Stat 5421 Lecture Notes: Statistical Inference for the Binomial Distribution

Charles J. Geyer

September 25, 2020

1 Data

In our example there are two successes in 25 trials.

```r
x <- 2
n <- 25
```

2 Maximum Likelihood Estimator

The usual estimator of the parameter p is $\hat{p} = x/n$. This turns out to also be the maximum likelihood estimator.

```r
phat <- x / n
```

3 Log Likelihood

We can plot the log likelihood function using the following code

```r
logl <- function(p) {
    result <- x * log(p) + (n - x) * log(1 - p)
    result[is.na(result)] <- 0
    return(result)
}
curve(logl, from = 0.0001, to = 0.99, n = 1001,
    xlab = "p", ylab = "log likelihood",
    ylim = logl(phat) + c(-10, 0))
abline(v = phat, lty = "dashed")
```

The dashed vertical line shows where the MLE is, and it does appear to be where the log likelihood is maximized. The log likelihood goes to minus infinity as $p \to 0$ or $p \to 1$. Except when $x = 0$ the log likelihood increases to 0 as $p \to 0$, and when $x = n$ it increases to 0 as $p \to 1$. Thus we cannot try to draw the curve from 0 to 1 but rather from a little bit above 0 to a little bit below 1. But that gives us a plot in which it is hard to see what is going on.

Hence the `ylim` optional argument to R function `curve`. It will turn out that the only interesting part of the log likelihood is the region near the maximum. Hence we include in our plot only the part of the curve in which the log likelihood is within 10 of the maximum. The 10 was pulled out of the air. We don't really want to get scientific about this yet (but do in the section on likelihood-based confidence intervals below). Also it

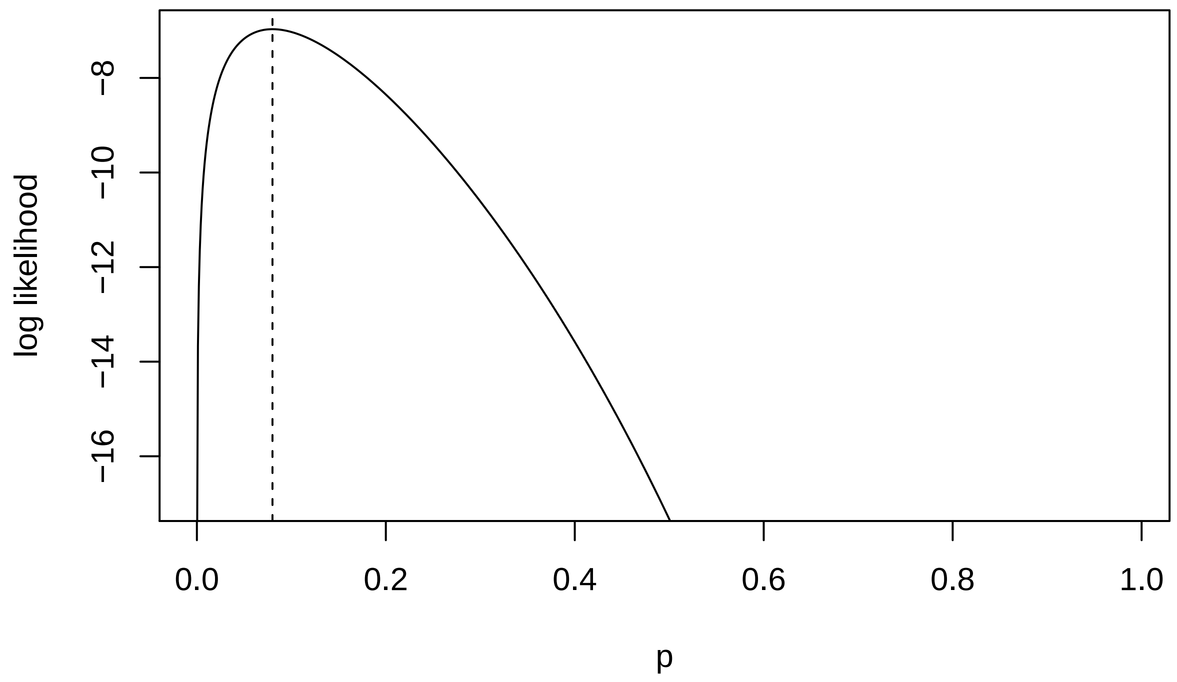

Figure 1: Log Likelihood

really doesn't matter, since we are just using this plot to get some idea what is going on. We don't use this plot for statistical inference.

