

1. We need to show

$$\text{jse}(\bar{X}) \equiv \sqrt{\frac{n-1}{n} \sum_{i=1}^n (\bar{X}_{(i)} - \bar{X})^2} = \frac{s}{\sqrt{n}},$$

where $\bar{X}_{(i)}$ is the mean of the data with the i th X left out and \bar{X} is the average of the n $\bar{X}_{(i)}$'s. It's easy to show that $\bar{X}_{(i)} = \bar{X}$ since the sum of the X 's with the i th one left out (call it T_i) is just $T_i = n\bar{X} - X_i$ and thus

$$\bar{X}_{(i)} = \frac{1}{n} \sum_{i=1}^n \bar{X}_{(i)} = \frac{1}{n} \sum_{i=1}^n \frac{T_i}{n-1} = \frac{1}{n(n-1)} \sum_{i=1}^n (n\bar{X} - X_i) = \frac{n^2\bar{X} - n\bar{X}}{n(n-1)} = \bar{X}.$$

Then we have

$$\begin{aligned} \frac{n-1}{n} \sum_{i=1}^n (\bar{X}_{(i)} - \bar{X})^2 &= \frac{n-1}{n} \sum_{i=1}^n \left(\frac{n\bar{X} - X_i}{n-1} - \bar{X} \right)^2 = \frac{n-1}{n} \sum_{i=1}^n \left(\frac{n\bar{X} - X_i - (n-1)\bar{X}}{n-1} \right)^2 \\ &= \frac{n-1}{n} \sum_{i=1}^n \left(\frac{X_i - \bar{X}}{n-1} \right)^2 = \frac{1}{n} \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2 = \frac{s^2}{n}. \end{aligned}$$

2. This can be done by using sweep (I didn't expect you to remember these formulas, but you get 5 points for saying that) or by just multiplying $(A + uv^T)$ on the right and left by the right hand side and showing the result is the identity (5 points for saying that). This is not easy but multiplying on the right gives:

$$\begin{aligned} (A + uv^T) \left(A^{-1} - \frac{A^{-1}uv^T A^{-1}}{1 + v^T A^{-1}u} \right) &= AA^{-1} + uv^T A^{-1} - \frac{uv^T A^{-1}}{1 + v^T A^{-1}u} - \frac{uv^T A^{-1}uv^T A^{-1}}{1 + v^T A^{-1}u} \\ &= I + uv^T A^{-1} - \frac{u(1 + v^T A^{-1}u)v^T A^{-1}}{1 + v^T A^{-1}u} = I + uv^T A^{-1} - uv^T A^{-1} = I, \end{aligned}$$

and the multiplication on the left is done similarly. You get a few more points for doing any interesting algebra.

3. Newton-Raphson consists of iterating

$$\theta_{i+1} = \theta_i - H_i^{-1} g_i, \quad i \geq 0,$$

until convergence of the θ 's, where g_i is the vector of first derivatives of S evaluated at $\theta = \theta_i$, H_i is the matrix of second derivatives of S evaluated at $\theta = \theta_i$, and the iteration starts with θ_0 being the starting value. When convergence is reached, one must check whether the final H_i matrix is negative definite. We often use $-S$ instead of S since then we need to check the resulting H_i matrix for positive definiteness which we have routines to do. The derivatives of $-S$ are just the negative of those of S . Note that the process can be applied to the log of S , but I will give the derivatives for S since most of you tried to find them.

Since $S(\theta) = A(\theta)B(\theta)$, where $A(\theta) = 2\theta_1^2 + 3\theta_2^2$ and $B(\theta) = \exp(-\theta_1^2 - \theta_2^2)$, we have in an obvious notation:

$$g_1 = S'_1 = \frac{\partial S}{\partial \theta_1} = AB'_1 + A'_1 B, \quad g_2 = S'_2 = \frac{\partial S}{\partial \theta_2} = AB'_2 + A'_2 B,$$

and

$$H_{11} = S''_1 = \frac{\partial^2 S}{\partial \theta_1^2} = AB''_1 + 2A'_1 B'_1 + A''_1 B, \quad H_{22} = S''_2 = \frac{\partial^2 S}{\partial \theta_2^2} = AB''_2 + 2A'_2 B'_2 + A''_2 B,$$

and

$$H_{12} = H_{21} = S''_{12} = \frac{\partial^2 S}{\partial \theta_1 \partial \theta_2} = AB''_{12} + A'_1 B'_2 + A'_2 B'_1 + A''_{12} B,$$

where

$$A'_1 = 4\theta_1, \quad A'_2 = 6\theta_2, \quad B'_1 = (-2\theta_1)B, \quad B'_2 = (-2\theta_2)B,$$

and

$$A''_1 = 4, \quad A''_2 = 6, \quad B''_1 = (4\theta_1^2 - 2)B, \quad B''_2 = (4\theta_2^2 - 2)B, \quad A''_{12} = 0, \quad B''_{12} = -2\theta_1 B'_2.$$

