Exercise

Steve Pittard

January 27, 2016

Computing with Loops

Create a for loop that, given a numeric vector, prints out the number, its square, and cube on one line:

```
set.seed(123)
xvec <- rnorm(10)
```

So some example output might look like the following. You can use the **cat** function to do the printing of the numbers. There are examples of this function being used in the class material.

```
## -0.5604756 0.314133 -0.1760639
## -0.2301775 0.05298168 -0.01219519
## 1.558708 2.429572 3.786993
## 0.07050839 0.004971433 0.0003505278
## 0.1292877 0.01671532 0.002161086
## 1.715065 2.941448 5.044774
## 0.4609162 0.2124437 0.09791877
## -1.265061 1.60038 -2.024579
## -0.6868529 0.4717668 -0.3240344
## -0.445662 0.1986146 -0.08851497
```

Use a while loop to accomplish the same thing

```
## -0.5604756 0.314133 -0.1760639
## -0.2301775 0.05298168 -0.01219519
## 1.558708 2.429572 3.786993
## 0.07050839 0.004971433 0.0003505278
## 0.1292877 0.01671532 0.002161086
## 1.715065 2.941448 5.044774
## 0.4609162 0.2124437 0.09791877
## -1.265061 1.60038 -2.024579
## -0.6868529 0.4717668 -0.3240344
## -0.445662 0.1986146 -0.08851497
```

Comparing Vectors

As mentioned in class we usually use for loops that involve use of if statements to make some decision as we work through a vector or a matrix. For example consider two vectors. Both of the them will contain the same number of elements but their respective values will be different.

```
set.seed(123)
vec1 <- rnorm(20)
vec1
```

```
##  [1] -0.56047565 -0.23017749  1.55870831  0.07050839  0.12928774
##  [6]  1.71506499  0.46091621 -1.26506123 -0.68685285 -0.44566197
## [11]  1.22408180  0.35981383  0.40077145  0.11068272 -0.55584113
## [16]  1.78691314  0.49785048 -1.96661716  0.70135590 -0.47279141
```

```
set.seed(147)
vec2 <- rnorm(20)
vec2
```

```
##  [1]  0.24021576 -0.29229594 -0.55429616  1.65118618 -0.75374337
##  [6] -0.53573589  0.44308593 -1.87892786  0.93164047 -0.50381329
## [11]  1.16194408 -0.67753515 -0.84649005 -0.22688748 -0.10524852
## [16]  1.65766757 -1.00159811  0.82801694 -0.09423603  0.11018479
```

```
# This is easy if we use some of the things we have learned

vec3 <- as.numeric(vec1 >= vec2)
```

We could write some code involving a for loop and some if statements to create a new vector called say vec3 whose elements contain a 1 or a 0 depending on whether a given element in vec1 is greater or equal to the comparable element in vec2. Implement this technique using a for loop

Computing the sum of a series

Here are some exercises that will help you understand looping stuctures. Consider the following function. $h(x,n) = 1 + x + x^2 + ... + x^n = \sum_{i=0}^{n} x^i$ Given a specific value for n implement this function using a for loop. So if we set n equal to 4 then the sum would be:

```
## [1] 289
```

Comparing Matrices

Let's extend these ideas by doing something similar two matrices. We setup an empty matrix called mat3 to the hold the results of the following comparisons. For each element in mat1 we compare it to the same element in mat2. If the element value from mat1 is >= the one on mat2 then we put a 1 in the same element of mat3.

```
set.seed(123)
mat1 <- matrix(rnorm(16),4,4)

set.seed(321)
mat2 <- matrix(rnorm(16),4,4)
```

Now, something like this would be easy using some of the techniques we've already learned.

```
mat3 <- matrix(as.numeric(mat1 >= mat2),4,4)
mat3
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    0    1    0    1
## [2,]    1    1    1    0
## [3,]    1    0    1    1
## [4,]    1    0    0    1
```

But write for loop structure that accomplishes the same thing. You will need two for loops - one for the rows and one for the columns. This means you will need a subscript variable for rows (ii) and one for columns (jj). You will need to initialize a matrix which is the same size of mat1 (or mat2). Set all elements to zeros.

```
for (1 to the number of rows in mat1)
  for (1 to the number of columns in mat1)
     if (mat1 element >= mat2 element)
         set mat3 element to 1
         else set it to 0
   }
}
```