Finanzas en R

Notebook 2

Sebastián Egaña Santibáñez Nicolás Leiva Díaz

Enlaces del profesor

https://segana.netlify.app

https://github.com/sebaegana

https://www.linkedin.com/in/sebastian-egana-santibanez/

Agregación y unión

Agregación

Corresponde a cuando generamos métricas de un set de datos en base a alguna variable de agrupamiento. En este sentido, la métrica básica corresponde a conteos, pero también podemos generar promedios, máximos, mínimos, etc.

Leemos la data presente en el siguiente repositorio:

Enlace acá

Utilizamos el set de datos star.

1. Cargamos las librerías:

```
library(tidyverse)
```

```
Warning: package 'tidyverse' was built under R version 4.3.3
```

```
Warning: package 'dplyr' was built under R version 4.3.3
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.0
v ggplot2   3.4.2     v tibble    3.2.1
v lubridate 1.9.2     v tidyr     1.3.0
v purrr     1.0.1
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```
library(httr)
library(readxl)
```

2. Con el siguiente código, podemos leer directamente el archivo:

```
# Define the URL for the raw Excel file
url <- "https://raw.githubusercontent.com/stringfestdata/advancing-into-analytics-book/mai

# Get the data from the URL
response <- httr::GET(url)

# Ensure the response status is 200 (OK)
if (response$status_code == 200) {
    # Read data from the raw binary content
    temp_file <- tempfile(fileext = ".xlsx")
    writeBin(httr::content(response, "raw"), temp_file)
    data <- readxl::read_excel(temp_file)
    # Optionally, delete the temporary file after reading
    unlink(temp_file)
    # Use the data
    print(head(data))
```

```
} else {
    print("Failed to download file: HTTP status code is not 200")
}
```

```
# A tibble: 6 x 8
  tmathssk treadssk classk            totexpk sex   freelunk race  schidkn
     <dbl>    <dbl> <chr>                <dbl> <chr> <chr>    <chr>   <dbl>
1      473      447 small.class              7 girl  no       white      63
2      536      450 small.class             21 girl  no       black      20
3      463      439 regular.with.aide        0 boy   yes      black      19
4      559      448 regular                 16 boy   no       white      69
5      489      447 small.class              5 boy   yes      white      79
6      454      431 regular                  8 boy   yes      white       5
```

Otro enfoque podría ser descargarlo y leerlo desde alguna carpeta.

Generamos una agrupación por la variables ´classk´

```
data %>% group_by(classk) %>% summarise(n())
```

```
# A tibble: 3 x 2
  classk            `n()`
  <chr>             <int>
1 regular            2000
2 regular.with.aide  2015
3 small.class        1733
```

Consideramos el promedio:

```
data %>% group_by(classk) %>% summarise(mean(tmathssk))
```

```
# A tibble: 3 x 2
  classk            `mean(tmathssk)`
  <chr>                        <dbl>
1 regular                       483.
2 regular.with.aide             483.
3 small.class                   491.
```

Uniones

Corresponde a la interacción entre dos tablas por al menos una llave. Traemos el set de datos distrcits:

1. Leemos el archivo districts:

```
# Define the URL for the raw CSV file
url <- "https://raw.githubusercontent.com/stringfestdata/advancing-into-analytics-book/mai

# Get the data from the URL
response <- httr::GET(url)

# Ensure the response status is 200 (OK)
if (response$status_code == 200) {
    # Read data directly from the raw binary content into a CSV format
    temp_file <- tempfile(fileext = ".csv")
    writeBin(httr::content(response, "raw"), temp_file)
    data_01 <- read.csv(temp_file)  # Change to read.csv for CSV files
    # Optionally, delete the temporary file after reading
    unlink(temp_file)
    # Use the data
    print(head(data_01))
} else {
    print("Failed to download file: HTTP status code is not 200")
}
```

```
  schidkn     school_name      county
1       1         Rosalia New Liberty
2       2 Montgomeryville      Topton
3       3            Davy    Wahpeton
4       4        Steelton   Palestine
5       6      Tolchester     Sattley
6       7         Cahokia     Sattley
```

2. Unimos los dos data sets:

```
data %>% left_join(data_01, by = "schidkn")
```

```
# A tibble: 5,748 x 10
   tmathssk treadssk classk     totexpk sex   freelunk race  schidkn school_name
      <dbl>    <dbl> <chr>        <dbl> <chr> <chr>    <chr>   <dbl> <chr>
```

```
 1      473      447 small.cla~       7 girl  no       white      63 Ridgeville
 2      536      450 small.cla~      21 girl  no       black      20 South Heig~
 3      463      439 regular.w~       0 boy   yes      black      19 Bunnlevel
 4      559      448 regular         16 boy   no       white      69 Hokah
 5      489      447 small.cla~       5 boy   yes      white      79 Lake Mathe~
 6      454      431 regular          8 boy   yes      white       5 <NA>
 7      423      395 regular.w~      17 girl  yes      black      16 Calimesa
 8      500      451 regular          3 girl  no       white      56 Lincoln He~
 9      439      478 small.cla~      11 girl  no       black      11 Moose Lake
10      528      455 small.cla~      10 girl  no       white      66 Siglerville
# i 5,738 more rows
# i 1 more variable: county <chr>
```

