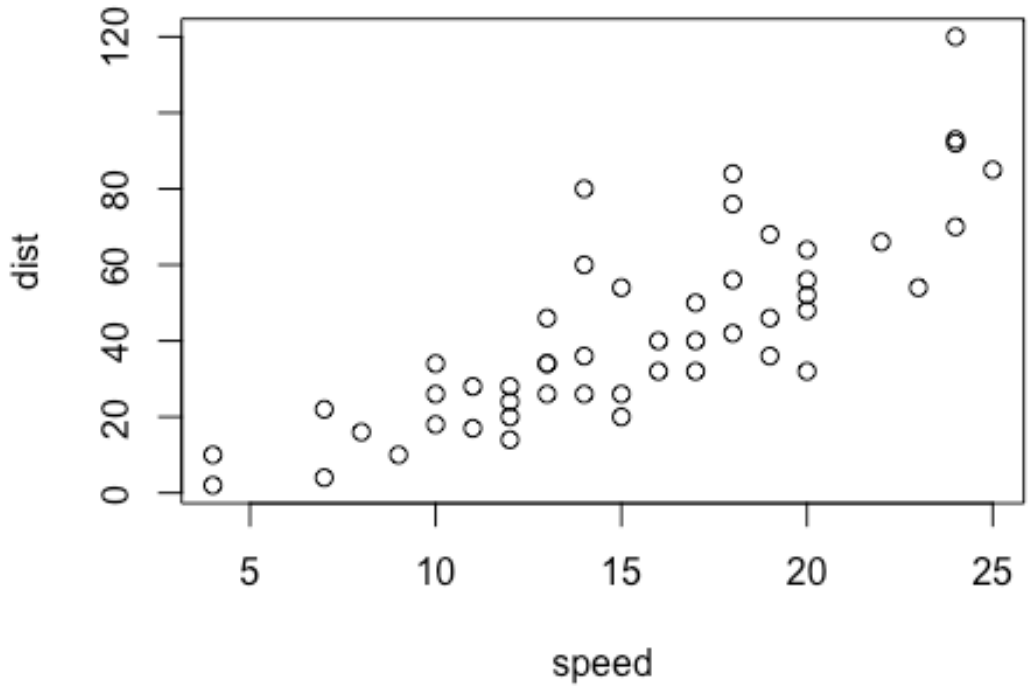
Sydney Bruce QBIO310 HW#1

This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.

```
plot(cars)
```



Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Cmd+Option+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Cmd+Shift+K* to preview the HTML file).

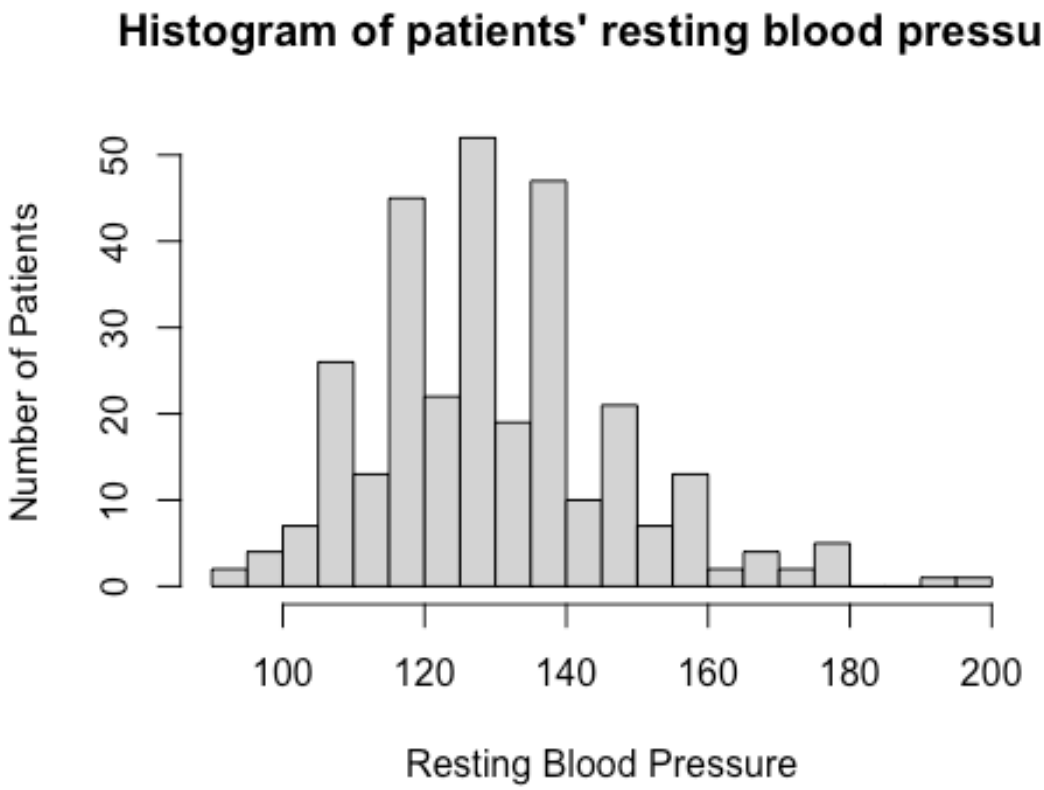
The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed.

