R Avancé

Partie 1: fondamentaux / manipulation de données avec dplyr

Romain François

romain@r-enthusiasts.com

@romain_francois

- R depuis 2002
- Consultant Freelance
- R/C++ : Rcpp
- Performance
- dplyr

à vous

Qui êtes vous et pourquoi utilisez-vous R ?

Vecteurs

Quels sont les types communs de vecteurs dits "atomic" ?

Discutez-en avec votre voisin une minute

character

```
> c("foo", "bar" )
[1] "foo" "bar"
> letters
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
[20] "t" "u" "v" "w" "x" "y" "z"
```

integer

```
> 1:10
 [1]  1  2  3  4  5  6  7  8  9 10
> c(1L, 5L)
[1] 1 5
```

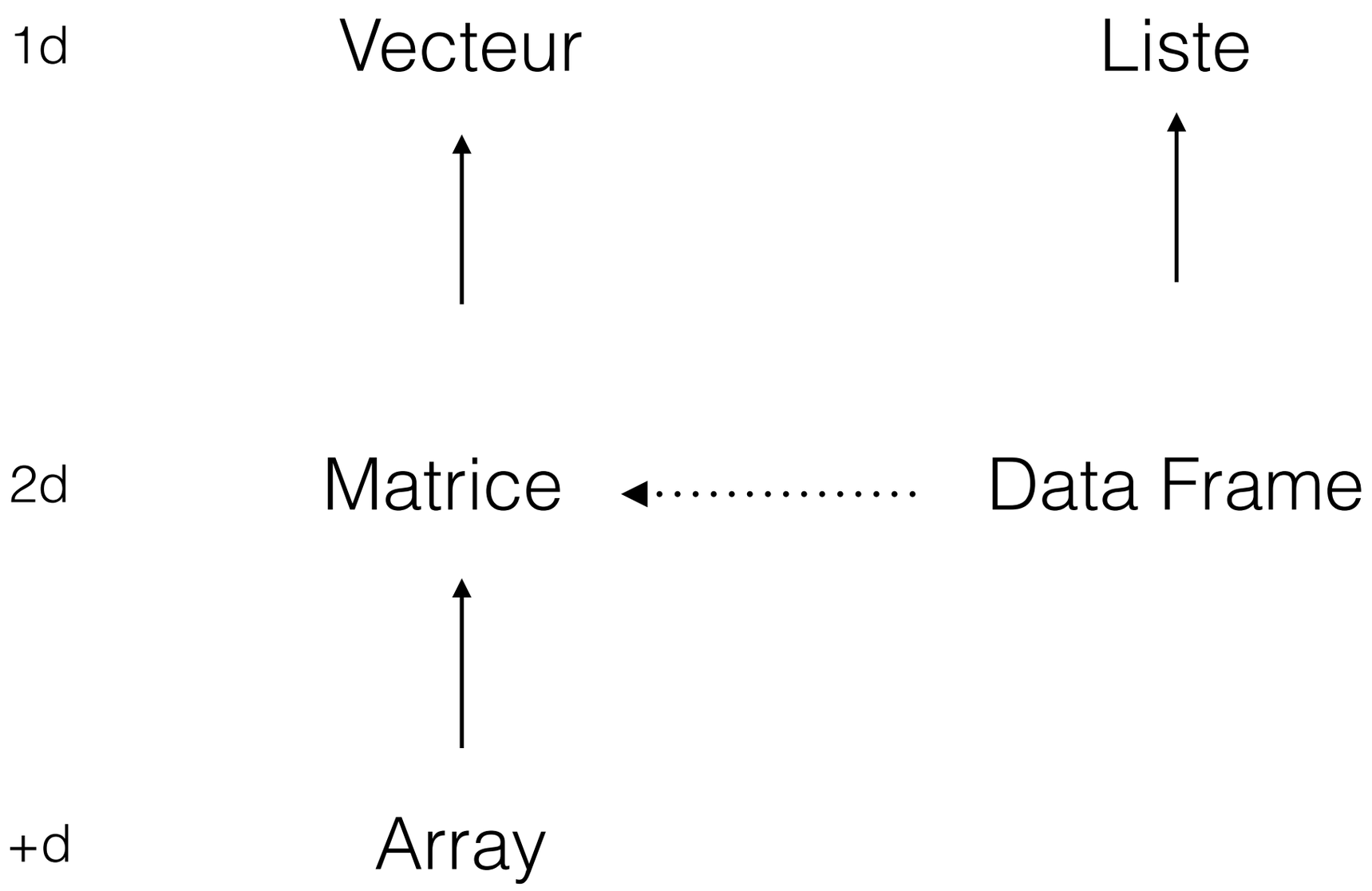
numeric

```
> rnorm(5)
[1]  0.7253443  0.4291835 -0.9519904  1.4014898 -1.4496602
> c(0,0,7)
[1] 0 0 7
```