Generamos un agrupamiento considerando el join aplicado:

```
data %>% left_join(data_01, by = "schidkn") %>% group_by(county) %>% summarise(n())
```

```
# A tibble: 18 x 2
   county         `n()`
   <chr>          <int>
 1 Belmar           393
 2 Corunna          269
 3 Edmore           176
 4 Gallipolis       236
 5 Grand Blanc      293
 6 Imbery           130
 7 Mabie            259
 8 Manteca          432
 9 New Liberty      315
10 Palestine        466
11 Reddell          262
12 Sattley          384
13 Selmont          664
14 Sugar Mountain   383
15 Summit Hill      332
16 Topton           407
17 Wahpeton         288
18 <NA>              59
```

Otra forma de hacer esto:

```r
data_join = data %>% left_join(data_01, by = "schidkn")

data_join %>% group_by(county) %>% summarise(n())
```

```
# A tibble: 18 x 2
   county         `n()`
   <chr>          <int>
 1 Belmar           393
 2 Corunna          269
 3 Edmore           176
 4 Gallipolis       236
 5 Grand Blanc      293
 6 Imbery           130
 7 Mabie            259
 8 Manteca          432
 9 New Liberty      315
10 Palestine        466
11 Reddell          262
12 Sattley          384
13 Selmont          664
14 Sugar Mountain   383
15 Summit Hill      332
16 Topton           407
17 Wahpeton         288
18 <NA>              59
```

Tipos de Join

Uniones: llave primaria y foránea

En SQL, se habla principalmente de llaves en el sentido de columnas que nos permitan unir dos bases. Intentamos realizar un metodo que mantenga la integridad de la tabla a la cual añadiré información.

Una llave primaria corresponde a la columna de la base a la cual yo deseo unir otra columna, usando como punto de interacción una llave foránea.

Consideremos las dos bases, en donde A corresponde a la llave primaria y B corresponde a la base foránea de cada una de las bases:

A	B
1	3
2	4
3	5
4	6

¿Qué valores se repiten dentro de cada llave?

Una complejidad de las uniones en SQL, corresponde a los requisitos que se deben cumplir para poder realizar uniones entre bases. Por ejemplo, una particularidad es que la llave primaria no debe tener valores repetidos y esto se repite en programas como RStudio. Por otra parte, Excel al utilizar funciones de manera individual (celda por celda), no posee dicho problema.

Veamos ahora distintos tipos de uniones que pueden generarse.

- Inner Join: Refiere a la unión en base a intersecciones.

A	B
3	3
4	4

- Outer Join: Refiere a la unión de todas las columnas en A y todas las columnas en B. Las celdas sin correlativo en la otra base, generarán un valor nulo.

A	B
1	NULL
2	NULL
3	3
4	4
NULL	5
NULL	6

Esto tipo de unión, puede tener dos variaciones:

- Outer Join > Left Outer Join: Predominan las columnas en A, y las comunes en B.

A	B
1	NULL
2	NULL

A	B
3	3
4	4

En Rstudio solo se conoce como Left Join.

- Outer Join > Right Outer Join: Predominan las columnas en B, y las comunes en A.

A	B
3	3
4	4
NULL	5
NULL	6

En Rstudio solo se conoce como Right Join.

Aplicación práctica de uniones

Set de datos

Considere que se tiene un set de datos data_uniones.xlsx, que contiene 4 tablas:

- Flights: Corresponde a distintos identificadores de vuelos por ruta.
- Routes: Rutas de vuelos.
- Aircraft: Tipos de aviones.
- Airports: Set de datos descriptor de aeropuertos en USA.

Import - Tidy - Transform

- ¿Cuáles son las variables de cada set de datos?

Por ejemplo, ¿qué es FuelCostperSeatMile (Cents)? Pero una pregunta más general es, ¿por qué es relevante aprender estos conceptos?

- ¿Se observa algún error en los datos?

Fijemonos en el formato de la fecha.

- ¿Posee la estructura correcta el set de datos?

No se ven errores; en las filas están las observaciones y en las columnas las variables. No existen formatos, ni celdas combinadas.

- ¿Existe identificadores únicos en los set de datos?

Existen múltiples identificadores únicos.

Se debe intentar llegar al análisis relevante del set de datos. En el caso de este set datos, claramente se puede llegar a un set de datos más completo.

