HW: Week 1

36-350 – Statistical Computing

Week 1 – Spring 2022

Name: Ethan Vertal

Andrew ID: evertal

You must submit **your own** homework as a PDF file on Gradescope.

In this week's homework you will learn about concepts that we did not cover in the lecture notes: sampling, recycling, and attributes. You are encouraged to use the documentation in the `R Studio` help pane as well as to `Google` terms. As always, `StackOverflow` is your friend.

Sampling

For Q1-Q2, use the `R` functions `sample()` or `sample.int()`. Refer to the documentation for both (which is on a single help page). In some cases, you can use either of them to answer a question, and in some cases you can use only one.

Question 1

(10 points)

Sample five integers between 1 and 100, inclusive. Then demonstrate how you would sample five integers between -5 and 15, inclusive.

```
sample.int(100, size = 5)
```

```
## [1] 10 50 22 80 59
```

```
sample(-5:15, size = 5)
```

```
## [1] -3  5 -4 15  6
```

Question 2

(10 points)

If you rerun the code chunk for Q1, you will see that you get different sets of five numbers every time. That's because the sampling is random. The details of how `R` performs random sampling are beyond the scope of this class (see, e.g., the documentation for `Random` if you are of a masochistic bent). However, it is generally important to be able to reproduce statistical analyses, and so it is generally important that we set a random number seed before calling `sample()` or `sample.int()`. That seed can be any (reasonable) number.

Use `set.seed()` to set the random number seed, and call `sample()`, then reset the seed to the same value as before and re-do the call to `sample()`. If you do this correctly you will see the same output from `sample()` in both cases.

```
set.seed(1)
sample(10)
```

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

```
set.seed(1)
sample(10)
```

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

Recycling

Question 3

(10 points)

Define a numeric vector x of length 2 and a numeric vector y of length 6. Multiply them. Write down how the observed result was obtained.

```
x <- c(2, 3)
y <- c(2, 3, 4 ,5 ,6, 7)
x * y
```

```
## [1]  4  9  8 15 12 21
```

```
The shorter vector is iterated upon until the longer vector has been fully multiplied.
```

What you observed when answering Q3 is "recycling". If the lengths of the two vectors in a mathematical operation do not match, then the shorter vector is repeated until its length matches that of the larger vector. In my own opinion, this is maddening behavior (since it can lead to relatively hard-to-trace bugs in your code), but no one asked me when `R` was being designed.

Note that if the length of the longer vector is not a multiple of the length of the shorter vector, a warning will be issued, but recycling will still occur.

Question 4

(10 points)

Use recycling to define a 4 x 5 matrix where the first three rows are zero and the fourth row contains the first five multiples of 4 (i.e., 4, 8, 12, 16, 20). You can do this in a single line.

```
m <- matrix(1:20 * c(0,0,0,1), nrow = 4, ncol = 5)
m
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    0    0    0    0
## [2,]    0    0    0    0    0
## [3,]    0    0    0    0    0
## [4,]    4    8   12   16   20
```

Attributes

Attributes in `R` are metadata that one can attach to any defined object. Some objects come with attributes already defined, but in all cases you can define your own.

- To list attributes, use the function `attributes()`.
- To define attributes, use the function `attr()`.

Question 5

(10 points)

Define a 2x3 matrix x. What is the predefined attribute? Access the second value of that attribute.

```
x <- matrix(1:6,nrow=2,ncol=3)
attributes(x)
```

```
## $dim
## [1] 2 3
```

```
attributes(x)$dim[2]
```

```
## [1] 3
```

```
The predefined attribute is dim
```

Question 6

(10 points)

Using the `Sys.timezone()` function, define an attribute `timezone` for the matrix x. Display the attributes of x to ensure that `timezone` is one of them.

```
attr(x, 'timezone') <- Sys.timezone()
attributes(x)
```

```
## $dim
## [1] 2 3
##
## $timezone
## [1] "America/New_York"
```

Question 7

(10 points)

Remove the `timezone` attribute from x. Display the attributes of x to ensure that `timezone` is not one of them.

```
attr(x, 'timezone') <- NULL
attributes(x)
```

```
## $dim
## [1] 2 3
```

Handy Vector Functions

Here we define some vectors:

```
set.seed(1201)
u = sample(100,100,replace=TRUE)
v = sample(100,100,replace=TRUE)
```

Question 8

(10 points)

Construct and display a single (sorted!) vector that shows the complement of the intersection of `u` and `v`. Then compute and display how many total elements of `u` and `v` are in the complement. (Hint: for the second part, consider using `is.element()` to determine if the elements of the concatenated vector that combines `u` and `v` is in the complement. Your final answer should be 83.)

```
sort(setdiff(union(u,v), intersect(u,v)))
```

```
## [1] 1 3 5 9 10 12 19 20 23 24 28 33 34 35 36 38 39 44 47
## [20] 49 55 57 60 63 64 65 67 70 72 75 78 80 82 85 89 93 94 95
## [39] 97 99 100
```

```
sum(is.element(c(u,v), setdiff(union(u,v), intersect(u,v))))
```

```
## [1] 66
```

Question 9

(10 points)

In Q29 of the lab, you displayed a table that shows how often each value of `v` appears. The same vector `v` is defined above. (We are able to replicate it because we use `set.seed()` with the same seed as in the lab and we then lay out the same sequence of vector definitions as we had in the lab. If we didn't include the definition of `u` here, then `v` would not match what we had in the lab. Reproducibility is a Good Thing™.) Use `R` code to extract, as a number (meaning: not as a character string), the value that was repeated the most times in `v`. (Hints: use `table()`; note that `which.max()` gives you index of a vector's maximum value, like the maximum value within a table; `names()` allows you the extract the values of the original vector, when applied to the output of `table()`.)

```
as.numeric(names(which.max(table(v))))
```

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

Question 10

(10 points)

In Q33 of the lab, you displayed a table showing how many values are in `v` but not in `u` fell into the bins [1,50] and [51,100]. Here: show your answer again, then add a line of code in which you confirm your answer using a relational operator (like >) and `sum()`. (This is meant to reinforce the use of relational operators.)

```
table(findInterval(setdiff(v, u), c(51,101)))
```

```
##
## 0 1
## 9 9
```

```
sum(is.element(setdiff(v,u)[setdiff(v,u) > 50],setdiff(v,u)))
```

```
## [1] 9
```

Question 11

(10 points)

Write down an expression that returns `FALSE` if the union of `u` and `v` has 100 elements and `TRUE` otherwise.

```
length(union(u,v)) != 100
```

```
## [1] TRUE
```

Question 12

(10 points)

Display the (sorted) values of `v` that do not appear in `u`.

```
sort(setdiff(v,u))
```

```
##  [1]  5 12 23 28 33 35 36 47 49 57 64 65 67 78 80 93 94 95
```