4 Confidence Level

Set the confidence level. It is best programming practice to never hard code numbers like this, that is, the number 0.95 should only occur in your document once where it is used to initialize a variable. Then use that variable elsewhere.

If there is ever a need to change this (to use say 0.90 for the confidence level), then it only need be changed in one place and will be consistently used everywhere.

```
conf.level <- 0.95
```

5 Wald, Wilks, Rao

Section 1.3.3 in Agresti discusses the three main strategies for constructing hypothesis tests. And each kind of hypothesis goes with a confidence interval that is derived by inverting the test. So there are the same three strategies for confidence intervals.

The three strategies are

- Wald test
- Wilks test, also called likelihood ratio test
- Rao test, also called score test and Lagrange multiplier test (this last name is used mostly by economists).

All of these procedures are asymptotically equivalent under the usual asymptotics of maximum likelihood. Hence none is better than the others for sufficiently large sample size. But they may be quite different for small sample sizes.

5.1 Wald Confidence Interval

This used to be the standard taught in intro stats, maybe it still is in many such courses. These are the only intervals of the type

$$\text{point estimate} \pm \text{standard error}$$

```
phat + c(-1, 1) * qnorm((1 + conf.level) / 2) * sqrt(phat * (1 - phat) / n)
```

```
## [1] -0.02634498  0.18634498
```

We know $0 \leq p \leq 1$ but this interval makes no use of that information, giving a lower bound that is negative.

This may look ridiculous, but is not wrong. If we know that $0 \leq p$, then it is true that $-0.026345 \leq p$ too.

Agresti (Section 1.4) points out that this interval gives ridiculous zero-width confidence intervals when $\hat{p}$ is equal to zero or one (when the data x is equal to zero or n). However, the Wald interval can be repaired by using a different procedure (Geyer, 2009, *Electronic Journal of Statistics*, **3**, 259–289) that was illustrated on the web page discussing coverage of confidence intervals. So that is not a reason not to use Wald intervals so long as one knows how to use this modification when the MLE is on the boundary of the parameter space.

6 Score Interval

Some intro stats books now teach this. In particular, Agresti's intro stats book teaches this. Some teach a dumbed down approximation to this. Since we are using R, we may as well do the right thing, not the dumbed down version suitable for hand calculation.

```
prop.test(x, n, conf.level = conf.level, correct = FALSE)
```

```
##
##  1-sample proportions test without continuity correction
##
## data:  x out of n, null probability 0.5
## X-squared = 17.64, df = 1, p-value = 2.669e-05
## alternative hypothesis: true p is not equal to 0.5
## 95 percent confidence interval:
##  0.0222204 0.2496611
## sample estimates:
##    p
## 0.08
```

The "`correct = FALSE`" is just bizarre.

Does that mean we don't want the correct answer? No. The R statement `help(prop.test)` explains that it means we do not want to use "continuity correction". And we don't. No authority recommends what `prop.test` does by default. Agresti, Section 1.4.2, recommends `correct = FALSE`. Brown, Cai and DasGupta (*Statistical Science*, 2005, **20**, pp. 375–379) criticize Geyer and Meeden (*Statistical Science*, 2005, **20**, pp. 358–366) for using `prop.test` with `correct = TRUE`, providing plots of coverage probability for with `correct = FALSE` and `correct = TRUE` to show this. AFAIK no authority defends `correct = TRUE`, even `help(prop.test)` cites no such authority. They do cite Newcombe (*Statistics in Medicine*, 1998, **17**, 857–872) where `correct = FALSE` is his method 3 and `correct = TRUE` is (apparently) his method 4, but Newcombe's simulations do not show that `correct = TRUE` is better than `correct = FALSE` and his conclusion does not recommend `correct = TRUE` over `correct = FALSE` (but he does not pick a particular method to recommend).

