Sunderlab tutorial on statistical power

Jahred Liddie

2024-04-11

What is power?

Questions to get us started:

- Can someone define statistical power in their own words?
- Have you ever done a power analysis and why?

In this tutorial, I'll give an overview on statistical power from the perspective of frequentist approaches, specifically when your study is trying to estimate some kind of parameter (e.g., some kind a coefficient representing an effect estimate, correlation coefficients, and many others). There are likely power and sample size considerations for other types of statistical analyses (e.g., prediction, Bayesian approaches, evaluation of a mechanistic model), but this won't be covered.

Recall that type II error refers to when we do not reject some null hypothesis in our analysis, but in reality, the alternative hypothesis is true. Confusingly, type II error is sometimes referred to as β, but I'll use δ to be clearer:

$$\delta = \Pr(\text{do not reject } H_0 \mid H_A \text{ is true})$$

We want type II error (and type I error) to be as small as possible. Power is $1 - \delta$, which means it is the probability that we reject the null hypothesis when the alternative hypothesis is true (in other words, how often we would expect to find evidence that the null hypothesis is false, when the alternative hypothesis is true). A default and "acceptable" (but arbitrary) level of power to detect some effect is considered to be greater than 0.8.

Before going further, let's discuss at least some of the questions power analyses can help answer:

- How many samples do I need to have an acceptable level of power based on what has been observed previously? This is especially helpful before beginning a study.
- With a known sample size (maybe based on funding/sampling/time constraints), how small of an effect can I reasonably detect?
- How is my statistical power affected under different possible scenarios?

Calculating statistical power requires assumptions, including to assumptions related to some anticipated/reasonable effect size, variation or residual error of your dependent variable, and potentially more complex factors (e.g., intracluster correlation) depending on your intended final statistical analyses. *Typically, you would use prior knowledge and understanding of the literature to make assumptions for these parameters when doing a power calculation (leave some time to discuss this).*

You may remember that there are closed-form/neat equations that describe power or sample size requirements for simple statistical analyses, like a t-test or simple linear regression. Here's an example for a comparison of (unpaired) sample means for two groups (where n is the sample size in each group):

$$n = \frac{(z_{\alpha/2} + z_{\delta})^2(\alpha_0^2 + \alpha_1^2)}{(\mu_0 - \mu_1)^2}$$

where $z_{\alpha/2}$ and z_{δ} are critical z-values (typically 1.96 and 0.84 for an alpha of 0.05 and delta of 0.2). Take a look and get some quick intuition for how n would change as we change the values of other parameters (holding the z values constant). Just for completeness' sake, we can rearrange the above to solve for power (sorry this is not fully simplified):

$$1 - \delta = \Phi\left(\sqrt{\frac{n \times (\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}} + z_{\alpha/2}\right)$$

where Φ refers to the cumulative distribution function. (If there's time, we can also talk about representing power in graphically alongside the null and alternative hypothesis distribution.)

The caveat here is that, for more complex modeling situations, we may not have a closed-form equation for power calculations to rely on. In these cases, simulations are particularly helpful for calculating power or finding a necessary minimum sample size. We will touch on two examples in this tutorial relating to each situation.

Example 1: a simple power calculation

This example follows directly from the comparison of sample means above. Let's have a simple situation where our difference in sample means is 1 and each group has a sample standard deviation of 2. We will calculate our statistical power as our sample size in each group increases.

