SODA Power Analysis

Ben Matthews

2023-06-23

Set up

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
source(here::here("script", "xx-functions.R"))
set.seed(nchar("soda power analysis") ^ 3)
```

```
# define parameters

soda_effect_size <- 0.1
# assumes larger soda at t1 gives larger soda at t2
# although we don't use this in the power calculations
# this is a small effect size - around 0.2-0.6 soda points
# (it depends on the location on the scale exactly how much)


miss_prop <- 0.25
# proportion of data at t2 (unrelated to t1 soda)

study_n <- 1000
# sample size

clusters_n <- 4
# number of recruitment conditions

cluster_effect_size <- 0.1
# this is the variance for the cluster random effect, again
# this isn't large (could be around a whole soda point either way)

id_effect_size <- 1
# this is between person variability and is very large
# so like 5 or 6 soda points
```

```
intervention_effect <- -1
# this is change over time
# this is a very large effect size
```

We can generate data with the following function (see xx-functions.R)

```
tmp <-
gen_soda_data(
  soda_effect_size = soda_effect_size,
  miss_prop = miss_prop,
  study_n = study_n,
  clusters_n = 4,
  cluster_effect_size = 0.1,
  id_effect_size = 1,
  intervention_effect = -0.5
)
```

This simulates a t1 soda score from the binomial distribution based on the cluster and individual effects then simulates a t2 soda score from the same distribution using the same cluster and individual effects, plus a small effect of the t1 cluster score.

We then only keep the proportion of the t2 data specified by miss_prop. this missing probability is uncorrelated with any other factors.

As a result soda at t1 and t2 are correlated:

```
cor(tmp |> filter(time == 0) |> pull(soda),
    tmp |> filter(time == 1) |> pull(soda),
    use = "complete.obs")
```

```
## [1] 0.7834749
```

P-values

For the analysis model we use:

```
glm_mod <-
  tmp |>
  MASS::glm.nb(soda ~ time, data = _)

# and then extract the p-value

broom::tidy(glm_mod)$p.value[[2]]
```

```
## [1] 1.453429e-09
```

This just models SODA as a count variable. We don't account for clustering within individuals within this model. we could do using:

```
glmer_mod <-
  tmp |>
  lme4::glmer.nb(soda ~ time + (1 | id), data = _)
```

```
## Warning in theta.ml(Y, mu, weights = object@resp$weights, limit = limit, :
## iteration limit reached
```

