

Name (Last, First):
Kunjangada Anun
Sushma

Student ID:
002473132

Assignment 2

NEU_COE_INFO6105_Fall2024

Instructions:

1. For answering **programming questions**, please use Adobe Acrobat to edit the pdf file in two steps [See Appendix: Example Question and Answer]:
 - a. Copy and paste your R or python code as text in the box provided (so that your teaching team can run your code);
 - b. Screenshot your R or python console outputs, save them as a .PNG image file, and paste/insert them in the box provided.
 - c. Show all work - credit will not be given for code without showing the code in action by including the screenshot of R or python console outputs.
2. To answer **non-programming questions**, please type or handwrite your final answers clearly in the boxes. Show all work - credit will not be given for numerical solutions that appear without explanation in the space above the boxes. **You're encouraged to use R or python to graph/plot the data and produce numerical summaries; please append your code and screenshot of the outputs at the end of your pdf submission.**
3. [Total 129 pts = 9 + 21 + 48 + 39 pts + 12 Extra Credit pts]

Grading Rubric

Each question is worth 3 points and will be graded as follows:

3 points: Correct answer with work shown

2 points: Incorrect answer but attempt shows some understanding (work shown)

1 point: Incorrect answer but an attempt was made (work shown), or **correct answer without explanation (work not shown)**

0 points: Left blank or made little to no effort/work not shown

Reflective Journal [3 pts]

(Copy and paste the link to your live Google doc in the box below)

<https://docs.google.com/document/d/1ptEhnYHniNt1yxDPcvXK7LpJaPGzi80BCSGZZhom7Y/edit?usp=sharing>

Part I. Categorical Data [9 pts = 3 x 3 pts]

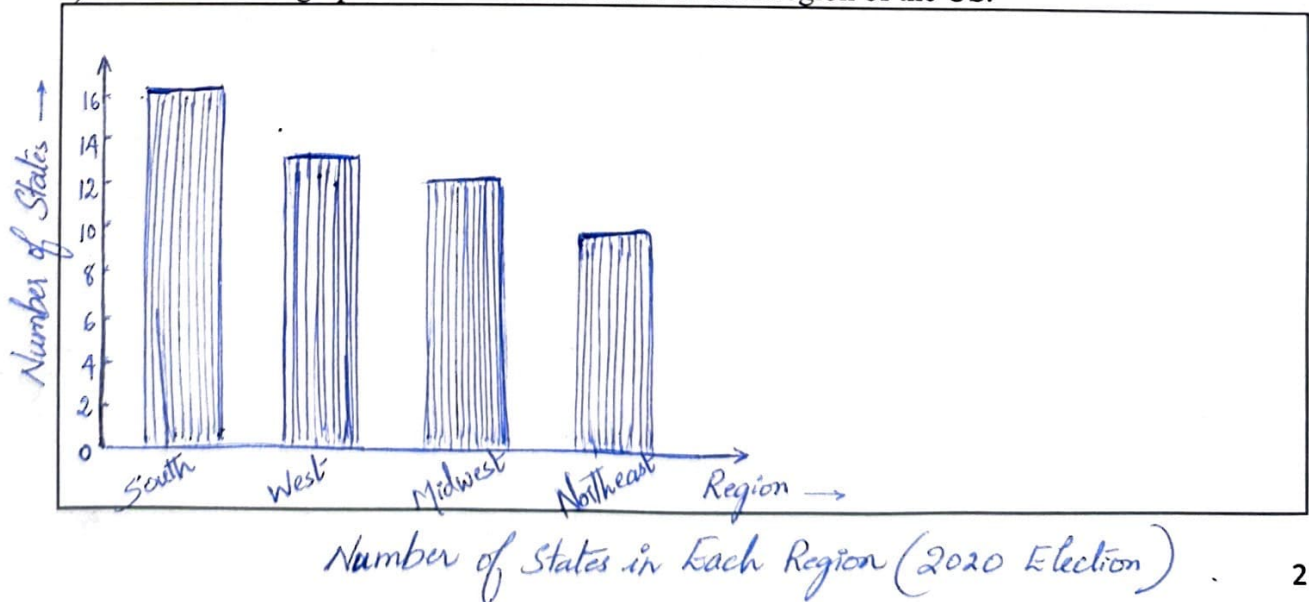
1) Below is a list of unorganized data from the 2020 election. This gives the state, their geographical location, and the voting majority. [The data is accessible via a .csv file saved under the course shared folder "Course Contents".]

State	Location	Electoral College Vote in 2020
AL	South ✓	Republican ✓
AK	West ✓	Republican ✓
AZ	West ✓	Democrat
AR	South ✓	Republican ✓
CA	West ✓	Democrat
CO	West ✓	Democrat
CT	Northeast ✓	Democrat
DE	South ✓	Democrat
FL	South ✓	Republican ✓
GA	South ✓	Democrat
HI	West ✓	Democrat
ID	West ✓	Republican ✓
IL	Midwest	Democrat
IN	Midwest	Republican ✓
IA	Midwest	Republican ✓
KS	Midwest	Republican ✓
KY	South ✓	Republican ✓
LA	South ✓	Republican ✓
ME	Northeast ✓	Democrat*
MD	South ✓	Democrat
MA	Northeast ✓	Democrat
MI	Midwest	Democrat
MN	Midwest	Democrat
MS	South ✓	Republican ✓