logical

```
> c(T,F)
[1]  TRUE FALSE
> as.logical(c(0, 1, 100))
[1] FALSE  TRUE  TRUE
> c(TRUE, FALSE)
[1]  TRUE FALSE
```

- Pourquoi une liste est différente d'un vecteur "atomic" ?
- En quoi une data.frame diffère d'une matrix ?
- Comment examine t'on la structure d'un object ?


1d
Vecteur
Liste
2d
Matrice
Data Frame
+d
Array
même types
de données
types
de données
heterogenes

```
> head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1

> str(mtcars)
'data.frame':   32 obs. of  11 variables:
$ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
$ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
$ disp: num  160 160 108 258 360 ...
$ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
$ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
$ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
$ qsec: num  16.5 17 18.6 19.4 17 ...
$ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
$ am  : num  1 1 1 0 0 0 0 0 0 0 ...
$ gear: num  4 4 4 3 3 3 3 4 4 4 ...
$ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

```
> .Internal(inspect(mtcars))
@7fe2e68b3d48 19 VECSXP g0c5 [OBJ,MARK,NAM(2),ATT] (len=11, tl=0)
  @7fe2e0441890 14 REALSXP g0c7 [MARK,NAM(2)] (len=32, tl=0) 21,21,22.8,21.4,18.7,...
  @7fe2e0476bd0 14 REALSXP g0c7 [MARK,NAM(2)] (len=32, tl=0) 6,6,4,6,8,...
  @7fe2e04698b0 14 REALSXP g0c7 [MARK,NAM(2)] (len=32, tl=0) 160,160,108,258,360,...
  @7fe2e049ad80 14 REALSXP g0c7 [MARK,NAM(2)] (len=32, tl=0) 110,110,93,110,175,...
  @7fe2e04b2540 14 REALSXP g0c7 [MARK,NAM(2)] (len=32, tl=0) 3.9,3.9,3.85,3.08,3.15,...
  ...
ATTRIB:
  @7fe2e42388e8 02 LISTSXP g0c0 [MARK]
    TAG: @7fe2e081ba78 01 SYMSXP g1c0 [MARK,NAM(2),LCK,gp=0x4000] "names" (has value)
    @7fe2e68b3df0 16 STRSXP g0c5 [MARK,NAM(2)] (len=11, tl=0)
      @7fe2e1580268 09 CHARSXP g0c1 [MARK,gp=0x61] [ASCII] [cached] "mpg"
      @7fe2e1580238 09 CHARSXP g0c1 [MARK,gp=0x61,ATT] [ASCII] [cached] "cyl"
      @7fe2e1580208 09 CHARSXP g0c1 [MARK,gp=0x61] [ASCII] [cached] "disp"
      @7fe2e1581158 09 CHARSXP g0c1 [MARK,gp=0x61] [ASCII] [cached] "hp"
      @7fe2e1581128 09 CHARSXP g0c1 [MARK,gp=0x61] [ASCII] [cached] "drat"
    ...
TAG: @7fe2e08259f0 01 SYMSXP g1c0 [MARK,LCK,gp=0x4000] "row.names" (has value)
@7fe2e042da80 16 STRSXP g0c7 [MARK,NAM(2)] (len=32, tl=0)
  @7fe2e0d410a8 09 CHARSXP g0c2 [MARK,gp=0x60] [ASCII] [cached] "Mazda RX4"
  @7fe2e0d411f8 09 CHARSXP g0c2 [MARK,gp=0x60] [ASCII] [cached] "Mazda RX4 Wag"
  @7fe2e0d41230 09 CHARSXP g0c2 [MARK,gp=0x60] [ASCII] [cached] "Datsun 710"
  @7fe2e0d412d8 09 CHARSXP g0c2 [MARK,gp=0x60] [ASCII] [cached] "Hornet 4 Drive"
  @7fe2e1915288 09 CHARSXP g0c3 [MARK,gp=0x60] [ASCII] [cached] "Hornet Sportabout"
  ...
TAG: @7fe2e081bf48 01 SYMSXP g1c0 [MARK,LCK,gp=0x6000] "class" (has value)
@7fe2e1580fd8 16 STRSXP g0c1 [MARK,NAM(2)] (len=1, tl=0)
  @7fe2e08862d0 09 CHARSXP g1c2 [MARK,gp=0x61] [ASCII] [cached] "data.frame"
```

> mercredi

$\lceil, \lceil\lceil, \$$

- Quels sont les 5 types utilisables pour extraire d'un vecteur avec [
- Quelle est la différence entre `[` et `[[`
- Quand peut-on utiliser `drop = FALSE`

vide	inclure tout
integer	positif: inclusion negatif: exclusion
logical	garde les TRUE
character	via les noms

