Problem set 8

2025-03-25

You are not allowed to load any package or use for-loop. For exercises 1 and 3-6 you only get to write one line of code for the solution.

For better preparation for midterm, we recommend not using chatGPT for this homework.

1. Create a 100 by 10 matrix of randomly generated standard normal numbers. Put the result in `x`. Show the subset of `x` defined by the first 5 rows and the first 4 columns.

```r
set.seed(2025); x <- matrix(rnorm(1000), 100, 10); x[1:5, 1:4]
```

```
          [,1]       [,2]        [,3]        [,4]
[1,] 0.6207567  0.6904259 -0.19608802 -0.89523391
[2,] 0.0356414 -0.2547308 -2.84924294 -0.33781881
[3,] 0.7731545  1.0423578 -0.70500068  0.09515953
[4,] 1.2724891  0.2370112 -0.08986966 -1.03025235
[5,] 0.3709754 -1.3272372 -0.17400476 -0.97284603
```

2. Apply the three R functions that give you the dimension of `x`, the number of rows of `x`, and the number of columns of `x`, respectively. Print the responses.

```r
cat("dim:", dim(x), "\nnrow:", nrow(x), "\nncol:", ncol(x), "\n")
```

```
dim: 100 10
nrow: 100
ncol: 10
```

3. Generate matrix `y` obtained from adding the scalar 1 to row 1, the scalar 2 to row 2, and so on, to the matrix `x`. Show the subset of `y` defined by the first 5 rows and the first 4 columns.

```
y <- x + 1:100; y[1:5, 1:4]
```

```
         [,1]     [,2]       [,3]      [,4]
[1,] 1.620757 1.690426  0.8039120 0.1047661
[2,] 2.035641 1.745269 -0.8492429 1.6621812
[3,] 3.773154 4.042358  2.2949993 3.0951595
[4,] 5.272489 4.237011  3.9101303 2.9697476
[5,] 5.370975 3.672763  4.8259952 4.0271540
```

4. Generate matrix `z` obtained from adding the scalar 2 to column 1, the scalar 4 to column 2, and so on, to the matrix `y`. Hint: Use `sweep` with `FUN = "+"`. Show the subset of `z` defined by the first 5 rows and the first 4 columns.

```
z <- sweep(y, 2, 2*(1:10), FUN = "+"); z[1:5, 1:4]
```

```
         [,1]     [,2]      [,3]      [,4]
[1,] 3.620757 5.690426  6.803912  8.104766
[2,] 4.035641 5.745269  5.150757  9.662181
[3,] 5.773154 8.042358  8.294999 11.095160
[4,] 7.272489 8.237011  9.910130 10.969748
[5,] 7.370975 7.672763 10.825995 12.027154
```

5. Compute the average of each row of `z`. Show the first 10 elements

```
(rowMeans(z))[1:10]
```

```
 [1] 11.64601 12.38171 14.43874 15.61393 16.29209 16.68659 17.89474 19.06858
 [9] 19.85527 21.36689
```

6. Use matrix multiplication to compute the average of each column of `z` and store in a single row matrix. Hint define a $1 \times n$ matrix $(1/n, \dots, 1/n)$ with n the `nrow(z)`. Show the first 10 elements

```
(matrix(rep(1/nrow(z), nrow(z)), nrow = 1) %*% z)[1, 1:10]
```

```
 [1] 52.49472 54.50904 56.50924 58.44142 60.52604 62.56375 64.51164 66.42631
 [9] 68.57376 70.46589
```

7. Use matrix multiplication and other matrix / vector operations to compute the standard deviation of each column of z. Do not use `sweep` or `apply`. Print the results. For this exercise, you must only use the following operations: `t`, `-`, `%*%`, `*`, `/`, and `as.vector`

```r
as.vector(sqrt(colSums((z - matrix(rep(1, nrow(z)), ncol = 1) %*% (matrix(rep(1/nrow(z), nro
```

```
 [1] 28.84534 29.28167 29.10108 29.05247 29.00242 29.12466 28.79415 29.00033
 [9] 29.00299 29.05853
```

