Statistical Inference: power

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

Power

- Suppose $H_0 : \theta = 10$, $Ha : \theta \neq 10$ for some parameter θ.
- Suppose H_0 wrong. What does that say about θ?
- Not much. Could have $\theta = 11$ or $\theta = 8$ or $\theta = 496$. In each case, H_0 wrong.
- How likely a type II error is depends on what θ is:
  - If $\theta = 496$, should be able to reject $H_0 : \theta = 10$ even for small sample, so β should be small (power large).
  - If $\theta = 11$, might have hard time rejecting H_0 even with large sample, so β would be larger (power smaller).
- Power depends on true parameter value, and on sample size.
- So we play "what if": "if θ 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 μ 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 σ 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
##  [5]  8.852952 10.835874  3.686684 11.165242
##  [9]  8.016188 12.383518  1.378099  3.172503
## [13] 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

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

```
## [1] 0.0815652
```

Run this lots of times

- Two steps:
  - Generate a bunch of random samples
  - extract the P-value for the t-test from each
- without a loop!
- Use `rerun` to generate the random samples
- Use `map` to run the test on each random sample
- Use `map_dbl` to pull out the P-value for each test
- Count up how many of the P-values are 0.05 or less.

In code

```
rerun(1000, rnorm(15, 8, 4)) %>%
  map( ~ t.test(., mu = 10)) %>%
  map_dbl("p.value") %>%
  enframe(value="pvals") %>%
  count(pvals <= 0.05)
```

pvals <= 0.05	n
FALSE	578
TRUE	422

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

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`. delta difference between null and true mean:

```
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
`power.t.test`	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 ± 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 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).

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

- 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
```

- Calculate powers:

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

```
##  [1] 0.2928286 0.5644829 0.7539627 0.8693979
##  [5] 0.9338976 0.9677886 0.9847848 0.9929987
##  [9] 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
```

n	power
10	0.2928286
20	0.5644829
30	0.7539627
40	0.8693979
50	0.9338976
60	0.9677886
70	0.9847848
80	0.9929987
90	0.9968496
100	0.9986097

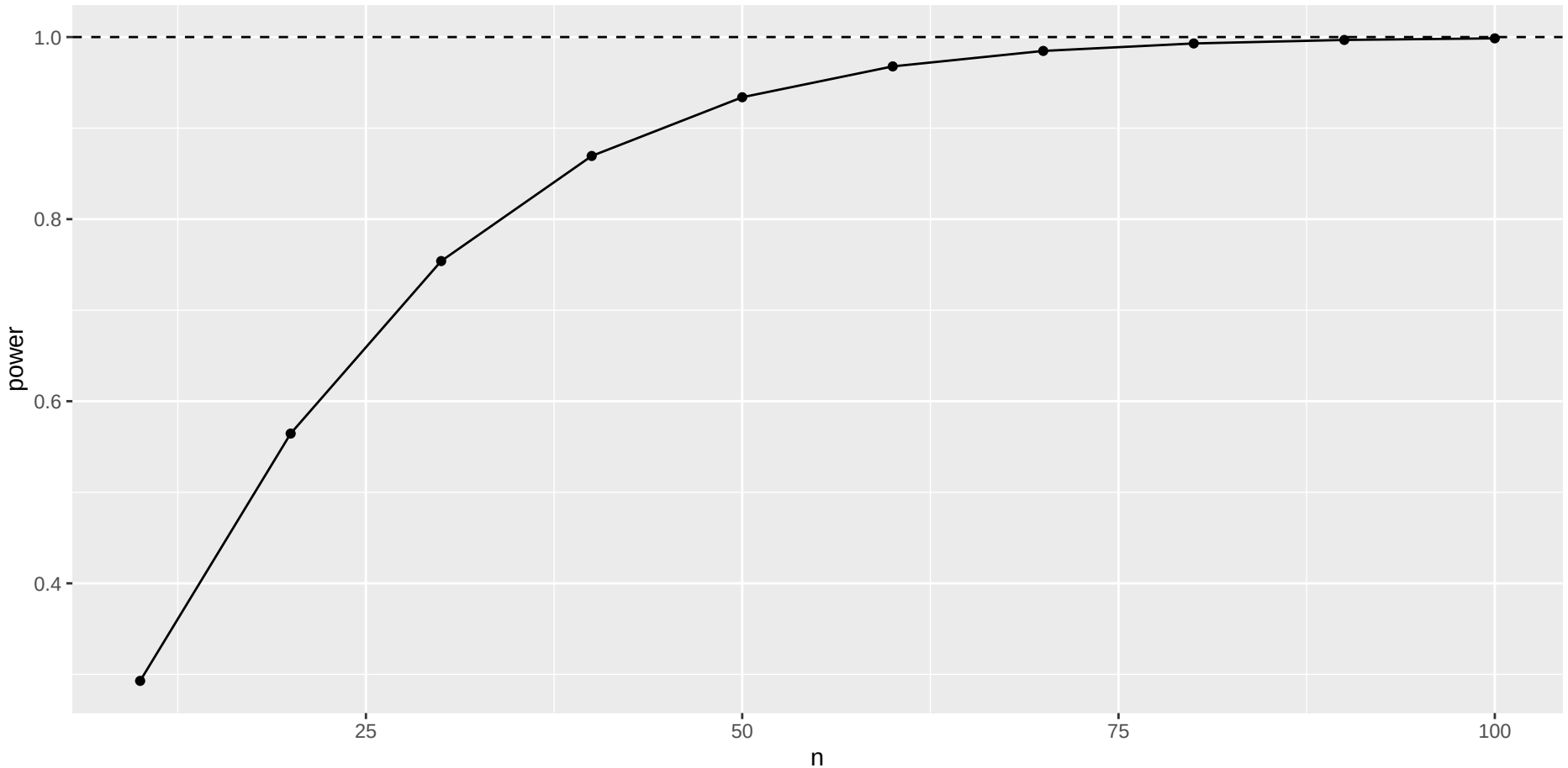
Building a plot (2/2)

