ASSIGNMENT 2

DSCI 6607 – Programmatic Data Analysis Using Python and R

Name: Sahil Khan
Student ID: 202482066

Submission Date: 26-Oct-2024

Question 1

Recall the bisection method, we leaned in assignment 1. The bisection method can be generalized to deal with the case f(xl)f(xr) > 0 (i.e., the two end points do not have opposite signs), by broadening the bracket. That is, we reduce xl and/or increase xr, and try again. A reasonable choice for broadening the bracket is to double the width of the interval [xl, xr], that is

$$\mathrm{m} \leftarrow (x_l + x_r)/2,$$

$$\mathrm{w} \leftarrow x_r - x_l,$$

$$x_l \leftarrow m - w,$$

$$x_r \leftarrow m + w,$$

a) Incorporate bracket broadening into the bisection method

```
bisectionBroadening <- function(f, xl, xr, tol = 1e-6, max_iter = 50) {
    iter <- 0
    while (iter < max_iter) {
        m <- (xl + xr) / 2
        w <- xr - xl
        if (f(xl) * f(xr) <= 0) {
            break
        }
        xl <- m - w
        xr <- m + w
        iter <- iter + 1
    }
    while (abs(xr - xl) > tol && iter < max_iter) {
        m <- (xl + xr) / 2
        if (f(xl) * f(m) < 0) {
            xr <- m
        } else {
            xl <- m
        }
        iter <- iter + 1
    }
    return(m)
}
```

b) Finding the root of

$$f(x) = (x-1)^3 - 2x^2 + 10 - sin(x)$$

```
f <- function(x) (x - 1)^3 - 2 * x^2 + 10 - sin(x)
root <- bisectionBroadening(f, 1, 2)
print(root)
```

```
## [1] -1.052896
```

Question 2

We plan to test the equality for the means of two samples in Python.

1. Let x and y be two samples of sizes n1 and n2, respectively. To test

   $$H0 : \mu x = \mu y$$

   vs

   $$H1 : \mu x \neq \mu y,$$

   we compute the test statistic

   $$t = \frac{\bar{x} - \bar{y}}{\sqrt{s_p^2(\frac{1}{n_1} + \frac{1}{n_2})}}$$

   where

   $$\bar{x}, \bar{y}$$

   denote the means of x and y samples and

   $$s_p^2 = \frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 1}$$

   where

   $$s_1^2 = \frac{1}{n_1 - 1}\sum_{i=1}^{n_1}(x_i - \bar{x})^2$$

   $$s_2^2 = \frac{1}{n_2 - 1}\sum_{i=1}^{n_2}(y_i - \bar{y})^2$$

2. Python function which takes x and y as two lists and returns the observed test statistic.

```
import numpy as np

def tStatistic(x, y):

  n1 = len(x)
  n2 = len(y)
  xBar = np.mean(x)
  yBar = np.mean(y)
  s1 = np.var(x, ddof=1)
  s2 = np.var(y, ddof=1)
  sp = ((n1-1)*s1 + (n2-1)*s2) / (n1 + n2 - 2)
  t = (xBar - yBar) / np.sqrt(sp * (1/n1 + 1/n2))

  return t
```

3. Generate 50 observations form normal distribution with mean =1 and standard deviation=2 and assign the data to list x. Generate 57 observations from uniform distribution between -2 and 2 and assign data to list y.

```
# Generating sample X from Normal Distribution
x = np.random.normal(loc=1, scale=2, size=50)

# Generating sample Y from Uniform Distribution
y = np.random.uniform(low=-2, high=2, size=57)
```

4. Applying our function and computing the test statistic for the two samples from part 3.

```
t = tStatistic(x, y)

print("The observed t-statistic is:", t)
```

```
## The observed t-statistic is: 6.361803848002604
```

Question 3

Consider the python list

$$x = [3, 8, 13, 18, 108, 25, 23, 17, 203, 11, 23]$$

Write a python function where takes the list and uses only list comprehension and returns the odd values smaller than 23.

```
def oddValuesUnder23(lst):
  return [x for x in lst if x % 2 != 0 and x < 23]

x = [3, 8, 13, 18, 108, 25, 23, 17, 203, 11, 23]

print("Odd values smaller than 23:", oddValuesUnder23(x))
```

```
## Odd values smaller than 23: [3, 13, 17, 11]
```

Question 4

1. Find the distribution of statistic

$$\sum_{i=1}^{10} X_i$$

Given that$X_i \sim N(\mu, \sigma^2)$ for i=1,2,...,10 where μ and σ^2 are the mean and variance of the population,

we want to find the distribution of statistics $\sum_{i=1}^{10} X_i$

Let:

$$S = \sum_{i=1}^{10} X_i$$

Since each Xi is normally distributed, and the sum of independent normal random variables is also normally distributed, S will be normally distributed.

