Algorithms_II_Lab-01

August 17, 2021

0.1 LAB 01 : Matrices, vectors, and beginning R for statistics

0.1.1 How to define a vector in R

The R computing language shares many similarities to Python, however, we need to learn a slightly different syntax for defining objects in R.

For example, in Python we defined a function called "sum" that took two objects a and b like this

```
def sum(a,b):
    return a+b
```

In R you can also define a function, but how you define a function looks a bit different.

```
[3]: sum = function(a,b){
         return(a+b)
     }
     sum(8,9)
```

17

Above, we create an object called "sum" that is assigned a function that takes as input two objects a,b. After a few weeks of work you will see that R and Python behave similarly, and by learning a R you will strengthen your understanding of both R and Python (really).

0.1.2 Vectors

The vector is a fundemental object in R. We can define a vector v like this

```
[4]: v = c(1,2,3) # (the "c" stands for combine, as in, combine these numbers into a␣
      ↪vector)
```

This is a vector of length three containing the ordered numbers 1,2, and 3. This should remind you of the Python list.

In R, you can acces the first element of a vector by referencign the name of the vector, square brackets, and the number 1.

```
[6]: v[1]
```

1

You can access the second element like this

[9]:
```
v[2]
```

2

The number 2 is often called an **index**.

In Python, we were able to locate items in a list by entering a positive number (forward indexing) or by entering a negative number (looking backward). Negative indices in R do **NOT** work the same.

[13]:
```
v[-1]
```

1. 2 2. 3

A negative index **removes** that item from the vector and returns the remaining items.

[15]:
```
v[-2]
```

1. 1 2. 3

The length of a vector can be computed in R using the length function. The length function is useful for accessing the last item in a vector.

[17]:
```
L = length(v)
v[L]
```

3

Vectors in R work just like they do in the mathematics we learned during lecture. You can multiply a vector by a scalar

[18]:
```
3*v
```

1. 3 2. 6 3. 9

and you can add two vector so long as they have the same length

[26]:
```
v = c(1,2,3)
k = c(3,2,1)
q = c(3,2,1,-1)
j = c(1,0)
```

[27]:
```
v+k
```

1. 4 2. 4 3. 4

But you need to be very careful if two vectors do not have the same length because R will still return a vector. Lets look at what happens when we add k of length 3 and j of length 2

[28]:
```
k+j
```

```
Warning message in k + j:
“longer object length is not a multiple of shorter object length”
```

1. 4 2. 2 3. 2

R still returns a new vector! But what happened here? The vector *k* contains the ordered item (3,2,1) and *j* contains the items (1,0) and is one item shorter than *k*. To make the vector *j* a length of 3, R tacked onto the end of *j* the first item again. That is, R changed *i* from (1,0) to (1,0,1) and then added together *k* and this “extended” *j* vector. Yikes!

Lets look at another example, adding *q* of length 4 and *j* of length 2.

[29]:
```
q+j
```

1. 4 2. 2 3. 2 4. -1

What is happening here? The vector *q* has four items but *j* has only two. To make *j* the same length as *q*, R extended *j*. The first item of *j* was copied into the third position and the second item in *j* was copied into the fourth item. R changed *j* from (1,0) to (1,0,1,0).

This process in R is called **recycling**

0.1.3 Matrices

Matrices in R can be built by calling, you guessed it, the function called matrix. The matrix function requires you to provide a vector of elements to include in the matrix and either the number of rows, number of columns, or both.

[40]:
```
M = matrix(c(1,2,3,4),nrow=2)
print(M)
```

```
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

By default, R fills in the matrix column by column. If you would rather the matrix be filled row by row you can add the option byrow=TRUE. Just like in python, the word TRUE is a special object in R.

[41]:
```
M = matrix(c(1,2,3,4),nrow=2,byrow=TRUE)
print(M)
```

```
     [,1] [,2]
[1,]    1    2
[2,]    3    4
```

Individual elements in a matrix can be accessed by typing the name of the matrix, square brackets, and entering the row number and column number of the itme you want.

[47]:
```
M[1,2]
```

1. 1 2. 2

If you only provide a row number, then R returns the entire row

[50]:
```
M[1,]
```

1. 1 2. 2

If you only provide a column number, then R returns the entire column

[53]:
```
M[,2]
```

1. 2 2. 4

Just like we learned in class, you can add and subtract matrices in R.

[45]:
```
M = matrix(c(1,2,3,4),nrow=2,byrow=TRUE)
N = matrix(c(-1,0.98,-10,4),nrow=2,byrow=TRUE)

print("M")
print(M)

print("N")
print(N)

Z = M+N
print("M plus N")
print(Z)

print("M minus N")
print(M-N)
```

```
[1] "M"
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[1] "N"
     [,1] [,2]
[1,]   -1 0.98
[2,]  -10 4.00
[1] "M plus N"
     [,1] [,2]
[1,]    0 2.98
[2,]   -7 8.00
[1] "M minus N"
     [,1] [,2]
[1,]    2 1.02
[2,]   13 0.00
```

To multiply a matrix times a vector, we need to use a special multiply symbol in R

```
[55]: M = matrix(c(1,2,3,4),nrow=2,byrow=TRUE)
      v = c(-1,1)

      print("M")
      print(M)

      print("v")
      print(v)

      mTimesv = M%*%v

      print("M times v")
      print(mTimesv)
```

```
[1] "M"
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[1] "v"
[1] -1  1
[1] "M times v"
     [,1]
[1,]    1
[2,]    1
```

0.1.4 Loading a package in R

To load a package in R you use the require or library command.