- ¿Cúal es el set de datos que debe mantenerse? Veamos alguna manera de llegar a dicha conclusión.

Por último, se puede pensar en la inclusión de datos a la base de datos, que podrían ser de algún interés. En este caso intentaremos agregar los siguientes datos:

Acá

Hasta este punto, aún no hemos terminado con dicho procedimiento de **batallar con los datos**.

Antes de aplicar la unión de datos, sería bueno poder diagramar la relación que existe entre cada base datos. Acá debemos abordar un contenido relacionado.

Modelo de datos

Corresponde a una serie de conceptos que pueden utilizarse para describir un conjunto de datos y las operaciones relacionadas para su manipulación.

En la actualidad el modelo de bases de datos más utilizado corresponde al modelo relacional, que se define como el intento de obtener datos de distintas fuentes a través de relaciones o consultas que se basan en llaves (o columnas) que permiten la relación entre las distintas tablas. Esto quiere decir, que no necesariamente se tiene una **base gigante**, sino que se tiene la opción de acceder a bases pequeñas que pueden ir complementando los análisis.

Modelo Relacional

Las entidades y relaciones se representan en formas de tablas.

- Las tablas son las relaciones.
- Las filas (tuplas) contienen datos sobre cada entidad.
- Las columnas corresponden a atributos de las entidades.

Se pretende determinar operaciones a realizar: Unión, intersección, diferencia, producto cartesiano, selección, proyección, reunión, etc.

Por otra parte, existen restricciones de integridad de entidad como también de integridad referencial (relacionado con el tema de llaves que vimos anteriormente).

Por ejemplo:

Entidades	Proveedor	Pieza
Atributos	Código	Código
	Nombre	Nombre
	Ciudad	Dimensiones Peso

En donde:

Relación	Suministra
Entidades Participantes	Proveedor - Pieza
Cardinalidad	Muchos a muchos
Atributos	Cantidad

De manera más especifica, el modelo entidad/relación corresponde a la técnica basada en la identificación de las entidades y de las relaciones que se dan entre ellas, según la realidad que se intenta modelar.

Se debe recordar acá, el concepto de llave del que hablamos la clase pasada. Una llave (o también llamada clave), corresponde a un conjunto de atributos que permite identificar unívocamente a una entidad dentro de un conjunto de entidades.

Por ejemplo para una Facultad el modelo podría ser:

Entidades	Asignatura	Alumno	Profesor	Departamento	Aula	Grupo
Atributos	ID	RUT	RUT	ID	ID	ID
	Nombre	Nombre	Nombre	Nombre	Capacidad	Tipo
	Créditos	Dirección	Categoría			
	Curso	Email	Email			

En donde existen las siguientes relaciones:

Relación	Entidades participantes	Cardinalidad	Atributos
se matricula en	Alumno - Grupo	N:M	Calificación
enseña	Profesor - Grupo	N:M	
impartida en	Asignatura - Grupo	1:N	
asignada a	Aula - Grupo	N:M	Día, hora
pertenece a	Profesor - Departamento	N:1	
dirige	Profesor - Departamento	1:1	

Esto también puede ser representado de manera grafica. Veamos el siguiente ejemplo:

```
library(datamodelr)

file_path <- system.file("samples/example.yml", package = "datamodelr")
dm <- dm_read_yaml(file_path)

graph <- dm_create_graph(dm, rankdir = "LR")

#dm_render_graph(graph)
dm_export_graph(graph, 'imagenes/diagrama_01.png')
```

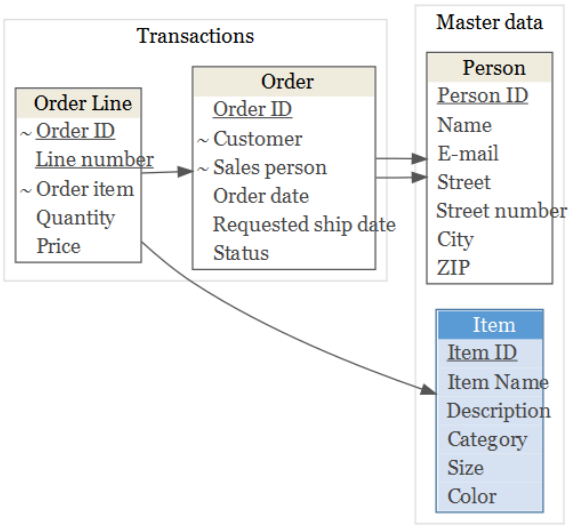

Figure 1: Diagrama 01

Veamos esto en relación a los datos que estuvimos revisando; primero debemos ingresar las tablas sobre las que estuvimos trabajando.

```
library(readxl)

flights_01 <- read_excel("data/data_uniones.xlsx",
              sheet = "Flights")

routes_01 <- read_excel("data/data_uniones.xlsx",
                        sheet = "Routes")

airports_01 <- read_excel("data/data_uniones.xlsx",
                        sheet = "Airports")

aircraft_01 <- read_excel("data/data_uniones.xlsx",
                        sheet = "Aircraft")


dm_f <- dm_from_data_frames(flights_01, routes_01, aircraft_01, airports_01)
graph <- dm_create_graph(dm_f, rankdir = "BT", col_attr = c("column", "type"))
#dm_render_graph(graph)

dm_export_graph(graph, 'imagenes/diagrama_02.png')
```

