Carrera Tidyverse. Preguntas Cortas.MARCELINO MARTINEZ

Tratamiento de Datos. Grado en Ciencia de Datos- UV

Pablo Quesada

2024-03-27

1 Usa `pacman` para cargar las librerías e instalarlas si no están disponibles

Todos los ejercicios planteados están relacionados con el uso de las librerías **dplyr, tidyr y lubridate**. Familiarízate con el uso del operador %>% para facilitar la compresión del código. Los ejercicios consideran que las librería están instaladas y cargadas.

2 Fichero de datos.

Todas las preguntas están referidas al fichero **2024-03-12.Rdata**, que contiene información de los registros (logs) de las descargas realizadas del repositorio CRAN en esa fecha. La información contenida en el fichero es la siguiente:

- date
- time (in UTC)
- size (in bytes)
- r_version, version of R used to download package
- r_arch (i386 = 32 bit, x86_64 = 64 bit)
- r_os (darwin9.8.0 = mac, mingw32 = windows)
- package
- country, two letter ISO country code. Geocoded from IP using [MaxMind's][https://dev.maxmind.com/geoip/geoip2/geolite2/] free database
- ip_id, a daily unique id assigned to each IP address

El material está basado en el curso 'Getting and Cleaning data' de la librería swirlr

Se pueden descargar ficheros actuales del `logs` usando la librería https://r-hub.github.io/cranlogs/

1. Carga el fichero de datos y almacénalo en un data frame llamado `mydf`. Asegúrate que la opción `stringsAsFactors=FALSE`. Mira las dimensiones y los primeros datos.

Nota: El siguiente bloque puede tardar en ejecutarse por lo que solo lo ejecutaremos una vez. Después usaremos la selección de registros almacenada en el fichero Rdata.

```r
# Fuente de datos http://cran-logs.rstudio.com/
mydf <- read_csv("./data/2024-03-12.csv")

set.seed(seed=14032024)
N<-200000
I<-sample(1:nrow(mydf),size=N,replace=FALSE)
mydf<-mydf[I,]
# Guardamos en formato binario, la primera vez que cargamos
save(file = './data/2024-03-12.Rdata',list=c('mydf'))
```

