Aaron Feinberg
CMPSC-497
Professor Avanzato

Part 1: Data Analysis

Given a list of voltage measurements denoted in my code as `v_measurements`, a statistical analysis is performed in MATLAB.

analysis.mlx

```
%% Analysis of Voltages


% This program will perform statisticall analysis on a time series
of voltages


v_measurements = [21.2, 19.5, 20.1, 18.3, 17.7, 15.0, 21.9, 24.7,
23.1, 20.2, 16.3, 22.8, 18.4, 23.5, 21.1]

% (1) Minimum
v_min = min(v_measurements)
fprintf("Average Voltage: %.2f\n",v_avg)


% (2) Maximum
v_max = max(v_measurements)
fprintf("Max Voltage: %.2f\n",v_max)

% (3) Average
v_avg = mean(v_measurements)
fprintf("Average Voltage: %.2f\n",v_avg)



% (4) Std. Deviation
```

```
v_std = std(v_measurements)
fprintf("Std. Deviation of Voltages: %.2f\n",v_std)


% (5) Median
v_med = median(v_measurements)
fprintf("Median Voltage: %.2f\n",v_med)

% (6) Num. Values > Avg
num_vals_greater_than_avg = sum(v_measurements>v_avg)

fprintf("Number of Voltage Values > %.2f: %.f\n",v_avg,
num_vals_greater_than_avg)

% (7) Values > Avg
v_greater_than_avg = v_measurements(v_measurements > v_avg)
fprintf("Voltage Greater than Average: %.2f\n", v_greater_than_avg)

% (8) Raw Plot
plot(v_measurements)
title('Voltage Measurements')
xlabel('Sample #')
ylabel('Voltage')

% (9) Histogram
histogram(v_measurements)
title('Histogram of Voltages')
xlabel('Voltage')
ylabel('Occurances')

% (10) Sorted
v_sorted = sort(v_measurements)

% printing sorted measurements in new lines
disp("Voltages in sorted order: ")
for i = 1:length(v_sorted)
        fprintf("%.2f\n", v_sorted(i));
end
```

Output

```
v_measurements = 1×15
   21.2000   19.5000   20.1000   18.3000   17.7000   15.0000
21.9000   24.7000   23.1000   20.2000   16.3000   22.8000   18.4000
23.5000   21.1000

v_min = 15

Average Voltage: 20.25

v_max = 24.7000

Max Voltage: 24.70

v_avg = 20.2533

Average Voltage: 20.25

v_std = 2.7622

Std. Deviation of Voltages: 2.76

v_med = 20.2000

Median Voltage: 20.20

num_vals_greater_than_avg = 7

Number of Voltage Values > 20.25: 7

v_greater_than_avg = 1×7
   21.2000   21.9000   24.7000   23.1000   22.8000   23.5000
21.1000

Voltage Greater than Average: 21.20
Voltage Greater than Average: 21.90
Voltage Greater than Average: 24.70
Voltage Greater than Average: 23.10
Voltage Greater than Average: 22.80
```

```
Voltage Greater than Average: 23.50
Voltage Greater than Average: 21.10


v_sorted = 1×15
   15.0000   16.3000   17.7000   18.3000   18.4000   19.5000
20.1000   20.2000   21.1000   21.2000   21.9000   22.8000   23.1000
23.5000   24.7000

Voltages in sorted order:

15.00
16.30
17.70
18.30
18.40
19.50
20.10
20.20
21.10
21.20
21.90
22.80
23.10
23.50
24.70userOutput =
```

Graphs

Part 2: Resistor Calculations

These are two MATLAB scripts each with 3 test values separate to calculate the total resistance of n - resistors in:

1. series

2. parallel

Series Calculation

To find the total resistance of a chain of resistors wired in series we can apply the formula

$$R_{total} = R_1 + R_2 + R_3 + \ldots + R_n$$

series.mlx

```
%% Resistors in Series

% This program will ask the user to specify the
% number of resistors in a chain and then have
% the user specify the resistance values for each resistor.
% The total resistance of these resistors wired in series
% is then returned.

r_count = input('Please input the number of resistors: ')


% getting resistance values

r_values = []

for i=1:r_count
        prompt = sprintf('Enter resistance of resistor_%d: ', i)
        r_values(i) = input(prompt)
end



r_total = sum(r_values)

fprintf("Total Resistance in series is: %d", r_total)
```

Testing `series.mlx`

Test 1: 3 Resistors

```
r_count = 3

r_values =

     []

prompt = 'Enter resistance of resistor_1: '

r_values = 1

prompt = 'Enter resistance of resistor_2: '

r_values = 1×2
     1     2

prompt = 'Enter resistance of resistor_3: '

r_values = 1×3
     1     2     3

r_total = 6

Total Resistance in series is: 6
```

