Sesión II: Manipulando datos

Guillermo de Anda-Jáuregui y Cristóbal Fresno

2019

¿Qué es tener datos limpios?

A	B	C	D
trabajo	puesto	nombre	inicio y fin
administracion_limpieza	limpieza	sanchez perez sa	Nov/2015 – Ene/2020
jurídico_legal	practicante	lopez dominguez l	Ene/2016 – Feb/2017
jurídico_legal	practicante	romero madero Ac	Ene/2015 – Jun/2017
operativo_ventas	ventas	jimenez ordoñez l	Ene/2016 – Jun/2018
administracion_transporte	chofer	Perez rubio Diana	Feb/2011 – Abr/2019
administracion_finanzas	analista	garcía moreno Da	Feb/2010 – Feb/ 2020
operativo_compras	compras	cabrera Miguel	Ene/2009 – Ene/ 2020

Los datos limpios

- Cada columna es una variable

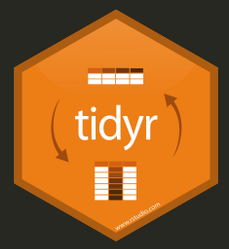
- Cada renglón es una observación

ID	direccion	subdireccion	puesto	apellido_paterno	apellido_materno	nombre	fecha_inicio	fecha_fin	interno
1	administracion	limpieza	limpieza	Sanchez	Pérez	Santiago	Nov/2015	Ene/2020	si
2	jurídico	legal	practicante	Lopez	Domínguez	Luisa	Ene/2016	Feb/2017	no
3	jurídico	legal	practicante	Romero	Madero	Agustín	Ene/2015	Jun/2017	no
4	operativo	ventas	ventas	Jimenez	Ordoñez	Luis	Ene/2016	Jun/2018	no
5	administracion	transporte	chofer	Perez	Rubio	Diana	Feb/2011	Abr/2019	si
6	administracion	finanzas	analista	García	Moreno	Daniela	Feb/2010	Feb/2020	si
7	operativo	compras	compras	Cabrera	NA	Miguel	Ene/2009	Ene/2020	si

```
my_data
```

```
## # A tibble: 7 x 10
##      ID direccion subdireccion puesto apellido_paterno apellido_materno
##   <dbl> <chr>     <chr>        <chr>  <chr>            <chr>
## 1     1 administ… limpieza     limpi… Sanchez          Pérez
## 2     2 jurídico  legal        pract… Lopez            Domínguez
## 3     3 jurídico  legal        pract… Romero           Madero
## 4     4 operativo ventas       ventas Jimenez          Ordoñez
## 5     5 administ… transporte   chofer Perez            Rubio
## 6     6 administ… finanzas     anali… García           Moreno
## 7     7 operativo compras      compr… Cabrera          NA
## # … with 4 more variables: nombre <chr>, fecha_inicio <chr>,
## #   fecha_fin <chr>, interno <chr>
```

El paquete tidyR



Datos limpios, código limpio

- Si tenemos datos limpios, podemos pensar en usar código parecido para diferentes problemas.

- Código limpio: más legible.
  - Más tiempo para pensar en nuestros análisis.

El paquete magrittR

Funciones anidadas

```
x <- 1:10

(exp(sqrt(log(x))))
```

```
##  [1] 1.000000 2.299185 2.852361 3.245956 3.556000 3.813572 4.034808
##  [8] 4.229259 4.403078 4.560477
```

Objetos intermedios

```
x <- 1:10

y <- log(x)

z <- sqrt(y)

w <- exp(z)

w
```

```
##  [1] 1.000000 2.299185 2.852361 3.245956 3.556000 3.813572 4.034808
##  [8] 4.229259 4.403078 4.560477
```

Sobreescribir mi objeto

```
x <- 1:10

x <- log(x)

x <- sqrt(x)

x <- exp(x)

x
```

```
##  [1] 1.000000 2.299185 2.852361 3.245956 3.556000 3.813572 4.034808
##  [8] 4.229259 4.403078 4.560477
```

El pipe

%>%

El pipe %>%

```
x <- 1:10

my_result <-

  x %>%
log() %>%
sqrt() %>%
  exp

my_result
```

```
##  [1] 1.000000 2.299185 2.852361 3.245956 3.556000 3.813572 4.034808
##  [8] 4.229259 4.403078 4.560477
```

Otros pipes

la TE

```
rnorm(100) %>%
matrix(ncol = 2) %T>%
plot() %>%
str()
```

Otros pipes

la EXPLOSIÓN

```
mtcars %$%
cor(disp, mpg)
```

```
## [1] -0.8475514
```

¿Qué queremos manipular?

En un set de datos rectangulares...

- Queremos modificar variables completas.

- Queremos hacer operaciones y sacar nuevas variables.

- Queremos seleccionar observaciones que cumplan características.

- Queremos agrupar.

- Queremos sacar descripciones de los datos.

- Queremos reordenar variables.

Dataset: nycflights13

Filtrados

El paquete dplyr

Una gramática para la manipulación de datos

Tidyverse => Verbos

mutate

select

filter

group_by

summarise

arrange

Analicemos flights

```
flights <- vroom::vroom(file = "data/flights.txt")
```

```
## Observations: 336,776
## Variables: 19
## chr  [ 4]: carrier, tailnum, origin, dest
## dbl  [14]: year, month, day, dep_time, sched_dep_time, dep_delay, arr_time, sched_
## dttm [ 1]: time_hour
##
## Call `spec()` for a copy-pastable column specification
## Specify the column types with `col_types` to quiet this message
```

```
flights
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <dbl> <dbl> <dbl>    <dbl>          <dbl>     <dbl>    <dbl>
##  1  2013     1     1      517            515         2      830
##  2  2013     1     1      533            529         4      850
##  3  2013     1     1      542            540         2      923
##  4  2013     1     1      544            545        -1     1004
##  5  2013     1     1      554            600        -6      812
##  6  2013     1     1      554            558        -4      740
##  7  2013     1     1      555            600        -5      913
##  8  2013     1     1      557            600        -3      709
##  9  2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
## # … with 336,766 more rows, and 12 more variables: sched_arr_time <dbl>,
## #   arr_delay <dbl>, carrier <chr>, flight <dbl>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

Filtrados

Buscamos observaciones (renglones) que cumplan una condición lógica

Verbo filter

filter

```
filter(.data = flights,
       dest  == "SBN")
```

```
## # A tibble: 10 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <dbl> <dbl> <dbl>    <dbl>          <dbl>     <dbl>    <dbl>
##  1  2013    10    18     1820           1745        35     2030
##  2  2013    11     1     2012           1905        67     2221
##  3  2013    11    22     2013           1905        68     2224
##  4  2013    12     1     1241           1215        26     1431
##  5  2013     8    30     1909           1910        -1     2117
##  6  2013     9     1      833            840        -7     1030
##  7  2013     9     8      847            840         7     1043
##  8  2013     9    20     1948           1950        -2     2207
##  9  2013     9    22      837            840        -3     1025
## 10  2013     9    27     2011           1950        21     2209
## # … with 12 more variables: sched_arr_time <dbl>, arr_delay <dbl>,
## #   carrier <chr>, flight <dbl>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dttm>
```

filter

