

# Dplyr

Alexandre

04/12/2020

Téléchargement et installation de Tidyverse (comprend ggplot2 et dplyr entre autre)

`install.packages("tidyverse")` #téléchargement librairie Tidyverse

```
library(tidyverse) #chargement librairie tidyverse contenant le package dplyr
```

```
## -- Attaching packages ----- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2      v purrr   0.3.4
## v tibble  3.0.4      v dplyr   1.0.2
## v tidyr   1.1.2      v stringr 1.4.0
## v readr   1.4.0      v forcats 0.5.0
```

```
## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()     masks stats::lag()
```

*#Ou alors*

```
library(dplyr) #chargement package dplyr si on veut uniquement le package dplyr
```

Téléchargement et installation d'une base de donnée à étudier

#téléchargement base de données vols New York 2013 `install.packages("nycflights13")`

```
#chargement de la base et 3 tables: flights, airports et airlines
library(nycflights13)
data(flights)
data(airports)
data(airlines)
```

Fonctions Slice

```
#utilisation slice, sélectionner une ligne d'une table
slice(airports, 345)
```

```
## # A tibble: 1 x 8
##   faa   name                lat   lon   alt   tz dst  tzone
##   <chr> <chr>                <dbl> <dbl> <dbl> <dbl> <chr> <chr>
## 1 CYF   Chefnak Airport    60.1 -164.   40    -9 A   America/Anchorage
```

```
slice(airlines, 10)
```

```
## # A tibble: 1 x 2
##   carrier name
##   <chr>   <chr>
## 1 MQ      Envoy Air
```

```
slice(flights, 666)
```

```
## # A tibble: 1 x 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    1832           1835        -3    2059           2103
## # ... 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 <dtm>
```

*#sélectionner plusieurs lignes avec slice*

*slice(airports, 0:5) #débuter à 0 ou 1 donne même résultat*

```
## # A tibble: 5 x 8
##   faa   name                lat   lon   alt   tz dst  tzone
##   <chr> <chr>                <dbl> <dbl> <dbl> <dbl> <chr> <chr>
## 1 04G   Lansdowne Airport      41.1 -80.6 1044   -5 A   America/New_Y~
## 2 06A   Moton Field Municipal Airp~ 32.5 -85.7  264   -6 A   America/Chica~
## 3 06C   Schaumburg Regional     42.0 -88.1  801   -6 A   America/Chica~
## 4 06N   Randall Airport        41.4 -74.4  523   -5 A   America/New_Y~
## 5 09J   Jekyll Island Airport   31.1 -81.4   11   -5 A   America/New_Y~
```

```
slice(flights,666:777)
```

```
## # A tibble: 112 x 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    1832           1835        -3    2059           2103
## 2  2013     1     1    1832           1828         4    2144           2144
## 3  2013     1     1    1834           1840        -6    2027           2020
## 4  2013     1     1    1834           1800        34    2014           1942
## 5  2013     1     1    1836           1726        70    2107           1933
## 6  2013     1     1    1840           1836         4    2022           2010
## 7  2013     1     1    1840           1845        -5    2055           2030
## 8  2013     1     1    1840           1845        -5    2223           2226
## 9  2013     1     1    1842           1422       260    1958           1535
## 10 2013     1     1    1842           1829        13    2144           2136
## # ... with 102 more rows, and 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 <dtm>
```

*#dbl = nombre à virgule ?*

*#slice\_head et slice\_tail permettent de sélectionner les n premières ou n dernières lignes du tableau*

*airports%>%slice\_head(n=3)#on sélectionne les 3 premières lignes du tableau*

```
## # A tibble: 3 x 8
##   faa   name                lat   lon   alt   tz dst   tzone
##   <chr> <chr>                <dbl> <dbl> <dbl> <dbl> <chr> <chr>
## 1 04G   Lansdowne Airport        41.1 -80.6  1044   -5 A   America/New_Y~
## 2 06A   Moton Field Municipal Airp~ 32.5 -85.7   264   -6 A   America/Chica~
## 3 06C   Schaumburg Regional        42.0 -88.1   801   -6 A   America/Chica~
```

```
airports%>%slice_head(n=0.2) #on sélectionne les premières lignes constituant 20% du tableau total
```

```
## # A tibble: 0 x 8
## # ... with 8 variables: faa <chr>, name <chr>, lat <dbl>, lon <dbl>, alt <dbl>,
## #   tz <dbl>, dst <chr>, tzone <chr>
```

