Homework 4

Stats 20 Lec 1

Winter 2023

General Guidelines

Please use R Markdown for your submission. Include the following files:

- Your .Rmd file.
- The compiled/knitted HTML document.

Name your .Rmd file with the convention `123456789_stats20_hw0.Rmd`, where `123456789` is replaced with your UID and `hw0` is updated to the actual homework number. Include your first and last name and UID in your exam as well. When you knit to HTML, the HTML file will inherit the same naming convention.

The knitted document should be clear, well-formatted, and contain all relevant R code, output, and explanations. R code style should follow the Tidyverse style guide: https://style.tidyverse.org/.

Any and all course material, including these homework questions, may not be posted online or shared with anyone at any time without explicit written permission by the instructor (Michael Tsiang), even after the quarter is over. Failure to comply is a breach of academic integrity.

Note: All questions on this homework should be done using only functions or syntax discussed in Chapters 1–6 of the lecture notes or Homeworks 1–4. No credit will be given for use of outside functions.

Basic Questions

Collaboration on basic questions must adhere to Level 0 collaboration described in the Stats 20 Collaboration Policy.

Question 1

The objective of this question is to give practice with writing functions involving matrices.

For any matrix A, the **transpose** of A, denoted A^T (or sometimes A'), is the matrix whose rows are the columns of A and whose columns are the rows of A. For example:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

The `t()` function returns the **transpose** of an input matrix.

Write a function called `my_t()` that returns the transpose of a matrix without the `t()` function. Account for vector or matrix input. The output of `my_t(x)` and `t(x)` should be identical for any vector or matrix `x`.

Question 2

The objective of this question is to show how R can be applied in the context of linear regression using matrices.

(a)

Given n pairs of values $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, the sample (Pearson) correlation coefficient r is defined by

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}.$$

The `cor()` function can input two vector arguments of the same length and output the correlation coefficient between them.

Write a function called `my_cor()` that computes the correlation between two numeric vectors `x` and `y` without the `cor()`, `cov()`, `var()`, or `sd()` functions. Include a character argument called `use` that specifies whether to use all observations (`use = "everything"`) or only pairwise complete observations (`use = "pairwise.complete.obs"`) by removing a pair (x_i, y_i) if either x_i or y_i is missing. The output of `my_cor(x, y)` and `cor(x, y)` should be identical for any numeric vectors `x` and `y`.

Hint: The `cor()` function also allows for matrix inputs `x` and `y` and computes the correlation matrix between columns of `x` and columns of `y`. Your `my_cor()` function does not need to include this functionality.

(b)

Assume there is a linear relationship between variables x and y. The least squares regression line is a linear model that predicts y from x. The equation of the regression line is denoted by $y = a + bx$.

The coefficients of the regression line are computed by the formulas

$$b = r\frac{s_y}{s_x} \quad \text{and} \quad a = \bar{y} - b\bar{x},$$

where $\bar{x}$ and $\bar{y}$ and the respective means and s_x and s_y are the respective standard deviations for the data vectors x and y.

Write a function called `linreg()` that inputs two numeric vectors `x` and `y` and outputs a numeric vector of length 2 that corresponds to the coefficients `a` and `b` of the least squares regression line that predicts `y` from `x`. Include a character argument called `use` that specifies whether to use all observations (`use = "everything"`) or only pairwise complete observations (`use = "pairwise.complete.obs"`) by removing a pair (x_i, y_i) if either x_i or y_i is missing.

(c)

The heights and weights of six self-identified women are given below.

Height (inches)	Weight (pounds)
61	104
62	110
63	125
64	141
66	160
68	170

Assume there is a linear relationship between height and weight. We want to fit a linear regression model that predicts weight from height. Use your `linreg()` function from (b) to find the equation of the regression line $y = a + bx$.

(d)

Write a function called `linreg_mat()` that inputs two numeric vectors `x` and `y` and outputs the expression $(X^T X)^{-1} X^T y$, where X is the **design matrix** given by

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix},$$

where $x_1, x_2, \ldots, x_n$ are the observed data values of the predictor variable x (n is the sample size).

Hint 1: Add a column of 1's to the predictor (explanatory) vector to create the design matrix X.

Hint 2: Make sure to check that the vector arguments are the same length.