Mean of the Sum: The mean of S, E(S) is the sum of the means of Xi. Variance of the Sum: The variance of S, Var(S), is the sum of the variances of Xi.

$$E(S) = E(\sum_{i=1}^{10} X_i) = \sum_{i=1}^{10} E(X_i) = \sum_{i=1}^{10} \mu = 10\mu$$

$$Var(S) = Var(\sum_{i=1}^{10} X_i) = \sum_{i=1}^{10} Var(X_i) = \sum_{i=1}^{10} \sigma^2 = 10\sigma^2$$

Therefore,

$$S = \sum_{i=1}^{10} X_i \sim N(10\mu, 10\sigma^2)$$

2. We would like to test the finding of part (1) numerically. To do that first generate a sample of size 10 from the Normal distribution with parameters

$$\mu = 23 \text{ and } \sigma^2 = 3.6$$

and then compute the sum of the generated observations.

```
def genNormalDist(sims, size=10, mu=23, sigSq=3.6):

    sampleSums = []

    for i in range(sims):
      # Generating a sample of size=10 from Normal Distribution
        sample = np.random.normal(loc=mu, scale=np.sqrt(sigSq), size=size)

        sampleSums.append(np.sum(sample))

    return sampleSums

print("Sample Sum:", genNormalDist(sims=1))
```

```
## Sample Sum: [231.0792959545624]
```

3. Use python and simulate 10000 times part (2) and compute the sum of the generated samples of size 10.

```
simulations = 10000
sampleSumList = genNormalDist(sims=simulations)
print("Sample Sums after 10000 simulations (first 10 obs):")
```

```
## Sample Sums after 10000 simulations (first 10 obs):
```

```
print("\n".join(str(sum_value) for sum_value in sampleSumList[0:10]))
```

```
## 234.05418453686315
## 225.5626148716646
## 227.20356214438925
## 219.31547690587053
## 245.92156111087837
## 225.9405647194501
## 225.46568455840168
## 237.22150842586325
## 237.76401288576358
## 228.53941440296614
```

4. Plot the histogram of the 10000 observed statistics from part (3). Then show the density curve of the theoretical distribution you found in part (1) on the histogram.

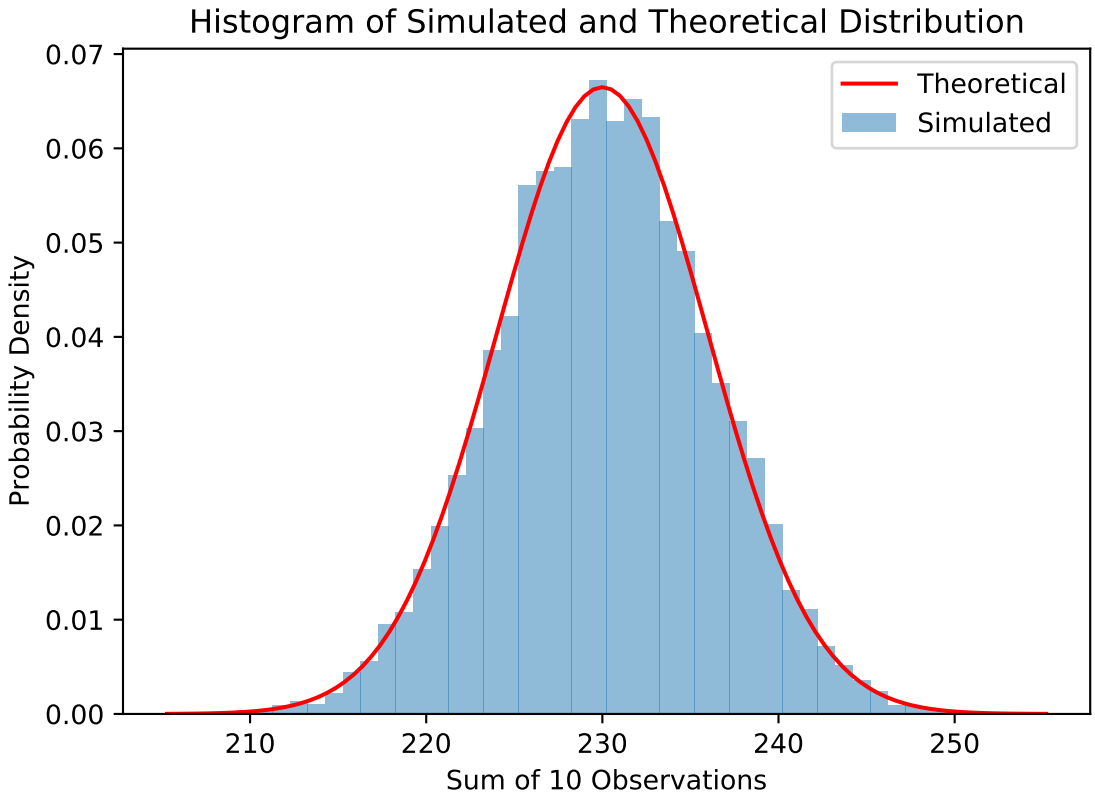
```
import matplotlib.pyplot as plt
from scipy.stats import norm

mu=23
sigSq=3.6

plt.hist(sampleSumList, bins=50, density=True, alpha=0.5, label='Simulated')

# Plotting the density curve of the Theoretical Distribution
x = np.linspace(np.min(sampleSumList), np.max(sampleSumList), 100)
plt.plot(x, norm.pdf(x, loc=10*mu, scale=np.sqrt(10*sigSq)), color='red', label='Theoretical')

plt.xlabel('Sum of 10 Observations')
plt.ylabel('Probability Density')
plt.legend()
plt.title('Histogram of Simulated and Theoretical Distribution')
plt.show()
```