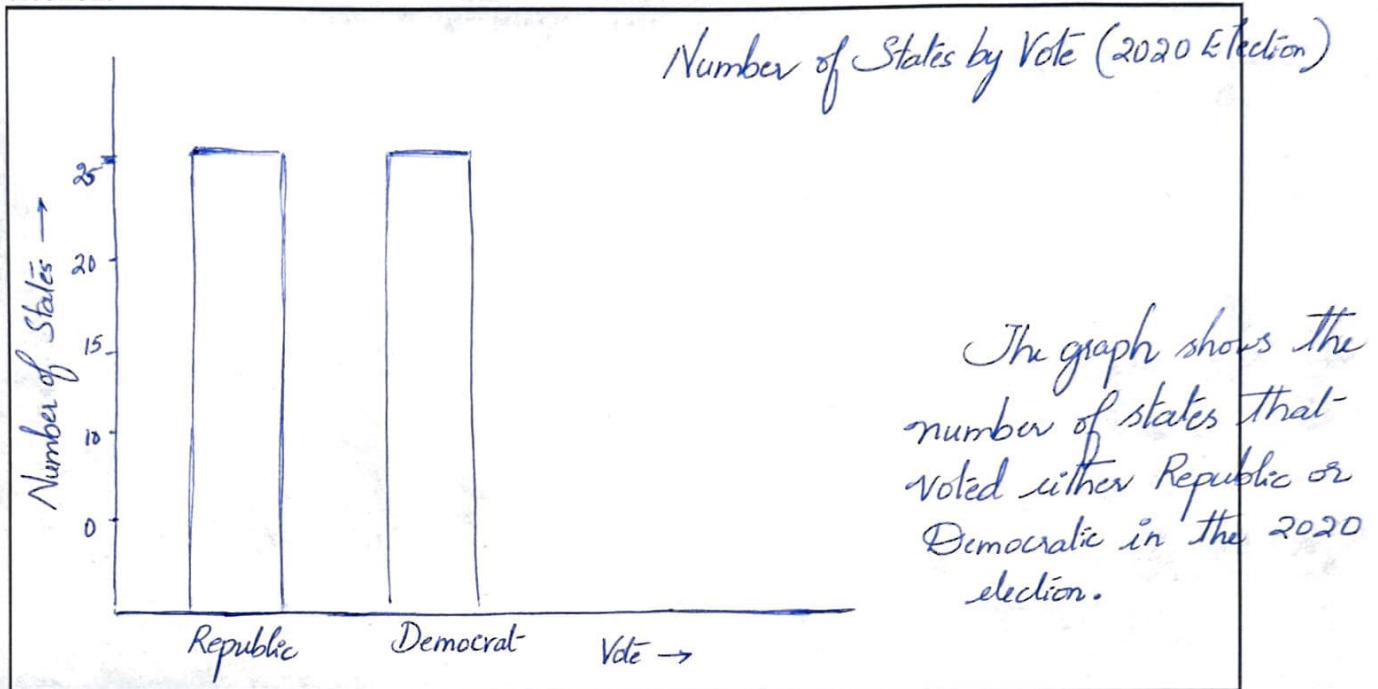
MO	Midwest	Republican ✓
MT	West ✓	Republican ✓
NE	Midwest	Republican* ✓
NV	West ✓	Democrat
NH	Northeast ✓	Democrat
NJ	Northeast ✓	Democrat
NM	West ✓	Democrat
NY	Northeast ✓	Democrat
NC	South ✓	Republican ✓
ND	Midwest	Republican ✓
OH	Midwest	Republican ✓
OK	South ✓	Republican ✓
OR	West ✓	Democrat
PA	Northeast ✓	Democrat
RI	Northeast ✓	Democrat
SC	South ✓	Republican ✓
SD	Midwest	Republican ✓
TN	South ✓	Republican ✓
TX	South ✓	Republican ✓
UT	West ✓	Republican ✓
VT	Northeast ✓	Democrat
VA	South ✓	Democrat
WA	West ✓	Democrat
WV	South ✓	Republican ✓
WI	Midwest	Democrat
WY	West ✓	Republican ✓

*Split Votes

a) Construct a bar graph of the amount of states in each region of the US.



b) Construct a bar graph of the amount of states who voted either republican or democrat in the 2020 election.



c) Create a two way table showing the distribution of region and political party.

Region	Democrat	Republic
Midwest	4	8
Northeast	9	0
South	4	12
West	8	5

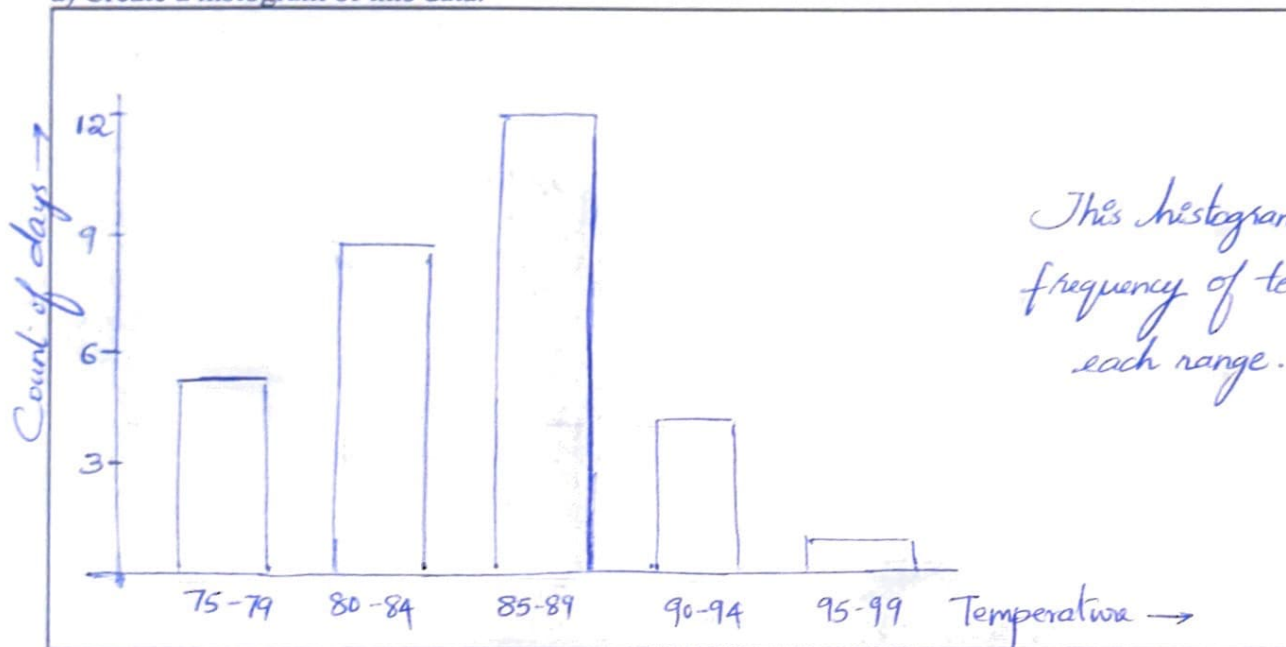
2-way table presents distribution of states by region and voting majority

Part II. Quantitative Data ([21 pts = 7 x 3 pts])

