Kay Quiballo | MSDS 460 | 05.17.2022

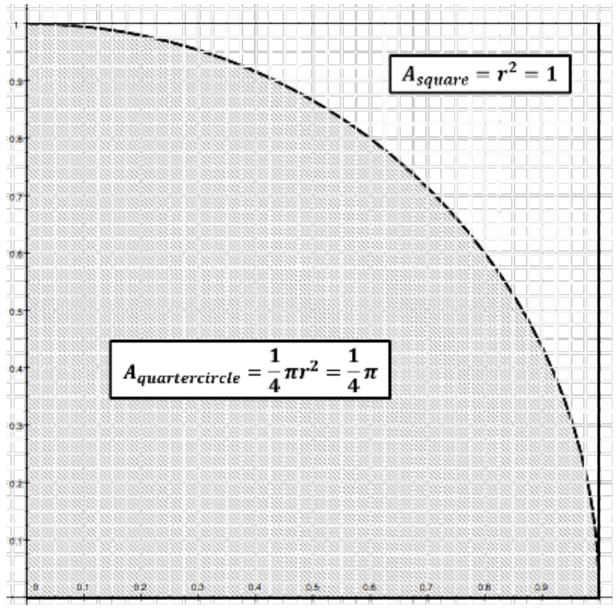
Homework Assignment (Problem Set) 4:
Note, Problem Set 3 directly focuses on Modules 7 and 8: Metaheuristic Algorithms and Monte Carlo Simulation

4 Questions

Rubric:
All questions worth 37.5 points
37.5 Points: Answer and solution are fully correct and detailed professionally.
25-37 Points: Answer and solution are deficient in some manner but mostly correct.
15-24 Points: Answer and solution are missing a key element or two.
1-14 Points: Answer and solution are missing multiple elements are significantly deficient/incomprehensible.
0 Points: No answer provided.

Kay Quiballo | MSDS 460 | 05.17.2022

1. Perform Monte Carlo integration using R statistical programming or Python programming to estimate the value of π. To summarize the approach, consider the unit quarter circle illustrated in the figure below:



Generate N pairs of uniform random numbers (x, y), where $x \sim U(0,1)$ and $y \sim U(0,1)$, and each (x, y) pair represents a point in the unit square. To obtain an estimate of π, count the fraction of points that fall inside the unit quarter circle and multiply by 4. Note that the fraction of points that fall inside the quarter circle should tend to the ratio between the area of the unit quarter circle (i.e., ¼ π) as compared to area of the unit square (i.e., 1). We proceed step-by-step:

a) Create a function insidecircle that takes two inputs between 0 and 1 and returns 1 if these points fall within the unit circle.
Box A- columns x and y take on random numbers from 0 to 1 "=RAND()"

b) Create a function estimatepi that takes a single input N, generates N pairs of uniform random numbers and uses insidecircle to produce an estimate of π as described above. In addition to the estimate of π, estimatepi should also return the standard error of this estimate, and a 95% confidence interval for the estimate.
Box B- column N indicates how many pairs of uniform random numbers are used to estimate pi.
Column estimatepi is how many points are in the circle * 4.
Column SE is the standard error "=STDEV([points])/SQRT(COUNT([points]))*4"
Column 95%CI is how much to add/subtract to estimatepi for confidence intervals
"=CONFIDENCE(0.05,4*STDEV.P([points]),COUNT([points]))"

c) Use estimatepi to estimate π for N = 1000 to 10000 in increments of 500 and record the estimate, its standard error and the upper and lower bounds of the 95% CI. How large must N be in order to ensure that your estimate of π is within 0.1 of the true value? In other words, how large must N be in order to ensure that the confidence interval is within 0.1 of the estimate of π?
Box C- column 95%CI shows for all N>=1500, we are 95% confident that the CI is within 0.1 of estimatepi

d) Using the value of N you determined in part c), run estimatepi 500 times and collect 500 different estimates of π. Produce a histogram of the estimates and note the shape of this distribution. Calculate the standard deviation of the estimates – does it match the standard error you obtained in part c)? What percentage of the estimates lies within the 95% CI you obtained in part c)?
In Box D, 500 replications of estimatepi (N=1500), are placed in a histogram which is distributed as approximately normal. SD of estimates (0.0419) is aprox. equal to SE of part C (0.0427) and 93.2% of estimates lie within the CI of part C.