Loading the file and making sure its loaded properly

```
clevheart = read.csv("ClevelandHeart.csv")
dim(clevheart)

## [1] 303  12
```

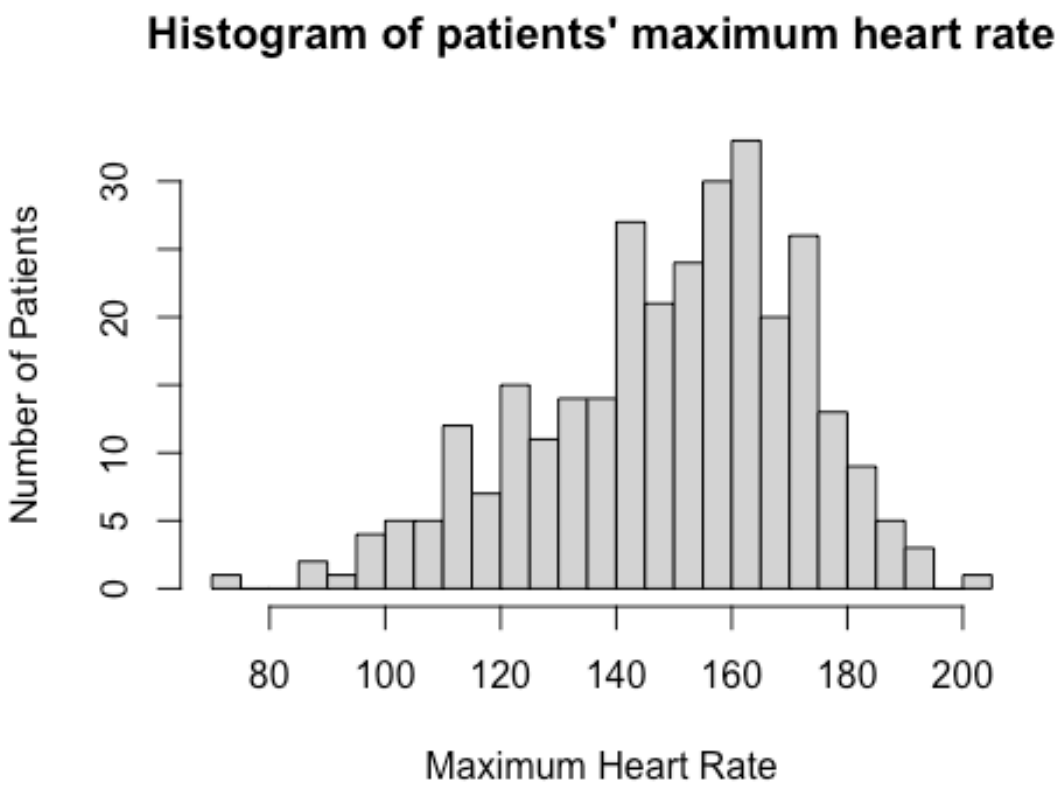
Plotting a histogram of the patients' resting BP

```
hist(clevheart$RestBP,xlab="Resting Blood Pressure",ylab="Number of
Patients",main="Histogram of patients' resting blood pressure",breaks=20)
```



Plotting a histogram of the patients' maximum HR

```
hist(
  clevheart$MaxHR,
  xlab="Maximum Heart Rate",
  ylab="Number of Patients",
  main="Histogram of patients' maximum heart rate",
  breaks=20)
```



