R básico para ciencia de datos - Clase 4

Ph D.Stephanie Hereira-Pacheco
Centro Tlaxcala de Biología de la Conducta
UATx

16 - 02 - 2022

Manipulación de datos con Rbase

- Filtrado y obtención de subconjuntos: seleccionando columnas, filtrando filas y creando nuevas columnas.
- Modificando valores en una columa
- Juntando tablas: cbind, rbind y merge
- Resumiendo datos con aggregate()
- Ordenando tabla
- Ejemplo aplicado

Filtrado y obtención de subconjuntos

En los capítulos anteriores ya vimos algunos ejemplos de subconjuntos y filtrados, retomaremos algunos de estos y también veremos otros nuevos. Para ejemplificar mejor esta parte, trabajaremos con el dataset de **ToothGrowth** y utilizaré la función `head()` para ver solo las 6 primeras filas:

```
data("ToothGrowth")
```

1. Seleccionando columnas

```
head(ToothGrowth[1:2])
```

```
##    len supp
## 1  4.2   VC
## 2 11.5   VC
## 3  7.3   VC
## 4  5.8   VC
## 5  6.4   VC
## 6 10.0   VC
```

```
head(ToothGrowth[-3])
```

```
##    len supp
## 1  4.2   VC
## 2 11.5   VC
## 3  7.3   VC
## 4  5.8   VC
## 5  6.4   VC
## 6 10.0   VC
```

1. Seleccionando columnas

```
head(ToothGrowth[c("dose", "len")])
```

```
##   dose  len
## 1  0.5  4.2
## 2  0.5 11.5
## 3  0.5  7.3
## 4  0.5  5.8
## 5  0.5  6.4
## 6  0.5 10.0
```

```
head(ToothGrowth[-1:-2,1:2])
```

```
##    len supp
## 3  7.3   VC
## 4  5.8   VC
## 5  6.4   VC
## 6 10.0   VC
## 7 11.2   VC
## 8 11.2   VC
```

2. Filtrando filas

Indexando podemos filtrar nuestra tabla usando:

```
head(ToothGrowth[-1:-2,])
```

```
##    len supp dose
## 3  7.3   VC  0.5
## 4  5.8   VC  0.5
## 5  6.4   VC  0.5
## 6 10.0   VC  0.5
## 7 11.2   VC  0.5
## 8 11.2   VC  0.5
```

```
head(ToothGrowth[which(ToothGrowth$supp == "VC"),])
```

```
##    len supp dose
## 1  4.2   VC  0.5
## 2 11.5   VC  0.5
## 3  7.3   VC  0.5
## 4  5.8   VC  0.5
## 5  6.4   VC  0.5
```

2. Filtrando filas

```
ind<-ToothGrowth$len>27
ind2<- ToothGrowth$sup=="OJ"
head(ToothGrowth[ind,])
```

```
##     len supp dose
## 23 33.9   VC    2
## 26 32.5   VC    2
## 30 29.5   VC    2
## 50 27.3   OJ    1
## 56 30.9   OJ    2
## 58 27.3   OJ    2
```

```
head(ToothGrowth[ind2,])
```

```
##     len supp dose
## 31 15.2   OJ  0.5
## 32 21.5   OJ  0.5
## 33 17.6   OJ  0.5
## 34  9.7   OJ  0.5
```

2. Filtrando filas: NA's

Un detalle importante en los **dataframes** son la inserción de los llamados *"NA"* que son datos que no han sido introducidos por error o porque no se tienen los datos. En algunos análisis estos tipos de datos no son deseados porque pueden generar ruido por lo que se sugiere identificarlos, omitirlos y/o eliminarlos.

