Tema 2

Organizar los datos en R

2.1 Introducción

```
> library(tidyverse)
> library(UsingR)
> library(readxl)
> library(data.table)
> library(formattable)
> library(xtable)
```

2.2 Estructura básica de datos en R: data.frame

La estructura básica de datos en R es el **data.frame**. Básicamente, es el equivalente a una tabla donde los datos aparecen organizados de manera que en las columnas tenemos las variables y en las filas los individuos[1]. En R tenemos disponibles muchos conjuntos de datos con los que podemos experimentar. Por ejemplo, en el paquete **UsingR** tenemos los datos **Indometh** se recogen los datos de la farmacocinética de indometacina medida a distintos tiempos en un conjunto de sujetos:

```
> Indometh
```

```
Grouped Data: conc ~ time | Subject
   Subject time conc
1        1 0.25 1.50
2        1 0.50 0.94
3        1 0.75 0.78
4        1 1.00 0.48
5        1 1.25 0.37
6        1 2.00 0.19
7        1 3.00 0.12
```

[1] Veremos más adelante que esta estructura permite otras variantes.

```
8       1 4.00 0.11
9       1 5.00 0.08
10      1 6.00 0.07
11      1 8.00 0.05
12      2 0.25 2.03
13      2 0.50 1.63
14      2 0.75 0.71
15      2 1.00 0.70
16      2 1.25 0.64
17      2 2.00 0.36
18      2 3.00 0.32
19      2 4.00 0.20
20      2 5.00 0.25
21      2 6.00 0.12
22      2 8.00 0.08
23      3 0.25 2.72
24      3 0.50 1.49
25      3 0.75 1.16
26      3 1.00 0.80
27      3 1.25 0.80
28      3 2.00 0.39
29      3 3.00 0.22
30      3 4.00 0.12
31      3 5.00 0.11
32      3 6.00 0.08
33      3 8.00 0.08
34      4 0.25 1.85
35      4 0.50 1.39
36      4 0.75 1.02
37      4 1.00 0.89
38      4 1.25 0.59
39      4 2.00 0.40
40      4 3.00 0.16
41      4 4.00 0.11
42      4 5.00 0.10
43      4 6.00 0.07
44      4 8.00 0.07
45      5 0.25 2.05
46      5 0.50 1.04
47      5 0.75 0.81
48      5 1.00 0.39
49      5 1.25 0.30
50      5 2.00 0.23
51      5 3.00 0.13
52      5 4.00 0.11
53      5 5.00 0.08
54      5 6.00 0.10
55      5 8.00 0.06
56      6 0.25 2.31
57      6 0.50 1.44
58      6 0.75 1.03
```

```
59       6 1.00 0.84
60       6 1.25 0.64
61       6 2.00 0.42
62       6 3.00 0.24
63       6 4.00 0.17
64       6 5.00 0.13
65       6 6.00 0.10
66       6 8.00 0.09
```

Como vemos, hay 66 filas y 3 columnas. En las columnas tenemos las variables Subject, time, conc. Podemos observar que cada sujeto aparece repetido asociado a los distintos tiempos de medida y a la concentración detectada en cada tiempo. El listado de todo un data.frame es muy engorroso, en especial si contiene mucha información. Podemos listar solo los primeros casos:

```
> head(Indometh)
```

```
Grouped Data: conc ~ time | Subject
  Subject time conc
1       1 0.25 1.50
2       1 0.50 0.94
3       1 0.75 0.78
4       1 1.00 0.48
5       1 1.25 0.37
6       1 2.00 0.19
```

También, podemos explorar solo los nombres de las variables contenidas en el data.frame:

```
> colnames(Indometh)
```

```
[1] "Subject" "time"    "conc"
```

También podemos obtener el número de casos:

```
> nrow(Indometh)
```

```
[1] 66
```

Podemo seleccionar los casos por sus índices:

```
> Indometh[3:6,]
```

```
Grouped Data: conc ~ time | Subject
  Subject time conc
3       1 0.75 0.78
4       1 1.00 0.48
5       1 1.25 0.37
6       1 2.00 0.19
```

En este caso, seleccionamos los casos del 3 al 6 y mostramos todas las variables. Si indicamos:

```
> Indometh[3,2]
```

```
[1] 0.75
```

Obtenemos los datos de la segunda variable en la fila 3.

