**Matt Fertakos**

**w/ input from group members: John, Bonnie, Mandy, Heather**

**Q1**

n=12345

vec\_1=sample(12,n,replace=TRUE)

head(vec\_1)

vec\_2<-vec\_1=='3'

**Q2**

1) There are a lot of entries so it will be easy to lose your place or miscount (especially since many are excluded due to the max.print size).

2) It would be very time consuming.

**Q3**

n=10 means there are 10 repeats of selecting a value from 0 to 12. This means there can be a different number of 3s in the 10 possible values each time the code is run.

**Q4**

A logical test like 'vec\_1==3' will simply pull out all entries that are exactly equal to 3. This does not change based on the number of threes within the set of 10 numbers.

**Q5**

Performing logical subsetting 'by hand' is a bad practice because it would be very time consuming to go through and find all of the entries that match in a long dataset, and could introduce human error. Also, if you share your code, the next person will have to spend that same time counting, and they could make a mistake as well. Running with logical subsetting assures that two people will receive the same value with the same code. This is especially true for large datasets where there is a greater chance for human error as opposed to smaller datasets.

**Q6**

for (i in 1:10)

{

print(paste0("This is loop interation: ",i))

}

**Q7**

n=15

for (i in 1:n)

{

print(paste0("This is loop interation: ",i))

}

**Q8**

n=17

vec\_1=sample(10,n,replace=TRUE)

for (i in 1:n)

{

print(paste0("The element of vec\_1 at index ", i, " is: ",vec\_1[i]))

}

**Q9**

#note to self if min and max are equal it will always do 1 to max

create\_and\_print\_vec = function(n, min=1,max=10)

{

vec\_1=sample(min:max,n,replace=TRUE)

for (i in 1:n)

{

print(paste0("The element at index ", i, " is: ",vec\_1[i]))

}

}