Simulations for 'Characterising the use of internal meta-analyses and assessing their impact'

Mandy Norrbo & Lisa DeBruine

27/02/2020

Contents

Dependencies

```
library(tidyverse)
library(metafor)
library(colorspace)
library(knitr)
```

Session info

```
sessionInfo()
```

```
## R version 3.6.0 (2019-04-26)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS  10.15.3
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_GB.UTF-8/en_GB.UTF-8/en_GB.UTF-8/C/en_GB.UTF-8/en_GB.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] knitr_1.28       colorspace_1.4-1 metafor_2.2-8    Matrix_1.2-17
##  [5] forcats_0.5.0    stringr_1.4.0    dplyr_0.8.4      purrr_0.3.3
##  [9] readr_1.3.1      tidyr_1.0.2      tibble_2.1.3     ggplot2_3.2.1
## [13] tidyverse_1.3.0
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_1.0.0 xfun_0.12        haven_2.2.0      lattice_0.20-38
##  [5] vctrs_0.2.3      generics_0.0.2   htmltools_0.4.0  yaml_2.2.1
##  [9] rlang_0.4.4      pillar_1.4.3     withr_2.1.2      glue_1.3.1
## [13] DBI_1.1.0        dbplyr_1.4.2     modelr_0.1.6     readxl_1.3.1
## [17] lifecycle_0.1.0  munsell_0.5.0    gtable_0.3.0     cellranger_1.1.0
## [21] rvest_0.3.5      evaluate_0.14    fansi_0.4.1      broom_0.5.5
## [25] Rcpp_1.0.3       scales_1.1.0     backports_1.1.5  jsonlite_1.6.1
## [29] fs_1.3.1         hms_0.5.3        digest_0.6.25    stringi_1.4.6
## [33] grid_3.6.0       cli_2.0.2        tools_3.6.0      magrittr_1.5
## [37] lazyeval_0.2.2   crayon_1.3.4     pkgconfig_2.0.3  xml2_1.2.2
## [41] reprex_0.3.0     lubridate_1.7.4  assertthat_0.2.1 rmarkdown_2.1
## [45] httr_1.4.1       rstudioapi_0.11  R6_2.4.1         nlme_3.1-139
## [49] compiler_3.6.0
```

Define functions

sim_func()

A function for simulating an independent samples t-test with sample size n and population effect size d. Observed scores are drawn from a normal distribution. The function lists resulting effect size (Cohen's d) and p-value.

```
sim_func <- function(n, d = 0) {
  dat <- tibble(
    grp = rep(LETTERS[1:2], each = n),
    score = c(rnorm(n, d, 1), rnorm(n, 0, 1)
  ))
  #t-test on simulated data
  myt <- t.test(score ~ grp, dat)
  #get p-value
  p <- myt$p.value
  #get effect size
  es <- cohens_d(myt$statistic[[1]], n, n)
  list("p" = p, "es" = es)
}
```

cohens_d()

A function for calculating Cohen's d for an independent samples t-test using a formula from Lakens (2013). The function requires a t-statistic and two sample sizes.

```
cohens_d <- function(t, n1, n2 = n1){
  t*sqrt(1/n1 + 1/n2)
}
```

var_d()

A function for calculating variance needed to calculate internal meta-analysis. The formula was obtained from Vosgerau, Simonsohn, Nelson, & Simmons (2019). Requires an effect size and a sample size.

```
var_d <- function(d,n){
  df <- (2*n-2)
  (2/n+(d^2)/(2*df)) * ((2*n)/(df))
}
```

mini_meta()

A function that runs an internal meta-analysis for a user-specified number of studies, with a population effect size d and sample size n (per group). Can use methods available in rma(), here "FE" (fixed effect) and "HE" (random effects).

```
mini_meta <- function(n.studies = 4, # no. of studies
                      d, # effect size
                      n, # sample size (per group)
                      method = "HE"){ # method for rma()

  study.ps <- vector() # vector for p-values
  study.effects <- vector() # vector for effect sizes
  study.var <- vector() # vector for variances

  for (i in 1:n.studies) { # loop until specified number of studies is reached
    study <- sim_func(n, d) # simulate study with n sample and d effect size
    study.ps[i] <- study$p # p-values in vector
    study.effects[i] <- study$es # effect sizes in vector
    study.var[i] <- var_d(study.effects[i], n) # variances in vector
  }
  # run internal meta-analysis
  minimeta <- rma(yi = study.effects, vi = study.var, method = method)

  data.frame( # add results to a data.frame
    study = 1:length(study.ps),
    p = study.ps,
    es = study.effects
  ) %>%
    add_row(
      study = 0, # internal meta-analysis results
      p = minimeta$pval, # meta-analysed p-value
      es = minimeta$beta[[1]] # meta-analysed effect size
    )
}
```

p_bound_meta()

