Introducción a R para Ciencias C1-S3-clase2

Contents

1 Práctica de Casa 02 - Programación con R y dplyr

Last compiled: 2021-03-12

Link de la práctica desarrollada: Introducción a R para Ciencias C1-S3-clase2 (irwingss.github.io)

Realiza los siguientes ejercicios durante tu tiempo libre para mejorar tus habilidades de manejo del lenguaje de programación R.

Recuerda realizar esta práctica luego de haber aprendido la información de la clase 1 y 2 de la semana 3:

- R-Notebook-C1-S3-clase1y2.Rmd

  Nota: Si necesitas crear un code chunk los atajos en el teclado son en WINDOWS: `Crtl+Alt+i`, y en MAC: `Command+Alt+i`.

```
library(tidyverse)
library(gapminder)
```

1.1 Ejercicio 1:

Calcula el promedio del GDP Percápita (columna `gdpPercap`) para cada continente de la base de datos `gapminder`. Crea un resumen estadístico que contenga estos nuevos datos en la columna llamada `pib_percapita_prom`. Asegúrate de reordenar la tabla final para que muestre el valor más alto en la primera fila. Asigna el resultado final a `gdp_continentes`.

```
data(gapminder)

gdp_continentes <- gapminder %>%
  group_by(continent) %>%
  summarise(pib_percapita_prom = mean(gdpPercap)) %>%
  arrange(desc(pib_percapita_prom))

gdp_continentes
```

```
## # A tibble: 5 x 2
##   continent pib_percapita_prom
##   <fct>                  <dbl>
## 1 Oceania               18622.
## 2 Europe                14469.
## 3 Asia                   7902.
```

```
## 4 Americas               7136.
## 5 Africa                 2194.
```

¿Qué continente tiene el mayor Producto Interno Bruto per cápita promedio?

Rpta/. Oceanía.

1.2 Ejercicio 2:

1.2.1 match operator %in%

El realizar filtrados es muy útil. Cuando deseamos filtrar un valor dentro de una columna usamos una estructura lógica similar a `COL == "Valor"`. Sin embargo, algunas veces es necesario buscar más de un valor en la columna. En esos casos uno imaginaría que es útil hacer algo como `COL == c("Valor1", "Valor2", "Valor3")`. Veamos si es correcto.

```
# Verifica la cantidad de filas que existe
# para Perú y Ecuador en la base de datos gapminder
# Usa el operador lógico "o" | para definir dos condiciones
# que deben de cumplirse, una o la otra, para filtrar las filas
gapminder %>% filter(country == "Peru" | country == "Ecuador") %>%
  group_by(country) %>%  count(country)
```

```
## # A tibble: 2 x 2
## # Groups:   country [2]
##   country     n
##   <fct>   <int>
## 1 Ecuador    12
## 2 Peru       12
```

```
# Utilicemos la estructura COL == c("Valor1", "Valor2")
gapminder %>% filter(country == c("Peru","Ecuador")) %>%
  group_by(country) %>%  count(country)
```

```
## # A tibble: 2 x 2
## # Groups:   country [2]
##   country     n
##   <fct>   <int>
## 1 Ecuador     6
## 2 Peru        6
```

Claramente, esta estructura no funciona. Nos indica 6 filas para cada país, en lugar de 12 que son las que tiene cada uno. Siempre que quieras buscar más de un valor en una condición lógica, evita el uso de `==` y usa en su lugar `%in%`. El operador `%in%` busca cada elemento del vector que le proporcionemos a la derecha, en el conjunto de datos que le brindemos a la izquierda. Por ejemplo, `COL %in% c("A", "B")` busca `A` en `COL` y luego busca `B` en `COL`, filtrando todas las filas que cumplan con una u otra búsqueda. **Por tanto, `%in%` reemplaza una secuencia de condiciones lógicas "o" `|`.**

```
# Utilicemos el operador %in%
gapminder %>% filter(country %in% c("Peru","Ecuador")) %>%
  group_by(country) %>%  count(country)
```

```
## # A tibble: 2 x 2
## # Groups:   country [2]
##   country     n
##   <fct>   <int>
## 1 Ecuador    12
## 2 Peru       12
```

```
# Incluso podemos usar un index
index <- c("Ecuador","Peru","Chile","Mexico","Colombia")

gapminder %>% filter(country %in% index) %>%
  group_by(country) %>%  count(country)
```

```
## # A tibble: 5 x 2
```

```
## # Groups:   country [5]
##   country      n
##   <fct>    <int>
## 1 Chile       12
## 2 Colombia    12
## 3 Ecuador     12
## 4 Mexico      12
## 5 Peru        12
```

