

# Statistical Inference: Power

# Packages

```
library(tidyverse)
```

# Errors in testing

What can happen:

Decision		
Truth	Do not reject	Reject null
Null true	Correct	Type I error
Null false	Type II error	Correct

Tension between truth and decision about truth (imperfect).

- Prob. of type I error denoted  $\alpha$ . Usually fix  $\alpha$ , eg.  $\alpha = 0.05$ .
- Prob. of type II error denoted  $\beta$ . Determined by the planned experiment. Low  $\beta$  good.
- Prob. of not making type II error called **power** ( $= 1 - \beta$ ). *High power good.*

# Power

- Suppose  $H_0 : \mu = 10$ ,  $H_a : \mu \neq 10$  for some population mean  $\mu$ .
- Suppose  $H_0$  wrong. What does that say about  $\mu$ ?
- Not much. Could have  $\mu = 11$  or  $\mu = 8$  or  $\mu = 496$ . In each case,  $H_0$  wrong.
- How likely a type II error is depends on what  $\mu$  is:
  - ▶ If  $\mu = 496$ , should be able to reject  $H_0 : \mu = 10$  even for small sample, so  $\beta$  should be small (power large).
  - ▶ If  $\mu = 11$ , might have hard time rejecting  $H_0$  even with large sample, so  $\beta$  would be larger (power smaller).
- Power depends on true parameter value, and on sample size.
- So we play “what if”: “if  $\mu$  were 11 (or 8 or 496), what would power be?”.

# Figuring out power

- Time to figure out power is before you collect any data, as part of planning process.
- Need to have idea of what kind of departure from null hypothesis of interest to you, eg. average improvement of 5 points on reading test scores. (Subject-matter decision, not statistical one.)
- Then, either:
  - ▶ “I have this big a sample and this big a departure I want to detect. What is my power for detecting it?”
  - ▶ “I want to detect this big a departure with this much power. How big a sample size do I need?”

## How to understand/estimate power?

- Suppose we test  $H_0 : \mu = 10$  against  $H_a : \mu \neq 10$ , where  $\mu$  is population mean.
- Suppose in actual fact,  $\mu = 8$ , so  $H_0$  is wrong. We want to reject it. How likely is that to happen?
- Need population SD (take  $\sigma = 4$ ) and sample size (take  $n = 15$ ). In practice, get  $\sigma$  from pilot/previous study, and take the  $n$  we plan to use.
- Idea: draw a random sample from the true distribution, test whether its mean is 10 or not.
- Repeat previous step “many” times.
- “Simulation”.

# Making it go

- Random sample of 15 normal observations with mean 8 and SD 4:

```
x <- rnorm(15, 8, 4)
x
```

```
[1] 14.487469  5.014611  6.924277  5.201860  8.852952
[6] 10.835874  3.686684 11.165242  8.016188 12.383518
[11]  1.378099  3.172503 13.074996 11.353573  5.015575
```

- Test whether  $x$  from population with mean 10 or not (over):

...continued

```
t.test(x, mu = 10)
```

### One Sample t-test

data: x

t = -1.8767, df = 14, p-value = 0.08157

alternative hypothesis: true mean is not equal to 10

95 percent confidence interval:

5.794735 10.280387

sample estimates:

mean of x

8.037561

- Fail to reject the mean being 10 (a Type II error).



or get just P-value

```
ans <- t.test(x, mu = 10)  
ans$p.value
```

```
[1] 0.0815652
```

# How I knew it was called that

```
str(ans)
```

List of 10

```
$ statistic : Named num -1.88
..- attr(*, "names")= chr "t"
$ parameter : Named num 14
..- attr(*, "names")= chr "df"
$ p.value : num 0.0816
$ conf.int : num [1:2] 5.79 10.28
..- attr(*, "conf.level")= num 0.95
$ estimate : Named num 8.04
..- attr(*, "names")= chr "mean of x"
$ null.value : Named num 10
..- attr(*, "names")= chr "mean"
$ stderr : num 1.05
$ alternative: chr "two.sided"
$ method : chr "One Sample t-test"
$ data.name : chr "x"
- attr(*, "class")= chr "htest"
```

## Run this lots of times

- without a loop!
- use `rowwise` to work one random sample at a time
- draw random samples from the truth
- test that  $\mu = 10$
- get P-value
- Count up how many of the P-values are 0.05 or less.

