Manipulación de datos

HR Analytics: Teoría y Práctica

http://pablohaya.com/contact

09/2018

Importamos los datos

Empleamos la funcion `load()` para cargar la tabla que habíamos preparado en el tema anterior.

```
library(tidyverse)
library(here)
load(here("data","core_dataset_clean.RData"))
```

Filtrar

El verbo `filter()` permite quedarse un subconjunto de las filas de un tabla.

```
filter(df, MaritalDesc == "Single")
```

```
## # A tibble: 127 x 21
##    EmployeeName EmployeeNumber State Zip   DOB          Age
##    <chr>        <chr>          <fct> <fct> <date>     <int>
##  1 Steans, Tyr~ 1302053333     MA    02703 1986-09-01    31
##  2 Singh, Nan   1307059817     MA    02330 1988-05-19    29
##  3 Zamora, Jen~ 1112030816     MA    02067 1979-08-30    38
##  4 Becker, Ren~ 1102024056     MA    02026 1986-04-04    31
##  5 Goble, Tais~ 0905013738     MA    02127 1971-10-23    46
##  6 Horton, Jay~ 1105025718     MA    02493 1984-02-21    33
##  7 Zhou, Julia  1110029732     MA    02148 1979-02-24    38
##  8 Foss, Jason  1192991000     MA    01460 1980-07-05    37
```

Es posible combinar más de un filtro separando las nuevas condiciones por comas.

```
filter(df, MaritalDesc == "Single", Age < 30)
```

```
## # A tibble: 17 x 21
##    EmployeeName EmployeeNumber State Zip   DOB          Age
##    <chr>        <chr>          <fct> <fct> <date>     <int>
##  1 Singh, Nan   1307059817     MA    02330 1988-05-19    29
##  2 Lindsay, Le~ 0602000312     CT    06070 1988-10-05    29
##  3 Cierpiszews~ 1012023295     MA    02044 1988-05-31    29
##  4 Evensen, Ap~ 1107027392     MA    02458 1989-05-06    28
##  5 Gold, Sheni~ 1408069539     MA    02451 1992-06-18    25
##  6 Ivey, Rose   1408069882     MA    01775 1991-01-28    27
##  7 Ndzi, Colom~ 1204033041     MA    02110 1989-05-02    28
##  8 Pelech, Emil 1307060058     MA    02458 1988-03-17    29
##  9 Blount, Dia~ 1403066125     MA    02171 1990-09-21    27
## 10 Hankard, Ea~ 1307059937     MA    02155 1988-08-10    29
## 11 Costa, Latia 1409070567     CA    90007 1988-11-08    29
```

Se pueden incluir operadores lógicos para realizar condiciones más complejas.

```
filter(df, (MaritalDesc == "Single" | Age < 30) &
             Sex == "Male")
```

```
## # A tibble: 65 x 21
##    EmployeeName EmployeeNumber State Zip   DOB          Age
##    <chr>        <chr>          <fct> <fct> <date>     <int>
##  1 Steans, Tyr~ 1302053333     MA    02703 1986-09-01    31
##  2 Murray, Tho~ 1406068403     TX    78230 1988-07-04    29
##  3 Foss, Jason  1192991000     MA    01460 1980-07-05    37
##  4 Roup,Simon   1106026933     MA    02481 1973-04-05    44
##  5 Dougall, Er~ 1101023754     MA    01886 1970-07-09    47
##  6 Clayton, Ri~ 1301052902     MA    02170 1985-09-05    32
##  7 Cisco, Anth~ 1102024173     MA    02135 1989-11-24    28
##  8 Merlos, Car~ 1012023013     MA    02138 1987-06-18    30
##  9 Tredinnick,~ 1104025466     MA    01420 1988-05-05    29
## 10 Favis, Dona~ 1412071562     CT    06033 1964-07-30    53
```

