Introduction to the Design and Analysis of Randomized Experiments
Class 4: Power Analysis

Jake Bowers

July 31, 2024

Overview and Review

Today

0. Quiz and Questions
1. Statistical Power (1 - false negative rate of tests)
2. Pre-registration of analysis plans

Note: You can download (and contribute to) course materials at https://github.com/bowers-illinois-edu/short_course_experiments

Hay recursos en español aqui: https://egap.github.io/theory_and_practice_of_field_experiments_spanish/

Lingering Questions?

Questions arising?

Quiz 1

What is wrong about these statements?

- "When we estimate the ATE using the difference of means, we have to assume that $Y \sim \text{Normal}()$."
- "We use a linear model to estimate the ATE. You can't just subtract the mean outcome of the treatment group from the mean outcome of the control group."

Quiz 2

What is wrong about these statements?

- “When we write $E_R[\widehat{ATE}] = ATE$ it means that our estimator of the ATE is unbiased. I looked up what ChatGPT says about”unbiased” and it says:

```
Here are some synonyms for "biased":

   1.  Prejudiced
   2.  Partial
   3.  One-sided
   4.  Slanted
   5.  Skewed
   6.  Predisposed
   7.  Influenced
   8.  Tendentious
   9.  Unfair
   10. Swayed
```

So, when we use an unbiased estimator we are not prejudiced or unfair. I also think that an unbiased estimator tells you the truth.”

Quiz 2

What is wrong about these statements?

- "When you use a block randomized trial you have to used a fixed effects estimator, with fixed effects for the blocks."
- "When you use a cluster randomized trial you have to use a multilevel model with random effects for the cluster."
- "When you have a binary outcome, the only meaningful estimates come from a logit model."

What is power?

What is power?

We want to separate signal from noise.

- Power $=$ probability of rejecting null hypothesis, given true effect $\neq 0$. We would like to have $p \leq \alpha$ for the hypothesis of no effects when the truth is not zero.
- It is the ability to detect signal from noise (assuming there is a signal).
- Formally: power $=$ (1 - Type II) error rate.
- Thus, power $\in$ (0, 1).
- How often should we see a $p \leq \alpha$? Standard thresholds: 0.8 or 0.9 — “nearly always detect a signal when one exists”

Starting point for power analysis

- Power analysis is something we do *before* we run a study.
  - Helps you figure out the sample you need to detect a given effect size.
  - Or helps you figure out a minimal detectable difference given a set sample size.
  - **May help you decide whether to run a study at all.** (A power analysis is part of answering the question, "Should we do this study?")
- It is hard to learn from an under-powered null finding.
  - Was there an effect, but we were unable to detect it? or was there no effect? We can't say.
  - How should we interpret "The difference in proportion vaccinated between Message A and Control was .02 ($p = .4$)."?

Power

- Say there truly is a treatment effect and you run your experiment many times (hypothetically) with the same group of people. How often will you get a $p \leq .05$?
- It depends:
  - How big is your treatment effect?
  - How many units are treated, measured?
  - How much noise is there in the measurement of your outcome?
- **We do not know the answers to all those questions in advance. So some guesswork required to answer this question.**

Approaches to power calculation

- Analytical calculations of power
- Simulate: Repeat the experiment with guessed-at treatment effect sizes, outcome variability, and N.

Power calculation tools

- Interactive
  - EGAP Power Calculator
  - rpsychologist
- R Packages
  - pwr
  - DeclareDesign, see also https://declaredesign.org/

Analytical calculations of power

Analytical calculations of power for hypotheses about no average treatment effects

- Formula:

$$\text{Power} = \Phi\left(\frac{|\tau|\sqrt{N}}{2\sigma} - \Phi^{-1}(1 - \frac{\alpha}{2})\right)$$

- Components:
  - ϕ: standard normal CDF is monotonically increasing
  - τ: the effect size
  - N: the sample size
  - σ: the standard deviation of the outcome
  - α: the significance level (typically 0.05)

Why standard normal? (the CLT!)

