Statistical Computing Project 1

Mario Ibanez

January 23, 2016

```
N <- 10
library(knitr)
```

```
# Generates n values from standard uniform with seed x0
uniform_generator <- function(n, seed, sorted=TRUE){
  # assignments to save time
  a <- 7^5
  mod <- 2^31 - 1
  # initialize vector of 0's
  vector <- c(rep(0, times=n))
  # assign seed
  vector[1] <- seed %% mod
  # calculate rest of the values
  for(i in 2:n){
    vector[i] <- (a * vector[i-1]) %% mod
  }
  # divide by the mod value
  vector <- vector/mod
  # return value, default is sorted
  if(sorted==FALSE){
    return (vector)
  } else{
    return (sort(vector))
  }
}
```

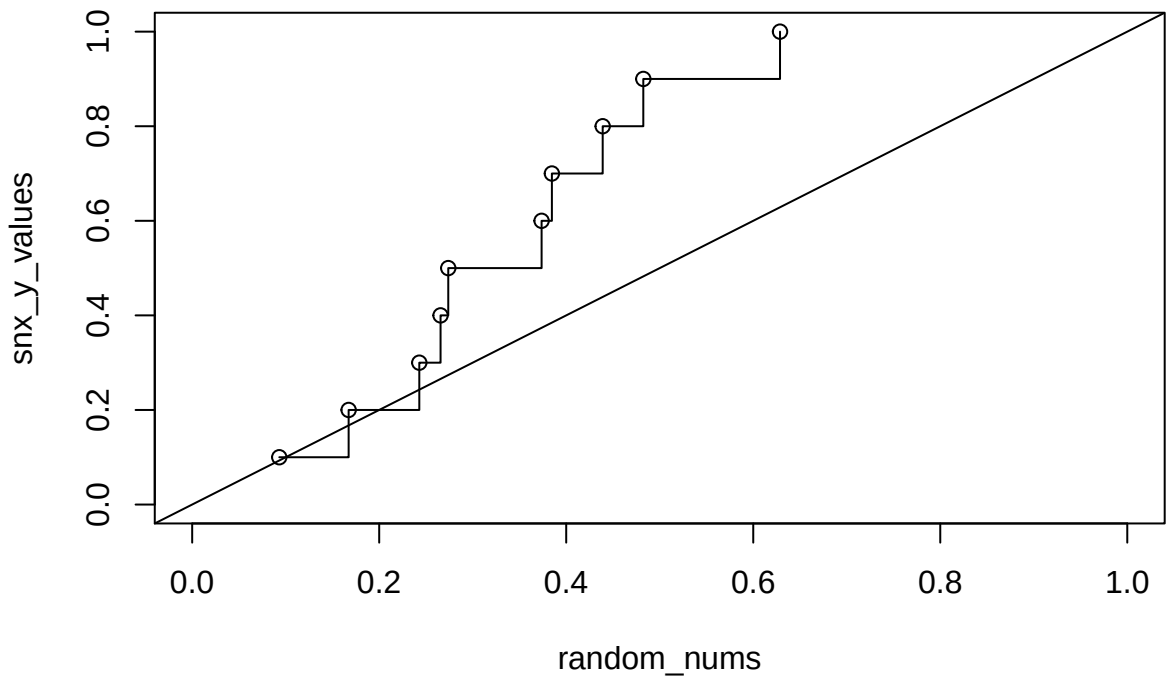
```
## plots the Sn(x) function (might need to double check this)
## woah this isn't snx .....

#snx <- sort(uniform_generator(N, 200000000))
#x <- seq.int(1,N) / N
#plot(snx ~ x, type="S")

# real snx
## snx functon values
snx_y_values <- seq.int(1, N)/N
random_nums <- sort(uniform_generator(N, seed = 200000000))

func_values <- data.frame(random_nums = random_nums,
                          low_y = seq.int(0, N-1)/N,
                          high_y = seq.int(1, N)/N)
func_values$low_diff = abs(func_values$random_nums - func_values$low_y)
func_values$high_diff = abs(func_values$random_nums - func_values$high_y)
kable(func_values)
```

random_nums	low_y	high_y	low_diff	high_diff
0.0931323	0.0	0.1	0.0931323	0.0068677
0.1673224	0.1	0.2	0.0673224	0.0326776
0.2428972	0.2	0.3	0.0428972	0.0571028
0.2656181	0.3	0.4	0.0343819	0.1343819
0.2738519	0.4	0.5	0.1261481	0.2261481
0.3736638	0.5	0.6	0.1263362	0.2263362
0.3846899	0.6	0.7	0.2153101	0.3153101
0.4390066	0.7	0.8	0.2609934	0.3609934
0.4824338	0.8	0.9	0.3175662	0.4175662
0.6286332	0.9	1.0	0.2713668	0.3713668

```
supx <- max(max(func_values$low_diff), max(func_values$high_diff))
supx
```

```
## [1] 0.4175662
```