Ahora aplica lo aprendido para calcular el promedio de vida (columna `lifeExp`) de cada uno de los paises (columna `country`) mencionados en el `index` para `gapminder`.

```
gapminder %>% filter(country %in% index) %>%
  group_by(country) %>%
  summarise(prom = mean(lifeExp)) %>%
  arrange(desc(prom))
```

```
## # A tibble: 5 x 2
##   country   prom
##   <fct>    <dbl>
## 1 Chile     67.4
## 2 Mexico    65.4
## 3 Colombia  63.9
## 4 Ecuador   62.8
## 5 Peru      58.9
```

¿Qué país, de los seleccionados, tiene el mayor Producto Interno Bruto per cápita promedio?

Rpta/. Chile.

Filtrado de valores que "Comienzan con la letra ..."
Esto a veces es una gran necesidad. Aprenderemos a filtrar valores que comienzan con una letra usando la función `substr()` para crear una condición lógica dentro de `filter()`. Primero conoce como usar `substr()`.

```
# Crea un vector con tres países
paises <- c("República dominicana", "Costa Rica", "Colombia")

# Ejecuta substr para que veas qué hacen los argumentos
# start y stop de substr(). Cambia los números si gustas
substr(paises, start = 1, stop = 1)
```

```
## [1] "R" "C" "C"
```

```
substr(paises, start = 1, stop = 3)
```

```
## [1] "Rep" "Cos" "Col"
```

```
substr(paises, start = 3, stop = 5)
```

```
## [1] "púb" "sta" "lom"
```

La función `substr()` le aplica lo mismo a todos los elementos de un vector de tipo carácter (o una columna de tipo carácter dentro de una tabla), indicando en qué posición dentro de cada elemento iniciar a mostrar los caracteres (`start`) y dónde detenerse (`stop`). `start =1, stop = 2` significaría: muéstrame desde la primera hasta la segunda letra de cada elemento del conjunto de datos.

Podemos usar esta función como *el lado izquierdo* de una condición lógica con `%in%`. Indica que busque, por ejemplo, la letra `"A"` en el conjunto de datos "recortado" con `substr()` que muestre sólo la primera letra del texto en cada fila en la columna `continent`.

```
gapminder %>%
  filter(substr(continent, start = 1, stop = 1) %in% "A") %>%
  distinct(continent)
```

```
## # A tibble: 3 x 1
##   continent
##   <fct>
## 1 Asia
## 2 Africa
## 3 Americas
```

Ahora aplica lo aprendido, filtra la base de datos para que solo aparezcan los países que comienzan con "Co".

```
gapminder %>%
  filter(substr(country, start = 1, stop = 2) %in% "Co") %>%
  distinct(country)
```

```
## # A tibble: 6 x 1
##   country
##   <fct>
## 1 Colombia
## 2 Comoros
## 3 Congo, Dem. Rep.
## 4 Congo, Rep.
## 5 Costa Rica
## 6 Cote d'Ivoire
```

¿Cuántos nombres de países inician con la sílaba "Co"?

Rpta/. Son seis países.