Veamos un ejemplo:

```
promedio_clases<- data.frame(clase=c("M", "M", "M", "B", "B", "B"),
                             notas=c(7,8,9, 10, 9, NA))
mean(promedio_clases$notas)
```

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

2. Filtrando filas: NA's

Una opción que tenemos es colocar el argumento **na.rm=TRUE** para que nos ignore los NA's la función:

```
mean(promedio_clases$notas, na.rm=TRUE)
```

```
## [1] 8.6
```

Pero si queremos identificar cuales son los datos que nos dan NA y filtrarlos usamos la función *is.na()*:

```
is.na(promedio_clases$notas)
```

```
## [1] FALSE FALSE FALSE FALSE FALSE  TRUE
```

2. Filtrando filas: NA's

Indexando también podemos remover estos **NA**:

```
promedio_clases[!is.na(promedio_clases$notas),]
```

```
##   clase notas
## 1     M     7
## 2     M     8
## 3     M     9
## 4     B    10
## 5     B     9
```

También, R tiene una función que nos hace más fácil esto:

```
na.omit(promedio_clases)
```

```
##   clase notas
## 1     M     7
## 2     M     8
## 3     M     9
## 4     B    10
```

Filtrando con *subset()*

Hay una función en R básico que nos permite obtener subconjuntos o filtrar las filas de nuestras tablas de manera más intuitiva.

```
head(subset(ToothGrowth, dose=="0.5"))
```

```
##    len supp dose
## 1  4.2   VC  0.5
## 2 11.5   VC  0.5
## 3  7.3   VC  0.5
## 4  5.8   VC  0.5
## 5  6.4   VC  0.5
## 6 10.0   VC  0.5
```

```
subset(ToothGrowth, dose=="0.5" & supp =="OJ" & len >10)
```

```
##     len supp dose
## 31 15.2   OJ  0.5
## 32 21.5   OJ  0.5
## 33 17.6   OJ  0.5
## 35 14.5   OJ  0.5
```

3. Creando una nueva columna

Para crear una nueva columna de una *dataframe* podemos utilizar varios métodos:
- El primero sería declarar una variable nueva de la *dataframe* y de ahí indicarle que se desea como nueva columna en la *dataframe* utilizando el *"$"*. Creemos una data como la anterior de las notas:

```
promedio_notas<- data.frame(estudiante = c("E1", "E2", "E3",
                                           "E1", "E2", "E3"),
                            clase=c("M", "M", "M",
                                    "B", "B", "B"),
                             notas=c(7,8,9, 10, 9, 8))

promedio_notas$ponderacion <- c(0.7,0.7,0.7, 0.3,0.3,0.3)
```

3. Creando una nueva columna

```
promedio_notas
```

```
##   estudiante clase notas ponderacion
## 1         E1     M     7         0.7
## 2         E2     M     8         0.7
## 3         E3     M     9         0.7
## 4         E1     B    10         0.3
## 5         E2     B     9         0.3
## 6         E3     B     8         0.3
```

- También podemos usar las funciones `within()` y `transform()`:

```
promedio_notas<-within(promedio_notas,
                       nota_ponderada <- notas*ponderacion)

promedio_notas<-transform(promedio_notas,
                          nota_ponderada = notas*ponderacion)
```

3. Creando una nueva columna

```
promedio_notas
```

```
##   estudiante clase notas ponderacion nota_ponderada
## 1         E1     M     7          0.7             4.9
## 2         E2     M     8          0.7             5.6
## 3         E3     M     9          0.7             6.3
## 4         E1     B    10          0.3             3.0
## 5         E2     B     9          0.3             2.7
## 6         E3     B     8          0.3             2.4
```

aggregate() : Resumiendo los datos

Con la función *aggregate()* podemos resumir nuestros datos, por ejemplo:

```
aggregate(notas ~ clase, data = promedio_notas, mean)
```

```
##   clase notas
## 1     B     9
## 2     M     8
```

```
aggregate(notas ~ estudiante, data = promedio_notas, median)
```

```
##   estudiante notas
## 1         E1   8.5
## 2         E2   8.5
## 3         E3   8.5
```

```
aggregate(notas ~ clase+estudiante, data = promedio_notas, sum)
```

```
##   clase estudiante notas
## 1     B         E1    10
## 2     M         E1     7
```