For a full derivation of the score interval and a check that `correct = FALSE` actually calculates it see http://www.stat.umn.edu/geyer/5102/slides/s2.pdf, slides 113–116 and 123. (Hmmmm. That does not actually seem to check. We'll check here.)

```
crit <- qnorm((1 + conf.level) / 2)
foo <- (phat + crit^2 / (2 * n) + c(-1, 1) * crit *
    sqrt(phat * (1 - phat) / n + crit^2 / (4 * n^2))) / (1 + crit^2 / n)
foo
```

```
## [1] 0.0222204 0.2496611
```

```
bar <- prop.test(x, n, conf.level = conf.level, correct = FALSE)
names(bar)
```

```
## [1] "statistic"   "parameter"   "p.value"     "estimate"    "null.value"
## [6] "conf.int"    "alternative" "method"      "data.name"
```

```
bar$conf.int
```

```
## [1] 0.0222204 0.2496611
## attr(,"conf.level")
## [1] 0.95
```

```
all.equal(bar$conf.int, foo, check.attributes = FALSE)
```

```
## [1] TRUE
```

7 Likelihood Interval

Confidence interval that is a level set of the log likelihood. This is too esoteric for intro stats.

```
crit <- qchisq(conf.level, df = 1)
fred <- function(p) 2 * (logl(phat) - logl(p)) - crit
tol <- sqrt(.Machine$double.eps)
if (phat == 0) {
    low <- 0
} else {
    low <- uniroot(fred, lower = 0, upper = phat, tol = tol)$root
}
if (phat == 1) {
    hig <- 1
} else {
    hig <- uniroot(fred, lower = phat, upper = 1, tol = tol)$root
}
c(low, hig)
```

```
## [1] 0.01376568 0.22711456
```

We can check we have done the right thing by redoing our log likelihood plot.

```
sally <- function(p) 2 * logl(p)
curve(sally, from = 0.0001, to = 0.99, n = 1001,
    xlab = "p", ylab = "two times log likelihood",
    ylim = sally(phat) + c(-6, 0))
abline(h = sally(phat) - crit, lty = "dashed")
```

```r
abline(v = low, lty = "dashed")
abline(v = hig, lty = "dashed")
```