```
# and then extract the p-value with

broom.mixed::tidy(glmer_mod)$p.value[[2]]
```

```
## [1] 1.540298e-15
```

This actually gives more precise estimates, possibly because the data were generated with an individual random effect?

As a result I go for the more conservative model.

Effect sizes

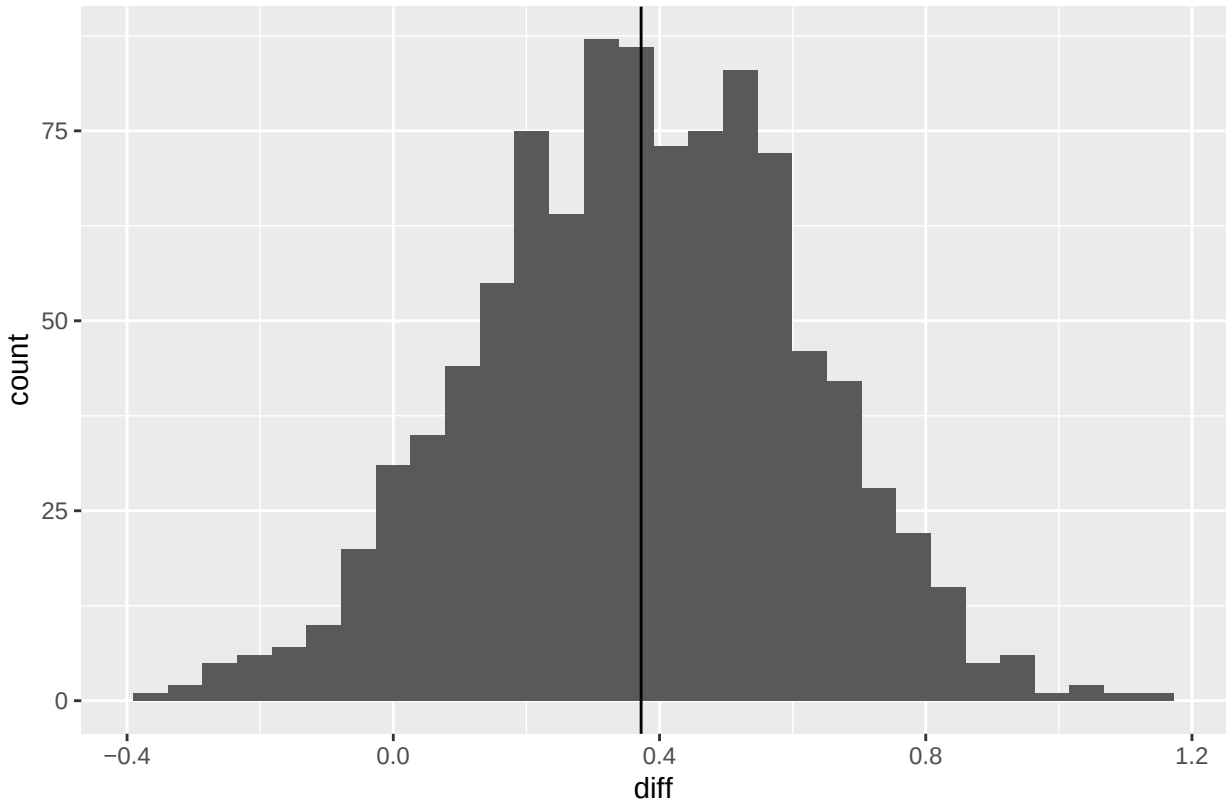
We can calculate some example effect sizes to see what might be plausible results. I try -0.1, -0.25, -0.5 and -1:

Effect size -0.1

```
sims_0.1 <-
tibble(
  sim_n = seq(1:1000)
) |>
  mutate(sim = map(sim_n, ~ gen_soda_data(
    soda_effect_size = soda_effect_size,
    miss_prop = miss_prop,
    study_n = study_n,
    clusters_n = 4,
    cluster_effect_size = 0.1,
    id_effect_size = 1,
    intervention_effect = -0.1
  )))


plot_effect_size(sims_0.1)
```

```
## `summarise()` has grouped output by 'sim_n'. You can override using the
## `.groups` argument.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Sold line is average effect size

```
calc_group_change(sims_0.1)
```

```
## # A tibble: 15 x 4
## # Groups:   0 [4]
##    `0`     `1`         n      prop
##    <chr>   <chr>   <int>     <dbl>
##  1 Group 1 Group 1 19205 0.599
##  2 Group 1 Group 2 11565 0.361
##  3 Group 1 Group 3  1292 0.0403
##  4 Group 2 Group 1 15148 0.210
##  5 Group 2 Group 2 38260 0.531
##  6 Group 2 Group 3 18506 0.257
##  7 Group 2 Group 4    86 0.00119
##  8 Group 3 Group 1  2507 0.0205
##  9 Group 3 Group 2 26082 0.213
## 10 Group 3 Group 3 85043 0.694
## 11 Group 3 Group 4  8915 0.0727
## 12 Group 4 Group 1     2 0.0000826
## 13 Group 4 Group 2   222 0.00917
## 14 Group 4 Group 3 12611 0.521
## 15 Group 4 Group 4 11369 0.470
```

Effect size of -0.1 would give:

- ~20% in group 2 down to 1
- around 22% in group 3 down to 2 or 1
- around 50% of people in group 4 dropping down

