RWorksheet_Llanera#1

LlaneraExerRepo

2024-09-04

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

1. Vector Age

```
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.)
length(age)
```

```
## [1] 34
```

2.Values for Age

```
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.)
reciprocal_age <- 1/ age
print(reciprocal_age)
```

```
##  [1] 0.02941176 0.03571429 0.04545455 0.02777778 0.03703704 0.05555556
##  [7] 0.01923077 0.02564103 0.02380952 0.03448276 0.02857143 0.03225806
## [13] 0.03703704 0.04545455 0.02702703 0.02941176 0.05263158 0.05000000
## [19] 0.01754386 0.02040816 0.02000000 0.02702703 0.02173913 0.04000000
## [25] 0.05882353 0.02702703 0.02380952 0.01886792 0.02439024 0.01960784
## [31] 0.02857143 0.04166667 0.03030303 0.02439024
```

3.New Age

```
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)

new_age <- c(age, 0, age)
print(new_age)
```

```
##  [1] 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
## [26] 37 42 53 41 51 35 24 33 41  0 34 28 22 36 27 18 52 39 42 29 35 31 27 22 37
## [51] 34 19 20 57 49 50 37 46 25 17 37 42 53 41 51 35 24 33 41
```

4.Sort Age

```
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)
sortage <- sort(age)
print(sortage)
```

```
##  [1] 17 18 19 20 22 22 24 25 27 27 28 29 31 33 34 34 35 35 36 37 37 37 39 41 41
## [26] 42 42 46 49 50 51 52 53 57
```

5.Maximum and Minimum Age

```
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_min <- min(age)
max_age <- max(age)
print(age_min)
```

```
## [1] 17
```

```
print(max_age)
```

```
## [1] 57
```

6.Vector Data

```
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)
length(data)
```

```
## [1] 12
```

7. Double Data

```
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)
double_data <- data * 2
print(double_data)
```

```
##  [1] 4.8 5.6 4.2 5.0 4.8 4.4 5.0 4.6 5.0 4.6 4.8 5.4
```

8. Sequence

- 8.1

```
seq(1:100)
```

```
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  [19]  19  20  21  22  23  24  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  50  51  52  53  54
##  [55]  55  56  57  58  59  60  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  86  87  88  89  90
##  [91]  91  92  93  94  95  96  97  98  99 100
```

- 8.2

```
seq(20:60)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
```

```
## [26] 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
```