```
> x <- 1:26
> names(x) <- letters

> x[]
 a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s  t  u  v  w  x  y  z
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26

> x[c(1,2,3)]
a b c
1 2 3
> x[-(1:10)]
 k  l  m  n  o  p  q  r  s  t  u  v  w  x  y  z
11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26

> x[ c("x", "y", "z" ) ]
 x  y  z
24 25 26

> x[ x > 15 ]
 p  q  r  s  t  u  v  w  x  y  z
16 17 18 19 20 21 22 23 24 25 26
```

```
> data <- list( x = 1:10, y = letters, z = list(1,2,3) )

> data[[2]]
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
[20] "t" "u" "v" "w" "x" "y" "z"

> data[2]
$y
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
[20] "t" "u" "v" "w" "x" "y" "z"

> data[1:2]
$x
 [1]  1  2  3  4  5  6  7  8  9 10

$y
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
[20] "t" "u" "v" "w" "x" "y" "z"

> data[[1:2]]
# ???
```

```
> m <- matrix( 1:20, nrow = 4 )

> m
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
> m[1,]
[1]  1  5  9 13 17
> m[1,, drop = FALSE]
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
>
> m[, 1, drop = FALSE]
     [,1]
[1,]    1
[2,]    2
[3,]    3
[4,]    4
```

```
> mtcars[1:5, "cyl"]
[1] 6 6 4 6 8
> mtcars[1:5, "cyl", drop = FALSE]
                  cyl
Mazda RX4           6
Mazda RX4 Wag       6
Datsun 710          4
Hornet 4 Drive      6
Hornet Sportabout   8
```

- Quels sont les 3 façons de passer un argument à une fonction ?

- A quoi sert . . . ?

Argument matching:

- nom complet
- nom partiel
- position

... capture tout le reste

Scoping

```
x <- 5
f <- function(){
  y <- 10
  c(x=x, y=y)
}
f()

g <- function(){
  x <- 20
  y <- 10
  c(x=x, y=y)
}
g()

h <- function(){
  y <- 10
  i <- function(){
    z <- 20
    c(x=x, y=y, z=z)
  }
  i()
}
h()
```

```
j <- function(){
  if( exists("a") ){
     a <- a  + 1
  } else {
     a <- 5
  }
  a
}
```


dpiyr

`nycflights13`: Data about flights departing NYC in 2013

Données (nycflights13)

- flights [336 776 x 16]. Vols partant de NYC en 2013
- planes [3322 x 9]. Meta données sur les avions
- airports [1397 x 7]. Meta données sur les aéroports.

flights