Renombrando columnas y datos

Para renombrar columnas podemos sólo reescribir el nuevo nombre por el viejo, por ejemplo:

```
colnames(promedio_notas)
```

```
## [1] "estudiante"     "clase"          "notas"          "ponderacion"
## [5] "nota_ponderada"
```

```
colnames(promedio_notas)[2] <- "curso"
colnames(promedio_notas)
```

```
## [1] "estudiante"     "curso"          "notas"          "ponderacion"
## [5] "nota_ponderada"
```

También si queremos cambiar todos los nombres de las columnas (no lo correré pero dejaré el ejemplo):

```
names(promedio_notas) <- c("a", "b", "c", "d")
```

Renombrando valores en una columna:

```
promedio_notas$curso <- ifelse(
  promedio_notas$curso == "M", "Matemáticas", "Biología")

promedio_notas$curso
```

```
## [1] "Matemáticas" "Matemáticas" "Matemáticas" "Biología"    "Biologí
## [6] "Biología"
```

```
## [1] "M" "M" "M" "B" "B" "B"
```

cbind() y rbind()

`cbind()` y `rbind()` son funciones que nos permiten combinar y juntar vectores, matrices y tablas.

"c" es para juntar por columnas (horizontalmente, una al lado de otra) y "r" para combinar combinar por filas (verticalmente, una abajo de otra).

Ejemplos:

cbind:

```
correcion_nota<- c(10,9,8,8,9,10)
cbind(promedio_notas, correcion_nota)
```

```
##   estudiante curso notas ponderacion nota_ponderada correcion_nota
## 1         E1     M     7         0.7             4.9            10
## 2         E2     M     8         0.7             5.6             9
## 3         E3     M     9         0.7             6.3             8
## 4         E1     B    10         0.3             3.0             8
## 5         E2     B     9         0.3             2.7             9
## 6         E3     B     8         0.3             2.4            10
```

cbind() y rbind()

```
knitr::kable(cbind(promedio_notas, promedio_notas))
```

estud	notas	ponder	nota_pond	NA	estud	notas	ponder	nota_pond
E1	M	7	0.7	4.9	E1	M	7	0.7
E2	M	8	0.7	5.6	E2	M	8	0.7
E3	M	9	0.7	6.3	E3	M	9	0.7
E1	B	10	0.3	3.0	E1	B	10	0.3
E2	B	9	0.3	2.7	E2	B	9	0.3
E3	B	8	0.3	2.4	E3	B	8	0.3

cbind() y rbind()

rbind:

```
notas_fisica<- data.frame(estudiante = c("E1", "E2", "E3"),
                          curso=c("F", "F", "F", "F", "F", "F"),
                           notas=c(7, 9, 8))
rbind(promedio_notas, notas_fisica)
```

Error in rbind(deparse.level, . . .) : numbers of columns of arguments do not match

cbind() y rbind()

Vemos este error debido a que tanto `cbind` como `rbind` requieren objetos (tablas, vectores y matrices) con las mismas dimensiones.

Y en el caso de `rbind`, las columnas deben tener el mismo nombre (colnames).

En este caso quería agregar unas filas abajo en esta tabla pero nos faltó la columna de ponderación, probemos de nuevo:

cbind() y rbind()

```
notas_fisica<- data.frame(estudiante = c("E1", "E2", "E3"),
                          curso=c("F", "F", "F"),
                           notas=c(7, 9, 8),
                          ponderacion = c(0.1, 0.1, 0.1),
                        nota_ponderada= c(10, 9, 8))
rbind(promedio_notas, notas_fisica)
```

```
##   estudiante curso notas ponderacion nota_ponderada
## 1         E1     M     7         0.7             4.9
## 2         E2     M     8         0.7             5.6
## 3         E3     M     9         0.7             6.3
## 4         E1     B    10         0.3             3.0
## 5         E2     B     9         0.3             2.7
## 6         E3     B     8         0.3             2.4
## 7         E1     F     7         0.1            10.0
## 8         E2     F     9         0.1             9.0
## 9         E3     F     8         0.1             8.0
```

