Module8

Joe Vargovich

9/26/2020

Exercise 1 - Create vectors of length three and add them.

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

vec_c = vec_a + vec_b

vec_c
```

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

Exercise 2 - Addition of vectors of differing size

```
vec_d = c(14,20)

#The recycling rule allows us to add two vectors of differing length as we can
#recycle the values of the shorter one and add them to the longer vector.
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
```

Exercise 3 - Add a constant to a vector

```
# Add 5 to the vector a.
vec_a + 5
```

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

```
#This works without error or warning as the one element is applied to each element
# of the target vector. We are not adding vectors here! No warning!
```

#Exercise 4 - Generate a vector of integers with two methods

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

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

```
#2
1:5
```

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

#Exercise 5 - Generate a vector of even integers with two methods

```
#1
seq(2,20,2)
```

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

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

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

Exercise 6 - Generate a vector of 21 evenly spaced elements between 0 and 1

```
x = seq(0,1, length=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
```

Exercise 7 - Generate a vector by repeating another one

```
to_repeat = c(2,4,8)

result = rep(to_repeat, 3)

result
```

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

#Exercise 8 - Generate another vector with rep, with repeats in order using each

```
result = rep(to_repeat, each=4)

result
```

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

#Exercise 9 - Work with the letters R vector

```
cat(as.character(letters), sep = ",", file = "letters.txt", append = TRUE)
#a. Get the 9th element of letters
letters[9]
```

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

```
#b. Extract the subvector with the 9th, 11th, and 18th elements
letters[c(9,11,18)]
```

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

```
#c. Extract the subvector that is everything but the last two letters
letters[1:24]
```

```
##  [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"
```

#Exercise 10 - Create a matrix and perform operations

```
#ai.- Using the matrix function and seq function
matrix1 = matrix(seq(2,30, 2), nrow = 3, ncol = 5, byrow=TRUE)
matrix1
```

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

```
#aii. - Chaining rbinds
row1 = seq(2,10, 2)
row2 = seq(12,20, 2)
row3 = seq(22,30, 2)
matrix2 = rbind(row1,row2) %>%
  rbind(row3)

matrix2
```

```
##      [,1] [,2] [,3] [,4] [,5]
## row1    2    4    6    8   10
## row2   12   14   16   18   20
## row3   22   24   26   28   30
```

```
#b. Grab the second row of our matrix
matrix1[2,]
```

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

```
#c. Grab the second element of the third row of our matrix
matrix1[3,2]
```

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

#Exercise 11 - Create and manipulate a data frame.

```
#a. Create dataframe my.trees
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. Extract the third observation (row) fron the dataframe
my.trees[3,]
```

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

```
#c. Extract the Girth column by name
my.trees$Girth
```

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

```
#d. Extract all but the fourth row
my.trees[-c(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. Use the which command to create a vector on condition
index = which(my.trees$Girth > 10)
index
```

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

```
#f. Create a data set with just the large trees
large.trees = data.frame(
  Girth = my.trees$Girth[index],
  Height = my.trees$Height[index],
  Volume = my.trees$Volume[index]
)
large.trees
```

```
##   Girth Height Volume
## 1  10.5     72   16.4
## 2  10.7     81   18.8
## 3  10.8     83   19.7
## 4  11.0     66   15.6
```

```
#g. Create a data set with just the small trees
small.trees = data.frame(
  Girth = my.trees$Girth[-c(index)],
  Height = my.trees$Height[-c(index)],
  Volume = my.trees$Volume[-c(index)]
)
small.trees
```

```
##   Girth Height Volume
## 1   8.3     70   10.3
## 2   8.6     65   10.3
## 3   8.8     63   10.2
```

#Exercise 12 - Describe the differnece between the two data frame row removal methods

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

#This will remove the rows that are na as per the - sign.
df[ -which(  is.na(df$Grade) ), ]
```

```
##     name Grade
## 1  Alice     6
## 2    Bob     8
## 4 Daniel     9
```

```
#This will only select the rows that are not na. The ! operator negates the is.na boolean.
df[  which( !is.na(df$Grade) ), ]
```

```
##     name Grade
## 1  Alice     6
## 2    Bob     8
```