```
> flights
Source: local data frame [336,776 x 16]

    year month   day dep_time dep_delay arr_time arr_delay carrier tailnum
   (int) (int) (int)    (int)     (dbl)    (int)     (dbl)   (chr)   (chr)
1   2013     1     1      517         2      830        11      UA  N14228
2   2013     1     1      533         4      850        20      UA  N24211
3   2013     1     1      542         2      923        33      AA  N619AA
4   2013     1     1      544        -1     1004       -18      B6  N804JB
5   2013     1     1      554        -6      812       -25      DL  N668DN
6   2013     1     1      554        -4      740        12      UA  N39463
7   2013     1     1      555        -5      913        19      B6  N516JB
8   2013     1     1      557        -3      709       -14      EV  N829AS
9   2013     1     1      557        -3      838        -8      B6  N593JB
10  2013     1     1      558        -2      753         8      AA  N3ALAA
..   ...   ...   ...      ...       ...      ...       ...     ...     ...
Variables not shown: flight (int), origin (chr), dest (chr), air_time (dbl),
  distance (dbl), hour (dbl), minute (dbl)

> class( flights )
[1] "tbl_df"     "tbl"        "data.frame"
```

flights

```
> glimpse(flights)
Observations: 336,776
Variables: 16
$ year      (int) 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 201...
$ month     (int) 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
$ day       (int) 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
$ dep_time  (int) 517, 533, 542, 544, 554, 554, 555, 557, 557, 558, 558, 55...
$ dep_delay (dbl) 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2, -2, -2, -1, ...
$ arr_time  (int) 830, 850, 923, 1004, 812, 740, 913, 709, 838, 753, 849, 8...
$ arr_delay (dbl) 11, 20, 33, -18, -25, 12, 19, -14, -8, 8, -2, -3, 7, -14,...
$ carrier   (chr) "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV", "B6", "AA...
$ tailnum   (chr) "N14228", "N24211", "N619AA", "N804JB", "N668DN", "N39463...
$ flight    (int) 1545, 1714, 1141, 725, 461, 1696, 507, 5708, 79, 301, 49,...
$ origin    (chr) "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR", "LGA", "...
$ dest      (chr) "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL", "IAD", "...
$ air_time  (dbl) 227, 227, 160, 183, 116, 150, 158, 53, 140, 138, 149, 158...
$ distance  (dbl) 1400, 1416, 1089, 1576, 762, 719, 1065, 229, 944, 733, 10...
$ hour      (dbl) 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, ...
$ minute    (dbl) 17, 33, 42, 44, 54, 54, 55, 57, 57, 58, 58, 58, 58, 58, 5...
```

filter

```
df <- data_frame(
  color = c("blue", "black", "blue", "blue", "black"),
  value = 1:5
)
```

```
> df
  color value
1  blue     1
2 black     2
3  blue     3
4  blue     4
5 black     5
```

```
> filter( df, color == "blue" )
 color value
1  blue     1
2  blue     3
3  blue     4
```

```
> df
  color value
1  blue     1
2 black     2
3  blue     3
4  blue     4
5 black     5

> filter( df, value < 3 )
  color value
1  blue     1
2 black     2
```

Trouver les vols:

- Pour LAX ou SFO
- De janvier
- Qui ont plus d'une heure de délai
- Qui sont partis entre minuit et 5h
- Qui sont partis le matin et en avance
- Qui ont un délai à l'arrivée supérieur au délai au départ

```
filter( flights, dest %in% c("LAX", "SFO") )
filter( flights, dest %in% c("LAX", "SFO") )
filter( flights, month == 1 )
filter( flights, dep_delay > 60 )
filter( flights, hour <  5 )
filter( flights, hour < 12, dep_delay < 0 )
filter( flights, arr_delay > dep_delay  )
```

select

```
> df
  color value
1  blue     1
2 black     2
3  blue     3
4  blue     4
5 black     5
```

```
>  select( df, color )
  color
1  blue
2 black
3  blue
4  blue
5 black
```

```
>  select( df, -color )
  value
1     1
2     2
3     3
4     4
5     5
```

- ?select. Trouver d'autres moyen de sélectionner des variables avec select

- Ecrire plusieurs façons de sélectionner les deux variables de délai (`arr_delay` et `dep_delay`)

arrange