Question 5

The continuous random variable X has the following probability density function (pdf), for some positive constant c,

$$f(x) = \frac{3}{(1+x)^3}, \quad 0 \leq x \leq c.$$

a. Find c which makes f a legitimate pdf?

To find the constant c that makes the fxn f(x) a legitimate pdf over the interval [0,c], we need to make sure that the total area under the pdf curve from 0 to c equals 1.

$$\int_0^c f(x)dx = 1$$

$$\int_0^c \frac{3}{(1+x)^3}dx = 1$$

$$\int \frac{3}{(1+x)^3}dx = -\frac{3}{2(1+x)^2} + C$$

$$\int \frac{3}{(1+x)^3}dx = \left[-\frac{3}{2(1+x)^2}\right]_0^c$$

$$= -\frac{3}{2(1+c)^2} - \left(-\frac{3}{2(1+0)^2}\right)$$

$$= -\frac{3}{2(1+c)^2} + \frac{3}{2}$$

Now, Set the integral to 1:

$$-\frac{3}{2(1+c)^2} + \frac{3}{2} = 1$$

$$-\frac{3}{2(1+c)^2} = -\frac{1}{2}$$

$$\frac{3}{2(1+c)^2} = \frac{1}{2}$$

$$(1+c)^2 = 3$$

$$(1+c) = \sqrt{3}$$

$$c = \sqrt{3} - 1 = 0.732$$

The value of c that makes f(x) a legitimate probability density function is: 0.732

b. Use R and plot the pdf curve of the random variable.

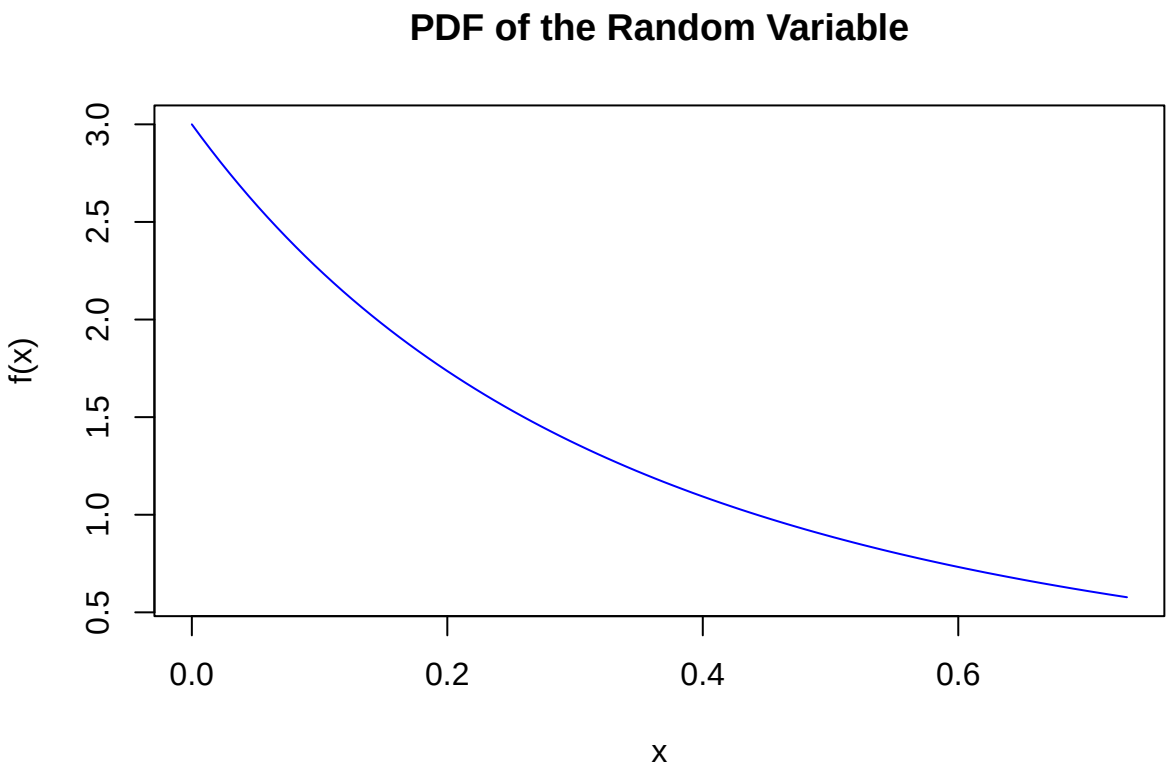
```r
# PDF function
f <- function(x, c) {
  3 / (1 + x)^3
}

# Setting the value for C
c <- sqrt(3) - 1

# Generate x values for plotting
x <- seq(0, c, length.out = 100)

# Calculating the PDF values
y <- f(x, c)

# Plotting the PDF curve
plot(x, y, type = "l", col = "blue", xlab = "x", ylab = "f(x)", main = "PDF of the Random Variable")
```