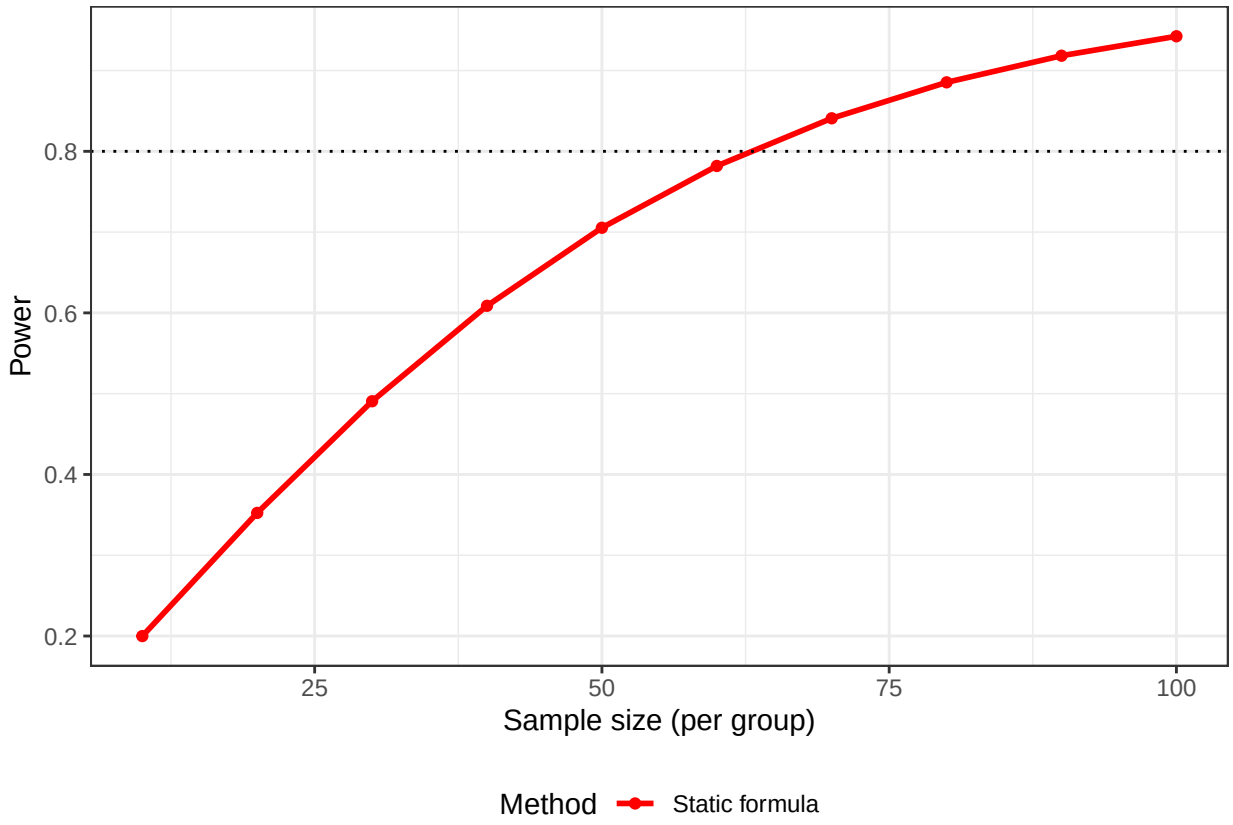
Static equation

```
mu1 <- 8
mu2 <- 7
sd1 <- 2
sd2 <- sd1

two_sample_function.f <- function(g1 = NULL, g2 = NULL, sigma1 = NULL,
                                  sigma2 = NULL, n = NULL) {

  power <- pnorm(sqrt((n * (g1 - g2)^2) /(sigma1^2 + sigma2^2)) -
                   qnorm(0.025, lower.tail = F))

  power.summary <- tibble(
    power = power,
    sample_size = n
  )

  return(power.summary)

}
```

```r
# test for n = 20 in each group:
two_sample_function.f(g1 = mu1, g2 = mu2, sigma1 = sd1, sigma2 = sd2, n = 20)
```

```
## # A tibble: 1 x 2
##   power sample_size
##   <dbl>       <dbl>
## 1 0.352          20
```

```r
# now test and save several sample sizes:
n.values <- seq(10, 100, by = 10)

# iterate function above over these sample sizes:
static.power <- map_dfr(n.values,
                        ~two_sample_function.f(g1 = mu1, g2 = mu2, sigma1 = sd1,
                                               sigma2 = sd2, n = .x)
        )

static.power$type <- "Static formula"

# now plot as sample size increases:
ggplot(static.power, aes(x = sample_size, y = power, color = type)) +
  geom_point() +
  geom_line(linewidth = 1) +
  geom_hline(yintercept = 0.8, color = "black", linetype = "dotted") +
  scale_color_manual(values = "red", name = "Method") +
  labs(x = "Sample size (per group)", y = "Power") +
  theme_bw() +
  theme(legend.position = "bottom")
```