*#slice\_max et slice\_min affichent n ou prop lignes du tableau ayant les valeurs les plus ou les moins élevées de cette variable*

```
airports %>% slice_min(alt, n = 2) #Si on veut les 2 aéroports avec l'altitude la plus basse
```

```
## # A tibble: 2 x 8
##   faa   name                lat   lon   alt   tz dst   tzone
##   <chr> <chr>                <dbl> <dbl> <dbl> <dbl> <chr> <chr>
## 1 IPL   Imperial Co        32.8 -116.   -54   -8 A   America/Los_Angeles
## 2 NJK   El Centro Naf    32.8 -116.   -42   -8 A   America/Los_Angeles
```

## Fonction FILTER

*#filter sélectionne des lignes d'une table selon une condition, similaire à "WHERE" en SQL*

```
filter(flights, month==1) #mois de janvier = month 1
```

```
## # A tibble: 27,004 x 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
## 5  2013     1     1     554             600          -6     812             837
## 6  2013     1     1     554             558          -4     740             728
## 7  2013     1     1     555             600          -5     913             854
## 8  2013     1     1     557             600          -3     709             723
## 9  2013     1     1     557             600          -3     838             846
## 10 2013     1     1     558             600          -2     753             745
## # ... with 26,994 more rows, and 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 <dtm>
```

```
filter(airports, alt>8000) #aéroport à plus de 8000 d'altitude, 8000 pieds ?
```

```
## # A tibble: 2 x 8
##   faa   name          lat lon alt   tz dst tzone
##   <chr> <chr>        <dbl> <dbl> <dbl> <dbl> <chr> <chr>
## 1 TEX   Telluride      38.0 -108. 9078   -7 A   America/Denver
## 2 TVL   Lake Tahoe Airport 38.9 -120. 8544   -8 A   America/Los_Angeles
```

```
filter(airports, faa=="LCY") #on cherche l'aéroport de Londres, introuvable dans la base
```

```
## # A tibble: 0 x 8
## # ... with 8 variables: faa <chr>, name <chr>, lat <dbl>, lon <dbl>, alt <dbl>,
## #   tz <dbl>, dst <chr>, tzone <chr>
```

```
filter(airports, faa=="LAX") #on cherche l'aéroport de Los Angeles, présent dans la base
```

```
## # A tibble: 1 x 8
##   faa   name          lat lon alt   tz dst tzone
##   <chr> <chr>        <dbl> <dbl> <dbl> <dbl> <chr> <chr>
## 1 LAX   Los Angeles Intl 33.9 -118. 126   -8 A   America/Los_Angeles
```

## Fonction SELECT

*#fonction select fonctionnement similaire à SELECT en SQL, affiche les colonnes indiquées sans filtre*

```
select(airports, lat, lon, name, tzone)
```

```
## # A tibble: 1,458 x 4
##   lat lon name          tzone
##   <dbl> <dbl> <chr>        <chr>
## 1 41.1 -80.6 Lansdowne Airport America/New_York
## 2 32.5 -85.7 Moton Field Municipal Airport America/Chicago
## 3 42.0 -88.1 Schaumburg Regional America/Chicago
## 4 41.4 -74.4 Randall Airport America/New_York
## 5 31.1 -81.4 Jekyll Island Airport America/New_York
## 6 36.4 -82.2 Elizabethton Municipal Airport America/New_York
## 7 41.5 -84.5 Williams County Airport America/New_York
## 8 42.9 -76.8 Finger Lakes Regional Airport America/New_York
## 9 39.8 -76.6 Shoestring Aviation Airfield America/New_York
## 10 48.1 -123. Jefferson County Intl America/Los_Angeles
## # ... with 1,448 more rows
```

```
select(airports, lat:tzone) #sélection de toutes les colonnes dans cet interval
```

```
## # A tibble: 1,458 x 6
##   lat lon alt tz dst tzone
##   <dbl> <dbl> <dbl> <dbl> <chr> <chr>
## 1 41.1 -80.6 1044 -5 A America/New_York
## 2 32.5 -85.7 264 -6 A America/Chicago
## 3 42.0 -88.1 801 -6 A America/Chicago
## 4 41.4 -74.4 523 -5 A America/New_York
## 5 31.1 -81.4 11 -5 A America/New_York
## 6 36.4 -82.2 1593 -5 A America/New_York
## 7 41.5 -84.5 730 -5 A America/New_York
## 8 42.9 -76.8 492 -5 A America/New_York
## 9 39.8 -76.6 1000 -5 U America/New_York
## 10 48.1 -123. 108 -8 A America/Los_Angeles
## # ... with 1,448 more rows
```