```
flights %>%
  filter(dest  == "HDN")
```

```
## # A tibble: 15 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <dbl> <dbl> <dbl>    <dbl>          <dbl>     <dbl>    <dbl>
##  1  2013     1     5      829            830        -1     1047
##  2  2013     1    12      827            830        -3     1112
##  3  2013     1    19      843            830        13     1123
##  4  2013     1    26      828            830        -2     1114
##  5  2013    12    21      916            830        46     1149
##  6  2013    12    28      913            829        44     1128
##  7  2013     2     2      858            830        28     1124
##  8  2013     2     9       NA            830        NA       NA
##  9  2013     2    16      834            830         4     1114
## 10  2013     2    23      826            830        -4     1050
## 11  2013     3     2      854            830        24     1104
## 12  2013     3     9      838            830         8     1107
## 13  2013     3    16      845            830        15     1154
## 14  2013     3    23      835            830         5     1104
## 15  2013     3    30      825            830        -5     1045
## # … with 12 more variables: sched_arr_time <dbl>, arr_delay <dbl>,
## #   carrier <chr>, flight <dbl>, tailnum <chr>, origin <chr>, dest <chr>,
```

filter

```
flights %>%
  filter(dest  == "HDN",
         !is.na(dep_time)
         )
```

```
## # A tibble: 14 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time
##   <dbl> <dbl> <dbl>    <dbl>          <dbl>     <dbl>    <dbl>
##  1  2013     1     5      829            830        -1     1047
##  2  2013     1    12      827            830        -3     1112
##  3  2013     1    19      843            830        13     1123
##  4  2013     1    26      828            830        -2     1114
##  5  2013    12    21      916            830        46     1149
##  6  2013    12    28      913            829        44     1128
##  7  2013     2     2      858            830        28     1124
##  8  2013     2    16      834            830         4     1114
##  9  2013     2    23      826            830        -4     1050
## 10  2013     3     2      854            830        24     1104
## 11  2013     3     9      838            830         8     1107
## 12  2013     3    16      845            830        15     1154
## 13  2013     3    23      835            830         5     1104
## 14  2013     3    30      825            830        -5     1045
## # … with 12 more variables: sched_arr_time <dbl>, arr_delay <dbl>,
```

slice

Nos deja seleccionar renglones por posición

```
flights %>%
 slice(14:17)
```

```
## # A tibble: 4 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time
##   <dbl> <dbl> <dbl>    <dbl>          <dbl>     <dbl>    <dbl>
## 1  2013     1     1      558            600        -2      923
## 2  2013     1     1      559            600        -1      941
## 3  2013     1     1      559            559         0      702
## 4  2013     1     1      559            600        -1      854
## # … with 12 more variables: sched_arr_time <dbl>, arr_delay <dbl>,
## #   carrier <chr>, flight <dbl>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dttm>
```

select

Permite escoger algunas variables (columnas)

select

```
flights %>%
select(flight, carrier, origin, dest, distance)
```

```
## # A tibble: 336,776 x 5
##    flight carrier origin dest  distance
##     <dbl> <chr>   <chr>  <chr>    <dbl>
##  1   1545 UA      EWR    IAH       1400
##  2   1714 UA      LGA    IAH       1416
##  3   1141 AA      JFK    MIA       1089
##  4    725 B6      JFK    BQN       1576
##  5    461 DL      LGA    ATL        762
##  6   1696 UA      EWR    ORD        719
##  7    507 B6      EWR    FLL       1065
##  8   5708 EV      LGA    IAD        229
##  9     79 B6      JFK    MCO        944
## 10    301 AA      LGA    ORD        733
## # … with 336,766 more rows
```

```
flights %>%
  select(contains("time"))
```

```
## # A tibble: 336,776 x 6
##    dep_time sched_dep_time arr_time sched_arr_time air_time
##       <dbl>          <dbl>    <dbl>          <dbl>    <dbl>
##  1      517            515      830            819      227
##  2      533            529      850            830      227
##  3      542            540      923            850      160
##  4      544            545     1004           1022      183
##  5      554            600      812            837      116
##  6      554            558      740            728      150
##  7      555            600      913            854      158
##  8      557            600      709            723       53
##  9      557            600      838            846      140
## 10      558            600      753            745      138
## # … with 336,766 more rows, and 1 more variable: time_hour <dttm>
```

```
flights %>%
  select_if(is_character)
```

```
## # A tibble: 336,776 x 4
##    carrier tailnum origin dest
##    <chr>   <chr>   <chr>  <chr>
##  1 UA      N14228  EWR    IAH
##  2 UA      N24211  LGA    IAH
##  3 AA      N619AA  JFK    MIA
##  4 B6      N804JB  JFK    BQN
##  5 DL      N668DN  LGA    ATL
##  6 UA      N39463  EWR    ORD
##  7 B6      N516JB  EWR    FLL
##  8 EV      N829AS  LGA    IAD
##  9 B6      N593JB  JFK    MCO
## 10 AA      N3ALAA  LGA    ORD
## # … with 336,766 more rows
```

Podemos concatenar verbos usando el pipe

```
flights %>%
  select("dest", contains("time")) %>%
  filter(dest == "HDN") %>%
  tidyr::drop_na() %>%
  slice(1:7)
```

```
## # A tibble: 7 x 7
##   dest  dep_time sched_dep_time arr_time sched_arr_time air_time
##   <chr>    <dbl>          <dbl>    <dbl>          <dbl>    <dbl>
## 1 HDN        829            830     1047           1111      243
## 2 HDN        827            830     1112           1111      259
## 3 HDN        843            830     1123           1111      260
## 4 HDN        828            830     1114           1111      265
## 5 HDN        916            830     1149           1117      254
## 6 HDN        913            829     1128           1116      239
## 7 HDN        858            830     1124           1111      246
## # … with 1 more variable: time_hour <dttm>
```

pull

Permite sacar una variable como vector

```
flights %>%
  pull("dest") %>%
  head
```

```
## [1] "IAH" "IAH" "MIA" "BQN" "ATL" "ORD"
```

Agrupamientos

group_by me permite agregar por variables categóricas (o coercible a variables categóricas).

Esto me servirá para hacer resúmenes después.

Agrupamientos

```
flights %>%
group_by(dest)
```

```
## # A tibble: 336,776 x 19
## # Groups:   dest [105]
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <dbl> <dbl> <dbl>    <dbl>          <dbl>     <dbl>    <dbl>
##  1  2013     1     1      517            515         2      830
##  2  2013     1     1      533            529         4      850
##  3  2013     1     1      542            540         2      923
##  4  2013     1     1      544            545        -1     1004
##  5  2013     1     1      554            600        -6      812
##  6  2013     1     1      554            558        -4      740
##  7  2013     1     1      555            600        -5      913
##  8  2013     1     1      557            600        -3      709
##  9  2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
## # … with 336,766 more rows, and 12 more variables: sched_arr_time <dbl>,
## #   arr_delay <dbl>, carrier <chr>, flight <dbl>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