Simulation-based approach

Now let's try the same thing, but using a simulation-based approach with the following steps:

1. Sample from a population distribution with using some assumptions for the parameters describing it. This is a "data generating process."
2. Run our statistical test
3. Repeat
4. Find the percent of the time that we have evidence to reject H_0 - this is our estimated power

```
# it can be useful to first wrap this into a function:
simulation1.f <- function(g1 = NULL, g2 = NULL, sigma1 = NULL, sigma2 = NULL, n = NULL) {

  # step 1
  sample1 <- rnorm(n = n, mean = g1, sd = sigma1)
  sample2 <- rnorm(n = n, mean = g2, sd = sigma2)
  # note: since sigma1 = sigma2, you can also easily do this with a
    # binary variable (B):
  # sample <- rnorm(n = n*2, mean = 7 + B*1, sd = sigma1 )
  # step 2
  result <- t.test(sample1, sample2,
                   alternative = "two.sided",
                   var.equal = FALSE)
```

```
    # clean up result
    result <- broom::tidy(result)

    # trim our results a bit
    result.summary <- result %>%
      dplyr::select(estimate:estimate2, statistic, p.value) %>%
      mutate(sample_size = n)

    return(result.summary)

}

# test once:
simulation1.f(g1 = mu1, g2 = mu2, sigma1 = sd1, sigma2 = sd2, n = 20)
```

```
## # A tibble: 1 x 6
##   estimate estimate1 estimate2 statistic p.value sample_size
##      <dbl>     <dbl>     <dbl>     <dbl>   <dbl>       <dbl>
## 1    0.729      7.57      6.84      1.08   0.286          20
```

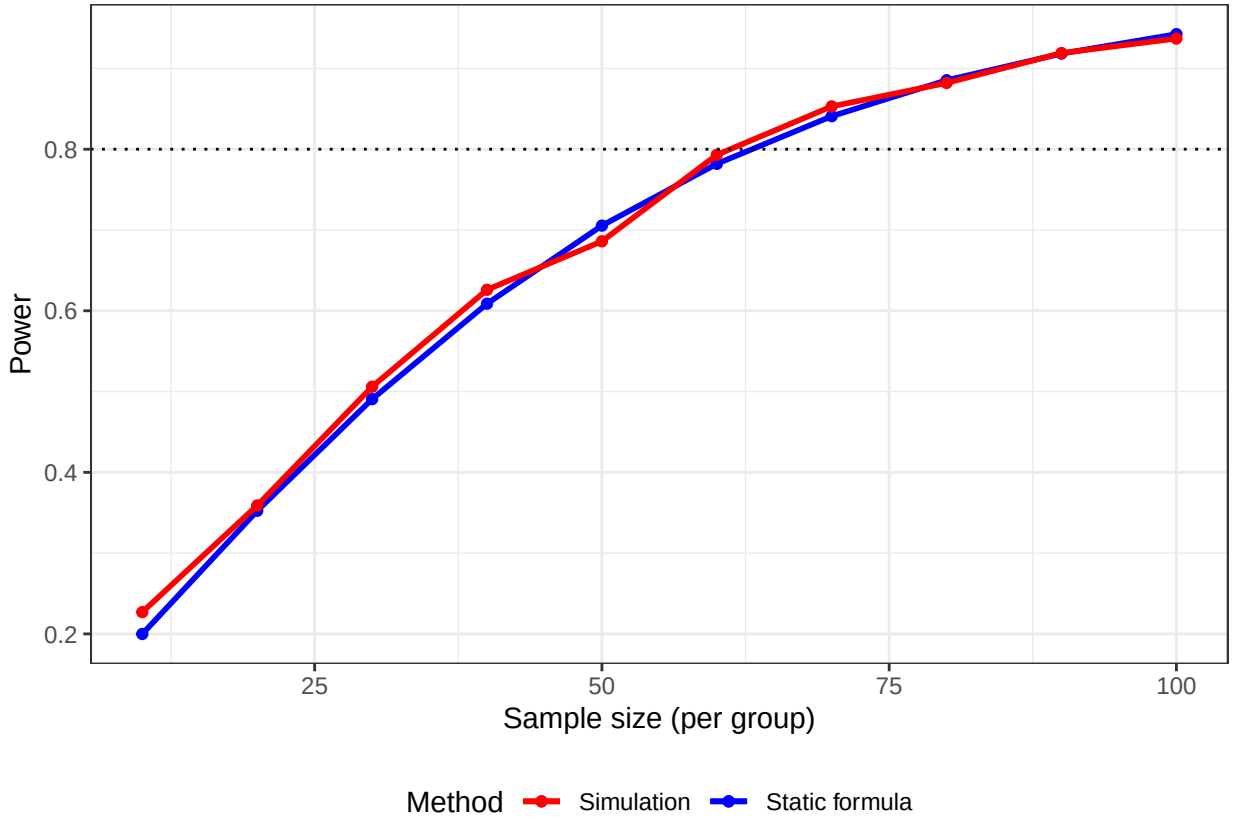
```
# now run this 100 times to and save:
full_sim1 <- map_df(1:100,
                    ~simulation1.f(g1 = mu1, g2 = mu2, sigma1 = sd1,
                                   sigma2 = sd2, n = 20)
                    )

# this is our power:
full_sim1$reject <- abs(full_sim1$statistic) > 1.96
mean(full_sim1$reject)
```

```
## [1] 0.32
```