*# en ajoutant - avant la colonne, on exclue celle-ci et affiche les autres*

```
select(airports, -lat, -lon)
```

```
## # A tibble: 1,458 x 6
##   faa name alt tz dst tzone
##   <chr> <chr> <dbl> <dbl> <chr> <chr>
## 1 04G Lansdowne Airport 1044 -5 A America/New_York
## 2 06A Moton Field Municipal Airport 264 -6 A America/Chicago
## 3 06C Schaumburg Regional 801 -6 A America/Chicago
## 4 06N Randall Airport 523 -5 A America/New_York
## 5 09J Jekyll Island Airport 11 -5 A America/New_York
## 6 0A9 Elizabethton Municipal Airport 1593 -5 A America/New_York
## 7 0G6 Williams County Airport 730 -5 A America/New_York
## 8 0G7 Finger Lakes Regional Airport 492 -5 A America/New_York
## 9 0P2 Shoestring Aviation Airfield 1000 -5 U America/New_York
## 10 0S9 Jefferson County Intl 108 -8 A America/Los_Angeles
## # ... with 1,448 more rows
```

*#fonctions starts\_with , ends\_with, contains ou matches; différence entre contains et matches ?*

```
select(airports, matches("name"))
```

```
## # A tibble: 1,458 x 1
##   name
##   <chr>
## 1 Lansdowne Airport
## 2 Moton Field Municipal Airport
## 3 Schaumburg Regional
## 4 Randall Airport
## 5 Jekyll Island Airport
## 6 Elizabethton Municipal Airport
## 7 Williams County Airport
## 8 Finger Lakes Regional Airport
## 9 Shoestring Aviation Airfield
## 10 Jefferson County Intl
## # ... with 1,448 more rows
```

```
select(flights, starts_with("dep"))
```

```
## # A tibble: 336,776 x 2
##   dep_time dep_delay
##   <int>     <dbl>
## 1      517         2
## 2      533         4
## 3      542         2
## 4      544        -1
## 5      554        -6
## 6      554        -4
## 7      555        -5
## 8      557        -3
## 9      557        -3
## 10     558        -2
## # ... with 336,766 more rows
```

## Fonctions RENAME et ARRANGE

*#rename fonctionne comme select mais permet de renommer les colonnes simultanément*  
 rename(airports, longitude=lon, latitude=lat, "test plusieurs mots"=alt)

```
## # A tibble: 1,458 x 8
##   faa   name      latitude longitude `test plusieurs m~   tz dst  tzone
##   <chr> <chr>      <dbl>     <dbl>         <dbl> <dbl> <chr> <chr>
## 1 04G  Lansdowne A~   41.1     -80.6         1044   -5 A   America~
## 2 06A  Moton Field~   32.5     -85.7         264   -6 A   America~
## 3 06C  Schaumburg ~   42.0     -88.1         801   -6 A   America~
## 4 06N  Randall Air~   41.4     -74.4         523   -5 A   America~
## 5 09J  Jekyll Isla~   31.1     -81.4         11    -5 A   America~
## 6 0A9  Elizabethto~   36.4     -82.2        1593   -5 A   America~
## 7 0G6  Williams Co~   41.5     -84.5         730   -5 A   America~
## 8 0G7  Finger Lake~   42.9     -76.8         492   -5 A   America~
## 9 0P2  Shoestring ~   39.8     -76.6        1000   -5 U   America~
## 10 0S9  Jefferson C~   48.1    -123.         108   -8 A   America~
## # ... with 1,448 more rows
```

