Problem Set 1

CS 169 Problem Set 1 Due Friday, September 22, 2023

1. Modeling Alzheimer's Risk

a. (5 points) Read tab-delimited data from the file `ADpred.txt` into a data frame called `ADpred` in R. This file contains several cognitive and genetic test results for 50 patients being evaluated for the possible onset of Alzheimer's disease. Make sure to look at the data to see that you are reading in what you expect. (Hint: first visualize the contents of the text file itself to make sure you're not adding anything unexpected.)

To get a sense of the data, print the range of each of the variables (test results) in the data set. Show your code below:

```
file_path <- "ADpred.txt"
ADpred <- read.table(file_path, header = TRUE, sep = "\t")
summary(ADpred)
```

```
##    APOEstatus     digitspan         game          forms         finance
##  Min.   :0.00   Min.   : 0.0   Min.   :1.00   Min.   :0.00   Min.   :1.00
##  1st Qu.:0.00   1st Qu.: 3.0   1st Qu.:2.00   1st Qu.:0.00   1st Qu.:2.00
##  Median :1.00   Median : 4.0   Median :3.00   Median :1.00   Median :3.00
##  Mean   :0.82   Mean   : 4.7   Mean   :3.00   Mean   :1.08   Mean   :3.12
##  3rd Qu.:2.00   3rd Qu.: 6.0   3rd Qu.:4.75   3rd Qu.:2.00   3rd Qu.:4.00
##  Max.   :2.00   Max.   :11.0   Max.   :5.00   Max.   :2.00   Max.   :5.00
```

b. (2 points) Create a numeric vector called `multiplier` that we will use to apply weights to the various tests. The weights to be included are:

APOEstatus: 0.200

digitspan: 0.029

game: 0.004

forms: 0.180

finance: 0.004

```
 multiplier <- c(APOEstatus=0.200, digitspan=0.029, game=0.004, forms=0.180, finance=0.004)
multiplier
```

```
## APOEstatus  digitspan       game      forms    finance
##      0.200      0.029      0.004      0.180      0.004
```

c. (5 points) Create a numeric vector called `ADscore` containing the products of the test scores and the multipliers for all the patients in ADpred.

Note that the dot-product operator in R is `%*%`. E.g., the R commands

```
x <- c(5,2,3,4)
y <- c(2,8,4,6)
dot <- x %*% y
```

end with `dot` containing the value 62 (that is, `5*2 + 2*8 + 3*4 + 4*6`). You will need a numeric matrix, not a data frame, to do matrix multiplication. Look at the data.matrix() function for this.