Después de la primera carga poner **eval=FALSE** en el *chunk* anterior para acelerar la ejecución.

```r
# Cargamos el fichero binario
load(file ='./data/2024-03-12.Rdata')
dim(mydf)
```

```
[1] 200000     10
```

```r
head(mydf)
```

```
# A tibble: 6 x 10
  date       time        size r_version r_arch r_os      package      version country ip_id
  <date>     <time>     <dbl> <chr>     <chr>  <chr>     <chr>        <chr>   <chr>   <dbl>
1 2024-03-12 13:20:36  103412 4.3.3     x86_64 mingw32   callr        3.7.5   <NA>       27
2 2024-03-12 17:42:41  526554 4.3.3     x86_64 mingw32   jquerylib    0.1.4   AT      22280
3 2024-03-12 19:42:22 1280374 4.1.2     x86_64 linux-gnu colourpicker 1.3.0   ET      16115
4 2024-03-12 12:04:00  375281 <NA>      <NA>   <NA>      devtools     2.4.5   GB          3
5 2024-03-12 01:29:45  160694 <NA>      <NA>   <NA>      glue         1.7.0   US       6190
6 2024-03-12 00:30:10  227646 4.2.0     x86_64 mingw32   magrittr     2.0.3   CN      42486
```

```r
mydf %>% head
```

```
# A tibble: 6 x 10
  date       time        size r_version r_arch r_os      package      version country ip_id
  <date>     <time>     <dbl> <chr>     <chr>  <chr>     <chr>        <chr>   <chr>   <dbl>
1 2024-03-12 13:20:36  103412 4.3.3     x86_64 mingw32   callr        3.7.5   <NA>       27
2 2024-03-12 17:42:41  526554 4.3.3     x86_64 mingw32   jquerylib    0.1.4   AT      22280
3 2024-03-12 19:42:22 1280374 4.1.2     x86_64 linux-gnu colourpicker 1.3.0   ET      16115
4 2024-03-12 12:04:00  375281 <NA>      <NA>   <NA>      devtools     2.4.5   GB          3
5 2024-03-12 01:29:45  160694 <NA>      <NA>   <NA>      glue         1.7.0   US       6190
6 2024-03-12 00:30:10  227646 4.2.0     x86_64 mingw32   magrittr     2.0.3   CN      42486
```

```r
mydf %>% tail
```

```
# A tibble: 6 x 10
  date       time        size r_version r_arch r_os      package     version  country ip_id
  <date>     <time>     <dbl> <chr>     <chr>  <chr>     <chr>       <chr>    <chr>   <dbl>
1 2024-03-12 16:26:54  496402 3.6.3     x86_64 linux-gnu MASS        7.3-53   US      63154
2 2024-03-12 09:32:38  326029 4.3.2     x86_64 mingw32   coda        0.19-4.1 GB      65223
3 2024-03-12 13:36:34  566545 <NA>      <NA>   <NA>      tibble      3.2.1    US       2078
4 2024-03-12 13:44:53  812880 4.2.1     x86_64 mingw32   htmlwidgets 1.6.4    DE      58160
5 2024-03-12 06:12:46 2884416 <NA>      <NA>   <NA>      Matrix      1.6-5    US       1177
6 2024-03-12 01:15:13  201099 <NA>      <NA>   <NA>      pkgbuild    1.4.3    US       3346
```

```
mydf %>% str
```

```
tibble [200,000 x 10] (S3: tbl_df/tbl/data.frame)
 $ date     : Date[1:200000], format: "2024-03-12" "2024-03-12" "2024-03-12" ...
 $ time     : 'hms' num [1:200000] 13:20:36 17:42:41 19:42:22 12:04:00 ...
  ..- attr(*, "units")= chr "secs"
 $ size     : num [1:200000] 103412 526554 1280374 375281 160694 ...
 $ r_version: chr [1:200000] "4.3.3" "4.3.3" "4.1.2" NA ...
 $ r_arch   : chr [1:200000] "x86_64" "x86_64" "x86_64" NA ...
 $ r_os     : chr [1:200000] "mingw32" "mingw32" "linux-gnu" NA ...
 $ package  : chr [1:200000] "callr" "jquerylib" "colourpicker" "devtools" ...
 $ version  : chr [1:200000] "3.7.5" "0.1.4" "1.3.0" "2.4.5" ...
 $ country  : chr [1:200000] NA "AT" "ET" "GB" ...
 $ ip_id    : num [1:200000] 27 22280 16115 3 6190 ...
```

```
mydf %>% dim
```

```
[1] 200000     10
```

2. Para facilitar el manejo transforma los datos en un tibble `as_tibble` y almacena el resultado con el nombre `cran`, y borra el data frame original (usa `rm`)
3. Una ventaja de trasformar en tibble es la visualización. Escribe directamente el nombre de la variable y observa la visualización. Posteriormente usa la función `glimpse` sobre `cran` y observa el resultado.

3 dplyr

No almacenes el resultado de las operaciones, salvo que sea necesario

4. Elige únicamente las variables `ip_id, package, country`
5. Elige todas las variables entre `r_arch y country`
6. Puedes usar el signo (-) para no coger una columna determinada. Selecciona todas las columnas menos las que están entre `date y size`
7. Elige aquellos registros en los que se haya descargado el paquete `plotly`, desde un sistema que ejecuta la versión de R 3.5.1
8. Elige aquellas descargas realizadas desde España (ES), con versiones de R superiores a la 3.4.0
9. Elige aquellas descargas realizadas desde España (ES), o Portugal (PT).
10. Elige aquellas descargas cuyo tamaño supera los 100500 bytes desde un sistema operativo `linux-gnu`.
11. Elige aquellas descargas en las que la versión del sistema operativo no está vacía.
12. Almacena en `cran2` todas las columnas entre `size` y `ip_id`.
13. Ordena `cran2` en orden ascendente según la variable `ip_id`.
14. Ordena `cran2` en orden descendente según la variable `ip_id`.
15. Ordena `cran2` según los valores de `package e ip_id`.