- and an average difference of around 0.4 points

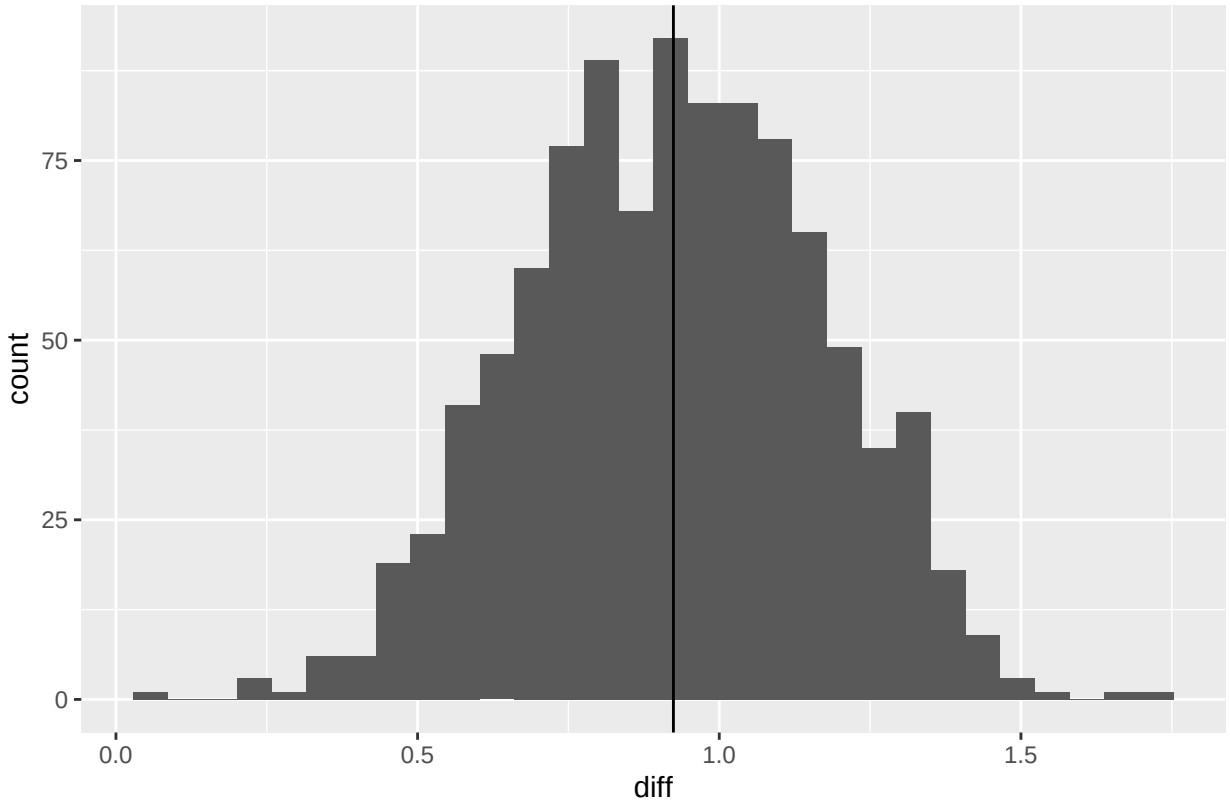
Note that some will also go up from group 1 to group 2 etc.

Effect size -0.25

```
sims_0.25 <-
  tibble(
    sim_n = seq(1:1000)
  ) |>
  mutate(sim = map(sim_n, ~ gen_soda_data(
    soda_effect_size = soda_effect_size,
    miss_prop = miss_prop,
    study_n = study_n,
    clusters_n = 4,
    cluster_effect_size = 0.1,
    id_effect_size = 1,
    intervention_effect = -0.25
  )))

plot_effect_size(sims_0.25)
```

```
## `summarise()` has grouped output by 'sim_n'. You can override using the
## `.groups` argument.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Sold line is average effect size

