Lab assignment-1

Read and work through all code in the lab demonstration section, and do all exercises below

Note: The version on the website is just an HTML preview, please go to the share-point folder to download the actual assignment and dependencies.

SUBMISSION: Upload the completed *assignment* to Canvas (in HTML or PDF). Make sure ALL images & outputs are included! Please do not submit the completed demonstration, only the completed assignment.

Problem-1

- This relates to Example 1 in Lab-1, you can recycle code from there
- Mike had the first three successes in trials 6,8 , and 9 . He had six failures until he reached three successes. Do you think Mike has success probability $p = 0.5$ or better? Can a simulation give an answer? To do this do.
  - simulate the number of tosses needed to get to three successes, use the probability success probability p = 0.5. (Hint: use the "mytoss" function we did in the Lab class)
  - Run many simulations (say 10,000) with this success probability. (use the "myattempts" function from the class)
  - If Mike's success probability were 0.5 or better, he would not need a lot of attempts. Find the fraction of simulations where three successes were reached after 9 tosses or later by somebody with success probability 0.5.

```
mytoss = function(p){
  u = runif(1)
  x = as.numeric(u < p)
  return(x)
}
```

```
# This function calculates the number of total attempts to get a success

myattempts = function(p){
  counter = 1
  while (mytoss(p) == 0){ counter = counter + 1 }
  return(counter)
}
```

```
replicate(10,mytoss(.5))
```

1. 1
2. 0
3. 0
4. 0
5. 1
6. 0
7. 1
8. 0
9. 0
10. 0

```
set.seed(23)
replicate(3,myattempts(0.5))
```

1. 2
2. 1
3. 3

This means there were 2 attempts to get a first scuccess, 1 attempt to get a second success, and 3 attempts to get a third success, which is total of 6 attempts.

```
# Number of total attempts until the 3 successes

sum(replicate(3,myattempts(0.5)))
```

10

```
# 10000 simulations of calculating the total number of attempts until the 3 successes

replicate(10000,sum(replicate(3,myattempts(0.5))))
```

1. 9

2. 8
3. 4
4. 6
5. 6
6. 7
7. 6
8. 4
9. 9
10. 7
11. 7
12. 3
13. 7
14. 4
15. 7
16. 8
17. 6
18. 6
19. 3
20. 6
21. 4
22. 5
23. 9
24. 7
25. 9
26. 5
27. 10
28. 3
29. 14
30. 6
31. 6
32. 7
33. 7
34. 7
35. 3
36. 6
37. 4
38. 9
39. 12
40. 5
41. 11
42. 5
43. 5
44. 7

45. 3
46. 5
47. 6
48. 6
49. 5
50. 7
51. 6
52. 11
53. 4
54. 3
55. 8
56. 11
57. 3
58. 4
59. 11
60. 4
61. 7
62. 3
63. 3
64. 6
65. 4
66. 3
67. 5
68. 5
69. 5
70. 5
71. 3
72. 3
73. 5
74. 5
75. 4
76. 9
77. 8
78. 6
79. 4
80. 7
81. 5
82. 7
83. 3
84. 5
85. 4
86. 8
87. 7

88. 4
89. 4
90. 10
91. 9
92. 8
93. 4
94. 6
95. 5
96. 4
97. 6
98. 6
99. 5
100. 9
101. 6
102. 7
103. 9
104. 10
105. 4
106. 8
107. 5
108. 8
109. 6
110. 10
111. 5
112. 6
113. 3
114. 11
115. 5
116. 9
117. 7
118. 8
119. 7
120. 6
121. 6
122. 9
123. 4
124. 9
125. 8
126. 8
127. 4
128. 6
129. 9
130. 6

131. 3
132. 5
133. 5
134. 8
135. 4
136. 8
137. 9
138. 7
139. 4
140. 4
141. 3
142. 6
143. 5
144. 5
145. 4
146. 6
147. 4
148. 12
149. 6
150. 3
151. 9
152. 7
153. 3
154. 4
155. 6
156. 6
157. 3
158. 6
159. 7
160. 4
161. 5
162. 8
163. 5
164. 3
165. 10
166. 5
167. 10
168. 6
169. 12
170. 6
171. 8
172. 9
173. 6

174. 7
175. 10
176. 8
177. 7
178. 5
179. 3
180. 9
181. 4
182. 7
183. 4
184. 4
185. 6
186. 8
187. 6
188. 9
189. 4
190. 7
191. 10
192. 6
193. 4
194. 5
195. 8
196. 5
197. 4
198. 7
199. 10
200. 5
201. ...
202. 9
203. 3
204. 5
205. 6
206. 9
207. 9
208. 8
209. 5
210. 5
211. 4
212. 5
213. 5
214. 9
215. 5
216. 9

