

## Worksheet-1 in R

### Worksheet for R Programming

#### Instructions:

- Use RStudio or the RStudio Cloud accomplish this worksheet. + Save the R script as *RWorksheet\_lastname#1.R*.
- Create your own *GitHub repository* and push the R script as well as this pdf worksheet to your own repo.

Accomplish this worksheet by answering the questions being asked and writing the code manually.

#### Using functions:

`seq()`, `assign()`, `min()`, `max()`, `c()`, `sort()`, `sum()`, `filter()`

1. Set up a vector named age, consisting of 34, 28, 22, 36, 27, 18, 52, 39, 42, 29, 35, 31, 27, 22, 37, 34, 19, 20, 57, 49, 50, 37, 46, 25, 17, 37, 42, 53, 41, 51, 35, 24, 33, 41.

a. How many data points?

There are 34 data points.

b. Write the R code and its output.

R code:

```
age <- c(34, 28, 22, 36, 27, 18, 52, 39, 42, 29, 35, 31, 27, 22, 37, 34, 19, 20, 57, 49, 50, 37, 46, 25, 17, 37, 42, 53, 41, 51, 35, 24, 33, 41)
age
```

Output:

```
[1] 34 28 22 36 27 18 52 39 42 29 35 31 27 22 37 34 19 20 57 49 50
[22] 37 46 25 17 37 42 53 41 51 35 24 33 41
```

The output is the values assigned to the vector 'age'.

2. Find the reciprocal of the values for age.

Write the R code and its output.

R code continuation:

```
library(MASS)
reciprocal_number <- 1/age
reciprocal_number
```

Output:

```
[1] 0.02941176 0.03571429 0.04545455 0.02777778 0.03703704
[6] 0.05555556 0.01923077 0.02564103 0.02380952 0.03448276
[11] 0.02857143 0.03225806 0.03703704 0.04545455 0.02702703
[16] 0.02941176 0.05263158 0.05000000 0.01754386 0.02040816
[21] 0.02000000 0.02702703 0.02173913 0.04000000 0.05882353
[26] 0.02702703 0.02380952 0.01886792 0.02439024 0.01960784
[31] 0.02857143 0.04166667 0.03030303 0.02439024
```

The output produces the reciprocal number of each values in the vector age.

3. Assign also `new_age <- c(age, 0, age)`.

What happen to the `new_age`?

Answer: The output in `new_age` produces a combination of values; first the values of `age` followed by 0, then followed by the values of `age` again.

4. Sort the values for `age`.

Write the R code and its output.

R code continuation:

```
sort(age)
```

Output:

```
[1] 17 18 19 20 22 22 24 25 27 27 28 29 31 33 34 34 35 35 36 37 37
```

```
[22] 37 39 41 41 42 42 46 49 50 51 52 53 57
```

The output is the values of `age` in ascending order.

5. Find the minimum and maximum value for `age`.

Write the R code and its output.

R code continuation:

```
min(age)
```

```
max(age)
```

Output:

```
[1] 17
```

```
[1] 57
```

17 is the output for the minimum value and 57 for maximum value for `age`.

6. Set up a vector named data, consisting of 2.4, 2.8, 2.1, 2.5, 2.4, 2.2, 2.5, 2.3, 2.5, 2.3, 2.4, and 2.7.

a. How many data points?

There are 12 data points.

b. Write the R code and its output.

R code:

```
data <- c(2.4, 2.8, 2.1, 2.5, 2.4, 2.2, 2.5, 2.3, 2.5, 2.3, 2.4, 2.7)
data
length(data)
```

Output:

```
data <- c(2.4, 2.8, 2.1, 2.5, 2.4, 2.2, 2.5, 2.3, 2.5, 2.3, 2.4, 2.7)
[1] 2.4 2.8 2.1 2.5 2.4 2.2 2.5 2.3 2.5 2.3 2.4 2.7
> length(data)
[1] 12
```

7. Generates a new vector for data where you double every value of the data. | What happen to the data?

Rcode:

```
double_value <- data*2
double_value
```

When I assigned a new vector named 'double\_value' the product of the number two(2) and data, the values of the data doubled when I display the output of the new vector because each value is times to two.

