

Manipulación de datos

Jorge Meneses y Paulo Peña

Manipulación de datos

La manipulación y arreglo de datos es un paso fundamental en proceso de análisis de datos e información. En muchas ocasiones, las bases de datos que importamos tienen mucha más información de la que vamos a necesitar, por lo que existen diferentes comandos que nos permiten filtrar, seleccionar, o recortar las tablas para trabajar solo con los datos que necesitamos.

Por ejemplo: Descargamos una base de datos de incidencias de enfermedades en todo el país, pero queremos hacer un análisis sobre la situación en Cusco. Con R podemos filtrar los datos correspondientes solo a la región.

También tenemos herramientas para crear nuevas variables, con datos que resulten de cálculos sobre otras variables.

Por ejemplo: Tenemos una base de datos con los ingresos de una empresa en cada mes. Podemos crear una nueva columna total, con los resultados para ese año.

Paquete Dplyr



El paquete **dplyr** es un paquete que incluye múltiples herramientas para la manipulación de datos. Podemos encontrar más información sobre el paquete en su página web (<https://dplyr.tidyverse.org/>)

```
# Para instalar dplyr  
install.packages("dplyr")
```

Solo es necesario ejecutar este comando una vez en la consola. Luego tendremos instalado ese paquete en nuestra computadora

Cargando dplyr

Una vez que tenemos instalado dplyr en nuestra computadora es necesario que carguemos el paquete en nuestro entorno de trabajo (Environment). Para hacerlo, hay que correr (run) la siguiente línea en la parte superior de nuestro guión:

```
library(dplyr) # Carga dplyr en nuestro entorno
```

Una vez cargado dplyr, podemos usar sus comandos como parte de nuestro trabajo. Dada la importancia de dplyr, es casi seguro que lo usemos en todos nuestros archivos.

Trabajando con *pipes* (tuberías)

Una tubería o *pipe* en R, nos permite pasar información de un comando a otro. Esto nos va a ser muy útil al momento de manipular datos y hacer ajustes sobre nuestras tablas.

El trabajo con *pipes* fue introducido por dplyr usando el comando `%>%`. Este comando lo podemos usar de esta forma:

```
# Sin pipe
tabla <- paso1(tabla)
tabla <- paso2(tabla)
tabla <- paso3(tabla)
tabla <- paso4(tabla)

# Con pipe
tabla <- tabla %>%
  paso1() %>%
  paso2() %>%
  paso3() %>%
  paso4() %>%
```

Usando pipes, tenemos un código más fácil de leer.

Pipe nativo

A partir de la versión 4 de R, se introdujo un nuevo *pipe*, que no requiere de dplyr. Ese comando se escribe `|>`. En términos generales, se usa igual que `%>%`.

```
# Sin pipe
tabla <- paso1(tabla)
tabla <- paso2(tabla)
tabla <- paso3(tabla)
tabla <- paso4(tabla)
```

```
# Con pipe
tabla <- tabla |>
  paso1() |>
  paso2() |>
  paso3() |>
  paso4() |>
```

Comandos útiles

En la sesión de hoy veremos los siguientes comandos usados en el paquete **dplyr**:

- `count()`. Permite contar los datos de una variable.
- `filter()`. Filtrado de datos.
- `select()`. Selecciona columnas específicas.
- `rename()`. Renombra columnas específicas.
- `mutate()`. Crea nuevas variables en base a otras anteriores.

Un tema importante

Siempre es importante trabajar con datos “limpios”. Con esto nos referimos que nuestras tablas deben estar ordenadas (tener nombres claros, cada columna debe tener solo una variable, y cada fila solo una observación o caso). El paquete **janitor** incluye muchas herramientas que nos ayudan con esta tarea. No nos extenderemos en el uso de este paquete, pero una herramienta muy útil es el comando `clean_names()`, con el cual nos aseguraremos que las tablas que importemos tengan nombres fáciles de usar.

```
# Cargamos base de datos de Información de usuarios de Juntos en I Bimestre del 2020  
# Fuente https://datosabiertos.midis.gob.pe/dataset/juntos-informaci%C3%B3n-de-usuar
```

```
db_juntos <- read.csv(  
  "sesiones/06-manipulación_datos/03-Dataset-JUNTOS-informacion-usuarios-I-bimestre-  
  sep = ";" # indicamos que el separador del csv es ;  
  ) |>  
  janitor::clean_names() # Limpia los nombres de las columnas
```


