SectionD_tln32_HW6

HW 6

1

```
% housekeeping commands would go here, but we leave them out to keep
Grader happy

%bring in data from data file
load("HW6_vector.mat", "V")

V

%  In one line, determine how many elements in V are equal to 0.
zero_el = length(V(V == 0))

%  In one line, determine how many numbers in V are greater than or
equal to 25.
more_than_25 = length(V(V >= 25))

%  In one line, calculate the sum of all numbers in V that are positive.
pos_sum = sum(V(V > 0))

%  In one line, calculate the mean value of the numbers in V that are
less than 70.
avg_less_70 = mean(V(V < 70))

%  In one line, determine how many values in V are equal to the mode of
V.
mode_freq = length(V(V == mode(V)))

%  In one line, create a new vector that contains only the odd numbers
in V.
not_even = V(mod(V, 2) == 1)
```

2

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Trevor Nichols               %
% ENGR 130 Module HW 6         %
% Section D                    %
% 2024-09-30T21:51:18-04:00    %
```

Copy

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Question 2 %%%

clear; clc; close all;

% Load the data from the specified file

load("HW6_coaster_data.mat", "riders")

% riders = [
%     789   385   159;
%     470   648   218;
%     1028  758   414;
%     943   1098  561;
%     1201  1062  809;
%     1424  709   727;
%     1553  644   598;
%     1283  871   893;
%     1008  868   1289;
%     635   660   1304;
%     356   410   1132;
%     328   223   923;
% ];

%%%%%
%
% Define the ranges that we are interested in, with a tag for the name
of the range
% I have used a struct here as it is a datatype that suits what we are
trying to represent here
% I have learned of structs from the matlab documentation in the first
week of class
% https://www.mathworks.com/help/matlab/ref/struct.html
%
%%%%%

ranges = struct( ...
    "Slow", [-0.1 0.4], ...
    "Medium", [0.4 0.8], ...
    "Full", [0.8 1.1] ...
);

% Store the capacities and names of the rides in vectors

capacities = [ 1600 1200 1400 ];
names = ["Iron Revenge", "Free Spirit", "Century Strength"];

% Use broadcasting to calculate the utilization of the ride at different
hours
```

```
relCap = riders ./ capacities;

% This is how I would define the number of hours that are within a range
of capacity
% The sum function for logical statements treat true and 1 and false as
0
% filterCount = @(args) sum(relCap ⩾ args(1) & relCap < args(2))';

%%%%%
%
% I have used the concept of currying in order to be able to nest
mapping functions
% I have also learned of all the *fun functions in the first week of
class when I read the documentation of the language
% In this case, the cellfun function will take cells, which I have
defined to be the columns in our input matrix
% Then run the function countWithinRange on that cell, which will return
the number of elements within the cell that are in the range
% The num2cell function is a converter function that will group elements
within an array, in this case selecting columns
% Then, we apply this mapping function upon our definition of the ranges
above using the structfun function, which is one of the *fun functions
% The struct2table function later then just converts the mapped result
into a table, with the columns of names of the range struct
% We then name the rows using the array of names defined above
% https://www.mathworks.com/help/matlab/ref/num2cell.html
% https://www.mathworks.com/help/matlab/ref/cellfun.html
% https://www.mathworks.com/help/matlab/ref/structfun.html
% https://www.mathworks.com/help/matlab/ref/struct2table.html
%
%%%%%

% Count the number of elements within a range using LOGICAL INDEXING
countWithinRange = @(rang) @(row) numel(row(row ⩾ rang(1) & row <
rang(2)));
% Run a count on each column of the supplied matrix with the cellfun
function
filterCount = @(args) cellfun(countWithinRange(args), num2cell(relCap,
1))';

% Run the defined functions on the ranges variable
counts = structfun(filterCount, ranges, "UniformOutput", false);

% Convert the resulting struct to a table with columns already named
finalData = struct2table(counts);
% Name the rows according to the name array
finalData.Properties.RowNames = names;

% Show the results of the computation
```

```
disp(finalData);

% Find the maximums and their indexes using the max function
[maxi, maxv] = max(riders);

% I chose to use a table here to represent the final data as it is
consise
% We place the maximum values, index, and the name of the ride in the
columns
maxInfo = struct2table(struct( ...
    "name", names', ...
    "guests_per_hr", maxi', ...
    "hour", maxv' ...
));

%%%%%
%
% Use the rowfun function to apply a function to each row of the table
% The sprintf function formats a string with the information from the
rows of the table
% Then, we use the varfun function to join all the rows into one long
string
% https://www.mathworks.com/help/matlab/ref/varfun.html
% https://www.mathworks.com/help/matlab/ref/rowfun.html
% https://www.mathworks.com/help/matlab/ref/string.sprintf.html
%
%%%%%

% Map the table into a formatted string to be displayed
displayString = varfun(@join, rowfun(@(n, m, i) sprintf("%s had the
maximum ridership at hour %i with %i riders\n", n, i, m), maxInfo));

% Display the table and its formatted string
disp(maxInfo);
disp(displayString{:,:})
```

3

```
%%% Question 3 %%%

clear; clc; close all;

% Load the data from the specified file into variable heartbeat
load("HW6_Heartbeat.mat", "heartbeat");

% Mark down the length of the heartbeat
s = size(heartbeat, 2);
```

```
% Construct a table that has the point names, ranges, and function to
call within the range
% Same sources as the same pattern above in question 2

hbranges = struct2table(struct( ...
    "point", [{"P"} {"Q"} {"R"} {"S"} {"T"}]', ...
    "ranges", [{[0.1 0.25]} {[0.25 0.32]} {[0.3 0.4]} {[0.35 0.45]}
{[0.5 0.7]}]', ...
    "functions", [{@max} {@min} {@max} {@min} {@max}]' ...
));
hbranges.Properties.RowNames = hbranges.point;

% Calculate the important voltage point using the function specified
upon the range of the heartbeat
% We use the rowfun function to call this on each range we care about
% We use logical indexing the create a slice of the heartbeat array
% Then we store the results back into our table in the voltage column
hbranges.voltage = rowfun(@(point, range, fun) fun{1}
(heartbeat(s*range(1):s*range(2))), hbranges, "OutputVariableNames",
"key").key;

% Display our results for PQRST
disp(hbranges)

% Determine if we need to have a checkup based on the distance from P to
T, and the distance from Q to S
needsCheckup = (abs(hbranges{"P", "voltage"} - hbranges{"T", "voltage"})
> 0.1) & (abs(hbranges{"Q", "voltage"} - hbranges{"S", "voltage"}) <
0.25);

% Tell user if a checkup is needed
if needsCheckup
    disp("You need a checkup");
end

% Extract a column in order to save to a mat file
voltages = hbranges.voltage;

% Save results to the mat file
save("HW6_results.mat", "hbranges", "voltages");
```

4

- We used eighth inch plywood and the laser cutter and 3d-printers to create out light box
- I designed the pattern and cutting of the plywood, which was then cut with the laser cutter

- Vectorizing the pattern that we wanted so that it may be cut
- I think it was a smooth experience except for not having any time to do it in class
- I liked the color gels to create colored shadows
- Laser cutters are cool