Homework 4 Arrays and Lists

6-26-20

Instructions

There are six exercises below. You are required to provide five solutions, with the same options for choosing languages as with the last exercise. Four exercises refer to formula from the previous homework, you may reuse code as you wish. You can provide solutions in two languages for one exercise only (for example, Ex. 1,2,3,5 in R and Ex. 1 in SAS is acceptable, Ex. 1,2,3 in SAS and Ex. 1,2 in R is not).

Reuse

For many of these exercises, you may be able to reuse functions written in prior homework. Include those functions here. You may find that you will need to modify your functions to work correctly for these exercises.

I'm also including data vectors that can be used in some exercises.

```
CaloriesPerServingMean <- c(268.1, 271.1, 280.9, 294.7, 285.6, 288.6, 384.4)
CaloriesPerServingSD <- c(124.8, 124.2, 116.2, 117.7, 118.3, 122.0, 168.3)
Year <- c(1936, 1946, 1951, 1963, 1975, 1997, 2006)
```

Exercise 1.

Part a.

We will produce a graph similar to the graph in Homework 1, but we will plot effect size by year instead of mean by year, and there will be no error bars.

Define m_1 to be the 7 means for Calories per Serving from Wansink Table 1. Define m_2 to be the single mean for Calories per Serving, 1936. Similarly, define s_1 to be the 7 standard deviations from Wansink Table 1 and define s_2 to be the single standard deviation for Calories per Serving, 1936.

Calculate Cohen's d for each m_1 vs m_2 using vector operations; you may also use your previously defined function. The result will be a vector of effect sizes relative to 1936 (the first will be 0). Plot this vector against a vector composed of the publication years 1936,1946, … ,2006

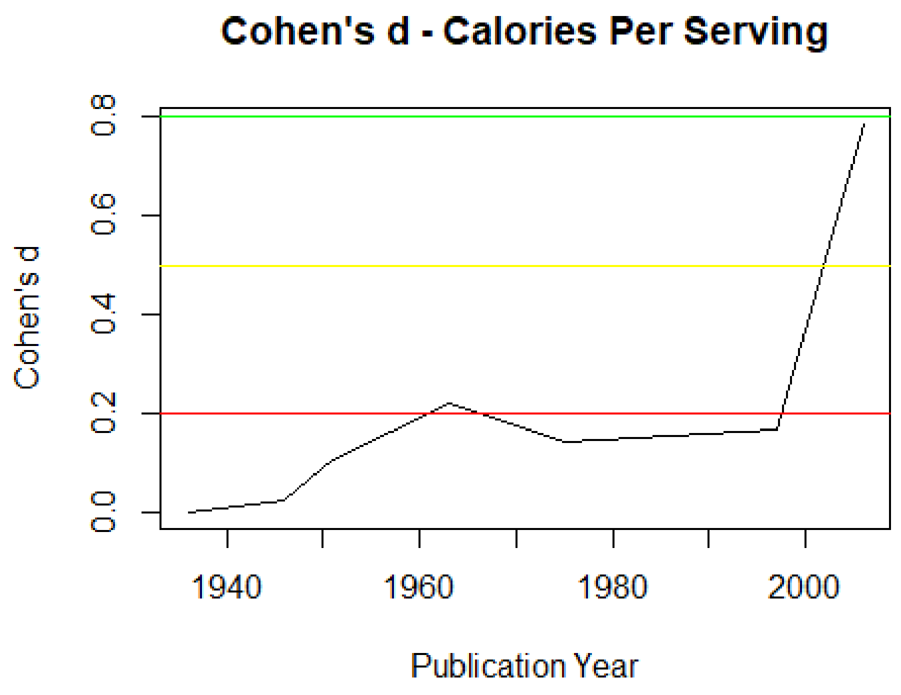
Add to this plot three horizontal lines, one at $d = 0.2$, one at $d = 0.5$ and one at $d = 0.8$. You should use different colors or different styles for each line. Should any of the effect sizes be considered *large*?

```r
m_1 <- CaloriesPerServingMean
m_2 <- CaloriesPerServingMean[1]
s_1 <- CaloriesPerServingSD
s_2 <- CaloriesPerServingSD[1]

cohen.d <- function(m_1,m_2,s_1,s_2) {
  s.pooled <- sqrt((s_1^2 + s_2^2)/2)
  return(abs(m_1-m_2)/s.pooled)
}
cd <- cohen.d(m_1,m_2,s_1,s_2)
cd

## [1] 0.0000000000000 0.0240963155864 0.1061564983665 0.2192874744299
## [5] 0.1439222362117 0.1661157277873 0.7849876039586

plot(Year, cd, main="Cohen's d - Calories Per Serving", type="l",
     ylab = "Cohen's d", xlab = 'Publication Year')
abline(a = 0.2,b = 0, col = 'red')
abline(a = 0.5, b = 0, col = 'yellow')
abline(a = 0.8, b = 0, col = 'green')
```