To write better and more efficient R code, avoid using explicit loops. Write your code below.

```
matrix <- data.matrix(ADpred)
matrix
```

```
##    APOEstatus digitspan game forms finance
## 1           0         5    2     1       3
## 2           2         4    3     2       3
## 3           1         8    5     1       3
## 4           0         3    2     0       2
## 5           1        11    5     2       5
## 6           1         8    4     2       4
## 7           2         5    3     2       3
## 8           1         7    2     0       1
## 9           0         4    3     0       5
## 10          0         2    2     0       3
## 11          0         5    3     2       2
## 12          2         4    3     1       2
## 13          2         4    3     2       2
## 14          2         5    2     0       1
## 15          0         2    3     2       4
## 16          2         4    2     0       4
## 17          2         4    1     0       4
## 18          0        11    4     0       5
## 19          1         3    2     2       4
## 20          2         3    5     2       4
## 21          0         7    5     0       1
## 22          2         3    5     2       4
## 23          1         4    4     1       3
## 24          2         5    2     0       4
## 25          0         6    4     0       1
## 26          0         4    5     1       5
## 27          0         4    1     1       4
## 28          2         1    1     2       5
## 29          0         3    5     2       1
## 30          2         9    2     1       1
## 31          0         2    1     1       1
## 32          1         6    3     2       3
## 33          1         5    5     1       5
## 34          0         4    4     2       3
## 35          0         4    1     0       5
## 36          2         0    5     2       4
## 37          0         9    1     2       3
## 38          0         1    2     1       2
## 39          0         1    2     2       1
## 40          0         4    5     0       2
## 41          0         4    3     0       2
```

```
## 42          2         6    5     0       5
## 43          1         3    2     1       2
## 44          1         6    5     1       4
## 45          1         6    3     2       5
## 46          0         3    1     2       5
## 47          0         3    1     0       1
## 48          0         4    5     2       5
## 49          0         8    2     0       4
## 50          2         8    1     2       1
```

```
ADscore <- matrix %*% multiplier
ADscore
```

```
##     [,1]
## 1  0.345
## 2  0.900
## 3  0.644
## 4  0.103
## 5  0.919
## 6  0.824
## 7  0.929
## 8  0.415
## 9  0.148
## 10 0.078
## 11 0.525
## 12 0.716
## 13 0.896
## 14 0.557
## 15 0.446
## 16 0.540
## 17 0.536
## 18 0.355
## 19 0.671
## 20 0.883
## 21 0.227
## 22 0.883
## 23 0.524
## 24 0.569
## 25 0.194
## 26 0.336
## 27 0.316
## 28 0.813
## 29 0.471
## 30 0.853
## 31 0.246
## 32 0.758
## 33 0.565
## 34 0.504
## 35 0.140
## 36 0.796
## 37 0.637
## 38 0.225
## 39 0.401
## 40 0.144
```

```
## 41 0.136
## 42 0.614
## 43 0.483
## 44 0.590
## 45 0.766
## 46 0.471
## 47 0.095
## 48 0.516
## 49 0.256
## 50 1.000
```

d. (4 points) What are the lowest, highest, average, and median AD scores?

```
lowest <- min(ADscore)
lowest
```

```
## [1] 0.078
```

```
highest <-max(ADscore)
highest
```

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

```
average <- mean(ADscore)
average
```

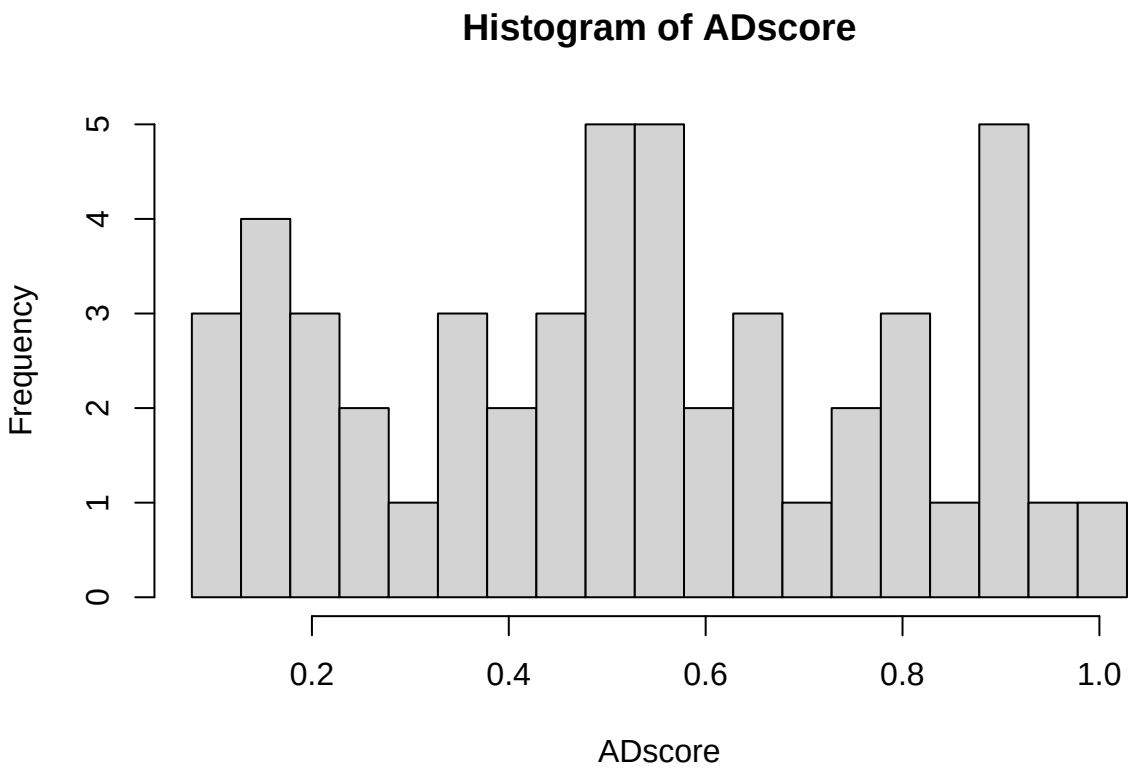
```
## [1] 0.51918
```

```
median <- median(ADscore)
median
```

```
## [1] 0.5245
```