A function that runs a user-specified number of studies all with a group sample size of n and true effect size of d. If specified conditions are met (e.g. first p-value < 0.05 and last p-value < 0.1), then results are mini meta-analysed. Can run fixed-effect or random-effects models (specified using "method").

```
p_bound_meta <- function(n.studies = 7, # no. of studies
                         d, # effect size
                         n, # sample size
                         method = "HE", # method for rma()
                         pmax.first = .05, # threshold for first p
                         pmax.last = .05){ # threshold for last p
  study.ps <- vector() # vector for p-values
  study.effects <- vector() # vector for effect sizes
  study.var <- vector() # vector for variances

  study <- sim_func(n, d) # simulate first study
  study.ps[1] <- study$p
  study.effects[1] <- study$es
  study.var[1] <- var_d(study.effects[1], n)

  if (study.ps[1] >= pmax.first) { # if first study p > pmax.first
    tbl <- data.frame( # create table of results
      study = 1,
      p = study.ps,
      es = study.effects,
      id = 0 # id studies that did not pass threshold
    )
    return(tbl)
  }
  # else, keep running studies until n.studies
  for (i in 2:n.studies) {
    study <- sim_func(n, d)
    study.ps[i] <- study$p
    study.effects[i] <- study$es
    study.var[i] <- var_d(study.effects[i], n)

    # unless p < pmax.last, then stop running more studies
    if (study$p < pmax.last) break
  }
  # run internal meta-analysis
  minimeta <- rma(yi = study.effects, vi = study.var, method = method)

  data.frame( # add results to a dataframe
    study = 1:length(study.ps),
    p = study.ps,
    es = study.effects,
    id = rep(1, each = length(study.ps))) %>%
      add_row(
      study = 0, # meta-analysis results
      p = minimeta$pval, # meta-analysed p-value
      es = minimeta$beta[[1]] # meta-analyseds effect size
    )
}
```

meta_hack()

A function that runs an internal meta-analysis after every new study is added. It allows user to specify both a first p-value threshold (pmax.first) as well as a 'minitarget', i.e. a threshold for the meta-analysed p-value that stops the running of more studies (unless max number of studies is reached before).

```
meta_hack <- function(n.studies = 7, # no. of studies
                      d, # effect size
                      n, # sample size
                      method = "HE", # method for rma()
                      pmax.first = .05, # first p threshold
                      minitarget = .05){ # meta-analysed p threshold
  # setting up results vectors
  study.ps <- vector()
  study.effects <- vector()
  study.var <- vector()

  # results of first study
  study <- sim_func(n, d)
  study.ps[1] <- study$p
  study.effects[1] <- study$es
  study.var[1] <- var_d(study.effects[1], n)

  if (study.ps[1] >= pmax.first) { # if first study p > pmax.first
    tbl <- data.frame( # create table of results
      study = 1,
      p = study.ps,
      es = study.effects,
      id = 0) # id studies that did not pass threshold
    return(tbl)
  }

  # else, keep running studies until n.studies or
  for (i in 2:n.studies) {
    study <- sim_func(n, d)
    study.ps[i] <- study$p
    study.effects[i] <- study$es
    study.var[i] <- var_d(study.effects[i], n)
    # meta p < minitarget
    minimeta <- rma(yi = study.effects, vi = study.var, method = method)
    if (minimeta$pval < minitarget) break
  }
  data.frame( # add results to data frame
    study = 1:length(study.ps),
    p = study.ps,
    es = study.effects,
    id = rep(1, each = length(study.ps))
  ) %>%
    add_row(
      study = 0, # meta-analysis results
      p = minimeta$pval, # meta p
      es = minimeta$beta[[1]] # meta es
    )
  }
```

Analysis

Power

Simulation 1

Simulation 1 is a power analysis for fixed and random effects internal meta-analyses, with varied number of studies, effect sizes and sample sizes. The simulation was split by number of studies (2, 3, 4, and 5) to reduce the duration of a single simulation. Iterations were also kept at 1000 to save time.