- 8.3

```
mean(20:60)
```

```
## [1] 40
```

- 8.4

```
sum(51:91)
```

```
## [1] 2911
```

- 8.5

```
seq(1:1000)
```

```
##    [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14
##   [15]  15  16  17  18  19  20  21  22  23  24  25  26  27  28
##   [29]  29  30  31  32  33  34  35  36  37  38  39  40  41  42
##   [43]  43  44  45  46  47  48  49  50  51  52  53  54  55  56
##   [57]  57  58  59  60  61  62  63  64  65  66  67  68  69  70
##   [71]  71  72  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  98
##   [99]  99 100 101 102 103 104 105 106 107 108 109 110 111 112
##  [113] 113 114 115 116 117 118 119 120 121 122 123 124 125 126
##  [127] 127 128 129 130 131 132 133 134 135 136 137 138 139 140
##  [141] 141 142 143 144 145 146 147 148 149 150 151 152 153 154
##  [155] 155 156 157 158 159 160 161 162 163 164 165 166 167 168
##  [169] 169 170 171 172 173 174 175 176 177 178 179 180 181 182
##  [183] 183 184 185 186 187 188 189 190 191 192 193 194 195 196
##  [197] 197 198 199 200 201 202 203 204 205 206 207 208 209 210
##  [211] 211 212 213 214 215 216 217 218 219 220 221 222 223 224
##  [225] 225 226 227 228 229 230 231 232 233 234 235 236 237 238
##  [239] 239 240 241 242 243 244 245 246 247 248 249 250 251 252
##  [253] 253 254 255 256 257 258 259 260 261 262 263 264 265 266
##  [267] 267 268 269 270 271 272 273 274 275 276 277 278 279 280
##  [281] 281 282 283 284 285 286 287 288 289 290 291 292 293 294
##  [295] 295 296 297 298 299 300 301 302 303 304 305 306 307 308
##  [309] 309 310 311 312 313 314 315 316 317 318 319 320 321 322
##  [323] 323 324 325 326 327 328 329 330 331 332 333 334 335 336
##  [337] 337 338 339 340 341 342 343 344 345 346 347 348 349 350
##  [351] 351 352 353 354 355 356 357 358 359 360 361 362 363 364
##  [365] 365 366 367 368 369 370 371 372 373 374 375 376 377 378
##  [379] 379 380 381 382 383 384 385 386 387 388 389 390 391 392
##  [393] 393 394 395 396 397 398 399 400 401 402 403 404 405 406
##  [407] 407 408 409 410 411 412 413 414 415 416 417 418 419 420
##  [421] 421 422 423 424 425 426 427 428 429 430 431 432 433 434
##  [435] 435 436 437 438 439 440 441 442 443 444 445 446 447 448
##  [449] 449 450 451 452 453 454 455 456 457 458 459 460 461 462
##  [463] 463 464 465 466 467 468 469 470 471 472 473 474 475 476
##  [477] 477 478 479 480 481 482 483 484 485 486 487 488 489 490
##  [491] 491 492 493 494 495 496 497 498 499 500 501 502 503 504
##  [505] 505 506 507 508 509 510 511 512 513 514 515 516 517 518
##  [519] 519 520 521 522 523 524 525 526 527 528 529 530 531 532
##  [533] 533 534 535 536 537 538 539 540 541 542 543 544 545 546
##  [547] 547 548 549 550 551 552 553 554 555 556 557 558 559 560
```

```
##  [561]  561  562  563  564  565  566  567  568  569  570  571  572  573  574
##  [575]  575  576  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  602
##  [603]  603  604  605  606  607  608  609  610  611  612  613  614  615  616
##  [617]  617  618  619  620  621  622  623  624  625  626  627  628  629  630
##  [631]  631  632  633  634  635  636  637  638  639  640  641  642  643  644
##  [645]  645  646  647  648  649  650  651  652  653  654  655  656  657  658
##  [659]  659  660  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  686
##  [687]  687  688  689  690  691  692  693  694  695  696  697  698  699  700
##  [701]  701  702  703  704  705  706  707  708  709  710  711  712  713  714
##  [715]  715  716  717  718  719  720  721  722  723  724  725  726  727  728
##  [729]  729  730  731  732  733  734  735  736  737  738  739  740  741  742
##  [743]  743  744  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  770
##  [771]  771  772  773  774  775  776  777  778  779  780  781  782  783  784
##  [785]  785  786  787  788  789  790  791  792  793  794  795  796  797  798
##  [799]  799  800  801  802  803  804  805  806  807  808  809  810  811  812
##  [813]  813  814  815  816  817  818  819  820  821  822  823  824  825  826
##  [827]  827  828  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  854
##  [855]  855  856  857  858  859  860  861  862  863  864  865  866  867  868
##  [869]  869  870  871  872  873  874  875  876  877  878  879  880  881  882
##  [883]  883  884  885  886  887  888  889  890  891  892  893  894  895  896
##  [897]  897  898  899  900  901  902  903  904  905  906  907  908  909  910
##  [911]  911  912  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  938
##  [939]  939  940  941  942  943  944  945  946  947  948  949  950  951  952
##  [953]  953  954  955  956  957  958  959  960  961  962  963  964  965  966
##  [967]  967  968  969  970  971  972  973  974  975  976  977  978  979  980
##  [981]  981  982  983  984  985  986  987  988  989  990  991  992  993  994
##  [995]  995  996  997  998  999 1000
```

9.Filter

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

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

10.Backward Sequence

```
rev(1:100)
```

```
##   [1] 100  99  98  97  96  95  94  93  92  91  90  89  88  87  86  85  84  83
##  [19]  82  81  80  79  78  77  76  75  74  73  72  71  70  69  68  67  66  65
##  [37]  64  63  62  61  60  59  58  57  56  55  54  53  52  51  50  49  48  47
##  [55]  46  45  44  43  42  41  40  39  38  37  36  35  34  33  32  31  30  29
##  [73]  28  27  26  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.