## In code

```
library(tidyverse)
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(my_sample = list(rnorm(15, 8, 4))) %>%
  mutate(t_test = list(t.test(my_sample, mu = 10))) %>%
  mutate(p_val = t_test$p.value) %>%
  count(p_val <= 0.05)
```

# A tibble: 2 x 2

# Rowwise:

	<code>`p_val &lt;= 0.05`</code>	<code>n</code>
	<code>&lt;lg1&gt;</code>	<code>&lt;int&gt;</code>
1	FALSE	578
2	TRUE	422

We correctly rejected 422 times out of 1000, so the estimated power is 0.422.

## Aside: Try again with bigger sample

```
tibble(sim = 1:1000) %>%  
  rowwise() %>%  
  mutate(my_sample = list(rnorm(40, 8, 4))) %>%  
  mutate(t_test = list(t.test(my_sample, mu = 10))) %>%  
  mutate(p_val = t_test$p.value) %>%  
  count(p_val <= 0.05)
```

```
# A tibble: 2 x 2
```

```
# Rowwise:
```

	`p_val <= 0.05`	n
	<lgl>	<int>
1	FALSE	119
2	TRUE	881

## Calculating power

- Simulation approach very flexible: will work for any test. But answer different each time because of randomness.
- In some cases, for example 1-sample and 2-sample t-tests, power can be calculated.
- `power.t.test`. Input `delta` is difference between null and true mean:

```
power.t.test(n = 40, delta = 10-8, sd = 4,  
             type = "one.sample")
```

# Results

```
power.t.test(n = 15, delta = 10-8, sd = 4,  
             type = "one.sample")
```

One-sample t test power calculation

```
      n = 15  
    delta = 2  
      sd = 4  
sig.level = 0.05  
  power = 0.4378466  
alternative = two.sided
```

## Comparison of results

Method	Power
Simulation	0.422
<code>power.t.test</code>	0.4378

- Simulation power is similar to calculated power; to get more accurate value, repeat more times (eg. 10,000 instead of 1,000), which takes longer.
- CI for power based on simulation approx.  $0.42 \pm 0.03$ .
- With this small a sample size, the power is not great. With a bigger sample, the sample mean should be closer to 8 most of the time, so would reject  $H_0 : \mu = 10$  more often.



## Calculating required sample size

- Often, when planning a study, we do not have a particular sample size in mind. Rather, we want to know how big a sample to take. This can be done by asking how big a sample is needed to achieve a certain power.
- The simulation approach does not work naturally with this, since you have to supply a sample size.
  - ▶ For that, you try different sample sizes until you get power close to what you want.
- For the power-calculation method, you supply a value for the power, but leave the sample size missing.
- Re-use the same problem:  $H_0 : \mu = 10$  against 2-sided alternative, true  $\mu = 8$ ,  $\sigma = 4$ , but now aim for power 0.80.

## Using power.t.test

- No n=, replaced by a power=:

```
power.t.test(power=0.80, delta=10-8, sd=4, type="one.sample")
```

One-sample t test power calculation

```
      n = 33.3672
delta = 2
    sd = 4
sig.level = 0.05
  power = 0.8
alternative = two.sided
```

- Sample size must be a whole number, so round up to 34 (to get at least as much power as you want).

# One-sided test

```
power.t.test(power=0.80, delta=10-8, sd=4,  
             type="one.sample", alternative = "one.sided")
```

One-sample t test power calculation

```
      n = 26.13751  
delta = 2  
    sd = 4  
sig.level = 0.05  
  power = 0.8  
alternative = one.sided
```

## By simulation

Try a sample size and see what power you get. Here's  $n = 15$  from before:

```
tibble(sim = 1:1000) %>%  
  rowwise() %>%  
  mutate(my_sample = list(rnorm(15, 8, 4))) %>%  
  mutate(t_test = list(t.test(my_sample, mu = 10))) %>%  
  mutate(p_val = t_test$p.value) %>%  
  count(p_val <= 0.05)
```

# A tibble: 2 x 2

# Rowwise:

	<code>`p_val &lt;= 0.05`</code>	<code>n</code>
	<code>&lt;lgl&gt;</code>	<code>&lt;int&gt;</code>
1	FALSE	578
2	TRUE	422

To get power 0.80, two-sided, need a *bigger sample*.