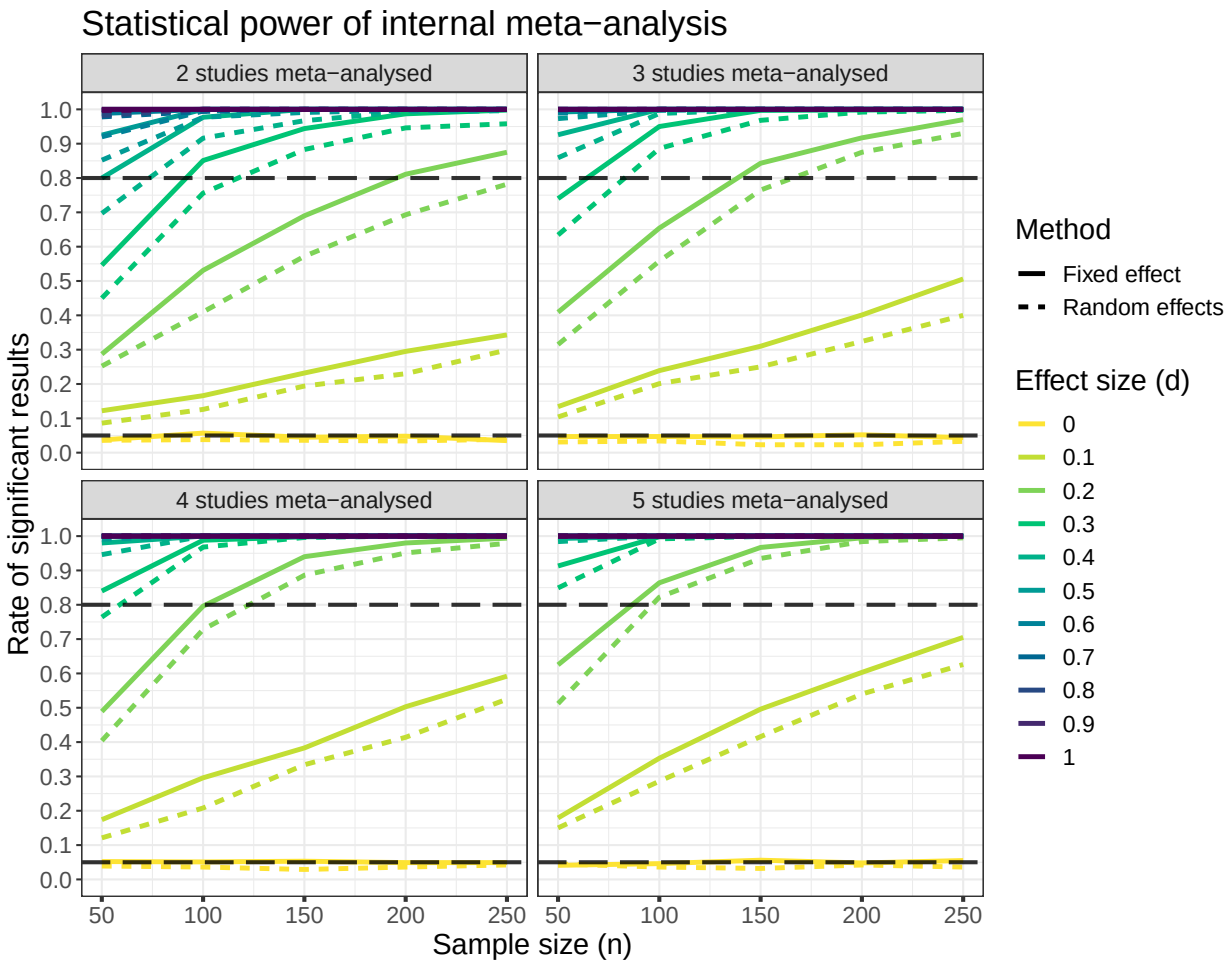
Parameters

- Iterations: 1000
- Number of studies: 2-5
- Effect size: 0-1
- Sample sizes: 50-250
- Method: random (HE) and fixed (FE) effects

```
set.seed(1337) # reproducible seed

params <- crossing( # all simulation parameters are fully crossed
  n.studies = c(2:5),
  n = seq(50, 250, 50),
  d = seq(0, 1, 0.1),
  iter = 1:1000,
  method = c("HE", "FE")
)

power_tmp <- purrr::pmap_dfr(params, function(...) {
  dots <- list(...)
  mini_meta(n.studies = dots$n.studies,
                d = dots$d,
                n = dots$n,
                method = dots$method) %>%
    mutate(!!!dots)
})

save(power_tmp, file = "power_fpr_tmp.RData")

# create dataframe of results
power_dat <- power_tmp %>%
  filter(study == 0) %>%
  group_by(n, method, d, n.studies) %>%
  summarise(power = mean(p < .05)) %>% # calculate power
  ungroup()

save(power_dat, file = "power_dat.RData")
```

Visualisation 1

```r
# load saved simulation results from chunk above
#load("power_fpr_tmp.RData)
load("power_dat.RData")

# facet labels
nstudies_labs <- c('2' = "2 studies meta-analysed",
                   '3' = "3 studies meta-analysed",
                   '4' = "4 studies meta-analysed",
                   '5' = "5 studies meta-analysed")


# plot
ggplot(power_dat, aes(n, power)) +
  geom_line(aes(color = factor(d),
                linetype = factor(method)), size=1.2) +
  facet_wrap(~n.studies,
             labeller = as_labeller(nstudies_labs)) +
  scale_y_continuous(limits = c(0, 1),
                     breaks = seq(0, 1, 0.1)) +
  scale_x_continuous(limits = c(50, 250),
                     breaks = seq(50, 250, 50)) +
  theme_bw() +
  scale_color_discrete_sequential("Viridis", name = "Effect size (d)") +
  scale_linetype(name = "Method",
                 labels = c("Fixed effect", "Random effects")) +
  theme(axis.title = element_text(size = 15),
        title = element_text(size = 15),
        axis.text = element_text(size = 10),
        axis.ticks.x = element_line(size = 1),
        axis.text.x = element_text(size=12),