```
calc_group_change(sims_0.25)
```

```
## # A tibble: 16 x 4
## # Groups:   0 [4]
##    `0`     `1`         n      prop
##    <chr>   <chr>   <int>     <dbl>
##  1 Group 1 Group 1 21775 0.673
##  2 Group 1 Group 2  9804 0.303
##  3 Group 1 Group 3   752 0.0233
##  4 Group 1 Group 4     2 0.0000619
##  5 Group 2 Group 1 19833 0.275
##  6 Group 2 Group 2 38302 0.531
##  7 Group 2 Group 3 13922 0.193
##  8 Group 2 Group 4    45 0.000624
##  9 Group 3 Group 1  3859 0.0317
## 10 Group 3 Group 2 32542 0.267
## 11 Group 3 Group 3 79131 0.650
## 12 Group 3 Group 4  6232 0.0512
## 13 Group 4 Group 1     2 0.0000833
## 14 Group 4 Group 2   359 0.0150
## 15 Group 4 Group 3 13989 0.583
## 16 Group 4 Group 4  9659 0.402
```

Effect size of -0.25 would give:

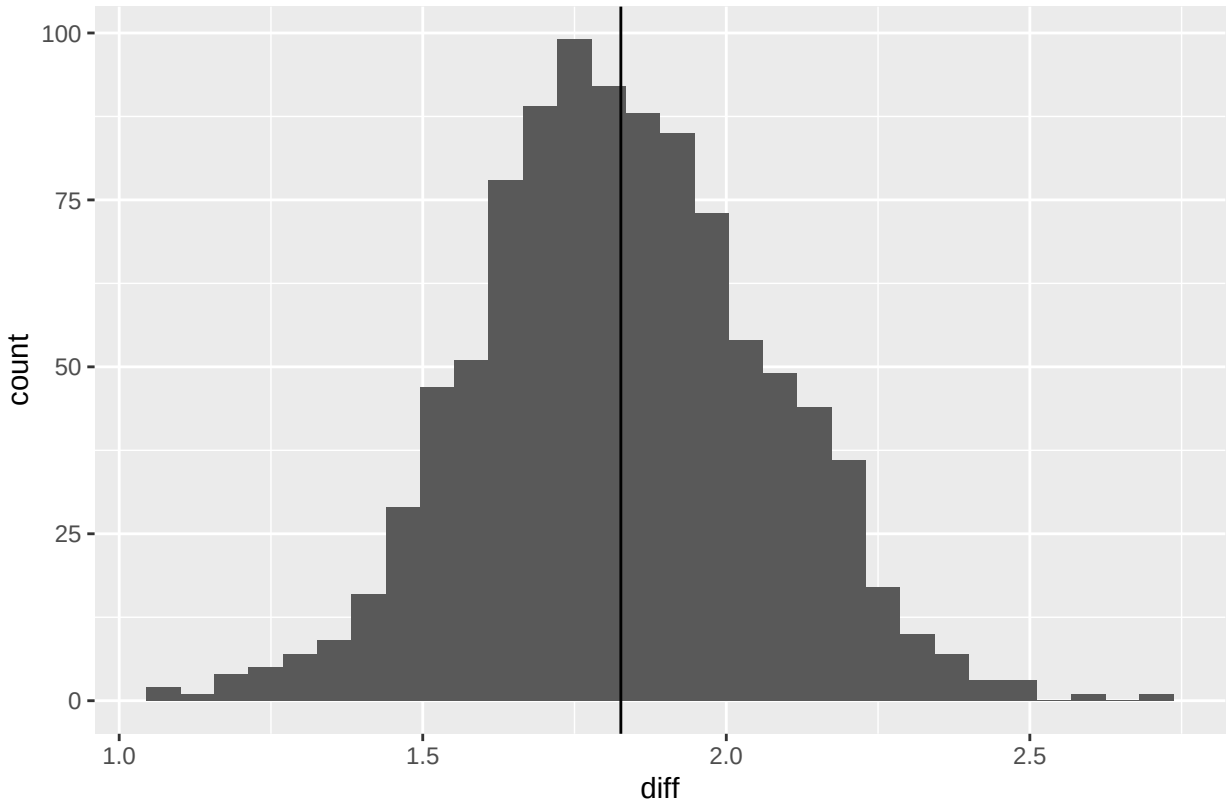
- ~27% in group 2 down to 1
- around 30% in group 3 down to 2 or 1
- and around 60% of people in group 4 dropping down
- and an average difference of around 0.8 points

Effect size -0.5