c. What is E(X)?

$$E(X) = \int_0^c x * f(x)dx$$

$$E(X) = \int_0^c x * \frac{3}{(1+x)^3}dx = 3\int_0^c \frac{x}{(1+x)^3}dx$$

Applying integration by parts:

$$\int u\ dv = uv - \int v\ du$$

$$\int \frac{x}{(1+x)^3}dx = x\left(-\frac{1}{2(1+x)^2}\right) - \int -\frac{1}{2(1+x)^2}dx$$

$$= -\frac{x}{2(1+x)^2} + \frac{1}{2}\int (1+x)^{-2}dx$$

Solving the integral:

$$\int (1+x)^{-2}dx = -\frac{1}{1+x}$$

So, we have:

$$\int \frac{x}{(1+x)^3}dx = -\frac{x}{2(1+x)^2} - \frac{1}{2(1+x)} + C$$

Evaluating with the limits [0,c]:

$$\int \frac{x}{(1+x)^3}dx = \left[-\frac{x}{2(1+x)^2} - \frac{1}{2(1+x)}\right]_0^c$$

At x=c:

$$-\frac{c}{2(1+c)^2}-\frac{1}{2(1+c)}$$

At x=0:

$$-\frac{0}{2(1+0)^2}-\frac{1}{2(1+0)}=-\frac{1}{2}$$

Therefore,

$$3\int \frac{x}{(1+x)^3}dx = 3\left(\left[-\frac{c}{2(1+c)^2}-\frac{1}{2(1+c)}\right]-\left[-\frac{1}{2}\right]\right)$$

$$\text{substituting } c=\sqrt{3}-1$$

$$E(X)=3\left(-\frac{\sqrt{3}-1}{2(\sqrt{3})^2}-\frac{1}{2\sqrt{3}}+\frac{1}{2}\right)=0.268$$