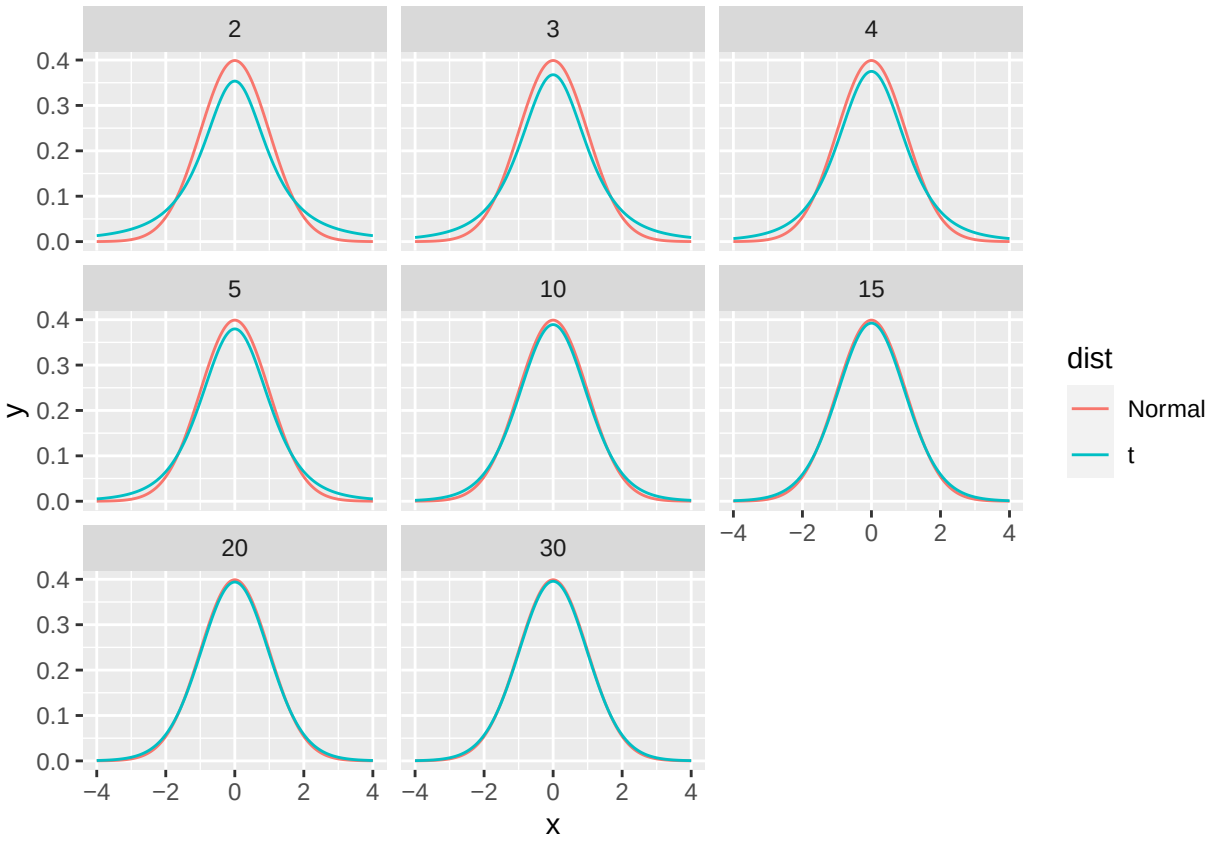
```
## 4 Daniel     9
```

#Exercise 13 - Using expand.grid

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

```
##    F1 F2 replicate
## 1   A  x         1
## 2   B  x         1
## 3   A  w         1
## 4   B  w         1
## 5   A  z         1
## 6   B  z         1
## 7   A  x         2
## 8   B  x         2
## 9   A  w         2
## 10  B  w         2
## 11  A  z         2
## 12  B  z         2
```

```
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)
```



#Exercise 14 - Create and manipulate a list

```r
#a. List creation
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

listStruct = list(X = x, Y=y, Slope=slope, Pval=p.value)
listStruct
```

```
## $X
## [1]  4  5  6  7  8  9 10
##
## $Y
## [1] 34 35 41 40 45 47 51
##
## $Slope
## [1] 2.82
##
## $Pval
## [1] 0.000131
```

```r
#b. Grab the second value of the list (which is y)
listStruct[2]
```

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

```r
#c. Grab our p-value
listStruct['Pval']
```

```
## $Pval
## [1] 0.000131
```