        axis.text.y = element_text(size=12),
        legend.text = element_text(size = 12),
        strip.text.x = element_text(size = 12)) +
  labs(x = "Sample size (n)",
       y = "Rate of significant results",
       title = "Statistical power of internal meta-analysis") +
  geom_hline(yintercept = 0.8, size = 1, alpha = 0.8, linetype = "longdash") +
  geom_hline(yintercept = 0.05, size = 1, alpha = 0.8, linetype = "longdash")
```




```
# ggsave("minimeta_power_plot.png", width = 10, height = 7)
```

p-threshold

Simulation 2

Simulation 2 estimates the false positive rate of internal meta-analyses with p-value thresholds for the first study and a p-value threshold stopping rule. Sample sizes and methods (fixed or random effects) were varied, whereas p thresholds and the max number of studies were constant.

Parameters

- Iterations: 1000
- Max number of studies: 4 or 8
- Effect size: 0
- Sample sizes: 20-250
- First p max: 0.1 or 0.05
- Last p max: 0.05
- Method: random (HE) and fixed (FE) effects

```
set.seed(1337) # reproducible seed


# simulation parameters
params <- crossing(
  i = 1:10000, # iterations
  n.studies = c(4, 8), # maximum no. of studies (based on lit coding)
  n = seq(50, 250, 50), # vector of sample sizes (based on lit coding)
  d = 0, # population effect size
  pmax.first = c(0.05, 0.1), # vector of first p-values (based on lit coding)
  pmax.last = c(0.05,0.1), # vector of last p-values (based on lit coding)
  method = c("HE", "FE")
)

# using p_bound_meta()

pbound_tmp <- purrr::pmap_dfr(params, function(...) {
  dots <- list(...)
  p_bound_meta(n.studies = dots$n.studies,
                 d = dots$d,
                 n = dots$n,
                 method = dots$method,
                 pmax.first = dots$pmax.first,
                 pmax.last = dots$pmax.last) %>%
    mutate(!!!dots)
})

# save all results
save(pbound_tmp, file = "pbound_tmp.RData")

# calculate how many meta-analysed p-values
outof <- pbound_tmp %>%
  filter(study == 0) %>%
  group_by(pmax.first, pmax.last, n, method, d, n.studies) %>%
  count() %>%
  rename(metatotal = nn) %>%
```

