# Introduction to Probability

*Chapter 2, Lab 1: Solutions*

*OpenIntro Biostatistics*

**Topics**

- Defining probability
- Rules of probability
- Simulation

Probabilities for events can be calculated via simulation by simply repeating an experiment a large number of times and then counting the number of times the event of interest occurs. According to the Law of Large Numbers, as the number of repetitions increase, the proportion $\hat{p}_n$ of occurrences converges to the probability $p$ of that event.

The material in this lab corresponds to Section 2.1 of *OpenIntro Biostatistics*.

## Introduction

Suppose that a biased coin is tossed 5 times; the coin is weighted such that the probability of obtaining a heads is 0.6. What is the probability of obtaining exactly 3 heads?

Before writing the code for a simulation to estimate probability, it is helpful to clearly define the experiment and event of interest. In this case, the experiment is tossing a biased coin 5 times and the event of interest is observing exactly 3 heads.

1. The following code illustrates the use of the `sample()` command to simulate the result for one set of 5 coin tosses.

```
#define parameters
prob.heads = 0.6
number.tosses = 5

#simulate the coin tosses
outcomes = sample(c(0, 1), size = number.tosses,
                  prob = c(1 - prob.heads, prob.heads), replace = TRUE)

#view the results
table(outcomes)
```

```
## outcomes
## 0 1
## 3 2
```

```
#store the results as a single number
total.heads = sum(outcomes)
total.heads
```

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

a) Using the information given about the experiment, set the parameters for `prob.heads` and `number.tosses` and run the code chunk.

b) To generate `outcomes`, the `sample()` command draws from the values `0` and `1` with probabilites corresponding to those specified by the argument `prob`. Which number corresponds to heads, and which corresponds to tails?

The number `0` corresponds to tails and `1` corresponds to heads. This information comes from the structure of the vectors used in the `sample()` command; the elements in `c(0, 1)` are sampled with probabilities `c(1 - prob.heads, prob.heads)`, respectively.

c) Why is it important to sample with replacement?

There are only two numbers being sampled from: `0` and `1`. From a coding perspective, sampling without replacement would only allow for two coin tosses, since the vector contains only two elements. From a statistical perspective, sampling with replacement allows any coin to have the outcome `0` (tails) or `1` (heads), even if another coin was observed to be heads or tails; i.e., sampling with replacement allows the coin tosses to be independent.

d) What is the advantage of representing the outcomes with the values `0` and `1`, rather than with letters like "T" and "H"?

Representing the outcomes with `0` and `1` allows for the use `sum()` to return the number of heads.

e) Run the code chunk again to simulate another set of 5 coin tosses. Is it reasonable to expect that the results might differ from the first set of tosses? Explain your answer.

Yes, it is reasonable to expect that the number of heads in a set 5 coin tosses will not always be the same, even as the probability of a single heads remains constant. The inherent randomness makes it possible to observe anywhere between 0 and 5 heads (inclusive) in any set of 5 tosses.

2. The following code uses a `for` loop to repeat (i.e., replicate) the experiment and record the results of each replicate. The term `k` is an index, used to keep track of each iteration of the loop; think of it as similar to the index of summation $k$ (or $i$) in sigma notation ($\sum_{k=1}^{n}$). The value `num.replicates` is set to 50, specifying that the experiment is to be repeated 50 times.

The command `set.seed()` is used to draw a reproducible random sample; i.e., re-running the chunk will produce the same set of outcomes.

```
#define parameters
prob.heads = 0.6
number.tosses = 5
number.replicates = 50

#create empty vector to store outcomes
outcomes = vector("numeric", number.replicates)

#set the seed for a pseudo-random sample
```

```r
set.seed(2018)

#simulate the coin tosses
for(k in 1:number.replicates){

  outcomes.replicate = sample(c(0, 1), size = number.tosses,
                  prob = c(1 - prob.heads, prob.heads), replace = TRUE)

  outcomes[k] = sum(outcomes.replicate)
}

#view the results
outcomes
```

```
##  [1] 5 3 1 2 4 4 4 4 2 3 3 3 4 3 4 4 4 3 3 2 4 4 2 3 3 3 2 3 3 2 4 3 3 1 3
## [36] 2 2 5 0 2 3 3 3 4 3 4 4 3 5 4
```

```r
addmargins(table(outcomes))
```

```
## outcomes
##   0   1   2   3   4   5 Sum
##   1   2   9  20  15   3  50
```

```r
heads.3 = (outcomes == 3)
table(heads.3)
```

```
## heads.3
## FALSE  TRUE
##    30    20
```

a) The parameters of the experiment have already been filled in; the probability of heads remains 0.6 and the number of tosses is set to 5. This code repeats the experiment 50 times, as specified by `number.replicates`. Run the code chunk.

b) How many heads were observed in the fourth replicate of the experiment? Hint: look at `outcomes`.

