assignment8

Noah Plant

2024-09-30

Excersise 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
```

Excersise 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
```

It looks like in this instance R added up the vectors element wise. And for the elements that C had that D didnt, R just did nothing.

Excersise 3

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

```
vec_a+5
```

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

In this instance R treated 5 like the vector (5,5,5), and did not give a warning. This is because R treats the scalar object 5 differently then a vector object.

Excersise 5

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

```
vec_5a=seq(from=2, to=20, by=2)

vec_5a
```

```
##  [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.*

```
vec_5b<-1:10

vec_5b<-vec_5b*2

vec_5b
```

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

Excersise 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,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
```

Excersise 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_8<-c(2,4,8)

vec_8<-rep(vec_8,each=4)
vec_8
```

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

Excersise 9

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)
)

my.trees
```

```
##   Girth Height Volume
## 1   8.3     70   10.3
## 2   8.6     65   10.3
## 3   8.8     63   10.2
## 4  10.5     72   16.4
## 5  10.7     81   18.8
## 6  10.8     83   19.7
## 7  11.0     66   15.6
```

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

```
my.trees[3,]
```

```
##   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.trees2<-my.trees[-c(4),]

my.trees2
```

```
##   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)
```

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

```
my.trees[index,]
```

```
##   Girth Height Volume
## 4  10.5     72   16.4
## 5  10.7     81   18.8
## 6  10.8     83   19.7
## 7  11.0     66   15.6
```

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

```
my.trees[-c(index),]
```

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

Excersise 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.

```
``` r
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) ), ]
```
```

In the first line, the program is finding all the rows without a grade and displaying all the rows except that one.

In the second line the program is finding all the rows with a grade and displaying those rows.

The difference here is subtle, one line uses R's - feature to basically remove rows, while the second line is using the not operator ! to select for the lines that do have grades.

Excersise 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

```
  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.val=x,y.val=y,slope=slope,p.val=p.value)

  str(my.test)
```

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

b) Extract the second element in the list.

```
my.test[2]
```

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

c) Extract the element named ‘p.value‘ from the list.

```
my.test['p.val']
```

```
## $p.val
## [1] 0.000131
```