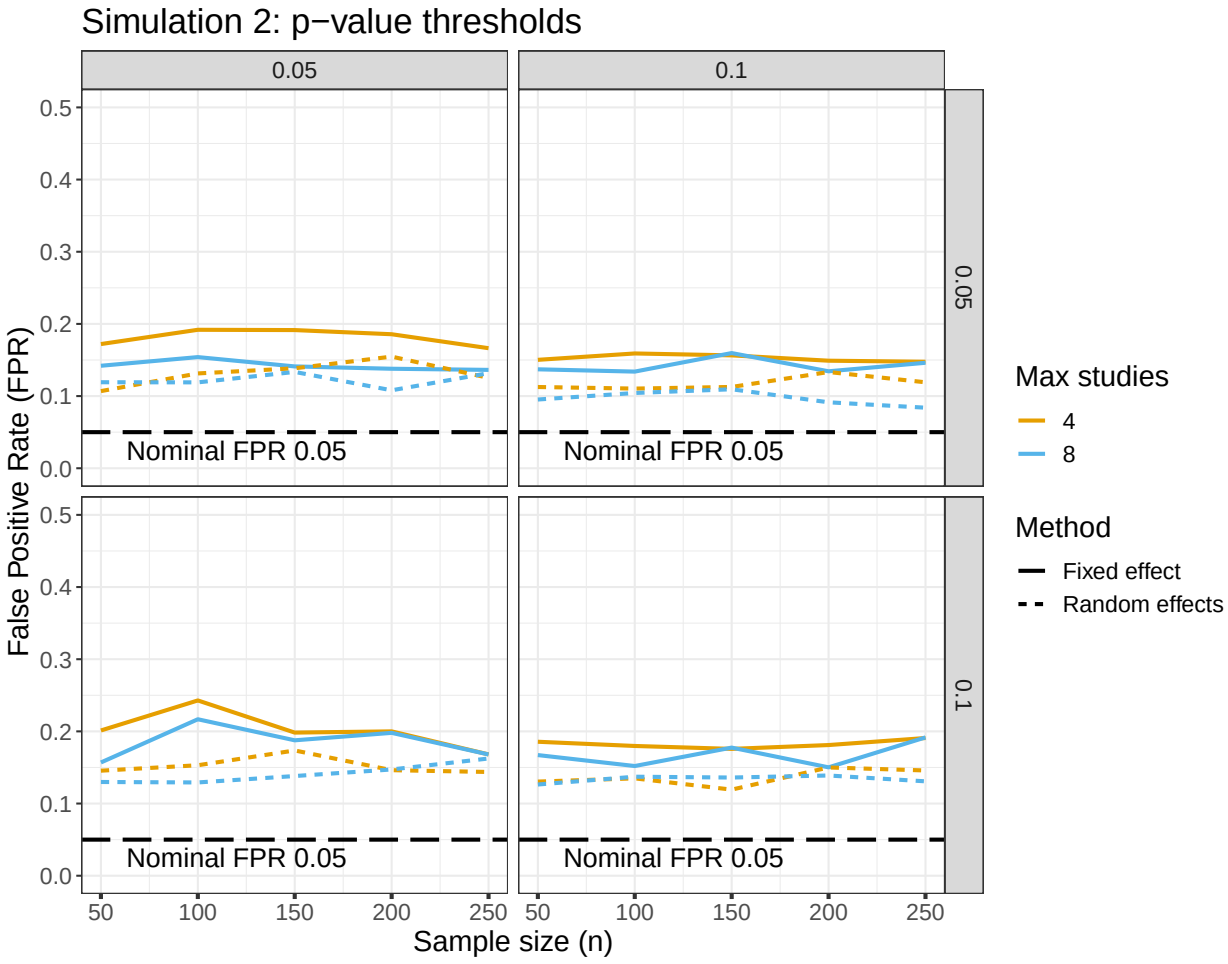
```
  ungroup()

dat <- pbound_tmp %>%
  filter(study == 0, p < .05) %>% # all significant meta results
  group_by(pmax.first, pmax.last, n, method, d, n.studies) %>%
  count() %>%
  ungroup()

pbound_dat <- left_join(dat, outof, by = c("n", "method", "pmax.first", "pmax.last", "n.studies")) %>%
  select(n, method, nn, metatotal, pmax.first, pmax.last, n.studies) %>%
  mutate(power = nn/metatotal) # calculate false positive rate

# save false positive rates for each combination
save(pbound_dat, outof, file = "pbound_dat.RData")
```