*#arrange classe les lignes selon une colonne, similaire à ORDER BY en SQL*  
 arrange(flights, dep\_delay) *#on classe les lignes selon le retard dans l'ordre croissant*

```
## # A tibble: 336,776 x 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
## 7  2013    10    23    1907           1932        -25    2143           2143
## 8  2013     3    30    2030           2055        -25    2213           2250
## 9  2013     3     2    1431           1455        -24    1601           1631
## 10 2013     5     5     934            958        -24    1225           1309
## # ... with 336,766 more rows, and 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 <dtm>
```

```
arrange(flights, month, dep_delay) #on classe les lignes selon le mois puis selon le retard
```

```
## # A tibble: 336,776 x 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    11    1900           1930         -30     2233           2243
## 2  2013     1    29    1703           1730         -27     1947           1957
## 3  2013     1    12    1354           1416         -22     1606           1650
## 4  2013     1    21    2137           2159         -22     2232           2316
## 5  2013     1    20     704            725         -21     1025           1035
## 6  2013     1    12    2050           2110         -20     2310           2355
## 7  2013     1    12    2134           2154         -20         4             50
## 8  2013     1    14    2050           2110         -20     2329           2355
## 9  2013     1     4    2140           2159         -19     2241           2316
## 10 2013     1    11    1947           2005         -18     2209           2230
## # ... with 336,766 more rows, and 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 <dtm>
```

```
arrange(flights, desc(dep_delay)) #on classe selon le retard dans l'ordre décroissant
```

```
## # A tibble: 336,776 x 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     9     641            900        1301     1242           1530
## 2  2013     6    15    1432           1935        1137     1607           2120
## 3  2013     1    10    1121           1635        1126     1239           1810
## 4  2013     9    20    1139           1845        1014     1457           2210
## 5  2013     7    22     845           1600        1005     1044           1815
## 6  2013     4    10    1100           1900         960     1342           2211
## 7  2013     3    17    2321            810         911      135           1020
## 8  2013     6    27     959           1900         899     1236           2226
## 9  2013     7    22    2257            759         898      121           1026
## 10 2013    12     5     756           1700         896     1058           2020
## # ... with 336,766 more rows, and 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 <dtm>
```

*#on sélectionne le top 3 des retards parmi les vols*

```
x <- arrange(flights, desc(dep_delay))
slice(x, 1:3)
```

```
## # A tibble: 3 x 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     9     641            900        1301     1242           1530
## 2  2013     6    15    1432           1935        1137     1607           2120
## 3  2013     1    10    1121           1635        1126     1239           1810
## # ... 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 <dtm>
```

## Fonction MUTATE

```
#mutate permet de créer de nouvelles colonnes
flights <- mutate(flights, duree_heure = air_time/60) #on créé une nouvelle contenant le temps de vol en heure à partir de la valeur air_time en minute
select(flights, air_time, duree_heure)
```

```
## # A tibble: 336,776 x 2
##   air_time duree_heure
##   <dbl>     <dbl>
## 1      227         3.78
## 2      227         3.78
## 3      160         2.67
## 4      183         3.05
## 5      116         1.93
## 6      150         2.5
## 7      158         2.63
## 8       53         0.883
## 9      140         2.33
## 10     138         2.3
## # ... with 336,766 more rows
```

```
#création de plusieurs colonnes simultanément, même à partir de colonnes nouvellement créées
flights <- mutate(flights,
  duree_heure = air_time / 60,
  distance_km = distance / 0.62137,
  vitesse = distance_km / duree_heure)
select(flights, air_time, duree_heure, distance, distance_km, vitesse)
```

```
## # A tibble: 336,776 x 5
##   air_time duree_heure distance distance_km vitesse
##   <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
## 1      227         3.78      1400      2253.     596.
## 2      227         3.78      1416      2279.     602.
## 3      160         2.67      1089      1753.     657.
## 4      183         3.05      1576      2536.     832.
## 5      116         1.93       762      1226.     634.
## 6      150         2.5       719      1157.     463.
## 7      158         2.63     1065      1714.     651.
## 8       53         0.883      229       369.     417.
## 9      140         2.33      944      1519.     651.
## 10     138         2.3       733      1180.     513.
## # ... with 336,766 more rows
```

## Enchaîner les actions

```
#utilisation d'un objet temporaire pour enchaîner des actions, on stock les résultats intermédiaires dans un objet temporaire au lieu d'imbriquer les formules (possible mais moins pratique)
x1 <- filter(flights, dest == "LAX") #on stock les lignes filtrées dans l'objet x1
x2 <- select(x1, dep_delay, arr_delay) #on stock les colonnes sélectionnées parmi x1 dans l'objet x2
arrange(x2, dep_delay) #on classe les lignes contenues dans l'objet x2 selon le retard des départs
```



```
## # A tibble: 16,174 x 2
##   dep_delay arr_delay
##   <dbl>      <dbl>
## 1      -16       -31
## 2      -15        4
## 3      -15       -4
## 4      -15      -22
## 5      -15      -29
## 6      -15      -20
## 7      -15      -25
## 8      -15      -35
## 9      -15      -17
## 10     -15       -7
## # ... with 16,164 more rows
```

