Clase 1.3
Acomodo de datos

Marcos Rosetti y Luis Pacheco-Cobos

Estadística y Manejo de Datos con R (EMDR) — Virtual

Acomodo de datos: `dplyr`

Acomodo de datos: `dplyr`

- Un extenso paquete para el manejo de datos y que lleva a R a la era moderna de la ciencia de datos.
- Nombres de funciones basadas en los verbos de las acciones que llevan a cabo.
- Uso de parámetros intuitivos y sencillos, cercanos al lenguaje natural.

Acomodo de datos: `dplyr`

- Usaremos `nycflights13`, un df que contiene los 336776 vuelos que salieron de la ciudad de Nueva York en 2013.

```
install.packages("dplyr")
library(dplyr)
install.packages("nycflights13")
library(nycflights13)
```

Acomodo de datos: **dplyr**

-¿Qué contiene `flights`?

```
dim(flights)
```

```
## [1] 336776     19
```

```
head(flights, 4)
```

```
## # A tibble: 4 × 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      517            515         2      830            819
## 2  2013     1     1      533            529         4      850            830
## 3  2013     1     1      542            540         2      923            850
## 4  2013     1     1      544            545        -1     1004           1022
## # … with 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

Acomodo de datos: `dplyr`

- `filter()` filtra filas utilizando criterios boleanos.

Acomodo de datos: `dplyr`

- Podemos listar los criterios para hacer el filtrado después de los datos (`flights`), separándolos por comas.

```
f1 <- filter(flights, month == 1, day == 1)
tail(f1)
```

```
## # A tibble: 6 × 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1     2353           2359        -6      418            442
## 2  2013     1     1     2356           2359        -3      425            437
## 3  2013     1     1       NA           1630        NA       NA           1815
## 4  2013     1     1       NA           1935        NA       NA           2240
## 5  2013     1     1       NA           1500        NA       NA           1825
## 6  2013     1     1       NA            600        NA       NA            901
## # … with 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

Acomodo de datos: **dplyr**

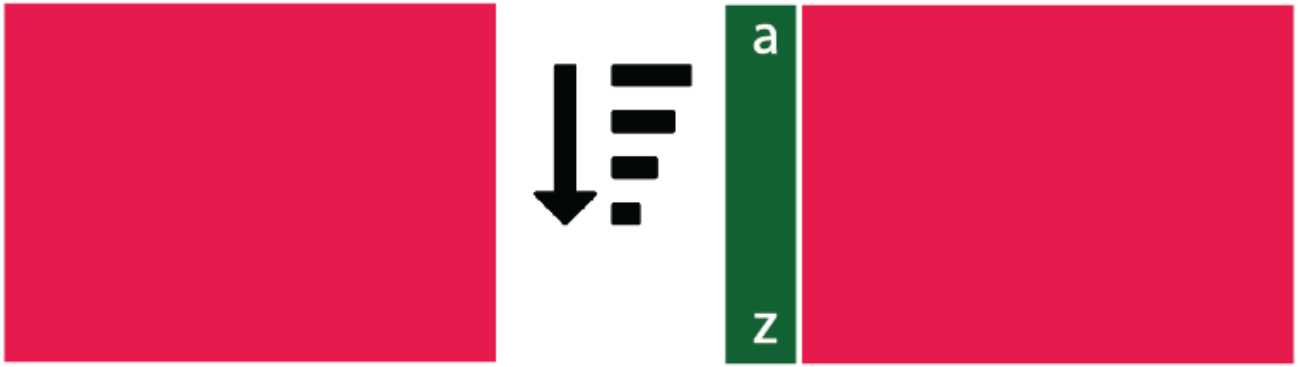
```r
# La manera tradicional de obtener los casos del 1-enero-2013 es:
flights[flights$month == 1 & flights$day == 1, ]

# ¿Qué hace slice()?
slice(flights, 1:10)
# Nota: observa que los datos están ordenados cronológicamente.

# El equivalente tradicional para filtrar las primeras 10 filas sería:
flights[1:10, ]
```

Acomodo de datos: `dplyr`

- `arrange()` re-arregla el orden de las filas.