Now repeat this for each sample size and compare with our calculations earlier:

```
set.seed(04122024)

n.values_sim1 <- tibble(n.values = rep(n.values, 1e3),
                        repeats = rep(1e3, length(n.values))
                        )

# this will take a second to run:
full_sim1 <- map2_df(.x = n.values_sim1$repeats, .y = n.values_sim1$n.values,
                     .f = ~simulation1.f(g1 = mu1, g2 = mu2, sigma1 = sd1,
                                         sigma2 = sd2, n = .y)
                     )

full_sim1$reject <- abs(full_sim1$statistic) > 1.96

full_sim1_summary <- full_sim1 %>%
  group_by(sample_size) %>%
  summarise(power = mean(reject)) %>%
  mutate(type = "Simulation")
```

```r
ggplot() +
  geom_point(data = static.power, aes(x = sample_size, y = power, color = type)) +
  geom_line(data = static.power, aes(x = sample_size, y = power, color = type),
            linewidth = 1) +
  geom_point(data = full_sim1_summary, aes(x = sample_size, y = power, color = type)) +
  geom_line(data = full_sim1_summary, aes(x = sample_size, y = power, color = type),
            linewidth = 1) +
  geom_hline(yintercept = 0.8, color = "black", linetype = "dotted") +
  scale_color_manual(values = c("red","blue"), name = "Method") +
  labs(x = "Sample size (per group)", y = "Power") +
  theme_bw() +
  theme(legend.position = "bottom")
```



Typically, larger repetitions (5 or 10k) are used. In a real situation, if you're unsure about an input parameter (in this case difference in means or variance) you may also test several scenarios to get a better sense of a decent sample size as well.

Example 2: A more complicated situation

Let's now use simulations to help in a more complicated situation, with a population model ("true relationship") like the following:

$$
\begin{aligned}
Y_i &= \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \varepsilon_i \\
\varepsilon &\sim \mathcal{N}(0, \sigma^2)
\end{aligned}
$$

or similarly:

$$E[Y_i] = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2$$

We'll consider X_1 and X_2 as binary variables. If you're unfamiliar with the idea of an interaction term, you can think of β_3 in this case as a "switch" that is turned "on" when $X_1 = 1$ and $X_2 = 1$. It is the additional effect when these two are present (above and beyond each of them individually).