Comment - The Cohen's D for 1936 vs. 2006 is very close to large, but not quite to the 0.8 mark.

Exercise 2

Create a table to show the required replicates for a range of combinations of $\%Diff$ and CV. Do this in steps as follows:

Part a.

Define two matrices, one for `CV` and one for `Diff`. Each matrix will be 5 rows by 6 columns. Let the rows in `CV` be the sequence 8,12,…,28 and let the columns of `Diff` be the squence 5,10,…,25. The matrices should look like:

$$CV = \left\{\begin{matrix} 8 & 12 & 16 & 20 & 24 & 28 \\ 8 & 12 & 16 & 20 & 24 & 28 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 8 & 12 & 16 & 20 & 24 & 28 \end{matrix}\right\}$$

$$Diff = \left\{\begin{matrix} 5 & 5 & 5 & 5 & 5 & 5 \\ 10 & 10 & 10 & 10 & 10 & 10 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 25 & 25 & 25 & 25 & 25 & 25 \end{matrix}\right\}$$

Define and print your matrices in the code below.

```r
CV <- matrix(seq(8,28,by=4),nrow=5, ncol=6, byrow=TRUE)
Diff <- matrix(seq(5,25,by=5),nrow=5,ncol=6)

print(CV)

##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    8   12   16   20   24   28
## [2,]    8   12   16   20   24   28
## [3,]    8   12   16   20   24   28
## [4,]    8   12   16   20   24   28
## [5,]    8   12   16   20   24   28

print(Diff)

##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    5    5    5    5    5    5
## [2,]   10   10   10   10   10   10
## [3,]   15   15   15   15   15   15
## [4,]   20   20   20   20   20   20
## [5,]   25   25   25   25   25   25
```

Part b.

Calculate require replicates for each combination of `CV` and `Diff`. Use $\alpha = 0.05$ and $\beta = 0.2$. You should be able to reuse code from previous exercises, and you should not use iteration.

Print the result below. The result should be a 5×6 matrix.

```
required.replicates <- function(cv,dif,alpha = 0.05, beta = 0.20) {
  z_alpha <- qnorm(1-alpha/2)
  z_beta <- qnorm(1-beta)
  return(ceiling(2*((cv/dif)^2)*((z_alpha + z_beta)^2)))
}
rrep <- required.replicates(CV,Diff)
rrep

##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]   41   91  161  252  362  493
## [2,]   11   23   41   63   91  124
## [3,]    5   11   18   28   41   55
## [4,]    3    6   11   16   23   31
## [5,]    2    4    7   11   15   20
```

To check your work, repeat the calculations using the rule of thumb from the previous exercises. What is largest deviation of the rule of thumb from the exact calculation? I find that for about half the combinations of `CV` and `Diff`, the two methods are identical; for most the difference is 1 or less.

```
delta <- (Diff)/(CV)
rule.thumb <- function(delta){
  return(ceiling(16/(delta^2)))
}
RoT <- rule.thumb(delta)
print('Rule of Thumb')

## [1] "Rule of Thumb"

RoT

##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]   41   93  164  256  369  502
## [2,]   11   24   41   64   93  126
## [3,]    5   11   19   29   41   56
## [4,]    3    6   11   16   24   32
## [5,]    2    4    7   11   15   21

print('Variance of Required Replicates vs. Rule of Thumb')

## [1] "Variance of Required Replicates vs. Rule of Thumb"

check <- rrep - RoT
check

##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    0   -2   -3   -4   -7   -9
## [2,]    0   -1    0   -1   -2   -2
## [3,]    0    0   -1   -1    0   -1
## [4,]    0    0    0    0   -1   -1
## [5,]    0    0    0    0    0   -1
```

Comment - I used an estimated algebraic expression to estimate delta based on the CV and Diff values given. According to these results, the larger the required replicates get, the larger the variance is between the calculation and the rule of thumb. This is in line with my results from homework 2 where the required replicates for 1936 vs 1997 was ~580 with a variance from the rule of thumb ~ 11. In this example, a required replicates of 502 has a variance from the rule of thumb of ~10.