- 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")
```

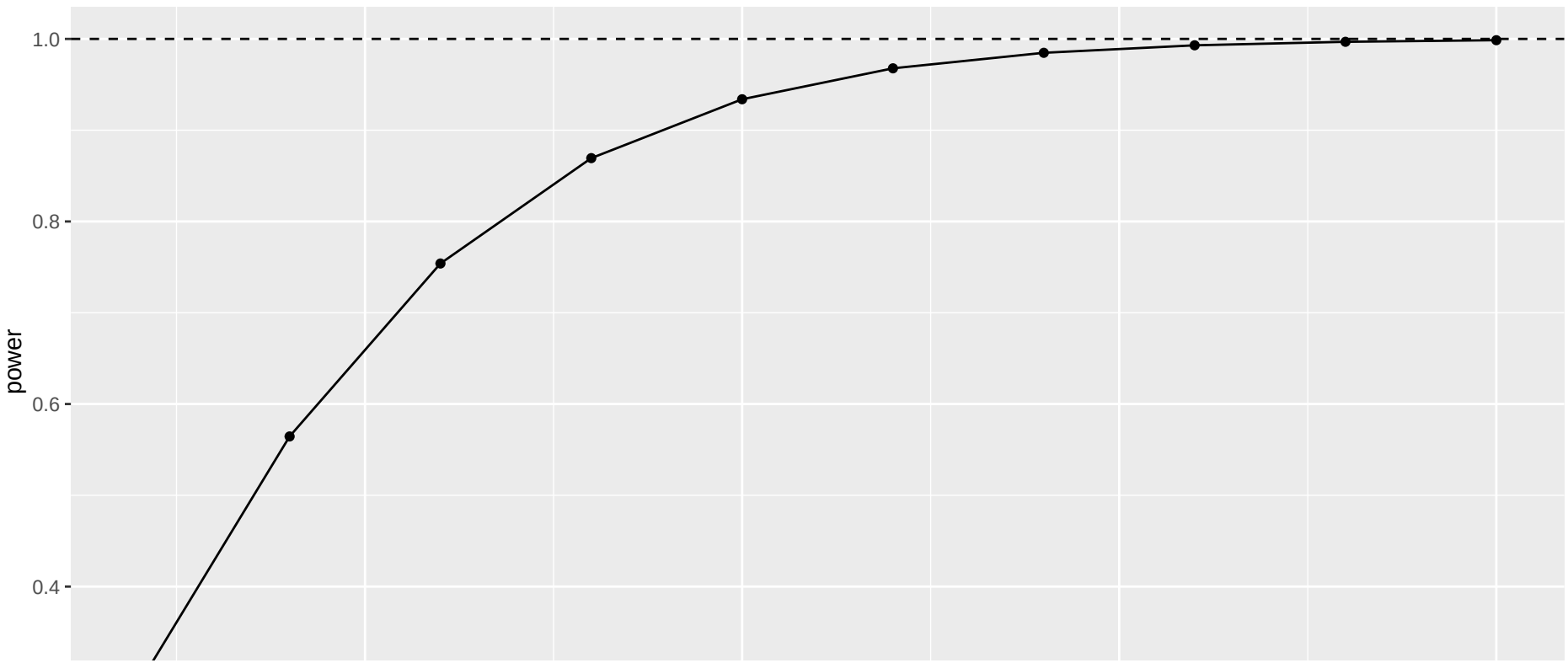
The power curve