This situation is much more complicated - I have not seen an closed-form solution in terms of understanding our power to detect β_3. However we can much more easily do this with simulations, if we can properly simulate a dataset (which is typically the hardest part). For simplicity, we'll ignore the possibility that X_1 and X_2 are correlated, but this is another thing to consider in a real situation.

```
set.seed(04122024)

simulation2.f <- function(b1 = 0.5, b2 = 0.5, b3 = 0.5, sigma = 1, n = 20) {
  # making the effect sizes a bit smaller to be interesting
  # create x1, x2, and y; defaulting x1 and x2 to have
    # 50% probability of equaling 1
  x1 <- sample(c(0, 1), n, replace = TRUE)

  x2 <- sample(c(0, 1), n, replace = TRUE)

  y <- rnorm(n = n, mean = 0.5 + b1*x1 + b2*x2 + b3*x1*x2, sd = sigma)

  our_sample <- tibble(
    y = y,
    x1 = x1,
    x2 = x2
  )

  return(our_sample)
}

# test to see what this does:
simulation2.f(b1 = 0.5, b2 = 0.5, b3 = 0.5, sigma = 0.5, n = 20)
```

```
## # A tibble: 20 x 3
##         y    x1    x2
##     <dbl> <dbl> <dbl>
##  1 0.732      0     1
##  2 0.361      0     0
##  3 3.02       1     1
##  4 2.26       1     1
##  5 0.568      0     1
##  6 0.374      1     0
##  7 1.53       0     1
##  8 1.63       1     0
##  9 1.84       1     1
## 10 0.753      1     0
## 11 1.67       1     1
## 12 0.760      1     0
## 13 0.261      1     0
## 14 0.584      0     0
```

```
## 15 1.86       1     1
## 16 1.72       1     1
## 17 1.60       1     0
## 18 1.02       0     1
## 19 1.53       1     1
## 20 0.0984     1     0
```

```r
# we'll now move on to step 2:
model2.f <- function(samp.size = NULL) {

  our_sample <- simulation2.f(n = samp.size)

  # use a linear model:
  our_model <- lm(y ~ x1 + x2 + x1*x2, data = our_sample)

  our_model_summary <- broom::tidy(our_model)

  our_model_summary$sample_size <- samp.size

  return(our_model_summary)

}

# test this too (note that i gave simulations2.f() default arguments)
model2.f(samp.size = 200)
```

```
## # A tibble: 4 x 6
##   term        estimate std.error statistic  p.value sample_size
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>       <dbl>
## 1 (Intercept)    0.507     0.140      3.62 0.000375          200
## 2 x1             0.643     0.200      3.21 0.00153           200
## 3 x2             0.505     0.197      2.56 0.0111            200
## 4 x1:x2          0.294     0.283      1.04 0.300             200
```