Example: Analytical calculations of power

```r
# Power for a study with 80 obserations and effect size of 0.25
library(pwr)
pwr.t.test(
  n = 40, d = 0.25, sig.level = 0.05,
  power = NULL, type = c(
    "two.sample",
    "one.sample", "paired"
  )
)
```

```
##
##      Two-sample t test power calculation
##
##               n = 40
##               d = 0.25
##       sig.level = 0.05
##           power = 0.1971831
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

Limitations to analytical power calculations

- Only derived for some estimands (ATE/ITT)
- Makes specific assumptions about the data-generating process (for example, N is large enough that the reference distribution for the test statistic is close to Normal)
- Difficult with more complex designs like block randomized designs with different probabilities of assignment in each block.

Simulation-based power calculation

Simulation-based power calculation steps

- Create dataset and simulate research design.
- Assumptions are necessary for simulation studies, but you make your own.
- For the DeclareDesign approach, see https://declaredesign.org/

Steps

- Define the sample and the potential outcomes function.
- Define the treatment assignment procedure.
- Create data.
- Assign treatment, then estimate the effect.
- Do this many times.

Examples

- Complete randomization
- With covariates
- With cluster randomization

Example: Simulation-based power for complete randomization

```r
# install.packages("randomizr")
library("randomizr")
library("estimatr")

## Y0 is fixed in most field experiments.
## So we only generate it once (here making it Normal parallels with
make_Y0 <- function(N) {
  rnorm(n = N)
}
repeat_experiment_and_test <- function(N, Y0, tau) {
  # N is size of experimental pool; Y0 is potential outcome to contro
  # tau is effect size (here, a constant additive effect)
  Z <- complete_ra(N = N)
  Y1 <- Y0 + Z * tau
  Yobs <- Z * Y1 + (1 - Z) * Y0
  estimator <- lm_robust(Yobs ~ Z)
  pval <- estimator$p.value[2]
  return(pval)
}
```

Example: Simulation-based power for complete randomization

```r
power_sim <- function(N, tau, sims) {
  Y0 <- make_Y0(N)
  pvals <- replicate(
    n = sims,
    repeat_experiment_and_test(N = N, Y0 = Y0, tau = tau)
  )
  pow <- sum(pvals < .05) / sims
  return(pow)
}

set.seed(12345)
## Notice simulation variability with sims=100
power_sim(N = 80, tau = .25, sims = 1000)
```

```
## [1] 0.157
```

```r
power_sim(N = 80, tau = .25, sims = 1000)
```

```
## [1] 0.237
```

Example: Using DeclareDesign I

```r
library(DeclareDesign)
library(tidyverse)
P0 <- declare_population(N, u0 = rnorm(N))
# declare Y(Z=1) and Y(Z=0)
O0 <- declare_potential_outcomes(Y_Z_0 = 5 + u0, Y_Z_1 = Y_Z_0 + tau)
# design is to assign m units to treatment
A0 <- declare_assignment(Z = conduct_ra(N = N, m = round(N / 2)))
# estimand is the average difference between Y(Z=1) and Y(Z=0)
estimand_ate <- declare_inquiry(ATE = mean(Y_Z_1 - Y_Z_0))
R0 <- declare_reveal(Y, Z)
design0_base <- P0 + A0 + O0 + R0

## For example with N=100 and tau=.25:
design0_N100_tau25 <- redesign(design0_base, N = 100, tau = .25)
dat0_N100_tau25 <- draw_data(design0_N100_tau25)
head(dat0_N100_tau25)
```

Example: Using DeclareDesign II

```
##    ID         u0 Z    Y_Z_0    Y_Z_1        Y
## 1 001  1.2790709 0 6.279071 6.529071 6.279071
## 2 002  2.3056456 0 7.305646 7.555646 7.305646
## 3 003 -0.9621603 1 4.037840 4.287840 4.287840
## 4 004 -0.8225563 1 4.177444 4.427444 4.427444
## 5 005  0.1702612 1 5.170261 5.420261 5.420261
## 6 006 -0.1926055 1 4.807394 5.057394 5.057394
```