Para acceder a los datos de una variable en un data.frame utilizamos la sintaxis:

```
> Indometh$conc
```

```
 [1] 1.50 0.94 0.78 0.48 0.37 0.19 0.12 0.11 0.08 0.07 0.05 2.03 1.63
[14] 0.71 0.70 0.64 0.36 0.32 0.20 0.25 0.12 0.08 2.72 1.49 1.16 0.80
[27] 0.80 0.39 0.22 0.12 0.11 0.08 0.08 1.85 1.39 1.02 0.89 0.59 0.40
[40] 0.16 0.11 0.10 0.07 0.07 2.05 1.04 0.81 0.39 0.30 0.23 0.13 0.11
[53] 0.08 0.10 0.06 2.31 1.44 1.03 0.84 0.64 0.42 0.24 0.17 0.13 0.10
[66] 0.09
```

Es importante entender el concepto de **factor**. En este caso, la vaiable **Subject** es un factor con 6 niveles:

```
> Indometh$Subject
```

```
 [1] 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3
[34] 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 6 6
Levels: 1 < 4 < 2 < 5 < 6 < 3
```

Es importante definir una variable como faor cuando se refiere a un característica que clasifica a los individuos (variable nominal). Por ejemplo: Hombre/Mujer, Mejora/No mejora, +/++/+++.

Podemos seleccionar sólo los datos que cumplan cierta condición:

```
> Indometh[Indometh$Subject==1,]
```

```
Grouped Data: conc ~ time | Subject
   Subject time conc
1        1 0.25 1.50
2        1 0.50 0.94
3        1 0.75 0.78
4        1 1.00 0.48
5        1 1.25 0.37
6        1 2.00 0.19
7        1 3.00 0.12
8        1 4.00 0.11
9        1 5.00 0.08
10       1 6.00 0.07
11       1 8.00 0.05
```

Podemos complicar un poco más la selección. Por ejemplo, los datos del individuo 2 a partir del tiempo 2:

```
> Indometh[Indometh$Subject==2 & Indometh$time>2,]
```

```
Grouped Data: conc ~ time | Subject
   Subject time conc
18       2    3 0.32
19       2    4 0.20
20       2    5 0.25
21       2    6 0.12
22       2    8 0.08
```

Calculemos ahora la media de concentraciones (variable 3) detectadas a tiempo 2.00 en todos los pacientes:

```
> conc2 <- Indometh[Indometh$time==2,3]
> conc2
```

```
[1] 0.19 0.36 0.39 0.40 0.23 0.42
```

```
> mean(conc2)
```

```
[1] 0.332
```

En el capítulo de dedicado a la gestión de datos, veremos como hacer selecciones más avanzadas agrupando datos con las funciones diponibles en los paquetes incluidos en **tidyverse**.

2.3 ¿Cómo definir un data.frame

2.3.1 Definición a partir de vectores

Si disponemos de un conjunto de vectores, podemos crear un data.frame de la siguiente manera:

```
> ind <- c(1,2,3,4,5,6,7)
> edad <- c(34,54,36,55,43,41,58)
> peso <- c(75,80,64,63,87,67,77)
> df <- data.frame(Caso=ind, Edad=edad, Peso=peso)
> df
```

```
  Caso Edad Peso
1    1   34   75
2    2   54   80
3    3   36   64
```

```
4    4   55   63
5    5   43   87
6    6   41   67
7    7   58   77
```

Este procedimiento es engorroso si se trata de incorporar grandes volúmenes de datos. En la práctica, es conveniente disponer los datos en una hoja de cálculo o un fichero de texto e incorporarlas a R. POr ejemplo **read_tsv** incorpora datos separados por tabuladores de un fichero :

```
> file <- 'E:\\Ejemplos\\fat.txt'
> fat <- read_tsv(file)
> head(fat)
```

```
# A tibble: 6 x 19
     id  PBF1 PBF2  Density   Age Weight Height AdipIndex FatFree
  <dbl> <dbl> <chr> <chr>   <dbl>  <dbl>  <dbl>     <dbl>   <dbl>
1     1   126 12,3  10.708     23  15425   6775       237    1349
2     2    69 6,1   10.853     22  17325   7225       234    1613
3     3   246 25,3  10.414     22    154   6625       247     116
4     4   109 10,4  10.751     26  18475   7225       249    1647
5     5   278 28,7  1.034      24  18425   7125       256    1331
6     6   206 20,9  10.502     24  21025   7475       265     167
# ... with 10 more variables: Neck <dbl>, Chest <dbl>, Abdomen <dbl>,
#   Hip <dbl>, Thigh <dbl>, Knee <dbl>, Ankle <dbl>, Biceps <dbl>,
#   Forearm <dbl>, Wrist <dbl>
```

En este caso, el resultado es un **tibble**, una estructura similar a **data.frame** que mejora la manera de gestionar los datos. La presentación de resultados es más compacta como tibble. Además, las funciones de los paquetes modernos de R permiten una mayor flexibilidad en el tratamiento de datos en formato tibble.