d. Use R and simulate 1000 observations from this statistical population?

```r
# Setting the value for c
c <- sqrt(3) - 1

# Simulating 1000 observations
n <- 1000
x <- (1 + runif(1000, 0, c)) - 1

print(x)
```

```
##    [1] 0.130105561 0.094924565 0.615072281 0.727737562 0.587139254 0.589696261
##    [7] 0.106076861 0.500773964 0.276388105 0.254104663 0.036242198 0.450213758
##   [13] 0.396032465 0.203766947 0.689698501 0.690040658 0.552073009 0.726427124
##   [19] 0.380780772 0.108547471 0.535085285 0.213668656 0.351739488 0.590512781
##   [25] 0.076614933 0.165585406 0.299357242 0.467988872 0.058513654 0.687738234
##   [31] 0.211879870 0.083546931 0.531702337 0.379441429 0.567908899 0.094173536
##   [37] 0.108082856 0.712879733 0.432295878 0.227250937 0.607967638 0.538502529
##   [43] 0.570301231 0.348644533 0.574046662 0.105421120 0.385230654 0.696127303
##   [49] 0.364068265 0.723077833 0.621606257 0.001145561 0.037460213 0.343342811
##   [55] 0.078697510 0.102123297 0.566939920 0.053464735 0.202561936 0.487146013
##   [61] 0.612539235 0.145874716 0.415732334 0.160903422 0.106401877 0.273223778
##   [67] 0.003957870 0.356371115 0.069246144 0.124340508 0.083014329 0.664856555
##   [73] 0.579836052 0.585474935 0.419949523 0.575825616 0.548299186 0.667289741
##   [79] 0.100069604 0.575103931 0.730424867 0.715845291 0.289587896 0.717765136
##   [85] 0.109738319 0.347369430 0.304482706 0.084619672 0.069861333 0.547692114
##   [91] 0.117611552 0.546951319 0.050501799 0.207055325 0.230950578 0.054488499
##   [97] 0.049646548 0.596659836 0.431415238 0.070739740 0.359060723 0.345349222
##  [103] 0.105023740 0.193748990 0.322315716 0.552173008 0.344017646 0.285193594
##  [109] 0.393429800 0.574555018 0.327951379 0.620659986 0.044948514 0.571623959
##  [115] 0.417752165 0.002874546 0.310694871 0.016288428 0.443550614 0.646393996
##  [121] 0.647818292 0.530759598 0.411434469 0.629391655 0.241731524 0.098920988
##  [127] 0.703008190 0.508587005 0.306337667 0.499037222 0.670763289 0.531803898
##  [133] 0.101968598 0.417045673 0.131827729 0.178143794 0.189947317 0.317243890
##  [139] 0.253444632 0.163244014 0.466673973 0.641770134 0.048144882 0.702347169
##  [145] 0.009373151 0.298451745 0.605035553 0.672448684 0.547698358 0.272757451
##  [151] 0.370088692 0.653360875 0.371038578 0.242924560 0.055543413 0.017067243
##  [157] 0.032057694 0.436156829 0.445750026 0.463189791 0.353862905 0.166097334
```

```
##  [163] 0.072772788 0.274845223 0.716675876 0.349085353 0.232459350 0.625786599
##  [169] 0.386703916 0.508812985 0.362000908 0.407193040 0.073868276 0.173460512
##  [175] 0.281557235 0.609752166 0.411674978 0.160212542 0.605498977 0.716212611
##  [181] 0.431946004 0.165808629 0.375923010 0.183892763 0.540540548 0.153685510
##  [187] 0.009374057 0.257339862 0.536684865 0.441314549 0.346378048 0.457748017
##  [193] 0.353829652 0.608756685 0.222894814 0.317623863 0.592859712 0.592002112
##  [199] 0.439654753 0.142412213 0.101812746 0.294720714 0.452472832 0.447886548
##  [205] 0.132666868 0.275989065 0.237928250 0.286183519 0.485379299 0.387109949
##  [211] 0.481366819 0.431765166 0.661924210 0.510826841 0.319814094 0.480289071
##  [217] 0.165392852 0.313855501 0.137845915 0.111689588 0.068598158 0.185064691
##  [223] 0.459891176 0.133784611 0.558598697 0.109228081 0.379332023 0.525596576
##  [229] 0.685701315 0.599190496 0.083709955 0.099063045 0.454238959 0.548264566
##  [235] 0.246069502 0.647112433 0.049161722 0.095271342 0.073662124 0.600430752
##  [241] 0.321534678 0.155258686 0.194241636 0.448151431 0.221683260 0.312237103
##  [247] 0.582627955 0.162423543 0.488161293 0.339171915 0.376239277 0.169590029
##  [253] 0.026544709 0.130808092 0.649703834 0.030351936 0.258413139 0.353811044
##  [259] 0.297426659 0.610028884 0.016130612 0.285343498 0.242985666 0.504034084
##  [265] 0.230364434 0.302109209 0.237251332 0.274178212 0.148583775 0.423468446
##  [271] 0.532405907 0.017021513 0.167266031 0.595470544 0.409478328 0.369489854
##  [277] 0.497154396 0.565184699 0.654984096 0.016055801 0.524631088 0.698728072
##  [283] 0.060848567 0.688516335 0.231571353 0.137971894 0.073019942 0.236393030
##  [289] 0.111147171 0.554208492 0.352223606 0.626617265 0.648549733 0.568495962
##  [295] 0.285064133 0.215464363 0.134630626 0.320614120 0.699722536 0.243147661
##  [301] 0.556522386 0.323306506 0.219793354 0.710053972 0.047955540 0.570130976
##  [307] 0.070370357 0.117948133 0.385290407 0.144500532 0.637046477 0.433552540
##  [313] 0.725754216 0.140733050 0.375243752 0.482485714 0.033951586 0.324973491
##  [319] 0.541410944 0.612656292 0.421870537 0.487953121 0.214181379 