```
[56]: library(MASS)
```

0.1.5 The data frame

One of the most important objects in R is the dataframe. The dataframe is a matrix where rows represent observations and columns correpsond to different covariates.

Inside the MASS package, which we just loaded, are a ton of datasets for free. lets look at the dataset *GAGurine*. This dataset is described as

"Data were collected on the concentration of a chemical GAG in the urine of 314 children aged from zero to seventeen years. The aim of the study was to produce a chart to help a paediatrican to assess if a child's GAG concentration is 'normal'."

```
[65]: d = GAGurine
      print(d)
```

```
      Age  GAG
1    0.00 23.0
2    0.00 23.8
```

```
3    0.00 16.9
4    0.00 18.6
5    0.01 17.9
6    0.01 25.9
7    0.01 16.5
8    0.01 26.3
9    0.01 26.9
10   0.01 17.9
11   0.01 29.1
12   0.02 32.6
13   0.02 41.9
14   0.03 38.7
15   0.03 37.6
16   0.03 46.4
17   0.04 30.5
18   0.04 34.8
19   0.04 26.7
20   0.05 33.0
21   0.05 56.3
22   0.05 33.5
23   0.05 29.7
24   0.05 32.8
25   0.05 27.6
26   0.06 31.9
27   0.07 55.4
28   0.10 33.3
29   0.10 36.4
30   0.14 25.9
31   0.14 21.6
32   0.15 26.0
33   0.17 12.8
34   0.18 34.2
35   0.23 19.4
36   0.24 21.9
37   0.24 19.3
38   0.25 26.2
39   0.27  9.6
40   0.28 36.1
41   0.29 19.7
42   0.29 19.2
43   0.30 20.4
44   0.33 23.1
45   0.33 23.8
46   0.40 21.6
47   0.41 25.4
48   0.45 18.4
49   0.46 18.6
50   0.47 26.4
```

```
51   0.50 25.7
52   0.50 20.4
53   0.53 15.9
54   0.55 20.8
55   0.56 20.9
56   0.56 15.7
57   0.57 20.4
58   0.58 14.1
59   0.59 21.4
60   0.59 15.6
61   0.62 30.5
62   0.64 15.4
63   0.65 19.2
64   0.65 22.6
65   0.69 15.2
66   0.71 14.9
67   0.75 39.5
68   0.80 17.3
69   0.80 17.2
70   0.82 13.8
71   0.83 17.7
72   0.84 15.6
73   0.85 20.4
74   0.86 14.8
75   0.86 19.5
76   0.87 18.0
77   0.89 22.5
78   0.94 20.4
79   0.96 13.1
80   0.97 17.4
81   0.98 18.9
82   0.99 16.1
83   1.02 21.9
84   1.03 18.4
85   1.03 15.2
86   1.05 19.1
87   1.07 15.1
88   1.07 17.2
89   1.07 18.4
90   1.08 24.1
91   1.15 19.0