Agrupamientos

```
flights %>%
  group_by(dest, origin)
```

```
## # A tibble: 336,776 x 19
## # Groups:   dest, origin [224]
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <dbl> <dbl> <dbl>    <dbl>          <dbl>     <dbl>    <dbl>
##  1  2013     1     1      517            515         2      830
##  2  2013     1     1      533            529         4      850
##  3  2013     1     1      542            540         2      923
##  4  2013     1     1      544            545        -1     1004
##  5  2013     1     1      554            600        -6      812
##  6  2013     1     1      554            558        -4      740
##  7  2013     1     1      555            600        -5      913
##  8  2013     1     1      557            600        -3      709
##  9  2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
## # … with 336,766 more rows, and 12 more variables: sched_arr_time <dbl>,
## #   arr_delay <dbl>, carrier <chr>, flight <dbl>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

Funciones de resumen

Nos van a dar descripciones de nuestro conjunto de datos

Funcionan en el conjunto original, filtrado, agrupado, o manipulado

```
flights %>%
summarise(mean_distance = mean(distance))
```

```
## # A tibble: 1 x 1
 ##   mean_distance
 ##           <dbl>
 ## 1         1040.
```

```r
flights %>%
  select(contains("time")) %>%
  summarise(mean_deptime = mean(dep_time),
            mean_sched_deptime = mean(sched_dep_time),
            mean_arr_time = mean(arr_time),
            mean_sched_arr_time = mean(sched_arr_time),
            mean_air_time = mean(air_time),
            mean_time_hour = mean(time_hour)
            )
```

```
## # A tibble: 1 x 6
##   mean_deptime mean_sched_dept… mean_arr_time mean_sched_arr_…
##          <dbl>            <dbl>         <dbl>            <dbl>
## 1           NA            1344.            NA            1536.
## # … with 2 more variables: mean_air_time <dbl>, mean_time_hour <dttm>
```

```r
flights %>%
  select(contains("time")) %>%
  summarise_all(mean, na.rm=TRUE)
```

```
## # A tibble: 1 x 6
##   dep_time sched_dep_time arr_time sched_arr_time air_time
##      <dbl>          <dbl>    <dbl>          <dbl>    <dbl>
## 1    1349.          1344.    1502.          1536.     151.
## # … with 1 more variable: time_hour <dttm>
```

Puedo usarlas sobre datos agrupados

```
flights %>%
  group_by(dest) %>%
  select(contains("time")) %>%
  summarise_all(mean, na.rm=TRUE)
```

```
## Adding missing grouping variables: `dest`
```

```
## # A tibble: 105 x 7
##    dest  dep_time sched_dep_time arr_time sched_arr_time air_time
##    <chr>    <dbl>          <dbl>    <dbl>          <dbl>    <dbl>
##  1 ABQ      2006.          2001.    2049.          2278.    249.
##  2 ACK      1033.          1033.    1145.          1139.     42.1
##  3 ALB      1627.          1610.    1702.          1718.     31.8
##  4 ANC      1635.          1618.    1968           1966.    413.
##  5 ATL      1293.          1287.    1513.          1529.    113.
##  6 AUS      1521.          1506.    1614.          1816.    213.
##  7 AVL      1175.          1169.    1373.          1362.     89.9
##  8 BDL      1490.          1506.    1549.          1610.     25.5
##  9 BGR      1690.          1673.    1715.          1836.     54.1
## 10 BHM      1944.          1909.    2028.          2078.    123.
## # … with 95 more rows, and 1 more variable: time_hour <dttm>
```

Puedo usarlas sobre datos agrupados

```
flights %>%
  group_by(origin, dest) %>%
  select(contains("time")) %>%
  summarise_all(list(med = median,
                     avg = mean
                     ),
                na.rm = TRUE
                )
```

```
## Adding missing grouping variables: `origin`, `dest`
```

```
## # A tibble: 224 x 14
## # Groups:   origin [3]
##    origin dest  dep_time_med sched_dep_time_… arr_time_med sched_arr_time_…
##    <chr>  <chr>        <dbl>            <dbl>        <dbl>            <dbl>
##  1 EWR    ALB           1738             1721         1832             1822
##  2 EWR    ANC           1618             1615         1957             1953
##  3 EWR    ATL           1257             1300         1513             1519
##  4 EWR    AUS           1503             1459         1746             1805
##  5 EWR    AVL           1156             1200         1339             1350
##  6 EWR    BDL           1328.            1327         1414.            1422
##  7 EWR    BNA           1306.            1300         1421             1425
##  8 EWR    BOS           1400             1400         1458             1511
```

Contar: n, count, tally

n

```
flights %>%
  group_by(month, origin, dest) %>%
  summarize(cuantos = n())
```

```
## # A tibble: 2,313 x 4
## # Groups:   month, origin [36]
##    month origin dest  cuantos
##    <dbl> <chr>  <chr>   <int>
##  1     1 EWR    ALB        64
##  2     1 EWR    ATL       362
##  3     1 EWR    AUS        51
##  4     1 EWR    AVL         2
##  5     1 EWR    BDL        37
##  6     1 EWR    BNA       111
##  7     1 EWR    BOS       430
##  8     1 EWR    BQN        31
##  9     1 EWR    BTV       100
## 10     1 EWR    BUF       119
## # … with 2,303 more rows
```

count

```
flights %>%
    count()
```

```
## # A tibble: 1 x 1
##         n
##     <int>
## 1 336776
```

count

```
flights %>%
 count(origin)
```

```
## # A tibble: 3 x 2
 ##   origin      n
 ##   <chr>   <int>
 ## 1 EWR    120835
 ## 2 JFK    111279
 ## 3 LGA    104662
```

count - datos agrupados

```
flights %>%
group_by(month) %>%
count(origin)
```

```
## # A tibble: 36 x 3
## # Groups:   month [12]
##    month origin     n
##    <dbl> <chr>  <int>
##  1     1 EWR     9893
##  2     1 JFK     9161
##  3     1 LGA     7950
##  4     2 EWR     9107
##  5     2 JFK     8421
##  6     2 LGA     7423
##  7     3 EWR    10420
##  8     3 JFK     9697
##  9     3 LGA     8717
## 10     4 EWR    10531
## # … with 26 more rows
```

tally