```
> df <- data_frame(
+   x = c(3,1,2,5,4),
+   y = c("a", "b", "c", "d", "e")
+ )

> arrange( df, x )
Source: local data frame [5 x 2]

      x     y
  (dbl) (chr)
1     1     b
2     2     c
3     3     a
4     4     e
5     5     d

> arrange( df, desc(x) )
Source: local data frame [5 x 2]

      x     y
  (dbl) (chr)
1     5     d
2     4     e
3     3     a
4     2     c
5     1     b
```

- Les vols qui ont eu le plus de retard

- Les vols qui ont le plus rattrapé leur retard

mutate

```
> df <- data_frame(
+   color = c("blue", "black", "blue", "blue", "black"),
+   value = 1:5
+ )

> mutate( df, double = 2*value )
Source: local data frame [5 x 3]

  color value double
  (chr) (int)  (dbl)
1  blue     1      2
2 black     2      4
3  blue     3      6
4  blue     4      8
5 black     5     10

> mutate( df, double = 2*value, quadruple = 2*double )
Source: local data frame [5 x 4]

  color value double quadruple
  (chr) (int)  (dbl)     (dbl)
1  blue     1      2         4
2 black     2      4         8
3  blue     3      6        12
4  blue     4      8        16
5 black     5     10        20
```

- Créer une nouvelle variable `dist_km`, distance en km (1 mile= 1.60934 km)

- Ajouter une variable `vitesse` (en km / h)

- Ajouter une variable qui mesure le temps rattrapé (ou perdu) pendant le vol

summarise

```
> df <- data_frame(
+   color = c("blue", "black", "blue", "blue", "black"),
+   value = 1:5
+ )

> summarise( df, total = sum(value) )
Source: local data frame [1 x 1]

  total
  (int)
1    15
```

- La plus petite distance parcourue

- Le délai moyen au départ, à l'arrivée

group_by

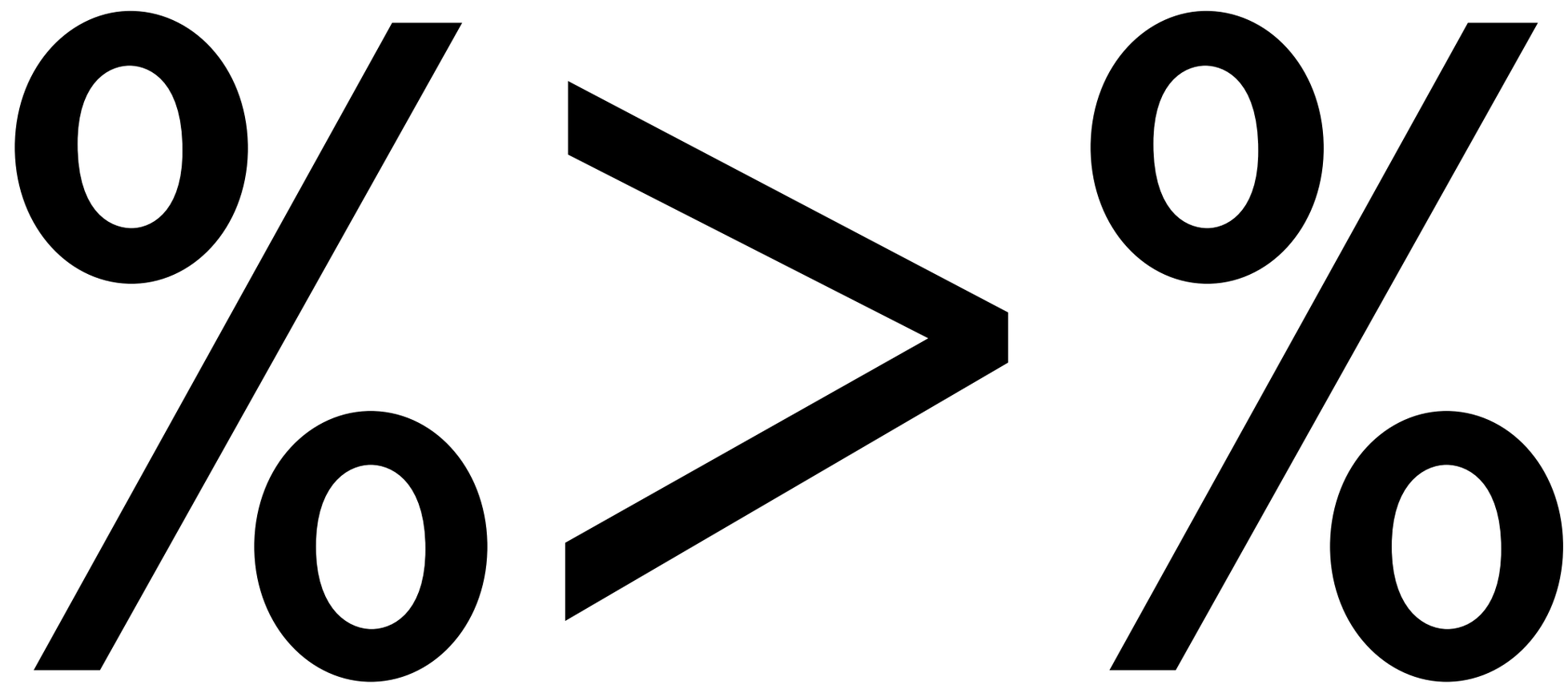
```
> df
Source: local data frame [5 x 2]

