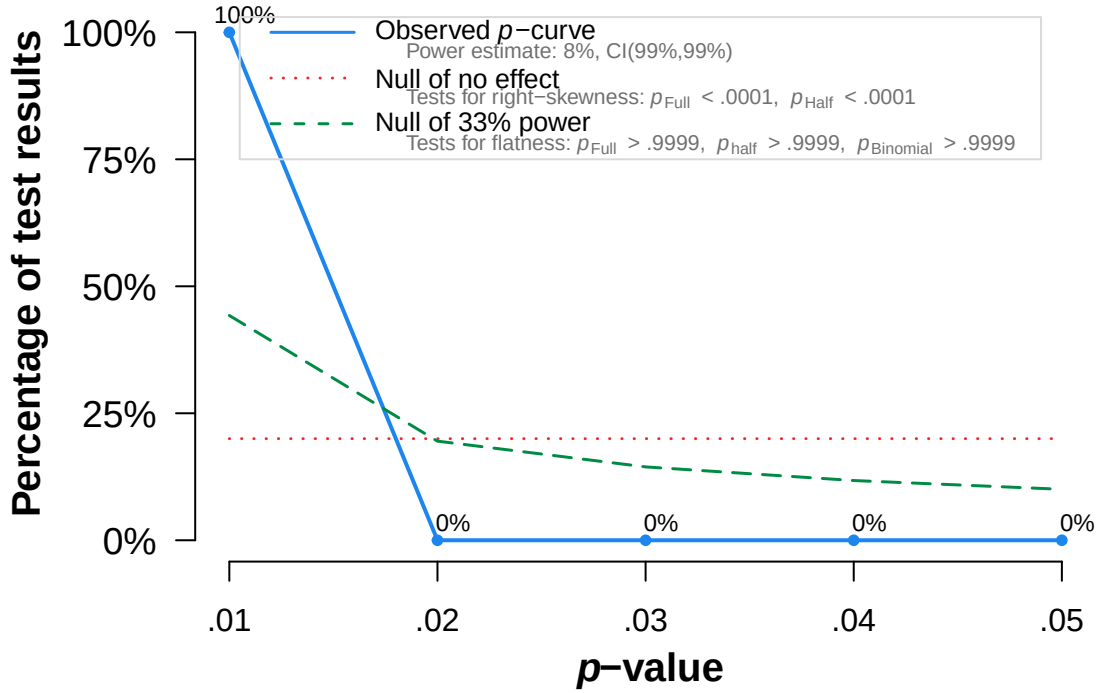
The problem with Egger's test is that it can lead to an inflation of false positive results when applied to standardized mean differences (SMDs) (Pustejovsky 2019; Harrer et al. 2021). The reason is that the standard error and the SMD are not independent of each other (smaller standard error = larger standardized effect). Should we find evidence for publication bias, we will therefore additionally conduct different analyses.

P-curve

We use a function from the `dmetar` package. It requires:

- a dataframe containing the calculated effect size (named TE, log-transformed if based on a ratio),
- a standard error (named seTE)
- a study label (named studlab) for each study.

```
# Create p-curve function compatible data
p_curve_accuracy_data <- accuracy_effect %>%
  mutate(TE = yi,
         seTE = sqrt(vi),
         studlab = observation_id)

# save p-curve plot
p_curve_accuracy <- pcurve(p_curve_accuracy_data)
```



Note: The observed p−curve includes 133 statistically significant (p < .05) results, of which 133 are p < .025. There were no non−significant results entered.

