MA26620: Practical 7

Practical 7: Discrete Inference

Good morning! Today's practical is mainly composed of pen-and-paper (and stats tables) exercises which are not a million miles away from the kinds of questions that may appear in the final exam for this module. They use the methods we've met in the lectures, in particular calculating confidence intervals and performing hypothesis tests for quantities relating to the Binomial distribution. The notes on the module webpages (https://stats.vellender.com) should be useful, as of course should your lecture notes.

There's then a small section about functions in R. I strongly encourage you to **make sure you can do all of Sections 1 and 2** of this practical (do ask for help) before moving onto Section 3. However tempting it is to jump to the fun(?) computery bits, it's best to hone your skills on the exam-type questions first!

1 Recap

1.1 Binomial example

It's been a while, so let's recap the Binomial example we met in the last lecture before Christmas:

Example: A hospital treats patients with a drug. The manufacturer claims a 70% success rate. What evidence is there that the true success rate is lower than 70% if:

(i) 12 recover out of 20?

(ii) 48 recover out of 80?

(iii) 60 recover out of 100?

In the lecture we tackled (i) as follows:

Assuming trials are independent with constant probability of success p, then the number of patients recovering, $R \sim \text{Bin}(n, p)$. We test $H_0 : p = 0.7$ vs $H_1 : p < 0.7$. In case (i), $n = 20$ and we observe $r = 12$ successes, so

$$p_0^- = P(R \leq r) = P(R \leq 12) = P(F \geq 8) = 0.2777,$$

where F, denoting the number of failures, is distributed as $\text{Bin}(20, 0.3)$ (reformulating in terms of F in the above allowed us to use the Statistical Tables).

Since $p_0^- > 0.1$, the test is not significant at the 10% level and so we have insufficient evidence to reject H_0.

Let's now look at part (ii). We can begin similarly:

Assuming trials are independent with constant probability of success p, then the number of patients recovering, $R \sim \text{Bin}(n, p)$. We test $H_0 : p = 0.7$ vs $H_1 : p < 0.7$. In case (ii), $n = 80$ and we observe $r = 48$ successes, so

$$p_0^- = P(R \leq r) = P(R \leq 48).$$

Our problem now becomes how to find the probability that a Bin(80,0.7) takes a value of at most 48. Our stats tables don't include this information. That means that we have two options: use R to calculate it (we'll do this later in the practical), or use an approximation. Let's do the latter.

Under H_0, $\mathbb{E}[R] = 80 \times 0.7 = 56$, and $\text{Var}(R) = 80 \times 0.7 \times 0.3 = 16.8$.

Thus, by the Normal approximation to the Binomial,

$$Z = \frac{R - 56}{\sqrt{16.8}} \dot{\sim} N(0, 1),$$

where $\dot{\sim}$ (note the dot) denotes "approximately distributed as". With a continuity correction (see your lecture notes),

$$\begin{aligned} p_0^- &= P(R \leq 48) \\ &= P(R < 48.5) \\ &= P(Z < \frac{48.5 - 56}{4.09878} = -1.83) \\ &= 0.03362 \text{ (from Normal tables).} \end{aligned}$$

Since $p_0^- < 0.05$, the test is significant at the 5% level, so we have moderately strong evidence to reject H_0 in case (ii) (i.e. moderately strong evidence that the drug is less than 70% effective).

Part (iii) proceeds similarly - have a go!

1.2 One more thing on approximations: the Poisson approximation to the Binomial

Above, we used the Normal approximation to the Binomial; in the lectures we'd said that this was sensible if n is large and p is not too close to 0 or 1. As a rule of thumb, it's often suggested that $0.1 < p < 0.9$, $np \geq 5$, and $n(1-p) \geq 5$ for this approximation to be suitable.

If p **is** small ($p < 0.1$), then all is not lost, thanks to the Poisson approximation to the Binomial. It just so happens that if n is large and p is small, then R has an approximate **Poisson** distribution with mean $m = np$, that is:

$$R \to \text{Po}(np) \text{ as } n \to \infty, p \to 0.$$

For $p > 0.9$, reformulating in terms of $F = n - R$, the number of failures, allows you to use the same result, since then $F \sim \text{Bin}(n, q)$, where $q = 1 - p < 0.1$

2 Pen and paper exercises

1. In his long career Roy of the Rovers has scored in 75% of his football matches. In a run of 10 matches, Roy scores in only 2. Has he lost his touch?

2. A TV channel operates a policy of not recommissioning any TV series which it is confident has retained less than 25% of the viewing audience after its initial 10 week run. Of 443 households surveyed who watched the first episode of new reality show *Z-List Celebrities Learn Statistics*, 102 were still watching 10 weeks later. Should the show be cancelled?

3. For (i) and (ii), state non-binomial distributions which well-approximate the following binomial distributions:

   (i) $\text{Bin}(185, 0.4)$;

   (ii) $\text{Bin}(360, 0.05)$.

3 Functions in R

So far when using R, we tend to run one command at a time. For the kinds of things we've been asking R to do, this is fine. However if we were doing something that R didn't have a built-in command to do, or some process with many steps, it would quickly become tedious or difficult.

Thankfully, R allows you to create your own *functions* that can do pretty much whatever you'd like them to.

