STAT2011 Statistical Models

Computer Exercise Week 2

We are going to investigate the following question:

> *What is more likely, getting ⚀ at least once in four rolls of one die, or getting ⚀⚀ at least once in 24 rolls of two dice?*

After completing week 1’s report, compose a report of the following exercise:

1. Using the function `sample()`, create a vector called `rolls1` of a random sample from $\{1, 2, 3, 4, 5, 6\}$ of size 4,000 drawn with replacement. **Hint:** use the R command `args(sample)` to see the arguments that `sample()` expects to see:

   ```
   > args(sample)
   function (x, size, replace = FALSE, prob = NULL)
   ```

   The argument `x` represents the “population”, `size` the sample size, `replace` can be `TRUE` or `FALSE` to indicate sampling with or without replacement and `prob` can specify a vector of probabilities or sampling weights. The `replace=FALSE` tells us that the *default* value of `replace` is `FALSE` (this need not be specified if `replace=FALSE` is desired) and `prob=NULL` tells us this is an optional argument; the default is equal sampling weights.

2. Form the vector `rolls1` into a 1,000-by-4 matrix called `four.rolls` using the `matrix()` function; again you can use `args(matrix)` to see what arguments `matrix()` expects to see.

3. Each row of your matrix represents four rolls of a die. We wish to count how many rows have at least one `1` in them (this corresponds to a ⚀). Equivalently, we can compute the minimum value of each row, and count how many of these are `1`. Use the `apply()` function to create a vector called `min.roll` consisting of the 1,000 row minimia of the matrix `four.rolls`. **Hint:** if `M` is a matrix then `apply(M,1,sum)` gives the row sums of `M`, `apply(M,2,min)` the column minima of `M`, etc..

4. Counting how many and/or what proportion of elements of `min.roll` are equal to `1` can be achieved using a *logical comparison*. The vector `min.roll==1` (**don’t print it in your report!!**) consists of 1,000 `TRUE`’s and `FALSE's`, with `TRUE` corresponding to a minimum of `1`. These `TRUE`’s and `FALSE`’s are subsequently interpreted as `1`’s and `0`’s respectively, if need be. So `sum(min.roll==1)` gives us the count.

5. We now do something similar for the second event.

   (a) create a vector called `rolls2` consisting of a random sample of size 48,000 with replacement from $\{1, 2, 3, 4, 5, 6\}$;

   (b) form the vector `rolls2` into a 24,000-by-2 matrix called `two.rolls`;

   (c) ⚀⚀ occurs if and only if the sum is 2; thus form a vector called `sum.rolls` consisting of 24,000 row sums of the matrix `two.rolls`;

   (d) form this vector of sums into a 24-by-1,000 matrix called `twodozen`;

   (e) each column now corresponds to 24 rolls of two dice; form a vector `min.pair` of column minima, and

   (f) finally count how many of these are equal to `2`.

6. Convert your counts obtained here and in question 4 into two estimates; call them `p1.est` and `p2.est`. Based on these, what can you say in response to the question posed at the beginning of the exercise?

7. How accurate are your estimates? We can get an idea of the accuracy of the estimates using a little simulation. Replicate the whole procedure 25 times using a `for`-loop:

```
results1 <- 0
results2 <- 0
for (i in 1:25){
...                                  # your commands from questions 1 to 3 go here
results1[i] <- sum(min.roll==1)
...                                  # your commands from question 5 parts (a)-(e) go here
results2[i] <- sum(min.pair==2)
}
```

8. Define `prob.ests1 <- results1/1000` and similarly `prob.ests2`; also `se1 <- sd(prob.ests1)` and similarly `se2`. These give an idea of how accurate the procedures are in general, and also how reliable our single estimates `p1.est` and `p2.est` are in particular. We call these the (estimated) *standard errors* of our estimates.

9. Compute the actual probabilities of these two events, assuming each number is equally likely at each roll, and that the rolls are independent (**Hint:** consider the complement). Call them `p1` and `p2`.

10. Compute

```
abs(p1.est - p1)/se1
```

   and do the same for case 2. In both cases your estimate should be within 2 or 3 standard errors of the true value. Did this occur?