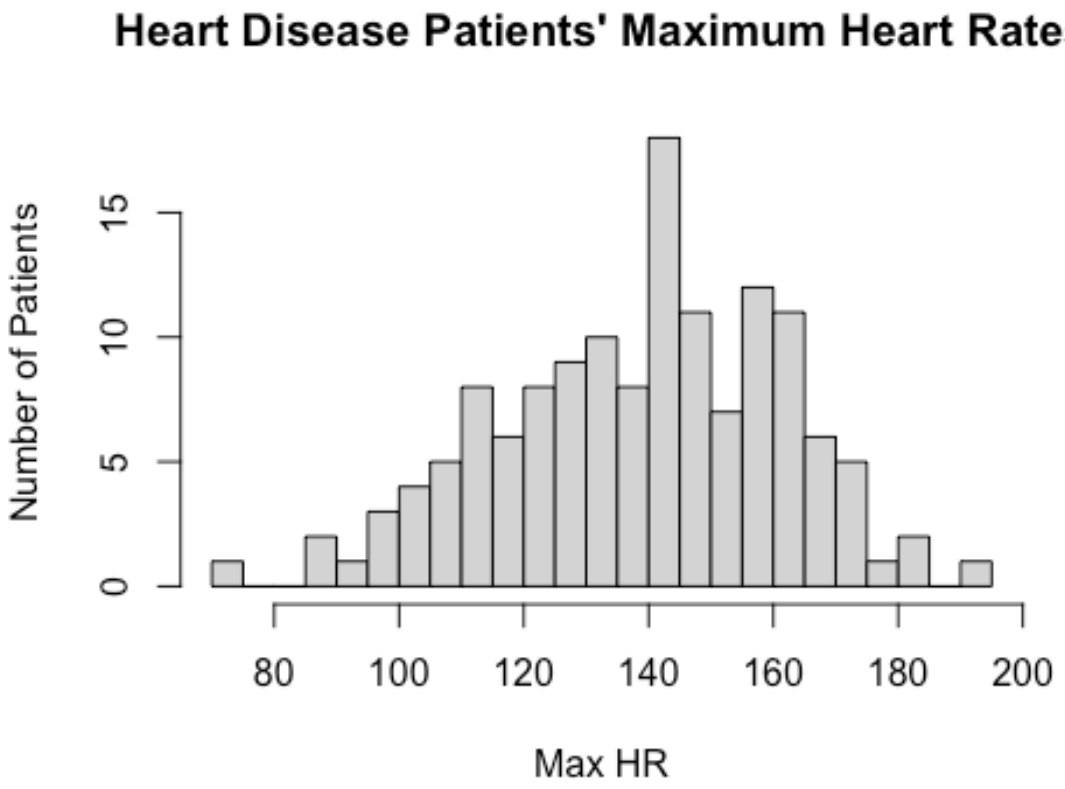
Displaying how many pts in dataset, how many have heart disease + fraction, as well as a table displaying chest pain types for negative/positive heart disease patients Trends between negative/positive heart disease diagnoses: Patients without diagnosed heart disease have much higher rates of nonanginal and nontypical chest pain, while patients with diagnosed heart disease have much higher rates of asymptomatic chest pain.

```
num_pts <- nrow(clevheart)
cat("Number of patients: ",num_pts,"\n")

## Number of patients:  303

hd_pts <- clevheart[clevheart$AHD == "Yes",]
nhd_pts <- nrow(hd_pts)
cat("Number of patients with diagnosed heart disease: ",nhd_pts,"\n")

## Number of patients with diagnosed heart disease:  139

fhd_pts <- nhd_pts/num_pts
cat("Fraction of patients with diagnosed heart disease: ",fhd_pts,"\n")

## Fraction of patients with diagnosed heart disease:  0.4587459

table(clevheart$AHD == "Yes",clevheart$ChestPain)
```

```
##
##         asymptomatic nonanginal nontypical typical
##   FALSE           39         68         41      16
##   TRUE           105         18          9       7
```