This gives

$$g_1 = (4\theta_1 - 4\theta_1^3 - 6\theta_1\theta_2^2)B, \quad g_2 = (6\theta_2 - 4\theta_1^2\theta_2 - 6\theta_2^3)B,$$

and

$$H_{11} = (8\theta_1^4 - 20\theta_1^2 - 6\theta_2^2 + 12\theta_1^2\theta_2^2 + 4)B, \quad H_{22} = (8\theta_1^2\theta_2^2 + 12\theta_2^4 - 4\theta_1^2 - 30\theta_2^2 + 6)B,$$

and finally

$$H_{12} = (8\theta_1^3\theta_2 + 12\theta_1\theta_2^3 - 20\theta_1\theta_2)B.$$

4. We'll consider which column x belongs in as the choice of row pixel is done the same way. There are $c_2 - c_1 + 1$ pixels from c_1 to c_2 and thus we divide x_1 to x_2 into $c_2 - c_1 + 1$ equal width intervals (numbered 0 through $c_2 - c_1$) and choose pixel $c_1 + i$ if x falls into the i th interval. Thus $c = c_1 + i$ if $x \in [x_1 + i * h, x_1 + (i + 1) * h)$, where $h = (x_2 - x_1)/(c_2 - c_1)$, that is, $i = [(x - x_1)/h]$.

5. Given data y_1, \dots, y_n , the kernel density estimator \hat{f}_K at the value x is given by

$$\hat{f}_K(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - y_i}{h}\right),$$

where h is the bandwidth of the estimator and K is a kernel, which in class we have assumed to be a pdf (and thus it integrates to one). Then we have

$$\int \hat{f}_K(x) dx = \int \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - y_i}{h}\right) dx = \sum_{i=1}^n \frac{1}{nh} \int K\left(\frac{x - y_i}{h}\right) dx = \frac{1}{nh} \sum_{i=1}^n \int hK(z) dz = \frac{h}{nh} \sum_{i=1}^n 1 = 1,$$

where we have made the substitution $z = (x - y_i)/h$.

6. Since $L_{ik} = 0$ for $k > i$ and $R_{kj} = 0$ for $k > j$, the (i, j) th element of the product of L and R is given by

$$(LR)_{ij} = \sum_{k=1}^{\min(i,j)} L_{ik} R_{kj},$$

which avoids multiplication by any element in the upper triangle of L or lower triangle of R . To avoid the multiplication by any diagonal element of either matrix, we must consider the three cases: $i < j$, $i = j$, and $i > j$. For example, if $i = j$ we have

$$(LR)_{ii} = 1 + \sum_{k=1}^{i-1} L_{ik} R_{ki},$$

which is fine unless $i = 1$ in which case the sum must be avoided. At the end of this report is a possible version of the subroutine. Obviously, I didn't expect you to have such an elaborate version, but you lose 5 points for not having an `ndim`, while to get full credit you had to at least say something about avoiding multiplication by diagonal elements and you couldn't have any glaring syntax errors.

7. Starting with the 11 points,

6 4 10 2 3 9 5 8 7 11 1

the first step in the first split in quicksort is to sort the 1st, 6th, and 11th points (which are 6, 9, and 1), noting that the median of these three (which is 6) is the splitting element, and then swapping the new 1st and 6th elements, which gives:

6 4 10 2 3 1 5 8 7 11 9

Next, starting with the 2nd point and moving from left to right, we find the first point greater than the splitting element. In this case, this is the 10. Then starting with the last point and moving from right to left, we find the first point less than or equal to the splitting element. This is the 5. Then we swap the 10 and the 5 which gives:

6 4 5 2 3 1 10 8 7 11 9

Now, since the difference of the two pointers (currently pointing at the 5 and the 10) is more than 1, we increment the left pointer, decrement the right pointer (so that they're now pointing at the 2 and the 1), and try to find another pair to swap. Since the 2, the 3, and the 1 are not greater than the splitting element, we find that the left pointer meets the right pointer at the 1 without finding a point greater than the splitting element. Finally, since the 6 is in fact greater than the 1, we just swap them (if the 1 had been a point greater than 6, we would swap the 6 with the point before the 1), which gives the final result

1 4 5 2 3 6 10 8 7 11 9.

Exam Scores

9| 2 3
8| 0 1 1 5 7
7| 0 2 7 7
6| 1 1 3 3 6 9
5| 2 6 6 7 7 7 8 8 9
4|1
3|0

```

      subroutine lrmult(xl,xr,n,ndim,xlr)
c*****
c
c   Subroutine to multiply the (n by n) unit lower triangular matrix xl
c   times the (n by n) unit upper triangular matrix xr and put the
c   result in the (n by n) matrix xlr.
c
c   ndim is the row dimension of xl, xr, and xlr in the calling routine.
c
c*****
c
c      dimension xl(ndim,1),xr(ndim,1),xlr(ndim,1)
c
c   Get diagonal elements:
c
c      do 20 i=1,n
c         c=1.0
c         if(i.gt.1) then
c            do 10 k=1,i-1
10             c=c+xl(i,k)*xr(k,i)
c         endif
20          xlr(i,i)=c
c
c   i<j:
c
c      do 50 i=1,n
c         if(i.ne.n) then
c            do 40 j=i+1,n
c               c=xr(i,j)
c               if(i.gt.1) then
c                  do 30 k=1,i-1
30                 c=c+xl(i,k)*xr(k,j)
c               endif
40                xlr(i,j)=c
c            endif
50           continue
c
c   i>j:
c
c      do 150 i=1,n
c         if(i.ne.1) then
c            do 140 j=1,i-1
c               c=xl(i,j)
c               if(j.gt.1) then
c                  do 130 k=1,j-1
130                 c=c+xl(i,k)*xr(k,j)
c               endif
140                xlr(i,j)=c
c            endif
150           continue
c
c      return
c      end

```