100 R Vector and Matrix Exercises

This is the R Version of 100 numpy exercises

Install package Magrittr (1 Star)

Load package Magrittr (1 Star)

```
library(magrittr)
```

Print the R version (1 Star)

```
print(version$version.string)
```

```
## [1] "R version 3.3.0 (2016-05-03)"
```

Create a null vector of size 10 (1 Star)

```
rep(NA, 10) %>% print
```

```
##  [1] NA NA NA NA NA NA NA NA NA NA
```

How to get the examples of the add (+) function from the command line ? (1 Star)

```
example(`+`)
```

```
##
## +> x <- -1:12
##
## +> x + 1
##  [1]  0  1  2  3  4  5  6  7  8  9 10 11 12 13
##
## +> 2 * x + 3
##  [1]  1  3  5  7  9 11 13 15 17 19 21 23 25 27
##
## +> x %% 2 #-- is periodic
##  [1] 1 0 1 0 1 0 1 0 1 0 1 0 1 0
##
## +> x %/% 5
##  [1] -1  0  0  0  0  0  1  1  1  1  1  2  2  2
```

Create a null vector of size 10 but the fifth value which is 1 (1 Star)

```
z <- rep(NA, 10)
z[5] <- 1
print(z)
```

```
##  [1] NA NA NA NA  1 NA NA NA NA NA
```

Create a vector with values ranging from 10 to 49 (1 Star)

```
10:49 %>% print
```

```
##  [1] 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
## [24] 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
```

Reverse a vector (first element becomes last) (1 Star)

```
1:10 %>% rev %>% print
```

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

Create a 3x3 matrix with values ranging from 0 to 8 (1 Star)

```
matrix(0:8, ncol = 3, nrow = 3) %>% print
```

```
##      [,1] [,2] [,3]
## [1,]    0    3    6
## [2,]    1    4    7
## [3,]    2    5    8
```

Find indices of non-zero elements from (1,2,0,0,4,0) (1 Star)

```
c(1,2,0,0,4,0) %>% as.logical %>% which %>% print
```

```
## [1] 1 2 5
```

Create a 3x3 identity matrix (1 Star)

```
diag(3) %>% print
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
```

Create a 3x3x3 array with random values (1 Star)

```
runif(3^3) %>% array(c(3, 3, 3)) %>% print
```

```
## , , 1
##
##           [,1]      [,2]      [,3]
## [1,] 0.1467886 0.8598928 0.8086439
## [2,] 0.1996387 0.7377839 0.5032338
## [3,] 0.7456743 0.2944287 0.5709111
##
## , , 2
##
##           [,1]      [,2]      [,3]
```

```
## [1,] 0.1198385 0.3728561 0.6310738
## [2,] 0.9073268 0.2396141 0.4479291
## [3,] 0.4672833 0.5336210 0.3111638
##
## , , 3
##
##           [,1]      [,2]       [,3]
## [1,] 0.4038644 0.4465031 0.19468248
## [2,] 0.7911760 0.7879483 0.07573934
## [3,] 0.4442609 0.5260825 0.53046187
```

Create a 10x10 array with random values and find the minimum and maximum values (1 Star)

```
z <- array(runif(10^2), c(10,10))
min(z) %>% print
```

```
## [1] 0.02954839
```

```
max(z) %>% print
```

```
## [1] 0.9838782
```

Create a random vector of size 30 and find the mean value (1 Star)

```
runif(30) %>% mean %>% print
```

```
## [1] 0.5046362
```

Create a 2d array with 1 on the border and 0 inside (1 Star)

```
z <- array(1, c(4, 5))
z[2:(nrow(z) - 1), 2:(ncol(z) - 1)] <- 0
print(z)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    1    1    1    1
## [2,]    1    0    0    0    1
## [3,]    1    0    0    0    1
## [4,]    1    1    1    1    1
```