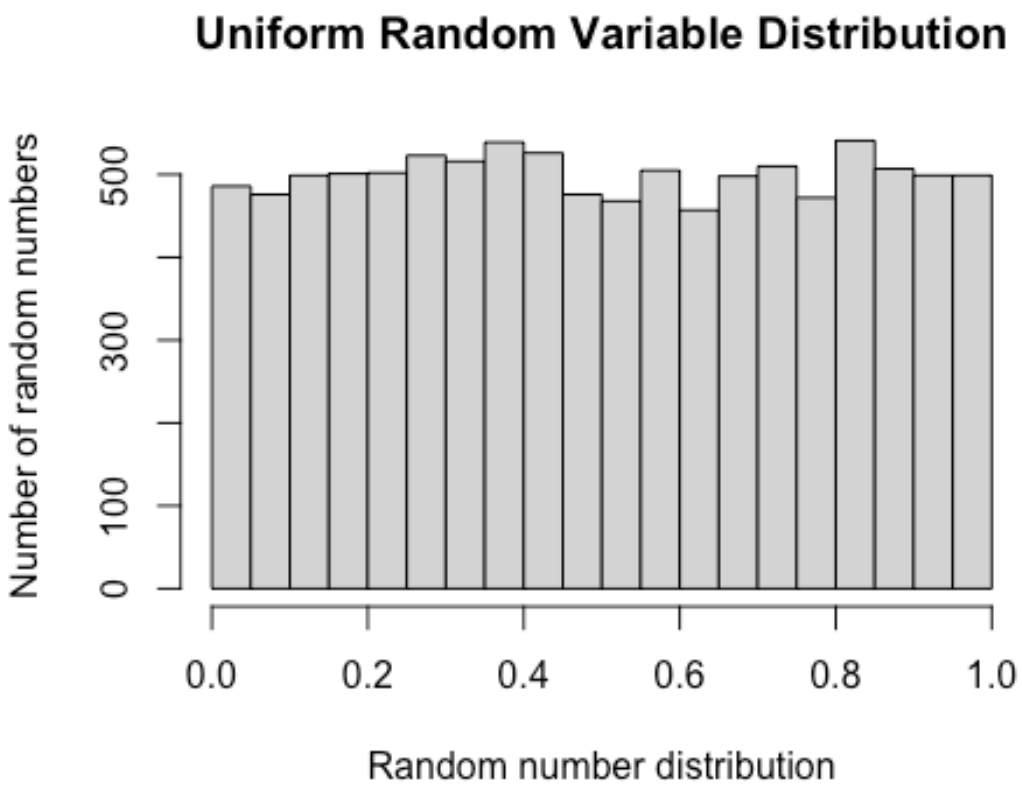
Plotting a histogram with the maximum HR for pts w/ HD:

```
hist(
  clevheart$MaxHR[clevheart$AHD=="Yes"],
  xlab="Max HR",
  ylab="Number of Patients",
  main="Heart Disease Patients' Maximum Heart Rates",
  breaks=20)
```



2a. Making a histogram of 10,000 uniform random variables

```
x=runif(10000)
hist(
  x,
  xlab="Random number distribution",
  ylab="Number of random numbers",
  main="Uniform Random Variable Distribution",
  breaks=20) # 2a
```



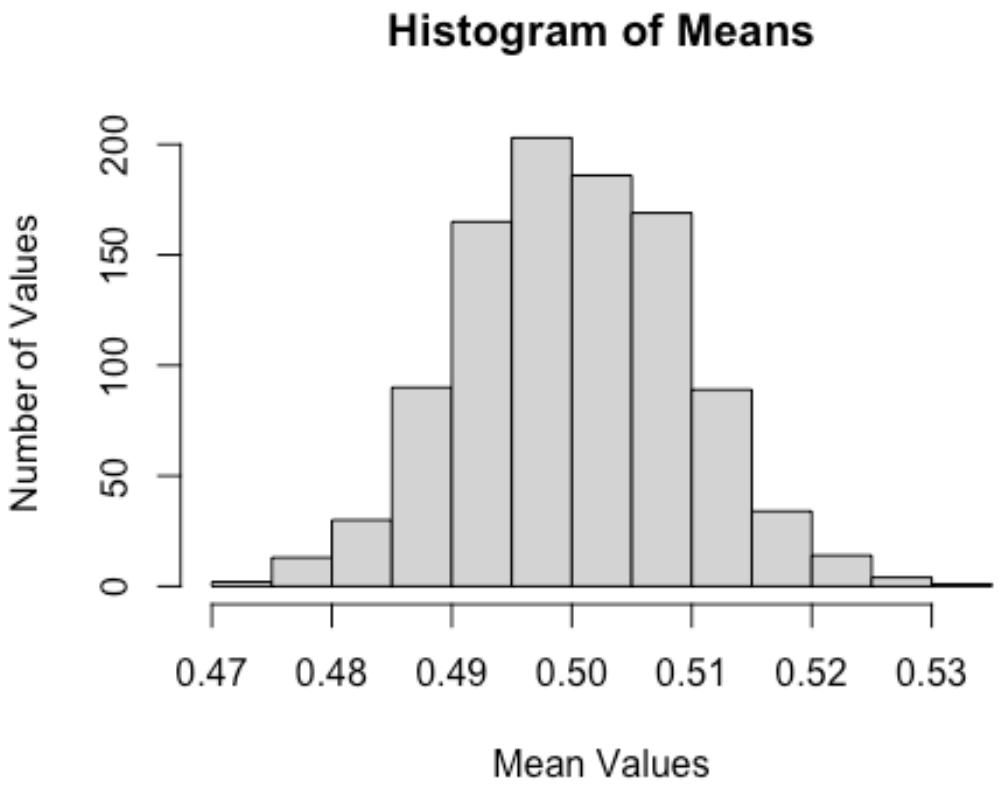
2b. Writing a function for and taking the mean of a vector of data (URVs) and a vectors of thresholds(defined)

```
thresholds <- c(0.25,0.5,0.75,1.0)
below_threshold <- function(data,thresholds) {
  sapply(thresholds,function(thresh) {
    # sapply APPLIES the thresholds vector to the function and iterates
through thresh
    mean(data<=thresh) # taking the mean of data less than or equal to the
current thresh number
    # <= to it, then taking the mean of the returned true values (default)
  })
}
fractions<-below_threshold(x,thresholds)
fractions # 2b

## [1] 0.2464 0.5044 0.7482 1.0000
```