Uniendo tablas con *merge()*

Con la función `merge()` podemos unir dos *dataframes* que tengan los mismos nombres en filas y columnas:

```
x <- data.frame(k1 = c(1,3,3,4,5), k2 = c("a1","a2","a3","a4","a5"), da
y <- data.frame(k3 = c(2,2,6,4,5), k2 = c("a1","a2","a3","a4","a5"), da

merge(x, y, by = "k2")
```

```
##   k2 k1 data.x k3 data.y
## 1 a1  1      1  2      1
## 2 a2  3      2  2      2
## 3 a3  3      3  6      3
## 4 a4  4      4  4      4
## 5 a5  5      5  5      5
```

Uniendo tablas con *merge()*

```
merge(x, y, by = c("k2", "data"), all = TRUE)
```

```
##   k2 data k1 k3
## 1 a1    1  1  2
## 2 a2    2  3  2
## 3 a3    3  3  6
## 4 a4    4  4  4
## 5 a5    5  5  5
```

Vimos cómo se pueden unir tablas con una o más columnas en común, aunque es preferible que las columnas que no se van a unir tengan nombres diferentes.

Ordenando tablas por un criterio o columna

Podemos ordenar nuestra tabla con uno o más criterios:

```
promedio_notas <- promedio_notas[order(promedio_notas$notas,
                                       decreasing = TRUE),]
promedio_notas
```

```
##   estudiante curso notas ponderacion nota_ponderada
## 4         E1     B    10         0.3             3.0
## 3         E3     M     9         0.7             6.3
## 5         E2     B     9         0.3             2.7
## 2         E2     M     8         0.7             5.6
## 6         E3     B     8         0.3             2.4
## 1         E1     M     7         0.7             4.9
```

Ordenando tablas por un criterio o columna

```
promedio_notas <- promedio_notas[order(promedio_notas$notas,
                                       promedio_notas$ponderacion),]
promedio_notas
```

```
##   estudiante curso notas ponderacion nota_ponderada
## 1         E1     M     7         0.7             4.9
## 6         E3     B     8         0.3             2.4
## 2         E2     M     8         0.7             5.6
## 5         E2     B     9         0.3             2.7
## 3         E3     M     9         0.7             6.3
## 4         E1     B    10         0.3             3.0
```

Funciones adicionales de agregación

Para calcular fácilmente los promedios o sumas de todas las columnas y las filas usamos `rowMeans()`, `colMeans()`, `rowSums()` y `colSums()`.

```
examen <- data.frame("q1" = c(1, 0, 0, 0, 0),
                     "q2" = c(1, 0, 1, 1, 0),
                     "q3" = c(1, 0, 1, 0, 0),
                     "q4" = c(1, 1, 1, 1, 1),
                     "q5" = c(1, 0, 0, 1, 1))
rowMeans(examen)
```

```
## [1] 1.0 0.2 0.6 0.6 0.4
```

```
colMeans(examen)
```

```
##  q1  q2  q3  q4  q5
## 0.2 0.6 0.4 1.0 0.6
```

```
rowSums(examen)
```

```
## [1] 5 1 3 3 2
```

Funciones adicionales de agregación

```
colSums(examen)
```

```
## q1 q2 q3 q4 q5
##  1  3  2  5  3
```

También, si queremos saber cuántas columnas y filas tienen nuestros datos, además de `dim()` y `str()` podemos usar:

```
nrow(examen)
```

```
## [1] 5
```

```
ncol(examen)
```

```
## [1] 5
```

Ejemplo aplicado

Las siguientes dos tablas muestran los resultados de dos encuestas hechas a 10 personas. En la primera encuesta preguntaron su género y su edad, y en la segunda preguntaron su superhéroe favorito y cantidad de tatuajes que tenía.

```
primera<- data.frame(Nombre= c("Astrid", "Lea", "Sarina","Remon",
                               "Letizia", "Babice", "Jonas",
                               "Wendy","Nivedithia", "Gioia"),
                     Sexo= c("F", "F", "F", "M", "F", "F",
                             "M", "F", "F", "F"),
                     Edad= c(30,25,25,29,22,22,35,19,32,21))

segunda<- data.frame(Nombre= c("Astrid", "Lea", "Sarina", "Remon",
                               "Letizia", "Babice", "Jonas", "Wendy",
                               "Nivedithia", "Gioia"),
                     Superheroe= c("Batman", "Superman", "Batman",
                                   "Spiderman","Batman","Antman",
                                   "Batman","Superman",
                                   "Maggot", "Superman"),
                     Tatuajes= c(11,15,12,5,65,3,9,13,900,0))
```