e. (5 points) Using base R graphics, print a histogram of ADscore, with bins of size 0.05. (You can use `seq()` to exactly define breakpoints between the bins.)

```
hist(ADscore,breaks=seq(min(ADscore), max(ADscore)+0.05, by=0.05))
```



f. (5 points) APOE is one of the risk genes for Alzheimer's disease. There are typically 3 "alleles" (variants) of the gene; the "E4" allele is associated with a higher risk of developing Alzheimer's disease. A person may have 0, 1, or 2 copies of the E4 allele, corresponding to the values of 0-2 in the ADpred data frame.

Compute the empirical probability that a patient with two APOE4 alleles (i.e., a score of 2 under APOEstatus) has an AD score greater than 0.8. Compare this to the probability that someone with zero APOE4 alleles has an AD score greater than 0.8.

```
APOE4_2 <- sum(ADpred$APOEstatus == 2)
APOE4_2
```

```
## [1] 15
```

```
ADSCORE <- sum(ADpred$APOEstatus == 2 & ADscore > 0.8)
ADSCORE
```

```
## [1] 8
```

```
APOE4_0<- sum(ADpred$APOEstatus == 0)
APOE4_0
```

```
## [1] 24
```

```
ADSCORE2 <- sum(ADpred$APOEstatus == 0 & ADscore > 0.8)
ADSCORE2
```

```
## [1] 0
```

```
PROB1 <- ADSCORE / APOE4_2
PROB1
```

```
## [1] 0.5333333
```

```
prob2<-ADSCORE2 / APOE4_0
prob2
```

```
## [1] 0
```

```
*Write your answer here.*
The probability that has two APOE4 is 0.53333, and the probability that has o
APOE4 is 0.
```