```
sims_0.5 <-
  tibble(
    sim_n = seq(1:1000)
  ) |>
  mutate(sim = map(sim_n, ~ gen_soda_data(
    soda_effect_size = soda_effect_size,
    miss_prop = miss_prop,
    study_n = study_n,
    clusters_n = 4,
    cluster_effect_size = 0.1,
    id_effect_size = 1,
    intervention_effect = -0.5
  )))

plot_effect_size(sims_0.5)
```

```
## `summarise()` has grouped output by 'sim_n'. You can override using the
## `.groups` argument.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Sold line is average effect size

```
calc_group_change(sims_0.5)
```

```
## # A tibble: 15 x 4
## # Groups:   0 [4]
##    `0`     `1`         n     prop
##    <chr>   <chr>   <int>    <dbl>
##  1 Group 1 Group 1 24820 0.773
##  2 Group 1 Group 2  6938 0.216
##  3 Group 1 Group 3   366 0.0114
##  4 Group 2 Group 1 27956 0.392
##  5 Group 2 Group 2 35553 0.498
##  6 Group 2 Group 3  7859 0.110
##  7 Group 2 Group 4    13 0.000182
##  8 Group 3 Group 1  7719 0.0633
##  9 Group 3 Group 2 43941 0.360
## 10 Group 3 Group 3 67175 0.551
## 11 Group 3 Group 4  3152 0.0258
## 12 Group 4 Group 1    19 0.000796
## 13 Group 4 Group 2   918 0.0385
## 14 Group 4 Group 3 16101 0.675
## 15 Group 4 Group 4  6825 0.286
```

Effect size of -0.5 would give:

- ~39% in group 2 down to 1
- around 42% in group 3 down to 2 or 1
- and around 70% of people in group 4 dropping down

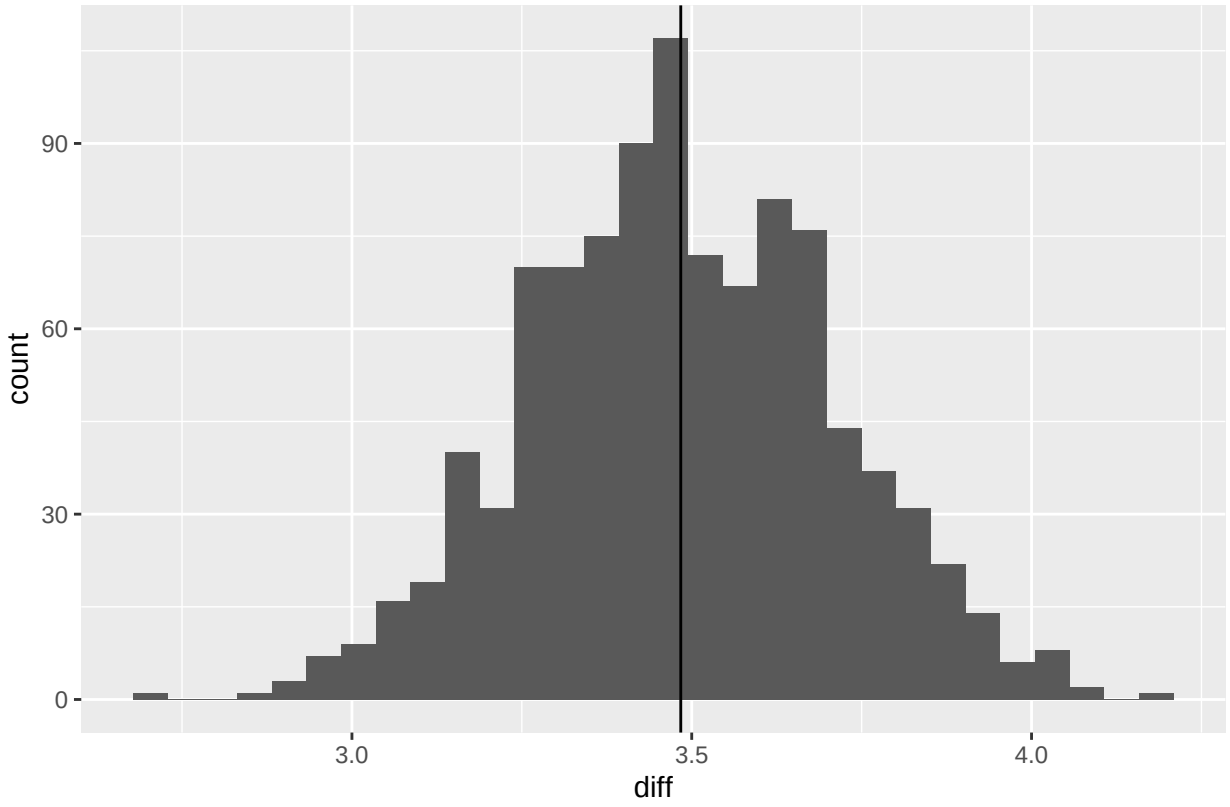
- and an average difference of around 1.7 points

