STAT 206 Lab 1

Due Monday, October 10, 5:00 PM

General instructions for labs: You are encouraged to work in pairs to complete the lab. Labs must be completed as an R Markdown file. Be sure to include your lab partner (if you have one) and your own name in the file. Give the commands to answer each question in its own code block, which will also produce plots that will be automatically embedded in the output file. Each answer must be supported by written statements as well as any code used.

Agenda: Manipulating data objects; using built-in functions, doing numerical calculations, and basic plots; reinforcing core probabilistic ideas.

Background

The exponential distribution is defined by its cumulative distribution function

$$F(x) = 1 - e^{-\lambda x}$$

The R function `rexp` generates random variables with an exponential distribution.

`rexp(n=10, rate=5)`

produces 10 exponentially-distributed numbers with rate (λ) of 5. If the second argument is omitted, the default rate is 1; this is the "standard exponential distribution".

Part I

1. Generate 200 random values from the standard exponential distribution and store them in a vector `exp.draws.1`. Find the mean and standard deviation of `exp.draws.1`.

```r
exp.draws.1 = rexp(200) #This generates 200 random variables and stores them
#to the vector object exp.draws.1

mean(exp.draws.1) #This is the mean of the vector object exp.draws.1
```

```
## [1] 1.062164
```

```r
sd(exp.draws.1) #This is the standard deviation of the vector object exp.draws.1
```

```
## [1] 1.035224
```

2. Repeat, but change the rate to 0.1, 0.5, 5 and 10, storing the results in vectors called `exp.draws.0.1`, `exp.draws.0.5`, `exp.draws.5` and `exp.draws.10`.

```r
exp.draws.0.1 = rexp(200, .1) #This generates 200 random variables and stores them
#to the vector object exp.draws.0.1

mean(exp.draws.0.1) #This is the mean of the vector object exp.draws.0.1
```

```
## [1] 9.702434
```

```r
sd(exp.draws.0.1) #This is the standard deviation of the vector object exp.draws.0.1
```

```
## [1] 9.95286
```

```r
exp.draws.0.5 = rexp(200, .5) #This generates 200 random variables and stores them
#to the vector object exp.draws.0.5

mean(exp.draws.0.5) #This is the mean of the vector object exp.draws.0.5
```

```
## [1] 2.082297
```

```r
sd(exp.draws.0.5) #This is the standard deviation of the vector object exp.draws.0.5
```

```
## [1] 2.062606
```

```r
exp.draws.5 = rexp(200, 5) #This generates 200 random variables and stores them to
#the vector object exp.draws.5

mean(exp.draws.5) #This is the mean of the vector object exp.draws.5
```

```
## [1] 0.2190035
```

```r
sd(exp.draws.5) #This is the standard deviation of the vector object exp.draws.0.5
```

```
## [1] 0.2010422
```

```r
exp.draws.10 = rexp(200, 10) #This generates 200 random variables and stores them
#to the vector object exp.draws.10

mean(exp.draws.10) #This is the mean of the vector object exp.draws.10
```

```
## [1] 0.09131461
```