Kay Quiballo | MSDS 460 | 05.17.2022

A

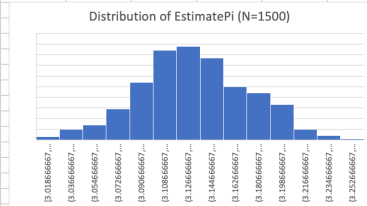
RANDOMIZED POINTS			
	x	y	inside_circle
1	0.8371002	0.2592672	1
2	0.4090016	0.8024613	1
3	0.9672878	0.976508	0
4	0.9245347	0.5100334	0
5	0.6052138	0.4280946	1
6	0.1175102	0.2059778	1
7	0.0469704	0.5951634	1
8	0.9797633	0.7545236	0
9	0.4953442	0.2172574	1
10	0.7774731	0.8464501	0
11	0.8048067	0.1734655	1
12	0.5597827	0.3669529	1
13	0.6715392	0.1559086	1
14	0.8778452	0.9784033	0
15	0.3512503	0.8409275	1
16	0.1621401	0.0138971	1
17	0.829054	0.2680477	1
18	0.8966355	0.8594428	0
19	0.2851265	0.5205988	1
20	0.5189462	0.7473343	1
21	0.5829242	0.4581119	1
22	0.8296781	0.4043132	1
23	0.4023976	0.7221956	1
24	0.7593669	0.8212686	0
25	0.3994892	0.3015134	1
26	0.8224656	0.3208082	1

B, C

TESTING N POINTS FROM 1000 TO 100000			
N	estimatepi	SE	95%CI
1000	3.084	0.0531768	0.1041725
1500	3.1226667	0.0427508	0.0837621
2000	3.114	0.0371509	0.0727963
2500	3.088	0.0335701	0.0657831
3000	3.0946667	0.0305649	0.0598961
3500	3.0971429	0.0282695	0.0553993
4000	3.094	0.0264758	0.0518851
4500	3.1048889	0.0248544	0.0487083
5000	3.1024	0.023602	0.0462544
5500	3.0989091	0.0225344	0.0441627
6000	3.1153333	0.021434	0.0420063
6500	3.1236923	0.0205229	0.0402211
7000	3.1257143	0.0197598	0.0387258
7500	3.1237333	0.0191053	0.0374432
8000	3.115	0.0185645	0.0363834
8500	3.1124706	0.0180285	0.0353332
9000	3.1053333	0.0175706	0.0344359
9500	3.1149474	0.0170361	0.0333884
10000	3.1164	0.016595	0.0325239

D

500 REPLICATIONS of N=1500				
	estimatepi	SE	95%CI	
replication	3.1226667	0.0427508	0.0837621	
1	3.1226667	0.0427508	0.0837621	1
2	3.1173333	0.0428439	0.0839445	1
3	3.176	0.0417833	0.0818665	1
4	3.1333333	0.0425626	0.0833934	1
5	3.088	0.0433446	0.0849256	1
6	3.056	0.0438694	0.0859538	1
7	3.1706667	0.0418831	0.082062	1
8	3.16	0.0420806	0.082449	1
9	3.152	0.042227	0.0827358	1
10	3.1306667	0.0426099	0.0834861	1
11	3.0373333	0.0441655	0.086534	0
12	3.184	0.0416323	0.0815707	1
13	3.0986667	0.0431648	0.0845732	1
14	3.128	0.042657	0.0835784	1
15	3.1386667	0.0424676	0.0832072	1
16	3.1226667	0.0427508	0.0837621	1
17	3.1786667	0.0417332	0.0817682	1
18	3.1013333	0.0431194	0.0844843	1
19	3.1946667	0.0414286	0.0811714	1
20	3.1466667	0.0423237	0.0829253	1
21	3.0933333	0.043255	0.08475	1
22	3.1466667	0.0423237	0.0829253	1
23	3.1306667	0.0426099	0.0834861	1
24	3.168	0.0419328	0.0821593	1
25	3.1573333	0.0421296	0.082545	1