8. Generate a sequence for the following scenario:

8.1 Integers from 1 to 100.

R code

```
1:100
```

Output:

```
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
[16] 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
[31] 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
[46] 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
[61] 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75
[76] 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90
[91] 91 92 93 94 95 96 97 98 99 100
```

8.2 Numbers from 20 to 60

R code:

```
seq(20,60)
```

Output:

```
[1] 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
[22] 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
```

\*8.3 Mean of numbers from 20 to 60

R code:

```
twoSixty <- 20:60
twoSixty
mean((twoSixty))
```

Output:

```
[1] 40
```

\*8.4 Sum of numbers from 51 to 91

R code:

```
Total <- 51:91
Total
sum((Total))
```

Output:

```
[1] 2911
```

# \*8.5 Integers from 1 to 1,000

R code:

```
integer <- seq(1, 1000)
integer
```

Output:

```
[1] 1 2 3 4 5 6 7 8 9 10 11 12
[13] 13 14 15 16 17 18 19 20 21 22 23 24
[25] 25 26 27 28 29 30 31 32 33 34 35 36
[37] 37 38 39 40 41 42 43 44 45 46 47 48
[49] 49 50 51 52 53 54 55 56 57 58 59 60
[61] 61 62 63 64 65 66 67 68 69 70 71 72
[73] 73 74 75 76 77 78 79 80 81 82 83 84
[85] 85 86 87 88 89 90 91 92 93 94 95 96
[97] 97 98 99 100 101 102 103 104 105 106 107 108
[109] 109 110 111 112 113 114 115 116 117 118 119 120
[121] 121 122 123 124 125 126 127 128 129 130 131 132
[133] 133 134 135 136 137 138 139 140 141 142 143 144
[145] 145 146 147 148 149 150 151 152 153 154 155 156
[157] 157 158 159 160 161 162 163 164 165 166 167 168
[169] 169 170 171 172 173 174 175 176 177 178 179 180
[181] 181 182 183 184 185 186 187 188 189 190 191 192
[193] 193 194 195 196 197 198 199 200 201 202 203 204
[205] 205 206 207 208 209 210 211 212 213 214 215 216
[217] 217 218 219 220 221 222 223 224 225 226 227 228
[229] 229 230 231 232 233 234 235 236 237 238 239 240
[241] 241 242 243 244 245 246 247 248 249 250 251 252
[253] 253 254 255 256 257 258 259 260 261 262 263 264
[265] 265 266 267 268 269 270 271 272 273 274 275 276
[277] 277 278 279 280 281 282 283 284 285 286 287 288
[289] 289 290 291 292 293 294 295 296 297 298 299 300
[301] 301 302 303 304 305 306 307 308 309 310 311 312
[313] 313 314 315 316 317 318 319 320 321 322 323 324
[325] 325 326 327 328 329 330 331 332 333 334 335 336
[337] 337 338 339 340 341 342 343 344 345 346 347 348
[349] 349 350 351 352 353 354 355 356 357 358 359 360
[361] 361 362 363 364 365 366 367 368 369 370 371 372
[373] 373 374 375 376 377 378 379 380 381 382 383 384
[385] 385 386 387 388 389 390 391 392 393 394 395 396
[397] 397 398 399 400 401 402 403 404 405 406 407 408
[409] 409 410 411 412 413 414 415 416 417 418 419 420
[421] 421 422 423 424 425 426 427 428 429 430 431 432
[433] 433 434 435 436 437 438 439 440 441 442 443 444
[445] 445 446 447 448 449 450 451 452 453 454 455 456
[457] 457 458 459 460 461 462 463 464 465 466 467 468
[469] 469 470 471 472 473 474 475 476 477 478 479 480
[481] 481 482 483 484 485 486 487 488 489 490 491 492
```

[493]	493	494	495	496	497	498	499	500	501	502	503	504
[505]	505	506	507	508	509	510	511	512	513	514	515	516
[517]	517	518	519	520	521	522	523	524	525	526	527	528
[529]	529	530	531	532	533	534	535	536	537	538	539	540
[541]	541	542	543	544	545	546	547	548	549	550	551	552
[553]	553	554	555	556	557	558	559	560	561	562	563	564
[565]	565	566	567	568	569	570	571	572	573	574	575	576
[577]	577	578	579	580	581	582	583	584	585	586	587	588
[589]	589	590	591	592	593	594	595	596	597	598	599	600
[601]	601	602	603	604	605	606	607	608	609	610	611	612
[613]	613	614	615	616	617	618	619	620	621	622	623	624
[625]	625	626	627	628	629	630	631	632	633	634	635	636
[637]	637	638	639	640	641	642	643	644	645	646	647	648
[649]	649	650	651	652	653	654	655	656	657	658	659	660
[661]	661	662	663	664	665	666	667	668	669	670	671	672
[673]	673	674	675	676	677	678	679	680	681	682	683	684
[685]	685	686	687	688	689	690	691	692	693	694	695	696
[697]	697	698	699	700	701	702	703	704	705	706	707	708
[709]	709	710	711	712	713	714	715	716	717	718	719	720
[721]	721	722	723	724	725	726	727	728	729	730	731	732
[733]	733	734	735	736	737	738	739	740	741	742	743	744
[745]	745	746	747	748	749	750	751	752	753	754	755	756
[757]	757	758	759	760	761	762	763	764	765	766	767	768
[769]	769	770	771	772	773	774	775	776	777	778	779	780
[781]	781	782	783	784	785	786	787	788	789	790	791	792
[793]	793	794	795	796	797	798	799	800	801	802	803	804
[805]	805	806	807	808	809	810	811	812	813	814	815	816
[817]	817	818	819	820	821	822	823	824	825	826	827	828
[829]	829	830	831	832	833	834	835	836	837	838	839	840
[841]	841	842	843	844	845	846	847	848	849	850	851	852
[853]	853	854	855	856	857	858	859	860	861	862	863	864
[865]	865	866	867	868	869	870	871	872	873	874	875	876
[877]	877	878	879	880	881	882	883	884	885	886	887	888
[889]	889	890	891	892	893	894	895	896	897	898	899	900
[901]	901	902	903	904	905	906	907	908	909	910	911	912
[913]	913	914	915	916	917	918	919	920	921	922	923	924
[925]	925	926	927	928	929	930	931	932	933	934	935	936
[937]	937	938	939	940	941	942	943	944	945	946	947	948
[949]	949	950	951	952	953	954	955	956	957	958	959	960
[961]	961	962	963	964	965	966	967	968	969	970	971	972
[973]	973	974	975	976	977	978	979	980	981	982	983	984
[985]	985	986	987	988	989	990	991	992	993	994	995	996
[997]	997	998	999	1000								