```
flights %>%
group_by(month, origin, dest) %>%
tally()
```

```
## # A tibble: 2,313 x 4
## # Groups:   month, origin [36]
##    month origin dest      n
##    <dbl> <chr>  <chr> <int>
##  1     1 EWR    ALB      64
##  2     1 EWR    ATL     362
##  3     1 EWR    AUS      51
##  4     1 EWR    AVL       2
##  5     1 EWR    BDL      37
##  6     1 EWR    BNA     111
##  7     1 EWR    BOS     430
##  8     1 EWR    BQN      31
##  9     1 EWR    BTV     100
## 10     1 EWR    BUF     119
## # … with 2,303 more rows
```

Rangos y disperciones

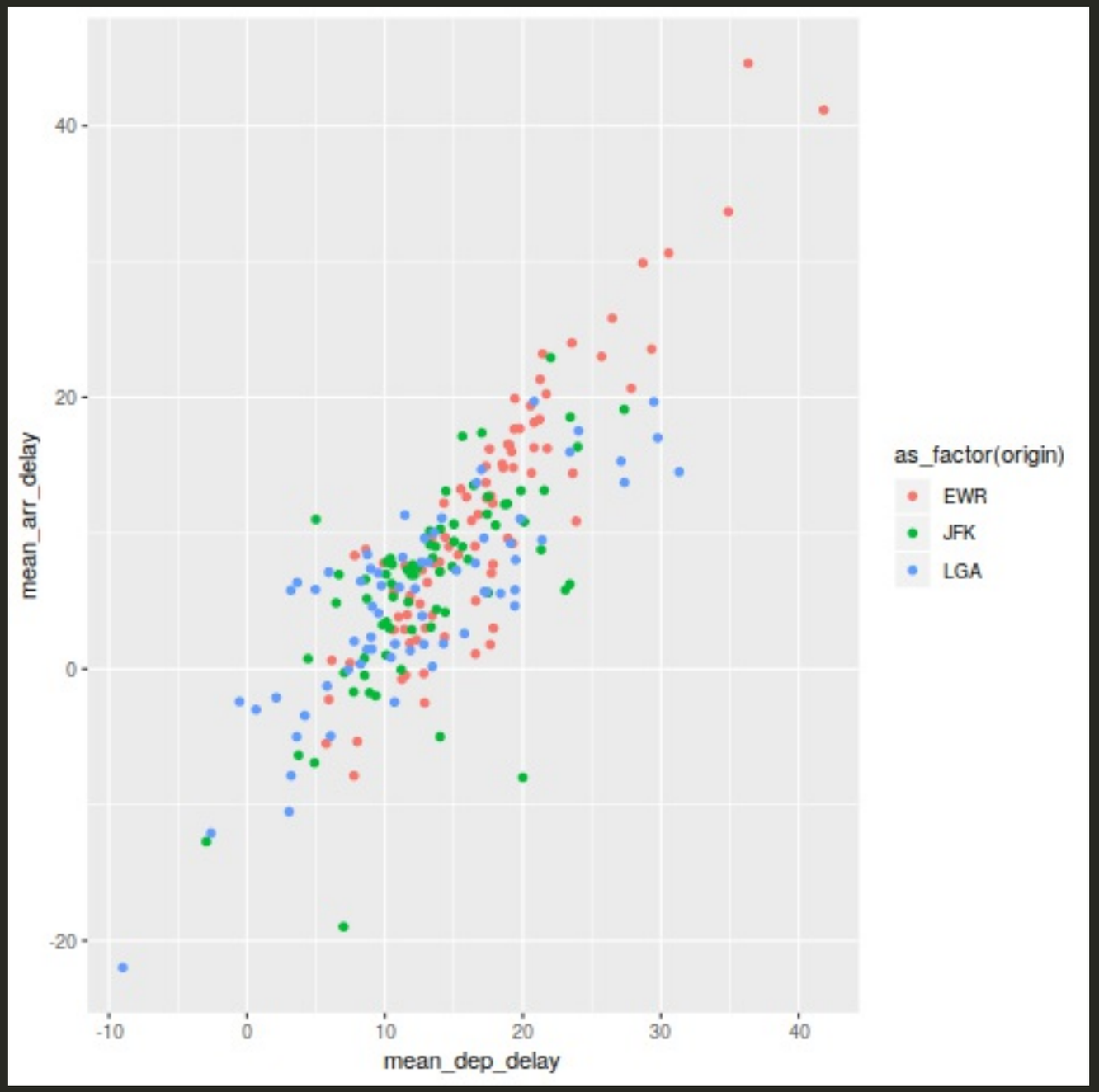
```
flights %>%
  group_by(origin, dest) %>%
summarise(min_airtime = min(air_time, na.rm = TRUE),
          max_airtime = max(air_time, na.rm = TRUE),
      std_dev_airtime = sd(air_time, na.rm = TRUE),
              cuantos = n()
              )
```

```
## # A tibble: 224 x 6
## # Groups:   origin [3]
##    origin dest  min_airtime max_airtime std_dev_airtime cuantos
##    <chr>  <chr>       <dbl>       <dbl>           <dbl>   <int>
##  1 EWR    ALB            24          50            3.08     439
##  2 EWR    ANC           388         434           14.7        8
##  3 EWR    ATL            88         176            9.99    5022
##  4 EWR    AUS           174         301           17.9      968
##  5 EWR    AVL            76         119            7.45     265
##  6 EWR    BDL            20          56            3.29     443
##  7 EWR    BNA            70         176           10.9     2336
##  8 EWR    BOS            30         112            4.92    5327
##  9 EWR    BQN           173         231            8.88     297
## 10 EWR    BTV            34          69            3.67     931
## # … with 214 more rows
```

dplyr + pipe + ggplot2

```
                    flights %>%
                  group_by(origin, dest) %>%
summarise(mean_dep_delay = mean(dep_delay, na.rm = TRUE),
          mean_arr_delay = mean(arr_delay, na.rm = TRUE)
                                ) %>%
        ggplot(mapping = aes(x = mean_dep_delay,
                             y = mean_arr_delay,
                           colour = as_factor(origin)
                                  )
                       ) +
                  geom_point()
```

dplyr + pipe + ggplot2



¡¡¡Nota!!!

dplyr + %>% + ggplot2 != Tidyverse !!!!

¡Hay muchas otros paquetes!

Lo importante es la filosofía: datos limpios, usar verbos...

Ejercicio:

Queremos saber como se comportaron las aerolíneas a lo largo del año 2013, mes a mes, en términos del tiempo promedio de vuelo.

Genere un gráfico de puntos que le permita contestar:

¿Qué aerolínea tiene el tiempo de viaje más largo? ¿Es la misma todo el año?

¿Qué aerolínea tiene el tiempo de viaje más corto? ¿Es la misma todo el año?

Si el gráfico no es suficiente, apóyese en resúmenes de su análisis de datos.

Mutate

Transformaciones de variables

Mutate

Quiero saber el tiempo que estaba esperado para los vuelos