```
gg
```



Another way to do it:

```
tibble(n=ns) %>%
  mutate(power_output=map(n, ~power.t.test(n=., delta=10-8, sd
  mutate(power=map_dbl(power_output, "power")) %>%
  ggplot(aes(x=n, y=power)) + geom_point() + geom_line() +
    geom_hline(yintercept=1, linetype="dashed")
```



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
```

mean	n
6.0	15
6.0	30
6.5	15
6.5	30
7.0	15
7.0	30
7.5	15
7.5	30
8.0	15
8.0	30
8.5	15
8.5	30
9.0	15

Calculate and plot

- Calculate the powers, carefully:

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

```
##
##      One-sample t test power calculation
##
##               n = 15, 30, 15, 30, 15, 30, 15, 30, 15, 30, 1
##           delta = 4.0, 4.0, 3.5, 3.5, 3.0, 3.0, 2.5, 2.5, 2
##              sd = 4
##       sig.level = 0.05
##           power = 0.94908647, 0.99956360, 0.88277128, 0.996
##     alternative = two.sided
```

More of what's in the output

```
names(ans)
```

```
## [1] "n"           "delta"       "sd"
## [4] "sig.level"   "power"       "alternative"
## [7] "note"        "method"
```

```
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
```

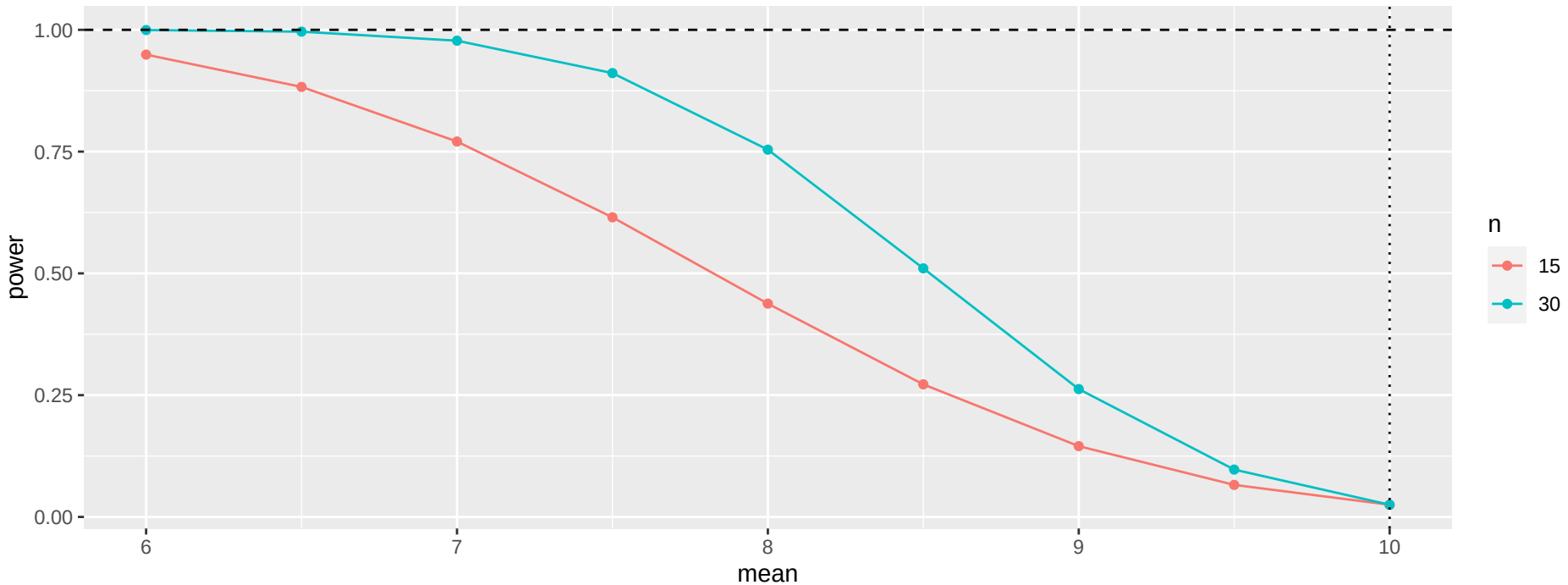
n	mean	power
15	6.0	0.9490865
30	6.0	0.9995636
15	6.5	0.8827713
30	6.5	0.9961929
15	7.0	0.7707066
30	7.0	0.9777038
15	7.5	0.6151303
30	7.5	0.9111570
15	8.0	0.4378466
30	8.0	0.7539627

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.)

