Confidence intervals for binomial data: Simulations

Patrick Breheny

September 29, 2016

The purpose of today's lab is to carry out simulations to see what the coverage is for the various intervals for binomial data we have derived in class: the Clopper-Pearson interval, the Bayesian HPD interval, the Wald interval, and the score interval. We proved in class that the Clopper-Pearson interval must have at least 95% coverage, but is the coverage exactly 95%, or it is, say, 99%. The Wald and score intervals have approximate 95% coverage, but how close is the approximation? And what's the coverage of a Bayesian interval? I'm not entirely sure if we'll have time for all this in today's lab, but if we don't finish, we'll pick up next time where we left off.

Let's let π denote the true probability of success; we'll start off supposing that $\pi = 0.25$ and $n = 10$. Let's carry out the simulation. Typically, simulations involve the use of `for` loops. The syntax of `for` loops is pretty straightforward, and shown below. The idea is that we want to loop over a block of code (enclosed in curly brackets), with an index (`i` in the example below) that updates every time the loop is run again. Note that with the `rbinom` function, we can actually do the sampling outside of a `for` loop, but we still need the `for` loop to calculate the CI:

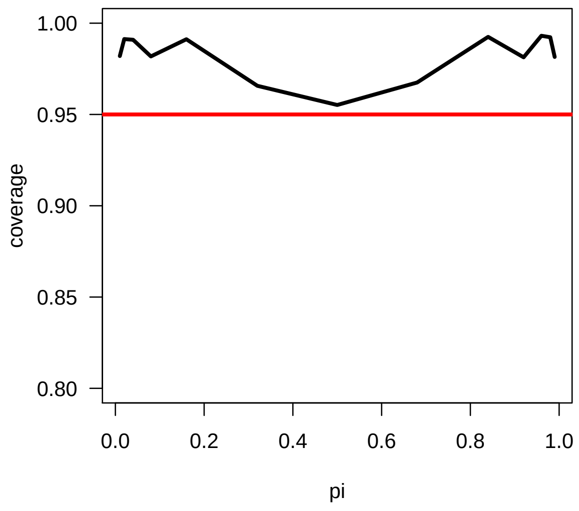
```
> N <- 10000
> n <- 10
> pi <- 0.25
> x <- rbinom(N, prob=0.25, size=n)
> covered <- numeric(N)
> for (i in 1:N) {
+   ci <- binom.test(x[i], n)$conf
+   covered[i] <- (ci[1] < pi) & (pi < ci[2])
+ }
> mean(covered)

[1] 0.9832
```

So our interval is actually fairly conservative here: its coverage is about 98%. What if we re-run the simulation with $n = 50$?

It would be interesting to see what happens to the coverage as a function of π. We can accomplish this with *nested* `for` loops. This actually takes a little while to run, so let's add a progress bar to let us know how things are going.

```
> N <- 10000
> n <- 20
> pi <- c(0.01, 0.02, 0.04, 0.08, 0.16, 0.32, 0.5, 0.68, 0.84, 0.92, 0.96, 0.98, 0.99)
> coverage <- numeric(length(pi))
> pb <- txtProgressBar(1, length(pi), style=3)
> for (j in 1:length(pi)) {
+   x <- rbinom(N, prob=pi[j], size=n)
+   covered <- numeric(N)
+   for (i in 1:N) {
+     ci <- binom.test(x[i], n)$conf
+     covered[i] <- (ci[1] < pi[j]) & (pi[j] < ci[2])
+   }
+   coverage[j] <- mean(covered)
+   setTxtProgressBar(pb, j)
+ }
```

```
> plot(pi, coverage, type="l", lwd=3, ylim=c(0.8,1), las=1)
> abline(h=0.95, col="red", lwd=3)
```



So our Clopper-Pearson interval seems quite conservative when π is close to 0 or 1, but pretty close to the *nominal coverage* when π is close to 0.5. As guaranteed by theory, the coverage is indeed always at least 95%.

There are lots of directions we could head here:

- Re-run the simulation above with a different sample size
- Try a similar simulation, only keeping π fixed and plotting coverage vs. `n`
- Include one or both of the Bayesian intervals and see what their coverage looks like

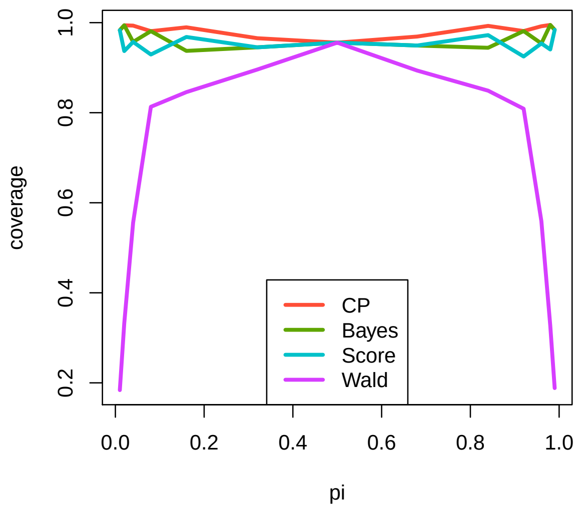
Now, let's expand on this simulation, adding the Wald, score, and Bayesian HPD intervals as competitors. I have posted a function online, `bayes.binom`, that we can use to calculate HPD intervals for binomial data (as you know from your homework assignment, there is no standard `R` function for this). The score interval is available via the `R` function `prop.test` (for inference concerning proportions). Lastly, there is no `R` function that returns the Wald interval, but it is trivial to write one. So, after we source the following:

```
> source("http://myweb.uiowa.edu/pbreheny/5710/f16/labs/binom.bayes.R")
> prop.wald <- function(x, n, level=0.95) {
+   pi.hat <- x/n
+   SE <- sqrt(pi.hat*(1-pi.hat)/n)
+   z <- qnorm(c((1-level)/2, 1-(1-level)/2))
+   pi.hat + z*SE
+ }
> prop.test(31,39) ## Just to show how prop.test works


1-sample proportions test with continuity correction

data:  31 out of 39, null probability 0.5
X-squared = 12.41, df = 1, p-value = 0.000427
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.6305740 0.9012996
sample estimates:
        p
0.7948718
```