```
numero <- 1:25
multiples <- numero[numero %% 3== 0 | numero %% 5 == 0]
multiples
```

```
##  [1]  3  5  6  9 10 12 15 18 20 21 24 25
```

```r
sum <- sum(multiples)
sum
```

```
## [1] 168
```

11.a

```r
num <- 10:11
count <- length(num)
count
```

```
## [1] 2
```

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 is a syntax error for the particular reason that the statements ends in a + and was not given another value .

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

```
## [1] 86
```

```r
score[3]
```

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

```
## [1]    1    2 -999    4 -999    6    7
```

the function makes the display the na values as -999 in the output. even though, it doesnt replace the na within the vector but only affects the output.

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

```r
name = readline(prompt="Input your name: ")
```

```
## Input your name:
```

```r
age = readline(prompt="Input your age: ")
```

```
## Input your age:
```

```r
print(paste("My name is",name, "and I am",age ,"years old."))
```

```
## [1] "My name is  and I am  years old."
```

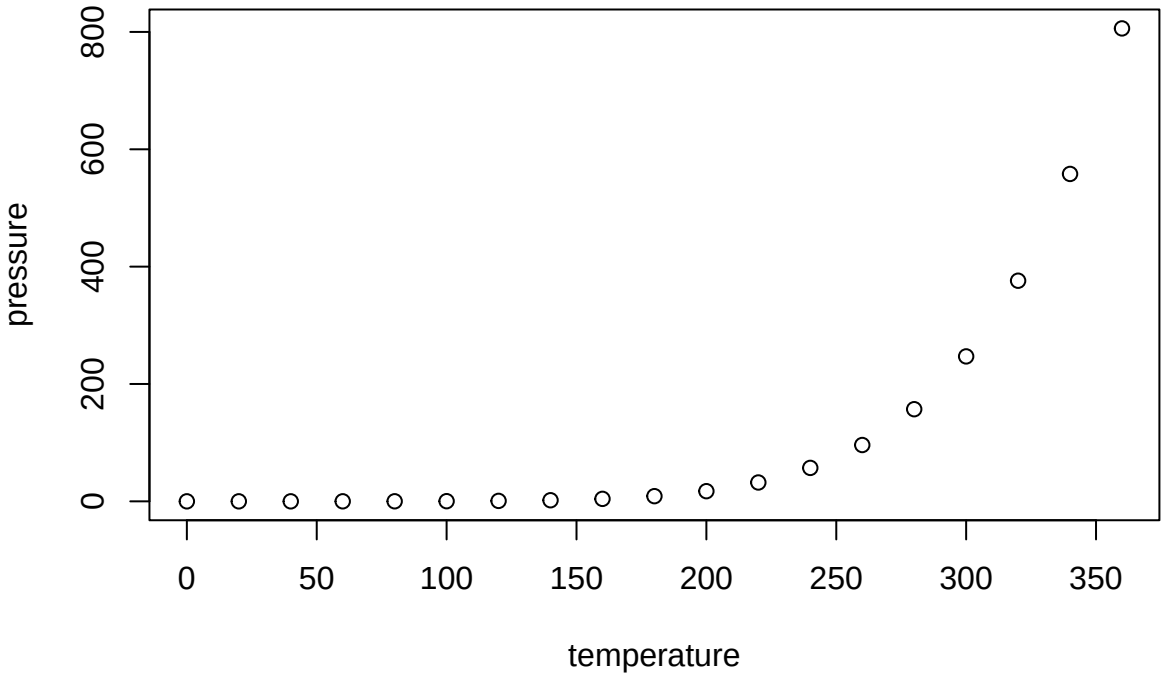
```r
print(R.version.string)
```

```
## [1] "R version 4.4.1 (2024-06-14)"
```

the output is “My name is Marco Luis Llanera and I am 19 years old.” “R version 4.4.1 (2024-06-14)”

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.