Transformación de datos II

Alejandro Bellati

11/04/2021

Recordando

- Los datos suelen no estar en la forma que necesitamos. Siempre necesitamos crear nuevas variables, resúmenes, renombrar, reordenar, etc.
- El paquete **dplyr** de tidyverse útil para transformar datos.
- filter(), arrange(), select().
- mutate(), summarise()
- pipes y el uso de group_by()

```
library(tidyverse)
library(datos)
library(dplyr)
```

A utilizar

Usamos el conjunto de datos de los vuelos que partieron de Nueva York en el 2013

```
vuelos[1:6,1:5]
```

```
## # A tibble: 6 x 5
##    anio   mes   dia horario_salida salida_programada
##   <int> <int> <int>          <int>             <int>
## 1  2013     1     1            517               515
## 2  2013     1     1            533               529
## 3  2013     1     1            542               540
## 4  2013     1     1            544               545
## 5  2013     1     1            554               600
## 6  2013     1     1            554               558
```

Mutate

Para crear columnas nuevas en función de las existentes podemos usar la función mutate(). Agrega columnas al final del dataframe.

```
vuelos_sml <- select(vuelos,
                     anio:dia,
                     starts_with("atraso"),
                     distancia,
                     tiempo_vuelo)
```

Si queremos calcular la velocidad, el tiempo de vuelo, etc

```
mutate(vuelos_sml,
  ganancia = atraso_salida - atraso_llegada,
  velocidad = distancia/tiempo_vuelo *60,
  horas = tiempo_vuelo/60,
  ganancia_por_hora = ganancia/horas
)
```

Transmute y utilidades

Si quiero solo quedarme con las nuevas variables

```
transmute(vuelos_sml,
  velocidad = distancia/tiempo_vuelo *60,
  horas = tiempo_vuelo/60,
)
```

Hay varias funciones útiles para mutate. Deben ser vectorizadas.

```
transmute(vuelos,
  horario_salida,
  hora = horario_salida %/% 100,
  minuto = horario_salida %% 100
)
```

Resumenes

La función summarise se encarga de colapsar un data frame en una sola fila, pueden ser varias columnas.

```
summarise(vuelos,
          atraso = mean(atraso_salida, na.rm = TRUE))
```

```
## # A tibble: 1 x 1
##   atraso
##    <dbl>
## 1   12.6
```

No es muy útil sin usar group_by. Cambia la unidad de análisis del conjunto completo a grupos. Al usar las funciones de dplyr con un dataframe agrupados, estas se aplican automaticamente en grupos.

Summarise con group_by

Quiero ver el promedio de los atrasos en cada día del año

```
por_dia <- group_by(vuelos, anio, mes, dia)
res <- summarise(por_dia,
          atraso = mean(atraso_salida, na.rm = TRUE)
          )
res[1:3,1:4]
```

```
## # A tibble: 3 x 4
## # Groups:   anio, mes [1]
##    anio   mes   dia atraso
##   <int> <int> <int>  <dbl>
## 1  2013     1     1   11.5
## 2  2013     1     2   13.9
## 3  2013     1     3   11.0
```

Pipe - Introducción

¿Qué relación hay entre la distancia y el atraso promedio, para cada destino?

```
por_destino <- group_by(vuelos,destino)
atraso <- summarise(por_destino,
  conteo = n(),
  distancia = mean(distancia,na.rm=TRUE),
  atraso = mean(atraso_llegada, na.rm=TRUE)
)
atraso <- filter(atraso, conteo>20, destino != "HNL")

ggplot(data = atraso, mapping = aes(x = distancia, y = atraso)) +
  geom_point(aes(size = conteo), alpha = 1/3) +
  geom_smooth(se = FALSE)
```

Tuvimos que dar nombre a cada dataframe intermedio.

Pipe

El operador pipeline %>% es útil para concatenar múltiples dplyr operaciones.

```
atrasos<- vuelos %>%
  group_by(destino) %>%
  summarise(
    conteo = n(),
    distancia = mean(distancia,na.rm=TRUE),
    atraso = mean(distancia, na.rm=TRUE),
  ) %>%
  filter(conteo>20, destino != "HNL")
```

Solo creo el dataframe final!

Pipe

- Se puede ver como una función compuesta, aplicación de transformaciones sucesivas: x %>% f(y) se convierte en f(x,y).
- Una serie de declaraciones imperativas.
- Leer el %>% como *luego*.

Ejemplo, atraso promedio en cada día del año.

```
vuelos %>%
  filter(!is.na(atraso_salida),!is.na(atraso_llegada)) %>%
  group_by(anio,mes,dia) %>%
  summarise(promedio = mean(atraso_salida))
```

Pipe

Si quiero guardar resultados intermedio debo "partir" el pipe:

```
no_cancelados <-vuelos %>%
  filter(!is.na(atraso_salida),!is.na(atraso_llegada))

no_cancelados %>%
  group_by(anio,mes,dia) %>%
  summarise(mean=(atraso_salida))
```

- Atajo: Ctrl + Shift + M
- Trabajar con Pipe se exige para ser paquete del tidyverse, la única excepción es ggplot2.

Conteos

Siempre que hacemos summarise esta bueno incluir alguna medida de conteo para ver que no estamos sacando conclusiones con pocos datos.

```
atrasos <- no_cancelados %>%
  group_by(codigo_cola) %>%
  summarise(
    atraso = mean(atraso_llegada),
    n = n()
  )

ggplot(atrasos, mapping = aes(x = n, y = atraso)) +
  geom_point(alpha = 1/10)
```

Obs: parto el pipe para que grafique.