SD of estimates	in 95% CI
0.0419298	0.932

Kay Quiballo | MSDS 460 | 05.17.2022

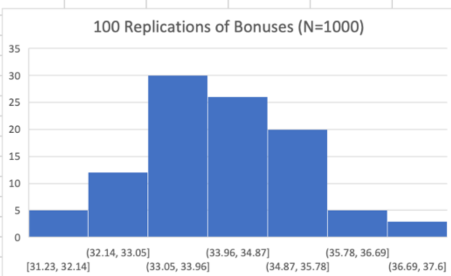
2. A salesperson in a large bicycle shop is paid a bonus if he sells more than 4 bicycles a day. The probability of selling more than 4 bicycles a day is only 0.40. If the number of bicycles sold is greater than 4, the distribution of sales as shown below. The shop has four different models of bicycles. The amount of the bonus paid out varies by type. The bonus for model A is \$10; 40% of the bicycles sold are of this type. Model B accounts for 35% of the sales and pays a bonus of \$15. Model C has a bonus rating of \$20 and makes up 20% of the sales. Finally, a model D pays a bonus of \$25 for each sale but accounts for only 5% of the sales. Develop a simulation model to calculate the bonus a salesperson can expect in a day. Note that if the salesman sells more than 4 bikes, he earns a bonus on each bike sold. That is, if 6 bikes are sold, he earns a bonus on each of the 6 bikes, not just the 2 bikes beyond the 4 bike cutoff.

Table

Number of Bicycles Sold	Probability
5	0.35
6	0.45
7	0.15
8	0.05

	Value	Probability	Cumulative
BIKES SOLD	4	0.6	0
	5	0.14	0.6
	6	0.18	0.74
	7	0.06	0.92
	8	0.02	0.98

BIKE TYPE	Type	Probability	Cunulative	Bonus
	A	0.4	0	10
	B	0.35	0.4	15
	C	0.2	0.75	20
	D	0.05	0.95	25



Simulation	bikes sold	bike 1	bike 2	bike 3	bike 4	bike 5	bike 6	bike 7	bike 8	bonus
1	6	B	A	A	B	B	B			80
2	4	C	B	A	A					0
3	4	D	A	B	B					0
4	5	B	A	C	C	B				80
5	5	C	B	D	A	A				80
6	4	A	C	A	A					0
7	5	A	C	B	D	A				80
8	4	A	B	A	B					0
9	4	A	B	A	C					0
10	5	B	A	C	A	A				65
11	4	A	A	A	A					0
12	4	A	B	A	A					0
13	4	B	A	B	A					0
14	4	A	C	D	A					0
15	4	A	A	A	B					0
16	6	C	C	D	B	A	A			100
17	4	A	C	C	B					0
18	6	B	C	C	B	B	A			95
19	4	B	A	B	C					0
20	4	B	A	A	B					0
21	4	C	B	A	A					0
22	5	A	D	A	C	A				75
23	6	A	C	C	A	A	B			85
24	5	B	C	A	C	B				80
25	4	A	A	C	B					0
26	4	B	A	B	D					0
27	5	D	B	A	B	C				85

	avg bonus
Replication	34.715
1	33.84
2	33.71
3	33.135
4	34.44
5	35.055
6	33.125
7	32.975
8	36.445
9	34.86
10	33.385
11	34.575
12	33.3
13	32.775
14	33.89
15	35.845
16	33.48
17	33.38
18	33.43
19	31.23
20	32.99
21	35.385
22	37.1
23	34.84
24	34.485
25	33.32
26	35.46

I first set up the tables with weighted probabilities – one for the number of bikes sold, and one for the type of bikes sold. Using random numbers, I simulate how many bikes are sold, and assign each bike to a type also using random numbers (and taking in to account the weighted probabilities).