```
flights %>%
select(carrier, flight, air_time, dep_delay, arr_delay)
```

```
## # A tibble: 336,776 x 5
##    carrier flight air_time dep_delay arr_delay
##    <chr>    <dbl>    <dbl>     <dbl>     <dbl>
##  1 UA        1545      227         2        11
##  2 UA        1714      227         4        20
##  3 AA        1141      160         2        33
##  4 B6         725      183        -1       -18
##  5 DL         461      116        -6       -25
##  6 UA        1696      150        -4        12
##  7 B6         507      158        -5        19
##  8 EV        5708       53        -3       -14
##  9 B6          79      140        -3        -8
## 10 AA         301      138        -2         8
## # … with 336,766 more rows
```

```
flights %>%
  select(carrier, flight, air_time, dep_delay, arr_delay) %>%
  mutate(expected_airtime = air_time - dep_delay - arr_delay)
```

```
## # A tibble: 336,776 x 6
##    carrier flight air_time dep_delay arr_delay expected_airtime
##    <chr>    <dbl>    <dbl>     <dbl>     <dbl>            <dbl>
##  1 UA        1545      227         2        11              214
##  2 UA        1714      227         4        20              203
##  3 AA        1141      160         2        33              125
##  4 B6         725      183        -1       -18              202
##  5 DL         461      116        -6       -25              147
##  6 UA        1696      150        -4        12              142
##  7 B6         507      158        -5        19              144
##  8 EV        5708       53        -3       -14               70
##  9 B6          79      140        -3        -8              151
## 10 AA         301      138        -2         8              132
## # … with 336,766 more rows
```

Tambien puedo hacer variables intermedias

```
                    ## # A tibble: 336,776 x 7
##    carrier flight air_time dep_delay arr_delay total_delay expected_airtime
##    <chr>    <dbl>    <dbl>     <dbl>     <dbl>       <dbl>            <dbl>
##  1 UA        1545      227         2        11          13              214
##  2 UA        1714      227         4        20          24              203
##  3 AA        1141      160         2        33          35              125
##  4 B6         725      183        -1       -18         -19              202
##  5 DL         461      116        -6       -25         -31              147
##  6 UA        1696      150        -4        12           8              142
##  7 B6         507      158        -5        19          14              144
##  8 EV        5708       53        -3       -14         -17               70
##  9 B6          79      140        -3        -8         -11              151
## 10 AA         301      138        -2         8           6              132
                  ## # … with 336,766 more rows
```

case_when para condicionales

Quiero una variable que diga si se adelantó o se atrasó

```
my_flights <-
  flights %>%
  mutate(total_delay = dep_delay + arr_delay,
         expected_airtime = air_time - total_delay
         ) %>%
  mutate(net_gainloss = case_when(total_delay > 0 ~ "retraso",
                                  total_delay == 0 ~ "a_tiempo",
                                  total_delay < 0 ~ "adelanto"
                                  )
         )
```

case_when para condicionales

Quiero una variable que diga si se adelantó o se atrasó

```
my_flights
```

```
## # A tibble: 336,776 x 22
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <dbl> <dbl> <dbl>    <dbl>          <dbl>     <dbl>    <dbl>
##  1  2013     1     1      517            515         2      830
##  2  2013     1     1      533            529         4      850
##  3  2013     1     1      542            540         2      923
##  4  2013     1     1      544            545        -1     1004
##  5  2013     1     1      554            600        -6      812
##  6  2013     1     1      554            558        -4      740
##  7  2013     1     1      555            600        -5      913
##  8  2013     1     1      557            600        -3      709
##  9  2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
## # … with 336,766 more rows, and 15 more variables: sched_arr_time <dbl>,
## #   arr_delay <dbl>, carrier <chr>, flight <dbl>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, total_delay <dbl>,
## #   expected_airtime <dbl>, net_gainloss <chr>
```

case_when para condicionales

Quiero una variable que diga si se adelantó o se atrasó

```
my_flights %>%
  select(carrier, flight, air_time, expected_airtime, total_delay, net_gainl
```

```
## # A tibble: 336,776 x 6
##    carrier flight air_time expected_airtime total_delay net_gainloss
##    <chr>    <dbl>    <dbl>            <dbl>       <dbl> <chr>
##  1 UA        1545      227              214          13 retraso
##  2 UA        1714      227              203          24 retraso
##  3 AA        1141      160              125          35 retraso
##  4 B6         725      183              202         -19 adelanto
##  5 DL         461      116              147         -31 adelanto
##  6 UA        1696      150              142           8 retraso
##  7 B6         507      158              144          14 retraso
##  8 EV        5708       53               70         -17 adelanto
##  9 B6          79      140              151         -11 adelanto
## 10 AA         301      138              132           6 retraso
## # … with 336,766 more rows
```

Ejercicio

calcule la velocidad promedio de cada aerolínea.

Los terribles joins

Los ~~terribles joins~~

El Hombre Araña

- Tuvo dos series de películas ANTES del Marvel Cinematic Universe.

- Cada una tuvo diferentes actores.

```
spiderman_raimi <- vroom::vroom(file = "data/spiderman_raimi.csv")
```

```
## Observations: 11
## Variables: 2
## chr [2]: character, cast
##
## Call `spec()` for a copy-pastable column specification
## Specify the column types with `col_types` to quiet this message
```

```
spiderman_amazing <- vroom::vroom(file = "data/amazing_spiderman.csv")
```

```
## Observations: 8
## Variables: 2
## chr [2]: character, cast
##
## Call `spec()` for a copy-pastable column specification
## Specify the column types with `col_types` to quiet this message
```

```
spiderman_raimi
```

```
## # A tibble: 11 x 2
##    character        cast
##    <chr>            <chr>
##  1 Spider-man       Tobey Maguire
##  2 Mary Jane Watson Kirsten Dunst
##  3 Harry Osborn     James Franco
##  4 Tía May          Rosemary Harris
##  5 J. Jonah Jameson J.K. Simmons
##  6 Norman Osborn    Willem Dafoe
##  7 Doctor Octopus   Alfred Molina
##  8 Venom            Topher Grace
##  9 Sandman          Thomas Haden Church
## 10 Gwen Stacy       Bryce Dallas Howard
## 11 Dr. Curt Connors Dylan Baker
```