Effect size -1

```
sims_1 <-
  tibble(
    sim_n = seq(1:1000)
  ) |>
  mutate(sim = map(sim_n, ~ gen_soda_data(
    soda_effect_size = soda_effect_size,
    miss_prop = miss_prop,
    study_n = study_n,
    clusters_n = 4,
    cluster_effect_size = 0.1,
    id_effect_size = 1,
    intervention_effect = -1
  )))


plot_effect_size(sims_1)
```

```
## `summarise()` has grouped output by 'sim_n'. You can override using the
## `.groups` argument.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Sold line is average effect size

```
calc_group_change(sims_1)
```

```
## # A tibble: 14 x 4
## # Groups:   0 [4]
##    `0`     `1`         n    prop
##    <chr>   <chr>   <int>   <dbl>
##  1 Group 1 Group 1 29125 0.909
##  2 Group 1 Group 2  2887 0.0901
##  3 Group 1 Group 3    43 0.00134
##  4 Group 2 Group 1 46334 0.643
##  5 Group 2 Group 2 23925 0.332
##  6 Group 2 Group 3  1774 0.0246
##  7 Group 3 Group 1 23128 0.190
##  8 Group 3 Group 2 59535 0.488
##  9 Group 3 Group 3 38667 0.317
## 10 Group 3 Group 4   646 0.00530
## 11 Group 4 Group 1   170 0.00711
## 12 Group 4 Group 2  3207 0.134
## 13 Group 4 Group 3 17523 0.733
## 14 Group 4 Group 4  3012 0.126
```

Effect size of -1 would give:

- ~60% in group 2 down to 1
- around 70% in group 3 down to 2 or 1
- and around 87% of people in group 4 dropping down
- and around 3.3 points difference on average

Power analysis

Based on the above I run a power analysis with effect sizes of 0.1, 0.25 and 0.5, assuming that an effect size of 1 is unlikely.

The code below runs the analysis and saves the result to disk. 1000 simulations took around 22 minutes.

```
res <-
tidyr::crossing(
  effect_size = c(-0.1, -0.25, -0.5),
  miss_prop = c(0.1, 0.25, 0.5),
  study_n = c(500, 1000, 1500)
) |>
  mutate(res = pmap(list(
    a = effect_size,
    b = miss_prop,
    c = study_n),
    \(a, b, c,  ...)
    run_power_soda(
               sim_n = 1000,
               soda_effect_size = soda_effect_size,
               miss_prop = b,
               study_n = c,
               clusters_n = 4,
               cluster_effect_size = 0.1,
               id_effect_size = 1,
              intervention_effect = a)))