0.341073154
##  [325] 0.128412018 0.042868310 0.399669440 0.083246876 0.726459985 0.691745053
##  [331] 0.369578261 0.481205710 0.399165211 0.692987669 0.265183505 0.502583688
##  [337] 0.329291395 0.216730484 0.540068589 0.175238167 0.182838828 0.169673465
##  [343] 0.654830658 0.577308963 0.147747583 0.450206565 0.222634657 0.067985022
##  [349] 0.603767044 0.202927354 0.590766990 0.633102988 0.533809764 0.391954080
##  [355] 0.197291018 0.394893950 0.177093348 0.462621893 0.501942362 0.575673788
##  [361] 0.467311391 0.136352085 0.531749983 0.394515880 0.318240257 0.556387140
##  [367] 0.014295424 0.242945550 0.369078600 0.674542505 0.663583711 0.585420668
##  [373] 0.320895468 0.703211263 0.476091251 0.647551694 0.108422370 0.370082678
##  [379] 0.257726896 0.656133565 0.629260145 0.279741348 0.664037080 0.077037167
##  [385] 0.673808520 0.450578196 0.715204227 0.272374719 0.489060235 0.441030868
##  [391] 0.428227752 0.011985815 0.114243728 0.537843701 0.410050923 0.608356315
##  [397] 0.253285227 0.305089259 0.209600304 0.584395339 0.530053880 0.103927868
##  [403] 0.313964574 0.625793148 0.061666594 0.096375790 0.094841112 0.468478735
##  [409] 0.163024592 0.324802432 0.053801456 0.669553591 0.705212596 0.106562549
##  [415] 0.650883621 0.311169722 0.098738418 0.322226222 0.177157627 0.510259158
##  [421] 0.577379865 0.075974469 0.100335591 0.336607107 0.241545190 0.512124795
##  [427] 0.538577809 0.129768148 0.519442751 0.192812633 0.690382965 0.652141207
##  [433] 0.219272097 0.136032762 0.625535170 0.393135824 0.400369693 0.181175339
##  [439] 0.704595305 0.116885166 0.028149104 0.142274810 0.498251723 0.470293230
##  [445] 0.395230718 0.664381681 0.385869523 0.098243641 0.120616666 0.596288648
##  [451] 0.072173030 0.129325046 0.325406209 0.074193548 0.188883735 0.246198654
##  [457] 0.448133415 0.663954008 0.684660058 0.135349326 0.446153009 0.483481609
##  [463] 0.111982272 0.175470535 0.344043487 0.174642807 0.591914242 0.568159109
##  [469] 0.717364631 0.393946048 0.346158807 0.615497651 0.577677708 0.422834129
##  [475] 0.422339287 0.244193811 0.176788762 0.410741182 0.368306691 0.356542516
##  [481] 0.730718573 0.673146726 0.004377920 0.293459532 0.366755383 0.658952714
```

```
##  [487] 0.467401472 0.618989010 0.533189991 0.012148333 0.096194678 0.325535518
##  [493] 0.672660085 0.663266723 0.698911988 0.618226462 0.478708106 0.341337628
##  [499] 0.526929773 0.038475362 0.154212213 0.082856774 0.042310380 0.134736514
##  [505] 0.133308459 0.676939986 0.219484613 0.612451137 0.418463910 0.121607363
##  [511] 0.485503850 0.006620627 0.616051733 0.629224773 0.330140414 0.013956919
##  [517] 0.183772062 0.716678969 0.330257879 0.235258565 0.344343135 0.587692557
##  [523] 0.242172841 0.195088564 0.030355542 0.373880231 0.558699173 0.599448932
##  [529] 0.145514868 0.686550184 0.130753924 0.217396686 0.490548367 0.420456173
##  [535] 0.342139283 0.726998102 0.112253547 0.509987303 0.465136519 0.541014176
##  [541] 0.303215838 0.142824446 0.700448822 0.246081192 0.652262394 0.094521784
##  [547] 0.689951229 0.118975234 0.240060787 0.221550542 0.693596693 0.299386487
##  [553] 0.600503223 0.604047731 0.473538642 0.531420532 0.371744104 0.399178837
##  [559] 0.078081488 0.123888854 0.455899652 0.004467724 0.587940014 0.538012740
##  [565] 0.655925367 0.071782381 0.446423730 0.286441324 0.636292213 0.569179897
##  [571] 0.150290239 0.172150853 0.097516034 0.505956784 0.340431300 0.132542885
##  [577] 0.472137503 0.439055535 0.697932309 0.038532012 0.055656899 0.338449964
##  [583] 0.455908343 0.180791556 0.586450198 0.421711249 0.490438326 0.598027039
##  [589] 0.422150660 0.534691456 0.055428865 0.146274578 0.663709834 0.118456552
##  [595] 0.574596537 0.357048522 0.557743056 0.271405837 0.638990422 0.172876907
##  [601] 0.239280827 0.650452390 0.333592635 0.319125005 0.137589634 0.493925394
##  [607] 0.723731157 0.693783022 0.263794192 0.079887409 0.117537434 0.023439018
##  [613] 0.397380149 0.149906955 0.045578524 0.569660639 0.365600862 0.341475747
##  [619] 0.407050024 0.292519432 0.633801874 0.428125648 0.728079172 0.638394013
##  [625] 0.414918831 0.068704772 0.434776203 0.225534004 0.312923834 0.381240242
##  [631] 0.087607302 0.305332697 0.460794275 0.211637792 0.478559159 0.004035665
##  [637] 0.276814724 0.444037136 0.283241663 0.476045485 0.099787346 0.724018315
##  [643] 0.569922648 0.143487208 0.664136760 0.281413034 0.675790653 0.342276357
##  [649] 0.089717646 0.414375808 0.240328311 0.358058534 0.084699820 0.392716783
##  [655] 0.143458455 0.662137011 0.136590319 0.495961250 0.668909273 0.186334643
##  [661] 0.272390021 