#Exercise 15 - Examine the data structures used with linear models lm()

```r
#a. Load the cherry trees dataset
data(trees)

#b. Examine the data frame using str()
str(trees)
```

```
## 'data.frame':    31 obs. of  3 variables:
##  $ Girth : num  8.3 8.6 8.8 10.5 10.7 10.8 11 11 11.1 11.2 ...
##  $ Height: num  70 65 63 72 81 83 66 75 80 75 ...
##  $ Volume: num  10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 ...
```

```r
#c. Perform a regression relating volume of lumber to the girth and height of the tree
m = lm(Volume ~ Girth + Height, data=trees)

#d. Use the str command to inspect the model m. Extract the coefficients from the list.
str(m)
```

```
## List of 12
##  $ coefficients : Named num [1:3] -57.988 4.708 0.339
##   ..- attr(*, "names")= chr [1:3] "(Intercept)" "Girth" "Height"
##  $ residuals    : Named num [1:31] 5.462 5.746 5.383 0.526 -1.069 ...
##   ..- attr(*, "names")= chr [1:31] "1" "2" "3" "4" ...
##  $ effects      : Named num [1:31] -167.985 87.073 10.118 -0.812 -1.489 ...
```

```
##   ..- attr(*, "names")= chr [1:31] "(Intercept)" "Girth" "Height" "" ...
##  $ rank         : int 3
##  $ fitted.values: Named num [1:31] 4.84 4.55 4.82 15.87 19.87 ...
##   ..- attr(*, "names")= chr [1:31] "1" "2" "3" "4" ...
##  $ assign       : int [1:3] 0 1 2
##  $ qr           :List of 5
##   ..$ qr   : num [1:31, 1:3] -5.57 0.18 0.18 0.18 0.18 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:31] "1" "2" "3" "4" ...
##   .. .. ..$ : chr [1:3] "(Intercept)" "Girth" "Height"
##   .. ..- attr(*, "assign")= int [1:3] 0 1 2
##   ..$ qraux: num [1:3] 1.18 1.23 1.24
##   ..$ pivot: int [1:3] 1 2 3
##   ..$ tol  : num 1e-07
##   ..$ rank : int 3
##   ..- attr(*, "class")= chr "qr"
##  $ df.residual  : int 28
##  $ xlevels      : Named list()
##  $ call         : language lm(formula = Volume ~ Girth + Height, data = trees)
##  $ terms        :Classes 'terms', 'formula'  language Volume ~ Girth + Height
##   .. ..- attr(*, "variables")= language list(Volume, Girth, Height)
##   .. ..- attr(*, "factors")= int [1:3, 1:2] 0 1 0 0 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:3] "Volume" "Girth" "Height"
##   .. .. .. ..$ : chr [1:2] "Girth" "Height"
##   .. ..- attr(*, "term.labels")= chr [1:2] "Girth" "Height"
##   .. ..- attr(*, "order")= int [1:2] 1 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. ..- attr(*, "predvars")= language list(Volume, Girth, Height)
##   .. ..- attr(*, "dataClasses")= Named chr [1:3] "numeric" "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:3] "Volume" "Girth" "Height"
##  $ model        :'data.frame':   31 obs. of  3 variables:
##   ..$ Volume: num [1:31] 10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 ...
##   ..$ Girth : num [1:31] 8.3 8.6 8.8 10.5 10.7 10.8 11 11 11.1 11.2 ...
##   ..$ Height: num [1:31] 70 65 63 72 81 83 66 75 80 75 ...
##   ..- attr(*, "terms")=Classes 'terms', 'formula'  language Volume ~ Girth + Height
##   .. .. ..- attr(*, "variables")= language list(Volume, Girth, Height)
##   .. .. ..- attr(*, "factors")= int [1:3, 1:2] 0 1 0 0 0 1
##   .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. ..$ : chr [1:3] "Volume" "Girth" "Height"
##   .. .. .. .. ..$ : chr [1:2] "Girth" "Height"
##   .. .. ..- attr(*, "term.labels")= chr [1:2] "Girth" "Height"
##   .. .. ..- attr(*, "order")= int [1:2] 1 1
##   .. .. ..- attr(*, "intercept")= int 1
##   .. .. ..- attr(*, "response")= int 1
##   .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. .. ..- attr(*, "predvars")= language list(Volume, Girth, Height)
##   .. .. ..- attr(*, "dataClasses")= Named chr [1:3] "numeric" "numeric" "numeric"
##   .. .. .. ..- attr(*, "names")= chr [1:3] "Volume" "Girth" "Height"
##  - attr(*, "class")= chr "lm"
```

```
m['coefficients']
```

```
## $coefficients
## (Intercept)       Girth      Height
## -57.9876589   4.7081605   0.3392512
```