Ejemplo aplicado

Nombre	Sexo	Edad
Astrid	F	30
Lea	F	25
Sarina	F	25
Remon	M	29
Letizia	F	22
Babice	F	22
Jonas	M	35
Wendy	F	19
Nivedithia	F	32
Gioia	F	21

Nombre	Superheroe	Tatuajes
Astrid	Batman	11
Lea	Superman	15
Sarina	Batman	12
Remon	Spiderman	5
Letizia	Batman	65
Babice	Antman	3
Jonas	Batman	9
Wendy	Superman	13
Nivedithia	Maggot	900
Gioia	Superman	0

Ejemplo aplicado

Para hacer:

1. Combina las dos tablas en una sola y completa las siguientes asignaciones.
2. ¿Cuál es la edad media de las mujeres y hombres por separado?
3. ¿Cuál fue el número más alto de tatuajes en un hombre?
4. ¿Cuál es el porcentaje de personas debajo de 32 años que son mujeres?
5. Agrega una nueva columna a la data llamada `tatuajes.por.año` que muestre cuántos tatuajes por año se ha hecho cada persona por cada año en su vida.
6. ¿Cuál persona tiene el mayor número de tatuajes por año?
7. ¿Cuáles son los nombres de las mujeres cuyo superheroe favorito es superman?
8. ¿Cuál es la mediana del número de tatuajes de cada persona que está por encima de los 20 años y que su personaje favorito es Batman?

Resolviendo

1.Combina las dos tablas en una sola y completa las siguientes asignaciones.

```
encuestas<- merge(primera, segunda, by = "Nombre")
```

2. ¿Cuál es la edad media de las mujeres y hombres por separado? cómo podemos hacer esto?

```
aggregate(Edad ~ Sexo, data = encuestas, mean)
```

```
##   Sexo Edad
## 1    F 24.5
## 2    M 32.0
```

Ejemplo aplicado

3. ¿Cuál fue el número más alto de tatuajes en un hombre?

```
males<- subset(encuestas, Sexo=="M")
max(males$Tatuajes)
```

```
## [1] 9
```

4. ¿Cuál es el porcentaje de mujeres debajo de 32 años?

```
fem<- subset(encuestas, Sexo=="F")
fem_32<- fem[fem$Edad<32,]

(nrow(fem_32)/nrow(fem))*100
```

```
## [1] 87.5
```

Ejemplo aplicado

5.Agrega una nueva columna a la data llamada `tatuajes.por.año` que muestre cuántos tatuajes por año se ha hecho cada persona por cada año en su vida.

```
encuestas$tatuajes.por.año<- encuestas$Tatuajes/encuestas$Edad
encuestas$tatuajes.por.año
```

```
##   [1]  0.3666667  0.1363636  0.0000000  0.2571429  0.6000000  2.95454
##   [7] 28.1250000  0.1724138  0.4800000  0.6842105
```

6. ¿Cuál persona tiene el mayor número de tatuajes por año?

```
mayor_tatuaje<-which.max(encuestas$tatuajes.por.año)
encuestas[mayor_tatuaje,]
```

```
##       Nombre Sexo Edad Superheroe Tatuajes tatuajes.por.año
## 7 Nivedithia    F   32     Maggot      900           28.125
```

Ejemplo aplicado

7.¿Cuáles son los nombres de las mujeres cuyo superheroe favorito es superman?

```
sup<-fem[fem$Superheroe=="Superman",]
sup$Nombre
```

```
## [1] "Gioia" "Lea"   "Wendy"
```

8.¿Cuál es la mediana del número de tatuajes de cada persona que está por encima de los 20 años y que su personaje favorito es superman?

```
ocho<- subset(encuestas, Edad>20 & Superheroe =="Batman")
median(ocho$Tatuajes)
```

```
## [1] 11.5
```