16. Ordena `cran2` según los valores de `country` (ascendente), `r_version` (desdencente) e `ip_id`(ascendente).

17. Selecciona el subconjunto formado por las variables, del `cran`, `ip_id, package y size` en este orden y almacénalas en `cran3`.

18. Crea una nueva variable en `cran3` llamada `size_mb` que contenga el tamaño de la descarga expresado en Mbytes ($1Mbyte = 2^{20}$ bytes).

19. Sabiendo que 1Gb son 2^{10} Mbytes, crea una nueva variable en `cran3` llamada `size_gb` que contenga el tamaño de la descarga expresado en Gbytes.

20. Considera que se ha detectado un error en el sistema donde todos los tamaños de las descargas son 1000 bytes menores de lo que corresponde. Crea una nueva variable llamada `correct_size` con el valor correcto.

21. Usa `summarise` para determinar el valor medio del tamaño de las descargas realizadas en `cran` y almacenálo en la variable `avg_bytes`

4 Agrupamientos

19. Crea un agrupamiento del tibble `cran` según la variable `package` y almacénalo en la variable `by_package`. Observa el nuevo elemento creado y aunque aparentemente no hay cambio al principio, se indica que se ha hecho un agrupamiento. Éste agrupamiento se puede deshacer con la función `ungroup`.

20. Usa la función `summarise` sobre el elemento agrupado y calcula el valor medio de la variable `size`. Observa el efecto del agrupamiento en el resultado.

21. Para el agrupamiento creado calcula:

- . count = n()
- . unique = n_distinct(ip_id)
- . countries = n_distinct(country)
- . avg_bytes = mean(size)

número de elementos; `n()`, elementos distintos,`n_distinct()` de la variables `ip_id y country`, y el la mediana (función `stats::median`) de la variable `size` y almacénalos en las variables `count, unique, countries y med_bytes` respectivamente. El resultado del sumario se almacenará en `pack_sum`.

22. Muestra el 1% de los paquetes que han sido más descargados. Para ello puedes utilizar la función `quantile(????, probs=0.99)`. Siendo ??? la variable de interés, que te permitirá estimar el umbral. Los valores por encima serán los que nos interesen. Almacena estos paquetes **top descargas** en la variable `top_counts`. **NOTA: De ahora en adelante concatena todas las operaciones partiendo del elemento `cran`, usando *pipes*.**

23. Repite el apartado anterior utilizando **pipes** para que finalmente el resultado se ordene, en orden descendente, según el número de descargas (`count`) y se almacene el la variable `top_counts_sorted`

24. En lugar de utilizar el recuento de descargas, es más lógico considerar el número de ordenadores distintos en los que se han descargado las librerías, para ello utilizaríamos la columna `unique` que cuenta solo una descarga por ordenador, independientemente del número de veces que se descarge una librería. Repite los cálculos considerando esta variable y almacena el resultado en `top_unique_sorted`.

25. Completa lo que se indica usando `pipes`. Selecciona las columnas `ip_id, country, package, size` de `cran` y muestra el resultado por pantalla usando `print`. No es necesario almacenar.

26. Añade una columna llamada `size_mb` que contenga el tamaño de cada descarga en megabytes.

27. Selecciona aquellas descargas cuyo tamaño es menor o igual que 0.5 Mbytes

28. Repite el apartado anterior ordenando los resultados, en orden descendente según el tamaño en Megabytes y muéstralo por pantalla.

