

Assignment1-445

Mandi Bluth

2024-09-30

Question 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 <- c(vec_a, vec_b)
vec_c
```

```
## [1] 2 4 6 8 10 12
```

Question 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_a + vec_d
```

```
## Warning in vec_a + vec_d: longer object length is not a multiple of shorter
## object length
```

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

There is a warning because the length of the longer vector is not a multiple of the shorter one, however, R will perform the addition anyways by recycling objects in the shorter vector. In this example, it added 14 to the first object in **vec_a**, 20 to the second object in **vec_a**, then ran out of objects in **vec_d** so it restarted/recycled by adding 14 to the last element in **vec_a**.

Question 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 + 5
```

```
## [1] 7 9 11
```

Here there was no warning given because the the length of the longer vector is a multiple of the shorter one.
(3 / 1 = 3)

Question 4

Generate the vector of integers $\{1, 2, \dots, 5\}$ in two different ways. a) First using the `seq()` function

```
seq(1,5)
```

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

b) Using the `a:b` shortcut.

```
1:5
```

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

Question 5

Generate the vector of even numbers $\{2, 4, 6, \dots, 20\}$ a) Using the `seq()` function and

```
seq(2,20,2)
```

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

b) Using the `a:b` shortcut and some subsequent algebra. *Hint: Generate the vector 1-10 and then multiple it by 2.*

```
(1:10)*2
```

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

Question 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(0, 1, ,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
```

Question 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)`.

```
rep(c(2,4,8), 3)
```

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

Question 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=`.

```
rep(c(2, 4, 8), each = 4)
```

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

Question 9

The vector `letters` is a built-in vector to R and contains the lower case English alphabet. a) Extract the 9th element of the `letters` vector.

```
letters[9]
```

```
## [1] "i"
```

b) Extract the sub-vector that contains the 9th, 11th, and 19th elements.

```
letters[c(9, 11, 19)]
```

```
## [1] "i" "k" "s"
```

c) Extract the sub-vector that contains everything except the last two elements.

```
letters[-c(25, 26)]
```

```
## [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"  
## [20] "t" "u" "v" "w" "x"
```

Question 10

In this problem, we will work with the matrix

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

a) Create the matrix in two ways and save the resulting matrix as M.

i. Create the matrix using some combination of the `seq()` and `matrix()` commands.

```
M <- matrix( seq(2, 30, 2), nrow=3, 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
```

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

```
M <- rbind(seq(2, 10, 2), seq(12, 20, 2), seq(22, 30, 2))
M
```

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

b) Extract the second row out of M.

```
M[2, ]
```

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

c) Extract the element in the third row and second column of M.

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

```
## [1] 24
```

Question 11

Create and manipulate a data frame. a) Create a `data.frame` named `my.trees` that has the following columns: + Girth = {8.3, 8.6, 8.8, 10.5, 10.7, 10.8, 11.0} + Height= {70, 65, 63, 72, 81, 83, 66} + Volume= {10.3, 10.3, 10.2, 16.4, 18.8, 19.7, 15.6}

```
my.trees <- data.frame ( Girth = c(8.3, 8.6, 8.8, 10.5, 10.7, 10.8, 11.0),
                        Height= c(70, 65, 63, 72, 81, 83, 66),
                        Volume= c(10.3, 10.3, 10.2, 16.4, 18.8, 19.7, 15.6)
                      )
```

b) Without using `dplyr` functions, extract the third observation (i.e. the third row)

```
my.trees[3, ] # how to make extraction work
```

```
##   Girth Height Volume
## 3   8.8     63   10.2
```

- c) Without using `dplyr` functions, extract the `Girth` column referring to it by name (don't use whatever order you placed the columns in).

```
my.trees[["Girth"]]
```

```
## [1]  8.3  8.6  8.8 10.5 10.7 10.8 11.0
```

- d) Without using `dplyr` functions, print out a data frame of all the observations *except* for the fourth observation. (i.e. Remove the fourth observation/row.)

```
my.trees[-4, ]
```

```
##   Girth Height Volume
## 1   8.3     70   10.3
## 2   8.6     65   10.3
## 3   8.8     63   10.2
## 5  10.7     81   18.8
## 6  10.8     83   19.7
## 7  11.0     66   15.6
```

- e) Without using `dplyr` functions, use the `which()` command to create a vector of row indices that have a `girth` greater than 10. Call that vector `index`.

```
index <- which(my.trees$Girth > 10)
index
```

```
## [1] 4 5 6 7
```

- f) Without using `dplyr` functions, use the `index` vector to create a small data set with just the large girth trees.

```
large.trees <- my.trees[index, ]
```

- g) Without using `dplyr` functions, use the `index` vector to create a small data set with just the small girth trees.

```
small.trees <- my.trees[-index, ]
```