Exercise 3

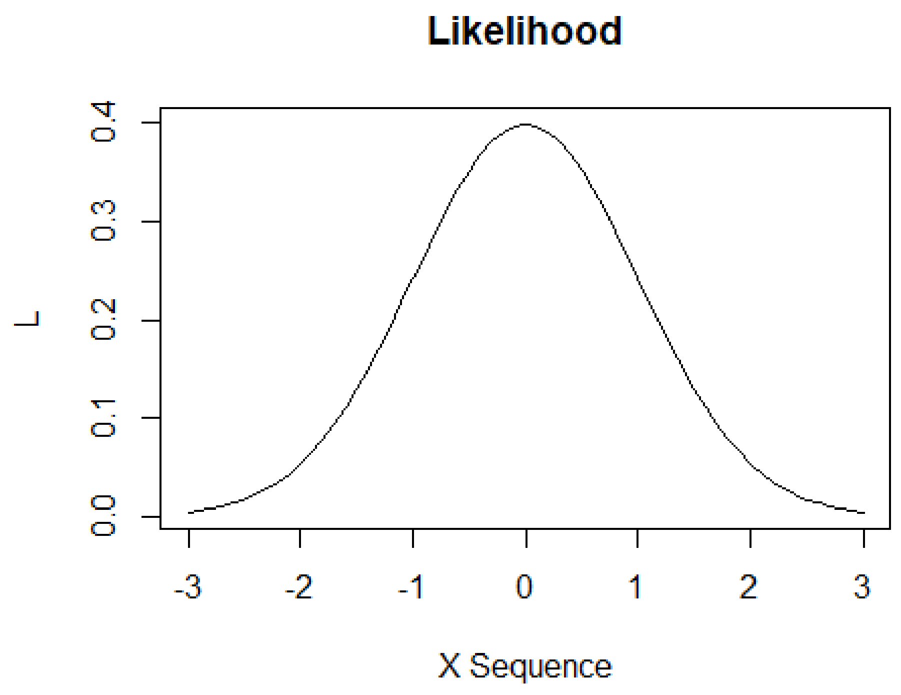
In this exercise, we will test your `norm.pdf` function with a range of inputs.

Do not print the vectors you create for this exercise in the final typeset submission We will check the results by examining the plots, and printing the vectors themselves will unnecessarily clutter your report. If you get stuck, use the built normal functions to create your plots.

Part a.

Generate a squence of values from $-3, \ldots, 3$ incremented by 0.1; let this be `x_1`. Calculate the likelihood of each value of `x_1` using the `norm.pdf` function from Homework 3, letting `mu=0` and `sd=1`. Plot the likelihood curve (L is the dependent variable, x is independent) as a line graph.

```
x_1 <- seq(-3,3,by=.1)
norm.pdf <- function(x,mu=0,sig=1) {
  return((1/(sig*sqrt(2.0*pi)))*exp(1)^(-1*(((x-mu)^2)/(2*sig^2))))
}
L <- norm.pdf(x_1)
plot(x_1,L, main="Likelihood", type="l",
     ylab = "L", xlab = 'X Sequence')
```



Part b.

Let m_{1936} be the mean Calories per Serving from 1936, and let m_{2006} be the mean Calories per Serving, 2006. Let s_{1936} and s_{2006} be the corresponding standard deviations.

