Probability and Statistics: A Primer for Beginners and Pre-Beginners

Coding Supplement to 1-2-6
R-1-2-6: Counting

Primary reference: Casella-Berger 2nd Edition

Setting up our environment

Before we can start looking at permutations and combinations in R, we need to install a package, which is a set of tools we can import into R. Luckily, this is very simple:

Remove the '#' before install.packages to install the package. Then you can put it back.

```
# R-1-2-6
#Install the gtools package if running for the first time
#(remove the '#' from the next line)
#install.packages('gtools')

#load the gtools package
library(gtools)
# let's remove whatever is in our global environment for a fresh start!
rm(list=ls())
```

The 'library' method loads the package we installed. You can also see we've cleared our "environment". This consists of all the variables, arrays, lists, etc. in the top-right corner of Rstudio.

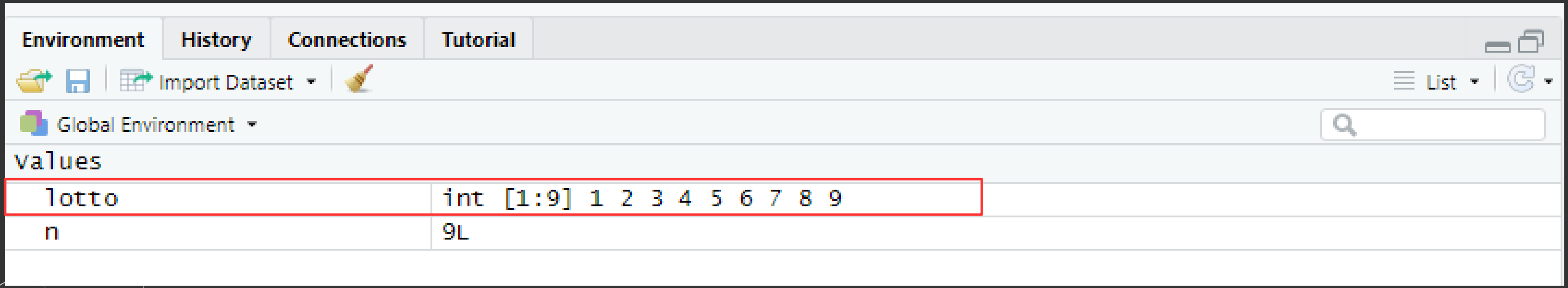
Setting up our environment

For the next part of our code, we need to set a seed. This is just a value that allows you to get the same results each time when you run code that generates random numbers. After that, we define a vector (a one-dimensional set of numbers) of integers from 1 to 9. You can see it in the global environment.

```
#set a seed so our results are reproducible (if you run a part of the code with a random sample,
#make sure you still include this line)
set.seed(83759)

#set up our nine-ball lottery as a vector containing integers 1 through 9, and then assign the total number of balls to
'n'
lotto = 1:9
n = length(lotto)
```

We assign the number of balls (9) to the variable 'n'.



```
Values
  lotto         int [1:9] 1 2 3 4 5 6 7 8 9
  n             9L
```

Counting where order is important

Remember that we can arrange the nine balls in 9! ways. If we pick 4 balls without replacing them, then on our first pick, there are nine possibilities, then eight, then seven, then six:

$$9*8*7*6=\frac{9!}{5!}=\frac{n!}{(n-r)!}=\frac{362,880}{120}=3,024$$

Counting where order is important

We can calculate 9! in R, but we can also express our problem as counting the ways we could arrange r=9 objects from n=9. 'permutations' generates a table (matrix) containing all the possible arrangements, and the number of possible arrangements is just 'nrow', the number of rows in that table.

```
#how many ways can we arrange the 9 balls?
permlotto = nrow(permutations(n, n, lotto))
permlotto
#notice that this is just nine factorial:
factorial(9)
```

From now on, console output will be shown in a red-outlined rectangle:

```
#how many ways can we arrange the 9 balls?
> permlotto = nrow(permutations(n, n, lotto))
> permlotto
[1] 362880
> #notice that this is just nine factorial:
> factorial(9)
[1] 362880
```

Counting where order is important

Just as a demonstration, we could "pick" our lottery numbers by taking a random sample without replacement from our nine lottery balls. This is done using the 'sample' method. We also assign the length of the pick (4) to the variable 'r'.