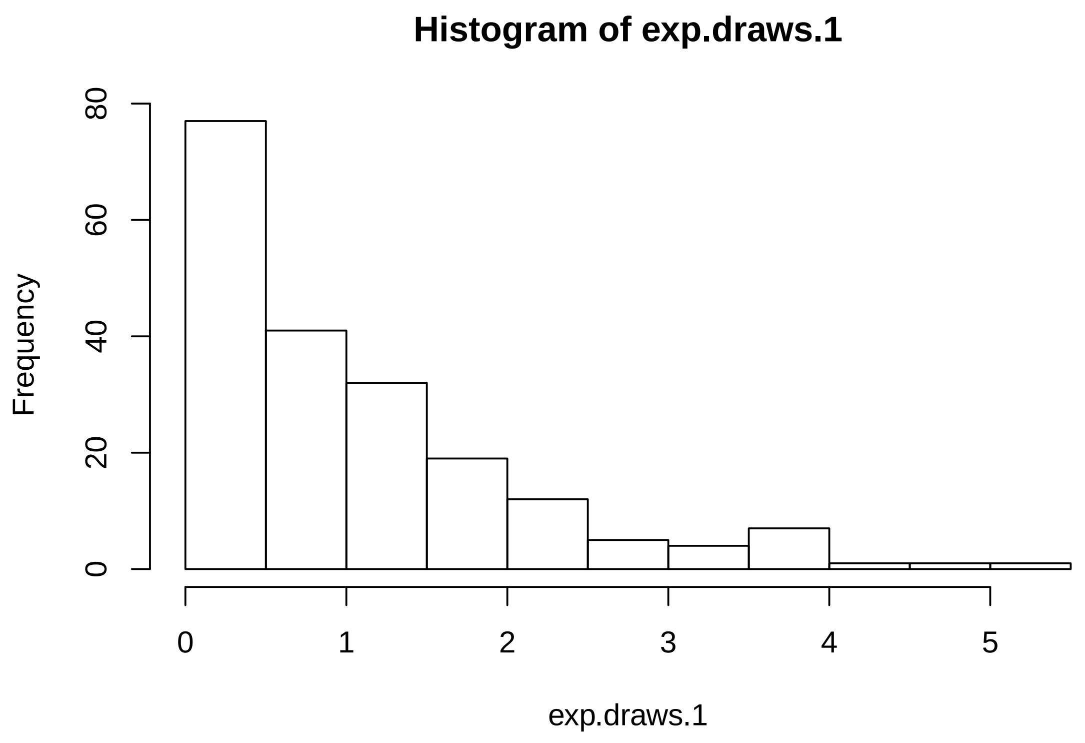
```r
sd(exp.draws.10) #This is the standard deviation of the vector object exp.draws.10
```

```
## [1] 0.0946446
```

3. The function `plot()` is the generic function in R for the visual display of data. `hist()` is a function that takes in and bins data as a side effect. To use this function, we must first specify what we'd like to plot.

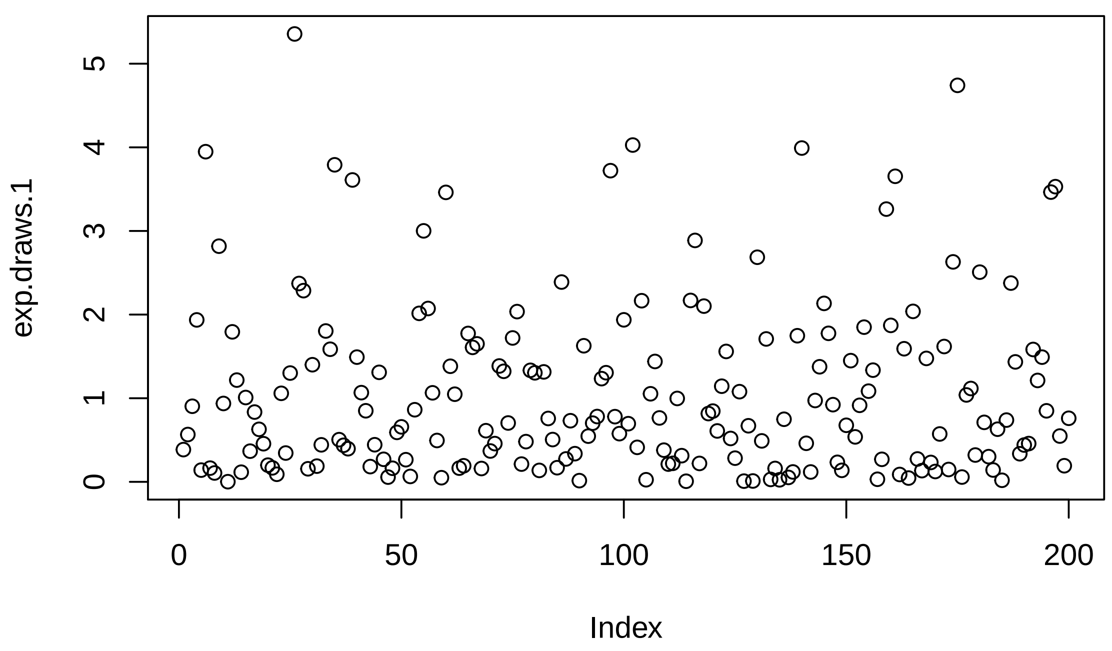
   a. Use the `hist()` function to produce a histogram of your standard exponential distribution.