```
spiderman_amazing
```

```
## # A tibble: 8 x 2
##   character        cast
##   <chr>            <chr>
## 1 Spider-man       Andrew Garfield
## 2 Harry Osborn     Dane DeHaan
## 3 Tía May          Sally Field
## 4 Norman Osborn    Chris Cooper
## 5 Electro          Jamie Foxx
## 6 Gwen Stacy       Emma Stone
## 7 Rhino            Paul Giamatti
## 8 Dr. Curt Connors Rhys Ifans
```

¿Qué actores interpretan a cada personaje?

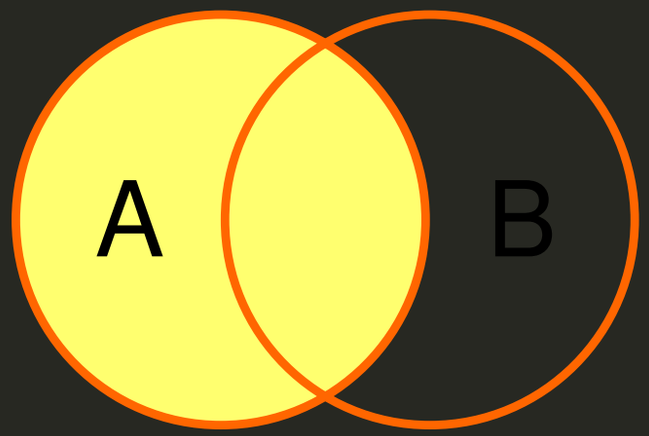
¿Qué personajes aparecen en la primera serie?

¿Qué personajes aparecen en la segunda serie?

¿Qué personajes aparecen en ambas?

El problema clásico de juntar diccionarios

Left join

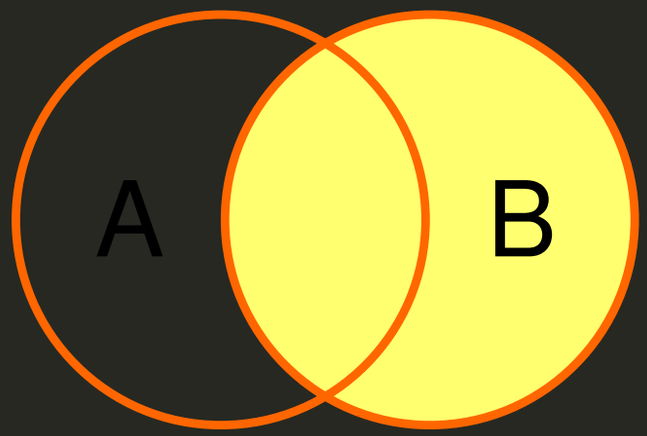


Left join

```
dplyr::left_join(x = spiderman_raimi,
                 y = spiderman_amazing,
                 suffix = c("_raimi", "_amazing"),
                 by = c("character" = "character")
                 )
```

```
## # A tibble: 11 x 3
##    character        cast_raimi          cast_amazing
##    <chr>            <chr>               <chr>
##  1 Spider-man       Tobey Maguire       Andrew Garfield
##  2 Mary Jane Watson Kirsten Dunst       <NA>
##  3 Harry Osborn     James Franco        Dane DeHaan
##  4 Tía May          Rosemary Harris     Sally Field
##  5 J. Jonah Jameson J.K. Simmons        <NA>
##  6 Norman Osborn    Willem Dafoe        Chris Cooper
##  7 Doctor Octopus   Alfred Molina       <NA>
##  8 Venom            Topher Grace        <NA>
##  9 Sandman          Thomas Haden Church <NA>
## 10 Gwen Stacy       Bryce Dallas Howard Emma Stone
## 11 Dr. Curt Connors Dylan Baker         Rhys Ifans
```

Right join

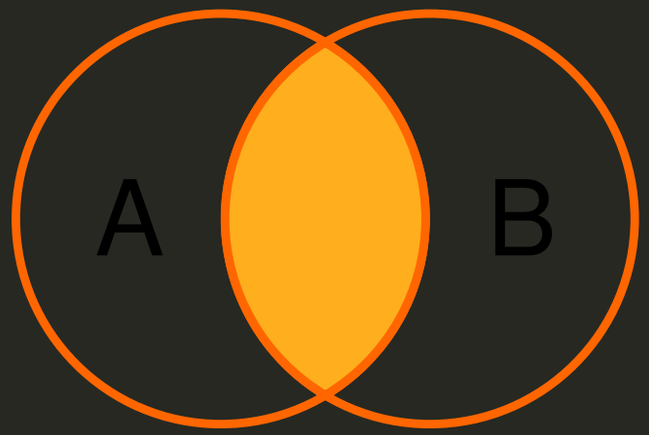


Right join

```
dplyr::right_join(x = spiderman_raimi,
                  y = spiderman_amazing,
                suffix = c("_raimi", "_amazing"),
                by = c("character" = "character")
                                  )
```

```
## # A tibble: 8 x 3
##   character      cast_raimi          cast_amazing
##   <chr>          <chr>               <chr>
## 1 Spider-man     Tobey Maguire       Andrew Garfield
## 2 Harry Osborn   James Franco        Dane DeHaan
## 3 Tía May        Rosemary Harris     Sally Field
## 4 Norman Osborn  Willem Dafoe        Chris Cooper
## 5 Electro        <NA>                Jamie Foxx
## 6 Gwen Stacy     Bryce Dallas Howard Emma Stone
## 7 Rhino          <NA>                Paul Giamatti
## 8 Dr. Curt Connors Dylan Baker         Rhys Ifans
```

semi join

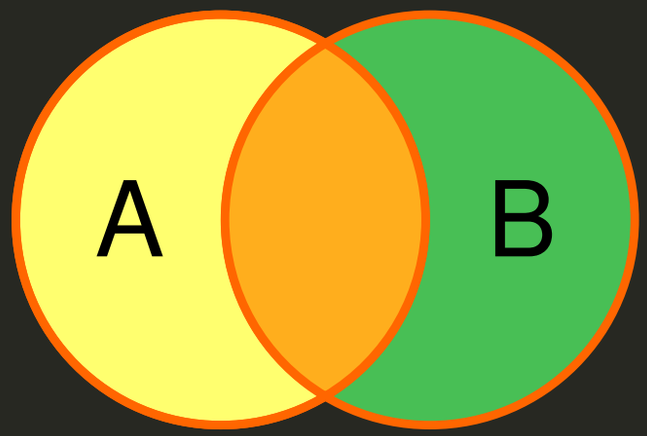


semi join

```
dplyr::semi_join(x = spiderman_raimi,
                 y = spiderman_amazing,
                 suffix = c("_raimi", "_amazing"),
                 by = c("character" = "character")
                                    )
```

```
## # A tibble: 6 x 2
##   character        cast
##   <chr>            <chr>
## 1 Spider-man       Tobey Maguire
## 2 Harry Osborn     James Franco
## 3 Tía May          Rosemary Harris
## 4 Norman Osborn    Willem Dafoe
## 5 Gwen Stacy       Bryce Dallas Howard
## 6 Dr. Curt Connors Dylan Baker
```

full join

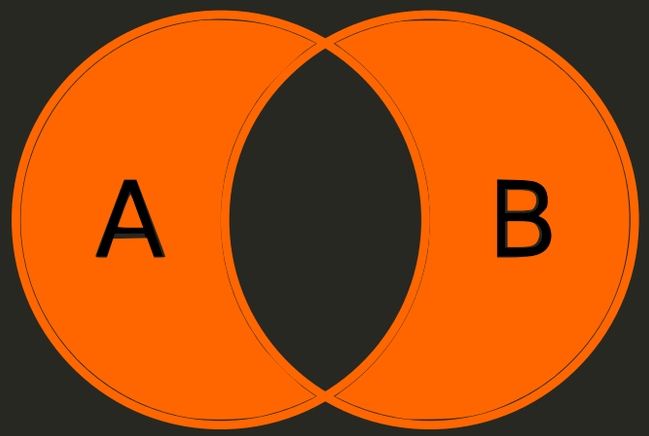


full join