217. 4
218. 5
219. 4
220. 6
221. 4
222. 7
223. 5
224. 5
225. 4
226. 7
227. 10
228. 6
229. 6
230. 12
231. 6
232. 6
233. 4
234. 4
235. 6
236. 8
237. 6
238. 4
239. 8
240. 6
241. 4
242. 5
243. 3
244. 12
245. 9
246. 7
247. 6
248. 10
249. 4
250. 6
251. 7
252. 10
253. 4
254. 4
255. 4
256. 4
257. 5
258. 8
259. 6

260. 6
261. 5
262. 6
263. 10
264. 3
265. 6
266. 9
267. 6
268. 8
269. 5
270. 6
271. 5
272. 6
273. 5
274. 4
275. 7
276. 5
277. 4
278. 7
279. 8
280. 7
281. 6
282. 9
283. 4
284. 6
285. 8
286. 5
287. 7
288. 6
289. 6
290. 3
291. 4
292. 5
293. 6
294. 5
295. 4
296. 5
297. 5
298. 10
299. 7
300. 5
301. 3
302. 8

303.	4
304.	4
305.	3
306.	4
307.	6
308.	6
309.	5
310.	7
311.	4
312.	8
313.	4
314.	3
315.	5
316.	3
317.	5
318.	5
319.	8
320.	9
321.	6
322.	4
323.	3
324.	8
325.	3
326.	6
327.	12
328.	5
329.	4
330.	4
331.	4
332.	5
333.	4
334.	8
335.	7
336.	3
337.	4
338.	3
339.	5
340.	4
341.	8
342.	5
343.	5
344.	7
345.	10

346. 8
347. 11
348. 5
349. 4
350. 11
351. 3
352. 3
353. 6
354. 4
355. 5
356. 4
357. 4
358. 3
359. 9
360. 8
361. 6
362. 9
363. 3
364. 5
365. 6
366. 14
367. 5
368. 7
369. 6
370. 5
371. 6
372. 7
373. 3
374. 4
375. 7
376. 8
377. 9
378. 10
379. 10
380. 4
381. 5
382. 13
383. 5
384. 4
385. 4
386. 7
387. 5
388. 9

```
389. 3
390. 4
391. 3
392. 4
393. 6
394. 5
395. 6
396. 5
397. 5
398. 5
399. 8
400. 6
401. 11
```

```r
# Number of simulations where Mike needs more than 9 attempts to get the 3 successes

set.seed(23)
s = sum(replicate(10000,sum(replicate(3,myattempts(0.5))))>9)
s
s/10000 # Calculating the fraction
```

```
925
```

```
0.0925
```

The fraction of simulation where it needs more than 9 attempts to get the 3 successes is 0.0925 when the success probability is 0.5. This means there is 90.75% chance that this person gets 3 successes in less than or equal to 9 attempts. It implies, therefore, Mike is most likely to have a better probability than 0.5, because he got 3 successes in 9 attempts.

Problem 2:

Baby names for male and female babies.

Repeat Example 2 using `data/yob2010.txt`. Interpret your results (in wording relates to the problem) for each simulation step by step.

Do the following

- Read csv file with baby names and make new column names
- Make a the table function returns a vector whose components have names.
- Compute the number of female and male births, using the aggregate() function
- Sample 30 names with replacement. Use probabilities that are proportional to the counts.
- Compute actual probabilities of occurrence

- Find all names which occur both as male and female baby names. First make subsets of female and male births. Then find the set intersection of the names.
- Find out how often each of these names is used for females and males. Use sub-setting to make data frames of female and male birth data for names that are used for both genders. We may drop the gender variable.
- Now merge the two data frame by names. R finds the merge variable (i.e. name) automatically.

```r
# Read csv file with baby names and make new column names

df = read.csv("data/yob2010.txt", header = FALSE)
names(df) = c("names","gender","count")
head(df)
```

A data.frame: 6 x 3

	names <chr>	gender <chr>	count <int>
1	Isabella	F	22883
2	Sophia	F	20612
3	Emma	F	17322
4	Olivia	F	17012
5	Ava	F	15418
6	Emily	F	14260

```r
# Make a the table function returns a vector whose components have names.

table(df$gender)
```

```
    F     M 
19800 14241 
```

There are 19800 female names and 14241 male names.

```r
# Compute the number of female and male births, using the aggregate() function

aggregate(count ~ gender, data = df, FUN = "sum")
```