Acomodo de datos: `dplyr`

- Podemos listar las columnas como criterio para hacer el arreglo, separándolas con coma.

```
f1 <- arrange(flights, dep_delay, day, month)
head(f1)
```

```
## # A tibble: 6 × 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013    12     7     2040           2123       -43       40           2352
## 2  2013     2     3     2022           2055       -33     2240           2338
## 3  2013    11    10     1408           1440       -32     1549           1559
## 4  2013     1    11     1900           1930       -30     2233           2243
## 5  2013     1    29     1703           1730       -27     1947           1957
## 6  2013     8     9      729            755       -26     1002            955
## # … with 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

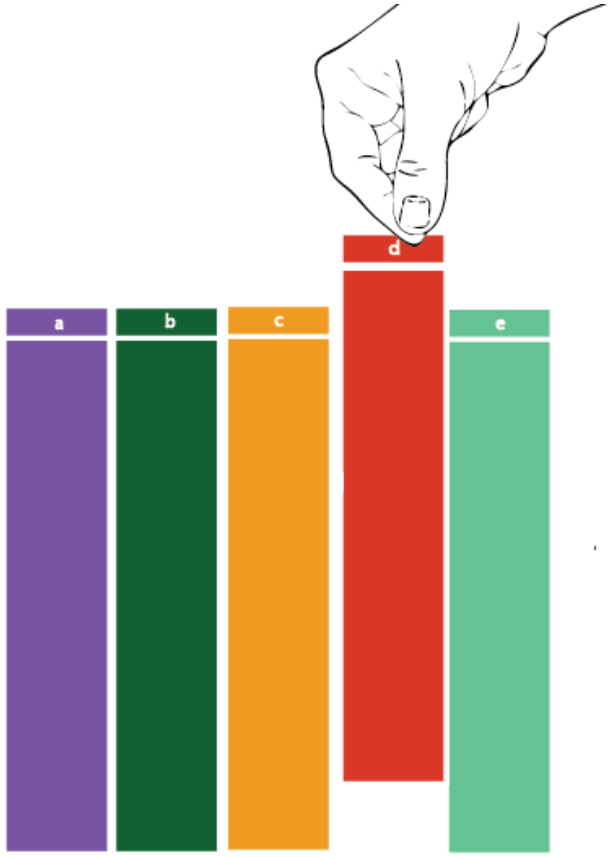
Acomodo de datos: **dplyr**

```
# La manera tradicional
flights[order(flights$year, flights$month, flights$day), ]
flights[order(flights$arr_delay, decreasing=TRUE), ]

# ¿Qué hace desc()?
arrange(flights, desc(arr_delay))
```

Acomodo de datos: `dplyr`

- `select()` selecciona columnas.



Acomodo de datos: `dplyr`

- Podemos listar las columnas a selecionar, separándolas con coma.

```
f1 <- select(flights, year, month, day)
head(f1)
```

```
## # A tibble: 6 × 3
##    year month   day
##   <int> <int> <int>
## 1  2013     1     1
## 2  2013     1     1
## 3  2013     1     1
## 4  2013     1     1
## 5  2013     1     1
## 6  2013     1     1
```

```
select(flights, year:day) # desde year hasta day
select(flights, -(year, day)) # todas menos year y day
select(flights, año = year) # podemos cambiar el nombre de la columna al seleccionarla
```

Acomodo de datos: `dplyr`

- `distinct()` extrae filas únicas.

Acomodo de datos: `dplyr`

- Podemos listar una columna para la que deseamos encontrar filas únicas, seleccionándola desde el df `flights`.