92   1.23 16.0
93   1.24 21.8
94   1.25 16.5
95   1.25 16.2
96   1.25 10.8
97   1.30 27.0
98   1.31 17.3
```

99	1.31	18.3
100	1.31	18.3
101	1.35	13.3
102	1.37	12.1
103	1.39	11.0
104	1.39	12.7
105	1.53	13.4
106	1.59	14.3
107	1.65	11.5
108	1.75	16.5
109	1.76	25.1
110	1.77	16.3
111	1.84	15.4
112	1.86	18.6
113	1.91	18.3
114	1.99	13.2
115	1.99	11.1
116	2.02	11.0
117	2.07	12.5
118	2.09	14.8
119	2.11	16.8
120	2.12	12.8
121	2.13	12.2
122	2.19	14.7
123	2.29	9.7
124	2.32	14.2
125	2.37	13.9
126	2.41	9.5
127	2.43	14.6
128	2.49	12.1
129	2.58	10.6
130	2.73	10.8
131	2.73	10.4
132	2.74	9.9
133	2.77	9.6
134	2.82	21.7
135	2.87	11.8
136	2.87	11.8
137	2.92	13.0
138	2.95	11.3
139	2.98	9.0
140	3.00	18.3
141	3.02	14.6
142	3.03	11.5
143	3.10	7.3
144	3.13	11.6
145	3.18	8.6
146	3.19	11.5

147	3.21	8.3
148	3.29	10.3
149	3.29	12.5
150	3.63	13.2
151	3.64	17.9
152	3.64	10.7
153	3.71	8.6
154	3.85	12.5
155	3.98	8.6
156	4.07	6.0
157	4.09	12.6
158	4.11	12.7
159	4.21	10.2
160	4.27	10.7
161	4.29	8.4
162	4.29	9.9
163	4.35	7.7
164	4.38	13.6
165	4.39	12.3
166	4.43	7.5
167	4.48	11.8
168	4.50	7.7
169	4.55	10.3
170	4.65	16.9
171	4.67	8.5
172	4.68	9.7
173	4.70	10.6
174	4.75	6.4
175	4.76	8.8
176	4.77	8.8
177	4.83	9.6
178	4.85	6.6
179	4.85	7.9
180	4.93	6.3
181	4.98	13.1
182	4.99	7.8
183	5.01	8.7
184	5.04	9.2
185	5.08	9.2
186	5.10	8.4
187	5.14	10.5
188	5.18	8.6
189	5.26	8.7
190	5.26	8.0
191	5.28	11.1
192	5.29	7.5
193	5.29	7.6
194	5.35	8.7

195	5.67	8.8
196	5.79	11.4
197	5.87	6.6
198	5.92	10.6
199	6.04	5.5
200	6.05	8.7
201	6.12	6.8
202	6.15	9.7
203	6.18	6.8
204	6.19	5.7
205	6.20	8.5
206	6.20	5.5
207	6.20	8.4
208	6.25	8.7
209	6.28	9.8
210	6.31	7.7
211	6.33	6.8
212	6.34	6.6
213	6.37	7.0
214	6.43	6.3
215	6.59	6.8
216	6.61	8.8
217	6.73	10.3
218	6.90	8.0
219	7.07	1.8
220	7.35	13.4
221	7.38	6.8
222	7.44	5.7
223	7.55	7.1
224	7.57	6.7
225	7.62	6.5
226	7.62	7.7
227	7.79	5.8
228	7.80	6.4
229	7.95	6.5
230	7.95	4.1
231	7.95	8.0
232	8.03	6.8
233	8.04	6.4
234	8.17	8.3
235	8.24	7.3
236	8.28	5.6
237	8.35	5.1
238	8.36	6.9
239	8.40	7.7
240	8.46	5.4
241	8.73	5.9
242	8.81	5.0