a. How many data points from 8.1 to 8.4?

There are 143 data points

b. Write the R code and its output from 8.1 to 8.4.

R code:

8.1

```
data.frame(1:100)
```

8.2

```
data.frame(20:60)
```

8.3

```
data.frame(mean(20:60))
```

8.4

```
data.frame(sum(51:91))
```

Output:

8.1

```
> data.frame(1:100)
```

X1.100

1 1

2 2

3 3

4 4

5 5

6 6

7 7

8 8

9 9

10 10

11 11

12 12

13 13



14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21
22	22
23	23
24	24
25	25
26	26
27	27
28	28
29	29
30	30
31	31
32	32
33	33
34	34
35	35
36	36
37	37
38	38
39	39
40	40
41	41
42	42
43	43
44	44

45 45

46 46

47 47

48 48

49 49

50 50

51 51

52 52

53 53

54 54

55 55

56 56

57 57

58 58

59 59

60 60

61 61

62 62

63 63

64 64

65 65

66 66

67 67

68 68

69 69

70 70

71 71

72 72

73 73

74 74

75 75

76	76
77	77
78	78
79	79
80	80
81	81
82	82
83	83
84	84
85	85
86	86
87	87
88	88
89	89
90	90
91	91
92	92
93	93
94	94
95	95
96	96
97	97
98	98
99	99

**100** 100 8.1 is total of 100 data points

8.2

```
> data.frame(20:60)
```

X20.60

1 20

2 21

3	22
4	23
5	24
6	25
7	26
8	27
9	28
10	29
11	30
12	31
13	32
14	33
15	34
16	35
17	36
18	37
19	38
20	39
21	40
22	41
23	42
24	43
25	44
26	45
27	46
28	47
29	48
30	49
31	50
32	51
33	52

```

34  53
35  54
36  55
37  56
38  57
39  58
40  59
41 60    8.2 is total of 41 data points

```

8.3

```
data.frame(sum(51:91))
```

```
sum.51.91.
```

```
1    2911    8.3 is total of 1 data point
```

8.4

```
data.frame(mean(51:91))
```

```
data.frame(mean(51:91))
```

```
mean.51.91.
```

```
1      71    8.4 is total of 1 data point
```

```
#data points from 8.1 to 8.4 is total to 143 data points.
```

c. For 8.5 find only maximum data points until 10.

```
max(1:10)
```

9. \*Print a vector with the integers between 1 and 100 that are not divisible by 3, 5 and 7 using filter option.

```
filter(function(i) { all(i %% c(3,5,7) != 0) }, seq(100))
```

Write the R code and its output.

R code:

```
filter(function(i) { all(i %% c(3,5,7) != 0) } seq(100))
```

Output:

```
Error: unexpected symbol in "filter(function(i) { all(i %% c(3,5,7) != 0) } seq"
```

**If the ‘f’ in filter is capitalized though the code works, that is**

R code:

```
Filter(function(i) { all(i %% c(3,5,7) != 0) }, seq(100))
```

Output:

```
[1] 1 2 4 8 11 13 16 17 19 22 23 26 29 31 32 34 37 38 41 43 44  
[22] 46 47 52 53 58 59 61 62 64 67 68 71 73 74 76 79 82 83 86 88 89  
[43] 92 94 97
```

10. Generate a sequence backwards of the integers from 1 to 100.

Write the R code and its output.