Conociendo nuestra base de datos

Recordemos comandos para conocer nuestra base de datos

```
# Revisamos la estructura general de la tabla
str(db_juntos)
```

```
## 'data.frame':    1325 obs. of  9 variables:
## $ x                : int  1 2 3 4 5 6 7 8 9 10 ...
## $ ubigeo            : int  10102 10103 10104 10105 10106 10107 10108 10109 10110 ...
## $ departamento     : chr  "AMAZONAS" "AMAZONAS" "AMAZONAS" "AMAZONAS" ...
## $ provincia        : chr  "CHACHAPOYAS" "CHACHAPOYAS" "CHACHAPOYAS" "CHACHAPOYAS" ...
## $ distrito         : chr  "ASUNCION" "BALSAS" "CHETO" "CHILIQUN" ...
## $ hogares_afiliados : int  25 106 50 72 220 21 25 657 324 74 ...
## $ hogares_abonados  : int  20 103 51 70 207 20 23 649 315 56 ...
## $ miembros_objetivos: int  41 212 81 111 440 38 45 1413 605 125 ...
## $ transferencia     : num  4700 24900 10400 16300 42900 ...
```

Tenemos 9 variables (columnas) y 1325 observaciones (filas). También nos indica los nombres de las columnas, su tipo de variable y los primeros casos. ¿Que podemos conocer con esta información?

Resumen general

```
# Obtenemos un resumen general de la tabla
summary(db_juntos)
```

```
##           x           ubigeo      departamento      provincia
## Min.      :    1   Min.      : 10102   Length:1325      Length:1325
## 1st Qu.:  332   1st Qu.: 50304   Class :character   Class :character
## Median :  663   Median : 90203   Mode  :character   Mode  :character
## Mean     :  663   Mean      :100729
## 3rd Qu.:  994   3rd Qu.:151011
## Max.     :1325   Max.      :250401
## distrito      hogares_afiliados hogares_abonados miembros_objetivos
## Length:1325    Min.      :    1.0   Min.      :    1.0   Min.      :    1
## Class :character 1st Qu.:  116.0   1st Qu.:  115.0   1st Qu.:   225
## Mode  :character Median :  282.0   Median :  282.0   Median :   570
##                Mean     :  546.9   Mean     :  538.4   Mean     : 1129
##                3rd Qu.:  676.0   3rd Qu.:  671.0   3rd Qu.: 1359
##                Max.     :8870.0   Max.     :8519.0   Max.     :18519
## transferencia
## Min.      :    100
## 1st Qu.:  23400
## Median :  58300
## Mean     : 111833
## 3rd Qu.: 138400
## Max.     :1948100
```

```
# Unique nos da una lista con los valores únicos en cada variable  
unique(db_juntos$departamento)
```

```
## [1] "AMAZONAS"      "ANCASH"         "APURIMAC"       "AREQUIPA"  
## [5] "AYACUCHO"      "CAJAMARCA"      "CUSCO"          "HUANCAVELICA"  
## [9] "HUANUCO"       "JUNIN"          "LA LIBERTAD"    "LAMBAYEQUE"  
## [13] "LIMA"          "LORETO"         "MADRE DE DIOS"  "PASCO"  
## [17] "PIURA"        "PUNO"           "SAN MARTIN"     "TACNA"  
## [21] "UCAYALI"
```

```
# Podemos guardar el resultado en una variable  
departamentos <- unique(db_juntos$departamento)
```

Otros comandos útiles

- `sum()` Nos da la suma total de los valores de un vector.
- `mean()`. Da el promedio de los valores de un vector.
- `min()` y `max()`. Nos indican los valores minimos y máximos en un vector.

Todas estas respuestas se pueden guardar en sus propias variables.

```
sum(db_juntos$transferencia)
```

```
## [1] 148178291
```

```
mean(db_juntos$hogares_afiliados)
```

```
## [1] 546.877
```

```
min(db_juntos$transferencia)
```

```
## [1] 100
```

```
max(db_juntos$hogares_abonados)
```

Contando datos

El comando `count()` nos permite contar cuantas veces se repite una variable. En este caso, nos está dando la cantidad de distritos en cada departamento.

```
db_juntos |> count(departamento)
```

```
##      departamento      n
## 1      AMAZONAS      77
## 2      ANCASH      124
## 3      APURIMAC      83
## 4      AREQUIPA      25
## 5      AYACUCHO     115
## 6      CAJAMARCA     125
## 7      CUSCO        93
## 8      HUANCABELICA   99
## 9      HUANUCO       79
## 10     JUNIN         86
## 11     LA LIBERTAD    67
## 12     LAMBAYEQUE      5
## 13     LIMA          25
## 14     LORETO        52
## 15     MADRE DE DIOS  10
## 16     PASCO         24
## 17     PIURA        49
## 18     PUNO         108
## 19     SAN MARTIN     55
## 20     TACNA         10
```