```
243  9.01  5.2
244  9.05  6.1
245  9.28  6.3
246  9.28  6.3
247  9.36  7.1
248  9.55  7.4
249  9.86  5.0
250 10.00  5.9
251 10.02  6.0
252 10.09  4.8
253 10.10  9.9
254 10.12  5.4
255 10.15  5.0
256 10.28  6.1
257 10.40 13.8
258 10.49  6.2
259 10.63  5.5
260 10.70  5.5
261 10.81 11.7
262 11.15  5.9
263 11.45  9.8
264 11.47  6.1
265 11.55  7.5
266 11.78  4.1
267 11.85  7.7
268 11.90  8.1
269 11.98  4.5
270 12.15  5.8
271 12.23  5.4
272 12.32  5.7
273 12.39  3.1
274 12.46  6.4
275 12.53  7.0
276 12.75  5.7
277 12.80  3.9
278 12.88  9.4
279 12.89  4.4
280 12.97  5.0
281 13.13 15.9
282 13.40  3.7
283 13.42  9.1
284 13.73  4.7
285 13.73  3.6
286 13.92  3.7
287 13.97  4.1
288 14.12  7.9
289 14.22  3.3
290 14.61  6.6
```

```
291 14.64  1.9
292 14.85  3.0
293 14.88  5.7
294 14.92  3.2
295 14.95  3.8
296 14.96  5.3
297 15.06  3.2
298 15.15  4.2
299 15.55  6.0
300 15.72  9.7
301 15.86  3.4
302 15.87  3.2
303 15.91  2.5
304 16.01  2.0
305 16.12  4.0
306 16.23  4.3
307 16.60  2.8
308 16.74  2.2
309 16.78  4.7
310 17.08  2.5
311 17.14  2.2
312 17.23  2.2
313 17.30  1.9
314 17.67  9.3
```

Because a dataframe is a matrix, we can ask R to return columns, rows, etc. Below i ask R for the second column

[67]:
```
d[,2]
```

1. 23 2. 23.8 3. 16.9 4. 18.6 5. 17.9 6. 25.9 7. 16.5 8. 26.3 9. 26.9 10. 17.9 11. 29.1 12. 32.6 13. 41.9 14. 38.7 15. 37.6 16. 46.4 17. 30.5 18. 34.8 19. 26.7 20. 33 21. 56.3 22. 33.5 23. 29.7 24. 32.8 25. 27.6 26. 31.9 27. 55.4 28. 33.3 29. 36.4 30. 25.9 31. 21.6 32. 26 33. 12.8 34. 34.2 35. 19.4 36. 21.9 37. 19.3 38. 26.2 39. 9.6 40. 36.1 41. 19.7 42. 19.2 43. 20.4 44. 23.1 45. 23.8 46. 21.6 47. 25.4 48. 18.4 49. 18.6 50. 26.4 51. 25.7 52. 20.4 53. 15.9 54. 20.8 55. 20.9 56. 15.7 57. 20.4 58. 14.1 59. 21.4 60. 15.6 61. 30.5 62. 15.4 63. 19.2 64. 22.6 65. 15.2 66. 14.9 67. 39.5 68. 17.3 69. 17.2 70. 13.8 71. 17.7 72. 15.6 73. 20.4 74. 14.8 75. 19.5 76. 18 77. 22.5 78. 20.4 79. 13.1 80. 17.4 81. 18.9 82. 16.1 83. 21.9 84. 18.4 85. 15.2 86. 19.1 87. 15.1 88. 17.2 89. 18.4 90. 24.1 91. 19 92. 16 93. 21.8 94. 16.5 95. 16.2 96. 10.8 97. 27 98. 17.3 99. 18.3 100. 18.3 101. 13.3 102. 12.1 103. 11 104. 12.7 105. 13.4 106. 14.3 107. 11.5 108. 16.5 109. 25.1 110. 16.3 111. 15.4 112. 18.6 113. 18.3 114. 13.2 115. 11.1 116. 11 117. 12.5 118. 14.8 119. 16.8 120. 12.8 121. 12.2 122. 14.7 123. 9.7 124. 14.2 125. 13.9 126. 9.5 127. 14.6 128. 12.1 129. 10.6 130. 10.8 131. 10.4 132. 9.9 133. 9.6 134. 21.7 135. 11.8 136. 11.8 137. 