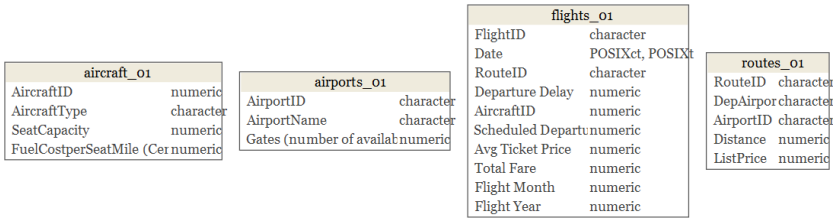

Figure 2: Diagrama 02

Después debemos generar las relaciones entre cada una de las tablas:

```
dm_f <- dm_add_references(
  dm_f,

  aircraft_01$AircraftID == flights_01$AircraftID,
  airports_01$AirportID == routes_01$AirportID,
  flights_01$RouteID == routes_01$RouteID
)
graph <- dm_create_graph(dm_f, rankdir = "BT", col_attr = c("column", "type"))
#dm_render_graph(graph)
dm_export_graph(graph, 'imagenes/diagrama_03.png')
```

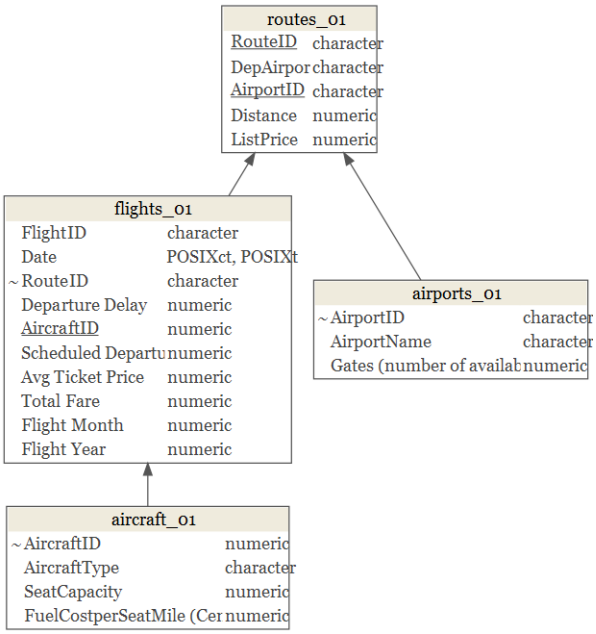

Figure 3: Diagrama 03

Podemos intentar mejorar un poco el formato de la visualización:

```
graph <- dm_create_graph(
  dm_f,
  graph_attrs = "rankdir = RL, bgcolor = '#F4F0EF' ",
  edge_attrs = "dir = both, arrowtail = crow, arrowhead = odiamond",
  node_attrs = "fontname = 'Arial'")

#dm_render_graph(graph)
dm_export_graph(graph, 'imagenes/diagrama_04.png')
```

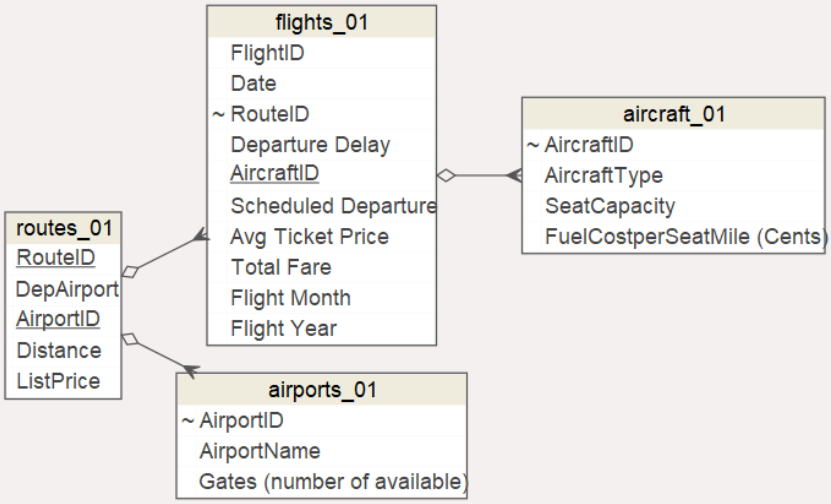

Figure 4: Diagrama 04