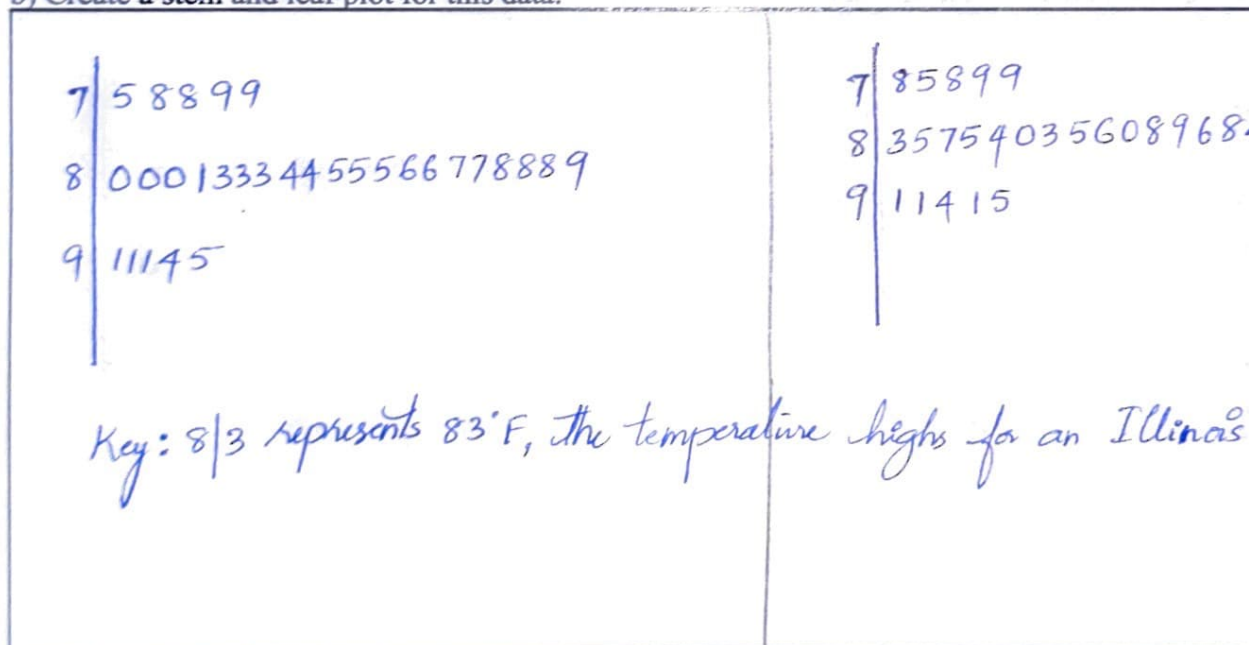
1) The following are the temperature highs for an Illinois town in August 2022.

78 75 83 85 87 91 78 91 85 84 80 79 83 85 86
80 88 94 89 86 88 84 81 85 83 80 79 87 88 91
95

a) Create a histogram of this data.



b) Create a stem and leaf plot for this data.



c) Which graph do you think displays the data the best? Why?

Both histogram and the stem-leaf plot have their merits, but I believe stem & leaf plot displays the data set better.

- Stem and leaf plot preserves the exact values of data point, allowing us to see more data points.
- The stem-and-leaf plot makes it easier to see the clustering of temperatures.
- The stem-and-leaf plot makes it easier to identify outliers.
- With stem-and-leaf plot it's easier to quickly estimate the median and mode, and get the range of the data.

2) 30 people were asked how many miles, to the nearest mile, they commute to work each day:

2 5 7 3 2 10 18 15 20 7 10 18 5 12 13
12 4 5 10 7 10 12 7 3 4 11 13 14 5 6

a) Create a cumulative relative frequency graph (include your table!)

Miles	Frequency	Cummulative Frequency	Relative Frequency	Cummulative Relative Frequency
2	2	2	$\frac{2}{30} = 0.067$	0.067
3	2	4	$\frac{2}{30} = 0.067$	$0.067 + 0.067 = 0.133$
4	2	6	$\frac{2}{30} = 0.067$	$0.133 + 0.067 = 0.200$
5	4	10	$\frac{4}{30} = 0.133$	$0.200 + 0.133 = 0.333$
6	1	11	$\frac{1}{30} = 0.033$	$0.333 + 0.033 = 0.367$
7	4	15	$\frac{4}{30} = 0.133$	$0.367 + 0.133 = 0.500$
10	4	19	$\frac{4}{30} = 0.133$	$0.500 + 0.133 = 0.633$
11	1	20	$\frac{1}{30} = 0.033$	$0.633 + 0.033 = 0.667$
12	3	23	$\frac{3}{30} = 0.100$	$0.667 + 0.100 = 0.767$
13	2	25	$\frac{2}{30} = 0.067$	$0.767 + 0.067 = 0.833$
14	1	26	$\frac{1}{30} = 0.033$	$0.833 + 0.033 = 0.867$
15	1	27	$\frac{1}{30} = 0.033$	$0.867 + 0.033 = 0.900$
18	2	29	$\frac{2}{30} = 0.067$	$0.900 + 0.067 = 0.967$
20	1	30	$\frac{1}{30} = 0.033$	$0.967 + 0.033 = 1.000$

b) What percent of people traveled less than 12 miles?

Answer:

The cumulative relative frequency for 11 miles which is just before 12 miles is 0.667%, i.e., 66.7% of people traveled less than 12 miles.

c) What percent of people traveled over 15 miles?

Answer:

The cumulative relative frequency for 15 miles is 0.900 i.e., 90% of people traveled 15 miles or less.

$\therefore 100\% - 90\% = 10\%$ of people traveled over 15 miles

d) What percent of people traveled between 10 and 15 miles?

Answer:

Cumulative relative frequency for 15 miles = 0.900 i.e., 90%.

Cumulative relative frequency for 10 miles = 0.633 i.e., 63.3%.

$\therefore 90\% - 63.3\% = 26.7\%$ of people traveled between 10 and 15 miles.

Part III. Summary Statistics and Boxplots [48 pts = 16 x 3 pts]

1) Below is a list of calories and cholesterol amounts in 4 randomly selected menu items from 4 different fast food companies. [The data is accessible via a .csv file saved under the course shared folder "Course Contents" as "calories_and_cholesterol_amounts.csv"]

Company	Menu Item	Calories	Cholesterol (mg)
McDonald's	Bacon, Egg, & Cheese Biscuit	460	215
McDonald's	Big Mac®	590	85
McDonald's	Filet-O-Fish®	390	30
McDonald's	Cheeseburger	300	40
Burger King	Whopper® Sandwich with Cheese	740	115
Burger King	Cheeseburger	280	45
Burger King	Crispy Chicken Sandwich	670	60
Burger King	Bacon, Egg & Cheese Biscuit	400	170
Wendy's	Baconator®	960	155
Wendy's	Bacon Double Stack®	440	65
Wendy's	Classic Chicken Sandwich	490	75
Wendy's	Sausage, Egg, and Cheese Biscuit	580	285
Chick-fil-A	Chicken Biscuit	460	45
Chick-fil-A	Bacon, Egg, and Cheese Biscuit	420	180
Chick-fil-A	Grilled Chicken Sandwich	390	75
Chick-fil-A	Chicken Nuggets (8 count)	250	80

a) Using R or python, find the following summary statistics for CALORIES for each company.

$$\text{Mean} = \frac{\sum \text{values}}{\text{Number of values}}$$

Answer:

	Mean	Min	Q1	Med	Q3	Max	Std. Dev
McDonald's	435	300	345	425	525	590	122.34
Burger King	522.5	280	340	535	705	740	218.23
Wendy's	617.5	440	465	535	770	960	235.57
Chick-fil-A	380	250	355	405	440	460	91.29

Which company has the greatest calorie variability in the distribution?