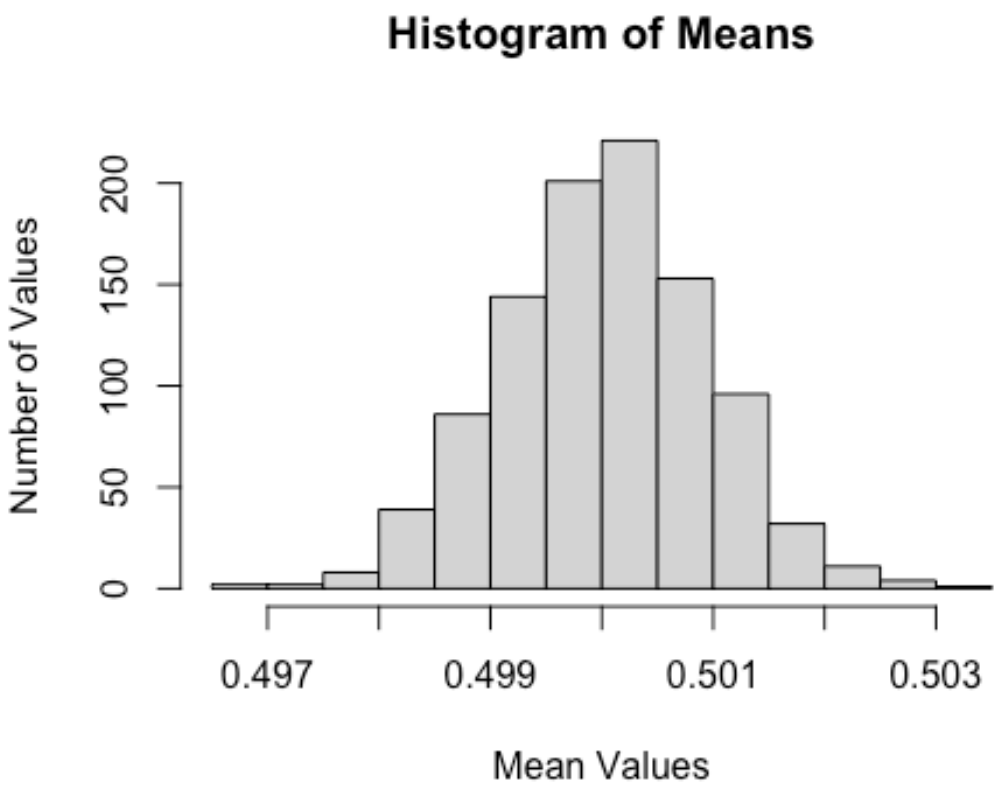
2c. Writing a function for two numbers, m & n…fxn will output an n-length vector…for each element of the return vector the fxn will simulate m URVs and compute the mean

```
simulate_means <- function(m,n) {
  means<-sapply(1:n, function(i) {
    # appyling 1 through n to the function and iterating through i
```

```r
    # for 1 through n, simulate m random variables and compute the mean
    mean(runif(m)) # taking the mean of the simulated uniform distribution
  })
  hist(
    means,
    xlab="Mean Values",
    ylab="Number of Values",
    main="Histogram of Means",
    breaks=20)
  return(means)
}
out1 <- simulate_means(1000,1000)
```