   ```r
   hist(exp.draws.1) # calling the hist function and passing exp.draws1 as the argument
   ```



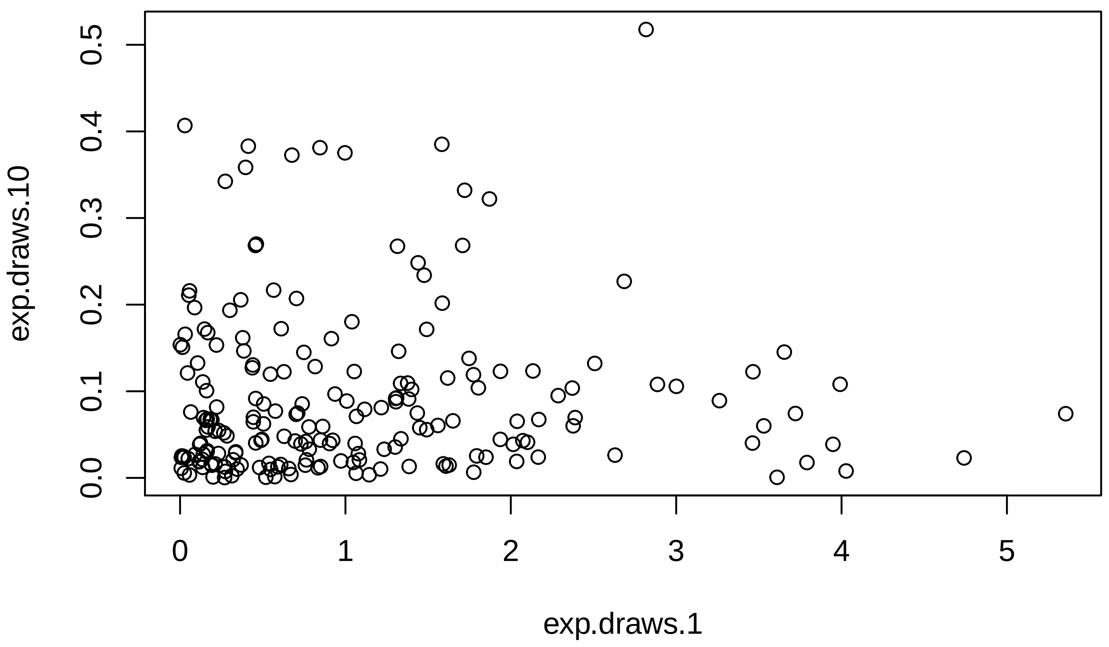
b. Use `plot()` with this vector to display the random values from your standard distribution in order.

```
plot(exp.draws.1) #creating a scatter plot for the same data as above
```



c. Now, use `plot()` with two arguments – any two of your other stored random value vectors – to create a scatterplot of the two vectors against each other.

```r
plot(exp.draws.1, exp.draws.10) # calling the plot fuction but passing in
```



```
#two different vectors as arguments
```

4. We'd now like to compare the properties of each of our vectors. Begin by creating a vector of the means of each of our five distributions in the order we created them and saving this to a variable name of your choice. Using this and other similar vectors, create the following scatterplots:

```
# below is a object vector to store the means of the vectors in order
five_means = c(mean(exp.draws.1), mean(exp.draws.0.1), mean(exp.draws.0.5), mean(exp.draws.5), mean(exp

#below is a object vector to store the rates of the vectors in order of creation
five_rates = c(1, 0.1, 0.5, 5, 10)
five_rates
```

```
## [1]  1.0  0.1  0.5  5.0 10.0
```

```
#below is an object vector to store the standard deviations of the vectors in order
five_sd = c(sd(exp.draws.1), sd(exp.draws.0.1), sd(exp.draws.0.5), sd(exp.draws.5), sd(exp.draws.10))
five_sd
```

```
## [1] 1.0352240 9.9528603 2.0626065 0.2010422 0.0946446
```

~~~
a. The five means versus the five rates used to generate the distribution.