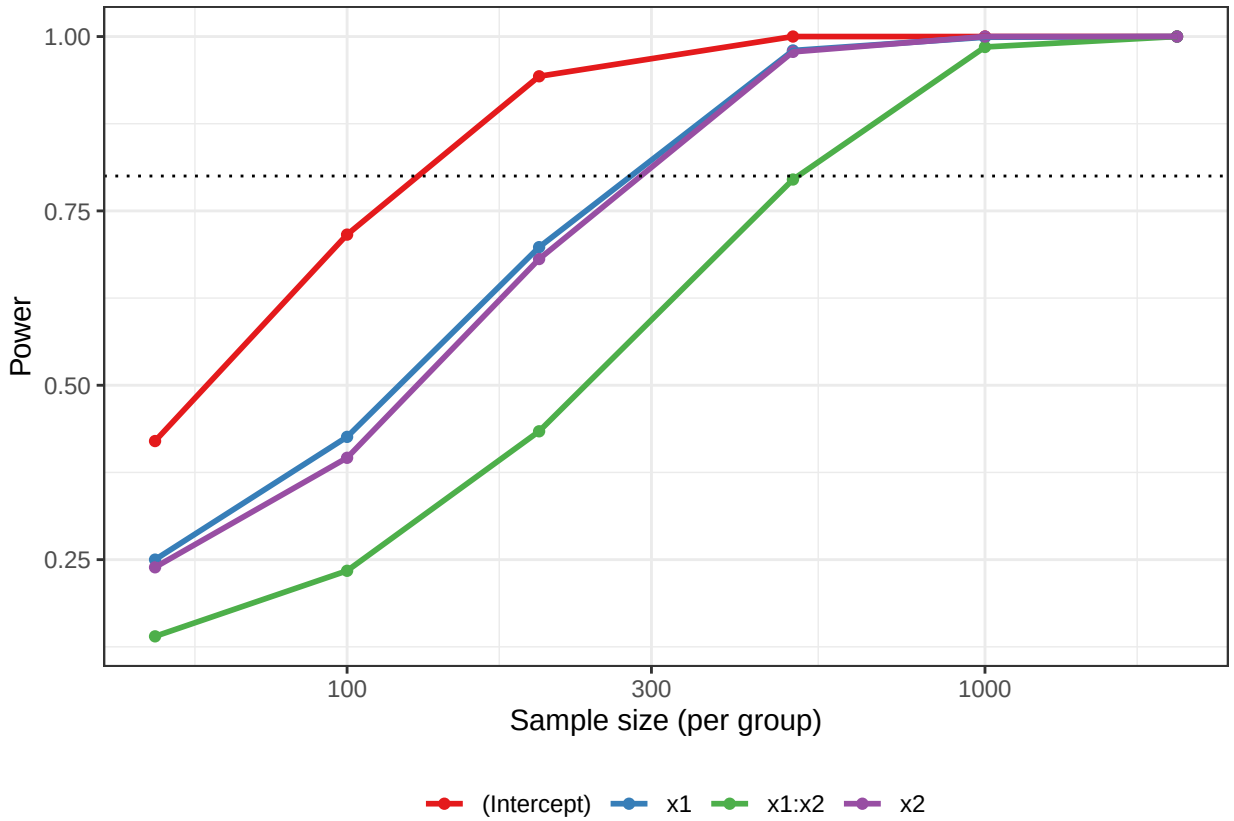
```r
# now let's scale this up with several sample sizes:
n.values <- c(50, 100, 200, 500, 1000, 2000)

n.values_sim2 <- tibble(
  n.values = rep(n.values, 1e3),
  repeats = rep(1e3, length(n.values))
  )

sim2_results <- map2_df(.x = n.values_sim2$repeats, .y = n.values_sim2$n.values,
                        ~model2.f(samp.size = .y))

sim2_results$reject <- abs(sim2_results$statistic) > 1.96

# quick check:
# View(
#   sim2_results %>%
#     group_by(term, sample_size) %>%
#     summarise(mean = mean(estimate))
# )
```

Now let's plot our power curves again and depict them as sample size is increasing:

```
sim2_power <- sim2_results %>%
  group_by(term, sample_size) %>%
  summarise(power = mean(reject))
```

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

```
ggplot(sim2_power, aes(x = sample_size, y = power, color = term)) +
  geom_point() +
  geom_line(linewidth = 1) +
  scale_colour_brewer(palette = "Set1", name = "") +
  labs(x = "Sample size (per group)", y = "Power") +
  geom_hline(yintercept = 0.8, color = "black", linetype = "dotted") +
  # remember to show unlogged version too:
  scale_x_log10() +
  theme_bw() +
  theme(legend.position = "bottom")
```