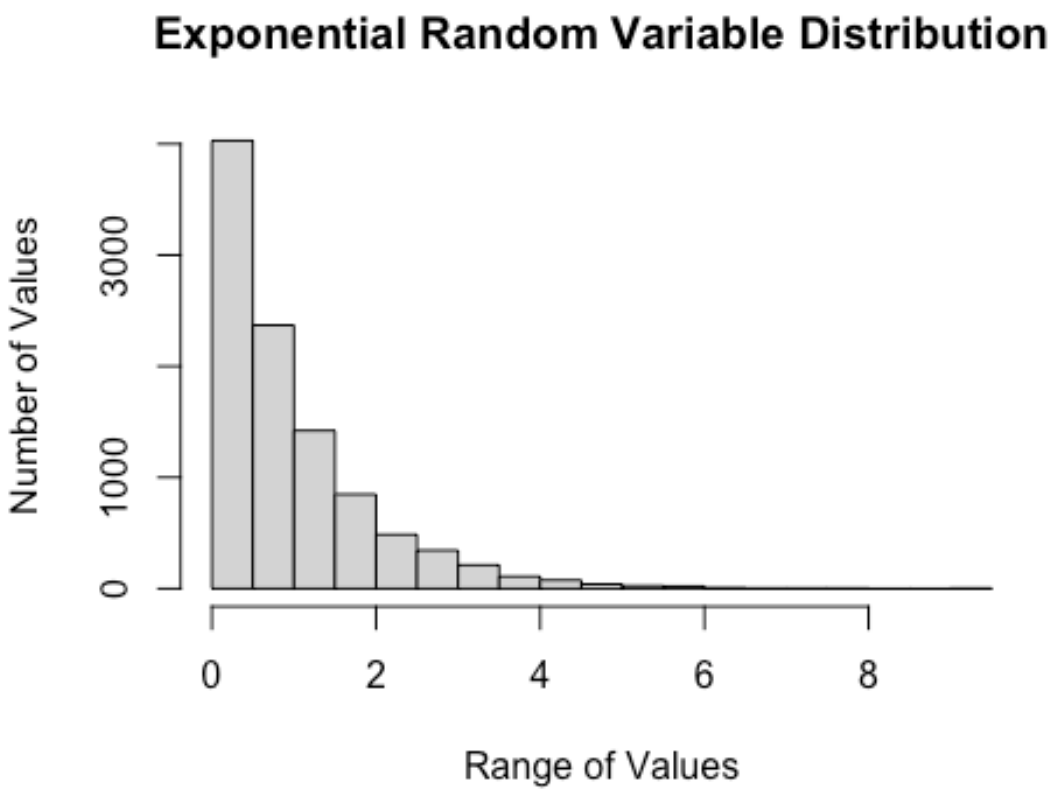

```r
out2 <- simulate_means(100000,1000) # 2c
```




```
# When m increases, the x-axis range decreases and the variability of the
# mean decreases because of the more concentrated distribution
```

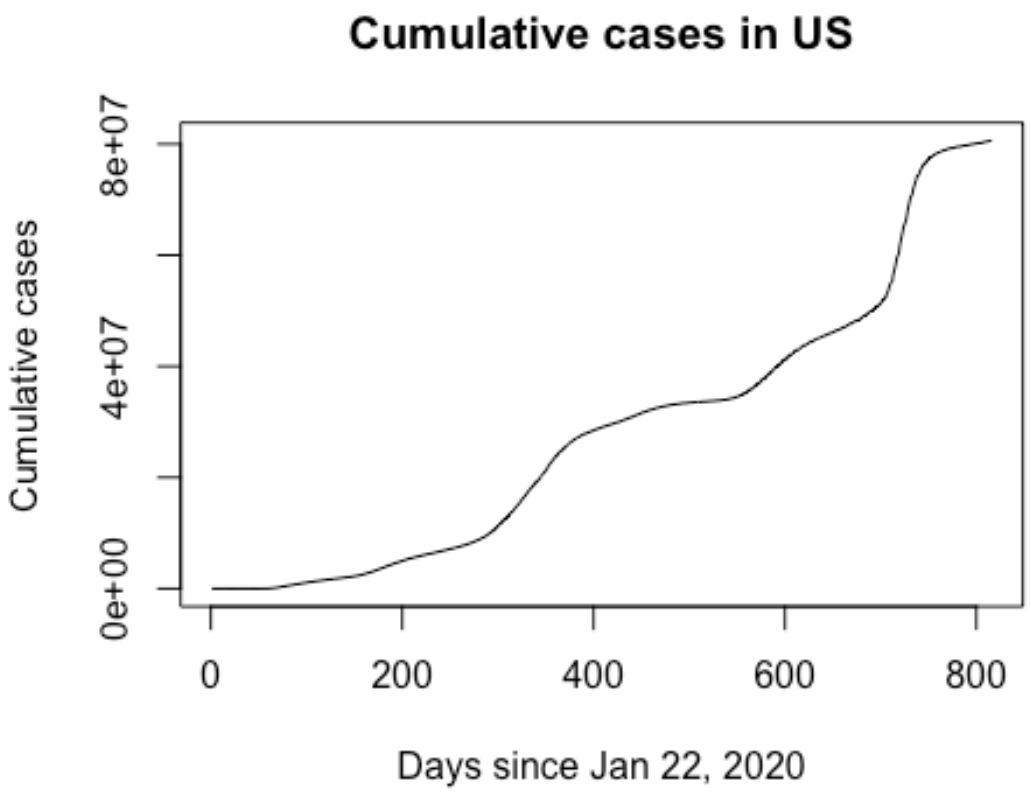
2d. Simulating 10,000 exponential random variables with rate=1 then plot histogram

```
y=rexp(10000,rate=1) # simulate exponential random variables with rate of 1
hist(
  y,
  xlab="Range of Values",
  ylab="Number of Values",
  main="Exponential Random Variable Distribution",
  breaks=20) # 2d
```



Problem 3 3a.