Filtrando datos

La base de Juntos tiene datos para todo el país, pero queremos hacer un estudio sobre Cusco. Necesitamos filtrar los datos según el departamento

Para eso usamos el comando `filter()`. Dentro del paréntesis escribimos el criterio con el que queremos seleccionar los casos.

```
# Creamos una nueva tabla solo con los datos de Cusco  
db_cusco <- db_juntos |>  
  filter(departamento == "CUSCO")
```

Veamos nuestra nueva tabla

```
str(db_cusco)
```

```
## 'data.frame':    93 obs. of  9 variables:
## $ x              : int  550 551 552 553 554 555 556 557 558 559 ...
## $ ubigeo          : int  80102 80201 80202 80203 80204 80205 80206 80207 80301
## $ departamento    : chr  "CUSCO" "CUSCO" "CUSCO" "CUSCO" ...
## $ provincia        : chr  "CUSCO" "ACOMAYO" "ACOMAYO" "ACOMAYO" ...
## $ distrito         : chr  "CCORCA" "ACOMAYO" "ACOPIA" "ACOS" ...
## $ hogares_afiliados : int  246 447 253 245 59 870 245 360 626 361 ...
## $ hogares_abonados  : int  236 442 253 239 60 874 243 362 616 368 ...
## $ miembros_objetivos: int  481 1086 506 572 117 1953 500 803 1218 696 ...
## $ transferencia     : num  48000 87700 50200 47400 12000 ...
```

Tenemos las mismas variables, pero solo 93 casos.

Seleccionar columnas

Para el trabajo que queremos hacer no necesitamos todas las columnas. El comando `select()` nos permite seleccionar solo algunas de ellas.

Como no nos interesa la variable `x` ni la variable `ubigeo` las eliminamos. Tampoco necesitamos ya la variable `departamento`.

```
nueva_tabla <- db_cusco |>
  select( # dentro de select listamos SOLO las columnas que queremos en la nueva tabla
    provincia,
    distrito,
    hogares_afiliados,
    hogares_abonados,
    miembros_objetivos,
    transferencia
  )
```

```
# Revisamos la nueva tabla
str(nueva_tabla)
```

```
## 'data.frame':   93 obs. of  6 variables:
## $ provincia      : chr  "CUSCO" "ACOMAYO" "ACOMAYO" "ACOMAYO" ...
## $ distrito       : chr  "CCORCA" "ACOMAYO" "ACOPIA" "ACOS" ...
## $ hogares_afiliados : int  246 447 253 245 59 870 245 360 626 361 ...
## $ hogares_abonados : int  236 442 253 239 60 874 243 362 616 368 ...
## $ miembros_objetivos: int  481 1086 506 572 117 1953 500 803 1218 696 ...
```


Eliminando columnas

Si la lista de columnas que queremos es muy larga, podemos usar select para *eliminar solo las columnas que no queremos*. Para eso, ponemos el signo – al frente del nombre de la columna

```
nueva_tabla2 <- db_cusco |>
  select(-x, -ubigeo, -departamento)
```

```
str(nueva_tabla2) # Verificamos que es igual que anterior
```

```
## 'data.frame':    93 obs. of  6 variables:
## $ provincia      : chr  "CUSCO" "ACOMAYO" "ACOMAYO" "ACOMAYO" ...
## $ distrito       : chr  "CCORCA" "ACOMAYO" "ACOPIA" "ACOS" ...
## $ hogares_afiliados : int  246 447 253 245 59 870 245 360 626 361 ...
## $ hogares_abonados  : int  236 442 253 239 60 874 243 362 616 368 ...
## $ miembros_objetivos: int  481 1086 506 572 117 1953 500 803 1218 696 ...
## $ transferencia    : num  48000 87700 50200 47400 12000 ...
```

```
db_cusco <- nueva_tabla2 # Guardamos la nueva tabla en lugar de la tabla original
```

```
rm(nueva_tabla, nueva_tabla2) # eliminamos tablas repetidas y que ya no necesitamos
```

Renombrando columnas

El comando `rename()` nos permite cambiar los nombres de algunas columnas. El formato del comando es:

```
nombre_nuevo = nombre_antiguo
```

```
db_cusco <- db_cusco |>  
  rename(  
    afiliados = hogares_afiliados,  
    abonados = hogares_abonados,  
    ojetivos = miembros_objetivos  
  )
```

Creando nuevas variables

El comando `mutate()` nos permite crear nuevas variables en base a información de otras variables.

