Homework1_Instructions

Aidan O'Hara

2023-09-06

Vector

1. Create the following vector:
(a) (1,2,3,. . . ,19,20), Name `v1`;
(b) (20,19,18,. . . ,2,1), Name `v2`;
(c) (1,3,5,. . . ,17,19), Name `v3`;
(d) (3,7,11,3,7,11,. . . ,3,7,11) where there are 10 occurrences of 3, Name `v4`;
(e) (3,7,11,3,7,11,. . . ,3) where there are 11 occurrences of 3, 10 occurrences of 7 and 10 occurrences of 11, Name `v5`.

2. Create a vector of the values of $e^x sin(x)$ at x = 3.0,3.1,3.2,. . . ,6.0 Name `x1`;

3. Calculate the following, name your answer `sum1`.

$$\sum_{i=10}^{100}(i^3 + 4i^2)$$

4. Use the function `paste` to create the following character vectors of length 30:
(a). ("label 1", "label 2", ..., "label 30"). Notice: there is a single space between 'label' and number following, Name `str1`;
(b). ("function1", "function2", . . . , "function30"). In this case, there is no space between 'function' and number following, Name `str2`;

Matrix

1. Suppose:

$$A = \begin{bmatrix} 1 & 1 & 3 \\ 5 & 2 & 6 \\ -2 & -1 & -3 \end{bmatrix}$$

(a) Check the $A^3 = 0$ where 0 is a 3×3 matrix with every entry equal to 0. Name `neo`
(b) Replace the third column of A by the sum of the second and third columns. Name `neo2`;

2.

$$B = \begin{bmatrix} 10 & -10 & 10 \\ 10 & -10 & 10 \\ ... & ... & ... \\ 10 & -10 & 10 \end{bmatrix}$$

Calculate and save the 3×3 matrix $B^t B$ (look at the help for `crossprod`.) Name `redOrBlue`;

3. Solve the following system of linear equations in five unknowns by considering an appropriate matrix equation $Ax = y$.

$$x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 = 7$$
$$2x_1 + x_2 + 2x_3 + 3x_4 + 4x_5 = -1$$
$$3x_1 + 2x_2 + x_3 + 2x_4 + 3x_5 = -3$$
$$4x_1 + 3x_2 + 2x_3 + x_4 + 2x_5 = 5$$
$$5x_1 + 4x_2 + 3x_3 + 2x_4 + x_5 = 17$$

Make use of the special form of the matrix A. The method used for the solution should easily generalize to a larger set of equations where the matrix A has the same structure; hence the solution should not involve typing in every number of A. *Use* `solve` *function.* Name: `smith`;

4. Create a 6×10 matrix of random integers chosen from 1, 2,..., 10 by executing the following two lines of code:

```
set.seed(75)
aMat <- matrix( sample(10, size=60, replace=T), nr=6)
```

(a) Find the number of entries in each row which are greater than 4. Name `fours`;
(b) Which rows contain exactly two occurrences of the number seven? Name `seven`;
(c) Find those pairs of columns whose total (over both columns) is greater than 75. The answer should be a matrix with two columns; so, form example, the row (1, 2) in the output matrix means that the sum of columns 1 and 2 in the original matrix is greater than 75. Repeating a column is permitted; so, for example, the final output matrix could contain the rows (1, 2), (2, 1) and (2, 2). Name `counts`;

5. Calculate the following and assign them to the given variable name:
(a) `somewhere`

$$\sum_{i=1}^{20} \sum_{j=1}^{5} \frac{i^4}{(3+j)}$$

(b) `something`

$$\sum_{i=1}^{20} \sum_{j=1}^{5} \frac{i^4}{(3+ij)}$$

(c) `somebody`

$$\sum_{i=1}^{10} \sum_{j=1}^{i} \frac{i^4}{(3+ij)}$$

Data Frame

1. Write a function, called `myListFn`, which takes a single argument n and implements the following algorithm:

1. Simulate `n` independent numbers, denoted $\mathbf{x} = (x_1, x_2, ..., x_n)$, from the N(0,1) distribution.
2. Calculate the mean, $\bar{x}$.
3. If $\bar{x} > 0$, simulate `n` independent numbers, denoted $\mathbf{y} = (y_1, y_2, ..., y_n)$, from the exponential density with mean $\bar{x}$. If $\bar{x} < 0$ then simulate n independent numbers, denoted $\mathbf{z} = (z_1, z_2, ..., z_n)$ from the exponential density with mean $-\bar{x}$. Set $\mathbf{y} = (y_1, y_2, ..., y_n) = -\mathbf{y}$.
4. Calculate k which is the number of j with $|y_j| > |x_j|$.
5. Return the list of $\mathbf{x}$, $\mathbf{y}$, and k with names `xVec`, `yVec`, and `count` respectively.