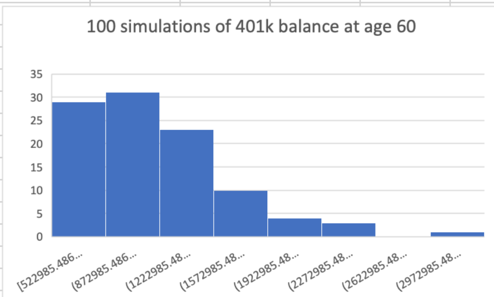
In the middle table, I ran 1000 simulations of how many bikes are sold and of what type to calculate the bonus. 4 or less results in no bonus whereas 5+ can be calculated from the bonus values.

In the right table, I replicate the simulation of N=1000, 100 times and plotted the average bonus amount in a histogram which shows we can expect \$33-\$34 of bonus on average.

Kay Quiballo | MSDS 460 | 05.17.2022

3. Michael is 24 years old and has a 401(k) plan through his employer, a large financial institution. His company matches 50% of his contributions up to 6% of his salary. He currently contributes the maximum amount he can (i.e., 6%). In his 401(k), he has three funds. Investment A is a large-cap index fund, which has had an average annual growth over the past 10 years of 6.63% with a standard deviation of 13.46%. Investment B is a mid-cap index fund with a 10-year average annual growth of 9.89% and a standard deviation of 15.28%. Finally, Investment C is a small-cap Index fund with a 10-year average annual growth rate of 8.55% and a standard deviation of 16.90%. Fifty percent of his contribution is directed to Investment A, 25% to Investment B, and 25% to Investment C. His current salary is $48,000 and based on a compensation survey of financial institutions, he expects an average raise of 2.7% with a standard deviation of 0.4% each year. Develop a simulation model to predict his 401(k) balance at age 60.

contributes 6%, company matches 50% = 9% total			
salary	48000		
raise avg	0.027		
raise sd	0.004		
	avg	sd	contribution amount
Investment A	0.0663	0.1346	0.5
Investment B	0.0989	0.1528	0.25
Investment C	0.0855	0.169	0.25



mean:	1206864.55
median:	1080390.67

year	salary	invest	A Total	B Total	C Total	TOTAL
24	48000	4320	2160	1080	1080	4320
25	49408.9278	4446.8035	4180.3829	2213.96663	2266.66117	8661.0107
26	50723.5423	4565.11881	7836.94013	3558.6152	3665.03093	15060.5863
27	51808.7567	4662.78811	10421.6746	4840.10544	5810.307	21072.087
28	52831.1483	4754.80335	14268.863	6401.25741	6387.04135	27057.1617
29	54496.9887	4904.72898	17961.0924	9186.38391	9788.82587	36936.3021
30	56019.9309	5041.79378	24737.1661	12780.5629	11101.1095	48618.8386
31	57435.8812	5169.2293	28161.7123	16642.2111	15771.804	60575.7274
32	58973.1907	5307.58716	37600.2429	18331.6431	14801.798	70733.6839
33	60838.7408	5475.48668	41791.4999	22765.0057	16267.6619	80824.1674
34	62350.3615	5611.53254	46841.3492	26520.209	17540.9222	90902.4804
35	63536.0511	5718.2446	56752.9012	31053.5425	20898.0596	108704.503
36	65092.2218	5858.29996	65881.5491	41331.2656	28355.9395	135568.754
37	66936.5431	6024.28888	72015.9901	42805.1764	28047.8927	142869.059
38	68874.6126	6198.71514	90686.7811	49442.3344	26490.0904	166619.206
39	71383.5761	6424.52185	99075.4018	51248.8267	34869.9893	185194.218
40	73577.233	6621.95097	126319.495	66437.8822	43671.6863	236429.064
41	75655.0769	6808.95692	121114.703	70903.4904	63073.9787	255092.172
42	77251.5356	6952.6382	134461.107	61556.2465	63886.9442	259904.298
43	79716.8116	7174.51304	174143.382	94496.7253	78146.2073	346786.315
44	82420.8575	7417.87718	172409.843	99196.9697	97803.7476	369410.56
45	84492.7046	7604.34341	198752.569	98185.5268	101715.237	398653.333
46	86270.227	7764.32043	202764.706	92434.7312	125738.888	420938.325
47	88479.5912	7963.16321	219981.877	78935.5608	152007.173	450924.611
48	90972.6003	8187.53403	268437.422	102319.044	203087.641	573844.108
49	93819.3141	8443.73827	325571.422	148839.386	186794.03	661204.838
50	96201.6028	8658.14425	342319.235	119684.969	241165.132	703169.336