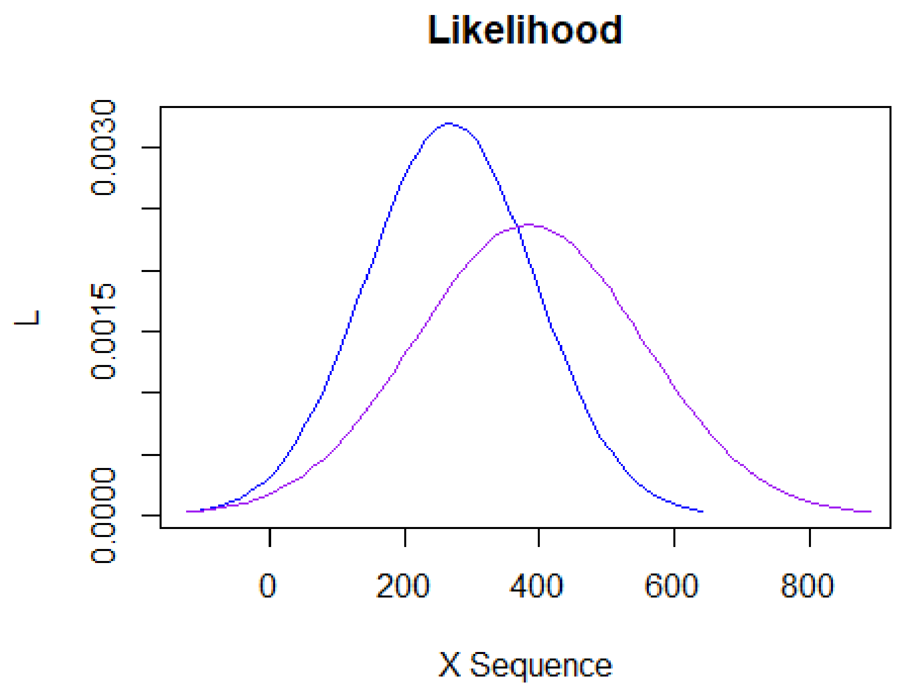
Create two sequences and name these `x_2` and `x_3`. Define `x_2` to be a range of values $[m_{1936} - 3 \times s_{1936}, \ldots, m_{1936} + 3 \times s_{1936}]$ and define `x_3` to be $[m_{2006} - 3 \times s_{2006}, \ldots, m_{2006} + 3 \times s_{2006}]$. `x_2` and `x_3` should be the same length as `x_1`.

Calculate the corresponding likelihood for these sequences, using $\{\mu = m_{1936}, \sigma = s_{1936}\}$ with `x_2` and use $\{\mu = m_{2006}, \sigma = s_{2006}\}$ with `x_3`.

As with part a, plot the likelihood curve for both sequences, but include both in the same graph. Use two different colors or line types for each curve. You may need to use `min` and `max` to find `xlim` values or `ylim` to fit both curves on the same plot. The first curve in this graph should appear identical to the curve in part a; the second curve will be similar but will differ in location and spread.

```
m_1936 <- 268.1
s_1936 <- 124.8
m_2006 <- 384.4
s_2006 <- 168.3
x_2 <- m_1936 + x_1 * s_1936
x_3 <- m_2006 + x_1 * s_2006
L_1936 <-  norm.pdf(x_2,m_1936,s_1936)
L_2006 <-  norm.pdf(x_3,m_2006,s_2006)
```

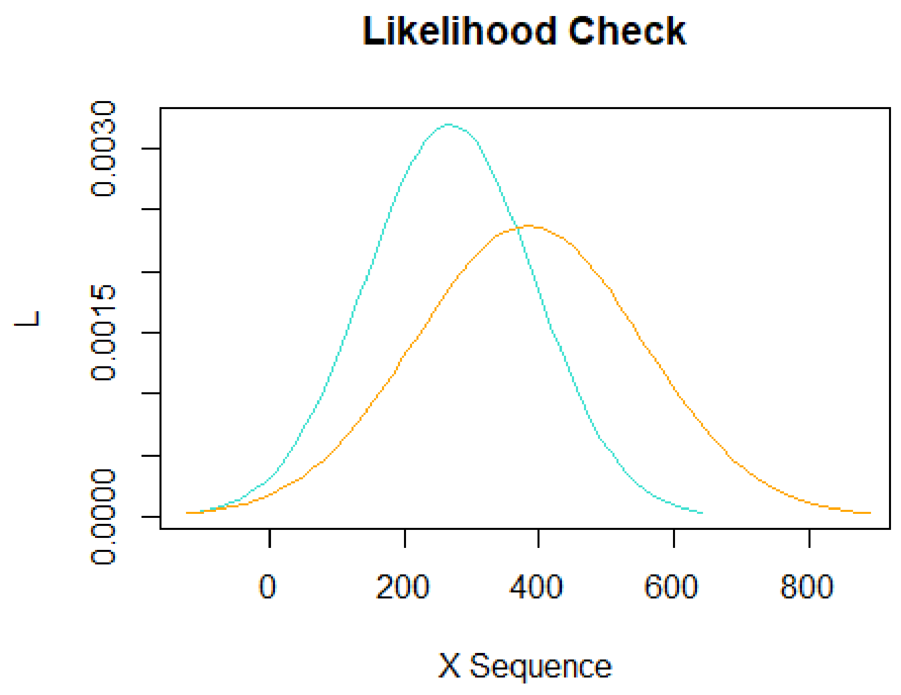
```
plot(x_2,L_1936, main="Likelihood", type="l",
     ylab = "L", xlab = 'X Sequence',col='blue',xlim=c(-120,880))
lines(x_3,L_2006,col='purple')
```