  color value
  (chr) (int)
1  blue     1
2 black     2
3  blue     3
4  blue     4
5 black     5
> by_color <- group_by(df, color)
> summarise( by_color, total = sum(value) )
Source: local data frame [2 x 2]

  color total
  (chr) (int)
1 black     7
2  blue     8
```

- Délais minimum, maximum et moyen pour chaque destination (dest)

- Le délai moyen au départ, à l'arrivée



```
enjoy(cool(bake(shape(beat(append(bowl(rep("flour",
2), "yeast", "water", "milk", "oil"), "flour", until
= "soft"), duration = "3mins"), as = "balls", style =
"slightly-flat"), degrees = 200, duration =
"15mins"), duration = "5mins"))
```

```
bowl(rep("flour", 2), "yeast", "water", "milk", "oil") %>%
    append("flour", until = "soft") %>%
    beat(duration = "3mins") %>%
    shape(as = "balls", style = "slightly-flat") %>%
    bake(degrees = 200, duration = "15mins") %>%
    cool(buns, duration = "5mins") %>%
    enjoy()
```

```
filter(
  summarise(
    group_by(
      filter( flights, !is.na(dep_delay) ),
      month, day, hour
    ),
    delay = mean(dep_delay),
    n = n()
  ),
  n > 10
)
```

```
flights %>%
  filter( !is.na(dep_delay) ) %>%
  group_by( month, day, hour ) %>%
  summarise( delay = mean(dep_delay), n = n() ) %>%
  filter( n > 10 )
```

```
flights %>%
  group_by(dest) %>%
  summarise( delay = mean(dep_delay, na.rm = TRUE ) ) %>%
  arrange( desc(delay) ) %>%
  head( 3 )
```

- Destinations qui ont le plus grand délai moyen

- Quels vols sont effectués tous les jours (carrier + flight).

```
> flights %>%
    group_by( carrier, flight, dest ) %>%
    summarise( n = n() ) %>%
    filter( n == 365 )
```

group_by

+

mutate/filter

- Créer une variable min_air_time, le vol le plus rapide allant à la même destination
- Pour chaque destination, isoler le vol le plus rapide
- isoler les 5 plus rapides (par ex, min_rank)

Joins

Type	Action
inner_join	Lignes **à la fois** dans x et y
left_join	Toutes les lignes de x, et celles qui correspondent dans y
semi_join	Lignes de x qui correspondent à des lignes de y
anti_join	Lignes de x qui ne correspondent pas à des lignes de y

- Les avions les plus anciens sont-ils plus lents ?
  - Calculer vitesse moyenne pour chaque avion (tailnum)
  - Join avec le jeu de données planes pour avoir l'année de l'avion