g. (4 points) Create a new data frame, `ADrisk`, that contains a copy of `ADpred` with the patient ID number (from 1:50) in the first column (which should be named "patientID"), and the ADscore‘ in the last column. Sort ADrisk by ADscore, so that the highest scoring patients are at the top.

```
ADrisk <- data.frame(ADpred)
ADrisk <- data.frame(patientID = 1:50, ADrisk)
ADrisk$ADscore <- ADscore
ADrisk <- ADrisk[order(-ADrisk$ADscore), ]
ADrisk
```

```
##    patientID APOEstatus digitspan game forms finance ADscore
## 50        50          2         8    1     2       1   1.000
## 7          7          2         5    3     2       3   0.929
## 5          5          1        11    5     2       5   0.919
## 2          2          2         4    3     2       3   0.900
## 13        13          2         4    3     2       2   0.896
## 20        20          2         3    5     2       4   0.883
## 22        22          2         3    5     2       4   0.883
## 30        30          2         9    2     1       1   0.853
## 6          6          1         8    4     2       4   0.824
## 28        28          2         1    1     2       5   0.813
## 36        36          2         0    5     2       4   0.796
## 45        45          1         6    3     2       5   0.766
## 32        32          1         6    3     2       3   0.758
## 12        12          2         4    3     1       2   0.716
## 19        19          1         3    2     2       4   0.671
## 3          3          1         8    5     1       3   0.644
## 37        37          0         9    1     2       3   0.637
## 42        42          2         6    5     0       5   0.614
## 44        44          1         6    5     1       4   0.590
## 24        24          2         5    2     0       4   0.569
## 33        33          1         5    5     1       5   0.565
```

```
## 14        14          2         5    2     0       1   0.557
## 16        16          2         4    2     0       4   0.540
## 17        17          2         4    1     0       4   0.536
## 11        11          0         5    3     2       2   0.525
## 23        23          1         4    4     1       3   0.524
## 48        48          0         4    5     2       5   0.516
## 34        34          0         4    4     2       3   0.504
## 43        43          1         3    2     1       2   0.483
## 46        46          0         3    1     2       5   0.471
## 29        29          0         3    5     2       1   0.471
## 15        15          0         2    3     2       4   0.446
## 8          8          1         7    2     0       1   0.415
## 39        39          0         1    2     2       1   0.401
## 18        18          0        11    4     0       5   0.355
## 1          1          0         5    2     1       3   0.345
## 26        26          0         4    5     1       5   0.336
## 27        27          0         4    1     1       4   0.316
## 49        49          0         8    2     0       4   0.256
## 31        31          0         2    1     1       1   0.246
## 21        21          0         7    5     0       1   0.227
## 38        38          0         1    2     1       2   0.225
## 25        25          0         6    4     0       1   0.194
## 9          9          0         4    3     0       5   0.148
## 40        40          0         4    5     0       2   0.144
## 35        35          0         4    1     0       5   0.140
## 41        41          0         4    3     0       2   0.136
## 4          4          0         3    2     0       2   0.103
## 47        47          0         3    1     0       1   0.095
## 10        10          0         2    2     0       3   0.078
```

h. (4 points) How many patients have an ADscore greater than or equal to 0.900? What are their patient IDs?

```
greater <- ADrisk[ADrisk$ADscore >= 0.900,]
greater
```

```
##    patientID APOEstatus digitspan game forms finance ADscore
## 50        50          2         8    1     2       1   1.000
## 7          7          2         5    3     2       3   0.929
## 5          5          1        11    5     2       5   0.919
## 2          2          2         4    3     2       3   0.900
```

```
sum <- nrow(greater)
sum
```

```
## [1] 4
```

```
patient_ids <- greater$patientID
patient_ids
```

```
## [1] 50  7  5  2
```

Write your answer here.

2. Why vectorize?

a. (4 points)In the R box below, define a variable, `n`, to have the value 1,000,000. Now define `a` and `b` to be two vectors of n normally distributed random numbers. The first should have mean 1 and standard deviation 1; the second, mean 4 and standard deviation 2.

```
n <- 1000000
a <-rnorm(n, mean=1, sd=1)
b<-rnorm(n,mean=4,sd=2)
```

b. (4 points) Use vectorized addition (a+b) to create a new vector, `mysum`, that contains the element-wise sum of a and b. Once you're sure this works, wrap that code using the `system.time()` function to determine how long this takes to run. Use `replicate` to run 100 iterations of this code timing your summation. Store the elapsed times from all 100 runs in a vector called `mytimes`. Use `summary()` to get summary statistics for the elapsed times.

```
 mysum <- a+b

mytimes <- replicate(100, {
  timing <- system.time({mysum<-a+b})
  timing["elapsed"]
})

summary(mytimes)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0e+00   1e-03   1e-03   8e-04   1e-03   2e-03
```