```
with(dat0_N100_tau25, mean(Y_Z_1 - Y_Z_0)) # true ATE
```

```
## [1] 0.25
```

```
with(dat0_N100_tau25, mean(Y[Z == 1]) - mean(Y[Z == 0])) # estimate
```

```
## [1] 0.2217877
```

```
lm_robust(Y ~ Z, data = dat0_N100_tau25)$coef # estimate
```

```
## (Intercept)           Z
##   5.0297228   0.2217877
```

Example: Using DeclareDesign III

```r
E0 <- declare_estimator(Y ~ Z,
  .method = lm_robust, label = "t test 1",
  inquiry = "ATE"
)
t_test <- function(data) {
  test <- with(data, t.test(x = Y[Z == 1], y = Y[Z == 0]))
  data.frame(statistic = test$statistic, p.value = test$p.value)
}
T0 <- declare_test(handler = label_test(t_test), label = "t test 2")
design0_plus_tests <- design0_base + E0 + T0

design0_N100_tau25_plus <- redesign(design0_plus_tests, N = 100, tau

## Only repeat the random assignment, not the creation of Y0. Ignore
names(design0_N100_tau25_plus)
```

```
## [1] "P0"          "A0"          "O0"          "R0"          "t test 1" "t test
```

Example: Using DeclareDesign IV

```r
design0_N100_tau25_sims <- simulate_design(design0_N100_tau25_plus,
  sims = c(1, 100, 1, 1, 1, 1)
) # only repeat the random assignment
# design0_N100_tau25_sims has 200 rows (2 tests * 100 random assignme
# just look at the first 6 rows
head(design0_N100_tau25_sims)
```

```
##                    design   N  tau sim_ID estimator term  estimate
## 1 design0_N100_tau25_plus 100 0.25      1  t test 1    Z 0.2040486
## 2 design0_N100_tau25_plus 100 0.25      1  t test 2 <NA>        NA
## 3 design0_N100_tau25_plus 100 0.25      2  t test 1    Z 0.2000803
## 4 design0_N100_tau25_plus 100 0.25      2  t test 2 <NA>        NA
## 5 design0_N100_tau25_plus 100 0.25      3  t test 1    Z 0.3892886
## 6 design0_N100_tau25_plus 100 0.25      3  t test 2 <NA>        NA
##   statistic    p.value    conf.low conf.high df outcome inquiry s
## 1  1.040959 0.30045567 -0.184946209 0.5930435 98       Y     ATE
## 2  1.040959 0.30054813           NA        NA NA    <NA>    <NA>
## 3  1.020767 0.30987837 -0.188894886 0.5890555 98       Y     ATE
## 4  1.020767 0.30987957           NA        NA NA    <NA>    <NA>
```

Example: Using DeclareDesign V

```
## 5  1.990541 0.04931644  0.001187836 0.7773893 98        Y      ATE
## 6  1.990541 0.04938403           NA        NA NA     <NA>     <NA>
##   step_2_draw
## 1           1
## 2           1
## 3           2
## 4           2
## 5           3
## 6           3
```

Power with complete randomization

In 26% of experiments, when the truth is .25sds, and $N = 100$, we get $p < .05$.

```
## # A tibble: 2 x 2
##   estimator   pow
##   <chr>     <dbl>
## 1 t test 1   0.26
## 2 t test 2   0.26
```

Power with covariate adjustment

Covariate adjustment and power

- Covariate/Covariance adjustment can improve power because it mops up variation in the outcome variable.
    - If prognostic (predictive of the outcome), covariate adjustment can reduce variance dramatically. Lower outcome variance means higher power.
    - If non-prognostic, power gains are minimal.
- All covariates must be pre-treatment. Do not drop observations on account of missingness.
    - See the Theory and Practice module on threats to internal validity and the 10 things to know about covariate adjustment.
- Freedman (2008) pointed out that covariance-adjusted estimators of the ATE are biased. Lin (2013) shows that bias decreases with N.

Blocking

- Blocking: randomly assign treatment within blocks
  - "Ex-ante" covariate adjustment
  - Higher precision/efficiency implies more power
  - Reduce "conditional bias": association between treatment assignment and potential outcomes
  - Benefits of blocking over covariate adjustment clearest in small experiments

Example: Simulation-based power with a covariate I