If you choose to solve this with SAS, I've included code in the SAS template to create the graphs, since combining plots in IML is not as easy as in R.

If you wish, you may reproduce the curves using `dnorm` to compare with your function.

```
L_1936_check <-  dnorm(x_2,m_1936,s_1936)
L_2006_check <-  dnorm(x_3,m_2006,s_2006)
plot(x_2,L_1936_check, main="Likelihood Check", type="l",
     ylab = "L", xlab = 'X Sequence',col='turquoise',xlim=c(-120,880))
lines(x_3,L_2006_check,col='orange')
```



Exercise 4

Suppose we wish to determine the linear relationship between per Calories per Serving and Year. We can determine this by solving a system of linear equations, of the form

$$\begin{aligned} 268.1 &= \beta_1 + \beta_2 \times 1936 \\ 271.1 &= \beta_1 + \beta_2 \times 1946 \\ \vdots &= \vdots \\ 384.4 &= \beta_1 + \beta_2 \times 2006 \end{aligned}$$

We write this in matrix notation as

$$\begin{pmatrix} 268.1 \\ 271.1 \\ \vdots \\ 384.4 \end{pmatrix} = \begin{pmatrix} 1 & 1936 \\ 1 & 1946 \\ \vdots & \vdots \\ 1 & 2006 \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}^t$$

We can also write this as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta}$$

and find a solution by computing $\widehat{\boldsymbol{\beta}} = \mathbf{X}^{-1}\mathbf{y}$.

However, an exact solution for the inverse, $\mathbf{X}^{-1}$ require square matrices, so commonly we use the *normal* equations,

$$\mathbf{X}^t\mathbf{y} = \mathbf{X}^t\mathbf{X}\boldsymbol{\beta}$$

(where $\mathbf{X}^t$ is the transpose of $\mathbf{X}$). We then find

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t\mathbf{y}$$

Answer

Define appropriate `X` and `y` matrices (`y` can be a vector in R) in the chunk below.

Multiply the transpose of `X` by `X`, then use `solve` (R) or `inv` (IML) to find the inverse $(\mathbf{X}^t\mathbf{X})^{-1}$. Multiply this by the product of transpose `X` and `y` to find `hat.beta`.

Print your `hat.beta`.

To check your work, calculate the values predicted by your statistical model. Compute `hat.y` by multiplying `X` and `hat.beta`,

$$\hat{y} = \mathbf{X}\hat{\beta}$$

Plot `y` vs the independent variable (the second column of `X`) as points, and `hat.y` vs independent variable as a line, preferably a different colors. The `hat.y` values should fall a straight line that interpolates `y` values.

You can also compare your result to the R function (set `eval=TRUE`).

```
summary(lm(CaloriesPerServingMean~Year))
```

Alternative methods

You can also compute $\hat{\beta}$ by passing both $\mathbf{X}^t\mathbf{X}^{-1}$ and $\mathbf{X}^t\mathbf{y}$ as arguments to `solve`.

Alternatively, you can install the `MASS` library and use `ginv` to compute a generalized inverse $\mathbf{X}^{-1}$. Use this to compute $\widehat{\boldsymbol{\beta}} = \mathbf{X}^{-}\mathbf{y}$ in the chunk below:

```
library(MASS)
print(hat.beta <- ginv(X) %*% y)
```