c. (4 points) Now, using the same function to time your code, re-compute mysum using a `for` loop over the indices from 1 to n. The values in mysum should be the same. Use `replicate` to run 100 iterations of this code timing your summation. Store the elapsed times from all 100 runs in a vector called `myfortimes`. Again look at the summary() statistics for the elapsed times.

```
time_loop <- function() {
  mysum <- numeric(n)
  for (i in 1:n) {
    mysum[i] <- a[i] + b[i]
  }
}


myfortimes <- replicate(100, {
  timing <- system.time(time_loop())
  timing["elapsed"]
})


summary(myfortimes)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.06100 0.06200 0.06200 0.06192 0.06200 0.06400
```

d. (4 points) Plot a histogram of mysum. Does the sum look like it too follows a normal distribution?

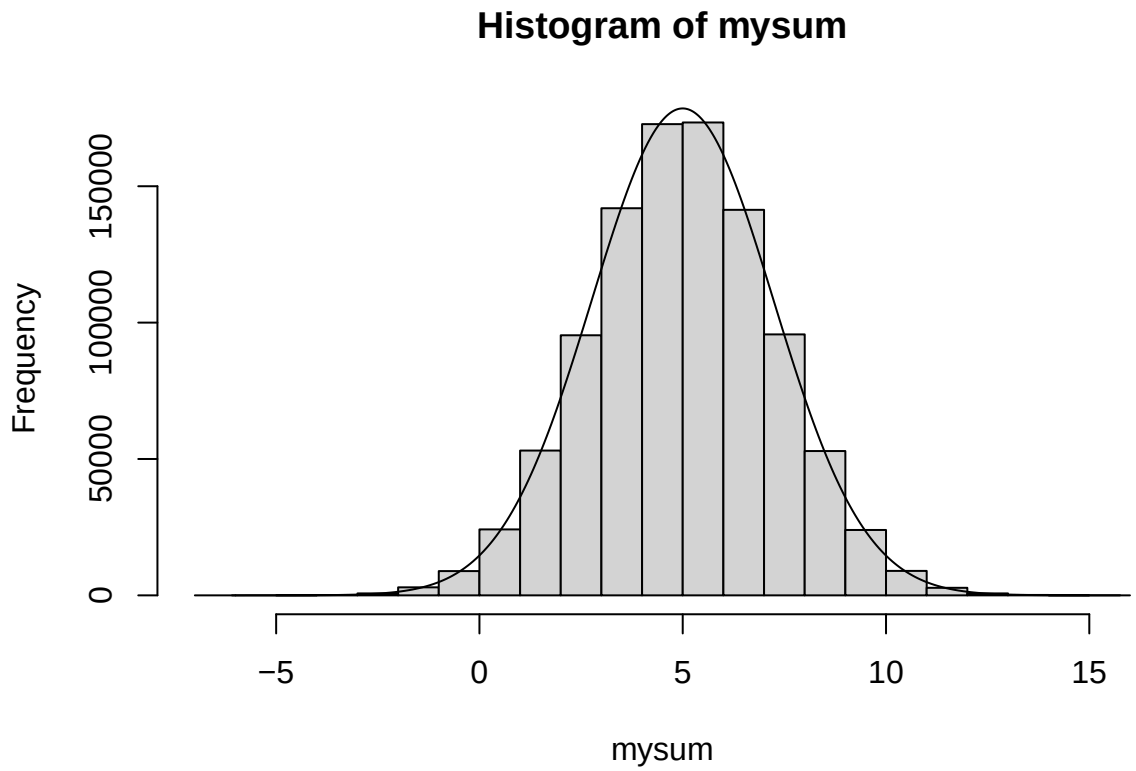
```
hist(mysum)
mean <- mean(mysum)
mean
```

```
## [1] 4.99802
```

```
sd<-sd(mysum)
sd
```

```
## [1] 2.234572
```

```
x <- seq(min(mysum), max(mysum),length=1000)
y <- dnorm(x, mean = mean, sd = sd)
lines(x, y * length(mysum) * diff(hist(mysum)$breaks)[1])
```



Write your answer here.
yes

e. (6 points) Make a guess at the mean and standard deviation of a normal distribution that might fit the data in mysum. Create a simulated vector, `simnormal`, of random instances from your hypothesized normal distribution, and plot the two histograms (the real mysum, and that of your simulated normal vector) one above the other.