```r
plot(five_means, five_rates)
~~~

![](CaseyKongpanickul_lab-01_files/figure-latex/unnamed-chunk-7-1.pdf)<!-- -->
```
```

~~~~
```r
# from the graph we can observe that as the mean grows the rate exponentially
#decreases. The rate (lambda) is a negative number so as we increase the magnitude
#of this negative number the output becomes smaller
```

b. The standard deviations versus the rates.


```r
plot(five_sd, five_rates)
```

![](CaseyKongpanickul_lab-01_files/figure-latex/unnamed-chunk-8-1.pdf)<!-- -->

```r
#this graph is equivalent to the one above. The only difference is that standard
#deviation is on the x axis instead of the mean. This means that the mean and standard
#deviation or equal to one another. However, the mean is 1/lambda so this means that
#the smaller the rate (lambda) is the larger the mean will be.
```

c. The means versus the standard deviations.


```r
plot(five_means, five_sd)
```

![](CaseyKongpanickul_lab-01_files/figure-latex/unnamed-chunk-9-1.pdf)<!-- -->

```r
# The graph of the mean vs the standard deviation shows a straight line with a
#slope of 1 based on a visual observation. This means that sd = mean which we
#know to be true because mean = 1/lambda = standard deviation
```
~~~~

For each plot, explain in words what's going on.

Part II

5. R's capacity for data and computation is large to what was available 10 years ago.
    a. To show this, generate 1.1 million numbers from the standard exponential distribution and store them in a vector called `big.exp.draws.1`. Calculate the mean and standard deviation.

```r
big.exp.draws.1 = rexp(1100000) #generating 1.1M numbers from the standard
#exponential distribution (leaving the second argument blank i.e. default value = 1 for rate)

mean(big.exp.draws.1) # calling mean function to calculate mean
```

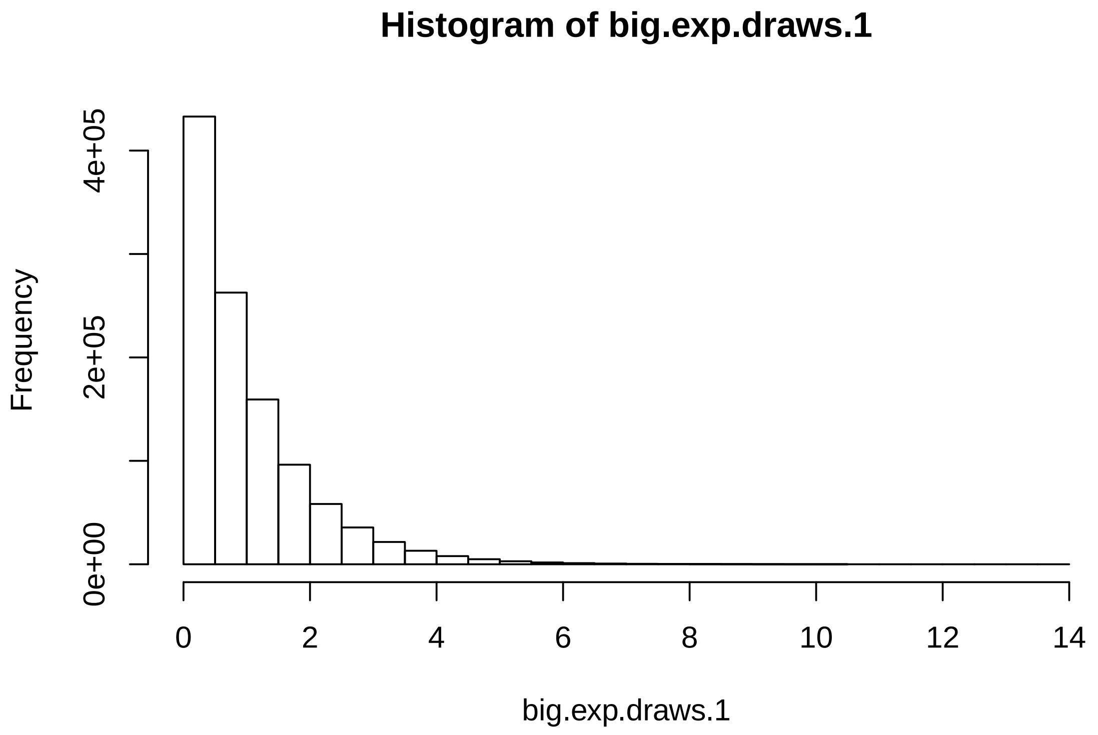
```
## [1] 1.000217
```

```r
sd(big.exp.draws.1) #calling standard deviation function to calculate the standard deviation
```

```
## [1] 1.001361
```

    b. Plot a histogram of `big.exp.draws.1`. Does it match the function $1 - e^{-x}$? Should it?