Answer:

Wendy's has the greatest calorie variability, with standard deviation of 235.57 calories.

b) There are many methods for determining outliers. Two methods frequently used are:

Rule #1: An outlier is a value greater than $1.5 \times \text{IQR}$ above the third quartile or more than $1.5 \times \text{IQR}$ below the first quartile.

Rule #2: An outlier is a value located 2 or more standard deviations above, or below, the mean.

Using rule #1, are there any outliers in the Wendy's distribution? Show your work.

Answer:

$\text{IQR} = Q3 - Q1$ <p>For Wendy's:- $Q1 = 465$ $Q3 = 770$ $\text{IQR} = 305$</p> <p>Lower bound for outliers = $465 - 1.5 \times 305 = 465 - 457.5 = 7.5$</p> <p>Upper bound for outliers = $770 + 1.5 \times 305 = 770 + 457.5 = 1227.5$</p>	<p>The minimum & maximum value is, 440 & 960 both fall within these bounds.</p> <p>\therefore No outliers.</p>
--	---

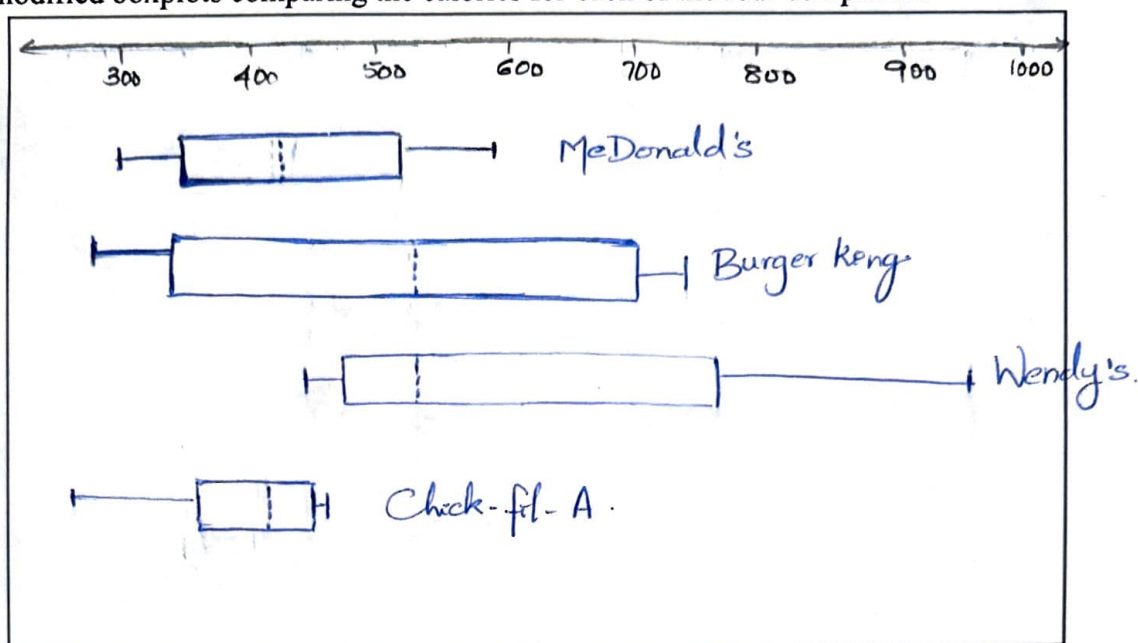
Using rule #2, are there any outliers in the Wendy's distribution? Show your work.

Answer:

<p>Mean = 617.5</p> <p>Standard Deviation = 235.57</p> <p>Lower bound = $617.5 - 2 \times 235.57 = 146.36$</p> <p>Upper bound = $617.5 + 2 \times 235.57 = 1088.64$</p> <p>All Wendy's value fall within 146.36 - 1088.64 range, so there is no outlier.</p>
--

c) Draw four modified boxplots comparing the calories for each of the four companies.

Answer:



d) In your town, McDonald's and Burger King are on the Northside and Wendy's and Chick-fil-A are on the Southside. Using R or python, find the following summary statistics for CHOLESTEROL for the Northside and Southside fast food restaurants. (6 pts = 2 x 3 pts)

	Mean	Min	Q1	Med	Q3	Max	Std. Dev
Northside	95	30	42.5	72.5	142.5	215	67.08
Southside	120	45	70	77.5	167.5	285	81.37

Which region has the greatest cholesterol variability in the distribution?

Answer:

From the table Southside has higher standard deviation (81.37 mg/dl) it has the highest ^{cholesterol} variability in the distribution.

e) Using rule #1, are there any outliers in the Southside's distribution? Show your work.

Answer:

For Southside: $Q1 = 70$, $Q3 = 167.5$.

$$IQR = Q3 - Q1 = 167.5 - 70 = 97.5$$

$$\text{Lower bound} : 70.0 - 1.5 \times 97.5 = -76.25$$

$$\text{Upper bound} : 167.5 + 1.5 \times 97.5 = 313.75$$

There are no outliers as all the values of Southside fall between -76.25 and 313.75.

Using rule #2, are there any outliers in the Southside's distribution? Show your work.

Answer:

Mean = 120
 Standard deviation = 81.37
 Lower bound = $120 - 2 \times 81.37 = -42.74$
 Upper bound = $120.0 + 2 \times 81.37 = 282.74$
 Standard deviation method identifies 285mg as an outlier as it doesn't fall inside the boundary range.

f) Remove the value of 285mg from the Southside cholesterol's data set. Use R or python to find the following values again:

	Mean	Min	Q1	Med	Q3	Max	Std. Dev
Southside	96.43	45	70	75	117.5	180	48.99 = 49

What values changed the most? What values changed the least?

Answer
(Most):

* The maximum value changed by 105mg (from 285 to 180)
 Q3 changed by 50mg (from 167.5 to 117.5), Standard Deviation changed by 32.37mg (81.37 to 49) & mean changed by 23.57mg (120 to 96.43).

Answer
(Least):

Minimum there was no change. & there was no change in Q1 value.
 Median minimal change 2.5mg (77.5 to 75).

Part IV. Statistical Programming (51 pts) [* Extra Credits] [Instructions in R]

1. **Sequences.** Generate the following sequences using `rep()`, `seq()` and arithmetic:

- (a) 1, 3, 5, 7, ..., 21.
- (b) 1, 10, 100, ..., 10^9 .
- (c) 0, 1, 2, 3, 0, ..., 3, 0, 1, 2, 3 [with each entry appearing 6 times]
- (d) 0, 0, 0, 1, 1, 1, 2, ..., 4, 4, 4.
- (e)* 50, 47, 44, ..., 14, 11.
- (f)* 1, 2, 5, 10, 20, 50, 100, ..., 5×10^4 .