```r
## Y0 is fixed in most field experiments. So we only generate it once
make_Y0_cov <- function(N) {
  u0 <- rnorm(n = N)
  x <- rpois(n = N, lambda = 2)
  Y0 <- .5 * sd(u0) * x + u0
  return(data.frame(Y0 = Y0, x = x))
}
##  X is moderately predictive of Y0.
test_dat <- make_Y0_cov(100)
test_lm <- lm_robust(Y0 ~ x, data = test_dat)
summary(test_lm)
```

```
##
## Call:
## lm_robust(formula = Y0 ~ x, data = test_dat)
##
## Standard error type:  HC2
##
## Coefficients:
```

Example: Simulation-based power with a covariate II

```
##             Estimate Std. Error t value  Pr(>|t|) CI Lower CI Uppe
## (Intercept)  -0.1223    0.17470 -0.7001 4.856e-01   -0.469   0.224
## x             0.5387    0.07341  7.3385 6.354e-11    0.393   0.684
##
## Multiple R-squared:  0.3716 ,    Adjusted R-squared:  0.3652
## F-statistic: 53.85 on 1 and 98 DF,  p-value: 6.354e-11
```

Example: Simulation-based power with a covariate III

```r
## now set up the simulation
repeat_experiment_and_test_cov <- function(N, tau, Y0, x) {
  Z <- complete_ra(N = N)
  Y1 <- Y0 + Z * tau
  Yobs <- Z * Y1 + (1 - Z) * Y0
  estimator <- lm_robust(Yobs ~ Z + x, data = data.frame(Y0, Z, x))
  pval <- estimator$p.value[2]
  return(pval)
}
## create the data once, randomly assign treatment sims times
## report what proportion return p-value < 0.05
power_sim_cov <- function(N, tau, sims) {
  dat <- make_Y0_cov(N)
  pvals <- replicate(n = sims, repeat_experiment_and_test_cov(
    N = N,
    tau = tau, Y0 = dat$Y0, x = dat$x
  ))
  pow <- sum(pvals < .05) / sims
  return(pow)
}
```

Example: Simulation-based power with a covariate I

Doing it twice to be clear that there is variability from simulation to simulation.

```
set.seed(12345)
power_sim_cov(N = 80, tau = .25, sims = 100)
```

```
## [1] 0.13
```

```
power_sim_cov(N = 80, tau = .25, sims = 100)
```

```
## [1] 0.19
```

Power for cluster randomization

Power and clustered designs

- Given a fixed N, a clustered design is often less powered than a non-clustered design.
  - The difference is often substantial.
- We have to estimate variance correctly:
  - Clustering standard errors (the usual)
  - Randomization inference
- To increase power:
  - Better to increase number of clusters than number of units per cluster if treatment is at level of cluster.
  - How much clusters reduce power depends critically on the intra-cluster correlation (the ratio of variance within clusters to total variance).

Example: Simulation-based power for cluster randomization I

```r
## Y0 is fixed in most field experiments. So we only generate it once
make_Y0_clus <- function(n_indivs, n_clus) {
  # n_indivs in number of people per cluster
  # n_clus is number of clusters
  clus_id <- gl(n_clus, n_indivs)
  N <- n_clus * n_indivs
  u0 <- fabricatr::draw_normal_icc(N = N, clusters = clus_id, ICC =
  Y0 <- u0
  return(data.frame(Y0 = Y0, clus_id = clus_id))
}
```

```r
test_dat <- make_Y0_clus(n_indivs = 10, n_clus = 100)
# confirm that this produces data with 10 in each of 100 clusters
table(test_dat$clus_id)
```

Example: Simulation-based power for cluster randomization II