A data.frame: 2 x 2

gender <chr>	count <int>
F	1772738
M	1913851

There are 1772738 births of females and 1913851 births of males.

```
# Sample 30 names with replacement. Use probabilities that are proportional to the counts.
# Compute actual probabilities of occurrence

df$probs = df$count/sum(df$count)
head(df)
sample(df$names,30,replace=TRUE,prob=df$probs)
```

A data.frame: 6 x 4

	names <chr>	gender <chr>	count <int>	probs <dbl>
1	Isabella	F	22883	0.006207093
2	Sophia	F	20612	0.005591076
3	Emma	F	17322	0.004698652
4	Olivia	F	17012	0.004614564
5	Ava	F	15418	0.004182186
6	Emily	F	14260	0.003868074

1. 'Dayanna'
2. 'Ruben'
3. 'Bridger'
4. 'Anthony'
5. 'Tamia'
6. 'Jadyn'
7. 'Richard'
8. 'Nevaeh'
9. 'Nicholas'
10. 'Liberty'
11. 'Joaquin'
12. 'Curtis'
13. 'Iker'
14. 'Finn'
15. 'Stacy'
16. 'Daisy'
17. 'Andre'
18. 'Jayden'

19. ‘Sienna’
20. ‘Lakayla’
21. ‘Sophia’
22. ‘Luke’
23. ‘Brett’
24. ‘Gavin’
25. ‘Paxton’
26. ‘Andrea’
27. ‘Cedric’
28. ‘Harper’
29. ‘Mackenzie’
30. ‘Jose’

Computing the probabilities of occurrence of each name, it is possible to sample 30 names using those probability.

```
# Find all names which occur both as male and female baby names. First make subsets of fem

femalebirths = df[df$gender == "F",]
malebirths = df[df$gender == "M",]
both = intersect(femalebirths$name, malebirths$name)
head(both)
length(unique(both))
```

1. ‘Isabella’
2. ‘Sophia’
3. ‘Emma’
4. ‘Olivia’
5. ‘Ava’
6. ‘Emily’

2438

There are 2438 names used on both female and male babies.

```
# Find out how often each of these names is used for females and males. Use sub-setting to
# that are used for both genders. We may drop the gender variable.

df.female.both <- df[is.element(df$name,both) & df$gender == "F",c(1,3)]
names(df.female.both) <- c("name","female.count")
head(df.female.both)
df.male.both <- df[is.element(df$name,both) & df$gender == "M",c(1,3)]
names(df.male.both) <- c("name","male.count")
```

```r
head(df.male.both)
```

A data.frame: 6 x 2

	name <chr>	female.count <int>
1	Isabella	22883
2	Sophia	20612
3	Emma	17322
4	Olivia	17012
5	Ava	15418
6	Emily	14260

A data.frame: 6 x 2

	name <chr>	male.count <int>
19801	Jacob	22082
19802	Ethan	17985
19803	Michael	17308
19804	Jayden	17152
19805	William	17030
19806	Alexander	16742

Among the names used in both gender, Isabella, Sophia, Emma, Olivia, Ava, and Emily are used the most in female babies, and Jacob, Ethan, Michael, Jayden, William, and Alexander are used the most in male babies.

```r
# Now merge the two data frame by names. R finds the merge variable (i.e. name) automatica

df.both = merge(df.female.both,df.male.both)
head(df.both)
```

A data.frame: 6 x 3

	name <chr>	female.count <int>	male.count <int>
1	Aaliyah	4657	6
2	Aamari	8	8
3	Aaren	5	31
4	Aarin	7	21
5	Aaron	23	7450
6	Aarya	72	26

name <chr>	female.count <int>	male.count <int>

Problem-3

- Problem 1.5 in ch. 1 of Dalgaard. On p. 27, replicate was used to simulate the distribution of the mean of 20 random numbers from the exponential distribution by repeating the operation 10 times. That code is `replicate(10,mean(rexp(20)))`

How would you do the same thing with sapply?

```
replicate(10,mean(rexp(20)))
```

1. 1.03612283387355
2. 0.923260881481287
3. 1.19727332571717
4. 1.17321020551394
5. 1.10806502017397
6. 1.56834551326154
7. 1.39519816576345
8. 0.7829826150716
9. 1.28158410672325
10. 0.969547110458222

```
sapply(data.frame(replicate(10,rexp(20))),mean)
```

X1 1.02869093508165X2
1.270363214501X3
0.999161576189135X4
1.12646868796969X5
1.01373915195644X6
0.939738756882871X7
0.959728856241185X8
1.24112073836954X9
0.817850003916886X10
0.764905193270278