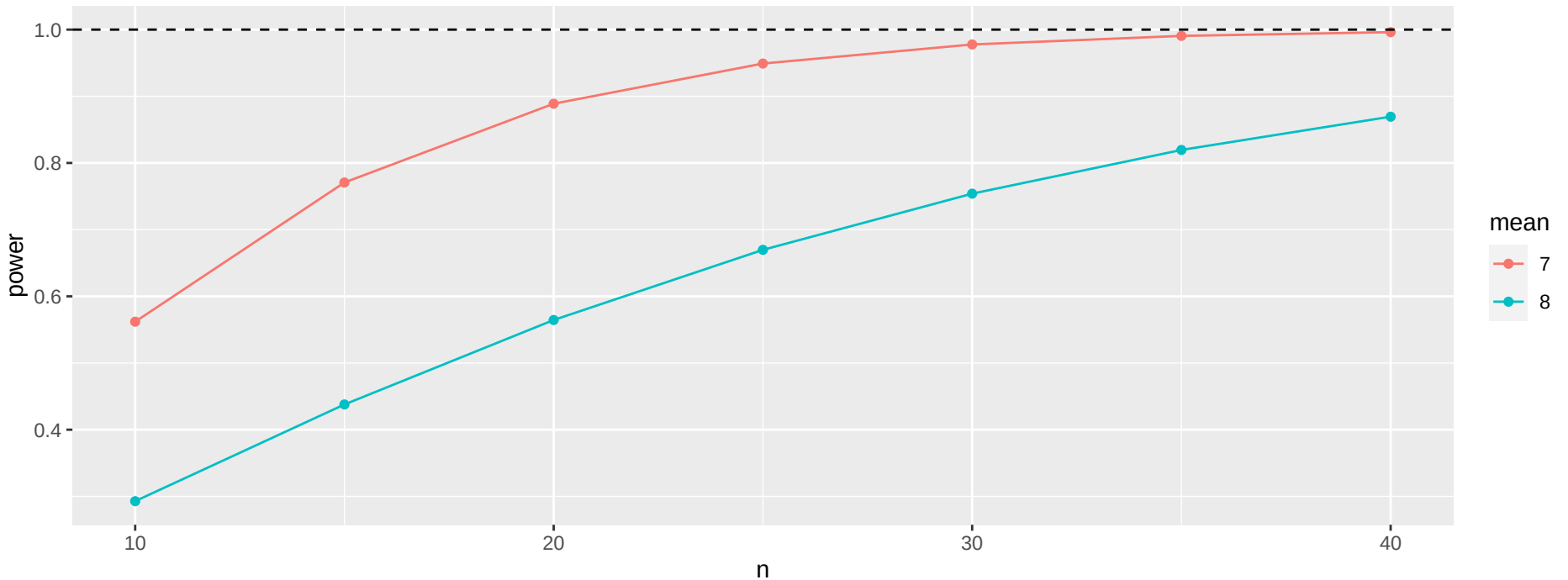
Power by sample size for means 7 and 8

Similar procedure to before:

```
means=c(7, 8)
ns=seq(10, 40, 5)
combos=crossing(mean=means, n=ns)
ans=with(combos, power.t.test(n=n, delta=10-mean, sd=4,
                              type="one.sample"))
d=tibble(mean=factor(combos$mean), n=combos$n,
         power=ans$power)
g=ggplot(d, aes(x=n, y=power, colour=mean)) +
  geom_point() + geom_line() +
  geom_hline(yintercept=1,linetype="dashed")
```

The power curves

```
g
```



Two-sample power

```
## Parsed with column specification:
## cols(
##   group = col_character(),
##   score = col_double()
## )
```

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

```
kids
```

group	score
t	24
t	61
t	59
t	46

Setting up

- suppose a 5-point improvement in reading score was considered important (on this scale)
- in a 2-sample test, null (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!