In R, functions essentially have four parts:

- **a name**;
- **a rule**: the body of the function, enclosed in curly brackets: { } (or *braces* to use their more grown-up formal name);
- **some inputs**: usually the arguments of the function;
- **an output**.

3.1 My first R function

Let's begin by learning how to make a really simple function (which we'll call `f1`) that takes a couple of numbers as an input and returns their sum. To do this, we'd type

```
f1 <- function(x,y){x + y}
```

Here, the function is named f1, takes two inputs called `x` and `y` (in the parentheses) and returns the result of the computations in the curly brackets. Try typing `f1(42,1)` and press Enter. You should get 43 (which is of course 42+1). Try a couple of other pairs of numbers to check that it works as expected.

Make a function called `f2` that takes three inputs and outputs the product of the first two, minus the third. Check whether it works with a few different values.

3.2 More useful

Recall that in the lecture and in the first section of this practical, we saw an example regarding success rates of a drug. Suppose you wanted to test a large number of drugs in a similar way. You *could* just do the work that we did in the lecture over and over again for many different scenarios by hand, but this would end up getting dull quickly. It's probably more efficient to write a function once in R, which can then take the key numbers and do the hard work for us.

Recall the example: *A hospital treats patients with a drug. The manufacturer claims a 70% success rate. What evidence is there that it's less than 70% if (i) 12 recover out of 20? (ii) 48 out of 80? (iii) 60 out of 100?*

We conducted a hypothesis test of $H_0 : p = 0.7$ vs $H_1 : p < 0.7$. The number of patients recovering, R is distributed as Bin(n, p), so under H_0, the probability of observing r or fewer recoveries is

$$p_0^- = P(R \leq r) = P(\text{Bin}(n, 0.7) \leq r).$$

In the case where $n = 20, r = 8$ (see your lecture notes), we found: $p_0^- = P(\text{Bin}(20, 0.7) = 0.2277$. To find this number from R (rather than using tables), we could use the (pbinom) command: `pbinom(size = 20, prob = 0.7, q = 12, lower.tail=TRUE)`.

If we want to be able to answer many such problems, our function should basically do the hypothesis test for us (thus reducing our workload), or at least spit out the p_0^- values that we arrived at in the lectures for minimal effort. It should take the claimed drug effectiveness rate as an input (in our case 0.7 but this would be different for different drugs so let's keep it general), along with observed number of

```
drugRecoveries <- function(claimed=0.7, n, observed){
      p0minus <- pbinom(size=n, prob=claimed, q=observed, lower.tail=TRUE)
      cat("The p-value is ",p0minus)
      }
```

recoveries and the sample size. Its output should be the p-value (which we denoted p_0^- in the lecture). Here's one way we might do it (read the bit that follows it before you try to type it!):

Note that I've written this on **several lines**. We could, in principle have written it in one massively long line with some semicolons (bad idea), but if there's an error it could be really really difficult to unpick. When you're working on things that might get a bit lengthy, the easiest thing to do in RStudio is to use the **Source** pane. To get this pane to appear, in the History tab, select a command (any command) and click "To source". Then you'll get a notepad in the top left, which you can type longer commands like the one above. To run a selection in your Source notepad, simply highlight (i.e. select) what you want to run and then either click the little icon with a green arrow or press Ctrl+Enter.

Let's examine what this function called `drugRecoveries` does. It takes three inputs: `claimed`, `n`, `observed`. It computes `p0minus` which is $P(Bin(\texttt{n}, \texttt{claimed}) \leq \texttt{observed}))$. None of these assignments output (print) anything to the console. Finally the `cat` command outputs some human-readable text (see `?cat` for more details) followed by the calculated value of `p0minus`.

Another thing to note is the `claimed=0.7` in the function definition. This gives a default value for the claimed effectiveness of the drug, so if we don't specify a value for `claimed`, R will assume it's 0.7.

Check that the function works and agrees with our hand-calulated results from the lecture: run `drugRecoveries(0.7, 20, 12)`. What about if 48 recoveries among 80 patients were observed (thus calculating precisely what we approximated in the lecture)? 60 out of 100? Try some other numbers. It's much easier now than doing it by hand! What if the claimed recovery rate was only 66% rather than 70%? Hopefully from this example you can see that pretty much any complicated procedure could be implemented as a function (or perhaps many functions), which can be big time-savers for repetitive or complicated tasks.

3.3 A practice exercise

Suppose we conduct a large number n of Bernoulli trials. Then the number of successes is distributed as Bin(n, p). If we observe r successes in n trials, then we can estimate p by $\hat{p} = r/n$.

Write a function called `binconf` that outputs a confidence interval for p. The function should take the following as inputs:

- the number of binomial trials, `n`;
- the number of sucesses observed, `r`;
- the confidence level (e.g. an input of 0.95 would cause a 95% confidence interval to be computed), `conf.level`.

In order to select critical values from a $N(0, 1)$ distribution, you might find the following useful: the command `qnorm(x)` returns the value, y say, for which $P(N(0, 1) < y) = \texttt{x}$.

As an example to test your function on: `binconf(n=200,r=133,conf.level=0.9)` should output:
`The 90% confidence interval for p hat is (0.6101034, 0.7198966)`

Some advice (applicable to programming in general, not just this example): make sure you can do it by hand and then try to make R do the steps that you've done.