1.3 Ejercicio 3:

Llama a la base de datos `airquality` y calcula el promedio y la desviación estándar de las primeras 4 columnas de la base de datos, agrupando los datos por mes (columna `Month`). Asigna el resultado con el nombre `prom.meses`. Redondea los valores numéricos de `prom.meses` a dos decimales con la función `round()`

```
# Escribe tu código aquí
prom.meses <- airquality %>%  group_by(Month) %>%
  select(1:4) %>%  summarise_all(funs(mean), na.rm=TRUE)
```

```
## Adding missing grouping variables: `Month`
```

```
# Visualiza prom.meses
round(prom.meses[2:5])
```

```
## # A tibble: 5 x 4
##   Ozone Solar.R  Wind  Temp
##   <dbl>   <dbl> <dbl> <dbl>
## 1    31     167    10    77
## 2    60     172     9    84
## 3    59     216     9    84
## 4    29     190    10    79
## 5    24     181    12    66
```

Modifica el tibble `prom.meses` convirtiendo la columna `Month` a factor y cambiando los valores numéricos por el nombre de cada mes según corresponda.

```
# Escribe tu código aquí
prom.meses$Month <- factor(prom.meses$Month, labels = c("5"="Mayo",
                                                        "6"="Junio",
                                                        "7"="Julio",
                                                        "8"="Agosto",
                                                        "9"="Setiembre"))
```

```
# Visualiza prom.meses
prom.meses
```

```
## # A tibble: 5 x 5
##   Month     Ozone Solar.R  Wind  Temp
##   <fct>     <dbl>   <dbl> <dbl> <dbl>
## 1 Mayo       31.4    167. 10.2   76.9
## 2 Junio      60.0    172.  8.79  84.0
## 3 Julio      59.1    216.  8.94  83.9
## 4 Agosto     29.4    190. 10.3   79.1
## 5 Setiembre  23.6    181. 11.6   65.5
```

1.4 Ejercicio 4:

Une los valores de la columna `Month` y `Day` de la base de datos `airquality` en una nueva columna llamada `Mes_Dia`, consignando el separador el guión bajo `"_"`. Usa alguna de las funciones de la parte de tablas anchas o largas de la clase, y usa pipe lo más que puedas en tu código. Filtra las filas que tengan valores de radiación solar (columna `Solar.R`) mayores a 250 y valores de ozono (columna `Ozone`) mayores a 40.

```
# Escribe tu código aquí
airquality %>%
  unite(Mes_Dia, Month, Day, sep="_") %>%
  filter(Solar.R > 250 & Ozone > 40)
```

```
##    Ozone Solar.R Wind Temp  Mes_Dia
## 1     45     252 14.9   81  Mayo_29
## 2     71     291 13.8   90  Junio_9
## 3    135     269  4.1   84  Julio_1
## 4     77     276  5.1   88  Julio_7
## 5     97     267  6.3   92  Julio_8
## 6     97     272  5.7   92  Julio_9
## 7     48     260  6.9   81 Julio_16
## 8     61     285  6.3   84 Julio_18
## 9     80     294  8.6   86 Julio_24
## 10    50     275  7.4   86 Julio_29
## 11    64     253  7.4   83 Julio_30
## 12    59     254  9.2   81 Julio_31
## 13   122     255  4.0   89 Agosto_7
```

¿A qué mes pertenecen la mayor cantidad de observaciones con estas características de radiación solar y ozono?

Rpta/. Al mes 7

1.5 Ejercicio 5:

Convierte la base de datos airquality a tabla larga, apilando todas las columnas numéricas y manteniendo las columnas `Month` y `Day`. Asignale el nombre `t_larga`. Una vez guardada la variable úsala para filtra la variable `Ozone` del mes `8` y calcula el promedio de dicho conjunto de datos.

```
# Crea la variable t_larga
t_larga <- gather(airquality, Variables, Valores, -Month, -Day)

# Visualiza la variable t_larga
View(t_larga)

# Realiza el filtrado
t_larga %>%
  filter(Month=="8" & Variables == "Ozone") %>%
  summarise(promedio = mean(Valores, na.rm=TRUE))
```

```
##   promedio
## 1      NaN
```