INTRODUCCIÓN A LA GEOCOMPUTACIÓN CON R
EJERCICIOS NO. 4 (OBJETOS: DATA.FRAME Y LIST)

INDER TECUAPETLA-GÓMEZ

1. Considera los vectores `a <- rpois(6,4)`, `b <- rep(NA, 6)`, `c <- rep(c(FALSE, TRUE), 3)` y `d <- c("Aguascalientes", "Baja California", "Campeche", "Coahuila", "Colima", "Chiapas")`.
- **(a)** Crea un `data.frame` a partir de los vectores `a`,`b`,`c` y `d`
- **(b)** Asigna nombres descriptivos a las columnas del `data.frame` que definiste arriba
- **(c)** Extrae alguna columna del `data.frame` utilizando el nombre de la columna
- **(d)** Extrae el primer y tercer renglón junto con la segunda y cuarta columna de tu `data.frame`
- **(e)** Agrega el vector `pilon <- rnorm(6,mean=10)` como una columna de tu `data.frame`

2. Considera el `data.frame` que definiste en el ejercicio anterior.
- **(a)** ¿Puedes generar código para contar el número de valores `NA` en el `data.frame`?
- **(b)** ¿Puedes generar código para reemplazar los valores `NA` de tu `data.frame` por el número `pi`?

3. En este ejercicio usamos el dataset `trees`, el cual está integrado en tu versión de `R`.
- **(a)** ¿Es `trees` un objeto `data.frame`?
- **(b)** ¿Qué obtienes al aplicar las funciones `colSums`, `rowSums`, `colMeans` y `rowMeans` a `trees`?
- **(c)** ¿Puedes usar la función `apply` y obtener los resultados que obtuviste en el inciso anterior?
- **(d)** A partir de `trees` crea el siguiente `data.frame`:

```
##                Girth Height    Volume
## mean_tree  13.24839     76  30.17097
## min_tree    8.30000     63  10.20000
## max_tree   20.60000     87  77.00000
## sum_tree  410.70000   2356 935.30000
```

  Hint.: `mean_tree`, `min_tree`, `max_tree` y `sum_tree` se refieren a la media, mínimo, máximo y suma de las variables `Girth`, `Height` y `Volume`, respectivamente.
- **(e)** Ordena el `data.frame` que acabas de definir usando la primera columna. **Hint:** `?order`.
- **(f)** Cambia el nombre de los renglones por `mean, min, max, sum`

4. Crea el `data.frame XY`:

```
##   X Y
## 1 1 0
## 2 2 3
## 3 3 2
## 4 1 0
## 5 4 5
## 6 5 9
## 7 2 3
```

- **(a)** Busca los elementos duplicados. **Hint:** `?duplicated`
- **(b)** Quédate únicamente con los renglones no duplicados de `XY`. **Hint:** Una posible solución es:

```
##   X Y
## 1 1 0
## 2 2 3
## 3 3 2
## 5 4 5
## 6 5 9
```

5. En este ejercicio usamos el dataset `airquality`, el cual está integrado en tu versión de `R`.
   (a) ¿Cuántas variables tiene este dataset?
   (b) ¿Cuántas observaciones tiene cada variable?
   (c) ¿Puedes calcular el mínimo, la media, la mediana y el máximo de cada variable?

6. En este ejercicio usamos el dataset `airquality`, el cual está integrado en tu versión de `R`.
   (a) Remueve las variables `Solar.R` y `Wind`
   (b) ¿Puedes reemplazar los valores `NA` de la variable `Ozone` por su mediana?
   (c) ¿Puedes calcular el mínimo, la media, la mediana y el máximo de cada variable? Observas algún cambio con el resultado obtenido en **5 (c)**, en particular en referencia a la variable `Ozone`.

7. Si `x <- list(a=5:10, c="Hola", e="AA")`. ¿Puedes escribir una instrucción para agregar el elemento `z="NewItem"` a la lista `x`? ¿Puedes escribir una instrucción para obtener la longitud (`length`) del vector `a` de la lista `x`?

8. Si `p <- c(2,7,8)`, `q <- c("A", "B", "C")` y `x <- list(p,q)`, ¿cuáles son las diferencias entre `x[2]` y `x[[2]]`?

9. Si `w <- c(2,7,8)`, `v <- c("A","B","C")` y `x <- list(w,v)`, entonces ¿qué instrucción permite reemplazar `"A"` por `"K"` en `x`?
   (a) `x[2] <- "K"`
   (b) `x[2][1] <- "K"`
   (c) `x[[2]][1] <- "K"`
   (d) `x[[2]][[1]] <- "K"`

10. Si `a <- list("x"=5, "y"=10, "z"=5)`, entonces ¿qué instrucción debes usar para sumar todos los elementos numéricos en `a`?
    (a) `sum(a)`
    (b) `sum(list(a)`
    (c) `sum(unlist(a))`

11. Si `newList <- list(a=1:10, b="Buenas, buenas!", c=NA)`. ¿Puedes escribir una instrucción para sumar el número 1 a cada elemento del primer vector en `newList`? ¿Puedes escribir una instrucción que devuelva todos los elementos, excepto el segundo, del primer vector de `a`?

12. Si `y <- list("a", "b", "c")` y `q <- list("A", "B", "C", "a", "b", "c")`, ¿puedes escribir una instrucción para obtener todos los elementos de la lista `q` que *no* están en la lista `y`? **Hint:** Una posible solución es:

```
## [[1]]
## [1] "A"
##
## [[2]]
## [1] "B"
##
## [[3]]
## [1] "C"
```