Exercise 5

Given a vector of mean estimates $x = x_1, x_2, \dots, x_k$, a vector of standard deviations $s = s_1, s_2, \dots, s_k$ and a vector of sample sizes $n = n_1, n_2, \dots, n_k$, we can calculate a one-way analysis of variance by

$$MSB = \frac{n_1(x_1 - \overline{x})^2 + n_2(x_2 - \overline{x})^2 + \cdots + n_k(x_k - \overline{x})^2}{k - 1} = \frac{\sum_i n_i\,(x_i - \overline{x})^2}{k - 1}$$

and

$$MSW = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2 + \cdots (n_k - 1)s_k^2}{N - k} = \frac{\sum_i (n_i - 1)s_i^2}{N - k}$$

where $\overline{x}$ is the weighted mean of $x_i = \frac{\sum_i n_i * x_i}{N}$ and $N = \sum_i n_i$. The test statistic is $F = \frac{MSB}{MSW}$ which is distributed as $F_{\alpha,k-1,N-k}$

Part a

Calculate MSW and MSB for Calories per Serving from Wansink Table 1. You can use the variables `CaloriesPerServingMean` and `CaloriesPerServingSD` defined below. Let $n_1 = n_2 \ldots = n_k = 18$

Use array functions and arithmetic for your calculations, you should not need iteration (for loops). Do not hard code values for N and k, calculate these from the `CaloriesPerServingMean` or `CaloriesPerServingSD`.

Print both MSB and MSW.

```
x <- CaloriesPerServingMean
s <- CaloriesPerServingSD
x_bar <- mean(CaloriesPerServingMean)
n <- 18
k <- length(CaloriesPerServingMean)
N <- k*n
SSA <- n*((x-x_bar)^2)
MSB <- sum(SSA)/(k-1)
SSW <- (n-1)*(s^2)
MSW <- sum(SSW)/(N-k)
cat('MSB =',MSB,'and MSW =',MSW)

## MSB = 28815.96 and MSW = 16508.5985714
```

Part b

Calculate an F-ratio and a p for this F, using the F distribution with $k-1$ and $N-k$ degrees of freedom. Use $\alpha = 0.05$. Compare these values to the corresponding values reported in Wansink Table 1.

```
F.ratio <- MSB/MSW
p <- 1- pf(F.ratio,k-1,N-k)
cat('F ratio =',F.ratio,'and p value =',p)

## F ratio = 1.7455121872 and p value = 0.116313264501
```

You can also check results by entering appropriate values in an online calculator like http://statpages.info/anova1sm.html .

Comment - I used the online calculator to confirm the above F ratio and p value matched. However, when I compare it to Wansink Table 1, the values differ by a large margin. The F value and P value in the Wansink table are 436.9 and 0.000, respectfully.

Exercise 6

In this, we compare the normal and Poisson distributions, using the functions you've written previously. This is also a way to test your normal and Poisson functions over a range of arguments.

Do not print the vectors you create for this exercise in the final typeset submission We will check the results by examining the plots, and printing the vectors themselves will unnecessarily clutter your report. If you get stuck, use the built functions to create your plots. However, the final submission must call your functions.

Part a

Create a sequence of x_a from $(-5\ldots5)$, incremented by 0.1. Calculate the normal likelihood for each x, assuming $\mu = 0$ and $\sigma = 1$. Also calculate Poisson probability of each x given a `lambda=1`.

Plot both sets of probablities against `x_a` as lines, using a different color for each curve. Make sure that both curves fit in the plot; you may need to determine minimum and maximum values and set these as graphic parameters (see `ylim`).