```
##
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  1
##  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  1
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  3
##  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  1
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  5
##  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  1
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  7
##  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  1
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  9
##  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  1
```

```
# confirm ICC
ICC::ICCbare(y = Y0, x = clus_id, data = test_dat)
```

```
## [1] 0.09654799
```

Example: Simulation-based power for cluster randomization III

```
repeat_experiment_and_test_clus <- function(N, tau, Y0, clus_id) {
  # here we randomize Z at the cluster level
  Z <- cluster_ra(clusters = clus_id)
  Y1 <- Y0 + Z * tau
  Yobs <- Z * Y1 + (1 - Z) * Y0
  estimator <- lm_robust(Yobs ~ Z,
    clusters = clus_id,
    data = data.frame(Y0, Z, clus_id), se_type = "CR2"
  )
  pval <- estimator$p.value[2]
  return(pval)
}
power_sim_clus <- function(n_indivs, n_clus, tau, sims) {
  dat <- make_Y0_clus(n_indivs, n_clus)
  N <- n_indivs * n_clus
  # randomize treatment sims times
  pvals <- replicate(
    n = sims,
    repeat_experiment_and_test_clus(
```

Example: Simulation-based power for cluster randomization (DeclareDesign) I

```
P1 <- declare_population(
  N = n_clus * n_indivs,
  clusters = gl(n_clus, n_indivs),
  u0 = draw_normal_icc(N = N, clusters = clusters, ICC = .2)
)
O1 <- declare_potential_outcomes(Y_Z_0 = 5 + u0, Y_Z_1 = Y_Z_0 + tau)
A1 <- declare_assignment(Z = conduct_ra(N = N, clusters = clusters))
estimand_ate <- declare_inquiry(ATE = mean(Y_Z_1 - Y_Z_0))
R1 <- declare_reveal(Y, Z)
design1_base <- P1 + A1 + O1 + R1 + estimand_ate

## For example:
design1_test <- redesign(design1_base, n_clus = 10, n_indivs = 100, t
test_d1 <- draw_data(design1_test)
# confirm all individuals in a cluster have the same treatment assign
with(test_d1, table(Z, clusters))
```

Example: Simulation-based power for cluster randomization (DeclareDesign) II

```
##    clusters
## Z      1   2   3   4   5   6   7   8   9  10
##   0 100   0 100 100 100   0   0 100   0   0
##   1   0 100   0   0   0 100 100   0 100 100
```

Example: Simulation-based power for cluster randomization (DeclareDesign) III

```r
# four estimators, differ in se_type:
E1a <- declare_estimator(Y ~ Z,
  .method = lm_robust, clusters = clusters,
  se_type = "CR2", label = "CR2 cluster t test",
  inquiry = "ATE"
)
E1b <- declare_estimator(Y ~ Z,
  .method = lm_robust, clusters = clusters,
  se_type = "CR0", label = "CR0 cluster t test",
  inquiry = "ATE"
)
E1c <- declare_estimator(Y ~ Z,
  .method = lm_robust, clusters = clusters,
  se_type = "stata", label = "stata RCSE t test",
  inquiry = "ATE"
)
E1d <- declare_estimator(Y ~ Z,
  .method = lm_robust,
  se_type = "classical", label = "plain IID OLS t test",
```

Example: Simulation-based power for cluster randomization (DeclareDesign) I

```r
## Only repeat the random assignment, not the creation of Y0. Ignore
## We would want more simulations in practice.
set.seed(12355)
design1_sims <- simulate_design(design1_plus_tosim,
  sims = c(1, 1000, rep(1, length(design1_plus_tosim) - 2))
)
```

Simulation-based power for cluster randomizatino

Notice: high power but low coverage for plain OLS (“coverage” of a confidence interval is the same as false positive rate of a hypothesis test.)