Can any of your answers be simplified using recycling?

2. **Arithmetic.** Create a vector containing each of the following sequences:

- (a) $\cos\left(\frac{\pi n}{3}\right)$, for $n = 0, \dots, 10$.
- (b) 1, 9, 98, 997, ..., 999994.
- (c) $e^n - 3n$, for $n = 0, \dots, 10$.
- (d)* $3n \bmod 7$, for $n = 0, \dots, 10$.

Let

$$S_n = \sum_{i=1}^n \frac{(-1)^{i+1}}{2i-1} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots + \frac{(-1)^{n+1}}{2n-1}.$$

You will recall that $\lim_n S_n = \pi/4$.

- (e) Evaluate $4S_{10}$, $4S_{100}$ and $4S_{1000}$. [Hint: use the `sum()` function.]
- (f) Create a vector with entries $S_i - \frac{\pi}{4}$, for $i = 1, \dots, 1000$. [Hint: try creating the vector with entries S_i first; the function `cumsum()` may be useful.]

3. **Subsetting**

Create a vector `x` of normal random variables as follows:

```
> set.seed(123)
> x <- rnorm(100)
```

The `set.seed()` fixes the random number generator so that we all obtain the same `x`; changing the argument 123 to something else will give different results. This is useful for replication.

Give commands to select a vector containing:

- (a) the 25th, 50th and 75th elements;
- (b) the first 25 elements;
- (c) all elements except those from the 31st to the 40th.

Recall the logical operators `|`, `&` and `!`. Give commands to select:

- (d) all values larger than 1.5 (how many are there?);
- (e) what about the entries that are either > 1.5 or < -1 ?

Answer: Copy and paste your R or python code in the box below (not an image but the text).

```
#1. Sequences
import numpy as np

# (a) 1, 3, 5, 7, ..., 21
a = np.arange(1, 22, 2)
print("(a):", a)

# (b) 1, 10, 100, ..., 10^9
b = 10 ** np.arange(0, 10)
print("(b):", b)

# (c) 0, 1, 2, 3, 0, ..., 3, 0, 1, 2, 3 [with each entry appearing 6 times]
c = np.tile(np.arange(4), 6)
print("(c):", c)

# (d) 0, 0, 0, 1, 1, 1, 2, ..., 4, 4, 4
d = np.repeat(np.arange(5), 3)
print("(d):", d)

# (e) 50, 47, 44, ..., 14, 11
e = np.arange(50, 10, -3)
print("(e):", e)

# (f) 1, 2, 5, 10, 20, 50, 100, ..., 5 x 10^4
f = np.array([1, 2, 5])
while f[-1] < 5e4:
    f = np.append(f, f[-1] * 2)
print("(f):", f)

# Can any of your answers be simplified using recycling?
# Yes, (c) and (d) use recycling with np.tile and np.repeat respectively.

(a): [ 1  3  5  7  9 11 13 15 17 19 21]
(b): [ 1      10     100    1000   10000  100000]
      1000000 10000000 100000000 1000000000]
(c): [0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3]
(d): [0 0 0 1 1 1 2 2 2 3 3 3 4 4 4]
(e): [50 47 44 41 38 35 32 29 26 23 20 17 14 11]
(f): [ 1    2    5   10   20   40   80  160  320  640 1280 2560
      5120 10240 20480 40960 81920]
```