Question 12

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 statement uses the which to finds the na values, then puts a negative in front of the which, which will remove the na values found. The second statement uses which to finds the values that are NOT na, and keeps those in the data frame

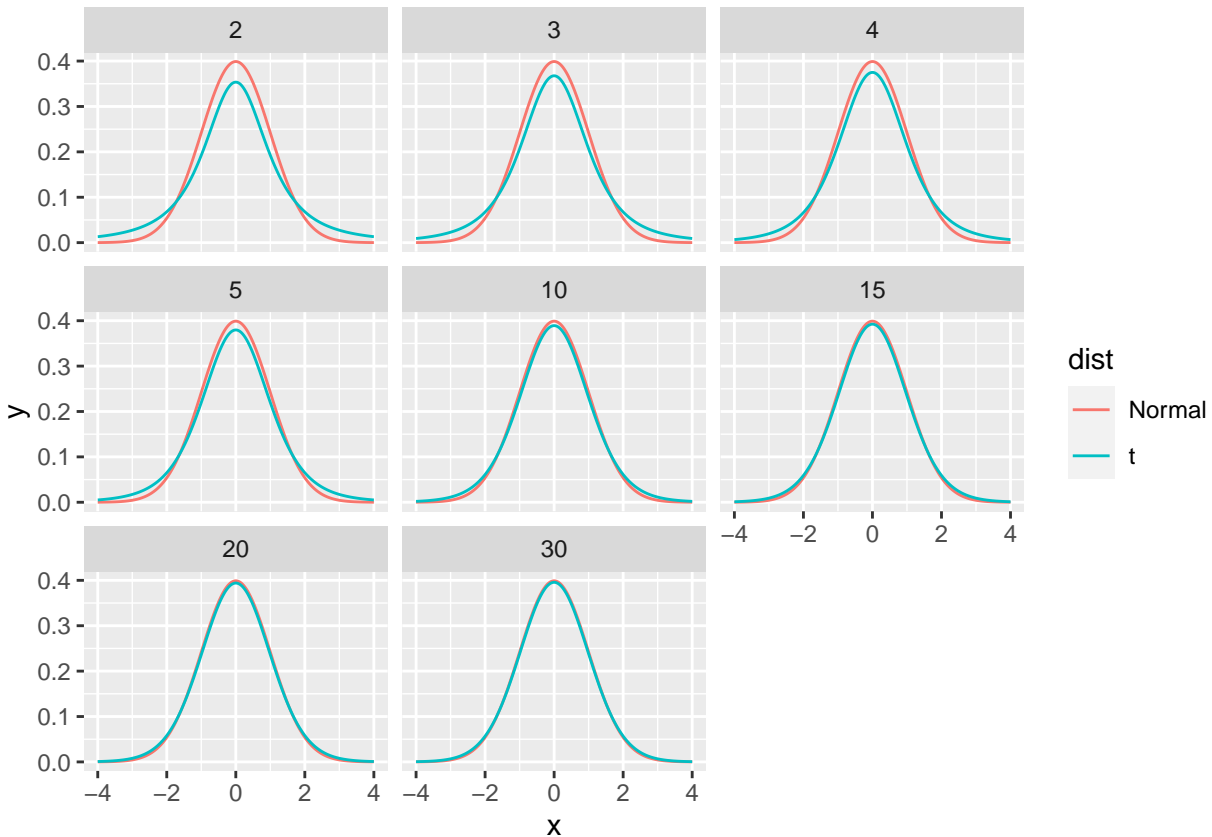
Question 13

Creation of data frames is usually done by binding together vectors while using `seq` and `rep` commands. However often we need to create a data frame that contains all possible combinations of several variables. The function `expand.grid()` addresses this need.

```
expand.grid( F1=c('A','B'), F2=c('x','w','z'), replicate=1:2 )
```

A fun example of using this function is making several graphs of the standard normal distribution versus the t-distribution. Use the `expand.grid` function to create a `data.frame` with all combinations of `x=seq(-4,4,by=.01)`, `dist=c('Normal','t')`, and `df=c(2,3,4,5,10,15,20,30)`. Use the `dplyr::mutate` command with the `if_else` command to generate the function heights `y` using either `dt(x,df)` or `dnorm(x)` depending on what is in the distribution column.

```
expand.grid( x=seq(-4,4,by=.01),
              dist=c('Normal','t'),
              df=c(2,3,4,5,10,15,20,30) ) %>%
mutate( y = if_else(dist == 't', dt(x, df), dnorm(x) ) ) %>%
  ggplot( aes( x=x, y=y, color=dist) ) +
  geom_line() +
  facet_wrap(~df)
```



Question 14

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
```

```
my.test[["p.value"]]
```

```
## [1] 0.000131
```