In-Class Exercise: Power Analysis

XDASI Fall 2021

11/4/2021

Contents

Example

Aho, Ex. 6.10 (also see Fig. 6.7)

We will set up this example and go through it together in class.

Manual power calculation

First, compute the power by hand:

```r
# ==================================================== #
# set up variables

effect.size = -7  # effect size = Exp(X) under H_A
n = 200
sigma = 45
alpha = 0.05
type = "one.sample" # one or two sample
alt = "one.sided"  # one- or two-sided

# set Exp(X) = 0 under null H_o

# critical value (z*) for lower-tail test at alpha=0.05
z.crit = qnorm(alpha)  # -1.644854

# check alpha using standard normal distribution
pnorm(0,abs(z.crit),lower.tail=T)  # 0.05
## [1] 0.05

# compute SEM for sample size
sem = sigma/sqrt(n)
```

```r
sem
## [1] 3.181981

# ==================================================== #
# manual power calculation

# compute power using area under the curve for H_A

# get value of critical x at z.crit for H_A
# want P( X.bar  <= z.crit * sem)
# percent difference for lower-tail significance
x.crit = z.crit*sem  # x-value at critical z-score
x.crit
## [1] -5.233892

# Expected X.bar (pop. mean) under H_A is (mu_o - mu_A): Exp(X) = -7
pwr = pnorm(x.crit, mean = effect.size, sd = sem)  # power = 0.71
pwr
## [1] 0.7105643

# check z-score for H_A at expected power
qnorm(pwr, effect.size, sem)  # alpha = P(X.bar <= -5.24)
## [1] -5.233892
```

Compute power in R

Given any 4 of the 5 variables that go into the power equation, we can use `power.t.test()` to compute the missing value. Since n is large, we could also use the `power.z.test()` command from the `asbio` package. These give slightly different results, as the t-test is a bit more conservative. (They also use different names for their arguments, and the objects the produce are also different.)

NOTE: *Effect size used for these functions should be given as a positive number, otherwise these functions will not work as expected.*

```r
# =================================================== #
# provide expected effect size as a positive number
power.t.test(n, delta = abs(effect.size), sd = sigma, sig.level = alpha,
             type="one.sample", alternative="one.sided", strict=T)
```

```
##
##      One-sample t test power calculation
##
##               n = 200
##           delta = 7
##              sd = 45
##       sig.level = 0.05
##           power = 0.7079982
##     alternative = one.sided
```

```r
# note that arguments for this command differ
power.z.test(n, effect = abs(effect.size), sigma = sigma,
             alpha = alpha, test="one.tail", strict=T)
```

```
## $sigma
## [1] 45
##
## $n
## [1] 200
##
## $power
## [1] 0.7105643
##
## $alpha
## [1] 0.05
##
## $effect
## [1] 7
##
## $test
## [1] "one.tail"
```

What if you change different variables that influence power?

- Increase effect size => increase power (reduce Type II error)
- Increase sample size => increase power (reduce Type II error)
- Raise α => lower stringency (increase Type I error)

We can compute the new power by hand, or use the `power.z.test()` command:

```
# ==================================================== #
# increase effect size
# ==================================================== #
# what happens if E = -8? => increase power
# (keep alpha the same)
pnorm(x.crit, effect.size - 1, sem)  # power = 0.81
```

```
## [1] 0.8076595
```

```
power.z.test(n, effect = -(effect.size-1), sigma = sigma,
             alpha = alpha, test="one.tail", strict=T)
```

```
## $sigma
## [1] 45
##
## $n
## [1] 200
##
## $power
## [1] 0.8076595
##
## $alpha
## [1] 0.05
##
## $effect
## [1] 8
```

```
##
## $test
## [1] "one.tail"
```

```
# ================================================== #
# increase sample size
# ================================================== #
# what if sample size = 300? => more power for same E
# (keep alpha the same)
n = 300

# get x-bar and SEM
sem = sigma / sqrt(n)
sem
```

```
## [1] 2.598076
```

```
x.crit = qnorm(0.05)*sem
x.crit
```

```
## [1] -4.273455
```

```
# power
pnorm(x.crit, effect.size, sem)  # power = 0.85
```

```
## [1] 0.8530139
```

```
power.z.test(n, effect = -effect.size, sigma = sigma,
             alpha = alpha, test="one.tail", strict=T)
```

```
## $sigma
## [1] 45
##
## $n
## [1] 300
##
## $power
## [1] 0.8530139
##
## $alpha
## [1] 0.05
##
## $effect
## [1] 7
##
## $test
## [1] "one.tail"
```

```
# ================================================== #
# relax stringency: raise alpha
# ================================================== #
```

```r
# raising alpha increases Type I error
# what happens to power? => power goes down
alpha = 0.2
z.crit = qnorm(alpha)  # -0.842

# check alpha2 using standard normal distribution
pnorm(0,abs(z.crit),lower.tail=T)
```

```
## [1] 0.2
```

```r
x.crit = z.crit*sem
x.crit
```

```
## [1] -2.186596
```

```r
pwr = pnorm(x.crit, mean = effect.size, sd = sem)  # power = 0.913
pwr
```

```
## [1] 0.9680359
```

```r
qnorm(pwr, effect.size, sem) # check power
```

```
## [1] -2.186596
```

```r
# power is the same with the z-test power function
power.z.test(n, effect = -effect.size, sigma = sigma,
             alpha = alpha, test="one.tail", strict=T)
```

```
## $sigma
## [1] 45
##
## $n
## [1] 300
##
## $power
## [1] 0.9680359
##
## $alpha
## [1] 0.2
##
## $effect
## [1] 7
##
## $test
## [1] "one.tail"
```

Design for a targeted power

What if you want to design the experiment for power = 0.8, for the same sample size and effect size? What is the Type II error? What happens to the Type I error?

```r
# ================================================== #
# increase desired power to 0.8
# ================================================== #
# if raise desired power without increasing effect size,
#   => alpha goes up (less stringent)
x.bar2 = qnorm(0.8, effect.size, sem) # effect size -4.32
x.bar2
```

```
## [1] -4.813404
```

```r
# what significance level is this?
alpha2 = x.bar2 / sem
alpha2
```

```
## [1] -1.85268
```

```r
pnorm(0,abs(alpha2),lower.tail=T)  # 0.087
```

```
## [1] 0.03196412
```


```r
# what significance level is this?
alpha2 = x.bar2 / sem
alpha2
```

```
## [1] -1.85268
```

```r
pnorm(0,abs(alpha2),lower.tail=T)  # 0.087
```

```
## [1] 0.03196412
```


```r
# ================================================== #
# using power.t.test command
# now supply power and ask what new alpha is
power.t.test(n, delta = -effect.size, sd = sigma,
             sig.level = NULL, power = 0.8,
            type="one.sample", alternative="one.sided", strict=T)
```

```
## Warning in pt(qt(sig.level/tside, nu, lower.tail = FALSE), nu, ncp = sqrt(n/
## tsample) * : full precision may not have been achieved in 'pnt{final}'
```

```
##
##      One-sample t test power calculation
##
##               n = 300
##           delta = 7
##              sd = 45
##       sig.level = 0.03251671
##           power = 0.8
##     alternative = one.sided
```

```
# gives alpha = 0.088
```