(The command `par(mfrow=c(2,1)` will create a plot layout with two rows and one column. [If you get a complaint from RStudio about figure margins being too large, enlarge the plot window until this error goes away.])

You may need to try a few values of the parameters to match the distributions reasonably.

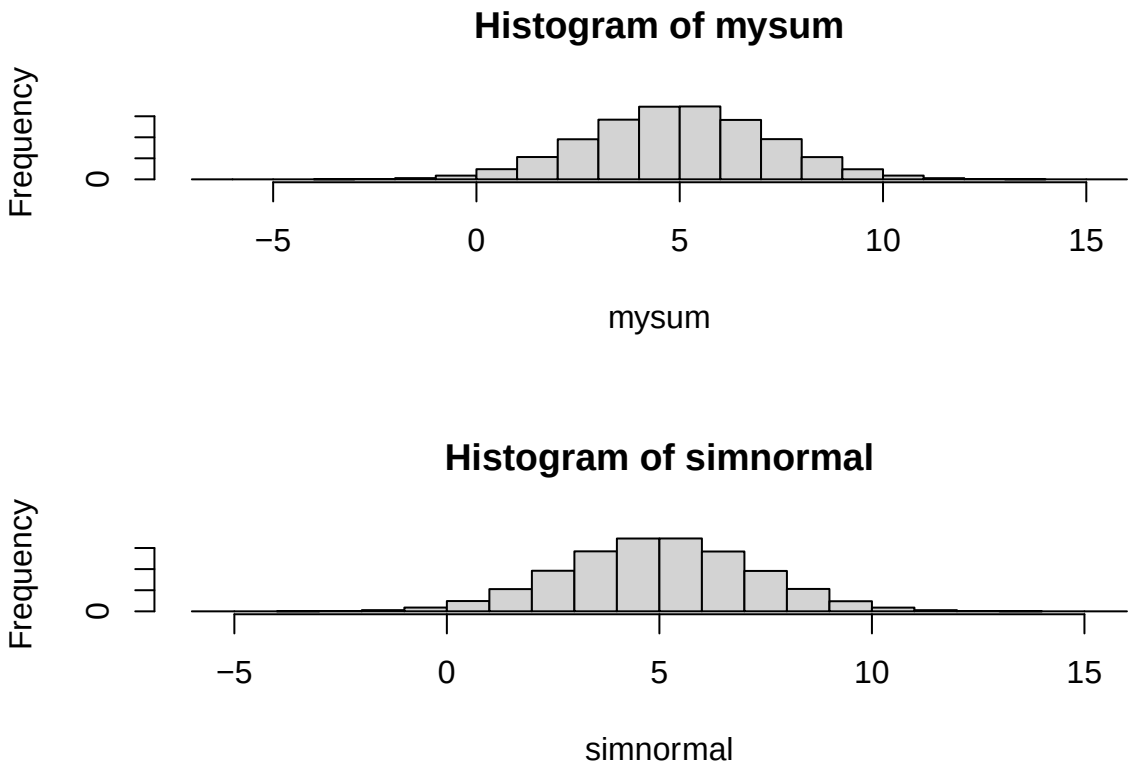
```
library(MASS)
fit <- fitdistr(mysum, "normal")


estimated_mean <- fit$estimate[1]
estimated_sd <- fit$estimate[2]


set.seed(123)

simnormal <- rnorm(n, mean = estimated_mean, sd = estimated_sd)

par(mfrow=c(2,1))
hist(mysum)
hist(simnormal)
```



f. (4 points) What is the apparent relationship between the means of mysum and the means of a and b? What is the apparent relationship between the variance of mysum and the variance of a and b?