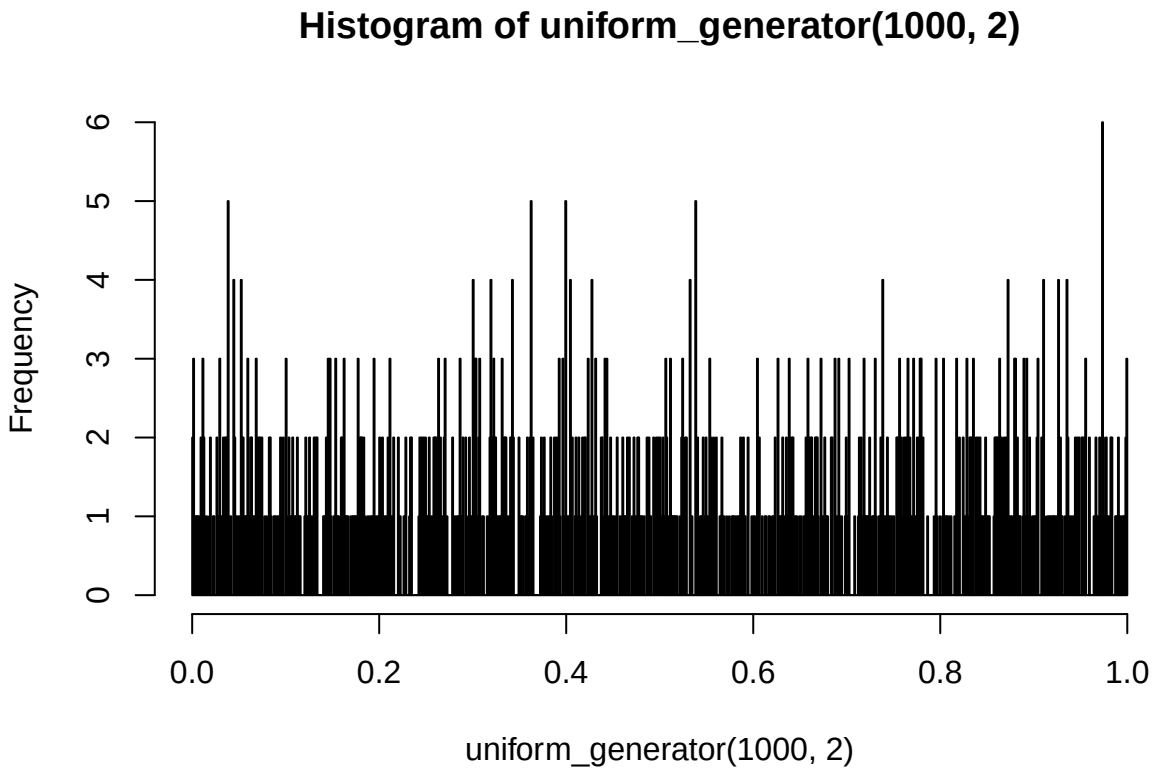
```
## this may or may not make sense to put into the k test, or make
## the k test call this and put this in some other function

plot(snx_y_values ~ random_nums, type="s", xlim=c(0,1), ylim=c(0,1))
points(random_nums, snx_y_values)
abline(0,1)
```



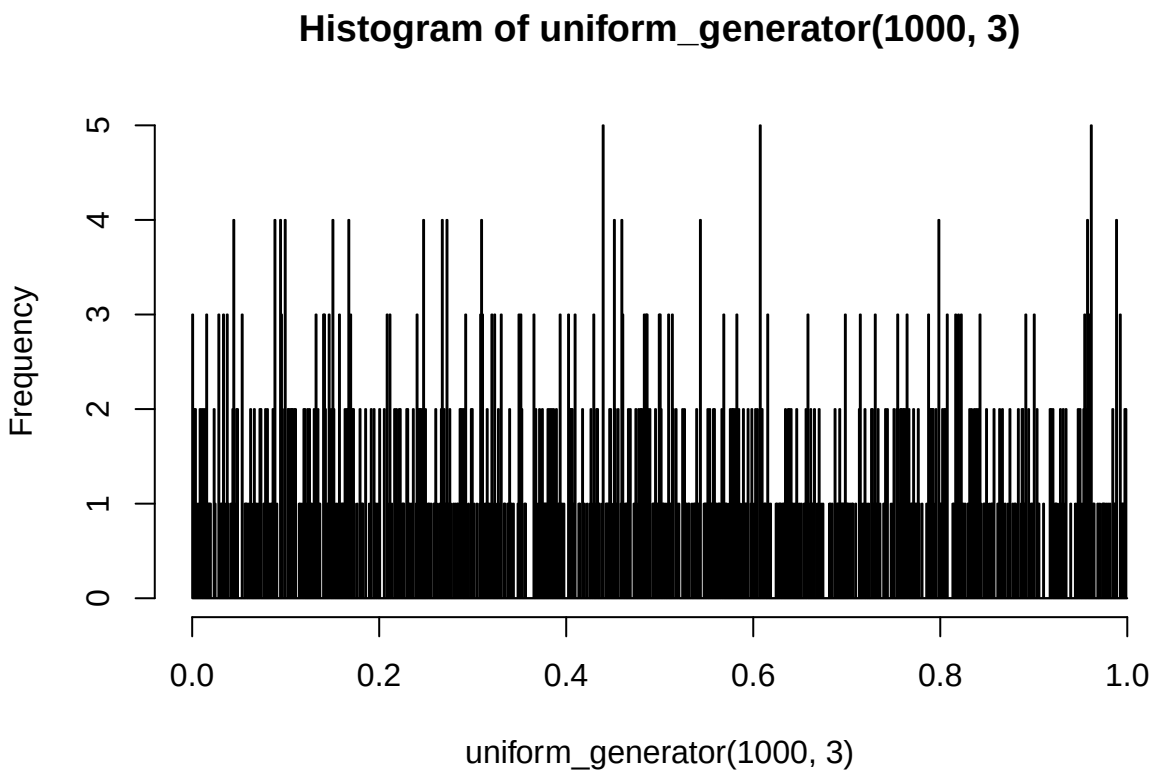

```
## notes: need to add (0,0) to this plot?
# add labels, legend, colors, and filled in and open points
```

```r
# K test for uniform
kTest <- function(random_numbers){
  random_numbers <- sort(random_numbers)



  if(){
    return (TRUE)
  } else {
    return (FALSE)
  }
}
```