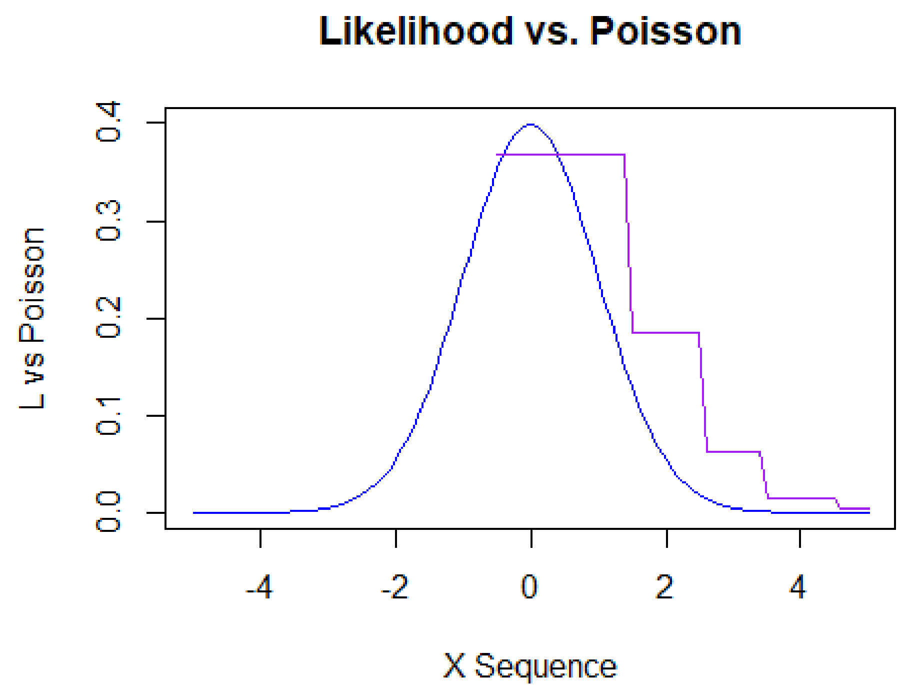
Warning: if you do this in SAS, you may have to adjust the lower bound of x.

```r
x_a <- seq(-5,5,by=0.1)
L <- norm.pdf(x_a)

pois.pmf <- function(x,lambda) {
  x <- round(x,0)
  return (exp(-lambda)*(1/factorial(x))*exp(x*log(lambda)))
}
p <- pois.pmf(x_a,1)

## Warning in gamma(x + 1): NaNs produced

plot(x_a,L, main="Likelihood vs. Poisson", type="l",
     ylab = "L vs Poisson", xlab = 'X Sequence',col='blue')
lines(x_a,p,col='purple')
```



Does this graph tell you if your Normal PDF function behaves properly? Does your Poisson handle negative or non-integer values as expected? You might need to call a rounding function on the parameters inside your function.

Comment - Yes, my normal pdf function appears to be behaving properly. It has a normal distribution curve as expected.

While the Poisson looks a bit odd, I do believe it handled the negatives and integers as expected. Since a negative factorial is undefined, all x values that are negative result in null values. Since x must be an integer (and I have included rounding code in my function), the plot shows a stair step pattern due to rounding to the nearest integer.

Part b

Create a sequence of $x_b = \lfloor \mu - 5 \times \sigma \rfloor, \dots, \lceil \mu + 5 \times \sigma \rceil$ using mean and standard deviation for Servings per Recipe from 1936.

Calculate the normal and Poission probability for each x_b as in part a, again using mean and standard deviation from servings per recipe, 1936. The length of this vector should be the same length as the x_a vector as in part a (± 1), so you will need to calculate an interval based on the range `x_b` and the number of elements in `x_a`

Show the the length of both x vectors are similar by calling `length` for each.

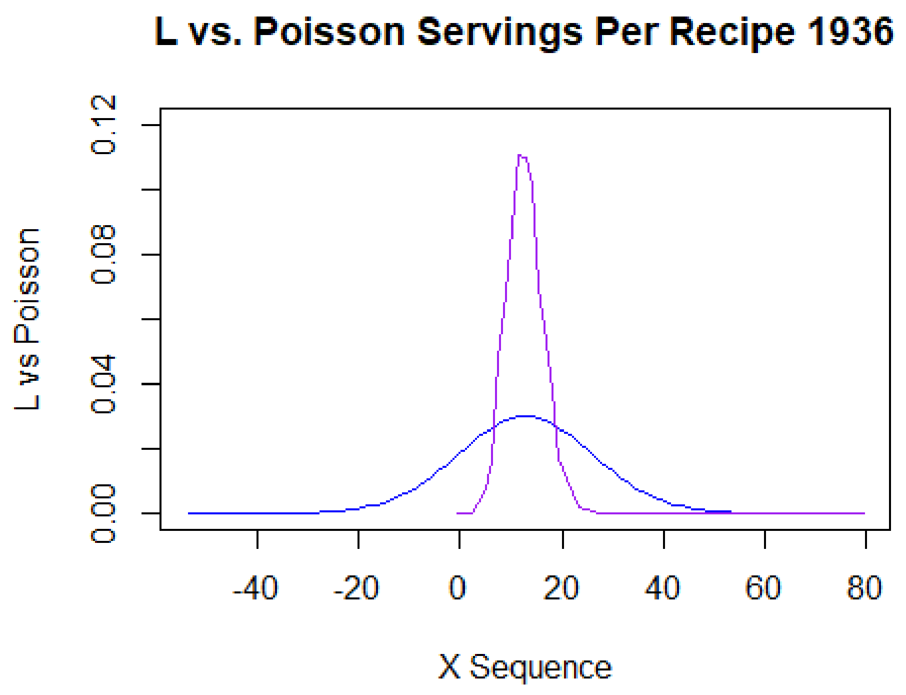
Repeat the plot from part a with this sequence.