```
dplyr::full_join(x = spiderman_raimi,
                 y = spiderman_amazing,
                 suffix = c("_raimi", "_amazing"),
                 by = c("character" = "character")
                 )
```

```
## # A tibble: 13 x 3
##    character        cast_raimi          cast_amazing
##    <chr>            <chr>               <chr>
##  1 Spider-man       Tobey Maguire       Andrew Garfield
##  2 Mary Jane Watson Kirsten Dunst       <NA>
##  3 Harry Osborn     James Franco        Dane DeHaan
##  4 Tía May          Rosemary Harris     Sally Field
##  5 J. Jonah Jameson J.K. Simmons        <NA>
##  6 Norman Osborn    Willem Dafoe        Chris Cooper
##  7 Doctor Octopus   Alfred Molina       <NA>
##  8 Venom            Topher Grace        <NA>
##  9 Sandman          Thomas Haden Church <NA>
## 10 Gwen Stacy       Bryce Dallas Howard Emma Stone
## 11 Dr. Curt Connors Dylan Baker         Rhys Ifans
## 12 Electro          <NA>                Jamie Foxx
 ## 13 Rhino           <NA>                Paul Giamatti
```

anti join



anti join

```
dplyr::anti_join(x = spiderman_raimi,
                 y = spiderman_amazing,
                 suffix = c("_raimi", "_amazing"),
                 by = c("character" = "character")
                                    )
```

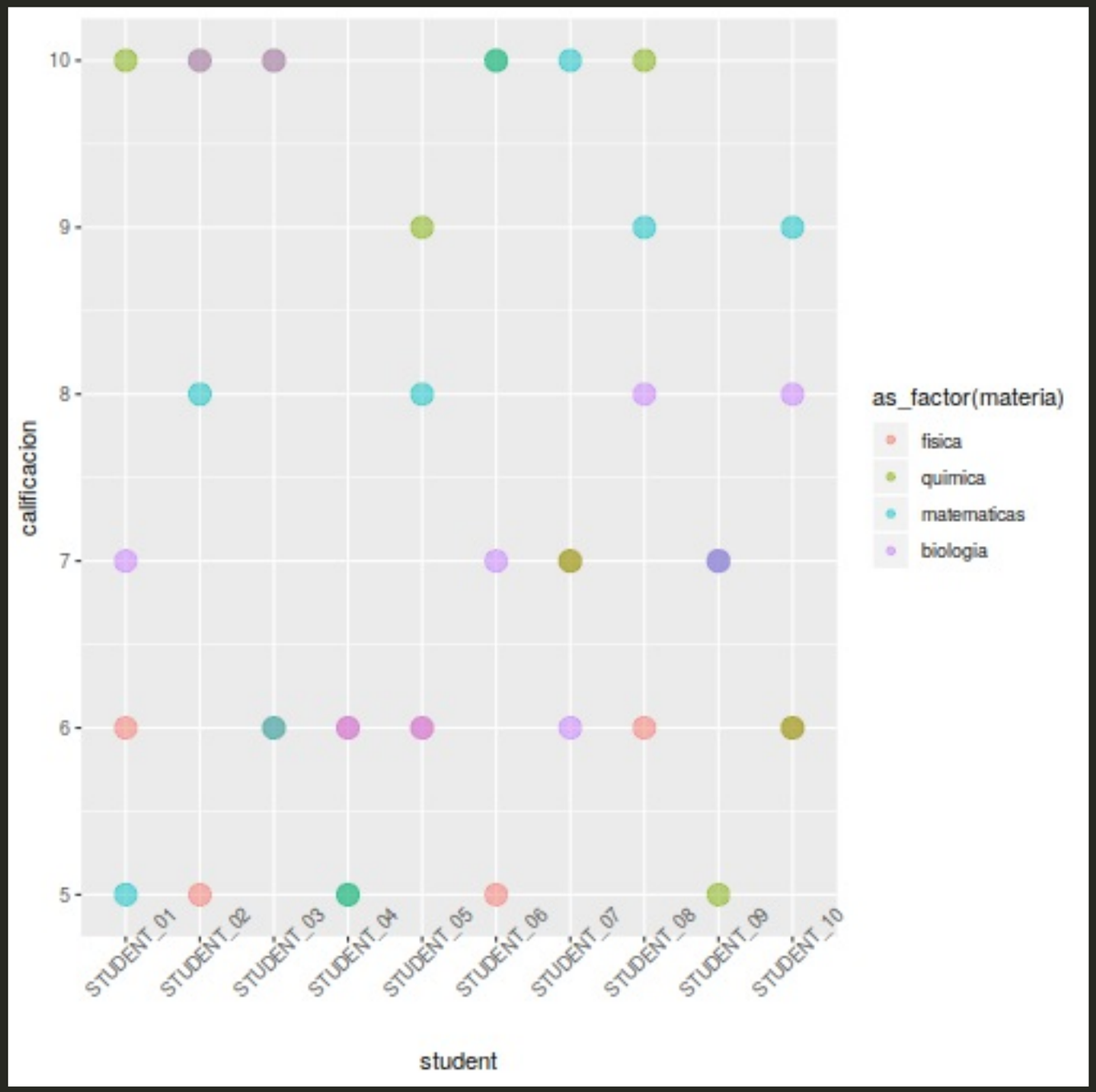
```
## # A tibble: 5 x 2
##   character        cast
##   <chr>            <chr>
## 1 Mary Jane Watson Kirsten Dunst
## 2 J. Jonah Jameson J.K. Simmons
## 3 Doctor Octopus   Alfred Molina
## 4 Venom            Topher Grace
## 5 Sandman          Thomas Haden Church
```

Ejercicio:

Haga el join adecuado para los datos en data/flights.txt y data/airlines.txt

Más tidying

Veamos un plot de calificaciones



Es un plot feo... no es la mejor forma de presentar la información

... pero digamos que queremos reproducirlo

--

La próxima semana veremos como hacerlo mejor

Leamos los datos

```
my_data <- vroom::vroom("data/calificaciones_untidy.txt")
my_data
```

```
## # A tibble: 10 x 5
##    student    fisica quimica matematicas biologia
##    <chr>       <dbl>   <dbl>       <dbl>    <dbl>
##  1 STUDENT_01      6      10           5        7
##  2 STUDENT_02      5      10           8       10
##  3 STUDENT_03      6      10           6       10
##  4 STUDENT_04      6       5           5        6
##  5 STUDENT_05      6       9           8        6
##  6 STUDENT_06      5      10          10        7
##  7 STUDENT_07      7       7          10        6
##  8 STUDENT_08      6      10           9        8
##  9 STUDENT_09      7       5           7        7
## 10 STUDENT_10      6       6           9        8
```

Hagamos el plot...