Visualisation 2

The colourblind-friendly palette used in the plot was obtained from Okaboe & Ito (2002).

```
#load("pbound_tmp.RData")
load("pbound_dat.RData") # load saved results for plot

# colourblind-friendly palette
cbPalette <- c("#E69F00", "#56B4E9", "#009E73",
               "#F0E442", "#0072B2", "#D55E00", "#CC79A7")

pbound_dat <- pbound_dat %>%
  rename(Method = method) %>%
  mutate(Method = recode(Method, "HE" = "Random effects", "FE" = "Fixed effect"))

first_labs <- c("p < 0.05", "p < 0.1")
last_labs <- c("p < 0.05", "p < 0.1")

ggplot(pbound_dat, aes(n, power)) +
  geom_line(aes(linetype = Method,
                color = factor(n.studies)), size=1) +
  scale_y_continuous(limits = c(0, 0.5),
                     breaks = seq(0, 0.5, 0.1)) +
  scale_x_continuous(limits = c(50, 250),
                     breaks = seq(50, 250, 50)) +
  geom_hline(yintercept = 0.05, size = 1,
             linetype = "longdash") +
  scale_color_manual(values=cbPalette, name = "Max studies") +
  theme_bw() +
  theme(axis.title = element_text(size = 15),
        title = element_text(size = 15),
        axis.text = element_text(size = 10),
        axis.ticks.x = element_line(size = 1),
        axis.text.x = element_text(size=12),
        axis.text.y = element_text(size=12),
        legend.text = element_text(size = 12),
        strip.text = element_text(size = 12)) +
```

```
  labs(x = "Sample size (n)",
       y = "False Positive Rate (FPR)",
       title = "Simulation 2: p-value thresholds") +
  annotate("text", x = 120, y = 0.025,
           label = "Nominal FPR 0.05", size = 5) +
  facet_grid(pmax.last~pmax.first)
```




```
# ggsave("minimeta_pbound_plot.png", width = 10, height = 7)
```

Meta-hacking

Simulation 3

Simulation 4 estimates the false positive rate of internal meta-analyses with p-value thresholds for the first study and a meta-analysed p threshold stopping rule. Sample sizes and methods (fixed or random effects) were varied, whereas p thresholds and the max number of studies were constant.

Parameters

- Iterations: 1000
- Max number of studies: 4 or 8
- Effect size: 0
- Sample sizes: 20-250
- First p max: 0.1 or 0.05
- Minitarget: 0.05
- Method: random (HE) and fixed (FE) effects

```
set.seed(1337) # reproducible seed

# simulation parameters
params <- crossing(
  i = 1:10000, # iterations
  n.studies = c(4,8), # max no. of studies
  d = 0, # true effect size
  n = seq(50, 250, 50), # vector of sample sizes
  pmax.first = 0.05, # first p-value threshold
  minitarget = 0.05, # minimeta threshold
  method = c("HE", "FE") # random (HE) and fixed effect (FE)
)

# simulation
metahack_tmp <- purrr::pmap_dfr(params, function(...) {
  dots <- list(...)
  meta_hack(n.studies = dots$n.studies,
                d = dots$d,
                n = dots$n,
                method = dots$method,
                pmax.first = dots$pmax.first,
                minitarget = dots$minitarget) %>%
    mutate(!!!dots)
})


# save all results
save(metahack_tmp, file = "metahack_tmp.RData")

# all meta-analysed p-values
outof <- metahack_tmp %>%
  filter(study == 0) %>%
  group_by(pmax.first, minitarget, n, method, d, n.studies) %>%
  count() %>%
  rename(metatotal = nn) %>%
  ungroup()
```