What is the result of the following expression ? (1 Star)

```
0 * NaN
```

```
## [1] NaN
```

```
NA == NA
```

```
## [1] NA
```

```
NA == NaN
```

```
## [1] NA
```

```
Inf > NA
```

```
## [1] NA
```

```
NaN - NaN
```

```
## [1] NaN
```

```
0.3 == 3 * 0.1
```

```
## [1] FALSE
```

Create a 5x5 matrix with values 1,2,3,4 just below the diagonal (1 Star)

```
z <- diag(0, ncol = 5, nrow = 5)
z[row(z) - 1 == col(z)] <- 1:4
print(z)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    0    0    0    0
## [2,]    1    0    0    0    0
## [3,]    0    2    0    0    0
## [4,]    0    0    3    0    0
## [5,]    0    0    0    4    0
```

Create a 8x8 matrix and fill it with a checkerboard pattern (1 Star)

```
z <- array(0, c(8, 8))
z[1:nrow(z) %% 2 != 0, 1:ncol(z) %% 2 == 0] <- 1
z[1:nrow(z) %% 2 == 0, 1:ncol(z) %% 2 != 0] <- 1
print(z)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]    0    1    0    1    0    1    0    1
## [2,]    1    0    1    0    1    0    1    0
## [3,]    0    1    0    1    0    1    0    1
## [4,]    1    0    1    0    1    0    1    0
## [5,]    0    1    0    1    0    1    0    1
## [6,]    1    0    1    0    1    0    1    0
## [7,]    0    1    0    1    0    1    0    1
## [8,]    1    0    1    0    1    0    1    0
```

Consider a (6,7,8) shape array, what is the index (x,y,z) of the 100th element ?

```
z <- array(runif(6 * 7 * 8), c(6, 7, 8))
which(z == z[100], arr.ind = T)
```

```
##      dim1 dim2 dim3
## [1,]    4    3    3
```

Create a checkerboard 8x8 matrix using the rep function (1 Star)

```
odd <- rep(0:1, 4)
evn <- rep(1:0, 4)
z <- array(c(odd, evn), c(8, 8))
print(z)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]    0    1    0    1    0    1    0    1
## [2,]    1    0    1    0    1    0    1    0
## [3,]    0    1    0    1    0    1    0    1
## [4,]    1    0    1    0    1    0    1    0
## [5,]    0    1    0    1    0    1    0    1
## [6,]    1    0    1    0    1    0    1    0
## [7,]    0    1    0    1    0    1    0    1
## [8,]    1    0    1    0    1    0    1    0
```

Normalize a 5x5 random matrix (1 Star)

```
runif(5 * 5) %>% matrix(nrow = 5, ncol = 5) %>% scale %>% print
```

```
##            [,1]       [,2]       [,3]        [,4]       [,5]
## [1,]  0.6535623 -1.2927908 -1.6165672 -0.06637855  0.5976826
## [2,]  0.8628847  1.2919027  0.2925858 -1.06757389 -0.5514221
## [3,] -1.1808348 -0.6185153  0.9031822 -0.88804371 -1.4477943
## [4,] -0.9977452  0.5162059  0.6558376  1.11829216  0.3398983
## [5,]  0.6621330  0.1031975 -0.2350384  0.90370398  1.0616356
## attr(,"scaled:center")
## [1] 0.6493817 0.4291895 0.8239928 0.6560874 0.6106454
## attr(,"scaled:scale")
## [1] 0.2178393 0.1422153 0.1024269 0.2440772 0.2912395
```

Multiply a 5x3 matrix by a 3x2 matrix (real matrix product) (1 Star)

```
matrix(1, 5, 3) %*% matrix(1, 3, 2) %>% print
```

```
##      [,1] [,2]
## [1,]    3    3
## [2,]    3    3
## [3,]    3    3
## [4,]    3    3
## [5,]    3    3
```