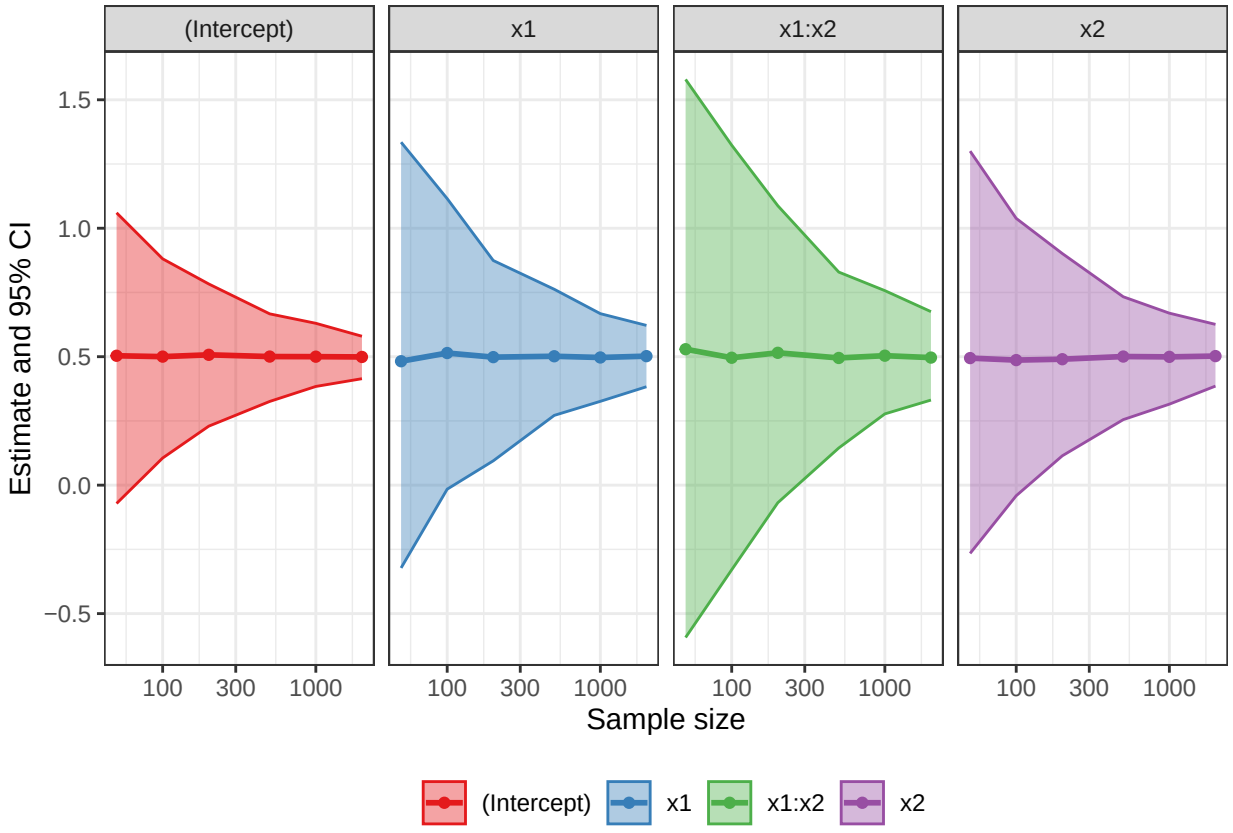
Let's also plot our confidence interval and its width as sample size increases:

```
# get confidence intervals using quantiles:
estimate_bounds <- sim2_results %>%
  group_by(term, sample_size) %>%
  summarise(
    UCL = quantile(estimate, 0.975),
```

```
    LCL = quantile(estimate, 0.025),
    mean = mean(estimate)
    )
```

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

```
ggplot(estimate_bounds, aes(x = sample_size, y = mean, ymin = UCL, ymax = LCL,
                            color = term, fill = term)) +
  geom_point() +
  geom_line(linewidth = 1) +
  geom_ribbon(alpha = 0.4) +
  scale_colour_brewer(palette = "Set1", name = "") +
  scale_fill_brewer(palette = "Set1", name = "") +
  facet_grid(~term) +
  scale_x_log10() +
  labs(x = "Sample size", y = "Estimate and 95% CI") +
  theme_bw() +
  theme(legend.position = "bottom")
```



Takeaways here?

More reading

If you're interested in more details on these approaches and other content related to simulations, one recommendation is this free online textbook, "Designing Simulations in R" (Miratrix and Pustejovsky, 2023).