In the fourth replicate of the experiment, 2 heads were observed.

```r
outcomes[4]
```

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

c) Out of the 50 replicates, how often were exactly 3 heads observed in a single experiment?

Out of the 50 replicates, exactly 3 heads were observed in 20 of the replicates.

d) From the tabled results of the simulation, calculate an estimate of the probability of observing exactly 3 heads when the biased coin is tossed 5 times.

The estimate of the probability is $20/50 = 0.40$.

e) Re-run the simulation with 10,000 replicates of the experiment; calculate a new estimate of the probability of observing exactly 3 heads when the biased coin is tossed 5 times.

The new estimate of the probability is 3521/10000 = 0.352.

```
## heads.3
## FALSE  TRUE
##  6479  3521
```

f) Using the laws of probability, calculate the exact probability of observing exactly 3 heads when the biased coin is tossed 5 times.

There are 10 ways that exactly 3 heads can occur when a coin is tossed 5 times: TTHHH, HTTHH, HHTTH, HHHTT, THTHH, THHTH, THHHT, HTHHT, HTHTH, HHTHT. Each sequence represents a disjoint outcome, so the addition rule can be used. In this case, the probability of each disjoint outcome is equal. The exact probability of observing exactly 3 heads when a coin is tossed 5 times is $10 \times (0.60)^3(0.40)^2 = 0.346$.

```
prob.heads = 0.6

10*prob.heads^3*(1-prob.heads)^2
```

```
## [1] 0.3456
```

g) How do the values calculated in parts d) and e) compare to the value calculated in part f)? Of the two values derived from simulation, why can the one calculated from 1,000 replicates be considered more reliable?

From algebraic methods, the probability is 0.346. The value obtained from 50 replicates is 0.40, while the value obtained from 10,000 replicates is 0.352. The answer from part e) is very close to the exact number from part f). Increasing the number of replicates increases the likelihood that the estimated probability is close to the exact probability found from the algebraic method.

**Mandatory Drug Testing**

The simulation framework illustrated in the previous section can easily be adapted for other scenarios that may seem more complicated than coin tossing. The drug testing scenario examined in this section appears in *OI Biostat* as Example 2.30.

Mandatory drug testing in the workplace is common practice for certain professions, such as air traffic controllers and transportation workers. A false positive in a drug screening test occurs when the test incorrectly indicates that a screened person is an illegal drug user. Suppose a mandatory drug test has a false positive rate of 1.2% (i.e., has probability 0.012 of indicating that an employee is using illegal drugs when that is not the case). Given 150 employees who are in reality drug free, what is the probability that at least one will (falsely) test positive? Assume that the outcome of one drug test has no effect on the others.

3. Calculate the probability using an algebraic approach.

   First, note that the complement of at least 1 person testing positive is that no one tests positive (i.e., all employees test negative). The multiplication rule can then be used to calculate the probability of 150 negative tests: $1 - (0.988)^{150} = 1 - 0.16 = 0.84$.

```
prob.false.positive = 0.012
prob.true.negative = 1 - prob.false.positive

1 - prob.true.negative^150
```

```
## [1] 0.836491
```

4. Calculate an estimate of the probability via simulation.

   a) Before writing the code to run repeated simulations, it is helpful to first think about the problem structure and how to model one run of the experiment with the `sample()` command.

      i. What are the two possible outcomes when a person is tested?

         A person can either test positive or negative. In this scenario, since all employees are drug-free, the positive results are false positives and the negative results are true negatives.

      ii. What are the associated probabilites of the possible outcomes?

         The probability of a true positive result is 0.012 and the probability of a true negative is $1 - 0.012 = 0.988$.

      iii. Describe a reasonable way to represent the two possible outcomes with values `0` and `1`.