Los datos pueden estar en ficheros delimitados por comas, punto y coma, y otros delimitadores. En este enlace se describen los distintos casos.

2.3.2 Adquisición de datos a partir de programas externos

R puede adquirir datos de prácticamente cualquier fuente. Como primer ejemplo, veamos como podemos acceder a un fichero de Microsoft Excel que tengamos en nuestro disco duro[2]

```
> file <- 'E:\\Ejemplos\\Diabetes.xls'
> diabetes <- read_excel(file)
> head(diabetes)
```

```
# A tibble: 6 x 19
     id  chol stab.glu   hdl ratio glyhb location   age gender height
```

[2]Se debe especificar, en cada caso, donde está el fichero. La dirección utilizada es un ejemplo.

```
  <dbl> <dbl>    <dbl> <dbl> <dbl> <dbl> <chr>    <dbl> <chr>   <dbl>
1  1000   203       82    56  3.60  4.31 Bucking~    46 female     62
2  1001   165       97    24  6.90  4.44 Bucking~    29 female     64
3  1002   228       92    37  6.20  4.64 Bucking~    58 female     61
4  1003    78       93    12  6.5   4.63 Bucking~    67 male       67
5  1005   249       90    28  8.90  7.72 Bucking~    64 male       68
6  1008   248       94    69  3.60  4.81 Bucking~    34 male       71
# ... with 9 more variables: weight <dbl>, frame <chr>, bp.1s <dbl>,
#   bp.1d <dbl>, bp.2s <dbl>, bp.2d <dbl>, waist <dbl>, hip <dbl>,
#   time.ppn <dbl>
```

El resultado está en formato tibble. En caso necesario, se puede tranformar un tibble en un data.frame[3]:

```
> df.diabetes<-as.data.frame(diabetes)
> head(df.diabetes)
```

```
    id chol stab.glu hdl ratio glyhb   location age gender height
1 1000  203       82  56   3.6  4.31 Buckingham  46 female     62
2 1001  165       97  24   6.9  4.44 Buckingham  29 female     64
3 1002  228       92  37   6.2  4.64 Buckingham  58 female     61
4 1003   78       93  12   6.5  4.63 Buckingham  67   male     67
5 1005  249       90  28   8.9  7.72 Buckingham  64   male     68
6 1008  248       94  69   3.6  4.81 Buckingham  34   male     71
  weight  frame bp.1s bp.1d bp.2s bp.2d waist hip time.ppn
1    121 medium   118    59    NA    NA    29  38      720
2    218  large   112    68    NA    NA    46  48      360
3    256  large   190    92   185    92    49  57      180
4    119  large   110    50    NA    NA    33  38      480
5    183 medium   138    80    NA    NA    44  41      300
6    190  large   132    86    NA    NA    36  42      195
```

En algunos casos puede ser conveniente acceder a datos en formato excel que estan en una determinada url. El procedimiento es algo complicado, pero permite adquirir este tipo de datos. Por ejemplo

```
> url <- 'http://jse.amstat.org/datasets/kuiper.xls'
> temp <- tempfile()
> download.file(url,temp,mode='wb')
> kuiper <-read_excel(path=temp)
> head(kuiper)
```

```
# A tibble: 6 x 12
   Price Mileage Make  Model Trim  Type  Cylinder Liter Doors Cruise
   <dbl>   <dbl> <chr> <chr> <chr> <chr>    <dbl> <dbl> <dbl>  <dbl>
1 17314.    8221 Buick Cent~ Seda~ Sedan        6   3.1     4      1
```