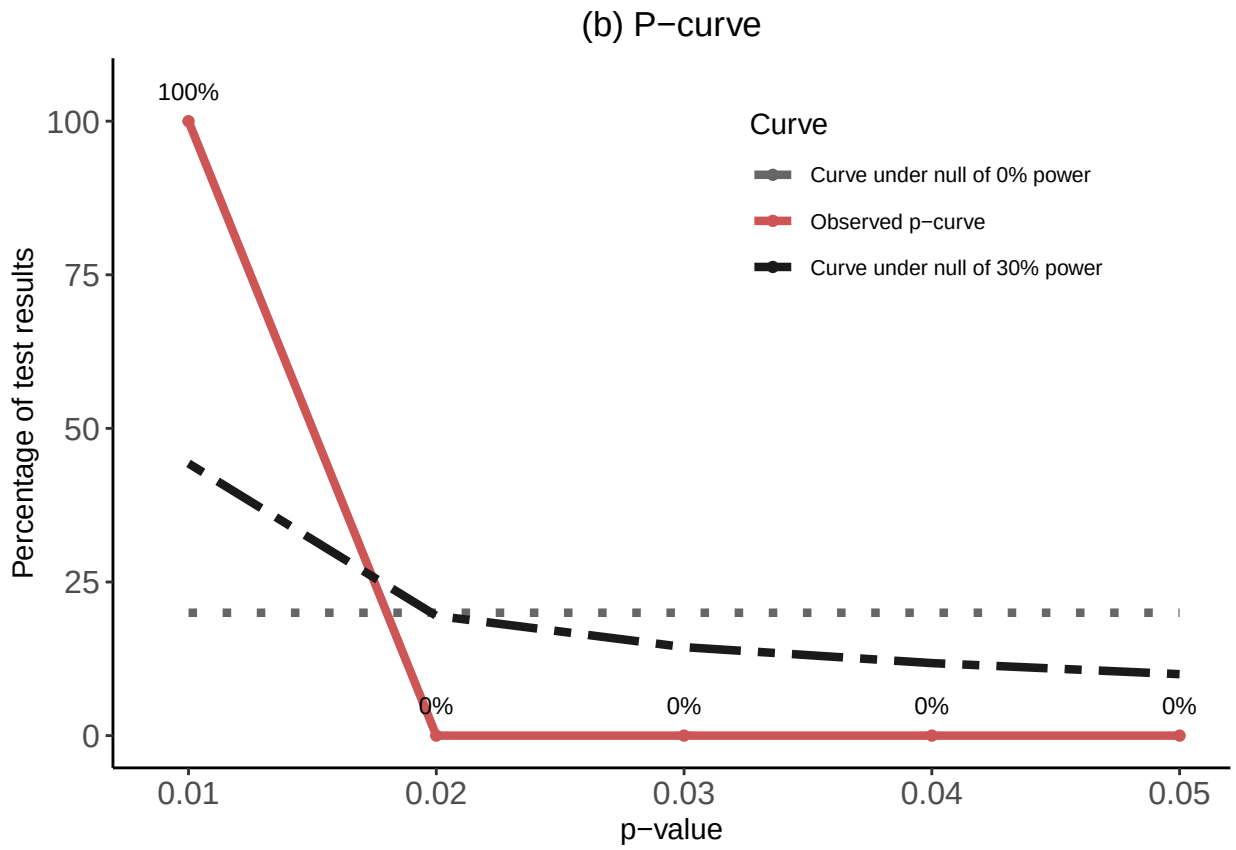
```
# Extract data to put into ggplot.
p_curve_accuracy_plot <- p_curve_accuracy$PlotData

# long format data works better in ggplot
p_curve_accuracy_plot <- pivot_longer(p_curve_accuracy_plot, cols = - `p-value`) %>%
              mutate(value = round(value, digits =1))
```

```
# Generates new p_curve plot from p_curve model results
p_curve_accuracy_plot <- ggplot(p_curve_accuracy_plot , aes(x = `p-value`, y = value, color = name)) +
  geom_line(aes(linetype = name), size = 1.5) + theme_classic() +
  geom_point(data = filter(p_curve_accuracy_plot ,  name == "Observed (blue)")) +
  geom_text( data = filter(p_curve_accuracy_plot ,  name == "Observed (blue)"),
            aes(label = paste(value, "%", sep = "")),
            size = 3.25, color = "Black",    vjust = -1.25 ) +
  ylab("Percentage of test results") +
  scale_color_manual(values=c('grey40', 'indianred3', "grey10"),
                     name = "Curve",
                     labels = c("Curve under null of 0% power",
                                "Observed p-curve",
                                "Curve under null of 30% power")) +
  ggtitle("(b) P-curve")  +
  theme(legend.position = c(.90, .95),
        legend.justification = c("right", "top"),
        legend.box.just = "right",
        legend.margin = margin(6, 6, 6, 6),
        legend.text = element_text(size = 8),
```

```
        plot.title = element_text(hjust = 0.5),
        axis.text.x = element_text(size=12),
        axis.text.y = element_text(size=12)) +
  scale_linetype_manual(values=c("dotted", "solid", "twodash")) +
  guides(linetype = FALSE) +ylim(NA, 105) + xlim(0.009,NA)
```

```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```

```
p_curve_accuracy_plot
```




```
# Combines funnel and p-curve plot into single plot.
# funnel + p_curve_accuracy_plot
# ggpubr::ggarrange(funnel, p_curve_accuracy_plot, ncol =2 , nrow = 1)
```

References

Chinn, Susan. 2000. "A Simple Method for Converting an Odds Ratio to Effect Size for Use in Meta-Analysis." *Statistics in Medicine* 19 (22): 3127–31. https://doi.org/10.1002/1097-0258(20001130)19:22%3C3127::AID-SIM784%3E3.0.CO;2-M.

Egger, M., G. D. Smith, M. Schneider, and C. Minder. 1997. "Bias in Meta-Analysis Detected by a Simple, Graphical Test." *BMJ* 315 (7109): 629–34. https://doi.org/10.1136/bmj.315.7109.629.

Gibbons, Robert D., Donald R. Hedeker, and John M. Davis. 1993. "Estimation of Effect Size from a Series of Experiments Involving Paired Comparisons." *Journal of Educational Statistics* 18 (3): 271–79. https://doi.org/10.3102/10769986018003271.

Harrer, Mathias, Pim Cuijpers, Furukawa Toshi A, and David D Ebert. 2021. *Doing Meta-Analysis with r: A Hands-on Guide*. 1st ed. Boca Raton, FL; London: Chapman & Hall/CRC Press.

Hedges, Larry V. 1981. "Distribution Theory for Glass's Estimator of Effect Size and Related Estimators." *Journal of Educational Statistics* 6 (2): 107–28. https://doi.org/10.3102/10769986006002107.

Higgins, Julian PT, James Thomas, Jacqueline Chandler, Miranda Cumpston, Tianjing Li, Matthew J. Page, and Vivian A. Welch. 2019. *Cochrane Handbook for Systematic Reviews of Interventions*. John Wiley & Sons.

Pustejovsky, James E. 2019. "Simulating Correlated Standardized Mean Differences for Meta-Analysis." https://www.jepusto.com/simulating-correlated-smds/.

Viechtbauer, Wolfgang. 2010. "Conducting Meta-Analyses in r with the **Metafor** Package." *Journal of Statistical Software* 36 (3). https://doi.org/10.18637/jss.v036.i03.