8. For each digit in the MNIST training data, compute and print the overall proportion of pixels that are in a *grey area*, defined as values between 50 and 205, inclusive. Hint: use the `read_mnist` function from the `dslabs` package.

```r
library(dslabs)
```

```
Warning: package 'dslabs' was built under R version 4.4.3
```

```r
mnist <- read_mnist()

is_gray <- mnist$train$images >= 50 & mnist$train$images <= 205

gray_prop_overall <- sapply(0:9, function(d) {
  images_d <- mnist$train$images[mnist$train$labels == d, ]
  total_pixels <- length(images_d)
  gray_pixels <- sum(images_d >= 50 & images_d <= 205)
  gray_pixels / total_pixels
})

gray_prop_overall
```

```
 [1] 0.07478220 0.03599523 0.06893878 0.06873338 0.06125680 0.06802156
 [7] 0.06449624 0.05501022 0.07290333 0.06190673
```

9. Compute and print the average grey proportion by digit class. Hint: Use logical operators and `sapply`.

```r
is_gray <- mnist$train$images >= 50 & mnist$train$images <= 205
gray_per_image <- rowMeans(is_gray)

gray_avg_by_digit <- sapply(0:9, function(d) {
  mean(gray_per_image[mnist$train$labels == d])
})

gray_avg_by_digit
```

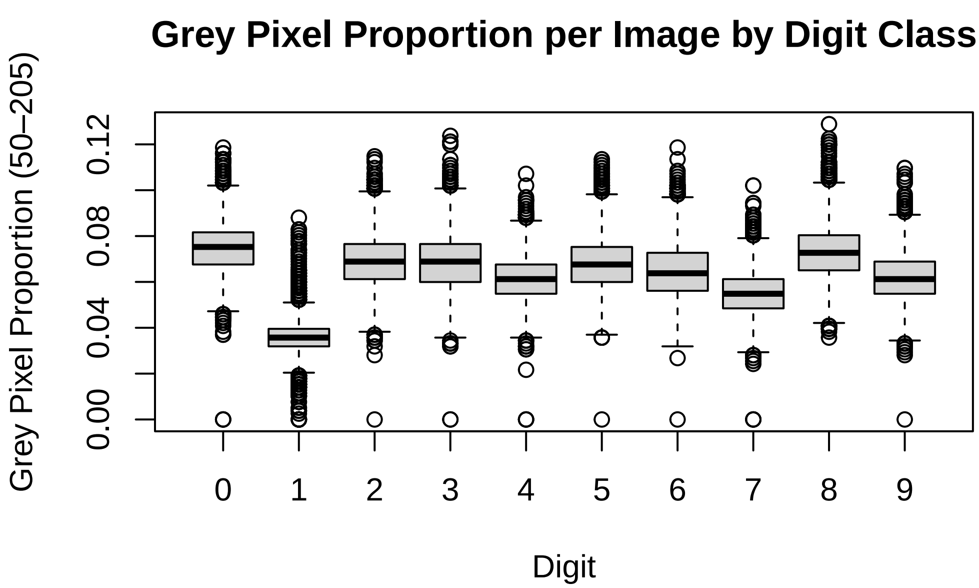
```
[1] 0.07478220 0.03599523 0.06893878 0.06873338 0.06125680 0.06802156
[7] 0.06449624 0.05501022 0.07290333 0.06190673
```

10. Make a box plot of grey proportion by digit class. Each point on the boxplot should represent one training image. Hint: Use logical operators and `rowMeans`.

```r
gray_prop_per_image <- rowMeans(mnist$train$images >= 50 & mnist$train$images <= 205)

labels <- mnist$train$labels

boxplot(gray_prop_per_image ~ labels,
        main = "Grey Pixel Proportion per Image by Digit Class",
        xlab = "Digit",
        ylab = "Grey Pixel Proportion (50-205)",
        col = "lightgray")
```



11. Use the function `solve` to solve the following system of equations. Hint: use the function `solve`. Show the solution.

$$x + 2y - 2z = -15 \quad (1)$$
$$2x + y - 5z = -21 \quad (2)$$
$$x - 4y + z = 18 \quad (3)$$

```r
A <- matrix(c(1, 2, -2,
              2, 1, -5,
              1, -4, 1), nrow = 3, byrow = TRUE)

b <- c(-15, -21, 18)
solution <- solve(A, b)
cat("x =", solution[1], "\ny =", solution[2], "\nz =", solution[3])
```

```
x = -1
y = -4
z = 3
```