13 138. 11.3 139. 9 140. 18.3 141. 14.6 142. 11.5 143. 7.3 144. 11.6 145. 8.6 146. 11.5 147. 8.3 148. 10.3 149. 12.5 150. 13.2 151. 17.9 152. 10.7 153. 8.6 154. 12.5 155. 8.6 156. 6 157. 12.6 158. 12.7 159. 10.2 160. 10.7 161. 8.4 162. 9.9 163. 7.7 164. 13.6 165. 12.3 166. 7.5 167. 11.8 168. 7.7 169. 10.3 170. 16.9 171. 8.5 172. 9.7 173. 10.6 174. 6.4 175. 8.8 176. 8.8 177. 9.6 178. 6.6 179. 7.9 180. 6.3 181. 13.1 182. 7.8 183. 8.7 184. 9.2 185. 9.2 186. 8.4 187. 10.5 188. 8.6 189. 8.7 190. 8 191. 11.1 192. 7.5 193. 7.6 194. 8.7 195. 8.8 196. 11.4 197. 6.6 198. 10.6 199. 5.5 200. 8.7 201. 6.8 202. 9.7 203. 6.8 204. 5.7 205. 8.5 206. 5.5 207. 8.4 208. 8.7 209. 9.8 210. 7.7 211. 6.8 212. 6.6 213. 7 214. 6.3

215. 6.8 216. 8.8 217. 10.3 218. 8 219. 1.8 220. 13.4 221. 6.8 222. 5.7 223. 7.1 224. 6.7 225. 6.5 226. 7.7 227. 5.8 228. 6.4 229. 6.5 230. 4.1 231. 8 232. 6.8 233. 6.4 234. 8.3 235. 7.3 236. 5.6 237. 5.1 238. 6.9 239. 7.7 240. 5.4 241. 5.9 242. 5 243. 5.2 244. 6.1 245. 6.3 246. 6.3 247. 7.1 248. 7.4 249. 5 250. 5.9 251. 6 252. 4.8 253. 9.9 254. 5.4 255. 5 256. 6.1 257. 13.8 258. 6.2 259. 5.5 260. 5.5 261. 11.7 262. 5.9 263. 9.8 264. 6.1 265. 7.5 266. 4.1 267. 7.7 268. 8.1 269. 4.5 270. 5.8 271. 5.4 272. 5.7 273. 3.1 274. 6.4 275. 7 276. 5.7 277. 3.9 278. 9.4 279. 4.4 280. 5 281. 15.9 282. 3.7 283. 9.1 284. 4.7 285. 3.6 286. 3.7 287. 4.1 288. 7.9 289. 3.3 290. 6.6 291. 1.9 292. 3 293. 5.7 294. 3.2 295. 3.8 296. 5.3 297. 3.2 298. 4.2 299. 6 300. 9.7 301. 3.4 302. 3.2 303. 2.5 304. 2 305. 4 306. 4.3 307. 2.8 308. 2.2 309. 4.7 310. 2.5 311. 2.2 312. 2.2 313. 1.9 314. 9.3

Because R considers this objects a dataframe, there are column names for each column. We can ask R for the column names like this

[68]:
```
colnames(d)
```

1. 'Age' 2. 'GAG'

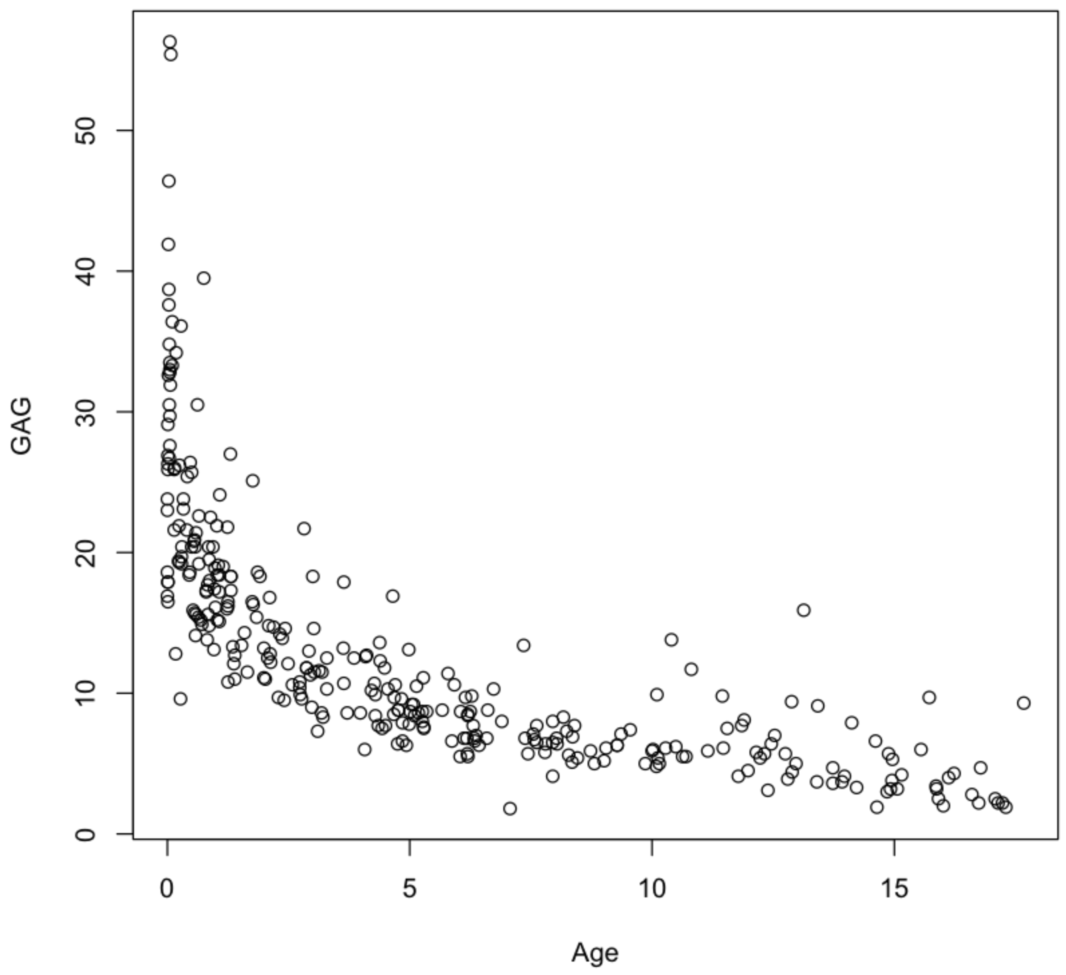
and refer to columns by typing the name of the dataframe, a dollar sign, and the name of the column we want

[69]:
```
d$GAG
```

1. 23 2. 23.8 3. 16.9 4. 18.6 5. 17.9 6. 25.9 7. 16.5 8. 26.3 9. 26.9 10. 17.9 11. 29.1 12. 32.6 13. 41.9 14. 38.7 15. 37.6 16. 46.4 17. 30.5 18. 34.8 19. 26.7 20. 33 21. 56.3 22. 33.5 23. 29.7 24. 32.8 25. 27.6 26. 31.9 27. 55.4 28. 33.3 29. 36.4 30. 25.9 31. 21.6 32. 26 33. 12.8 34. 34.2 35. 19.4 36. 21.9 37. 19.3 38. 26.2 39. 9.6 40. 36.1 41. 19.7 42. 19.2 43. 20.4 44. 23.1 45. 23.8 46. 21.6 47. 25.4 48. 18.4 49. 18.6 50. 26.4 51. 25.7 52. 20.4 53. 15.9 54. 20.8 55. 20.9 56. 15.7 57. 20.4 58. 14.1 59. 21.4 60. 