Test 2: 5 Resistors

```
r_count = 5

r_values =

     []

prompt = 'Enter resistance of resistor_1: '
```

```
r_values = 100

prompt = 'Enter resistance of resistor_2: '

r_values = 1×2
         100        1000

prompt = 'Enter resistance of resistor_3: '

r_values = 1×3
         100        1000         200

prompt = 'Enter resistance of resistor_4: '

r_values = 1×4
         100        1000         200         500

prompt = 'Enter resistance of resistor_5: '

r_values = 1×5
         100        1000         200         500         425

r_total = 2225

Total Resistance in series is: 2225
```

test 3: 1 Resistor

```
r_count = 1

r_values =

     []

prompt = 'Enter resistance of resistor_1: '
```

```
r_values = 300

r_total = 300

Total Resistance in series is: 300
```

Parallel Calculation

The same calculations is done for resistors wired in parallel. The formula to obtain the total resistance for a parallel configuration of resistors is:

$$\frac{1}{((\frac{1}{R_1})+(\frac{1}{R_2})+(\frac{1}{R_3})+(\frac{1}{R_n}))}$$

parallel.mlx

```
%% Resistors in Parallel

% This program will ask the user to specify the
% number of resistors in a chain and then have
% the user specify the resistance values for each resistor.
% The total resistance of these resistors wired in parallel
% is then returned.

r_count = input('Please input the number of resistors: ')

% getting resistance values
r_values = []
denom = 0

for i=1:r_count
        prompt = sprintf('Enter resistance of resistor_%d: ', i)
        r_values(i) = input(prompt)
        denom = denom + (1/r_values(i))
end

r_total = 1/denom
```

```
sprintf("Total Resistance in series is: %2f", r_total)
```

Testing

Test: 3 resistors

```
r_count = 3

r_values =

     []

denom = 0

prompt = 'Enter resistance of resistor_1: '

r_values = 1

denom = 1

prompt = 'Enter resistance of resistor_2: '

r_values = 1×2
     1     2

denom = 1.5000

prompt = 'Enter resistance of resistor_3: '

r_values = 1×3
     1     2     3

denom = 1.8333
```

```
r_total = 0.5455

ans = "Total Resistance in series is: 0.545455"
```

Test: 5 Resistors

```
r_count = 5

r_values =

     []

denom = 0

prompt = 'Enter resistance of resistor_1: '

r_values = 10

denom = 0.1000

prompt = 'Enter resistance of resistor_2: '

r_values = 1×2
    10    20

denom = 0.1500

prompt = 'Enter resistance of resistor_3: '

r_values = 1×3
    10    20    30

denom = 0.1833

prompt = 'Enter resistance of resistor_4: '

r_values = 1×4
    10    20    30    40

denom = 0.2083
```

```
prompt = 'Enter resistance of resistor_5: '

denom = 0.2283

r_total = 4.3796

ans = "Total Resistance in series is: 4.379562"
```

Test: 1 Resistor

```
r_count = 1

r_values =

     []

denom = 0

prompt = 'Enter resistance of resistor_1: '

r_values = 100

denom = 0.0100

r_total = 100

ans = "Total Resistance in series is: 100.000000"
```

Questions

1. **What is MATLAB?** - Matlab is a programming language specifically made for numerical computation and analysis. Its centered around performing matrix operations and allows for matrix computation to be done in a streamlined way. It also has a lot of built in visualization and plotting tools.

2. **Is MATLAB Compiled or Interpreted**? - Matlab is an interpreted language. This is evident by the fact that one may enter commands and have them executed on the fly.
3. **Is MATLAB case sensitive?** - yes, consider the following example:

```
a = 40
A = 123

disp(a) % 40
disp(A) % 123
```

4. **Advantages of MATLAB over c,C++,python** - Compared to all of these languages. MATLAB fills the niche of allowing for programmers to write code in a way that is much closer to the standard mathematical notation. This language is optimized for mathematics.
5. **Is MATLAB Dynamically or Statically Typed?** - MATLAB is dynamically typed. This is evident from the fact that we dont need to explicitely give types to variables and variables may change their types throughout program execution
6. **The Colon Operator** - The colon operator is used to denote the start, increment and end of a vector

```
vector = 2:2:10 % vector = [2, 4, 6, 8]
```

7. **What is Simulink?** - Simulink lets us create complex models using block diagrams. It allows a visual way to describe systems with a clickable, drag and drop UI