Operador *pipe*

Un operador muy útil que permite encadenar distintas manipulaciones es %>%. Este operador índica el flujo de datos de manera que la salida de una operación se emplea como entrada de la siguente.

```
df %>% filter(MaritalDesc == "Single") %>% filter(Age < 30)
```

```
## # A tibble: 17 x 21
##    EmployeeName EmployeeNumber State Zip   DOB          Age
##    <chr>        <chr>          <fct> <fct> <date>     <int>
##  1 Singh, Nan   1307059817     MA    02330 1988-05-19    29
##  2 Lindsay, Le~ 0602000312     CT    06070 1988-10-05    29
##  3 Cierpiszews~ 1012023295     MA    02044 1988-05-31    29
##  4 Evensen, Ap~ 1107027392     MA    02458 1989-05-06    28
##  5 Gold, Sheni~ 1408069539     MA    02451 1992-06-18    25
##  6 Ivey, Rose   1408069882     MA    01775 1991-01-28    27
##  7 Ndzi, Colom~ 1204033041     MA    02110 1989-05-02    28
```

Seleccionar

El verbo `select()` devuelve una nueva tabla con las columnas seleccionadas.

```
df %>% filter(MaritalDesc == "Single") %>%
       filter(Age < 30) %>%
       select(EmployeeName, State, MaritalDesc, Age)
```

```
## # A tibble: 17 x 4
##    EmployeeName                State MaritalDesc   Age
##    <chr>                       <fct> <fct>         <int>
##  1 Singh, Nan                  MA    Single           29
##  2 Lindsay, Leonara            CT    Single           29
##  3 Cierpiszewski, Caroline     MA    Single           29
##  4 Evensen, April              MA    Single           28
##  5 Gold, Shenice               MA    Single           25
##  6 Ivey, Rose                  MA    Single           27
```

Ordenar

El verbo `arrange()` ordena la tabla según las columnas que se le indiquen.

```
df %>% filter(MaritalDesc == "Single") %>%
       filter(Age < 30) %>%
       select(State, Age, MaritalDesc) %>%
       arrange(Age, State)
```

```
## # A tibble: 17 x 3
##    State   Age MaritalDesc
##    <fct> <int> <fct>
##  1 MA       25 Single
##  2 FL       27 Single
##  3 MA       27 Single
##  4 MA       27 Single
##  5 VT       27 Single
##  6 GA       28 Single
```

Mediante `desc()` es posible cambiar el criterio de ordenación de manera que ordene mayor a menor, en vez de de la ordenación por defecto que es ascendente.

```
df %>% filter(MaritalDesc == "Single") %>%
       filter(Age < 30) %>%
       select(State, Age, MaritalDesc) %>%
       arrange(desc(Age), State)
```

```
## # A tibble: 17 x 3
##    State   Age MaritalDesc
##    <fct> <int> <fct>
##  1 CA       29 Single
##  2 CT       29 Single
##  3 MA       29 Single
##  4 MA       29 Single
##  5 MA       29 Single
##  6 MA       29 Single
##  7 CA       28 Single
```

Modificar

El verbo `mutate()` permite añadir nuevas variables a la tabla.

```
df %>% mutate(Days = Sys.Date() - DateofHire) %>%
       select(Age, Days)
```

```
## # A tibble: 302 x 2
##      Age Days
##    <int> <time>
##  1    32 3620 days
##  2    33 1723 days
##  3    31 1457 days
##  4    32 1317 days
##  5    29 1243 days
##  6    30 2556 days
##  7    33 " 994 days"
##  8    33 2773 days
```

Combinando tablas

Primeramente leemos el nuevo conjunto de datos en una tabla, y lo limpiamos.

Este conjunto tiene que compartir al menos una columna con la tabla queremos combinar. Idealmente tendría que ser una variable que fuera un identificador único.

```
df_pr <- read_csv(here("data", "production_staff.csv"))
colnames(df_pr) <- str_replace_all(colnames(df_pr), "[ /-]", "
colnames(df_pr)
```

```
##  [1] "EmployeeName"      "RaceDesc"          "DateofHire"
##  [4] "TermDate"          "ReasonforTerm"     "EmploymentStatu
##  [7] "Department"        "Position"          "Pay"
## [10] "ManagerName"       "PerformanceScore"  "AbutmentsHourWk
## [13] "AbutmentsHourWk2"  "DailyErrorRate"    "90dayComplaints
```

Comprobamos el tamaño de las dos tablas a combinar.

```
dim(df)
```

```
## [1] 302  21
```

```
dim(df_pr)
```

```
## [1] 209  15
```