Answer: Copy and paste your R or phthon code in the box below (not an image but the text).

```
#2. Arithmetic section

import numpy as np

# (a)  $\cos(n\pi/3)$ , for  $n = 0, \dots, 10$ 
a = np.cos(np.arange(11) * np.pi / 3)
print("(a):", a)

# (b) 1, 9, 98, 997, ..., 999994
b = np.array([int('9' * i + '1' * (6-i)) for i in range(6)])
print("(b):", b)

# (c)  $e^n - 3n$ , for  $n = 0, \dots, 10$ 
c = np.exp(np.arange(11)) - 3 * np.arange(11)
print("(c):", c)

# (d)  $3n \bmod 7$ , for  $n = 0, \dots, 10$ 
d = (3 * np.arange(11)) % 7
print("(d):", d)

# (e) Evaluate 4S_10, 4S_100 and 4S_1000
def S(n):
    return np.sum((-1)**(np.arange(1, n+1) + 1) / (2 * np.arange(1, n+1) - 1))

print("(e):")
print("4S_10 =", 4 * S(10))
print("4S_100 =", 4 * S(100))
print("4S_1000 =", 4 * S(1000))

# (f) Create a vector with entries  $S_i - \pi/4$ , for  $i = 1, \dots, 1000$ 
f = np.cumsum((-1)**(np.arange(1, 1001) + 1) / (2 * np.arange(1, 1001) - 1)) - np.pi/4
np.set_printoptions(threshold=np.inf)
print("(f):", f)
```

```
(a): [ 1.  0.5 -0.5 -1. -0.5  0.5  1.  0.5 -0.5 -1. -0.5]
(b): [111111 911111 991111 999111 999911 999991]
(c): [ 1.00000000e+00 -2.01718172e-01  1.38905610e+00  1.10855369e+01
  4.25981500e+01  1.33413159e+02  3.85428793e+02  1.07563316e+03
  2.95695799e+03  8.07608393e+03  2.19964658e+04]
(d): [0 3 6 2 5 1 4 0 3 6 2]
(e):
4S_10 = 3.0418396189294024
4S_100 = 3.131592903558552
4S_1000 = 3.140592653839792
(f): [ 0.21460184 -0.1187315  0.0812685 -0.06158864  0.04952247 -0.04138662
  0.03553646 -0.03113021  0.02769332 -0.02493826  0.02268079 -0.02079747
  0.01920253 -0.01783451  0.01664825 -0.01560981  0.01469322 -0.01387821
  0.01314881 -0.01249221  0.01189803 -0.01135778  0.01086444 -0.01041216
  0.00999601 -0.00961184  0.00925609 -0.00892573  0.00861813 -0.00833102
  0.00806242 -0.00781059  0.00757402 -0.00735135  0.00714141 -0.00694311
  0.00675552 -0.00657781  0.0064092 -0.00624902  0.00609665 -0.00595154
  0.00581317 -0.00568108  0.00555487 -0.00543414  0.00531855 -0.00520777
  0.00508793 -0.00497292  0.00486441 -0.00476181  0.00466461 -0.00457392
  0.00448421 -0.00440371  0.00433281 -0.00426991  0.00421351 -0.00416331
  0.00411521 -0.00407351  0.00403781 -0.00400281  0.00396841 -0.00393441
  0.00390081 -0.00386881  0.00383761 -0.00380761  0.00377881 -0.00375001
  0.00372241 -0.00369441  0.00366801 -0.00364241  0.00361761 -0.00359281
  0.00356841 -0.00354441  0.00352121 -0.00349881  0.00347761 -0.00345681
  0.00343761 -0.00341841  0.00339961 -0.00338161  0.00336401 -0.00334721
  0.00333121 -0.00331521  0.00329961 -0.00328441  0.00326441 -0.00324561
  0.00322721 -0.00320961  0.00319321 -0.00317681  0.00316001 -0.00314401
  0.00312801 -0.00311281  0.00309841 -0.00308401  0.00306521 -0.00305121
  0.00303321 -0.00302001  0.00300801 -0.00299441  0.00298121 -0.00296841
  0.00295601 -0.00294401  0.00293281 -0.00292121  0.00291001 -0.00289881
  0.00288801 -0.00287721  0.00286641 -0.00285601  0.00284521 -0.00283521
  0.00282481 -0.00281521  0.00280521 -0.00279601  0.00278641 -0.00277761
  0.00276841 -0.00276001  0.00275121 -0.00274321  0.00273561 -0.00272801
  0.00272081 -0.00271361  0.00270681 -0.00270001  0.00269321 -0.00268681
  0.00267961 -0.00267361  0.00266721 -0.00266081  0.00265441 -0.00264841
  0.00264241 -0.00263681  0.00263121 -0.00262601  0.00262081 -0.00261601
  0.00261121 -0.00260681  0.00260281 -0.00259881  0.00259521 -0.00259161
  0.00258801 -0.00258481  0.00258161 -0.00257841  0.00257521 -0.00257201
  0.00256881 -0.00256561  0.00256281 -0.00255961  0.00255681 -0.00255361
  0.00255081 -0.00254761  0.00254481 -0.00254201  0.00253921 -0.00253641
  0.00253361 -0.00253081  0.00252801 -0.00252521  0.00252241 -0.00251961
  0.00251681 -0.00251401  0.00251121 -0.00250841  0.00250561 -0.00250281
  0.00250001 -0.00249721  0.00249441 -0.00249161  0.00248881 -0.00248601
  0.00248321 -0.00248041  0.00247761 -0.00247481  0.00247201 -0.00246921
  0.00246641 -0.00246361  0.00246081 -0.00245801  0.00245521 -0.00245241
  0.00244961 -0.00244681  0.00244401 -0.00244121  0.00243841 -0.00243561
  0.00243281 -0.00243001  0.00242721 -0.00242441  0.00242081 -0.00241801
  0.00241521 -0.00241241  0.00240961 -0.00240681  0.00240401 -0.00240121
  0.00239841 -0.00239561  0.00239281 -0.00239001  0.00238721 -0.00238441
  0.00238161 -0.00237881  0.00237601 -0.00237321  0.00237041 -0.00236761
  0.00236481 -0.00236201  0.00235921 -0.00235641  0.00235361 -0.00235081
  0.00234801 -0.00234521  0.00234241 -0.00233961  0.00233681 -0.00233401
  0.00233121 -0.00232841  0.00232561 -0.00232281  0.00232001 -0.00231721
  0.00231441 -0.00231161  0.00230881 -0.00230601  0.00230321 -0.00230041
  0.00229761 -0.00229481  0.00229201 -0.00228921  0.00228641 -0.00228361
  0.00228081 -0.00227801  0.00227521 -0.00227241  0.00226961 -0.00226681
  0.00226401 -0.00226121  0.00225841 -0.00225561  0.00225281 -0.00225001
  0.00224721 -0.00224441  0.00224161 -0.00223881  0.00223601 -0.00223321
  0.00223041 -0.00222761  0.00222481 -0.00222201  0.00221921 -0.00221641
  0.00221361 -0.00221081  0.00220801 -0.00220521  0.00220241 -0.00219961
  0.00219681 -0.00219401  0.00219121 -0.00218841  0.00218561 -0.00218281
  0.00218001 -0.00217721  0.00217441 -0.00217161  0.00216881 -0.00216601
  0.00216321 -0.00216041  0.00215761 -0.00215481  0.00215201 -0.00214921
  0.00214641 -0.00214361  0.00214081 -0.00213801  0.00213521 -0.00213241
  0.00212961 -0.00212681  0.00212401 -0.00212121  0.00211841 -0.00211561
  0.00211281 -0.00211001  0.00210801 -0.00210521  0.00210241 -0.00209961
  0.00209681 -0.00209401  0.00209121 -0.00208841  0.00208561 -0.00208281
  0.00208001 -0.00207721  0.00207441 -0.00207161  0.00206881 -0.00206601
  0.00206321 -0.00206041  0.00205761 -0.00205481  0.00205201 -0.00204921
  0.00204641 -0.00204361  0.00204081 -0.00203801  0.00203521 -0.00203241
  0.00203121 -0.00202841  0.00202561 -0.00202281  0.00202001 -0.00201721
  0.00201441 -0.00201161  0.00200881 -0.00200601  0.00200321 -0.00200041
  0.00199761 -0.00199481  0.00199201 -0.00198921  0.00198641 -0.00198361
  0.00198081 -0.00197801  0.00197521 -0.00197241  0.00196961 -0.00196681
  0.00196401 -0.00196121  0.00195761 -0.00195481  0.00195201 -0.00194921
  0.00194641 -0.00194361  0.00194081 -0.00193801  0.00193521 -0.00193241
  0.00193121 -0.00192841  0.00192561 -0.00192281  0.00192001 -0.00191721
  0.00191441 -0.00191161  0.00190881 -0.00190601  0.00190321 -0.00190041
  0.00189761 -0.00189481  0.00189201 -0.00188921  0.00188641 -0.00188361
  0.00188081 -0.00187801  0.00187521 -0.00187241  0.00186961 -0.00186681
  0.00186401 -0.00186121  0.00185761 -0.00185481  0.00185201 -0.00184921
  0.00184641 -0.00184361  0.00184081 -0.00183801  0.00183521 -0.00183241