## To get a bigger power

How much bigger? No idea. *Make any guess.* What about  $n = 50$ ?

```
tibble(sim = 1:1000) %>%  
  rowwise() %>%  
  mutate(my_sample = list(rnorm(50, 8, 4))) %>%  
  mutate(t_test = list(t.test(my_sample, mu = 10))) %>%  
  mutate(p_val = t_test$p.value) %>%  
  count(p_val <= 0.05)
```

# A tibble: 2 x 2

# Rowwise:

	`p_val <= 0.05`	n
	<lgl>	<int>
1	FALSE	76
2	TRUE	924

Power now *too big*.

## Try again

sample size between 15 and 50, say  $n = 30$ :

```
tibble(sim = 1:1000) %>%  
  rowwise() %>%  
  mutate(my_sample = list(rnorm(30, 8, 4))) %>%  
  mutate(t_test = list(t.test(my_sample, mu = 10))) %>%  
  mutate(p_val = t_test$p.value) %>%  
  count(p_val <= 0.05)
```

# A tibble: 2 x 2

# Rowwise:

	`p_val <= 0.05`	n
	<lgl>	<int>
1	FALSE	252
2	TRUE	748

Now a little too small, hence right answer between 30 and 50, closer to 30.

## One last try ( $n = 35$ )

```
tibble(sim = 1:1000) %>%  
  rowwise() %>%  
  mutate(my_sample = list(rnorm(35, 8, 4))) %>%  
  mutate(t_test = list(t.test(my_sample, mu = 10))) %>%  
  mutate(p_val = t_test$p.value) %>%  
  count(p_val <= 0.05)
```

# A tibble: 2 x 2

# Rowwise:

	`p_val <= 0.05`	n
	<lgl>	<int>
1	FALSE	179
2	TRUE	821

# But...

... simulation has randomness: limit to how close you can get.

Rule of thumb: with 1000 simulations, estimated power within 0.03 (3%).



# Power curves

- Rather than calculating power for one sample size, or sample size for one power, might want a picture of relationship between sample size and power.
- Or, likewise, picture of relationship between difference between true and null-hypothesis means and power.
- Called power curve.
- Build and plot it yourself.

## Building it 1/2

- If you feed `power.t.test` a collection (“vector”) of values, it will do calculation for each one.
- Do power for variety of sample sizes, from 10 to 100 in steps of 10:

```
ns <- seq(10,100,10)  
ns
```

```
[1] 10 20 30 40 50 60 70 80 90 100
```

## Building it 2/2

- Calculate powers:

```
ans<- power.t.test(n=ns, delta=10-8, sd=4, type="one.sample")
ans
```

One-sample t test power calculation

```
      n = 10, 20, 30, 40, 50, 60, 70, 80, 90, 100
delta = 2
sd = 4
sig.level = 0.05
power = 0.2928286, 0.5644829, 0.7539627, 0.8693979, 0.9338
alternative = two.sided
```

## Just the power

```
ans$power
```

```
[1] 0.2928286 0.5644829 0.7539627 0.8693979 0.9338976  
[6] 0.9677886 0.9847848 0.9929987 0.9968496 0.9986097
```

## Building a plot (1/2)

- Make a data frame out of the values to plot:

```
d <- tibble(n=ns, power=ans$power)
d
```

```
# A tibble: 10 x 2
```

	n	power
	<dbl>	<dbl>
1	10	0.293
2	20	0.564
3	30	0.754
4	40	0.869
5	50	0.934
6	60	0.968
7	70	0.985
8	80	0.993
9	90	0.997
10	100	0.999

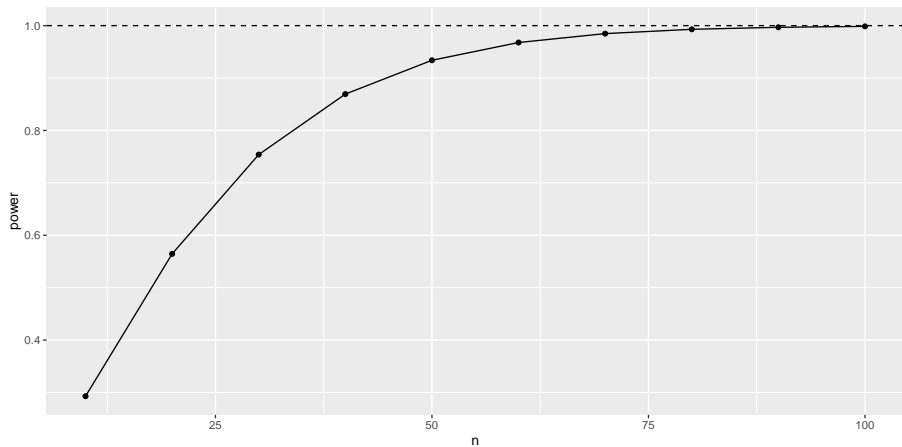
## Building a plot (2/2)