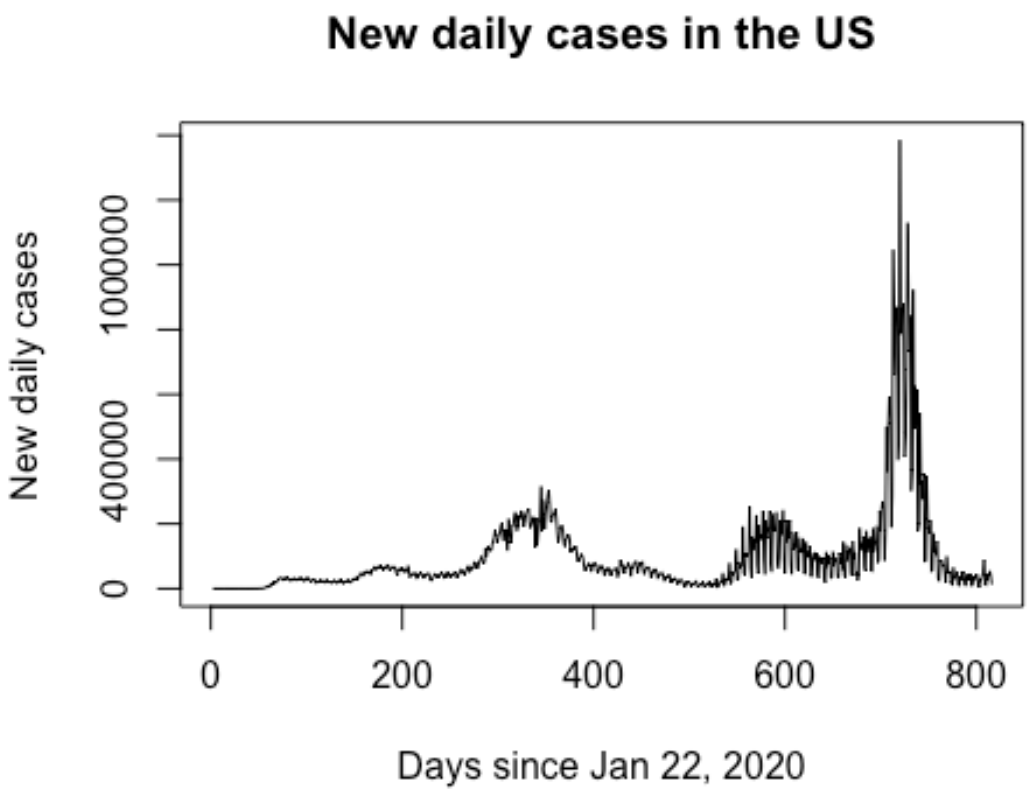
```
cv<-read.csv("key-countries-pivoted.csv")
dim(cv)

## [1] 816   9

plot(
  cv$US,
  type="l",
  xlab="Days since Jan 22, 2020",
  ylab="Cumulative cases",
  main="Cumulative cases in US")  # 3a
```




```
# type="l" sets the plot to be a line graph
```