(e)

Use your `linreg_mat()` function from (d) on the height and weight data in (c). Compare your answer with your results from part (c).

(f)

Interpret the slope coefficient in the context of the data.

Question 3

The objective of this question is to introduce how to write infix operators and give further practice writing functions with loops and matrices.

From the Wikipedia article on matrix multiplication (https://en.wikipedia.org/wiki/Matrix_multiplication):

If A is an $m \times n$ matrix and B is an $n \times p$ matrix,

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \quad B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{bmatrix},$$

the **matrix product** $C = AB$ is defined to be the $m \times p$ matrix

$$C = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1p} \\ c_{21} & c_{22} & \cdots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mp} \end{bmatrix}$$

such that

$$c_{ij} = a_{i1}b_{1j} + \cdots + a_{in}b_{nj} = \sum_{k=1}^{n} a_{ik}b_{kj},$$

for $i = 1, \ldots, m$ and $j = 1, \ldots, p$.

(a)

Write an **infix operator** function called `%m%` that inputs two numeric matrix arguments `A` and `B` and outputs the matrix product of `A` and `B`. The body of `%m%` *cannot* use `%*%`. The output of `A %m% B` and `A %*% B` should be identical for any numeric matrices `A` and `B`.

Hint 1: To create an infix operator, the name of the function must be contained within backticks. That is, assign the function to the name `` `%m%` ``.

Hint 2: Make sure to check that the matrix arguments are conformable.

(b)

Verify that your `%m%` operator in (a) works on

$$X = \begin{bmatrix} 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix} \quad \text{and} \quad Y = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}.$$

Hint: You may use `%*%` to check that your function outputs the correct product.

Question 4

The objective of this question is to show how to apply matrix multiplication to matrix exponentiation and give further practice with writing functions with loops and matrices.

For nonnegative integers k, the **power** of a square matrix A is the matrix product of k copies of A. For example, if A is an $n \times n$ matrix, then

$$\begin{aligned} A^0 &= I_n \\ A^1 &= A \\ A^2 &= AA \\ \vdots &= \vdots \\ A^k &= \underbrace{AA \cdots A}_{k \text{ copies}}, \end{aligned}$$

where I_n is the $n \times n$ identity matrix.

(a)

Use your matrix multiplication infix operator `%m%` from Question 5 to write an infix operator function called `%^%` such that `A %^% k` outputs the `k`th power of a square numeric matrix `A`. The body of `%^%` *cannot* use `%*%`.

(b)

Consider the matrix Z defined to be

$$Z = \begin{bmatrix} 0.2 & 0.7 & 0.1 \\ 0.6 & 0.2 & 0.2 \\ 0.4 & 0.1 & 0.5 \end{bmatrix}.$$

Use your `%^%` operator in (a) to compute Z^0, Z^5, Z^{50}, and Z^{500}.

Note: Notice that the values in each row of Z sum to 1. Matrix Z is an example of a **stochastic** matrix, which describes transition probabilities between states of a Markov chain. To learn more about Markov chains, take Stats 102C.

Question 5

The objective of this question is to give practice with logical indices and writing functions with different types of output.

Write a function called `my_which()` that returns the indices of `TRUE` values in a logical vector or matrix without the `which()` function. Include an optional logical argument called `arr.ind` argument with a default of `FALSE` to optionally return two dimensional indices of `TRUE` values in a logical matrix. The output of `my_which(x)` and `which(x)` should be identical for any logical vector or matrix `x`.

Question 6

The objective of this question is to give practice with using vectorization with factors.

Download the `mlb.RData`[1] file from BruinLearn and save it to your working directory. Then run the command `load("mlb.RData")` to load data from the 2018 season of Major League Baseball. This command will create 6 objects in your workspace:

```
league: A factor with 2 levels indicating if the player plays for a team in the
        American League (AL) or the National League (NL).
team:   A factor with 30 levels indicating which Major League Baseball team the
        player is a member of.
pos:    A factor with 9 levels indicating the primary position of the player.
ab:     An integer vector indicating the number of "at bats" the player had.
hit:    An integer vector indicating the number of "hits" the player had.
hr:     An integer vector indicating the number of "home runs" the player had.
```

(a)

i. Using one command find the maximum number of **hits** from each team..

ii. Using one command find the number of players on each team.

iii. Using one command and no subsetting, find the number of players on each team with at least one **home run**.

