Open Intro Biostat R Companion: Chapter 2

Morgan F. Breitmeyer

Contents

Chapter 2 explains probability, how its used, and what questions it can be used to answer. In this chapter, we will go through some examples of how `R` can be used to solve propability problems. Once again, we will see that there are several different methods that can be used to answer the same question.

We will focus on **conditional probability** here as these are the questions you would most commonly use `R` to solve. To review, a conditional probability is the probability of an outcome given prior knowledge about another factor or condition. For example, we could ask what is the conditional probability of an individual being male *given* they have brown hair. **Joint probability** is the probability of two events occuring simultaneously. Referring back to our example, the joint probability would be the probability of an individual being male *and* having brown hair. We could also consider the **marginal probability**, or the probability of a single event occuring, such as an individual being male. Another marginal probability would be the probability of an individual having brown hair.

The mathematical definition for conditional probability is, for events A and B, the probability of A given B is computed as

$$P(A|B) = \frac{P(A \text{ and } B)}{P(B)}$$

1 Case Study on Conditional Probability

In order to demonstrate the various techniques which can be used to solve a conditional probability question in R, we are going to consider a case study of cystic fibrosis. The prevalence of cystic fibrosis is considered to be 1 in 6000 people. A test used to screen for cystic fibrosis has a **sensitivity** of 0.950, meaning that it correctly gives a positive test result in a patient with the disease 95% of the time. This can also be referred to as the **true positive rate**. Additionally, the test has a

specificity of 0.99897, meaning that it correctly gives a negative test result in a patient without the disease 99.897% of the time. This can be referred to as the **true negative rate**. Using this information, we want to determine the **positive predictive value** of the test, or the probability of a patient having the disease given their test result is positive.

2 Method 1 – Bayes Theorem

To solve this problem using Bayes, we would want to define two sets of events:

- $T+$: the test returning a positive result
- $T-$: the test returning a negative result
- $D+$: the patient having cystic fibrosis
- $D-$: the patient not having cystic fibrosis

We want to find the positive predictive value, or $P(D+|T+)$, and this can be done using Bayes theorem as follows,

$$P(D+|T+) = \frac{P(T+|D+)P(D+)}{P(T+)} = \frac{\text{(sensitivity)*(prevalence)}}{\text{(sensitivity)*(prevalence)} + (1 - \text{specificity})*(1 - \text{prevalence})}$$

One of the handy features of R is that it can be used as a calculator, and we can use it to solve the above equation.

```
prevalence = 1/6000
sensitivity = 0.950
specificity = 0.99897

ppv.bayes = (sensitivity*prevalence)/
  ((sensitivity*prevalence)+((1-specificity)*(1-prevalence)))
ppv.bayes

## [1] 0.1332591
```

3 Method 2 – Using a Table

The second technique which can be used to solve this question is by filling a table with a figurative population of 100,000 people and calculating the PPV using this. The table would look like this

	D+	D-	Total
T+			
T-			
Total			10,000

The first step would be to calculate the expected number of disease cases and non-diseases cases in the population, which would correspond to the following calculation

```
population.size = 100000
expected.cases = prevalence * population.size
expected.cases

## [1] 16.66667

expected.noncases = (1 - prevalence) * population.size
expected.noncases

## [1] 99983.33
```

And with this calculation, we can update our table to this

	D+	D-	Total
T+			
T-			
Total	16.7	99983.3	10,000

The next step is to calculate the expected number of cases of true positives and the expected number of cases of false positives as follows.

```
expected.true.positives = expected.cases * sensitivity
expected.true.positives

## [1] 15.83333

expected.false.positives = expected.noncases * (1 - specificity)
expected.false.positives

## [1] 102.9828

total.expected.positives = expected.true.positives + expected.false.positives
total.expected.positives

## [1] 118.8162
```

The table can again be updated to the following,

	D+	D-	Total
T+	15.83	102.98	118.8
T-			
Total	16.7	99983.3	10,000

The final step is to calculate the positive predictive value, as follows

```
ppv.table = expected.true.positives/total.expected.positives
ppv.table

## [1] 0.1332591
```

We see that this result is the same as in the first method - success! This process of using a table is a great way to visualize this process, but it is not necessary to create the table as you go. Instead, this process can be done just using the *R* calculations that are seen above.

4 Method 3 – Simulation

The final method for solving this conditional probability problem is called **simulation** and involves the process of simulating 100,000 individuals who each have the same probability of having the disease. We then simulate a test performed on each of them, keeping the known specificity and sensitivity in mind. This will give us an imaginary dataset of 100,000 individuals from which we can solve for the conditional probability.