EXTRA. Se desea mostar gráficamente aquellos países que suponen el mayor tráfico de información, ocasionado por el total de descargas acumuladas. Para facilitar la visualización céntrate en aquellos que están por encima del 90 % del top de descargas. El gráfico debe mostrar los valores de forma ordenada de mayor a menor. En el eje X debe aparecer el código de país y en el eje Y el tráfico generado. Intenta hacerlo usando una tubería sin emplear ninguna variable auxiliar. `cran%>%...+geom_point()`

EXTRA Tal como se indica en el ejercicio, el significado de los códigos usados para la codificación de los paise se pueden obtener de https://dev.maxmind.com/geoip/geoip2/geolite2/. Descarga el fichero **GeoLite2 Country** en formato CSV. Este fichero contiene información en varios idiomas. Usa `GeoLite2-Country-Locations-es.csv` que tiene la información en español. Deseamos poder determinar las descargas por continentes por lo que necesitamos añadir esta información a los datos de las descargas almacenados en `cran`. Procede de la siguiente forma:

- Descarga el fichero de códigos de paises `GeoLite2-Country-Locations-es.csv` e impórtalo en el dataframe `code_paises`. **OJO con la codificación**
- Carga y almacena la información del fichero de los paises.
- Elimina todos los registros de `cran` que no incluyan el código del país.
- Elimina los registros de `code_paises` que no incluyen el código del país.
- Combina los dos data frames para añadir a `cran` el código del país y el continente, el resto de variables no son necesarias.
- Representa un diagrama de barras con el flujo de datos, por continente, ordenados de mayor a menor. Expresa el flujo de datos en Terabytes (1Tb=2^40bytes)

5 tidyr.

Carga el fichero de datos `datostidyr.Rdata` que contiene varios conjuntos de datos que vamos a usar en los siguientes ejercicios.

29. Observa el data frame `students` : grade male female 1 A 1 5 2 B 5 0 3 C 5 2 4 D 5 5 5 E 7 4

La columna `grade` es la calificación y las dos siguientes el número de estudiantes hombre y mujer que han recibido dicha calificación. ¿ Cuáles son las variables ?. Transfórmalo en un conjunto `tidy` con las variables, `grade, sex y count`

30. Considera el conjunto ´students2´, que es similar al primero pero que se han recogido datos correspondientes a dos grupos, (`male_1, female_1`, etc). Transfórmalo en un conjunto `tidy` añadiendo una columna adicional correspondiente al grupo (llama a la nueva variable `class`)

31. Considera el conjunto `students3` y observa por qué no es un `tidy data` . Las variables finales del conjunto deberían ser: `name, class, midterm, final`. Dado que un alumno solo puede estar en una de las clases aparecen muchos `NA`, al reorganizar, usa `na.rm=TRUE` en las opciones de `gather`

32. Considera que la columna `class` queremos especificarla con valores, 1,2,3,4 y 5 en lugar de class1, . . . , class5. En librería de importación de datos (`readr`) se proporcionan funciones que permite identificar y extraer diversos tipos de datos. Son las funciones `parse_XXXX`. Observa el efecto de aplicar `parse_number('class5')` y utiliza el resultado para realizar la tarea solicitada.

33. Observa el conjunto `students4` donde se presenta el problema de tener varias observaciones en la misma tabla. Separa el conjunto en dos tablas, una que contenga información relativa a los estudiantes, que llamarás `student_info`, que no debe contener repeticiones, y otra con las calificaciones y la clave primaria que enlaza ambas tablas(`id`), que llamarás `gradebook`.

34. Consideremos el caso de una observación almacenada en varias tablas. Observa los datos almacenados en `failed` y `passed` con la información de los alumnos que suspeden (calificaciones C,D,E,F) y los que aprueban (calificaciones A, B). Debes unir ambos conjuntos en uno que contenga una columna adicional llamada `status` que indicará `passed` o `failed` según la situación de cada alumno, puedes usar la función `bind_rows`.

6 Pongamos todo en común.

35. El SAT es un examen popular de preparación para la universidad en los Estados Unidos que consta de tres partes: lectura crítica, matemáticas y escritura. Los estudiantes pueden obtener hasta 800 puntos en cada parte. Este conjunto de datos presenta el número total de estudiantes, para cada combinación de parte y sexo del estudiante. Considera los datos almacenado en `sat` y haz las transformaciones adecuadas para que las columnas del conjunto `tidy` sean `score_range, part, sex, count` en ese orden. Ten en cuenta que los datos totales no es necesario almacenarlos ya que se pueden generar a posteriori.