  0.00183121 -0.00182841  0.00182561 -0.00182281  0.00182001 -0.00181721
  0.00181441 -0.00181161  0.00180881 -0.00180601  0.00180321 -0.00180041
  0.00179761 -0.00179481  0.00179201 -0.00178921  0.00178641 -0.00178361
  0.00178081 -0.00177801  0.00177521 -0.00177241  0.00176961 -0.00176681
  0.00176401 -0.00176121  0.00175761 -0.00175481  0.00175201 -0.00174921
  0.00174641 -0.00174361  0.00174081 -0.00173801  0.00173521 -0.00173241
  0.00173121 -0.00172841  0.00172561 -0.00172281  0.00172001 -0.00171721
  0.00171441 -0.00171161  0.00170881 -0.00170601  0.00170321 -0.00170041
  0.00169761 -0.00169481  0.00169201 -0.00168921  0.00168641 -0.00168361
  0.00168081 -0.00167801  0.00167521 -0.00167241  0.00166961 -0.00166681
  0.00166401 -0.00166121  0.00165761 -0.00165481  0.00165201 -0.00164921
  0.00164641 -0.00164361  0.00164081 -0.00163801  0.00163521 -0.00163241
  0.00163121 -0.00162841  0.00162561 -0.00162281  0.00162001 -0.00161721
  0.00161441 -0.00161161  0.00160881 -0.00160601  0.00160321 -0.00160041
  0.00159761 -0.00159481  0.00159201 -0.00158921  0.00158641 -0.00158361
  0.00158081 -0.00157801  0.00157521 -0.00157241  0.00156961 -0.00156681
  0.00156401 -0.00156121  0.00155761 -0.00155481  0.00155201 -0.00154921
  0.00154641 -0.00154361  0.00154081 -0.00153801  0.00153521 -0.00153241
  0.00153121 -0.00152841  0.00152561 -0.00152281  0.00152001 -0.00151721
  0.00151441 -0.00151161  0.00150881 -0.00150601  0.00150321 -0.00150041
  0.00149761 -0.00149481  0.00149201 -0.00148921  0.00148641 -0.00148361
  0.00148081 -0.00147801  0.00147521 -0.00147241  0.00146961 -0.00146681
  0.00146401 -0.00146121  0.00145761 -0.00145481  0.00145201 -0.00144921
  0.00144641 -0.00144361  0.00144081 -0.00143801  0.00143521 -0.00143241
  0.00143121 -0.00142841  0.00142561 -0.00142281  0.00142001 -0.00141721
  0.00141441 -0.00141161  0.00140881 -0.00140601  0.00140321 -0.00140041
  0.00139761 -0.00139481  0.00139201 -0.00138921  0.00138641 -0.00138361
  0.00138081 -0.00137801  0.00137521 -0.00137241  0.00136961 -0.00136681
  0.00136401 -0.00136121  0.00135761 -0.00135481  0.00135201 -0.00134921
  0.00134641 -0.00134361  0.00134081 -0.00133801  0.00133521 -0.00133241
  0.00133121 -0.00132841  0.00132561 -0.00132281  0.00132001 -0.00131721
  0.00131441 -0.00131161  0.00130881 -0.00130601  0.00130321 -0.00130041
  0.00129761 -0.00129481  0.00129201 -0.00128921  0.00128641 -0.00128361
  0.00128081 -0.00127801  0.00127521 -0.00127241  0.00126961 -0.00126681
  0.00126401 -0.00126121  0.00125761 -0.00125481  0.00125201 -0.00124921
  0.00124641 -0.00124361  0.00124081 -0.00123801  0.00123521 -0.00123241
  0.00123121 -0.00122841  0.00122561 -0.00122281  0.00122001 -0.00121721
  0.00121441 -0.00121161  0.00120881 -0.00120601  0.00120321 -0.00120041
  0.00119761 -0.00119481  0.00119201 -0.00118921  0.00118641 -0.00118361
  0.00118081 -0.00117801  0.00117521 -0.00117241  0.00116961 -0.00116681
  0.00116401 -0.00116121  0.00115761 -0.00115481  0.00115201 -0.00114921
  0.00114641 -0.00114361  0.00114081 -0.00113801  0.00113521 -0.00113241
  0.00113121 -0.00112841  0.00112561 -0.00112281  0.00112001 -0.00111721
  0.00111441 -0.00111161  0.00110881 -0.00110601  0.00110321 -0.00110041
  0.00109761 -0.00109481  0.00109201 -0.00108921  0.00108641 -0.00108361
  0.00108081 -0.00107801  0.00107521 -0.00107241  0.00106961 -0.00106681
  0.00106401 -0.00106121  0.00105761 -0.00105481  0.00105201 -0.00104921
  0.00104641 -0.00104361  0.00104081 -0.00103801  0.00103521 -0.00103241
  0.00103121 -0.00102841  0.00102561 -0.00102281  0.00102001 -0.00101721
  0.00101441 -0.00101161  0.00100881 -0.00100601  0.00100321 -0.00100041
  0.00099761 -0.00099481  0.00099201 -0.00098921  0.00098641 -0.00098361
  0.00098081 -0.00097801  0.00097521 -0.00097241  0.00096961 -0.00096681
  0.00096401 -0.00096121  0.00095761 -0.00095481  0.00095201 -0.00094921
  0.00094641 -0.00094361  0.00094081 -0.00093801  0.00093521 -0.00093241
  0.00093121 -0.00092841  0.00092561 -0.00092281  0.00092001 -0.00091721
  0.00091441 -0.00091161  0.00090881 -0.00090601  0.00090321 -0.00090041
  0.00089761 -0.00089481  0.00089201 -0.00088921  0.00088641 -0.00088361
  0.00088081 -0.00087801  0.00087521 -0.00087241  0.00086961 -0.00086681
  0.00086401 -0.00086121  0.00085761 -0.00085481  0.00085201 -0.00084921
  0.00084641 -0.00084361  0.00084081 -0.00083801  0.00083521 -0.00083241
  0.00083121 -0.00082841  0.00082561 -0.00082281  0.00082001 -0.00081721
  0.00081441 -0.00081161  0.00080881 -0.00080601  0.00080321 -0.00080041
  0.00079761 -0.00079481  0.00079201 -0.00078921  0.00078641 -0.00078361
  0.00078081 -0.00077801  0.00077521 -0.00077241  0.00076961 -0.00076681
  0.00076401 -0.00076121  0.00075761 -0.00075481  0.00075201 -0.00074921
  0.00074641 -0.00074361  0.00074081 -0.00073801  0.00073521 -0.00073241
  0.00073121 -0.00072841  0.00072561 -0.00072281  0.00072001 -0.00071721
  0.00071441 -0.00071161  0.00070881 -0.00070601  0.00070321 -0.00070041
  0.00069761 -0.00069481  0.00069201 -0.00068921  0.00068641 -0.00068361
  0.00068081 -0.00067801  0.00067521 -0.00067241  0.00066961 -0.00066681
  0.00066401 -0.00066121  0.00065761 -0.00065481  0.00065201 -0.00064921
  0.00064641 -0.00064361  0.00064081 -0.00063801  0.00063521 -0.00063241
  0.00063121 -0.00062841  0.00062561 -0.00062281  0.00062001 -0.00061721
  0.00061441 -0.00061161  0.00060881 -0.00060601  0.00060321 -0.00060041
  0.00059761 -0.00059481  0.00059201 -0.00058921  0.00058641 -0.00058361
  0.00058081 -0.00057801  0.00057521 -0.00057241  0.00056961 -0.00056681
  0.00056401 -0.00056121  0.00055761 -0.00055481  0.00055201 -0.00054921
  0.00054641 -0.00054361  0.00054081 -0.00053801  0.00053521 -0.00053241
  0.00053121 -0.00052841  0.00052561 -0.00052281  0.00052001 -0.00051721
  0.00051441 -0.00051161  0.00050881 -0.00050601  0.00050321 -0.00050041
  0.00049761 -0.00049481  0.00049201 -0.00048921  0.00048641 -0.00048361
  0.00048081 -0.00047801  0.00047521 -0.00047241  0.00046961 -0.00046681
  0.00046401 -0.00046121  0.00045761 -0.00045481  0.00045201 -0.00044921
  0.00044641 -0.00044361  0.00044081 -0.00043801  0.00043521 -0.00043241
  0.00043121 -0.00042841  0.00042561 -0.00042281  0.00042001 -0.00041721
  0.00041441 -0.00041161  0.00040881 -0.00040601  0.00040321 -0.00040041
  0.00039761 -0.00039481  0.00039201 -0.00038921  0.00038641 -0.00038361
  0.00038081 -0.00037801  0.00037521 -0.00037241  0.00036961 -0.00036681
  0.00036401 -0.00036121  0.00035761 -0.00035481  0.00035201 -0.00034921
  0.00034641 -0.00034361  0.00034081 -0.00033801  0.00033521 -0.00033241
  0.00033121 -0.00032841  0.00032561 -0.00032281  0.00032001 -0.00031721
 