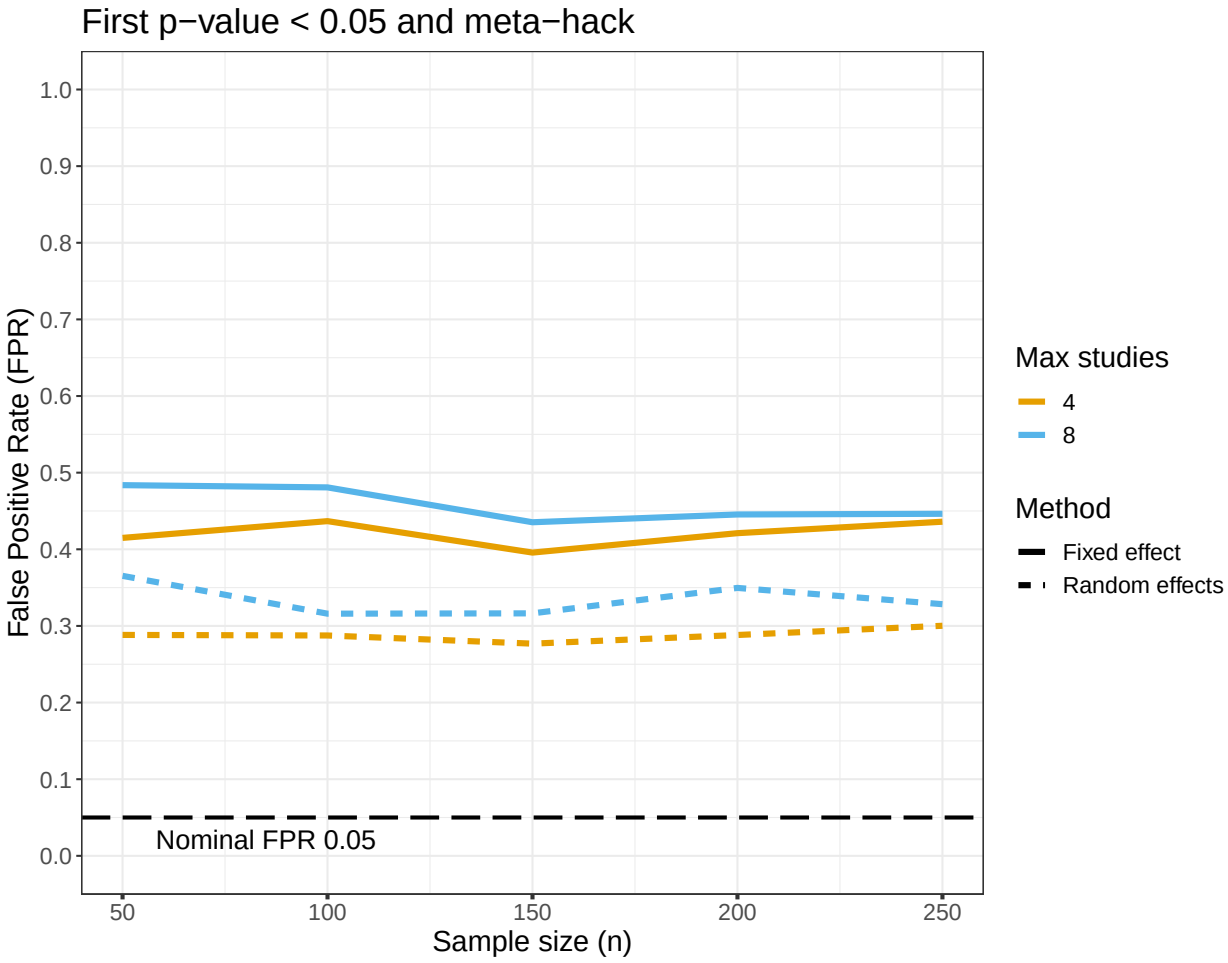
```r
# all significant meta-analysed p-values
dat <- metahack_tmp %>%
  filter(study == 0, p < .05) %>% # all significant meta results
  group_by(pmax.first, minitarget, n, method, d, n.studies) %>%
  count() %>%
  ungroup()

metahack_dat <- left_join(dat, outof, by = c("n", "method", "pmax.first", "minitarget", "n.studies")) %>%
  ungroup() %>%
  select(n, method, nn, metatotal, n.studies) %>%
  mutate(power = nn/metatotal) # calculate false positive rate

# save false positive rate results
save(dat, outof, metahack_dat, file = "metahack_dat.RData")
```