```
f1 <- distinct(select(flights, tailnum))
head(f1)
```

```
## # A tibble: 6 × 1
##   tailnum
##   <chr>
## 1 N14228
## 2 N24211
## 3 N619AA
## 4 N804JB
## 5 N668DN
## 6 N39463
```

Acomodo de datos: `dplyr`

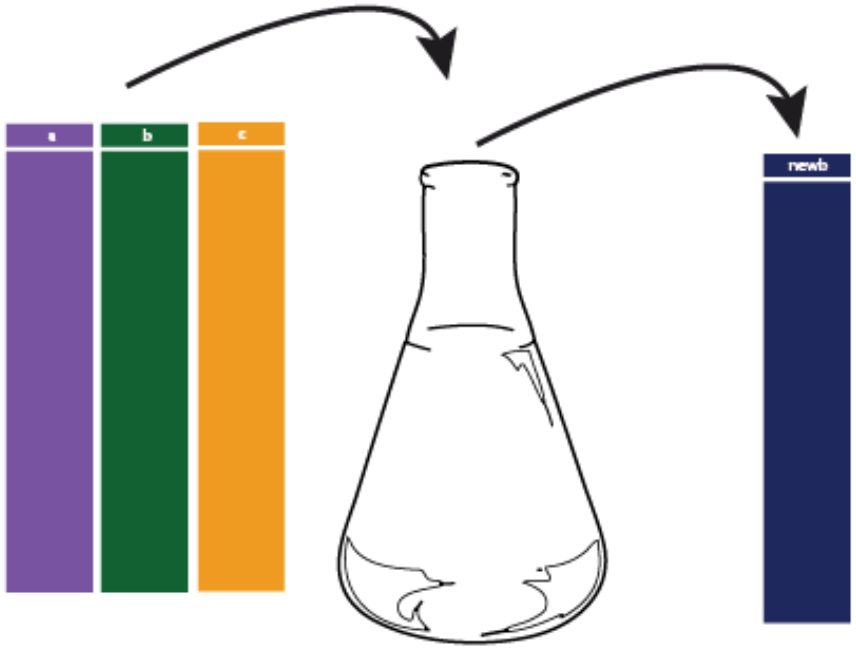
- Podemos listar las columnas para las que deseamos encontrar filas únicas, seleccionándolas y separándolas con coma después del df `flights`.

```
f2 <- distinct(select(flights, origin, dest))
head(f2)
```

```
## # A tibble: 6 × 2
##   origin dest
##   <chr>  <chr>
## 1 EWR    IAH
## 2 LGA    IAH
## 3 JFK    MIA
## 4 JFK    BQN
## 5 LGA    ATL
## 6 EWR    ORD
```

Acomodo de datos: `dplyr`

- `mutate()` crea nuevas columnas.



Acomodo de datos: `dplyr`

- `mutate()` crea nuevas columnas.

Acomodo de datos: **dplyr**

- Agregamos columnas y especificamos su contenido.

```
f1 <- mutate(flights, gain = arr_delay - dep_delay,
             speed = gain/air_time * 60)
head(f1)
```

```
## # A tibble: 6 × 21
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      517            515         2      830            819
## 2  2013     1     1      533            529         4      850            830
## 3  2013     1     1      542            540         2      923            850
## 4  2013     1     1      544            545        -1     1004           1022
## 5  2013     1     1      554            600        -6      812            837
## 6  2013     1     1      554            558        -4      740            728
## # … with 13 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>, gain <dbl>, speed <dbl>
```

```
# ¿Qué hace diferente transmmute()?
transmute(flights, gain = arr_delay - dep_delay,
gain_per_hour = gain/(air_time/60))
```

Acomodo de datos: `dplyr`

- `group_by()`, `ungroup()` agrupa/desagrupa las variables según uno o más factores.

Acomodo de datos: `dplyr`

- Inidcamos la(s) columna(s) que contiene(n) los factoresque utilizaremos como criterio de agrupación.

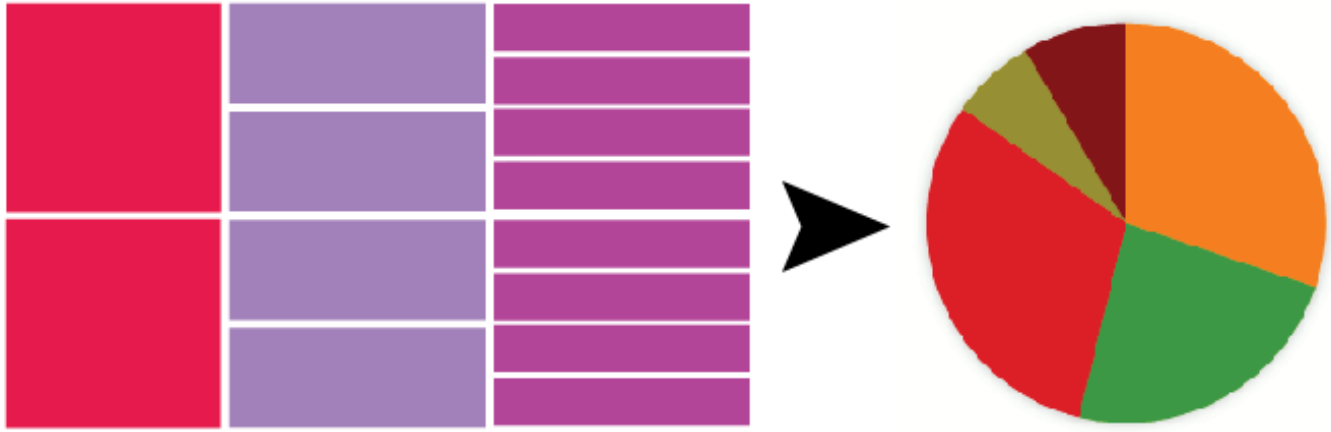
```
f.g <- group_by(flights, tailnum)
head(f.g)
```