R code:

```
backwards <- 100:1
```

```
backwards
```

Output:

```
backwards <- 100:1
```

```
> backwards
```

```
[1] 100 99 98 97 96 95 94 93 92 91 90 89 88 87 86
```

```
[16] 85 84 83 82 81 80 79 78 77 76 75 74 73 72 71
```

[31] 70 69 68 67 66 65 64 63 62 61 60 59 58 57 56

[46] 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41

[61] 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26

[76] 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11

[91] 10 9 8 7 6 5 4 3 2 1

11. List all the natural numbers below 25 that are multiples of 3 or 5.

Find the sum of these multiples.

R code:

```
sum((1 : 25)[((1 : 25)%%3 == 0) | ((1:25)%%5 == 0)])
```

Output: [1] 168

- a. How many data points from 10 to 11?

There are **101 data points** from items 10 to 11.

- b. Write the R code and its output from 10 and 11.

Rcode in item 10

```
data.frame(100:1)
```

Output in item 10

```
data.frame(100:1)
```

X100.1

1	100
2	99
3	98
4	97
5	96
6	95
7	94
8	93
9	92
10	91
11	90
12	89

13	88
14	87
15	86
16	85
17	84
18	83
19	82
20	81
21	80
22	79
23	78
24	77
25	76
26	75
27	74
28	73
29	72
30	71
31	70
32	69
33	68
34	67
35	66
36	65
37	64
38	63
39	62
40	61
41	60
42	59
43	58
44	57
45	56
46	55
47	54
48	53
49	52
50	51
51	50
52	49
53	48
54	47
55	46
56	45
57	44
58	43
59	42
60	41



61	40
62	39
63	38
64	37
65	36
66	35
67	34
68	33
69	32
70	31
71	30
72	29
73	28
74	27
75	26
76	25
77	24
78	23
79	22
80	21
81	20
82	19
83	18
84	17
85	16
86	15
87	14
88	13
89	12
90	11
91	10
92	9
93	8
94	7
95	6
96	5
97	4
98	3
99	2
<b>100</b>	<b>1</b> 100 data points

R code in item 11

```
data.frame(sum((1 : 25)[((1 : 25)%%3 == 0) | ((1:25)%%5 == 0)]))
```

Output in item 11

```
sum..1.25....1.25...3....0.....1.25...5....0...
```

```
1 168 1 data point
```

12. Statements can be grouped together using braces '{' and '}'. A group of statements is sometimes called a **block**. Single statements are evaluated when a new line is typed at the end of the syntactically complete statement. Blocks are not evaluated until a new line is entered after the closing brace.

Enter this statement:

```
{ x <- 0+ x + 5 + }
```

Describe the output.

The output in this item produced an error. It says that there is an unexpected brace ( } ) in the statement. But if you remove the brace in the end or add a number after the plus there is no error prompt in the code.

13. \*Set up a vector named score, consisting of 72, 86, 92, 63, 88, 89, 91, 92, 75, 75 and 77. To access individual elements of an atomic vector, one generally uses the x[i] construction.

Find x[2] and x[3]. Write the R code and its output.

R code:

```
score <- c(72, 86, 92, 63, 88, 89, 91, 92, 75, 75, 77)
score[2]
score[3]
```

Output:

```
score <- c(72, 86, 92, 63, 88, 89, 91, 92, 75, 75, 77)
> score[2]
[1] 86
> score[3]
[1] 92
```

The output produce the individual elements that we are finding, that is, the second element which is the score 86, and the third element that is the score 92.

14. \*Create a vector a = c(1,2,NA,4,NA,6,7).

a. Change the NA to 999 using the codes print(a,na.print="-999").

b. Write the R code and its output. Describe the output.

R code:

```
a <- c(1,2,NA,4,NA,6,7)
a
print(a,na.print="-999")
```

Output:

```
a <- c(1,2,NA,4,NA,6,7)
> a
[1] 1 2 NA 4 NA 6 7
> print(a,na.print="-999")
[1] 1 2 -999 4 -999 6 7
```

In the output all the two elements that is 'NA' changed their value to '-999'.

15. A special type of function calls can appear on the left hand side of the assignment operator as in `> class(x) <- "foo"`.

Follow the codes below:

```
name = readline(prompt="Input your name: ")
age = readline(prompt="Input your age: ")
print(paste("My name is",name, "and I am",age ,"years old. "))
print(R.version.string)
```

What is the output of the above code?

```
> name = readline(prompt="Input your name: ")
Input your name: Grace Anne Capanang
> age = readline(prompt="Input your age: ")
Input your age: 19
> print(paste("My name is",name, "and I am",age ,"years old. "))
[1] "My name is Grace Anne Capanang and I am 19 years old."
> print(R.version.string)
[1] "R version 4.2.1 (2022-06-23 ucrt)"
```