(b)

A definition: **Batting Average**: the batting average of a player (or a team), is calculated by the number of **hits** (`hit`) divided by the number of **at bats** (`ab`).

i. Using one command, find the highest **batting average** for each team among players with at least 100 **at bats**.

ii. Using one command, find the **batting average** for each team.

(c)

Using one command, find the average number of **home runs** for each position in each league. Which position has the largest difference between leagues?

(d)

Using one command, compute the median number of players for each position on any team.

[1] The data found in `mlb.RData` was obtained from the Lahman R package in CRAN, with modifications.

Intermediate Questions

Collaboration on intermediate questions must adhere to Level 1 collaboration described in the Stats 20 Collaboration Policy.

Question 7

The objective of this question is to give practice writing a function with different outputs for different types of inputs and give further practice with loops and matrices.

(a)

Write a function called `my_row()` that returns a matrix of integers indicating the row number (i.e., the `ij`th element is equal to `i`) without the `row()` function. The output of `my_row(x)` and `row(x)` should be identical for any matrix `x`.

(b)

Write a function called `my_col()` that returns a matrix of integers indicating the column number (i.e., the `ij`th element is equal to `j`) without the `col()` function. The output of `my_col(x)` and `col(x)` should be identical for any matrix `x`.

(c)

Write a function called `my_diag()` that returns the same output as `diag()` without the `diag()` function. Account for different types of inputs (scalars, vectors, matrices). Include optional arguments `nrow` and `ncol` that specify the dimensions of the output matrix when the input object is a vector. The output of `my_diag(x)` and `diag(x)` should be identical for any vector or matrix `x`.

Hint: In addition to the lecture notes, you may use the `missing()` function, if necessary. When used inside the body of a function, the `missing()` function returns `TRUE` if a formal argument of the function is missing (i.e., not specified) and has no default value. For more guidance on how to use the `missing()` function, read the "Formal Arguments" document in the Required Reading on BruinLearn.

Advanced Questions

Collaboration on advanced questions must adhere to Level 1 collaboration described in the Stats 20 Collaboration Policy.

Note: Advanced Questions are intended for further enrichment and a deeper challenge, so they will not count against your grade if they are not completed or attempted.

Question 8

Watch this video for an explanation of how ranked choice voting works: https://youtu.be/8Z2fRPRkWvY (For anyone who needs it, a transcript can be found here: https://bit.ly/32G4bWK)

Download the `votes.RData` file from BruinLearn and save it to your working directory. Then run the following command to load the `votes` object in your workspace:

```
load("votes.RData")
```

Note: Do *not* print the entire `votes` object. It is *extremely* bad practice/style to output more than about 10 rows of a matrix (or data frame).

(a)

Pseudocode (or outline) a function that inputs a matrix of ranked choice votes and returns a matrix of results.

(b)

Write a function called `tally_rcv()` that inputs a matrix of ranked choice votes and returns a matrix of results.

- Your `tally_rcv()` function must be able to handle any number of choices/candidates and any number of voters.
- The column names of the output matrix must be the names of the choices/candidates.
- The row names of the output matrix must correspond to the appropriate round numbers.
- If there is a tie for last place in a round, you must eliminate candidates in the following manner:
  - If the tie occurs in the first round, eliminate the tied candidate who comes last alphabetically by first name.
  - In any subsequent round:
    - If the sum of the the tied candidates votes is less than the number of votes for the next lowest candidate, eliminate *both* of the tied candidates.
    - Eliminate the tied candidate with the least votes in the previous round.

Hint 1: In addition to the lecture notes, you may use the `order()` and/or `rank()` functions, if necessary. The `order()` function inputs a vector and outputs the indices of the input vector that will return the sorted values. The `rank()` function inputs a vector and outputs the relative rank of each element.

Hint 2: It may be helpful to consider eliminating a candidate by setting all of their ranks to `Inf`.

(c)

It is Election Day in Pawnee! Use your `tally_rcv()` function on the `votes` data and print the results using the function `knitr::kable()`. The `knitr::kable()` function will print the output matrix in a well-formatted table after knitting your file.