Hint: The variance is the square of the standard deviation, but you can also compute the variance of a vector with the *var()* function.

Write your answer here.
mean(mysum)=mean(a)+mean(b)

var(mean)=var(a)+var(b)

```
mean(mysum)
```

```
## [1] 4.99802
```

```
mean(a)
```

```
## [1] 1.00088
```

```
mean(b)
```

```
## [1] 3.99714
```

```
var_mysum<-var(mysum)
var_mysum
```

```
## [1] 4.993312
```

```
var_a<-var(a)
var_a
```

```
## [1] 0.9999321
```

```
var_b<-var(b)
var_b
```

```
## [1] 3.997688
```

3. Functions and simulation

a. (4 points) Explain, in a sentence or two, what is "extreme" about extreme value analysis. Give an example of a case where you might want to use an extreme value analysis beyond the examples in the textbook or this problem.

Write your answer here.

extreme values are significantly larger or smaller than the central values, Probability of an event more extreme than any previously observed event.

example:Used in hydrology to estimate the probability of unusually large flood events, such as a 100-year flood that is more extreme than any previously observed event

b. (5 points) Write an R function named `extremelyNormal()` that uses simulation to estimate the probability that the maximum of n normally distributed random variables with mean 0 and standard deviation 1 is at least m. The parameters to your function will be n, m, and t. The function should conduct t trials, where each trial samples n random variables from the normal distribution, and should then use these trials to estimate the probability that the largest of the n values is at least m. The function should return this probability.

   Again, good R programming style avoids writing explicit loops in code, preferring vector operations or the "replicate" or "apply" functions. Functions that use loops to iterate over the data instead of these alternatives will receive partial credit for this problem.

   Include your code below.

```
extremelyNormal <- function(n, m, t) {

  trials <- matrix(rnorm(n * t, mean=0, sd=1), ncol = n)

  max_values <- apply(trials, 1, max)

probb <- sum(max_values >= m)/t

  return(probb)
}
```

c. (2 points) Note: Good programming style, regardless of language, dictates that we avoid hard-coding constant values throughout our programs. One solution to this problem is to define constants, variables whose values do not change during the program. E.g., you could start a function with the commands

```
norm_mean = 0.0
norm_stdev = 1.0
```

However, in this case, you do not even need to do that. Look at the manual page for the rnorm function and explain why we do not need to define constants to set the mean and standard devation to 0 and 1, respectively.

```
*Write your answer here.*
```

Because if mean or sd are not specified they assume the default values of 0 and 1, respectively.

d. (4 points) Use "replicate()" to run extremelyNormal() five times, where each run has 10 random variables, a threshold of m=2, and 10 trials. What range of values do you get in your five iterations? Do you think the probability estimates you get using this approach are accurate?

```
results <- replicate(5, extremelyNormal(10, 2, 10))
results
```

```
## [1] 0.2 0.2 0.2 0.1 0.0
```

```
*Write your answer here.*

No, the trials is too small. Increasing the number of trials (t) can help reduce the variability in the
```

e. (5 points) Run it five more times with the same n and m values but with each replicate having 10000 trials. What is the range of values you see now? Why is the range different from the range in part d?

```
results_2 <- replicate(5, extremelyNormal(10, 2,1000))

results_2
```

```
## [1] 0.228 0.220 0.196 0.195 0.195
```

Write your answer here.

The range of values should be narrower because as the number of trials increases, the estimates will be closer to the true probabilities, thus reducing the variability of the results.

4. The hypergeometric distribution Imagine that you have an urn containing 100 balls, 10 of which are brown and 90 of which are blue (Tufts colors, yay!). Now suppose that you pick a set of 8 balls at random from the urn, without putting any back in. Think about how to compute the probability that you have at least 3 brown balls in your set of 8.