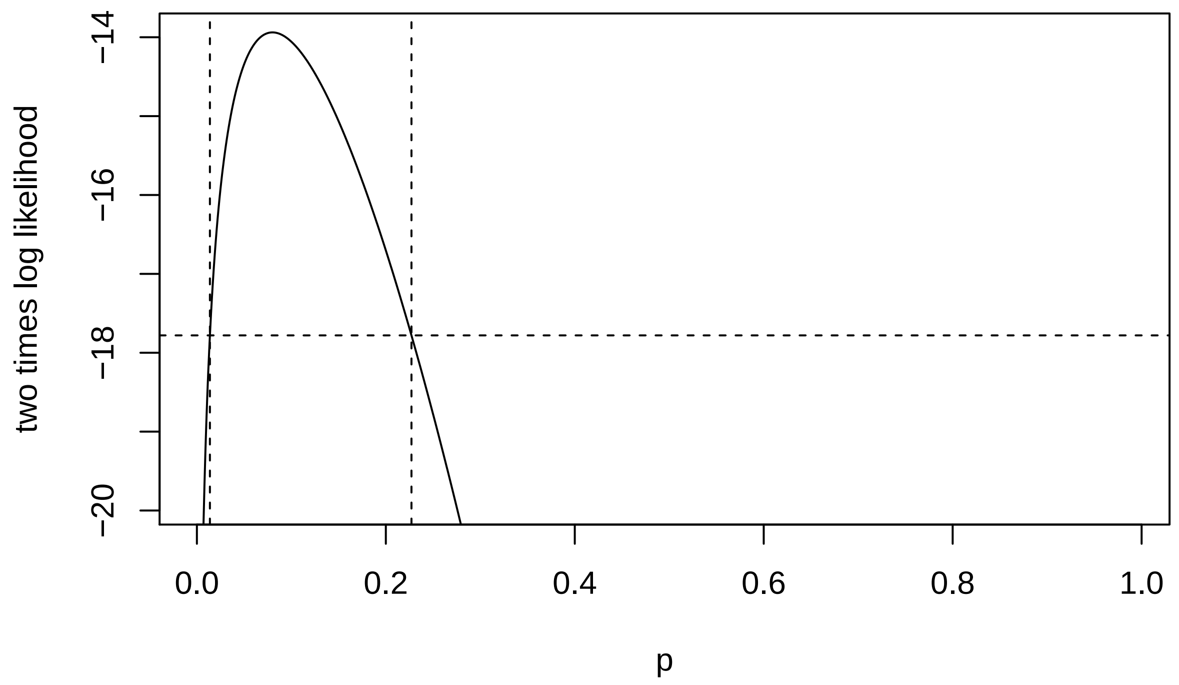

Figure 2: Likelihood-Based Confidence Interval

We can see that the vertical dashed lines, which are the endpoints of the likelihood-based confidence interval, do indeed intersect the graph of two times the log likelihood `crit` down from the maximum, where `crit` is the critical value derived from the chi-squared distribution.

Note that this figure has even less of a vertical range than the preceding one. This does say that the interesting part of the picture is captured by both graphs where now “interesting” means the part relevant to a 95% confidence interval.

8 Likelihood Interval using MASS Package

We can also do this with R function `confint` in R package `MASS`. Except this function botches the calculation when $x = 0$ or $x = n$. Our calculation above always does the right thing.

```r
library(MASS)
gout <- glm(rbind(c(x, n - x)) ~ 1, family = binomial)
cout <- confint(gout, level = conf.level)
```

```
## Waiting for profiling to be done...
```

```r
cout
```

```
##     2.5 %    97.5 %
## -4.272091 -1.225066
```

```r
fred <- 1 / (1 + exp(- cout))
fred
```

```
##      2.5 %     97.5 %
## 0.01376058 0.22704616
```

```r
all.equal(c(low, hig), fred, check.attributes = FALSE, tolerance = 4e-4)
```

```
## [1] TRUE
```

9 Five Intervals Compared

```r
phat + c(-1, 1) * qnorm((1 + conf.level) / 2) * sqrt(phat * (1 - phat) / n)
```

```
## [1] -0.02634498  0.18634498
```

```r
as.vector(bar$conf.int)
```

```
## [1] 0.0222204 0.2496611
```

```r
c(low, hig)
```

```
## [1] 0.01376568 0.22711456
```

As can be seen, the intervals are rather different. If we were to go back to the top and change the data to $x = 200$ and $n = 2500$ (both 100 times what they were before), then the intervals are quite close to each other (not to mention a lot shorter).

The web page discussing coverage of confidence intervals discusses two more intervals

```r
# Clopper-Pearson Binomial Confidence Intervals
binom.test(x, n, conf.level = conf.level)
```

```
##
##  Exact binomial test
##
## data:  x and n
## number of successes = 2, number of trials = 25, p-value = 1.943e-05
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.00983959 0.26030584
## sample estimates:
## probability of success
##                   0.08
```

```r
# Variance Stabilized Binomial Confidence Intervals
# http://www.stat.umn.edu/geyer/5102/slides/s2.pdf#page=120
thetahat <- asin(2 * phat - 1)
crit.val <- qnorm((1 + conf.level) / 2)
(1 + sin(thetahat + c(-1,1) * crit.val / sqrt(n))) / 2
```

```
## [1] 0.008214812 0.215499536
```

10 Hypothesis tests

11 Data

For some reason, we are going to use different data in the hypothesis tests section, presumably because with the small sample size before there was no power to reject almost all null hypotheses.

```
x <- 1300
n <- 2500
```

12 MLE

```
phat <- x / n
```

13 Null Hypothesis

```
p0 <- 1 / 2
```

14 Score Test

Now this is the standard test for proportions taught in intro stats (regardless of which interval is taught. We will do an upper-tail test. Test statistic and P-value.

```
tstat <- (x - n * p0) / sqrt(n * p0 * (1 - p0))
tstat
```

```
## [1] 2
```

```
pnorm(tstat, lower.tail = FALSE)
```

```
## [1] 0.02275013
```

This is an asymptotic procedure, only approximately correct for large sample sizes.

15 Exact Test

Here is an "exact" test (really only conservative-exact, the P-value is guaranteed to understate the statistical significance).

P-value for exact test

```
pbinom(x - 1, n, p0, lower.tail = FALSE)
```

```
## [1] 0.02384093
```

This should perhaps be standard in intro stats.