2. In order to test the functions in this question, you will need an `array`. We can create a three dimensional test `array` as follows:
`testArray <- array( sample( 1:60, 60, replace=F), dim=c(5,4,3) )`

The above line creates a $5 \times 4 \times 3$ array of integers which can be represented in mathematics by:
$\{x_{i,j,k} : i = 1, 2, ..., 5; j = 1, 2, 3, 4; k = 1, 2, 3\}$

Note that `apply(testArray, 3, tmpFn)` mean that the index k is retained in the answer and the function `tmpFn` is applied to the 3 matrices:
$\{x_{i,j,1} : 1 \le i \le 5; 1 \le j \le 4\}$,
$\{x_{i,j,2} : 1 \le i \le 5; 1 \le j \le 4\}$ and $\{x_{i,j,1} : 1 \le i \le 5; 1 \le j \le 4\}$.
Similarly `apply(testArray, c(1,3), tmpFn)` means that indices i and k are retained in the answer and the function `tmpFn` is applied to 15 vectors: $\{x_{1,j,1} : 1 \le j \le 4\}$, $\{x_{2,j,1} : 1 \le j \le 4\}$, etc.
The expression `apply(testArray, c(3,1), tmpFn)` does the same calculation but the format of the answer is different: when using `apply` in this manner, it is always worth writing a small example in order to check that the format of the output of `apply` is as you expect.

(a) Write a function `testFn` which takes a single argument which is a 3-dimensional array. If this array is denoted $\{x_{i,j,k} : i = 1, 2..., d_1; j = 1, 2, ..., d_2; k = 1, 2, ..., d_3\}$, then the function `testFn` returns a list of the $d_1 \times d_2 \times d_3$ matrix $\{w_{i,j,k}\}$ and the $d_2 \times d_3$ matrix $\{z_{j,k}\}$ where:

$w_{i,j,k} = x_{i,j,k} - \min_{i=1}^{d_1} x_{i,j,k}$ and $z_{j,k} = \sum_{i=1}^{d_1} x_{i,j,k} - \max_{i=1}^{d_1} x_{i,j,k}$

(b) Now suppose we want a function `testFn2` which returns the $d_2 \times d_3$ matrix $\{z_{j,k}\}$ where
$z_{j,k} = \sum_{i=1}^{d_1} x_{i,j,k}^k$

3. Load the `penguins` data set from `library(palmerpenguins)`. Copy the data to a new variable `peng` and then create the following new columns: - `bill_depth_in`: bill depth in inches. - `bill_length_in`: bill length in inches. - `flipper_length_in`: flipper length in inches. - `body_mass_kg`: body mass in kilograms - `body_mass_lb`: body mass in pounds

Convert mm to in by using a factor of `0.0393701`, and kg to lb using a factor of `2.20462`.

4. Create three subsets of `peng`, called `pengPres1`, `pengPres2`, `pengPres3`. Each subset should include only Gentoo, Chinstrap, and Adelie species respectively. Only include `flipper_length_in` and `body_mass_lb`, omit any rows with `NA` values.