0.378715911 0.428649112 0.601937084 0.296438978 0.535752110
##  [667] 0.711158440 0.439061914 0.280277804 0.180151923 0.497721908 0.686569994
##  [673] 0.431337678 0.330166445 0.525483213 0.257418993 0.383364944 0.669026967
##  [679] 0.332632524 0.168269359 0.287430998 0.487699583 0.062554262 0.217730745
##  [685] 0.354259039 0.722374350 0.212568464 0.530175161 0.085794182 0.687684363
##  [691] 0.491503944 0.214716768 0.434507334 0.460054591 0.082533989 0.338639381
##  [697] 0.381513444 0.154313880 0.710477293 0.098106083 0.721821576 0.144256344
##  [703] 0.633513585 0.240788655 0.006995399 0.467291425 0.641860334 0.358476526
##  [709] 0.312371416 0.486227388 0.236968129 0.086478196 0.717683093 0.396132103
##  [715] 0.690694908 0.132077230 0.730666564 0.713308654 0.191417453 0.061040105
##  [721] 0.070362254 0.237869343 0.626921717 0.367067731 0.442116897 0.633872806
##  [727] 0.554553009 0.696922139 0.408507267 0.594727417 0.209707434 0.257274587
##  [733] 0.252635706 0.040936852 0.269909466 0.620373267 0.244555354 0.198228658
##  [739] 0.716417803 0.626468866 0.171544155 0.293725339 0.670308856 0.392484784
##  [745] 0.164243335 0.119570035 0.575028726 0.562208366 0.591409274 0.225087604
##  [751] 0.692742918 0.239768879 0.027404816 0.320511633 0.561140297 0.450702090
##  [757] 0.304990859 0.339801665 0.407357046 0.473108274 0.291376940 0.196295962
##  [763] 0.018518317 0.411661826 0.482107557 0.417549204 0.491050599 0.008304148
##  [769] 0.617129260 0.108787538 0.156261199 0.014608369 0.479112760 0.684731110
##  [775] 0.226068812 0.105611168 0.678191928 0.441068938 0.050350860 0.076584129
##  [781] 0.235906120 0.455523480 0.422522362 0.021208336 0.040585165 0.263375526
##  [787] 0.140967985 0.152568979 0.025407918 0.451215251 0.156336247 0.127975640
##  [793] 0.290835176 0.073019701 0.241452305 0.331554092 0.308136483 0.181436258
##  [799] 0.192753439 0.010741896 0.510945425 0.391916674 0.594642825 0.086233659
##  [805] 0.272436779 0.532894314 0.016238329 0.371002402 0.142817734 0.675098091
```

```
##  [811] 0.533046180 0.271659267 0.482495413 0.275750547 0.599661644 0.428604689
##  [817] 0.402206672 0.449961349 0.307249724 0.058698850 0.659616515 0.281754465
##  [823] 0.612202780 0.081514912 0.608697590 0.042529799 0.257502104 0.249416252
##  [829] 0.060605635 0.054402479 0.081747481 0.003560675 0.029112085 0.489310282
##  [835] 0.234271563 0.371843133 0.464966292 0.112868954 0.722293882 0.352200720
##  [841] 0.441427446 0.419143705 0.566502092 0.259321992 0.604033974 0.654488267
##  [847] 0.715731670 0.026154634 0.560338745 0.214777716 0.127447871 0.334664623
##  [853] 0.251128821 0.201319667 0.674442312 0.127245830 0.623915196 0.042236175
##  [859] 0.360463173 0.194428107 0.586147401 0.321156947 0.602099769 0.596558871
##  [865] 0.643082746 0.097082417 0.600856856 0.580959098 0.319577016 0.377681230
##  [871] 0.266182825 0.731349696 0.484973846 0.331791760 0.390539339 0.245831893
##  [877] 0.260150011 0.267042625 0.718224970 0.078066922 0.490003540 0.399363356
##  [883] 0.298370842 0.091752349 0.310359301 0.276480223 0.479777847 0.028556827
##  [889] 0.113264329 0.464341771 0.262605649 0.683770020 0.499942519 0.227307952
##  [895] 0.120081559 0.676486190 0.287431446 0.026022331 0.304302938 0.388362022
##  [901] 0.265394380 0.209352262 0.288457735 0.026922272 0.347838130 0.114651915
##  [907] 0.446276727 0.606825736 0.097821621 0.008673691 0.631587496 0.433213432
##  [913] 0.511245008 0.305614699 0.667762781 0.429232448 0.203999267 0.464279144
##  [919] 0.167821047 0.295428121 0.489510462 0.696932702 0.432539906 0.228474588
##  [925] 0.708127172 0.529978784 0.600003521 0.309643905 0.504826466 0.470256418
##  [931] 0.489519747 0.270164623 0.649935709 0.307815011 0.311253710 0.138989839
##  [937] 0.512899677 0.155039056 0.143173370 0.250362467 0.649727499 0.123924493
##  [943] 0.676444195 0.319709601 0.128061596 0.123792683 0.157720523 0.618895815
##  [949] 0.693036967 0.687659680 0.342640109 0.078067062 0.670706917 0.450603159
##  [955] 0.466865519 0.176511665 0.364519827 0.429522743 0.729202312 0.112141276
##  [961] 0.060316362 0.041272111 0.679736992 0.632967499 0.358994891 0.164072665
##  [967] 0.175335440 0.666247237 0.563705677 0.323918229 0.192110839 0.315637580
##  [973] 0.547423514 0.374329130 0.721236800 0.497086359 0.192764880 0.425630847
##  [979] 0.629041566 0.630595141 0.686616425 0.498389863 0.557770130 0.673978268
##  [985] 0.308130650 0.511265020 0.183966943 0.073311760 0.627760881 0.297321891
##  [991] 0.499221522 0.420423262 0.334864867 0.243130361 0.465093036 0.241381158
##  [997] 0.124206725 0.184638379 0.725541683 0.153467959
```