Funciones de resumen útiles

- Medidas de centralidad: mean(x), median(x).
- Medidas de dispersión: sd(x), IQR(x), mad(x).
- Medidas de posición: first(x), nth(x,2), last(x).
- Conteos: n(x), sum(!is.na(x)), n_distinct(x), count()
- Conteos y proporciones lógicas: sum(x>10), mean(x==0)

Ejemplos

¿Qué destinos tienen la mayoría de las aerolineas?

```
no_cancelados %>%
  group_by(destino) %>%
  summarise(
    cantidad = n_distinct(aerolinea)) %>%
    arrange(desc(cantidad))
```

¿Qué proporción de vuelos se retrasa más de 1 hora, en cada día?

```
no_cancelados %>%
  group_by(anio,mes,dia) %>%
  summarise(
    proporcion = mean(atraso_llegada>60)
  )
```

¿Cuáles son los aeropuertos a donde llegaron la menor cantidad de aviones durante 2013?

```
no_cancelados %>%
  group_by(destino) %>%
  summarise(cantidad = n_distinct(codigo_cola)) %>%
  arrange(cantidad)
```

Ejemplos

¿Para cada aeropuerto de NY, cuál es el atraso promedio, el atraso positivo promedio y la cantidad de horas de vuelo?

```
no_cancelados %>%
  group_by(origen) %>%
  summarise(
    atrasos = mean(atraso_llegada),
    atrasos_pos = mean(atraso_llegada[atraso_llegada>0]),
    horas_vuelo = sum(tiempo_vuelo)/60
    )
```

```
## # A tibble: 3 x 4
##   origen atrasos atrasos_pos horas_vuelo
##   <chr>    <dbl>       <dbl>       <dbl>
## 1 EWR       9.11        41.8     299260.
## 2 JFK       5.55        40.0     324236.
## 3 LGA       5.78        38.9     198615.
```

Ejemplos

Contar es tan útil que dplyr proporciona un ayudante si lo único que buscas es contar:

```
a<-no_cancelados %>%
  count(destino)
a[1:4,1:2]
```

```
## # A tibble: 4 x 2
##   destino     n
##   <chr>   <int>
## 1 ABQ       254
## 2 ACK       264
## 3 ALB       418
## 4 ANC         8
```

```
no_cancelados %>%
  count(codigo_cola, wt=distancia)
```

```
## # A tibble: 4 x 2
##   codigo_cola      n
##   <chr>        <dbl>
## 1 D942DN        3418
## 2 N0EGMQ      239143
## 3 N10156      109664
## 4 N102UW       25722
```

Agrupación por múltiples variables y ungroup

- Si agrupamos por varias variables, cada vez que aplicamos summarise, el resultado se desprende de una agrupación.

```
a<-no_cancelados %>%
  group_by(mes,dia) %>%
  summarise(cantidad_vuelos = n())
a[1:5,1:3]
```

```
## # A tibble: 5 x 3
## # Groups:   mes [1]
##     mes   dia cantidad_vuelos
##   <int> <int>           <int>
## 1     1     1             831
## 2     1     2             928
## 3     1     3             900
## 4     1     4             908
## 5     1     5             717
```

- Si queremos deshacernos de una agrupación podemos usar ungroup().

Transformaciones y filtros agrupados

También se puede agrupar con filter() y mutate().

```
vuelos_sml %>%
  group_by(anio,mes,dia) %>%
  filter(rank(desc(atraso_llegada))<10)
```

```
destinos_populares <- vuelos %>%
  group_by(destino) %>%
  filter(n()>500)
```

Un filtro agrupado es una transformación agrupada seguida de un filtro desagrupado. EN general es preferible evitarlo pues de es difícil comprobar que has hecho la manipulación bien.

Transformaciones y filtros agrupados

También se puede agrupar con filter() y mutate().

```
a<-destinos_populares %>%
  filter(atraso_llegada > 0) %>%
  mutate(prop_atraso = atraso_llegada / sum(atraso_llegada)) %>%
  select(anio:dia, destino, atraso_llegada, prop_atraso)
a[1:3,1:6]
```

```
## # A tibble: 3 x 6
## # Groups:   destino [2]
##    anio   mes   dia destino atraso_llegada prop_atraso
##   <int> <int> <int> <chr>            <dbl>       <dbl>
## 1  2013     1     1 IAH                 11    0.000111
## 2  2013     1     1 IAH                 20    0.000201
## 3  2013     1     1 MIA                 33    0.000235
```

```
a<-destinos_populares %>%
  ungroup() %>%
  filter(atraso_llegada > 0) %>%
  mutate(prop_atraso = atraso_llegada / sum(atraso_llegada)) %>%
  select(anio:dia, destino, atraso_llegada, prop_atraso)
a[1:3,1:6]
```

```
## # A tibble: 3 x 6
##    anio   mes   dia destino atraso_llegada prop_atraso
##   <int> <int> <int> <chr>            <dbl>       <dbl>
## 1  2013     1     1 IAH                 11  0.00000210
## 2  2013     1     1 IAH                 20  0.00000381
## 3  2013     1     1 MIA                 33  0.00000629
```