- Plot these as points joined by lines, and add horizontal line at 1 (maximum power):

```
g <- ggplot(d, aes(x = n, y = power)) + geom_point() +  
  geom_line() +  
  geom_hline(yintercept = 1, linetype = "dashed")
```

# The power curve

09



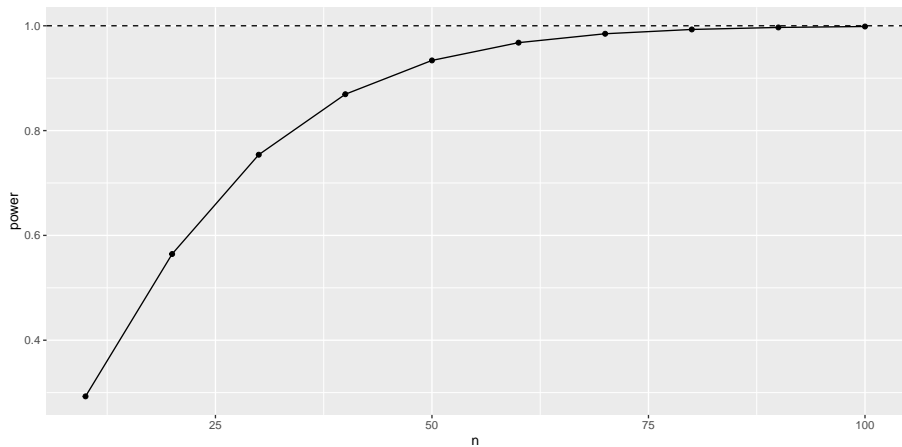
## Another way to do it:

```
tibble(n=ns) %>% rowwise() %>%  
  mutate(power_output =  
    list(power.t.test(n = n, delta = 10-8, sd = 4,  
                      type = "one.sample"))) %>%  
  mutate(power = power_output$power) %>%  
  ggplot(aes(x=n, y=power)) + geom_point() + geom_line() +  
    geom_hline(yintercept=1, linetype="dashed") -> g2
```



# The power curve done the other way

g2



## Power curves for means

- Can also investigate power as it depends on what the true mean is (the farther from null mean 10, the higher the power will be).
- Investigate for two different sample sizes, 15 and 30.
- First make all combos of mean and sample size:

```
means <- seq(6,10,0.5)
means
```

```
[1] 6.0 6.5 7.0 7.5 8.0 8.5 9.0 9.5 10.0
```

```
ns <- c(15,30)
ns
```

```
[1] 15 30
```

```
combos <- crossing(mean=means, n=ns)
```

# The combos

combos

```
# A tibble: 18 x 2
```

	mean	n
	<dbl>	<dbl>
1	6	15
2	6	30
3	6.5	15
4	6.5	30
5	7	15
6	7	30
7	7.5	15
8	7.5	30
9	8	15
10	8	30
11	8.5	15
12	8.5	30
13	9	15
14	9	30
15	9.5	15
16	9.5	30
17	10	15
18	10	30

## Calculate and plot

- Calculate the powers, carefully:

```
ans <- with(combos, power.t.test(n=n, delta=10-mean, sd=4,  
                                type="one.sample"))  
ans$power
```

```
[1] 0.94908647 0.99956360 0.88277128 0.99619287  
[5] 0.77070660 0.97770385 0.61513033 0.91115700  
[9] 0.43784659 0.75396272 0.27216777 0.51028173  
[13] 0.14530058 0.26245348 0.06577280 0.09719303  
[17] 0.02500000 0.02500000
```