```r
hist(big.exp.draws.1)
```



```r
range(big.exp.draws.1)
```

```
## [1] 1.445534e-06 1.350305e+01
```

```r
# this histogram looks essentially the same however without the lambda is
#a modifier of the exponent and will tune the shape of the graph. So, it may
#appear to be a match but it should not be.
```

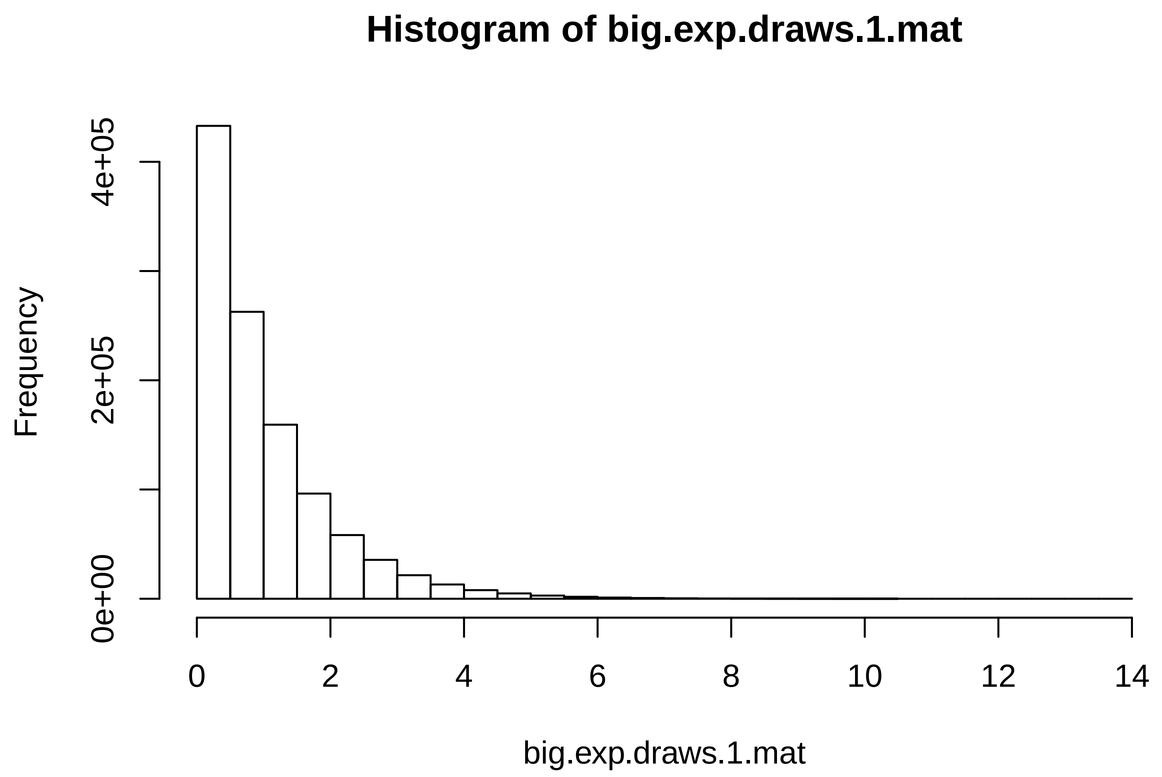
c. Find the mean of all of the entries in `big.exp.draws.1` which are strictly greater than 1. You may need to first create a new vector to identify which elements satisfy this.

```r
larger_than_1 = big.exp.draws.1[big.exp.draws.1>1] # created a object vector
#that draws from big.exp.draws.1 and takes a subset of the vector specifying
#values only larger than 1


mean(larger_than_1) #calling the mean fuction for the subset of data from above
```

```
## [1] 2.001116
```

d. Create a matrix, `big.exp.draws.1.mat`, containing the the values in `big.exp.draws.1`, with 1100 rows and 1000 columns. Use this matrix as the input to the `hist()` function and save the result to a variable of your choice. What happens to your data?