16 Fuzzy P-Value

This comes from Geyer and Meeden (*Statistical Science*, 2005, **20**, 358–387).

Here the P-value is considered to be uniformly distributed on interval

```
pbinom(c(x, x - 1), n, p0, lower.tail = FALSE)
```

```
## [1] 0.02168093 0.02384093
```

This test is truly exact (exact-exact rather than conservative-exact) in the sense that the probability $P \leq \alpha$ is equal to α for $0 \leq \alpha \leq 1$.

This test is what is actually comparable to an exact test with a continuous test statistic (like a t-test, for example).

17 Likelihood Ratio Test

For a one-tailed test we have to use the signed likelihood ratio test statistic. Here are the test statistic and P-value for this test.

```
tstat <- 2 * (logl(phat) - logl(p0))
tstat <- sqrt(tstat)
tstat <- tstat * sign(phat - p0)
tstat
```

```
## [1] 2.000267
```

```
pnorm(tstat, lower.tail = FALSE)
```

```
## [1] 0.02273573
```

This too is an asymptotic procedure, only approximately correct for large sample sizes. It is asymptotically equivalent to the score test. Both the test statistic and the P-value for these two tests (and the Wald test, which is next) will be nearly equal for large sample sizes.

18 Wald Test

Test statistic and P-value

```
tstat <- (x - n * p0) / sqrt(n * phat * (1 - phat))
tstat
```

```
## [1] 2.001602
```

```
pnorm(tstat, lower.tail = FALSE)
```

```
## [1] 0.02266378
```

This too is an asymptotic procedure, only approximately correct for large sample sizes. It is asymptotically equivalent to the score test and the likelihood ratio test. The test statistic and the P-value for these three tests will be nearly equal for large sample sizes.

This seems intuitively wrong. The P-value is calculated assuming p_0 is the true unknown parameter value (in general, assuming the null hypothesis is true). So if you know $p = p_0$, why not use that fact in doing the test? The score test and likelihood ratio test do; the Wald test doesn't. (This is related to the Wald test not needing the MLE in the null hypothesis. Here we have a point null hypothesis, so the MLE in the null hypothesis is p_0.) Hence no intro text recommends the Wald test for the binomial distribution.

Of course many textbooks recommend Wald tests in other situations, for example, those output by the R generic function `summary`.

19 Two-Tailed Test

Now we illustrate two-tailed tests for the same data.

19.1 Score Test

```
tstat <- (x - n * p0) / sqrt(n * p0 * (1 - p0))
tstat
```

```
## [1] 2
```

```
pnorm(abs(tstat), lower.tail = FALSE) * 2
```

```
## [1] 0.04550026
```

```
pnorm(- abs(tstat)) * 2
```

```
## [1] 0.04550026
```

```
pnorm(abs(tstat), lower.tail = FALSE) + pnorm(- abs(tstat))
```

```
## [1] 0.04550026
```

As can be seen the last three commands are three equivalent ways to calculate the P-value for the two-tailed test using the symmetry of the standard normal distribution.

19.2 Exact Test

Oops! There is no exact non-fuzzy two-tailed test.

19.3 Fuzzy Test

Fuzzy P-value for exact test (Geyer and Meeden, *Statistical Science*, 2005, **20**, 358–387). Now this is not simple, but there is an R function to do it in R package `ump`.

```
library(ump)
print(arpv.binom(x, n, p0, plot = FALSE))
```

```
## $alpha
## [1] 0.04336187 0.04336187 0.04768187
##
## $phi
## [1] 0.00000e+00 1.73133e-12 1.00000e+00
```

Here because the value of the cumulative distribution is so low at the middle knot, this is almost a uniform distribution. The fuzzy P-value is approximately uniformly distributed on the interval

```
aout <- arpv.binom(x, n, p0, plot = FALSE)
aout$alpha[2:3]
```

```
## [1] 0.04336187 0.04768187
```