Visualisation 3

The colourblind-friendly palette used in the plot was obtained from Okaboe & Ito (2002).

```r
#load("metahack_tmp.RData")
load("metahack_dat.RData") # load saved results for plot

# changing column and row names for plot
metahack_dat <- metahack_dat %>%
  rename(Method = method) %>%
  mutate(Method = recode(Method, "HE" = "Random effects", "FE" = "Fixed effect"))

# colourblind-friendly palette
cbPalette <- c("#E69F00", "#56B4E9", "#009E73",
               "#F0E442", "#0072B2", "#D55E00", "#CC79A7")

ggplot(metahack_dat, aes(n, power)) +
  geom_line(aes(linetype = Method,
                color = factor(n.studies)), size=1.5) +
  scale_y_continuous(limits = c(0, 1),
                     breaks = seq(0, 1, 0.1)) +
  scale_x_continuous(limits = c(50, 250),
                     breaks = seq(50, 250, 50)) +
  geom_hline(yintercept = 0.05, size = 1, linetype = "longdash") +
  scale_color_manual(values=cbPalette, name = "Max studies") +
  theme_bw() +
  theme(axis.title = element_text(size = 15),
        title = element_text(size = 15),
        axis.text = element_text(size = 10),
        axis.ticks.x = element_line(size = 1),
        axis.text.x = element_text(size=12),
        axis.text.y = element_text(size=12),
        legend.text = element_text(size = 12)) +
  labs(x = "Sample size (n)",
       y = "False Positive Rate (FPR)",
       title = "First p-value < 0.05 and meta-hack") +
  annotate("text", x = 85, y = 0.022,
```

```
           label = "Nominal FPR 0.05", size = 5)
```




```
# ggsave("minimeta_metahack_plot.png", width = 10, height = 7)
```

Table of results

```
load("pbound_tmp.RData")
load("metahack_tmp.RData")

# p-threshold table
pbound_table <- pbound_tmp %>%
  mutate(significant = ifelse(p < 0.05, "yes", "no")) %>%
  group_by(id, significant) %>%
  count() %>%
  ungroup() %>%
  mutate(id = ifelse(is.na(id), "meta", id)) %>%
  mutate(id = recode(id, `0` = "filedrawer", `1` = "studies")) %>%
  mutate(proportion = round(n/nrow(pbound_tmp),2))

pbound_table %>%
 kable()
```

id	significant	n	proportion
filedrawer	no	740112	0.67
studies	no	260253	0.23
studies	yes	52343	0.05
meta	no	51121	0.05
meta	yes	8767	0.01

```
# meta-hack table
metahack_table <- metahack_tmp %>%
  mutate(significant = ifelse(p < 0.05, "yes", "no")) %>%
  group_by(id, significant) %>%
  count() %>%
  ungroup() %>%
  mutate(id = ifelse(is.na(id), "meta", id)) %>%
  mutate(id = recode(id, `0` = "filedrawer", `1` = "studies")) %>%
  mutate(proportion = round(n/nrow(metahack_tmp),2))

metahack_table %>%
  kable()
```

id	significant	n	proportion
filedrawer	no	190031	0.77
studies	no	34559	0.14
studies	yes	11753	0.05
meta	no	6219	0.03
meta	yes	3750	0.02

Further analyses

```r
# false positive rate of mini meta across parameters
metahack_fpr <- metahack_table[[5,3]]/sum(metahack_table$n[4:5])

# false positive rate of mini meta across parameters
pbound_fpr <- pbound_table[[5,3]]/sum(pbound_table$n[4:5])
```

References

Lakens, D. (2013). Calculating and reporting effect sizes to facilitate cumulative science : A practical primer for t-tests and anovas. *Frontiers in Psychology*, *4*, 1–12.

Okaboe, M., & Ito, K. (2002). Color universal design (cud) how to make figures and presentations that are friendly to colorblind people. Retrieved from https://jfly.uni-koeln.de/color/

Vosgerau, J., Simonsohn, U., Nelson, L., & Simmons, J. (2019). 99. *Journal of Experimental Psychology: General*, *148*(9), 1628–1639.