```r
big.exp.draws.1.mat = matrix(big.exp.draws.1, 1100, 1000) #calling the matrix function
#with input data of big.exp.draws.1 and specifying that there will be 1100 rows and
#1000 columns

big_mat_hist = hist(big.exp.draws.1.mat) #creating a histogram from the newly created
```



```
#matrix and saving it to an object

#the data has been put into a 1100x1000 matrix the values are placed by column first
#so the first. the data is now accessible by row and column instead of just index value
#which is one of the differences between vectors and matrices
```

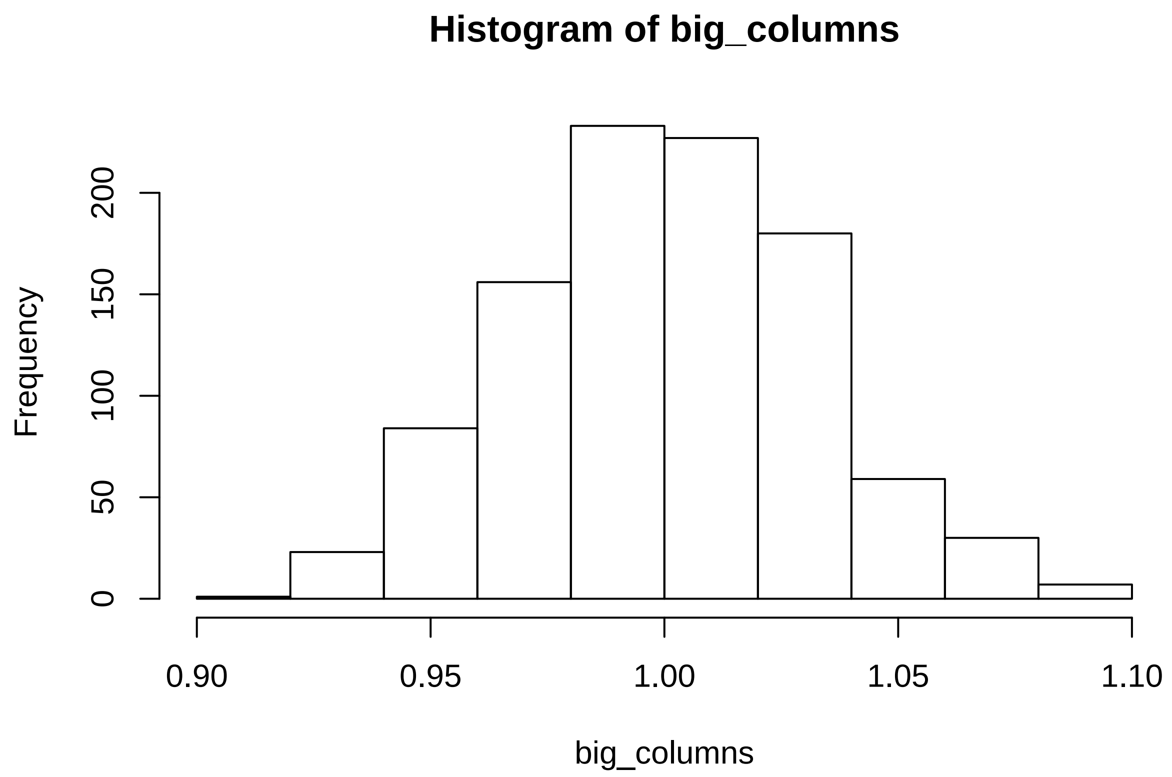
e. Calculate the mean of the 371st column of `big.exp.draws.1.mat`.

```
    column_371 = big.exp.draws.1.mat[,371] #creating a vector the the 371st column
    #of the matrix
    mean(column_371) #calling for the mean of the column 371 matrix
```

```
## [1] 1.047698
```

f. Now, find the means of all 1000 columns of `big.exp.draws.1.mat` simultaneously. Plot the histogram

```
    big_columns = colMeans(big.exp.draws.1.mat) #colMeans is a function to calculate
    #the mean values per column in a matrix

    hist(big_columns) #generates a histogram for the mean of the columns
```



```
# this histogram is normally distributed because it is plotting the means of various
#samples from the data which is the central limit theorm
```

```
g. Take the square of each number in `big.exp.draws.1`, and find the mean of this new vector.  Explain
```

```
big_squares = big.exp.draws.1 ^ 2 #squaring each value in the big.exp.draws.1 vector
big_square_mean = mean(big_squares) #calling for the mean of all squared values
big_square_mean #displaying mean of squared values
```

```
## [1] 2.003156
```

```
sd(big.exp.draws.1) #displaying standard deviation
```

```
## [1] 1.001361
```

```
mean(big.exp.draws.1) #mean of big.exp.draws.1
```

```
## [1] 1.000217
```

```
#The average of squared values is an element of the variance. From this average if
#we subtrace the squared average of the data we have the variance, and of course taking
#the square root of that gives us the standard deviation
```