5. Remove any rows from `peng` that contain `NA` values. Construct `groupPengs` to contain 100 rows about samples of 5 rows from `peng`. Populate your rows with the following columns:

- `groupMass_g`: combined mass in grams of all the penguins in the sample
- `longest_beak_mm`: the longest beak in mm recorded in the sample
- `happy`: a binary indicating if all 5 penguins are the same species
- `wellFed`: a binary indicating if the the mean of the sample is greater than the mean mass of penguins from `peng`

Function

1. Create two functions `function1` and `function2`,

(a). `function1(xv)` return the vector $(x_1, x_2^2, ..., x_n^n)$, calculate vector $xv = (0.0, 0.1, ..., 0.8, 0.9, 1.0)$ Name: `func1_ans`;

(b). `function2(xv)` return $(x_1, \frac{x_2^2}{2}, \frac{x_3^3}{3}, ..., \frac{x_n^n}{n})$, calculate vector $xv = (0.0, 0.1, ..., 0.8, 0.9, 1.0)$ Name: `func2_ans`;

(c). Write a function `function3` which takes 2 arguments x and n where x is a single number and n is a strictly positive integer. The function should return the value of

$$1 + \frac{x}{1} + \frac{x^2}{2} + \frac{x^3}{3} + ... + \frac{x^n}{n}$$

then calculate vector $xv = (0.0, 0.1, ..., 0.8, 0.9, 1.0)$
Name the variable: `func3_ans`;

2. Write a function called `findodd` that returns a lists the odd numbers between 1 and input `xv`.

3. Write a function that receives two inputs, a number x, and another number y, the function should return x or the next largest number divisible by y. *hint: use the modulo operator `%%`.*

Name: `modNumber(x,y)`.

ex: `modNumber(50,16)` should return `[1] 64` and `modNumber(64,16)` should also return `[1] 64`

Your supplied `modNumber(x,y)` should return `[1] 64` and `modNumber(64,16)` should also return `[1] 64`.

4. Write a function, `matFn`, which takes 2 arguments n and k which are positive integers. It should return the $n \times n$ matrix:

$$\begin{bmatrix} k & 1 & 0 & 0 & ... & 0 & 0 \\ 1 & k & 1 & 0 & ... & 0 & 0 \\ 0 & 1 & k & 1 & ... & 0 & 0 \\ 0 & 0 & 1 & k & ... & 0 & 0 \\ ... & ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & 0 & ... & k & 1 \\ 0 & 0 & 0 & 0 & ... & 1 & k \end{bmatrix}$$

Hint: First try to do it for a specific case such as $n = 5$ and $k = 2$ on the Command Line.

5. Zeller's congruence is the formula: $f = ([2.6m - 0.2] + k + y + [y/4] + [c/4] - 2c) mod 7$ where $[x]$ denotes the integer part of x; for example $[7.5] = 7$. Zeller's congruence returns the day of the week f given: $k =$ the day of the month,
$y =$ the year in the century
$c =$ the first 2 digits of the year (the century number)
$m =$ the month o=number (where January is month 11 of the preceding year, February is month 12 of the preceding year, March is month 1, etc.)

For example, the date 21/07/1963 has m = 5, k = 21, c = 19, y = 63; whilst the date 21/2/1963 has m = 12, k = 21, c = 19 and y = 62.

Write a function `weekday(day,month,year)` which returns the day of the week when given the numerical inputs of the day, month and year. Note that the value of 1 for f denotes Sunday, 2 denotes Monday, etc.

More Function

1.
(a) Write a function `quadmap( start, rho, niter)` which returns the vector $(x_1, ..., x_n)$ where $rx_{k-1}(1 - x_{k-1})$ and - `niter` denotes n, - `start` denotes x_1 - `rho` denotes r.

Try out the function you have written: - for $ r=2 $ and $0 < x_1 < 1$ you should get $x_n \to 0.5$ as $n \to \infty$. - try `tmp <- quadmap(start = 0.95, rho=2.99, niter=500)`, try `plot(tmp, type="l")` in console.

(b) Now write a function, `quad2`, which determines the number of iterations needed to get $|x_n - x_{n-1}|<0.02. So this function has only 2 arguments: `start` and `rho` (For `start=0.95` and `rho=2.99`, the answer is 84.)

2.
(a) Suppose `matA` is a matrix containing some occurrences of `NA`. Pick out the submatrix which consists of all columns which contain no occurrence of `NA`. So the objective is to write a function which takes a single argument which can be assumed to be a matrix and returns a matrix.

(b) Now write a function which takes a single argument which can be assumed to be a matrix and returns the submatrix which is obtained by deleting every row and column from the input matrix which contains an `NA`.