```


Answer: Copy and paste your R or python code in the box below (not an image but the text).

```
#3. Subsetting section

import numpy as np

# Set seed for reproducibility
np.random.seed(123)

# Create vector of normal random variables
x = np.random.normal(size=100)

# (a) Select the 25th, 50th and 75th elements
a = x[[24, 49, 74]] # Python uses 0-based indexing
print("(a):", a)

# (b) Select the first 25 elements
b = x[:25]
print("(b): First 25 elements of result:")
print("(b):", b)

# (c) Select all elements except those from the 31st to the 40th
c = np.concatenate((x[:30], x[40:]))
print("(c): Length of result:", len(c))
print("(c):", c)

# (d) Select all values larger than 1.5
d = x[x > 1.5]
print("(d): Number of values larger than 1.5:", len(d))
print("(d) Values:", d)

# (e) Select entries that are either > 1.5 or < -1
e = x[(x > 1.5) | (x < -1)]
print("(e): Number of entries > 1.5 or < -1:", len(e))
print("(e) Values:", e)

(a): [-1.25388067  2.23814334 -2.12310035]
(b): First 25 elements of result:
(b): [-1.0856306  0.99734545  0.2829785 -1.50629471 -0.57860025  1.65143654
-2.42667924 -0.42891263  1.26593626 -0.8667404 -0.67888615 -0.09470897
 1.49138963 -0.638902  -0.44398196 -0.43435128  2.20593008  2.18678609
 1.0040539  0.3861864  0.73736858  1.49073203 -0.93583387  1.17582904
-1.25388067]
(c): Length of result: 90
(c): [-1.0856306  0.99734545  0.2829785 -1.50629471 -0.57860025  1.65143654
-2.42667924 -0.42891263  1.26593626 -0.8667404 -0.67888615 -0.09470897
 1.49138963 -0.638902  -0.44398196 -0.43435128  2.20593008  2.18678609
 1.0040539  0.3861864  0.73736858  1.49073203 -0.93583387  1.17582904
-1.25388067 -0.6377515  0.9071052 -1.4286807 -0.14006872 -0.8617549
-0.80536652 -1.72766949 -0.39089979  0.57380586  0.33858905 -0.01183049
 2.39236527  0.41291216  0.97873601  2.23814334 -1.29408532 -1.03878821
 1.74371223 -0.79806274  0.02968323  1.06931597  0.89070639  1.75488618
 1.49564414  1.06939267 -0.77270871  0.79486267  0.31427199 -1.32626546
 1.41729905  0.80723653  0.04549008 -0.23309206 -1.19830114  0.19952407
 0.46843912 -0.83115498  1.16220405 -1.09720305 -2.12310035  1.03972709
-0.40336604 -0.12602959 -0.83751672 -1.60596276  1.25523737 -0.68886898
 1.66095249  0.80730819 -0.31475815 -1.0859024 -0.73246199 -1.21252313
 2.08711336  0.16444123  1.15020554 -1.26735205  0.18103513  1.17786194
-0.33501076  1.03111446 -1.08456791 -1.36347154  0.37940061 -0.37917643]
(d): Number of values larger than 1.5: 9
(d) Values: [1.65143654 2.20593008 2.18678609 2.39236527 2.23814334 1.74371223
 1.75488618 1.66095249 2.08711336]
(e): Number of entries > 1.5 or < -1: 29
(e) Values: [-1.0856306 -1.50629471  1.65143654 -2.42667924  2.20593008  2.18678609
-1.25388067 -1.4286807 -2.79858911 -1.7715331 -1.72766949  2.39236527
 2.23814334 -1.29408532 -1.03878821  1.74371223  1.75488618 -1.32626546
-1.19830114 -1.09720305 -2.12310035 -1.60596276  1.66095249 -1.0859024
-1.21252313  2.08711336 -1.26735205 -1.08456791 -1.36347154]
```

Appendix: Example Question and Answer for R or python programming questions:

Calculate the sum $\sum_{j=0}^n r^j$, where r has been assigned the value 1.08, and compare with $(1 - r^{n+1})/(1 - r)$, for $n = 10, 20, 30, 40$.

Answer: Copy and paste your R or python code in the box below (not an image but the text).

```
r <- 1.08
n <- c(10, 20, 30, 40)
sum1 <- c()
for(i in n){
  x <- 0:i
  sum1 <- c(sum1, sum(r^x))
}
sum1 # This gives the calculated sums for n = 10, 20, 30, 40.

sum2 <- (1 - r^(n + 1)) / (1 - r)
sum2

sum2 - sum1 # The formula works.
```

Screenshot of your R or python console outputs and paste/insert the image in the box below

```
> r <- 1.08
> n <- c(10, 20, 30, 40)
> sum1 <- c()
> for(i in n){
+   x <- 0:i
+   sum1 <- c(sum1, sum(r^x))
+ }
> sum1 # This gives the calculated sums for n = 10, 20, 30, 40.
[1] 16.64549 50.42292 123.34587 280.78104
> sum2 <- (1 - r^(n + 1)) / (1 - r)
> sum2
[1] 16.64549 50.42292 123.34587 280.78104
> sum2 - sum1 # The formula works.
[1] 0 0 0 0
```

THE END