	balance at 60
replication	1347263.64
1	1427840.57
2	2236025.64
3	843448.25
4	1029990.28
5	996216.193
6	1692734.68
7	1502555.56
8	1026308.3
9	826549.238
10	1452374.33
11	778572.184
12	1019737.45
13	1013723.82
14	1056708.07
15	1026614.02
16	849306.931
17	1166898.09
18	1435507.41
19	1094810.85
20	851139.58
21	811558.793
22	1780428.27
23	833732.77
24	872366.736
25	802827.177
26	1073439.22

I first set up the means and standard deviation tables on the left – one for starting salary/raises, and one for average and standard deviation of investments by type.

In the middle table, I simulated years 24-60. The salary raise increases using a random number along with the normal distribution of the average raise and standard deviation of raise. 6% is invested plus an additional 3% is matches, so yearly contributions = 9% of salary.

Contribution percentages are firm, but the return on investment will vary between funds. For example, year 30/investment A is 50% of the yearly contributions (9% of year 30 salary) + current funds (year 29/investment A) + growth from last year (year 29/investment A times growth factor – a random number of avg growth for A and sd of A).

I simulated the 401k balance at age 60, 100 times which are shown in the histogram. The average balance at age 60 is $1,206,864 and the median balance is $1,080,390.

Kay Quiballo | MSDS 460 | 05.17.2022

4. Develop a simulated annealing procedure in either R or Python to solve the following knapsack problem: (Note, this problem can be solved to optimality using integer programming; however, the focus of this question is on developing the simulated annealing method).

Simulated annealing:

Maximize $12x_1 + 16x_2 + 22x_3 + 8x_4$

S.T. $4x_1 + 5x_2 + 7x_3 + 3x_4 \leq 14$

$x_i \sim$ binary for all i

```
Begin knapsack simulated annealing demo
Goal is to maximize value subject    to max size constraint

Item values:
[12 16 22  8]

Item sizes:
[4 5 7 3]

Max total size = 14

Settings:
max_iter = 1000
start_temperature = 10000.0
alpha = 0.98

Starting solve()
Initial guess:
[1 1 1 1]
iter =      0 : curr value =      0 :         curr temp =   10000.00
iter =    100 : curr value =      0 :         curr temp =    1326.20
iter =    200 : curr value =     16 :         curr temp =     175.88
iter =    300 : curr value =     42 :         curr temp =      23.33
iter =    400 : curr value =     42 :         curr temp =       3.09
iter =    500 : curr value =     42 :         curr temp =       0.41
iter =    600 : curr value =     42 :         curr temp =       0.05
iter =    700 : curr value =     42 :         curr temp =       0.01
iter =    800 : curr value =     42 :         curr temp =       0.00
iter =    900 : curr value =     42 :         curr temp =       0.00
Finished solve()

Best packing found:
[1 0 1 1]

Total value of packing = 42.0
Total size  of packing = 14.0

End demo
```

Double check with pulp solutions:

```
Pulp Solutions
x1 1.0
x2 0.0
x3 1.0
x4 1.0
Total Value: 42.0
Total Space: 14.0
```

Both methods come to the solution of taking x1, x3, and x4, for a total value of 42 and total space of 14.