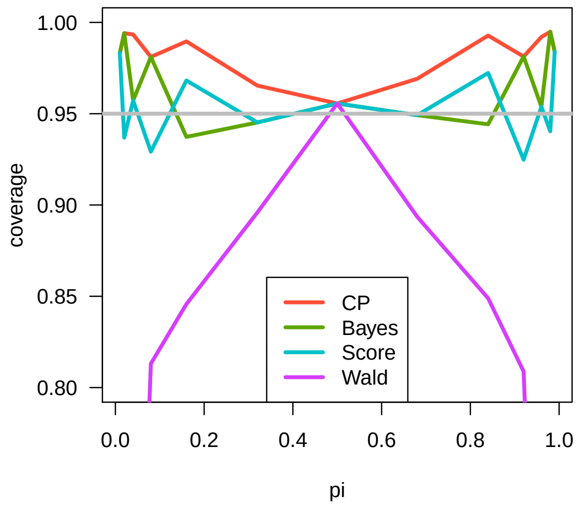
We can now run our simulation loop as in the last lab, using the `binom.test`, `binom.bayes`, `prop.test`, and `prop.wald` functions to calculate our various intervals. Here, for the sake of organization, I'll store all the coverage results in a matrix (previously, a vector was sufficient).

```
> N <- 10000
> n <- 20
> pi <- c(0.01, 0.02, 0.04, 0.08, 0.16, 0.32, 0.5, 0.68, 0.84, 0.92, 0.96, 0.98, 0.99)
> coverage <- matrix(NA, length(pi), 4, dimnames=list(pi, c("CP", "Bayes", "Score", "Wald")))
> pb <- txtProgressBar(1, length(pi), style=3)
> for (j in 1:length(pi)) {
+   x <- rbinom(N, prob=pi[j], size=n)
+   covered <- matrix(NA, N, 4)
+   for (i in 1:N) {
+     ci <- binom.test(x[i], n)$conf
+     covered[i,1] <- (ci[1] < pi[j]) & (pi[j] < ci[2])
+     ci <- binom.bayes(x[i], n)$ci.hpd
+     covered[i,2] <- (ci[1] < pi[j]) & (pi[j] < ci[2])
+     ci <- prop.test(x[i], n, correct=FALSE)$conf
+     covered[i,3] <- (ci[1] < pi[j]) & (pi[j] < ci[2])
+     ci <- prop.wald(x[i], n)
+     covered[i,4] <- (ci[1] < pi[j]) & (pi[j] < ci[2])
+   }
+   coverage[j,] <- apply(covered, 2, mean)
+   setTxtProgressBar(pb, j)
+ }
```

```
> col <- hcl(seq(15, 375, len = 5), 150, 60)[1:4] ## Setting up some colors
> matplot(pi, coverage, type="l", lwd=3, lty=1, col=col)
> legend("bottom", legend=colnames(coverage), col=col, lwd=3)
```



```
> matplot(pi, coverage, type="l", lwd=3, lty=1, col=col, ylim=c(0.8,1), las=1)
> legend("bottom", legend=colnames(coverage), col=col, lwd=3)
> abline(h=0.95, col="gray", lwd=3)
```



Note the use of `apply` here. `apply` is a very useful function that applies a function across all rows or columns of a matrix (or a larger array). We could write a for loop to calculate the average coverage for each method, but with `apply`, we can obtain the result we're interested in with just a single line of code. Also, the function `matplot` works a lot like `plot`, but allows you to plot several lines at once.

Some observations I would make:

- The Wald interval is simply unacceptable. Its coverage is nowhere even close to 95%. The other three intervals are at least reasonable here, never falling too far below 95%.

- The Clopper-Pearson interval is the most conservative, but is also the only interval that always provides at least 95% coverage.
- The Bayesian interval is not a confidence interval at all – it provides an interval for the middle 95% of the posterior distribution and is not concerned with coverage in the long-run frequency sense – but still has reasonable frequentist properties. This is often the case for Bayesian methods.

As a brief aside, the default behavior for `prop.test` is to carry out a minor "continuity correction". This is a fairly simple adjustment, but we didn't really discuss it in class. To get the version of the interval we discussed in class, you can turn off the correction with `correct=FALSE`.