```
design1_sims %>%
  group_by(estimator) %>%
  summarize(
    pow = mean(p.value < .05),
    coverage = mean(estimand <= conf.high & estimand >= conf.low),
    .groups = "drop"
  )
```

```
## # A tibble: 4 x 3
##   estimator                 pow coverage
##   <chr>                   <dbl>    <dbl>
## 1 CR0 cluster t test      0.155    0.911
## 2 CR2 cluster t test      0.105    0.936
## 3 plain IID OLS t test    0.723    0.333
## 4 stata RCSE t test       0.131    0.918
```

Example: Simulation-based power for cluster randomization (DeclareDesign) I

```r
## This may be simpler than the above:
library(DesignLibrary)
d1 <- block_cluster_two_arm_designer(
  N_blocks = 1,
  N_clusters_in_block = 10,
  N_i_in_cluster = 100,
  sd_block = 0,
  sd_cluster = .3,
  ate = .25
)
d1_plus <- d1 + E1b + E1c + E1d
d1_sims <- simulate_design(d1_plus, sims = c(1, 1, 1000, 1, 1, 1, 1,
```

Example: Simulation-based power for cluster randomization (DeclareDesign) II

```r
d1_sims %>%
  group_by(estimator) %>%
  summarize(
    pow = mean(p.value < .05),
    coverage = mean(estimand <= conf.high & estimand >= conf.low),
    .groups = "drop"
  )
```

```
## # A tibble: 4 x 3
##   estimator                pow coverage
##   <chr>                  <dbl>    <dbl>
## 1 CR0 cluster t test     0.209    0.914
## 2 estimator              0.143    0.941
## 3 plain IID OLS t test 0.707    0.464
## 4 stata RCSE t test      0.194    0.925
```

Power Ingredients and Their Relationship

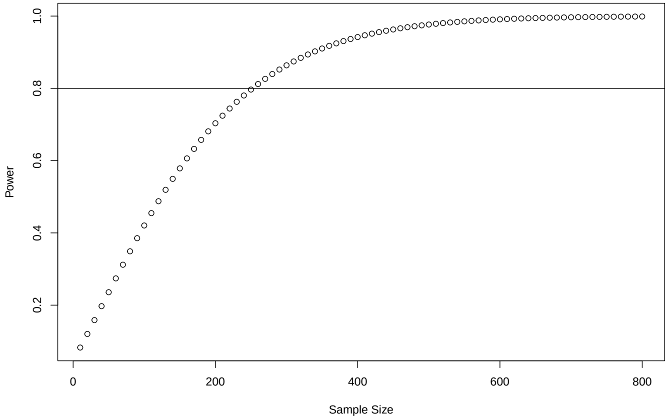
Comparative Statics

- Power is:
  - Increasing in N
  - Increasing in $|\tau|$
  - Decreasing in σ

Power by sample size I

```r
some_ns <- seq(10, 800, by = 10)
pow_by_n <- sapply(some_ns, function(then) {
  pwr.t.test(n = then, d = 0.25, sig.level = 0.05)$power
})
plot(some_ns, pow_by_n,
  xlab = "Sample Size",
  ylab = "Power"
)
abline(h = .8)
```

Power by sample size II

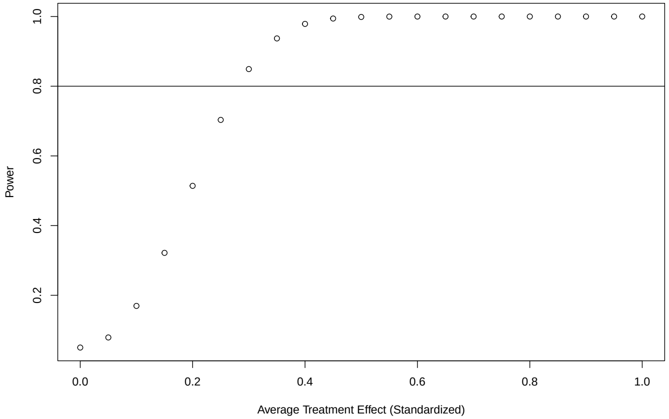


```
## See https://cran.r-project.org/web/packages/pwr/vignettes/pwr-vign
## for fancier plots
## ptest <-  pwr.t.test(n = NULL, d = 0.25, sig.level = 0.05, power =
## plot(ptest)
```

Power by treatment effect size I