[3] Aunque el formato tibble está ámpliamente aceptado, algunos paquetes importantes trabajan exclusivamente con data.frame. Por lo tanto, es importante entender cómo transformar un tipo en otro.

```
2 17542.    9135 Buick Cent~ Seda~ Sedan        6   3.1     4      1
3 16219.   13196 Buick Cent~ Seda~ Sedan        6   3.1     4      1
4 16337.   16342 Buick Cent~ Seda~ Sedan        6   3.1     4      1
5 16339.   19832 Buick Cent~ Seda~ Sedan        6   3.1     4      1
6 15709.   22236 Buick Cent~ Seda~ Sedan        6   3.1     4      1
# ... with 2 more variables: Sound <dbl>, Leather <dbl>
```

2.4 Ejemplos de aplicación y algunos casos de interés

2.4.1 Definición de factores

Supongamos que tenemos datos de hombres (código 1) y mujeres (código 2) en un vector. Si lo definimos así:

```
> sexo <- c(1,1,1,2,2,2,2)
> edad <- c(34,54,36,55,43,41,58)
> peso <- c(75,80,64,63,87,67,77)
> df <- data.frame(Sexo=sexo, Edad=edad, Peso=peso)
> df
```

```
  Sexo Edad Peso
1    1   34   75
2    1   54   80
3    1   36   64
4    2   55   63
5    2   43   87
6    2   41   67
7    2   58   77
```

la variable sexo es numérica (lo que no tiene sentido):

```
> df$Sexo
```

```
[1] 1 1 1 2 2 2 2
```

Es conveniente definir el sexo como un **factor**:

```
> sexo <- c(1,1,1,2,2,2,2)
> edad <- c(34,54,36,55,43,41,58)
> peso <- c(75,80,64,63,87,67,77)
> df2 <- data.frame(Sexo=factor(sexo), Edad=edad, Peso=peso)
> df2
```

```
  Sexo Edad Peso
1    1   34   75
```

```
2    1   54   80
3    1   36   64
4    2   55   63
5    2   43   87
6    2   41   67
7    2   58   77
```

```
> df2$Sexo
```

```
[1] 1 1 1 2 2 2 2
Levels: 1 2
```

La diferencia es importante para facilitar el tratamiento de la variable Sexo como nominal (grupos). Por ejemplo, ahora no podemos aplicar métodos numéricos al Sexo definido como factor. En el primer caso, Sexo contiene valores numésricos y se puede calcular la media, lo que no tiene ningún sentido. En el segundo caso, se produce un error dado que no es una variable numérica.

```
> mean(df$Sexo)
```

```
[1] 1.57
```

```
> mean(df2$Sexo)
```

```
[1] NA
```

2.4.2 Juntar varios data.frame por columnas

Si tenemos diversos data.frame con los mismos individuos en el mismo orden y distintas variables los podemos juntar mediante **cbind**:

```
> nombre <- c('Jose','Isabel','Jorge','Montse','Matias','Isidre','Mateo')
> sexo <- c(1,1,1,2,2,2,2)
> edad <- c(34,54,36,55,43,41,58)
> peso <- c(75,80,64,63,87,67,77)
> df1 <- data.frame(Nombre=nombre, Edad=edad, Peso=peso)
> df1
```

```
  Nombre Edad Peso
1   Jose   34   75
2 Isabel   54   80
3  Jorge   36   64
4 Montse   55   63
5 Matias   43   87
6 Isidre   41   67
7  Mateo   58   77
```

```
> df2 <- data.frame(Nombre=nombre, Sexo=factor(sexo))
> df2

  Nombre Sexo
1   Jose    1
2 Isabel    1
3  Jorge    1
4 Montse    2
5 Matias    2
6 Isidre    2
7  Mateo    2

> cbind(df1,df2)

  Nombre Edad Peso Nombre Sexo
1   Jose   34   75   Jose    1
2 Isabel   54   80 Isabel    1
3  Jorge   36   64  Jorge    1
4 Montse   55   63 Montse    2
5 Matias   43   87 Matias    2
6 Isidre   41   67 Isidre    2
7  Mateo   58   77  Mateo    2
```