         Since it is of interest to identify the number of employees that test positive, it is convenient to represent positive results with `1` and negative results with `0`.

   b) Simulate the results for one set of 150 tests. Use `set.seed()` so that the results are reproducible. Was there at least one employee who tested positive?

      Yes, there was at least one employee who tested positive. In this set of tests, two employees tested positive.

```r
#define parameters
prob.false.positive = 0.012
number.employees = 150

#set the seed for a pseudo-random sample
set.seed(2018)

#simulate the tests
results = sample(c(0,1), size = number.employees,
         prob = c(1 - prob.false.positive, prob.false.positive),
         replace = TRUE)

#view the results
table(results)
```

```
## results
##   0   1
## 148   2
```

```r
sum(results)
```

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

c) Based on 100,000 replicates, estimate the probability that at least one employee tests positive out of 150 drug-free employees.

The estimated probability that at least one employee tests positve out of 150 drug-free employees is $83,718/100,000 = 0.837$.

```r
#define parameters
prob.false.positive = 0.012
number.employees = 150
number.replicates = 100000

#create empty vector to store results
results = vector("numeric", number.replicates)

#set the seed for a pseudo-random sample
set.seed(2018)

#simulate the tests
for(k in 1:number.replicates){

  results.replicate = sample(c(0,1), size = number.employees,
         prob = c(1 - prob.false.positive, prob.false.positive),
         replace = TRUE)

  results[k] = sum(results.replicate)

}
```

```r
#view the results
table(results)
```

```
## results
##     0     1     2     3     4     5     6     7     8     9    10
## 16282 29847 27015 16201  7183  2524   720   169    49     7     3
```

```r
at.least.1.pos = (results >= 1)
table(at.least.1.pos)
```

```
## at.least.1.pos
## FALSE  TRUE
## 16282 83718
```

**Mammograms**

5. The specificity of a diagnostic test refers to the probability that a test is negative in the absence of disease. Mammograms have a specificity of 95% for detecting breast cancer.

   a) Define the relationship between the specificity of a test and the probability of a false positive.

   The specificity of a test is the probability of a true negative result. A true negative is the complement of a false positive. Thus, the specificity of a test equals 1 minus the probability of a false positive.

   b) Suppose a clinic conducts approximately 50 mammograms in a week. Use simulation to calculate the probability that no more than 1 woman will test positive if none of the women have breast cancer.

   The probability that no more than 1 woman will test positive if none of the woman have breast cancer is 0.28.

```
#define parameters
specificity = 0.95
number.women = 50
number.replicates = 100000

#create empty vector to store results
results = vector("numeric", number.replicates)

#set the seed for a pseudo-random sample
set.seed(2018)

#simulate the tests
for(k in 1:number.replicates){

  results.replicate = sample(c(0,1), size = number.women,
                     prob = c(specificity, 1 - specificity),
                     replace = TRUE)

  results[k] = sum(results.replicate)

}

#view the results
table(results)
```

```
## results
##     0     1     2     3     4     5     6     7     8     9    10    11
##  7858 20162 26036 21929 13560  6716  2568   855   237    66    11     2
```

```
at.most.1.pos = (results <= 1)
table(at.most.1.pos)
```

```
## at.most.1.pos
```

```
## FALSE  TRUE
## 71980 28020
```

c) Re-calculate the probability described in a) if the probability of a false positive result is 0.01.

With the probability of a false positive result at 0.01 (i.e., specificity of 0.99), the probability that no more than 1 woman will test positive if none of the woman have breast cancer is 0.91.

```
#define parameters
specificity = 0.99
number.women = 50
number.replicates = 100000

#create empty vector to store results
results = vector("numeric", number.replicates)

#set the seed for a pseudo-random sample
set.seed(2018)

#simulate the tests
for(k in 1:number.replicates){

  results.replicate = sample(c(0,1), size = number.women,
                      prob = c(specificity, 1 - specificity),
                      replace = TRUE)

  results[k] = sum(results.replicate)

}

#view the results
table(results)
```

```
## results
##     0     1     2     3     4     5
## 60643 30515  7470  1203   158    11
```

```
at.most.1.pos = (results <= 1)
table(at.most.1.pos)
```

```
## at.most.1.pos
## FALSE  TRUE
##  8842 91158
```