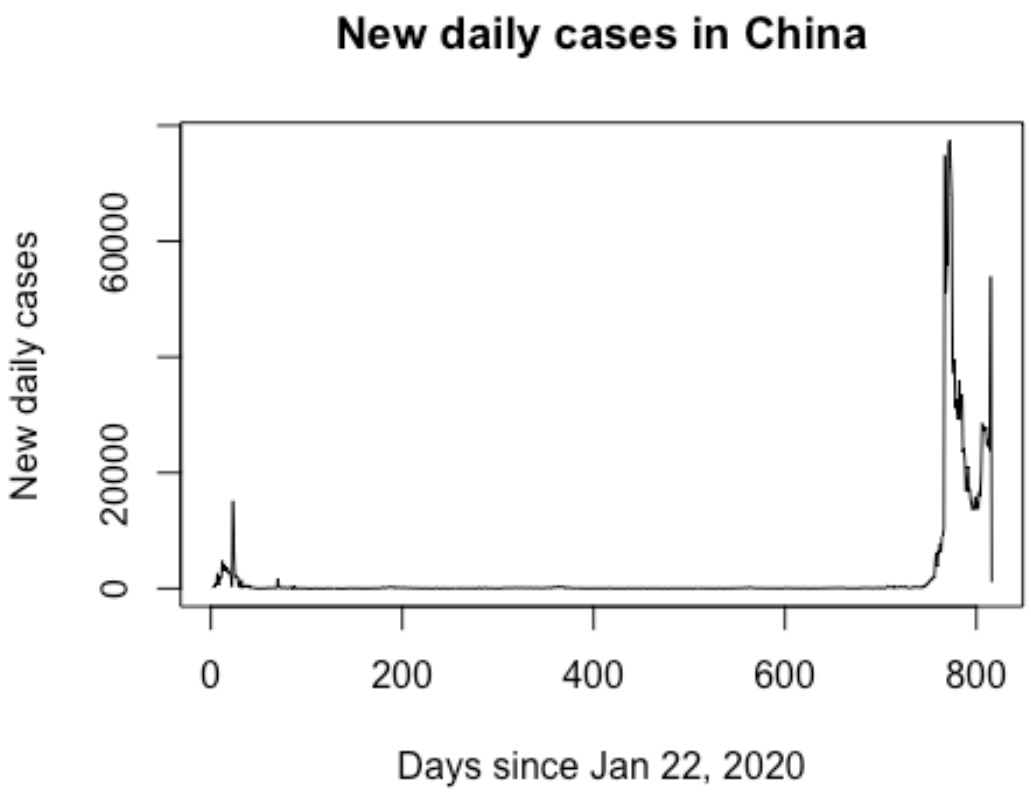
3b.

```
new_cases_US=c(NA,diff(cv$US))
#print(new_cases_US)
plot(
  new_cases_US,
  type="l",
  xlab="Days since Jan 22, 2020",
  ylab="New daily cases",
  main="New daily cases in the US") # 3b
```



3c.

```r
new_cases_C=c(NA,diff(cv$China)) # c to combine (making a new vector), diff steps forward but computes backwards...need NA to ignore 1st value so NA isn't returned in data (if we start at 1 it can't take the difference between 1 and null)
plot(
  new_cases_C,
  type="l",
  xlab="Days since Jan 22, 2020",
  ylab="New daily cases",
  main="New daily cases in China") # 3c
```



3d.

```
ap16<-which(cv$Date=="2022-04-16") # creating quick call
total_cases_ap16=colSums(cv[ap16,2:ncol(cv)])
# Need to exclude date column (1)...call won't work because not numeric same
type
print(total_cases_ap16) # Prints total cases for each country because of
colSums above
```

```
##          China             US United_Kingdom          Italy         France
##        1760211       80625120       21916961       15659835       27874269
##        Germany          Spain           Iran
##       23416663       11627487        7205064
```

```
fractions=total_cases_ap16/sum(total_cases_ap16) # Prints total cases for
each
# country because is vector ("list" of names that equal a value)
print(fractions)
```

```
##          China             US United_Kingdom          Italy         France
##    0.009260096    0.424151623    0.115300474    0.082383064    0.146640606
##        Germany          Spain           Iran
##    0.123190088    0.061169738    0.037904311
```

```r
populations<-c( # setting population numbers for each country
  China=1412000000,
  US=332000000,
  United_Kingdom=67000000,
  Italy=59000000,
  France=68000000,
  Germany=83000000,
  Spain=47000000,
  Iran=88000000
)

case_per_country<-total_cases_ap16/populations
# does on per country standard bcz both are vectors
cases_table<-data.frame( # creates new data.frame to present the information
  Country=names(case_per_country),
  FractionTotal=as.numeric(fractions),
  CasesPerCapita=as.numeric(case_per_country),
  Population=as.numeric(populations)
)
print(cases_table) # 3d
```

```
##          Country FractionTotal CasesPerCapita Population
## 1          China   0.009260096    0.001246608  1.412e+09
## 2             US   0.424151623    0.242846747  3.320e+08
## 3 United_Kingdom   0.115300474    0.327118821  6.700e+07
## 4          Italy   0.082383064    0.265420932  5.900e+07
## 5         France   0.146640606    0.409915721  6.800e+07
## 6        Germany   0.123190088    0.282128470  8.300e+07
## 7          Spain   0.061169738    0.247393340  4.700e+07
## 8           Iran   0.037904311    0.081875727  8.800e+07
```

```r
nrow(cv)
```

```
## [1] 816
```