```
                    my_data %>%
                     ggplot() %>%
geom_point(mapping = aes(x = estudiante?????,
                         y = calificaciones?????
                        donde pongo las materias???
                              )
                    )
```

--

Los datos no están organizados para esta aplicación

Tenemos que cambiar la *forma* de los datos = > *pivotear*

tidyr::gather

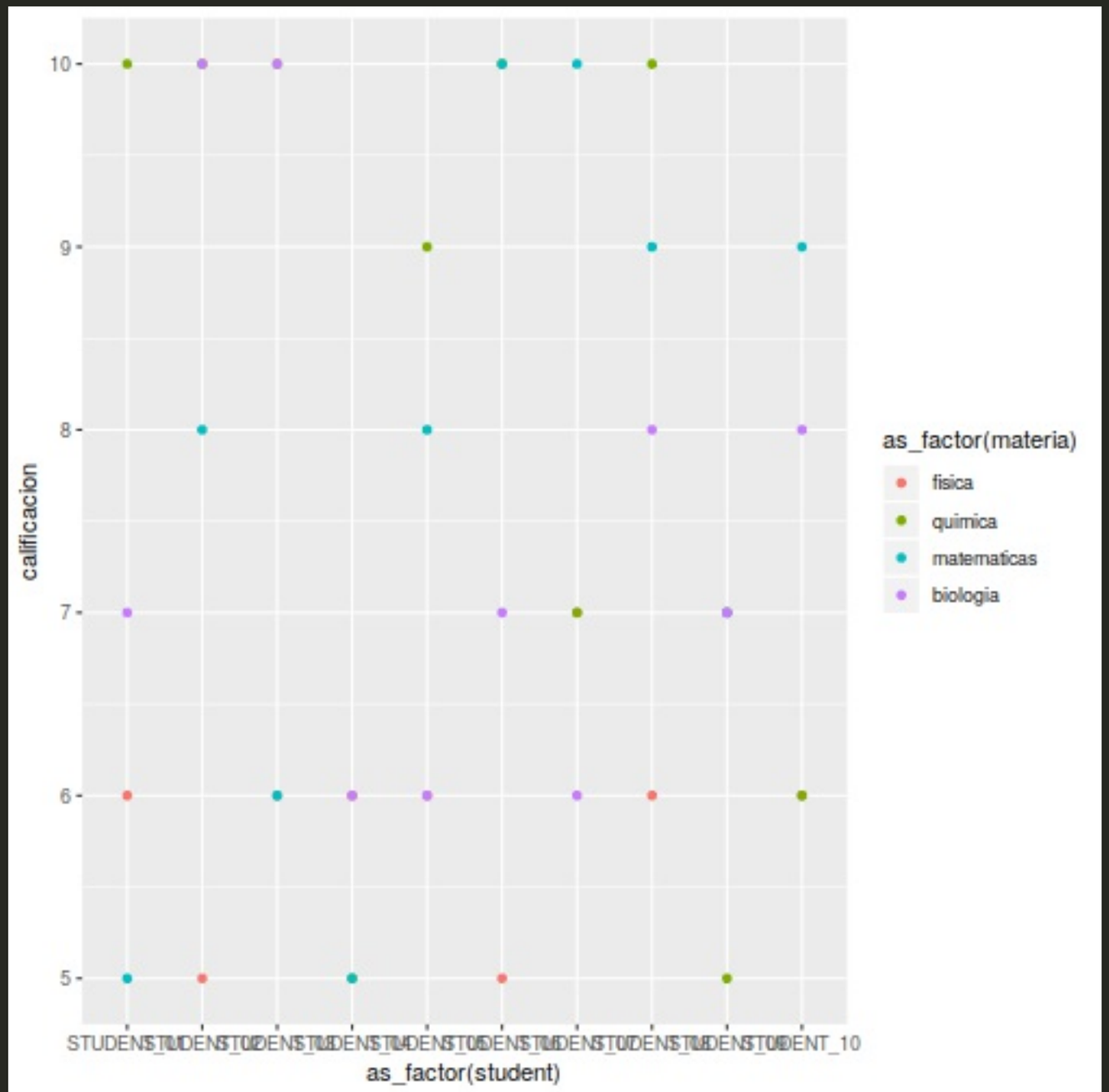
```
my_data_gather <-
   my_data %>%
gather(-"student", key = "materia", value = "calificacion")

my_data_gather
```

```
## # A tibble: 40 x 3
##    student    materia calificacion
##    <chr>      <chr>          <dbl>
##  1 STUDENT_01 fisica             6
##  2 STUDENT_02 fisica             5
##  3 STUDENT_03 fisica             6
##  4 STUDENT_04 fisica             6
##  5 STUDENT_05 fisica             6
##  6 STUDENT_06 fisica             5
##  7 STUDENT_07 fisica             7
##  8 STUDENT_08 fisica             6
##  9 STUDENT_09 fisica             7
## 10 STUDENT_10 fisica             6
      ## # … with 30 more rows
```

tidyr::gather

```
my_data_gather %>%
ggplot(mapping = aes(x = as_factor(student),
                     y = calificacion,
                  colour = as_factor(materia),
                         )
                  ) +
                geom_point()
```

tidyr::gather



tidyr::gather no es la única manera de pivotear los datos

reshape2::melt

tidyr::pivot_long en tidyr 1.0

reshape2::melt

```
my_data_melt <-
  my_data %>%
reshape2::melt(variable.name = "materia",
               value.name = "calificacion",
                   id.vars = "student")

head(my_data_melt)
```

```
##      student materia calificacion
## 1 STUDENT_01  fisica            6
## 2 STUDENT_02  fisica            5
## 3 STUDENT_03  fisica            6
## 4 STUDENT_04  fisica            6
## 5 STUDENT_05  fisica            6
## 6 STUDENT_06  fisica            5
```

tidyr::pivot_long (SOLO DE LA VERSIÓN 1.0 DE TIDYR EN ADELANTE)

```
my_data_pivot_longer <-
        my_data %>%
tidyr::pivot_longer(cols = -"student",
                    names_to = "materia",
                   values_to = "calificacion")

my_data_pivot_longer
```

```
## # A tibble: 40 x 3
##    student    materia     calificacion
##    <chr>      <chr>              <dbl>
##  1 STUDENT_01 fisica                 6
##  2 STUDENT_01 quimica               10
##  3 STUDENT_01 matematicas            5
##  4 STUDENT_01 biologia               7
##  5 STUDENT_02 fisica                 5
##  6 STUDENT_02 quimica               10
##  7 STUDENT_02 matematicas            8
##  8 STUDENT_02 biologia              10
##  9 STUDENT_03 fisica                 6
## 10 STUDENT_03 quimica               10
       ## # … with 30 more rows
```

Tareas...