The part about not putting the balls back after you pick each one out is important. Consider: the probability that the first ball will be brown is 10/100 or 0.1. If you then put the first ball back before you choose again, the probability that the next ball is brown will also be 0.1. So you would simply be performing 10 Bernoulli trials with a success probability of 0.1, and the resulting probability distribution could be modeled by the `binom` family of functions. Been there, done that.

But if you keep the first ball, what is the probability that the second ball is brown? Well, if the first one was brown, then it would be 9/99, but if the first one was blue, it would be 10/99. What about the second one? And what if you picked the first two in the other order? Suddenly this doesn't look so Bernoulli-like!

The answer to this question is addressed by the hypergeometric distribution, which models sampling "without replacement" from the urn. R has the usual family of functions for this distribution, `dhyper`, `phyper`, `qhyper`, and `rhyper`.

The distribution is often used to argue that an observed biological event is "surprising" - or that in fact it is not. Suppose that we have, instead of 100 balls, 100 genes that have been implicated in genetic association studies of schizophrenia. Of these 100 disease-associated genes, 10 of them relate to 5-HT serotonergic receptor signaling (a pathway that plays a role in many other psychiatric or neurological disorders) and the remaining 90 do not.

Now, suppose that you sequenced these 100 genes in a patient with schizophrenia and observed mutations in only 8 of them. Suppose three of those genes were in the 5-HT pathway, and the five other mutated genes were not. Based on the results of the genetic testing you wonder whether the 5-HT pathway may be dysregulated in this particular patient. What is the probability that you would find mutations in at least three of the ten 5-HT pathway genes by chance? a:(5 points) First, read the help page for phyper. Write an expression using the phypre function to compute this probability explicitly in R.

```
probability <- phyper(3 - 1, 10, 90, 8, lower.tail = FALSE)
probability
```

```
## [1] 0.03138817
```

b. (5 points) Now, estimate the probability by simulation, using 10000 trials and the `rhyper()` function in R. What is the probability of seeing at least 3 of the 5-HT serotonergic receptor signaling pathway genes in your simulation? The probability of seeing at least 4? The probability of seeing exactly 2?

```
set.seed(123)
trials <- 10000
rhyper <- rhyper(10000,10,90,8)
p3 <- sum(rhyper>=3)/10000
p3
```

```
## [1] 0.0289
```

```
p4 <-sum(rhyper >= 4)/10000
p4
```

```
## [1] 0.0028
```

```
p2 <- sum(rhyper == 2)/10000
p2
```

```
## [1] 0.1482
```

Write your answers here.

c. (5 points) Compute the probability, by simulation, of seeing at least 3 of the serotonin receptor pathway genes in a random **binomial** distribution where the probability of success is 10/100 (the number of 5 HT serotonin receptor signaling pathway genes out of the total number of genes in our data set). What is the probability of seeing at least 3 genes from this pathway in this simulation?

```
set.seed(123)
prob <- pbinom(3-1,8,0.1,lower.tail=FALSE)
prob
```

```
## [1] 0.03809179
```

d. (5 points) Do you think the binomial distribution is a good approximation to the hypergeometric distribution in this case? Show your code for answering this below. How do the two distributions differ? Generally speaking, when might the binomial be a good approximation to the hypergeometric distribution?

```
set.seed(123)
h<-rhyper(10000,10,90,8)

p_h<-sum(h >=3)/10000
p_h
```

```
## [1] 0.0289
```

```
binomial <-rbinom(10000,8,0.1)

p_binomial <- mean(binomial >=3)
p_binomial
```

```
## [1] 0.0384
```

Write your answers here.

NO

In general, the binomial distribution sampling with replacement, and the hypergeometric disteibution sampling without replacement. However, if the popupation size is much larger than sample size,binomial might be a good approximation to the hypergeometric distribution,