En nuestro caso, la columna `transferencia` nos indica el valor total en soles de la transferencia a cada distrito. Creemos una nueva variable `transferencia_promedio` que nos indique el promedio de transferencia por cada hogar para cada distrito.

```
db_cusco <- db_cusco |>
  mutate(
    transferencia_promedio = transferencia / abonados
  )
```

```
str(db_cusco)
```

```
## 'data.frame':   93 obs. of  7 variables:
## $ provincia      : chr  "CUSCO" "ACOMAYO" "ACOMAYO" "ACOMAYO" ...
## $ distrito       : chr  "CCORCA" "ACOMAYO" "ACOPIA" "ACOS" ...
## $ afiliados       : int   246  447  253  245  59  870  245  360  626  361 ...
## $ abonados        : int   236  442  253  239  60  874  243  362  616  368 ...
## $ ojetivos        : int   481 1086  506  572 117 1953  500  803 1218  696 ...
## $ transferencia   : num  48000 87700 50200 47400 12000 ...
## $ transferencia_promedio: num   203  198  198  198  200 ...
```

Verificamos la nueva tabla

```
summary(db_cusco)
```

```
##      provincia      distrito      afiliados      abonados
## Length:93      Length:93      Min.   : 22.0      Min.   : 21.0
## Class :character Class :character 1st Qu.: 230.0      1st Qu.: 227.0
## Mode  :character Mode  :character Median : 372.0      Median : 375.0
##                                     Mean  : 491.9      Mean   : 489.1
##                                     3rd Qu.: 667.0      3rd Qu.: 679.0
##                                     Max.   :1974.0      Max.   :1923.0
##      ojetivos      transferencia      transferencia_promedio
## Min.   : 51      Min.   : 4100      Min.   :195.2
## 1st Qu.: 430      1st Qu.: 45300      1st Qu.:198.0
## Median : 790      Median : 76200      Median :198.9
## Mean   :1027      Mean   : 98072      Mean   :200.5
## 3rd Qu.:1283      3rd Qu.:137200      3rd Qu.:200.3
## Max.   :4309      Max.   :384400      Max.   :243.1
```

Ahora sabemos que la transferencia promedio es de 200 soles por hogar, el valor mínimo 195.2 y el valor máximo 234.1 soles por hogar.

Resumiendo

Gracias a pipe podemos hacer todos estos pasos juntos

```
# Filtramos base de cusco y ajustamos los datos que queremos
db_cusco <- db_juntos |>
  filter(departamento == "CUSCO") |>      # filtramos datos de cusco
  select(-x, -ubigeo, -departamento) |>   # eliminamos columnas que no queremos
  rename(                                  # Cambiamos nombre de columnas
    afiliados = hogares_afiliados,
    abonados = hogares_abonados,
    ojetivos = miembros_objetivos
  ) |>
  mutate(                                  # Creamos nueva variable
    transferencia_promedio = transferencia / abonados
  )

# Verificamos tabla final
str(db_cusco)
```

```
## 'data.frame':   93 obs. of  7 variables:
## $ provincia      : chr  "CUSCO" "ACOMAYO" "ACOMAYO" "ACOMAYO" ...
## $ distrito       : chr  "CCORCA" "ACOMAYO" "ACOPIA" "ACOS" ...
## $ afiliados       : int   246  447  253  245  59  870  245  360  626  361 ...
## $ abonados        : int   236  442  253  239  60  874  243  362  616  368 ...
## $ ojetivos        : int   481 1086  506  572 117 1953  500  803 1218  696 ...
## $ transferencia   : num  48000 87700 50200 47400 12000 ...
## $ transferencia promedio: num  203 198 198 198 200 ...
```

Recursos útiles

- Ayuda memoria: “Transformación de datos con dplyr”
(https://raw.githubusercontent.com/rstudio/cheatsheets/main/translations/spanish/data-transformation_es.pdf)

Ejercicio para casa

Usando la misma base de datos creamos una nueva tabla:

1. Filtrar los datos para un departamento y provincia de su elección.
2. Eliminar columnas innecesarias.
3. Renombrar al menos dos columnas a nombres más cortos.
4. Crear 2 variables:
 - Transferencia promedio.
 - hogares que no han recibido el programa (se obtiene restando los abonados de los afiliados)
5. Exportar la nueva tabla en un csv

Sobre esta nueva tabla responder las siguientes preguntas:

1. ¿Qué distritos están afiliados en dicha provincia?
2. ¿Cuántos afiliados en total tiene la provincia?
3. ¿Cuántos hogares han sido abonados?
4. ¿Cuánto ha sido el total de la transferencia del programa juntos en esa provincia?
5. ¿Cual ha sido la transferencia promedio por hogar? ¿Cuál es el valor mínimo y cuál es el máximo?
6. ¿Cuántos hogares no han recibido abonos del programa juntos?

Las respuestas deben guardarse en variables (objetos) separadas.