```r
some_taus <- seq(0, 1, by = .05)
pow_by_tau <- sapply(some_taus, function(thetau) {
  pwr.t.test(n = 200, d = thetau, sig.level = 0.05)$power
})
plot(some_taus, pow_by_tau,
  xlab = "Average Treatment Effect (Standardized)",
  ylab = "Power"
)
abline(h = .8)
```

Power by treatment effect size II



EGAP Power Calculator

- The calculator at: https://egap.shinyapps.io/power-app/
- For cluster randomization designs, try adjusting:
  - Number of clusters
  - Number of units per clusters
  - Intra-cluster correlation
  - Treatment effect

Comments

- Know your outcome variable: what drives its variation regardless of treatment?
- What effects can you realistically expect from your treatment? What effects are substantively too small? (It may not be worth running *this experiment* at *this moment* if you cannot imagine detecting an effect at least this large.)
- What is the plausible range of variation of the outcome variable?
  - A design with limited possible movement in the outcome variable may not be well-powered.
  - See 10 Things Your Null Result Might Mean for discussion of the various reasons a given experiment might have produced a $p > .05$ (if you are using $\alpha = .05$ as a rejection criteria or if you have some substantively small $\hat{\bar{\tau}}$ that might not be reached).

Conclusion: How to improve your power

1. Increase the N
   - If clustered, increase the number of clusters if at all possible
2. Strengthen the treatment effect (τ_i and/or $\bar{\tau}$) (What might this mean in your study?)
3. Improve precision by removing extraneous noise from the outcome
   - Covariate adjustment
   - Blocking
4. Better measurement of the outcome variable (for example, using indicies to reduce noise in the outcome variable)

Pre-Registration of Analysis Plans

Bias in published research against null results I

A good design executed well will produce credible research, which might be a null result. We want credible and actionable null results.

- Manuscripts with null results are never submitted for review or put away in a "file drawer" after rejections.
- We face incentives to change your specifications, measurements, or even hypotheses to get a statistically significant result (p-hacking and HARKing) to improve chances of publication.
    - p-hacking: Trying many hypothesis tests increases the chances of a false positive result. (See Class 2 on Hythesis Testing Slides)
    - HARKing: (Hypotheses written After Results Known) Pretending that you are **testing** hypotheses when you are not (you are generating them — which is fine but a separate activity).
- Even people not facing these incentives make many decisions when they analyze data: handling missing values and duplicate observations, creating scales, etc. Different seemingly small decisions can produce substantively meaningful differences in published results.

Bias in published research against null results II

- Overall result: reduced credibility for individual pieces of research and (rightly) reduced confidence in whether we actually know what we claim to know.

Pre-registration of analysis plans and research designs I

- Pre-registration is the filing of your research design and hypotheses with a publicly-accessible repository with a credible date stamp. EGAP hosts one that you can use for free. Same with OSF. (EGAP registry lives within the larger OSF registry). See also the OES Analysis Plans.
- **Pre-registration does not preclude later exploratory analyses that were not stated in advance.** You just have to clearly distinguish between the two.

Pre-registration of analysis plans and research designs II

- Even if you will be submitting a paper with results to an academic journal or you are primarily interested in a final report with findings for a policy audience, there are important advantages to you and to other researchers from pre-registering your research.
  - You can learn about other research, completed and in progress; others can learn about yours. We can learn about studies that produced null results.
  - It forces you to state your hypotheses and plan of analysis in advance of seeing the results, which limits p-hacking and HARKing.

References

References I

Freedman, David A. (2008). “On regression adjustments to experimental data”. In: *Advances in Applied Mathematics* 40.2, pp. 180–193.

Lin, Winston (Mar. 2013). “Agnostic notes on regression adjustments to experimental data: Reexamining Freedman’s critique”. en. In: *The Annals of Applied Statistics* 7.1, pp. 295–318. ISSN: 1932-6157. (Visited on 10/07/2016).