2.4.3 Juntar varios data.frame por filas

Si tenemos diversos data.frame con las mismas variables los podemos juntar mediante **rbind**:

```
> nombre <- c('Jose','Isabel','Jorge')
> sexo <- c(1,1,1)
> edad <- c(34,54,36)
> peso <- c(75,80,64)
> df1 <- data.frame(Nombre=nombre, Sexo=sexo,Edad=edad, Peso=peso)
> df1

  Nombre Sexo Edad Peso
1   Jose    1   34   75
2 Isabel    1   54   80
3  Jorge    1   36   64

> nombre <- c('Montse','Matias','Isidre','Mateo')
> sexo <- c(2,2,2,2)
> edad <- c(355,43,41,58)
> peso <- c(63,87,67,77)
> df2 <- data.frame(Nombre=nombre, Sexo=sexo,Edad=edad, Peso=peso)
> df2

  Nombre Sexo Edad Peso
1 Montse    2  355   63
```

```
2 Matias    2   43   87
3 Isidre    2   41   67
4  Mateo    2   58   77
```

```
> rbind(df1,df2)
```

```
  Nombre Sexo Edad Peso
1   Jose    1   34   75
2 Isabel    1   54   80
3  Jorge    1   36   64
4 Montse    2  355   63
5 Matias    2   43   87
6 Isidre    2   41   67
7  Mateo    2   58   77
```

2.4.4 Crear un data.frame con resultados de un análisis

Como ejemplo, vamos a generar datos de una normal N(0,1) con la función **rnorm** y aplicaremos **t.test** para estimar el intervalo de confianza de μ:

```
> res <- rbind(t.test(rnorm(30))$conf.int,
+       t.test(rnorm(30))$conf.int,
+       t.test(rnorm(30))$conf.int)
> res
```

```
       [,1]     [,2]
[1,] -0.673 -0.00319
[2,] -0.519  0.27804
[3,] -0.449  0.27999
```

Para poner nombres a las variables y a las filas podemos hacer:

```
> colnames(res) <- c('Inferior','Superior')
> rownames(res) <- c('Muestra 1','Muestra 2','Muestra 3')
> res
```

```
          Inferior Superior
Muestra 1   -0.673 -0.00319
Muestra 2   -0.519  0.27804
Muestra 3   -0.449  0.27999
```

2.5 El paquete tidyverse

Ahora veremos como profundizar en la gestión de datos, su reorganización y el cálculo de resúmenes estadísticos por grupos. Con R podemos utilizar muchas estrategias en función de

lo que queramos hacer. Las funciones del paquete **tidyverse** facilitan mucho este trabajo. A continuación discutiremos su funcionalidad básica con ejemplos que permitan comprender la sintaxis de este útil paquete.

2.5.1 Agrupar datos: group_by

Para empezar, debemos disponer de nuestros datos en forma de data.frame. Utilizaremos como ejemplo los datos **Orange**. Estos datos contienen la circumferencia de 5 árboles a distintas edades:

```
> head(Orange)
```

```
Grouped Data: circumference ~ age | Tree
  Tree  age circumference
1    1  118            30
2    1  484            58
3    1  664            87
4    1 1004           115
5    1 1231           120
6    1 1372           142
```

¿Cómo podemos calcular la circumferencia media de los árboles estudiados en cada edad? Para ello, primero vamos a agrupar los datos por age:

```
> Orange %>% group_by(age)
```

```
# A tibble: 35 x 3
# Groups:   age [7]
   Tree    age circumference
 * <ord> <dbl>         <dbl>
 1 1       118            30
 2 1       484            58
 3 1       664            87
 4 1      1004           115
 5 1      1231           120
 6 1      1372           142
 7 1      1582           145
 8 2       118            33
 9 2       484            69
10 2       664           111
# ... with 25 more rows
```

Aparentemente no sucede mucha cosa. Básicamente, ahora disponemos de datos agrupados por age y podemos hacer cálculos que se aplicaran a cada subgrupo de edad. El operador especial %>% indica que aplicamos la función siguiente al resultado de la anterior:

```
> Orange %>% group_by(age) %>% summarise(n=n(),m=mean(circumference))
```

```
# A tibble: 7 x 3
    age     n     m
  <dbl> <int> <dbl>
1   118     5  31
2   484     5  57.8
3   664     5  93.2
4  1004     5 134.
5  1231     5 146.
6  1372     5 173.
7  1582     5 176.
```