36. Utiliza agrupamientos, según las variables `part, sex` y genera dos nuevas columnas que contengan el número total de estudiantes en dicho grupo, y la proporción dentro del grupo. Almacénalas en las variables `total y prop` respectivamente.

37. Representa gráficamente el conjunto obtenido para mostrar las diferencias entre hombres y mujeres, en las diferentes pruebas para cada una de las franjas de calificación.

7 Manejo de fechas con lubridate.

38. La función `today` devuelve la fecha actual. Almacéna la fecha actual en `this_day`, y muéstrala por pantalla. Usa las funciones `year, month y day` para extraer cada una de las partes. `wday` para obtener el día de la semana ()

39. Guarda el instante actual en una variable llamada `this_moment` con la función `now` y muéstralo por pantalla. Usa las funciónes `hour, minute, second` para extraer la información de cada elemento.

40. La librería `lubridate` dispone de múltiples funciones para importar en formato fecha datos con un gran variedad de formatos de entrada `ymd(), dmy(), hms(), ymd_hms()` donde cada letra indica *years (y),months (m), days (d), hours (h), minutes (m), y seconds (s)*. Prueba my_date <- ymd("2019-02-14"), y determina de que clase es `my_date`. Las funciones son muy robustas y pueden intepretar correctamente diversos formato. Prueba `ymd()` para importar la fecha **"1989 May 17"**. ¿Cómo importarías **"March 12, 1975"**?

41. Importa el valor numérico `25081985`, sabiendo que el formato es dia, mes, año, y posteriormente `192012` donde se ha almacenado año, mes y día. Modifica el campo fecha en el segundo caso añadiendo algún carácter separador para eliminar la ambigüedad.

```
fecha<-'19200102'
ymd(fecha)
```

```
[1] "1920-01-02"
```

42. Almacena la cadena `2014-08-23 17:23:02` en la variable `dt1` e impórtala para que incluya la fecha y la hora,posteriormente importa la cadena `"03:22:14"` (hh:mm:ss)

43. Importa el vector de fechas almacenado el `dt2`

44. Podemos reasingar nuevos valores a una variable fecha/hora con la función `update` , por ejemplo: `update(this_moment, hours = 8, minutes = 34, seconds = 55)`. Actualiza la variable `this_moment` creada anteriormente para que contenga la hora actual.

45. **Zonas horarias**. Considera que estás en Nueva York, que planeas visitar a un amigo en Hong Kong y que has perdido tu itinerario, pero sabes que tu vuelo sale de Nueva York a las 17:34 (17:34 horas) pasado mañana. También sabes que el vuelo está programado para llegar a Hong Kong exactamente 15 horas y 50 minutos después de la salida. Vamos a reconstruir el itinerario:

- Genera la fecha actual, con `now` e indica la zona horaria de Nueva York(America/New_York) (la función `OlsonNames()` devuelve todas las zonas horarias) y almacénala en la variable `nyc`.
- Añade dos días y almacénalo en la variable `salida`. Puedes usar la función `days` y sumar directamente. Comprueba que la salida tiene el valor correcto.
- Actualiza la variable salida con la hora real de salida.
- Averigua a qué hora llegarás a Hong Kong, con referencia horaria de NY.
- Averigua a qué hora llegarás a Hong Kong, con referencia horaria local. Usa la función `with_tz` para realizar esta tarea.
- La última vez que viste a tu amigo fue en Singapur el 17 de Junio de 2008. Genera dicha fecha en zona horaria de Singapur y utiliza la funcion `interval` para crear el tramo temporal entre ambas fechas y aplica la función `as.period` sobre el intervalo resultante para calcular exáctamente cuánto tiempo hace desde que os visteis. Cuando consideramos zonas horarias, peridos que incluyen años bisiestos, etc. El manejo de fecha requiere de librerías como **lubridate** para facilitar estas tareas.