15.6 61. 30.5 62. 15.4 63. 19.2 64. 22.6 65. 15.2 66. 14.9 67. 39.5 68. 17.3 69. 17.2 70. 13.8 71. 17.7 72. 15.6 73. 20.4 74. 14.8 75. 19.5 76. 18 77. 22.5 78. 20.4 79. 13.1 80. 17.4 81. 18.9 82. 16.1 83. 21.9 84. 18.4 85. 15.2 86. 19.1 87. 15.1 88. 17.2 89. 18.4 90. 24.1 91. 19 92. 16 93. 21.8 94. 16.5 95. 16.2 96. 10.8 97. 27 98. 17.3 99. 18.3 100. 18.3 101. 13.3 102. 12.1 103. 11 104. 12.7 105. 13.4 106. 14.3 107. 11.5 108. 16.5 109. 25.1 110. 16.3 111. 15.4 112. 18.6 113. 18.3 114. 13.2 115. 11.1 116. 11 117. 12.5 118. 14.8 119. 16.8 120. 12.8 121. 12.2 122. 14.7 123. 9.7 124. 14.2 125. 13.9 126. 9.5 127. 14.6 128. 12.1 129. 10.6 130. 10.8 131. 10.4 132. 9.9 133. 9.6 134. 21.7 135. 11.8 136. 11.8 137. 13 138. 11.3 139. 9 140. 18.3 141. 14.6 142. 11.5 143. 7.3 144. 11.6 145. 8.6 146. 11.5 147. 8.3 148. 10.3 149. 12.5 150. 13.2 151. 17.9 152. 10.7 153. 8.6 154. 12.5 155. 8.6 156. 6 157. 12.6 158. 12.7 159. 10.2 160. 10.7 161. 8.4 162. 9.9 163. 7.7 164. 13.6 165. 12.3 166. 7.5 167. 11.8 168. 7.7 169. 10.3 170. 16.9 171. 8.5 172. 9.7 173. 10.6 174. 6.4 175. 8.8 176. 8.8 177. 9.6 178. 6.6 179. 7.9 180. 6.3 181. 13.1 182. 7.8 183. 8.7 184. 9.2 185. 9.2 186. 8.4 187. 10.5 188. 8.6 189. 8.7 190. 8 191. 11.1 192. 7.5 193. 7.6 194. 8.7 195. 8.8 196. 11.4 197. 6.6 198. 10.6 199. 5.5 200. 8.7 201. 6.8 202. 9.7 203. 6.8 204. 5.7 205. 8.5 206. 5.5 207. 8.4 208. 8.7 209. 9.8 210. 7.7 211. 6.8 212. 6.6 213. 7 214. 6.3 215. 6.8 216. 8.8 217. 10.3 218. 8 219. 1.8 220. 13.4 221. 6.8 222. 5.7 223. 7.1 224. 6.7 225. 6.5 226. 7.7 227. 5.8 228. 6.4 229. 6.5 230. 4.1 231. 8 232. 6.8 233. 6.4 234. 8.3 235. 7.3 236. 5.6 237. 5.1 238. 6.9 239. 7.7 240. 5.4 241. 5.9 242. 5 243. 5.2 244. 6.1 245. 6.3 246. 6.3 247. 7.1 248. 7.4 249. 5 250. 5.9 251. 6 252. 4.8 253. 9.9 254. 5.4 255. 5 256. 6.1 257. 13.8 258. 6.2 259. 5.5 260. 5.5 261. 11.7 262. 5.9 263. 9.8 264. 6.1 265. 7.5 266. 4.1 267. 7.7 268. 8.1 269. 4.5 270. 5.8 271. 5.4 272. 5.7 273. 3.1 274. 6.4 275. 7 276. 5.7 277. 3.9 278. 9.4 279. 4.4 280. 5 281. 15.9 282. 3.7 283. 9.1 284. 4.7 285. 3.6 286. 3.7 287. 4.1 288. 7.9 289. 3.3 290. 6.6 291. 1.9 292. 3 293. 5.7 294. 3.2 295. 3.8 296. 5.3 297. 3.2 298. 4.2 299. 6 300. 9.7 301. 3.4 302. 3.2 303. 2.5 304. 2 305. 4 306. 4.3 307. 2.8 308. 2.2 309. 4.7 310. 2.5 311. 2.2 312. 2.2 313. 1.9 314. 9.3

Dataframes in R have a lot of built-in functionality. For example, we can call plot on this dataframe to quickly make a scatter plot of age versus GaG level.