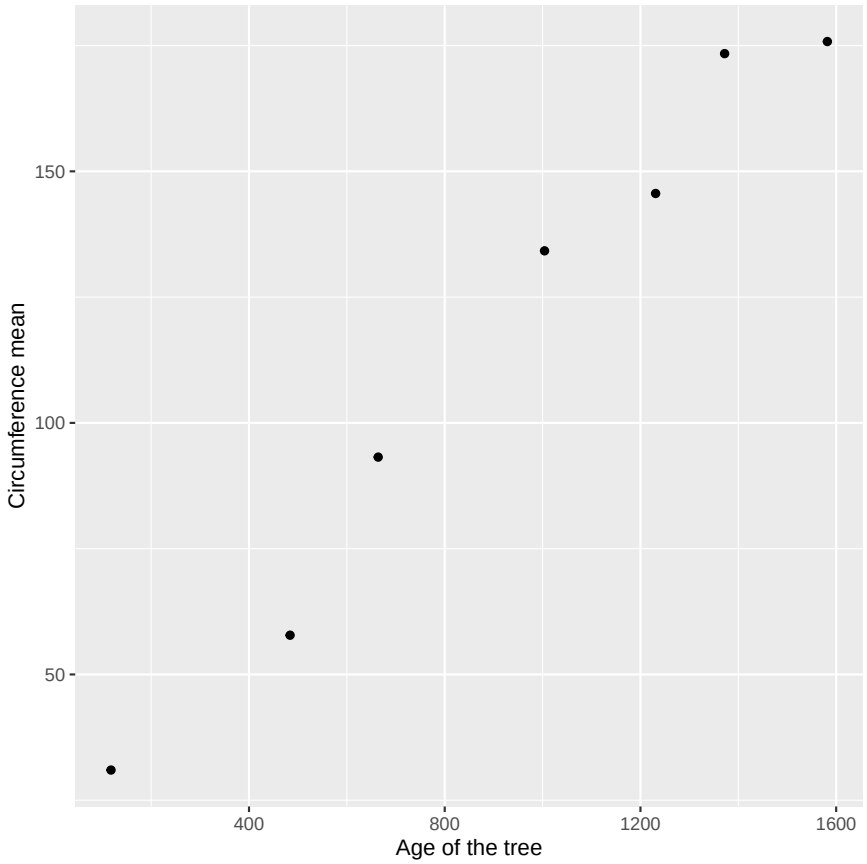
Podemos observar que se ha calculado el número de árboles en cada grupo de edad con la función **n()** y la media de circumferencia de los árboles de cada edad. El resultado es un tipo especial de data.frame que puede utilizarse para otros cálculos. Podemos añadir más información, por ejemplo la desviación estándar de la circumferencia en cada edad (asignamos el resultado a una variable para su uso posterior):

```
> d<-Orange %>% group_by(age) %>%
+   summarise(n=n(),m=mean(circumference),de=sd(circumference))
> d
```

```
# A tibble: 7 x 4
    age     n     m    de
  <dbl> <int> <dbl> <dbl>
1   118     5  31    1.41
2   484     5  57.8  8.17
3   664     5  93.2 17.2
4  1004     5 134.  25.9
5  1231     5 146.  29.2
6  1372     5 173.  32.8
7  1582     5 176.  33.3
```

Ahora, como ejemplo podemos hacer una gráfica que muestre la evolución de la media:

```
> ggplot(d,aes(x=age,y=m))+geom_point()+
+   ylab('Circumference mean')+
+   xlab('Age of the tree')
```


150
100
50
Circumference mean
400
800
1200
1600
Age of the tree

Veamos un ejemplo más complicado. Para ello, utilizaremos los daos **survey**:

```
> head(survey)
```

```
     Sex Wr.Hnd NW.Hnd W.Hnd    Fold Pulse    Clap Exer Smoke Height
1 Female   18.5   18.0 Right  R on L    92    Left Some Never    173
2   Male   19.5   20.5  Left  R on L   104    Left None Regul    178
3   Male   18.0   13.3 Right  L on R    87 Neither None Occas     NA
4   Male   18.8   18.9 Right  R on L    NA Neither None Never    160
5   Male   20.0   20.0 Right Neither    35   Right Some Never    165
6 Female   18.0   17.7 Right  L on R    64   Right Some Never    173
       M.I  Age
1   Metric 18.2
2 Imperial 17.6
3     <NA> 16.9
4   Metric 20.3
5   Metric 23.7
6 Imperial 21.0
```