saveRDS(res,
        here::here("results",
```

```
                    "soda-power-analysis_1000.rds"))
```

The results of the power analysis are presented below. A horizontal line is placed on the graph at power of 0.8. It calculates achieved N as a combination of the sample size and the proportion of missing cases.

An effect size of 0.5 could be detected with power larger than 0.8 in the smallest combination of N and largest proportion missing (500 and 11% - 55 cases.)

For an effect size of -0.25 it took until an achieved N of 200 to reach a power of 0.8.

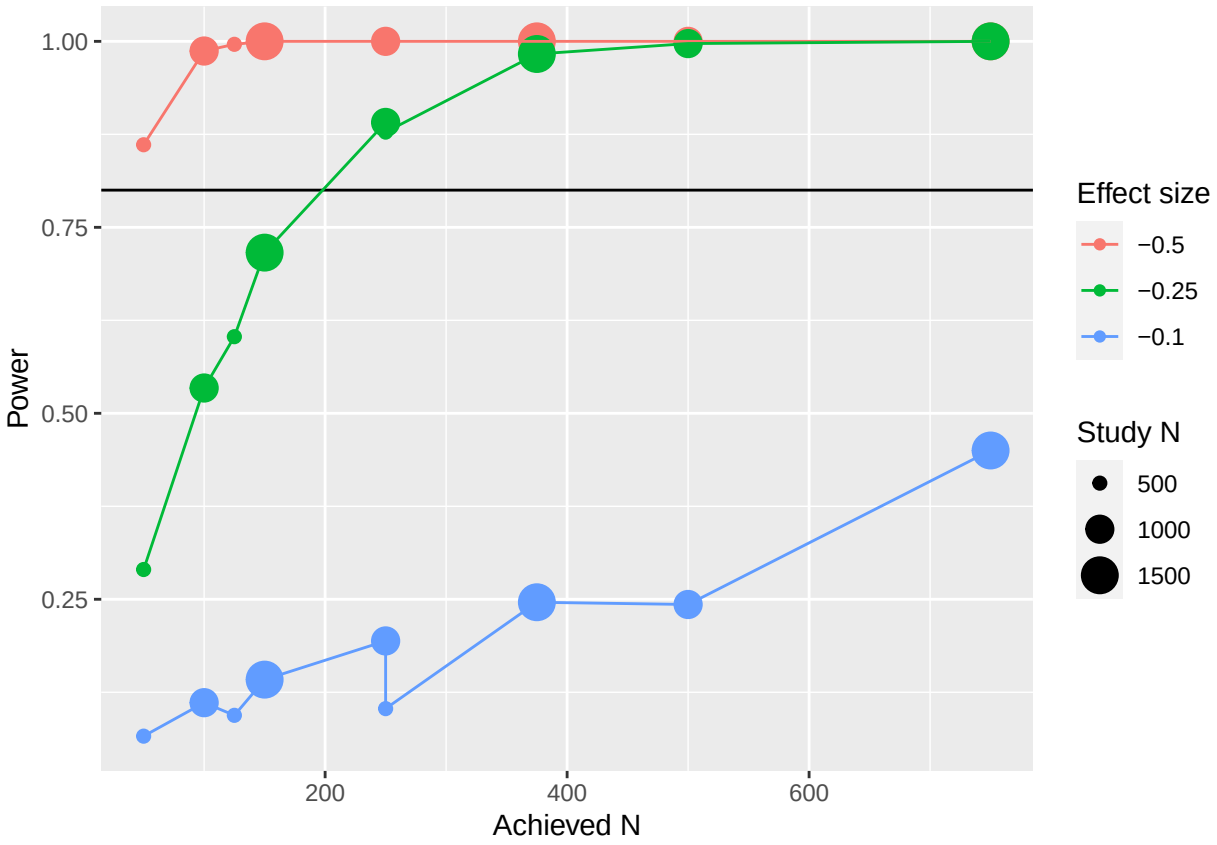
Effect size of -0.1 never achieved a power of 0.8 in these simulations.

```
res <- readRDS(here::here("results",
                          "soda-power-analysis_1000.rds"))

res |>
  unnest(res) |>
  group_by(effect_size, miss_prop, study_n) |>
  summarise(power = weighted.mean(signif, n)) |>
  ungroup() |>
  mutate(achieved_n = study_n * miss_prop) |>
  ggplot(aes(x = achieved_n,
             y = power,
             colour = factor(effect_size))) +
  geom_hline(yintercept = 0.8) +
  geom_point(aes(size = factor(study_n))) +
  geom_line(aes(group = effect_size)) +
  labs(x = "Achieved N",
       y = "Power",
       colour = "Effect size",
       size = "Study N")
```

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

```
## Warning: Using size for a discrete variable is not advised.
```


1.00
0.75
0.50
0.25
Power
200
400
600
Achieved N
Effect size
−0.5
−0.25
−0.1
Study N
500
1000
1500