3.
Experiment with different ways of defining a function which calculates the following double sum for any value of n.

$$f(n) = \sum_{i=l}^{n} \sum_{s=l}^{r} \frac{s^2}{10 + 4r^3}$$

(a) First use a loop within a loop.

(b) Write a function `funB` which uses the functions row and col to construct a matrix with suitable entries so that the sum of the matrix gives the required answer.

(c) Write a function `funC` which uses the function outer to construct a matrix with suitable entries so that the sum of the matrix gives the required answer.

(d) Create a function which takes a sungle argument `r` and calculates

$$\sum_{s=l}^{r} \frac{s^2}{10 + 4r^3}$$

Then write a function `funD` which uses `sapply` to calculate the double sum. Note that `sapply` is just a combination of `unlist` and `lapply`. Is there any increase in speed gained by using this information (`funE`)?

(e) Write a function which takes two arguments `r` and `s` and calculates

$$\frac{I(s \leq r)s^2}{10 + 4r^3}$$

where I denotes the indicator function. Then write a function `funF` which calculates the double sum by using mapply to calculate all the individual terms.

(f) Determine which function is fastest. Save your fastest function as `fastestFun`.

4. The waiting time of the n^th customer in a single server queue. Suppose customers labelled $C_0, C_1, ..., C_n$ arrive at times $\tau = 0, \tau_1, ..., \tau_n$ for service by a single server. The inter-arrival times $A_1 = \tau_1 - _0, ..., A_n = \tau_n - \tau_{n-1}$ are independent and identically distributed random variables with the exponential density

$$\lambda_a e^{-\lambda_a x} \text{ for } x \geq 0$$

The service times $S_0, S_1, ..., S_n$ are independent and identically distributed random variables which are also independent of the inter-arrival times with the exponential density

$$\lambda_s e^{-\lambda_s x} \text{ for } x \geq 0$$

Let W_j denote the waiting time of customer C_j . Hence customer C_j leaves at time $\tau_j + W_j + S_j$. If this time is greater than τ_{j+1} then the next customer, C_{j+1} must wait for the time $\tau_j + W_j + S_j - \tau_{j+1}$. Hence we have the recurrent relation $W_0 = 0$, $W_{j+1} = \max\{0, W_j + S_j - A_{j+1}\}$ for $j = 0, 1, ..., n-1$

Write a function `queue(n, aRate, sRate)` which simulates one outcome of W_n where `aRate` denotes λ_a and `sRate` denotes λ_s. Try out your function on an example such as `queue(50,2,2)`.

5. A random walk. A symmetric simple random walk starting at the origin is defined as follows. Suppose $X_1, X_2, ...$ are independent and identically distributed random variables with the distribution: $+1$ with probability $1/2$ or -1 with probability $1/2$

Define the sequence $\{S_n\}_{n \geq 0}$ by

$$S_0 = 0 S_n = S_{n-1} + X_n \text{ for } n = 1, 2, ...$$

Then $\{S_n\}_{n \geq 0}$ is a symmetric simple random walk starting at the origin. Note that the position of the walk at time n is just the sum of the previous n steps: $S_n = X_1 + ... + X_n$.

(a) Write a function `rwalk(n)` which takes a single argument n and returns a vector which is a realization of $(S_0, S_1, ..., S_n)$, the first n positions of a symmetric random walk starting at the origin. *Hint: the code `sample(c(-1,1), n, replace=TRUE, prob=c(0.5,0.5))` simulates n* steps

(b) Now write a function `rwalkPos(n)` which simulates one occurrence of the walk which lasts for a length of time n and then returns the length of time the walk spends above the x-axis. *(Note that a walk with* *length 6 and vertices at 0, 1, 0, -1, 0, 1, 0 spends 4 units of time above the axis and 2 units of time below the axis.)*

(c) Now suppose we wish to investigate the distribution of the time the walk spends above the x-axis. This means we need a large number of replications of `rwalkPos(n)`. Write two functions: `rwalkPos1(nReps,n)` which uses a loop and `rwalkPos2(nReps,n)` which uses replicate or `sapply`. Compare the execution times of these two functions.

(d) In the previous question on the waiting time in a queue, a very substantial increase was obtained by using a vector approach. Is that possible in this case?