

# STA 445 HW2

Levi Mault

2/29/2024

## Problem 1

Create a vector of three elements (2,4,6) and name that vector `vec_a`. Create a second vector, `vec_b`, that contains (8,10,12). Add these two vectors together and name the result `vec_c`.

```
vec_a <- c(2, 4, 6)
vec_b <- c(8, 10, 12)

vec_c <- vec_a + vec_b
vec_c
```

```
## [1] 10 14 18
```

## Problem 2

Create a vector, named `vec_d`, that contains only two elements (14,20). Add this vector to `vec_a`. What is the result and what do you think R did (look up the recycling rule using Google)? What is the warning message that R gives you?

```
vec_d <- c(14, 20)

vec_d + vec_a
```

```
## Warning in vec_d + vec_a: longer object length is not a multiple of shorter
## object length
## [1] 16 24 20
```

It appears that R repeated added the values in vector d into vector a, adding `vec_d[0] = 14` to `vec_a[0]` and `vec_a[3]`. The error was due to the longer vector not being an integer multiple of the shorter vector (i.e. it was able to add the first value of vector d into vector a twice, but the second value of vector d was only added into vector a once, not twice as well).

## Problem 3

Next add 5 to the vector `vec_a`. What is the result and what did R do? Why doesn't it give you a warning message similar to what you saw in the previous problem?

```
vec_a = c(vec_a, 5)

vec_d + vec_a
```

```
## [1] 16 24 20 25
```

That is because the longer vector is an integer multiple of the shorter vector.

## Problem 4

Generate the vector of integers  $\{1, 2, \dots, 5\}$  in two different ways.

a. First using the `seq()` function

```
vec_prob4 <- seq(from=1, to=5, by=1)
vec_prob4
```

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

b. Using the `a:b` shortcut.

```
vec_prob4b <- c(1:5)
vec_prob4b
```

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

## Problem 5

Generate the vector of even numbers  $\{2, 4, 6, \dots, 20\}$

a. Using the `seq()` function

```
vec_prob5a <- seq(from=2, to=20, by=2)
vec_prob5a
```

```
## [1] 2 4 6 8 10 12 14 16 18 20
```

b. Using the `a:b` shortcut and some subsequent algebra.

```
vec_prob5b <- c(1:10)
vec_prob5b * 2
```

```
## [1] 2 4 6 8 10 12 14 16 18 20
```

## Problem 6

Generate a vector of 21 elements that are evenly placed between 0 and 1 using the `seq()` command and name this vector `x`.

```
x <- seq(from=0, to=1, length.out=21)
x
```

```
## [1] 0.00 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50 0.55 0.60 0.65 0.70
## [16] 0.75 0.80 0.85 0.90 0.95 1.00
```

## Problem 7

Generate the vector  $\{2, 4, 8, 2, 4, 8, 2, 4, 8\}$  using the `rep()` command to replicate the vector `c(2,4,8)`.

```
vec_prob7 <- rep(c(2, 4, 8), 3)
vec_prob7
```

```
## [1] 2 4 8 2 4 8 2 4 8
```

## Problem 8

Generate the vector  $\{2, 2, 2, 2, 4, 4, 4, 4, 8, 8, 8, 8\}$  using the `rep()` command. You might need to check the help file for `rep()` to see all of the options that `rep()` will accept. In particular, look at the optional argument `each=`.

```
vec_prob8 <- rep(c(2, 4, 8), each=4)
vec_prob8
```

```
## [1] 2 2 2 2 4 4 4 4 8 8 8 8
```

## Problem 9

In this problem, we will work with the matrix

$$\begin{bmatrix} 2 & 4 & 6 & 8 & 10 \\ 12 & 14 & 16 & 18 & 20 \\ 22 & 24 & 26 & 28 & 30 \end{bmatrix}$$

- Create the matrix in two ways and save the resulting matrix as M.
- Create the matrix using some combination of the `seq()` and `matrix()` commands.

```
M <- matrix(seq(from=2, to=30, by=2), nrow = 3, ncol = 5, byrow = TRUE)
M
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    2    4    6    8   10
## [2,]   12   14   16   18   20
## [3,]   22   24   26   28   30
```

- Create the same matrix by some combination of multiple `seq()` commands and either the `rbind()` or `cbind()` command.

```
a <- seq(2, 10, by=2)
b <- seq(12, 20, by=2)
c <- seq(22, 30, by=2)

rbind(a,b,c)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## a      2    4    6    8   10
## b     12   14   16   18   20
## c     22   24   26   28   30
```

- Extract the second row out of M.

```
M[2, ]
```

```
## [1] 12 14 16 18 20
```

- Extract the element in the third row and second column of M

```
M[2,3]
```

```
## [1] 16
```

## Problem 10

The following code creates a `data.frame` and then has two different methods for removing the rows with NA values in the column `Grade`. Explain the difference between the two.

```
df <- data.frame(name= c('Alice', 'Bob', 'Charlie', 'Daniel'),
                  Grade = c(6,8,NA,9))

df[ -which( is.na(df$Grade) ), ]
```

```
df[ which( !is.na(df$Grade) ), ]
```

The first method is forcing the negative value of the bool evaluation by essentially multiplying the result by -1.

The second method is using the bang logical operator (!) within the evaluation to ask if the value at df\$Grade is NOT NA.

## Problem 11

Create and manipulate a list.

- a. Create a list named my.test with elements +  $x = c(4,5,6,7,8,9,10)$  +  $y = c(34,35,41,40,45,47,51)$  + slope = 2.82 + p.value = 0.000131

```
my.test <- list(x = c(4,5,6,7,8,9,10),
               y = c(34,35,41,40,45,47,51),
               slope = 2.82,
               p.value = 0.000131)

my.test
```

```
## $x
## [1] 4 5 6 7 8 9 10
##
## $y
## [1] 34 35 41 40 45 47 51
##
## $slope
## [1] 2.82
##
## $p.value
## [1] 0.000131
```

- b. Extract the second element in the list.

```
my.test[2]

## $y
## [1] 34 35 41 40 45 47 51
```

- c. Extract the element named p.value from the list.

```
my.test$p.value

## [1] 0.000131
```