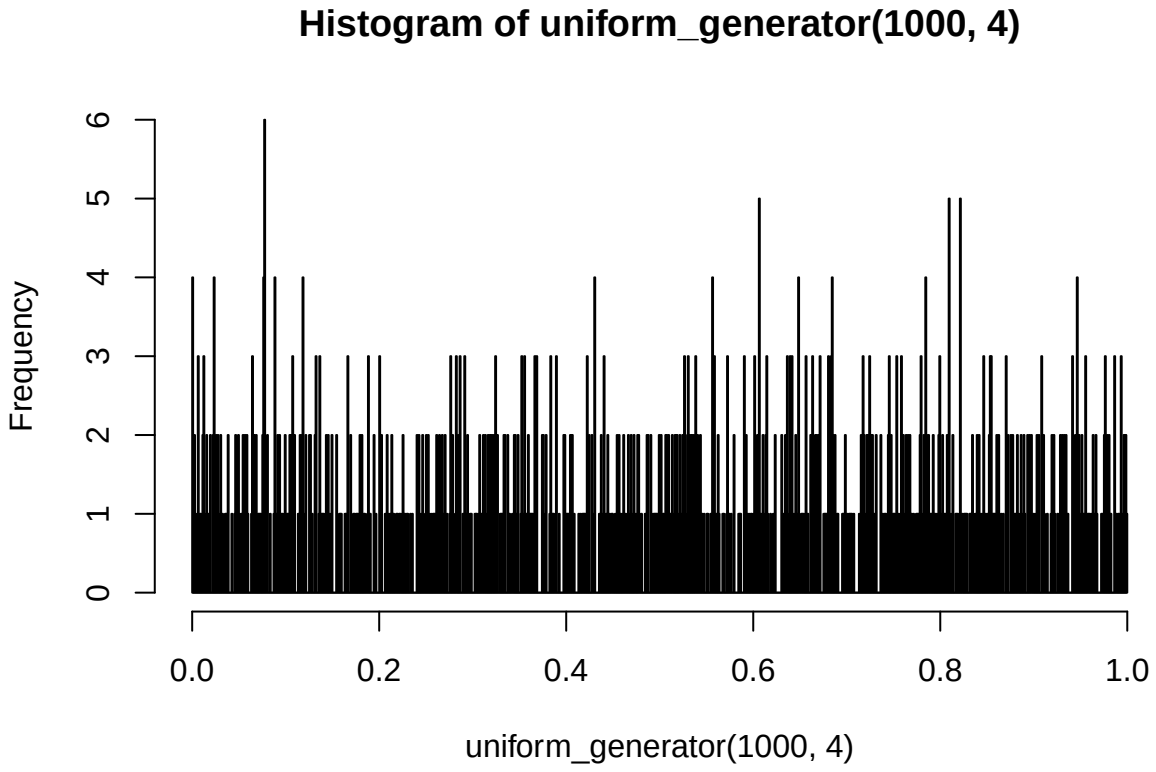
```r
# plot histograms to show distribution of random numbers, look for gaps
hist(uniform_generator(1000, 2), breaks=seq.int(0,1000)/1000)
```




```r
hist(uniform_generator(1000, 3), breaks=seq.int(0,1000)/1000)
```




```
hist(uniform_generator(1000, 4), breaks=seq.int(0,1000)/1000)
```



```r
hist(uniform_generator(1000, 5), breaks=seq.int(0,1000)/1000)
hist(uniform_generator(1000, 6), breaks=seq.int(0,1000)/1000)
hist(uniform_generator(1000, 7), breaks=seq.int(0,1000)/1000)
```

part 2

- what percentage of the 2 billion seeds fail the test when n=1000 ?
  (take a random (as large as possible) sample and make a confidence inteval for this percentage)