```
[72]: plot(d)
```



0.1.6 The For loop

Just as in Python, R has a for loop. A for loop is built in R using the

```
for( <variable> in <sequence of values> ){
    code
}
```

construct. The simplest example is printing out the numbers from 1 to 10.

[91]:
```
for(i in 1:10){
    print(i)
}
```

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
```

The loop assigns the variable i to each number from 1 to 10 and then executes all the code placed inside the brackets.

Sequences in R In R, a number a and number b with a colon inbetween is interpreted as a list of numbers starting from a, and incrementing by 1 until b (inclusive).

[92]:
```
3:13
```

1. 3 2. 4 3. 5 4. 6 5. 7 6. 8 7. 9 8. 10 9. 11 10. 12 11. 13

R also has the seq function that allows you to enter a start and end number

[93]:
```
seq(3,13)
```

1. 3 2. 4 3. 5 4. 6 5. 7 6. 8 7. 9 8. 10 9. 11 10. 12 11. 13

as well as a third value, telling R what to increment by

[94]:
```
seq(3,13,2)
```

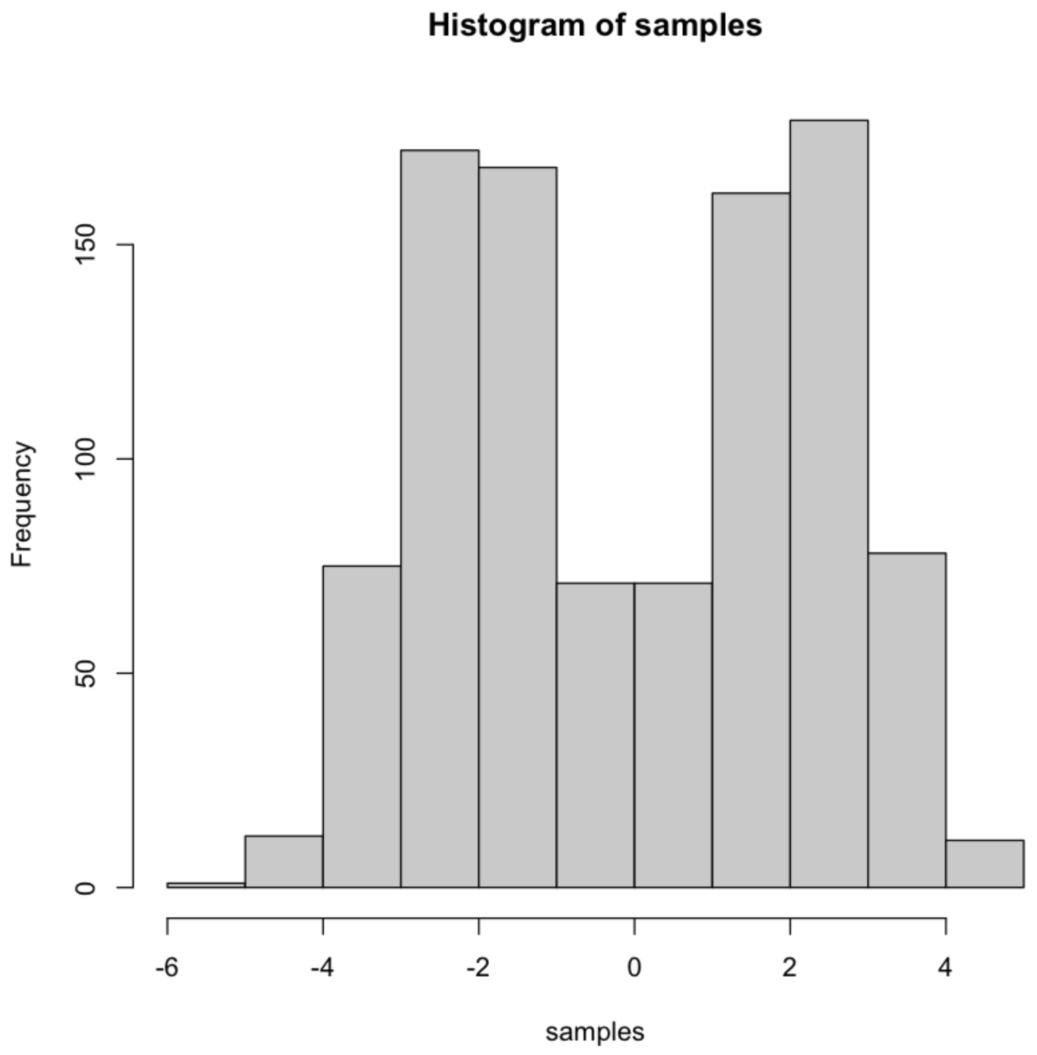
1. 3 2. 5 3. 7 4. 9 5. 11 6. 13

0.1.7 Conditionals and flow control

R has logic just as python does, allowing us to execute code if an expression evaluates to TRUE, else execute code when that expression evaluates to FALSE. TRUE and FALSE are special, reserved words in R? TRUE/FALSE when put inside an if/else statement in R, tells R which block of code to execute.