The first step in this process is to set some defined parameters as follows. Note that the last line "sets a seed" for our simulation. To perform the simulation, we will be using random number generation, and this line guarantees that we are getting the same set of "random" numbers everytime we perform the simulation. It does not affect the results at all, but rather is just to compare work between colleagues or compare against a previous simulation. Any integer can be put into the seed command.

```
#parameters
prevalence = 1/6000
sensitivity = 0.950
specificity = 0.99897
population.size = 100000
set.seed(8)
```

Next, we must create a space where we can store the information about our simulated individuals. Imagine as if you wrote on a sheet of paper the numbers 1 through 100,000 and left a blank line next to each number. You could then go back and fill in each line with the information you wanted to. We can create two empty lists as follows, one for the patient disease status and one for their test outcomes.

```
disease.presence = vector("numeric", population.size)
test.outcome = vector("numeric", population.size)
```

Next, we must determine whether each individual in our simulated population has the disease or not. We know the prevalence of the disease, so for each individual we can say that $P(D+) = prevalence$. For the data storage, we will keep track of diseased individuals as 1 and non-diseased individuals as 0. The following line determines the disease status by either giving outcomes of 0 or 1, as indicated by the command $x = c(0, 1)$. The next element of size dictates how many individuals the *sample* should consider, in this case 100,000 individuals. Next, we dictate the probability of

each outcome in our list of possible outcomes x. This prob element must have the same number of possibilities as x and each element corresponds to the same element in x. We see that for each individual, the probability of no disease, or 0, will be $1 - prevalence$ and the probability of disease, or 1, will be $prevalence$. The last element of this command is "replace = TRUE", and this commands tells the computer whether or not to return a possible outcome to the selection options after it has been picked once. For example, if the first person has no disease, i.e. they have outcome 0, this command says to put the 0 back in the options list, x, for the next person to possibly have. Essentially, this just means that values in x can be repeated.

```
disease.status = sample(x = c(0,1), size = population.size, prob=c(1 - prevalence,
              prevalence), replace = TRUE)
```

Next, we perform our simulation using a process called **looping**. Looping is the idea that we want to perform the same set of steps for each individual. This set of steps includes looking at the disease status for each individual and if they are diseased, considering the sensitivity to determine their test result, or if they are not diseased, considering the specificity to determine their test result. This point illustrates a key idea of why we use simulation studies. In a real life study, when a test is performed, the true disease status is not actually known, which is why you are performing the test. However, in this case, we can simulate true disease status as if we were omnipotent, and then, see how the test would react based on that.

There are a lot of commands in here, so let's try to break them down a bit. The first one we see is our loop, the command for. This command simply takes the numbers 1 through 100,000 as dictated by the code "1:population.size" and scrolls through them one at a time. It starts by setting a variable ii equal to 1, then all of the code inside the brackets is executed, then it sets ii equal to the next number and so forth. Within this loop are two if statements, which operate by checking if the code inside the parentheses is true and if so, performing the code inside the brackets. If not, the code inside the brackets is skipped over. In this case, we see the code inside the if parentheses says $disease.status[ii] == 1$, and this is testing whether or not the disease status of the iith person is equal equal to 1. The two equals signs here imply the question of whether or not the left hand side is equal to the right hand side. The next if statement does the opposite by testing whether or not an individual is not diseased, i.e. having disease status of 0. Inside the if statements, we see a similar $sample$ command as above which uses probabilities to determine an outcome.

Note that to run the below code, you must run the entire section from for to the last bracket. This can be done by selecting all of the lines and selecting run, or by hitting command enter on each line individually all the way through. R does not understand how to run a loop unless it has the entire loop.

```
for (ii in 1:population.size) {
  if(disease.status[ii] == 1) {
    test.outcome[ii] = sample(c(0,1), size=1,
                              prob = c(1 - sensitivity, sensitivity))}
  if(disease.status[ii] == 0) {
   test.outcome[ii] = sample(c(0,1), size=1,
                              prob = c(specificity, 1 - specificity))}
}
```

The final step is to calculate the ppv as we have been doing so far.

```
## count the total number of people with disease
num.disease = sum(disease.status)
## count the total number of positive tests
num.pos.test = sum(test.outcome)
## identify which individuals have the disease
d = (disease.status == 1)
## identify how many positive tests match with diseased individuals
num.true.pos = sum(test.outcome[d])
## calculate ppv
ppv.sim = num.true.pos/num.pos.test
ppv.sim

## [1] 0.1367521
```