Now n = 9 and r = 4.

```
#randomly pick four balls without replacement and assign
#the number of balls picked to 'r'
pick = sample(lotto, 4, replace = FALSE)
pick
r = length(pick)

#we've picked '6 2 5 1'
```

```
> #randomly pick four balls without replacement and assign
> #the number of balls picked to 'r'
> pick = sample(lotto, 4, replace = FALSE)
> pick
[1] 6 2 5 1
> r = length(pick)
>
> #we've picked '6 2 5 1'
```

Counting where order is important

We'll skip over the calculations of permutations of four (24). Next, let's use our permutations method to calculate the number of possible lottery picks when order is important and they are picked without replacement. Again, we get 3,024.

We're choosing r=4 balls without replacement from n=9. To choose WITH replacement, set repeats.allowed = TRUE

```
#how many arrangements of four balls picked without replacement
#can we get?
orderednoreplace = nrow(permutations(n, r, lotto, repeats.allowed = FALSE))
orderednoreplace

#Notice, 3,024 is the result of dividing nine factorial by (9-4) factorial:
factorial(n)/factorial(n-r)
```

```
> #how many arrangements of four balls picked without replacement
> #can we get?
> orderednoreplace = nrow(permutations(n, r, lotto, repeats.allowed = FALSE))
> orderednoreplace
[1] 3024
>
> #Notice, 3,024 is the result of dividing nine factorial by (9-4) factorial:
> factorial(n)/factorial(n-r)
[1] 3024
```

What if order is unimportant?

If order is unimportant, we have to divide the number of possible picks (3,024) by the number of ways we could arrange a given pick of four balls (4!). Remember our combination formula:

$$\binom{n}{r} = \frac{n!}{r!(n-r!)} = \frac{9!}{4!(9-4)!} = \frac{362,880}{24*120} = 126$$

What if order is unimportant?

All we have to change from our permutations code is to use the 'combinations' method instead

We're choosing r=4 balls without replacement from n=9 where order is unimportant.

```
#now, let's see how many combinations of 4 balls we can pick without replacement
unorderednoreplace = nrow(combinations(n, r, lotto, repeats.allowed = FALSE))
unorderednoreplace

#we can pick 126 combinations of 4 balls without replacement.  Note that we get the same result with our formula:
factorial(n)/(factorial(r)*factorial(n-r))
```

```
> #now, let's see how many combinations of 4 balls we can pick without replacement
> unorderednoreplace = nrow(combinations(n, r, lotto, repeats.allowed = FALSE))
> unorderednoreplace
[1] 126
>
> #we can pick 126 combinations of 4 balls without replacement.  Note that we get the same result with our formula:
> factorial(n)/(factorial(r)*factorial(n-r))
[1] 126
```

What if order is unimportant and we pick with replacement?

Remember the more complicated formula we need to use if we're picking with replacement when order is unimportant:

$$\frac{(n+r-1)!}{r!(n+r-1-r)!}=\frac{(n+r-1)!}{r!(n-1)!}=\frac{12!}{4!8!}=495$$

What if order is unimportant and we pick with replacement?

We simply set 'repeats.allowed = TRUE' and see that we were right!

We're choosing r=4 balls with replacement from n=9 where order is unimportant.

```
#finally, lets calculate the number of combinations of 4 balls picked WITH replacement
unorderedreplace = nrow(combinations(n, r, lotto, repeats.allowed = TRUE))
unorderedreplace

#and our formula:
factorial(n+r-1)/(factorial(r)*factorial(n+r-1-r))
```

```
> #finally, lets calculate the number of combinations of 4 balls picked WITH replacement
> unorderedreplace = nrow(combinations(n, r, lotto, repeats.allowed = TRUE))
> unorderedreplace
[1] 495
>
> #and our formula:
> factorial(n+r-1)/(factorial(r)*factorial(n+r-1-r))
[1] 495
```