*#plus simple, on utilise un pipe %>%, si j'exécute expr %>% f, alors le résultat de l'expression expr, à gauche du pipe, sera passé comme premier argument à la fonction f, à droite du pipe, ce qui revient à exécuter f(expr)*

*flights %>% filter(dest == "LAX") %>% select(dep\_delay, arr\_delay) %>% arrange(dep\_delay) #même opération que la précédente sans utilisation d'objets intermédiaires*

```
## # A tibble: 16,174 x 2
##   dep_delay arr_delay
##   <dbl>      <dbl>
## 1      -16       -31
## 2      -15        4
## 3      -15       -4
## 4      -15      -22
## 5      -15      -29
## 6      -15      -20
## 7      -15      -25
## 8      -15      -35
## 9      -15      -17
## 10     -15       -7
## # ... with 16,164 more rows
```

*#écriture avec retour à la ligne possible à condition que %>% soit en fin de ligne*

```
flights %>%
  filter(dest == "LAX") %>%
  select(dep_delay, arr_delay)%>%
  arrange(dep_delay)
```

```
## # A tibble: 16,174 x 2
##   dep_delay arr_delay
##   <dbl>     <dbl>
## 1      -16       -31
## 2      -15        4
## 3      -15       -4
## 4      -15      -22
## 5      -15      -29
## 6      -15      -20
## 7      -15      -25
## 8      -15      -35
## 9      -15      -17
## 10     -15       -7
## # ... with 16,164 more rows
```

```
#on peut stocker le tableau résultant du pipeline dans un objet
dep_delay_LA <- flights %>%
  filter(dest == "LAX") %>%
  select(dep_delay, arr_delay)%>%
  arrange(dep_delay)
```

## Fonction GROUP BY

*#fonction group\_by permet de créer des groupes à partir de valeurs d'une ou plusieurs colonnes en leur assignant un "ID" de groupe commun*

```
flights %>% group_by(month) %>% slice(1) #on récupère le premier vol de chaque mois, chaque mois est un groupe, 12 groupes
```

```
## # A tibble: 12 x 22
## # Groups:   month [12]
##   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     2     1     456           500          -4     652           648
## 3  2013     3     1        4          2159         125     318            56
## 4  2013     4     1     454           500          -6     636           640
## 5  2013     5     1        9          1655         434     308          2020
## 6  2013     6     1        2          2359           3     341           350
## 7  2013     7     1        1          2029         212     236          2359
## 8  2013     8     1       12          2130         162     257            14
## 9  2013     9     1        9          2359          10     343           340
## 10 2013    10     1     447           500         -13     614           648
## 11 2013    11     1        5          2359           6     352           345
## 12 2013    12     1       13          2359          14     446           445
## # ... with 14 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 <dtm>, duree_heure <dbl>,
## #   distance_km <dbl>, vitesse <dbl>
```

*#Attention : la clause group\_by marche pour les verbes déjà vus précédemment, sauf pour arrange, qui par défaut trie la table sans tenir compte des groupes. Pour obtenir un tri par groupe, il faut lui ajouter l'argument .by\_group = TRUE*

```
flights %>%
  group_by(month) %>%
  arrange(desc(dep_delay), .by_group = TRUE)
```

```
## # A tibble: 336,776 x 22
## # Groups:   month [12]
##   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     9     641             900        1301    1242         1530
## 2  2013     1    10    1121            1635        1126    1239         1810
## 3  2013     1     1     848            1835         853    1001         1950
## 4  2013     1    13    1809             810         599    2054         1042
## 5  2013     1    16    1622             800         502    1911         1054
## 6  2013     1    23    1551             753         478    1812         1006
## 7  2013     1    10    1525             900         385    1713         1039
## 8  2013     1     1    2343            1724         379     314         1938
## 9  2013     1     2    2131            1512         379    2340         1741
## 10 2013     1     7    2021            1415         366    2332         1724
## # ... with 336,766 more rows, and 14 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 <dtm>,
## #   duree_heure <dbl>, distance_km <dbl>, vitesse <dbl>
```

Tables multiples: Concaténation + Jointure à ajouter pour compléter !