e. Use the generated data from part (d), estimate the mean and variance of the distribution?

```r
mean_x <- mean(x)

variance_x <- var(x)

print(paste("Estimated mean:", mean_x))
```

```
## [1] "Estimated mean: 0.362702100481413"
```

```r
print(paste("Estimated variance:", variance_x))
```

```
## [1] "Estimated variance: 0.0445444770104016"
```

Question 6

1. Write a Python function which takes a list of numbers x and returns a dictionary including

- Min : minimum value of x
- Q_1 : first quartile of x
- M : median of x
- Q_3 : third quartile of x
- Max : maximum value of x
- IQR : Q3 − Q1,
- Outliers : a list of x values which are either smaller than Q1−1.5×IQR or greater than Q3+1.5×IQR.

```
def statistics(x):

  # Calculating the required statistics
  min_x = min(x)
  q1 = np.percentile(x, 25)
  median_x = np.median(x)
  q3 = np.percentile(x, 75)
  max_x = max(x)
  iqr = q3 - q1
  outliers = [value for value in x if value < q1 - 1.5 * iqr or value > q3 + 1.5 * iqr]


  stats_dict = {
    "Min": min_x,
    "Q_1": q1,
    "M": median_x,
    "Q_3": q3,
    "Max": max_x,
    "IQR": iqr,
    "Outliers": outliers
  }

  return stats_dict
```

2. Apply your function in part (1) to the following list

$$x = [2, 36, 12, 14, 204, 21.6, 22.5, 1, 32.8, 32.1, 13, 10, 88, 3.3, 3.1, 88]$$

```
x = [2, 36, 12, 14, 204, 21.6, 22.5, 1, 32.8, 32.1, 13, 10, 88, 3.3, 3.1, 88]

stats_dict = statistics(x)

for key, value in stats_dict.items():
    print(f"{key}: {value}")
```

```
## Min: 1
## Q_1: 8.325
## M: 17.8
## Q_3: 33.599999999999994
## Max: 204
## IQR: 25.274999999999995
## Outliers: [204, 88, 88]
```

Question 7

2.

```
def leaveOneOutCrossValidation(X, y):

  n = X.shape[0]  # Number of observations
  y_pred = np.zeros(n)  # Vector to store predictions

  # Iterating through each observation
  for i in range(n):
    # Removing the i-th observation from X and y
    X_train = np.delete(X, i, axis=0)
    y_train = np.delete(y, i)

    # Train the model using remaining observations
    beta_star = np.linalg.solve(X_train.T @ X_train, X_train.T @ y_train)

    # Predicting the response for removed observation
    y_pred[i] = X[i, :] @ beta_star

  # root mean squared error
  rmse = np.sqrt(np.sum((y - y_pred)**2) / n)

  return rmse
```

3. Load the diabetes data from sklearn package in python and take a random sample of 56 observations from the data set and compute your X as the first three features of these 56 observations. The target of these observations will be your response vector.

```
from sklearn.datasets import load_diabetes

diab = load_diabetes()

# Extract the features (X) and target (y)
X = diab.data[:, :3]  # Take the first three features
y = diab.target

# Randomly select 56 observations
n = 56
random_indices = np.random.choice(X.shape[0], size=n, replace=False)
X_selected = X[random_indices]
y_selected = y[random_indices]
```

4. Apply your function form part 2 to the data set of part 3 and report the root MSE.

```
rmse = leaveOneOutCrossValidation(X_selected, y_selected)

print("Root Mean Squared Error (RMSE):", rmse)
```

```
## Root Mean Squared Error (RMSE): 174.8635594593287
```