If you choose to solve this with SAS, I've included code in the SAS template to create the graphs, since combining plots in IML is not as easy as in R.

```
ServingsPerRecipeSD = c(13.3, 13.3, 14.5, 14.6, 14.3, 14.3, 13.0)
ServingsPerRecipeMean = c(12.9, 12.9, 13.0, 12.7, 12.4, 12.4, 12.7)
m_1936 <- ServingsPerRecipeMean[1]
s_1936 <- ServingsPerRecipeSD[1]
x_b <- m_1936+x_a*s_1936
L_1936 <- norm.pdf(x_b,m_1936,s_1936)
p_1936 <- pois.pmf(x_b,m_1936)

## Warning in gamma(x + 1): NaNs produced

plot(x_b,L_1936, main="L vs. Poisson Servings Per Recipe 1936", type="l",
     ylab = "L vs Poisson", xlab = 'X Sequence',col='blue',ylim = c(0,.12))
lines(x_b,p_1936,col='purple')
```

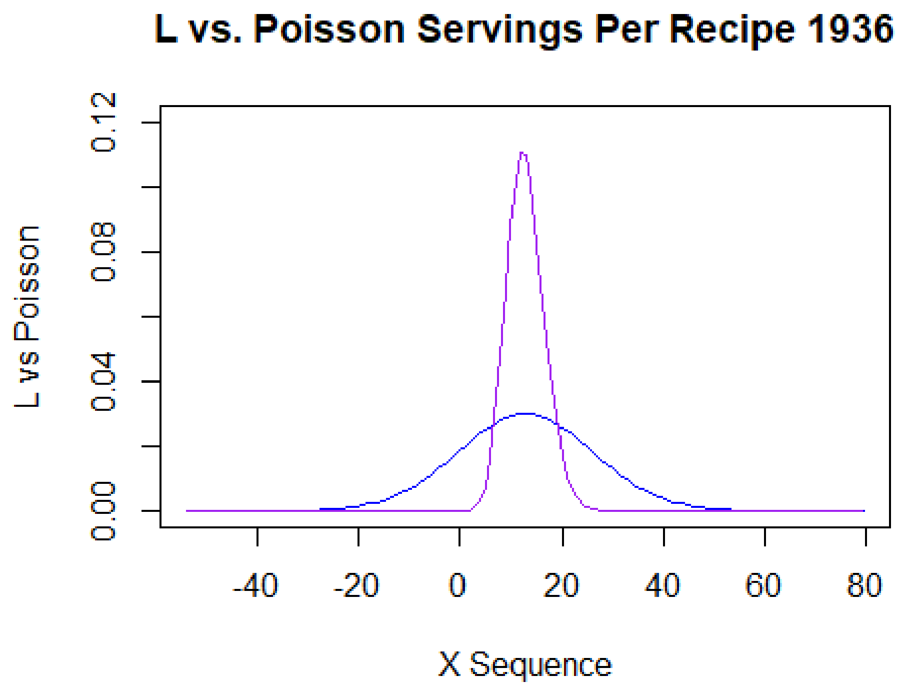


```
cat('The length of x_a = ',length(x_a),'and the length of x_b =',length(x_b))

## The length of x_a =  101 and the length of x_b = 101
```

To check you work, duplicate the plots by calling built in normal and Poisson functions. Does the built in Poisson function handle negative *x* differently than your function?

```
norm_check <- dnorm(x_b,m_1936,s_1936)
x_b_adj <- round(m_1936+x_a*s_1936,0)
pois_check <-dpois(x_b_adj,m_1936)
plot(x_b,norm_check, main="L vs. Poisson Servings Per Recipe 1936", type="l",
     ylab = "L vs Poisson", xlab = 'X Sequence',col='blue',ylim = c(0,.12))
lines(x_b_adj,pois_check,col='purple')
```



Comment - The Poisson function built into R will assign any negative values to zero. In addition, there is no rounding function built into it like mine. I had to round x_b in order to produce a similar graph as my function.