This behavior is not the way this test always works. Sometimes the distribution of the fuzzy P-value is quite complicated. Here's an example, which is Figure 3 in Geyer and Meeden (*Statistical Science*, 2005, vol 20, pp. 358–387).

```
aout <- arpv.binom(10, 10, 0.7, plot = FALSE)
aout
```

```
## $alpha
## [1] 0.000000000 0.000019683 0.000433026 0.004290894 0.025294842 0.052384365
##
```

```
## $phi
## [1] 0.0000000000 0.0004877631 0.0102430249 0.0956015654 0.5204974118
## [6] 1.0000000000
```

We make the plot that is the aforementioned Figure 3 as follows.

```r
k <- length(aout$alpha)
alpha <- aout$alpha
pdf <- diff(aout$phi) / diff(aout$alpha)
plot(alpha, c(0, pdf), type = "n", xlab = expression(alpha),
    ylab = "probability density")
segments(alpha[-k], pdf, alpha[-1], pdf)
segments(alpha, c(0, pdf), alpha, c(pdf, 0), lty = "dashed")
```

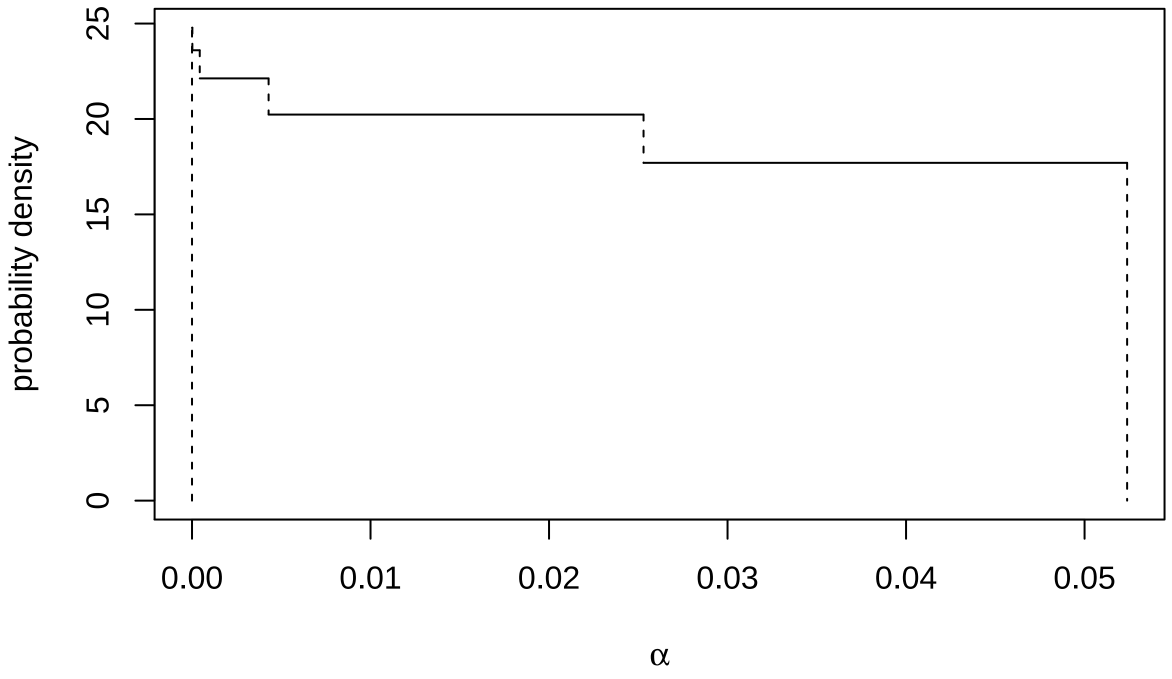

Figure 3: PDF of Fuzzy P-Value

The fuzzy *P*-value can be considered to be a random variable having the probability density function shown in the figure.

19.4 Likelihood Ratio Test

Test statistic and *P*-value

```r
tstat <- 2 * (logl(phat) - logl(p0))
tstat
```

```
## [1] 4.001067
```

```r
pchisq(tstat, df = 1, lower.tail = FALSE)
```

```
## [1] 0.04547146
```

19.5 Wald Test

Test statistic and P-value

```r
tstat <- (x - n * p0) / sqrt(n * phat * (1 - phat))
tstat
```

```
## [1] 2.001602
```

```r
pnorm(abs(tstat), lower.tail = FALSE) * 2
```

```
## [1] 0.04532756
```