```
## # A tibble: 6 × 19
## # Groups:   tailnum [6]
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      517            515         2      830            819
## 2  2013     1     1      533            529         4      850            830
## 3  2013     1     1      542            540         2      923            850
## 4  2013     1     1      544            545        -1     1004           1022
## 5  2013     1     1      554            600        -6      812            837
## 6  2013     1     1      554            558        -4      740            728
## # … with 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
# No hay cambios visibles, pero las columnas están agrupadas en el espacio
# de trabajo, podemos usar ungroup() para remover esta agrupación.
```

Acomodo de datos: `dplyr`

- `summarise()` resume las columas según las variables agrupadas.



Acomodo de datos: `dplyr`

- Inidcamos la(s) columna(s) que contiene(n) los factoresque utilizaremos como criterio de agrupación. Y luego hacemos el resumen.

```
f.g <- group_by(flights, tailnum)
f.sum <- summarise(f.g, n = n(),
                   mean_delay = mean(dep_delay, na.rm = TRUE))
f.sum
```

```
## # A tibble: 4,044 × 3
##    tailnum     n mean_delay
##    <chr>   <int>      <dbl>
##  1 D942DN      4     31.5
##  2 N0EGMQ    371      8.49
##  3 N10156    153     17.8
##  4 N102UW     48      8
##  5 N103US     46     -3.20
##  6 N104UW     47      9.94
##  7 N10575    289     22.7
##  8 N105UW     45      2.58
##  9 N107US     41     -0.463
## 10 N108UW     60      4.22
## # … with 4,034 more rows
```

Acomodo de datos: **dplyr**

- Otras funciones de interés: `sample_n()` y `sample_frac()`

```
sample_n(flights, 10) # muestrea 10 elementos
```

```
## # A tibble: 10 × 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     6    15      755            800        -5     1042           1105
##  2  2013     7    18      740            747        -7      851            902
##  3  2013     4    23      811            815        -4     1042           1038
##  4  2013     5     7     1618           1557        21     1834           1914
##  5  2013     7    17     1552           1600        -8     1724           1718
##  6  2013     8    19      822            830        -8     1003           1018
##  7  2013     1    24     1736           1619        77     1837           1723
##  8  2013     7    17     1458           1448        10     1724           1745
##  9  2013    12    24      513            515        -2      813            814
## 10  2013    11    17      937            945        -8     1231           1230
## # … with 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
# Intenta tambien sample_frac(flights, 0.01) para muestrear el 1 %
```

Acomodo de datos: `dplyr`

- Otras funciones de interés: `n()`, `n_distinct()`, `first()`, `last()` y `nth(x, n)`

```
destinations <- group_by(flights, dest)
summarise(destinations, planes = n_distinct(tailnum), flights = n())
```

```
## # A tibble: 105 × 3
##    dest  planes flights
##    <chr>  <int>   <int>
##  1 ABQ      108     254
##  2 ACK       58     265
##  3 ALB      172     439
##  4 ANC        6       8
##  5 ATL     1180   17215
##  6 AUS      993    2439
##  7 AVL      159     275
##  8 BDL      186     443
##  9 BGR       46     375
## 10 BHM       45     297
## # … with 95 more rows
```

Acomodo de datos: `dplyr`

Licencia CC BY