# Make a data frame to plot

pulling things from the right places:

```
d <- tibble(n=factor(combos$n), mean=combos$mean,  
            power=ans$power)
```

d

# A tibble: 18 x 3

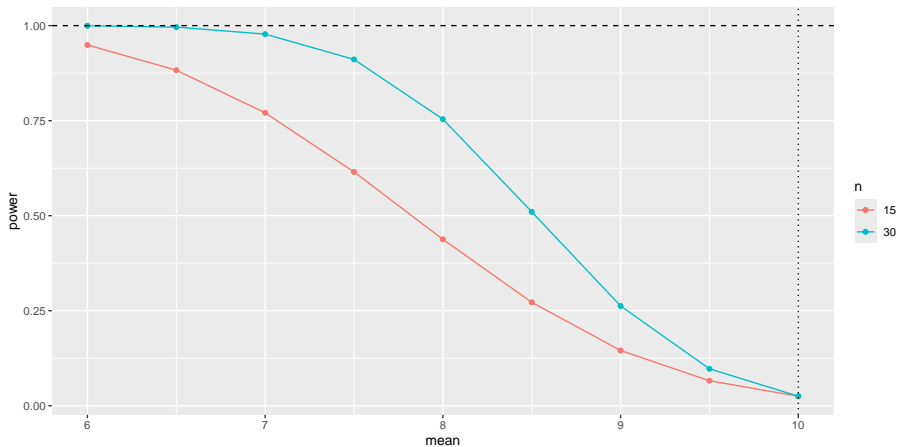
	n	mean	power
	<fct>	<dbl>	<dbl>
1	15	6	0.949
2	30	6	1.00
3	15	6.5	0.883
4	30	6.5	0.996
5	15	7	0.771
6	30	7	0.978
7	15	7.5	0.615
8	30	7.5	0.911
9	15	8	0.438

then make the plot:

```
g <- ggplot(d, aes(x = mean, y = power, colour = n)) +  
  geom_point() + geom_line() +  
  geom_hline(yintercept = 1, linetype = "dashed") +  
  geom_vline(xintercept = 10, linetype = "dotted")
```

# The power curves

g



## Comments

- When mean=10, that is, the true mean equals the null mean,  $H_0$  is actually true, and the probability of rejecting it then is  $\alpha = 0.05$ .
- As the null gets more wrong (mean decreases), it becomes easier to correctly reject it.
- The blue power curve is above the red one for any mean  $< 10$ , meaning that no matter how wrong  $H_0$  is, you always have a greater chance of correctly rejecting it with a larger sample size.
- Previously, we had  $H_0 : \mu = 10$  and a true  $\mu = 8$ , so a mean of 8 produces power 0.42 and 0.80 as shown on the graph.
- With  $n = 30$ , a true mean that is less than about 7 is almost certain to be correctly rejected. (With  $n = 15$ , the true mean needs to be less than 6.)



## Two-sample power

- For kids learning to read, had sample sizes of 22 (approx) in each group
- and these group SDs:

```
kids %>% group_by(group) %>%  
  summarize(n=n(), s=sd(score))
```

```
# A tibble: 2 x 3  
  group      n      s  
  <chr> <int> <dbl>  
1 c      23  17.1  
2 t      21  11.0
```

## Setting up

- suppose a 5-point improvement in reading score was considered important (on this scale)
- in a 2-sample test,  $\mu_0$ (difference of) mean is zero, so  $\delta$  is true difference in means
- what is power for these sample sizes, and what sample size would be needed to get power up to 0.80?
- SD in both groups has to be same in `power.t.test`, so take as 14.

## Calculating power for sample size 22 (per group)

```
power.t.test(n=22, delta=5, sd=14, type="two.sample",  
             alternative="one.sided")
```

Two-sample t test power calculation

```
      n = 22  
    delta = 5  
      sd = 14  
sig.level = 0.05  
  power = 0.3158199  
alternative = one.sided
```

NOTE: n is number in *each* group

## sample size for power 0.8

```
power.t.test(power=0.80, delta=5, sd=14, type="two.sample",  
             alternative="one.sided")
```

Two-sample t test power calculation

```
      n = 97.62598  
delta = 5  
    sd = 14  
sig.level = 0.05  
  power = 0.8  
alternative = one.sided
```

NOTE: n is number in *each* group

# Comments

- The power for the sample sizes we have is very small (to detect a 5-point increase).
- To get power 0.80, we need 98 kids in *each* group!