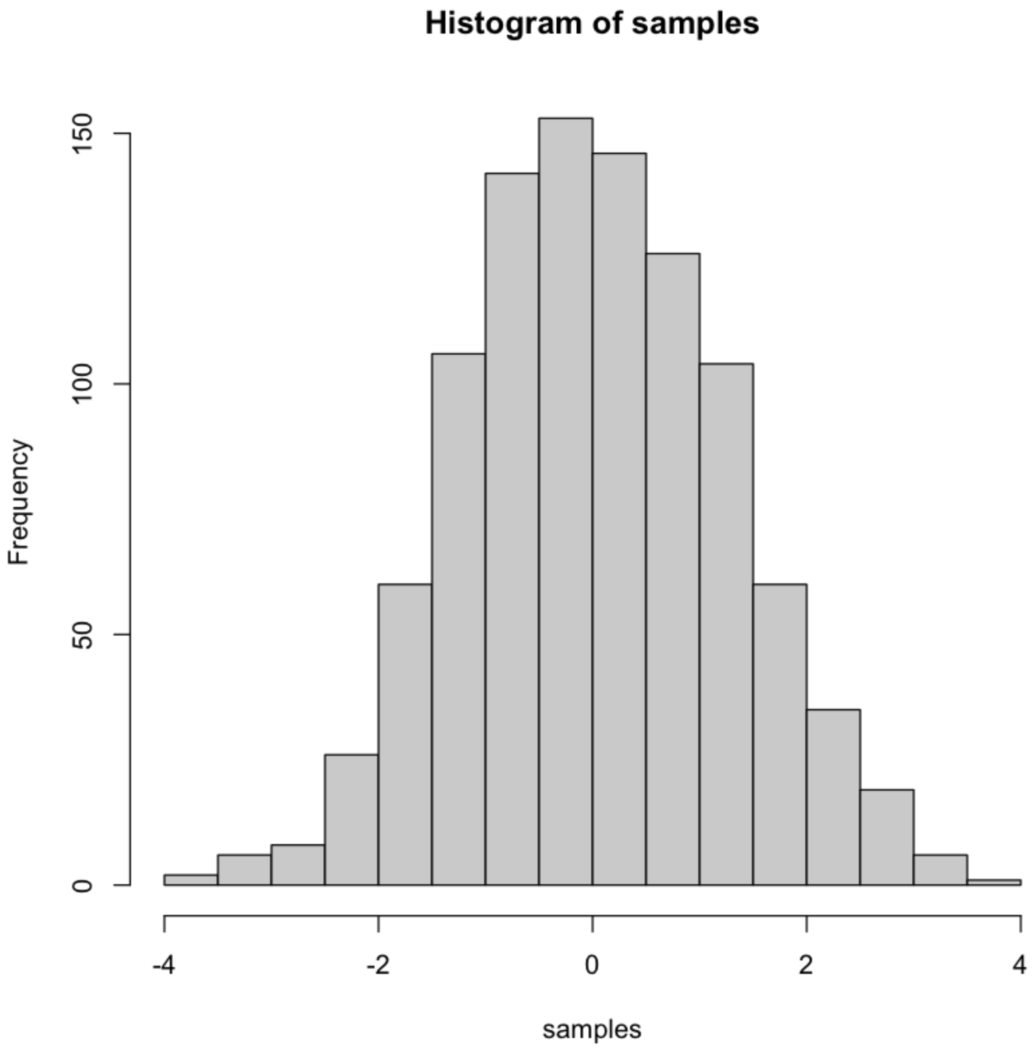
To demonstrate how to use the if/else and if/else if/else syntax, let's first simulate one of two random variables. We will run 1000 simulations and for each simulation flip a coin. If the coin is heads the draw a sample from a $\mathcal{N}(1,1)$ distribution and if the coin is tails then draw a sample from a $\mathcal{N}(-1,1)$ distribution. To simulate a coin flip, we can simulate a bernoulli random variable, treating the value 1 as heads and the value 0 as tails.

[14]:
```
samples = rep(0,10^3) # make an empty vector of 1,000 zeros that we will use to␣
↪store our 1,000 random variable samples.
for (i in 1:10^(3)){
    coinflip = rbinom(n=1,p=0.5,size=1)
    if (coinflip==1){
        sample = rnorm(n=1,mean=2,sd=1) # rnorm generates a sample at random and␣
↪according to the specified Normal dist.
    }else{
        sample = rnorm(n=1,mean=-2,sd=1)
    }
    samples[i]=sample
}
hist(samples) # This is how to build a histogram from a vector of values in R
```



Let's modify the above code to add a third normal distirbution. We can draw a 0,1, or 2 from a binomial distribution. If the value we draw is 0 then we'll sample from $\mathcal{N}(-1,1)$ distribution. If the value we draw is 1 then we'll sample from $\mathcal{N}(0,1)$ distribution, and if 2 we draw from a $\mathcal{N}(1,1)$ distribution.

```
[23]: samples = rep(0,10^3) # make an empty vector of 1,000 zeros that we will use to␣
       ↪store our 1,000 random variable samples.
      for (i in 1:10^(3)){
          threeOptions = rbinom(n=1,p=0.5,size=2)
          if (threeOptions==0){
              # rnorm generates a sample at random and according to the specified␣
       ↪Normal dist.
              sample = rnorm(n=1,mean=-1,sd=1)
          }else if (threeOptions == 1){ #< ELSE if statement!
              sample = rnorm(n=1,mean=0,sd=1)
          }else{
              sample = rnorm(n=1,mean=1,sd=1)
          }
          samples[i]=sample
      }
      hist(samples) # This is how to build a histogram from a vector of values in R
```



Histogram of samples
Frequency
0
50
100
150
-4
-2
0
2
4
samples