Given a 1D array, negate all elements which are between 3 and 8, in place. (1 Star)

```r
z <- 1:15
ifelse(z > 3 & z <= 8, -z, z) %>% print
```

```
##  [1]  1  2  3 -4 -5 -6 -7 -8  9 10 11 12 13 14 15
```

Create a 5x5 matrix with row values ranging from 0 to 4 (1 Star)

```r
matrix(0:4, 5, 5)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    0    0    0    0
## [2,]    1    1    1    1    1
## [3,]    2    2    2    2    2
## [4,]    3    3    3    3    3
## [5,]    4    4    4    4    4
```

Generate 5 vectors and use them to build a matrix (2 Stars)

```r
vecs <- paste0("vec", 1:5)

for (i in 1:5){
  assign(vecs[i], runif(5))
}

sapply(vecs, get)
```

```
##           vec1       vec2      vec3       vec4       vec5
## [1,] 0.2362224 0.04255305 0.3850008 0.52163208 0.85365019
## [2,] 0.5397320 0.31247895 0.4987810 0.09847288 0.34577392
## [3,] 0.4901433 0.16744916 0.8670547 0.12151656 0.06976057
## [4,] 0.3306689 0.54907564 0.7916659 0.65007070 0.24201878
## [5,] 0.3069354 0.47525079 0.4065482 0.43360158 0.38121455
```

Create a vector of size 10 with values ranging from 0 to 1, both excluded (2 Stars)

```r
z <- seq(0, 1, length.out = 12)
z[c(-1, -length(z))] %>% print
```

```
##  [1] 0.09090909 0.18181818 0.27272727 0.36363636 0.45454545 0.54545455
##  [7] 0.63636364 0.72727273 0.81818182 0.90909091
```

Create a random vector of size 10 and sort it (2 Stars)

```r
runif(10) %>% sort %>% print
```

```
##  [1] 0.05023180 0.06252409 0.18625289 0.23458163 0.54979664 0.69224840
##  [7] 0.74153004 0.89660297 0.98268724 0.99266560
```

How to sum a small array faster without sum ? (2 Stars)

```r
1:10 %>% {Reduce(`+`, .)} %>% print
```

```
## [1] 55
```

Consider two random array A anb B, check if they are equal (2 Stars)

```r
a <- replicate(5, runif(5)) %>% as.array
b <- replicate(5, runif(5)) %>% as.array
all(a == b) %>% print
```

```
## [1] FALSE
```

Make an array immutable (read-only) (2 Stars)

```r
z <- array(runif(3), c(3, 4))
lockBinding("z", env = .GlobalEnv)
try(z <- array(runif(3), c(3, 4))) %>% print
```

```
## [1] "Error in try(z <- array(runif(3), c(3, 4))) : \n  cannot change value of locked binding for 'z'"
## attr(,"class")
## [1] "try-error"
## attr(,"condition")
## <simpleError in doTryCatch(return(expr), name, parentenv, handler): cannot change value of locked bi
```

```r
unlockBinding("z", env = .GlobalEnv)
```

Consider a random 10x2 matrix representing cartesian coordinates, convert them to polar coordinates (2 Stars)

```r
z <- replicate(2, runif(10, min = 0, max = 360))
x <- z[,1]
y <- z[,2]
r = sqrt(x^2 + y^2) %>% print
```

```
##  [1] 450.2868 151.3716 349.1725 270.1782 319.7419 394.4376 354.3184
##  [8] 181.9334 184.3135 244.1567
```

```r
t = atan2(y, x) %>% print
```

```
##  [1] 0.81194884 1.01565605 1.25570160 0.24959676 0.07134581 1.13678533
##  [7] 0.89617370 0.76045196 1.31463662 1.44990485
```

Create random vector of size 10 and replace the maximum value by 0 (2 Stars)

```r
z <- runif(10)
z[max(z) == z] <- 0
print(z)
```

```
##  [1] 0.2295380 0.1107783 0.1463300 0.8643854 0.9028234 0.4999116 0.2787289
##  [8] 0.6698533 0.0000000 0.7677761
```