La función `inner_join()` crea una nueva tabla manteniendo las filas que compartan identificar en las tablas originales. Aquellas filas que no emparejan se descartan.

```
df_inner <- df %>% inner_join(df_pr, by="EmployeeName")
df_inner
```

```
## # A tibble: 209 x 35
##    EmployeeName EmployeeNumber State Zip   DOB          Age
##    <chr>        <chr>          <fct> <fct> <date>     <int>
##  1 King, Janet  1001495124     MA    01902 1954-09-21    63
##  2 Albert, Mic~ 1501072311     MA    02169 1968-10-10    49
##  3 Bozzi, Char~ 1303054580     MA    01901 1970-03-10    47
##  4 Butler, Web~ 1110029990     MA    02169 1983-08-09    34
##  5 Dunn, Amy    1409070147     MA    01731 1973-11-28    44
##  6 Gray, Eliji~ 1307060077     MA    01752 1981-07-11    36
##  7 Hogland, Jo~ 1001944783     MA    01890 1972-07-01    45
##  8 Immediato, ~ 1403065874     MA    02128 1976-11-15    41
##  9 Liebig, Ket~ 1103024679     MA    02110 1981-10-26    36
```

Otros maneras de combinar dos tablas son *uniones externas* mediante las funciones `left_join()`, `right_join()` y `full_join()`.

```
df_full <- df %>% full_join(df_pr, by = "EmployeeName")
dim(df_full)
```

```
## [1] 302  35
```

```
dim(df_inner)
```

```
## [1] 209  35
```

Los valores invalidos (`NA`) en la columna `DailyErrorRate` difieren en ambos casos, ya que esta columna sólo está completada para los empleados que se encuentran en el conjunto de datos `production_staff.csv`.

Al unir completamente ambas tablas, se rellena con `NA` en esa columna aquellos empleados de la primera tabla que no estuvieran en la segunda.

```
sum(is.na(df_full$DailyErrorRate))
```

```
## [1] 94
```

```
sum(is.na(df_inner$DailyErrorRate))
```

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

Independientemente del tipo de union, se conservan todas las columnas de ambas tablas.

Aquellas columnas que tuvieran el mismo nombre se duplican añadiendo el prefijo `.x` o `.y` al nuevo de nombre para localizar a que tabla corresponden.

Esto es un indicador de mala organización del conjunto de datos, o del mala elección del nombre de las columnas en las tablas originales.

```
cols <- df %>%
        inner_join(df_prod, by="EmployeeName") %>%
        colnames()
cols
```

```
##  [1] "EmployeeName"       "EmployeeNumber"     "State"
##  [4] "Zip"                "DOB"                "Age"
##  [7] "Sex"                "MaritalDesc"        "CitizenDesc
## [10] "HispanicLatino"     "RaceDesc.x"         "DateofHire.
## [13] "DateofTermination"  "ReasonForTerm"      "EmploymentS
## [16] "Department.x"       "Position.x"         "PayRate"
```

```r
df <- df %>%
      inner_join(df_prod, by="EmployeeName") %>%
      select(-contains(".y"))
colnames(df) <- str_replace_all(colnames(df), "\\.x", "")
colnames(df)
```

```
##  [1] "EmployeeName"       "EmployeeNumber"     "State"
##  [4] "Zip"                "DOB"                "Age"
##  [7] "Sex"                "MaritalDesc"        "CitizenDesc"
## [10] "HispanicLatino"     "RaceDesc"           "DateofHire"
## [13] "DateofTermination"  "ReasonForTerm"      "EmploymentSta
## [16] "Department"         "Position"           "PayRate"
## [19] "ManagerName"        "EmployeeSource"     "PerformanceSc
## [22] "TermDate"           "ReasonforTerm"      "Pay"
## [25] "AbutmentsHourWk1"   "AbutmentsHourWk2"   "DailyErrorRat
## [28] "90dayComplaints"
```

Guardamos la nueva tabla con las variables completas para cada empleado.

```
save(df, file = here("data","join_dataset_clean.RData"))
```

Referencias

- RStudio tidyverse Cheatsheet: resumen de cómo utilizar el paquete `dplyr`, incluido dentro de `tidyverse`, y que contiene todas las funciones empleadas para manipular datos de este tema.