Probability With R

Dublin R

November 7, 2013

Contents

1 Probability

1.1 The Birthday function

The R command `pbirthday()` computes the probability of a coincidence of a number of randomly chosen people sharing a birthday, given that there are n people to choose from. Suppose there are four people in a room. The probability of two of them sharing a birthday can be computed as follows: (Answer: about 1.6 %)

```
> pbirthday(4)
[1] 0.01635591
```

How many people do you need for a greater than 50% chance of a shared birthday for n people? Many would make the guess 183 people. Let us use the `sapply()` command. The first arguments is the data set (here a sequence of integers from 2 to 60) and then the command or function (here `pbirthday()`, but specified without the brackets. (starting from 2 – so the 5th element corresponds to 6 people, etc)

```
 > sapply(2:60,pbirthday)
 [1] 0.002739726 0.008204166 0.016355912 0.027135574
 [5] 0.040462484 0.056235703 0.074335292 0.094623834
 [9] 0.116948178 0.141141378 0.167024789 0.194410275
[13] 0.223102512 0.252901320 0.283604005 0.315007665
[17] 0.346911418 0.379118526 0.411438384 0.443688335
[21] 0.475695308 0.507297234 0.538344258 0.568699704
[25] 0.598240820 0.626859282 0.654461472 0.680968537
[29] 0.706316243 0.730454634 0.753347528 0.774971854
[33] 0.795316865 0.814383239 0.832182106 0.848734008
[37] 0.864067821 0.878219664 0.891231810 0.903151611
[41] 0.914030472 0.923922856 0.932885369 0.940975899
[45] 0.948252843 0.954774403 0.960597973 0.965779609
[49] 0.970373580 0.974431993 0.978004509 0.981138113
[53] 0.983876963 0.986262289 0.988332355 0.990122459
[57] 0.991664979 0.992989448 0.994122661
```

The answer is 23 people (see entry 22)- probably much less than you thought!!

1.2 Playing with Dice

A sequence of integers can be created using the ":" operator, specifying the upper and lower bounds on either side. This is very useful for Dice experiments. Importantly this sequence is constructed as a vector.

```
Dice = 1:6
```

Another important command is the `sample()` command. This command causes `R` to select n items from the specified vector. We can use it to simulate one roll of a die.

```
Dice = 1:6
sample(Dice,1)
```

```
> sample(Dice,1)
[1] 2
>
> sample(Dice,1)
[1] 4
>
> sample(Dice,1)
[1] 1
>
> sample(Dice,1)
[1] 2
```

Suppose we wish to simulate two rolls of a die. Surely we just specify 2 in the argument of the `sample()` command. In fact we do, but this is not enough. The `sample()` command works on the default basis of ***sampling without replacement***. That is to say: once a number has been selected, it can not be selected again.

```
> sample(Dice,6)
[1] 4 3 6 1 2 5
> sample(Dice,7)
Error in sample(Dice, 7) :
  cannot take a sample larger than the population when 'replace = FALSE'
```

What we do here is to additionally specify `replace=TRUE`. This specifies an experiment where there is ***sampling with replacement***. We can also use the `table()` function to study the outcomes of the simulation.

```
> sample(Dice,20,replace=TRUE)
 [1] 4 1 3 2 5 4 4 4 6 5 6 6 5 6 6 5 1 1 1 3
>
> X= sample(Dice,20,replace=TRUE)
> table(X)
X
1 2 3 4 5 6
4 4 2 3 3 4
> mean(X)
[1] 3.45
```

Let work on the basis of 100 dice rolls, and for the sake of simplicity let us consider the sum of those 100 rolls. Importantly, we are simulating a **fair** die. We expect to get a value of approximately 350, but each time we perform the experiment their will slight fluctuations. Try the last line of the code below a few times. How many times do you get a figure less than 340 or greater than 360?

```
X= sample(Dice,100,replace=TRUE)
sum(X)

#Equivalently
sum(sample(Dice,100,replace=TRUE))
```

1.3 Dice Experiment : Simulation Study

Lets consider this Dice Roll Summation experiment. We will perform the experiment 100000 times, and see what sort of distribution of summations we get. We will save the results in a vector called `sums`.

```
Sums=numeric()    # Initialize An Empty Vector
M=100000  # Number of Iterations

for (i in 1:M)
    {
     NewSum=sum(sample(Dice,100,replace=TRUE))
     Sums = c(Sums, NewSum)
     }
```

We can perform some basic statistical operations to study this vector. In particular we are interested in the extremes: How many times was there a summation less than 300, and how many times was there a summation greated than 400? (around 1.5% probability in each case)

```
> length(Sums[Sums<300])
[1] 144
> length(Sums[Sums>400])
[1] 160
```

Lets us look at a histogram (a type of bar chart) of the `Sums` vector (Use around breaks =100 to specify more intervals). What sort of shape is this histogram?

```
hist(Sums, breaks=100)
```

This is a ver crude introduction to the Central Limit Theorem. Even though the Dice Rolls are not normally distributed, the distribution of summations, are described in this experiment, are from a normally distributed sampling population. Also consider the probability of getting a sum more than 400. Recalling that dice simulation is for fair dice, the probability of getting a score more extreme than 400 is 1.5% approximately. This provides (again crudely) an introduction to the idea of p-values , which are used a lot in statistical inference procedures.

Suppose it was not certain whether a die was fair or crooked favouring higher values such as 4,5 and 6. The 100 roll experiment was performed and the score turned out

to be 400. It would be a very unusual outcome for a fair die, but not impossible. For crooked dice, larger summations would be expected and a score of approximately 400 would be common. Would you think the die was fair or crooked.

Footnote: Arbitrarily, let us consider a crooked dice, where 4,5 and 6 are twice a likely to appear. Try out the following code.

```
CrookedDice=c(1,2,3,4,4,5,5,6,6)
sum(sample(CrookedDice,100,replace=TRUE))
```