La función **summarise** permite hacer todo tipo de cálculo. Por ejemplo, utilizando **t.test**, podemos calcular los Intervalos de Confianza de la diferencia de media entre la amplitud de la mano que escribe y la que no escribe destro de cada grupo de Sex y de si es diestro o zurdo:

```
> d  <- na.omit(survey) %>% group_by(Sex,W.Hnd)
> r1 <- d %>% summarise(n=n(),
+                       MWrHnd=mean(Wr.Hnd),MNWHnd=mean(NW.Hnd),
+                       ci1=t.test(Wr.Hnd,NW.Hnd)$conf.int[1],
+                       ci2=t.test(Wr.Hnd,NW.Hnd)$conf.int[2],
+                       pvalue=t.test(Wr.Hnd,NW.Hnd)$p.value[1])
> r1
```

```
# A tibble: 4 x 8
# Groups:   Sex [2]
  Sex    W.Hnd     n MWrHnd MNWHnd    ci1   ci2 pvalue
  <fct>  <fct> <int>  <dbl>  <dbl>  <dbl> <dbl>  <dbl>
1 Female Left      5   18.1   18.0 -2.12  2.32   0.918
2 Female Right    79   17.6   17.4 -0.266 0.552  0.491
3 Male   Left      7   20.2   20.0 -1.74  2.31   0.763
4 Male   Right    77   20.0   20.0 -0.573 0.523  0.929
```

La tabla resultante puede organizarse de manera algo más elegante:

	Sex	W.Hnd	n	MWrHnd	MNWHnd	ci1	ci2	pvalue
1	Female	Left	5	18.08	17.98	-2.12	2.32	0.92
2	Female	Right	79	17.59	17.45	-0.27	0.55	0.49
3	Male	Left	7	20.24	19.96	-1.74	2.31	0.76
4	Male	Right	77	19.96	19.98	-0.57	0.52	0.93

2.5.2 Selección de casos: filter

En muchos casos, es conveniente seleccionar casos en función de algún criterio. La sintaxis para ello es:

```
> Orange %>% filter(Tree==1)
```

```
  Tree  age circumference
1    1  118            30
2    1  484            58
3    1  664            87
4    1 1004           115
5    1 1231           120
6    1 1372           142
7    1 1582           145
```

En este caso se selecionan los datos del árbol 1. Podemos seleccionar los datos de circumferencia superior a 150:

```
> Orange %>% filter(circumference>=150)
```

```
   Tree  age circumference
1     2 1004           156
2     2 1231           172
3     2 1372           203
4     2 1582           203
5     4 1004           167
6     4 1231           179
7     4 1372           209
8     4 1582           214
9     5 1372           174
10    5 1582           177
```

Podemos filtar con criterio en dos variables:

```
> Orange %>% filter(circumference>=150 & age>1200)
```

```
  Tree  age circumference
1    2 1231           172
2    2 1372           203
3    2 1582           203
4    4 1231           179
5    4 1372           209
6    4 1582           214
7    5 1372           174
8    5 1582           177
```

También se pueden seleccionar valores de una variable dentro de un determinado rango:

```
> Orange %>% filter(150<circumference & circumference<200)
```

```
  Tree  age circumference
1    2 1004           156
2    2 1231           172
3    4 1004           167
4    4 1231           179
5    5 1372           174
6    5 1582           177
```

Finalmente, podemos agrupar criterios:

```
> Orange %>% filter((150<circumference & circumference<200)|Tree==1)
```

```
   Tree  age circumference
1     1  118            30
2     1  484            58
3     1  664            87
4     1 1004           115
5     1 1231           120
6     1 1372           142
7     1 1582           145
8     2 1004           156
9     2 1231           172
10    4 1004           167
11    4 1231           179
12    5 1372           174
13    5 1582           177
```

En este último ejemplo se seleccionan los casos que cumplen alguna de las siguientes condiciones: (i) valores de circumferencia entre 150 y 200, o bien (2) son del árbol 1.

2.5.3 Ordenar variables: arrange

Podemos utilizar la función **arrange** para ordenar datos en función de los valores de una variable. Por ejemplo:

```
> d <- Orange %>% arrange(age,circumference)
> d[1:14,]
```

```
   Tree age circumference
1     1 118            30
2     3 118            30
3     5 118            30
4     4 118            32
5     2 118            33
```

```
6     5 484            49
7     3 484            51
8     1 484            58
9     4 484            62
10    2 484            69
11    3 664            75
12    5 664            81
13    1 664            87
14    2 664           111
```

En el siguiente ejemplo, ordenamos primero por Sex, después por W.Hnd, etc.

```
> d <- survey %>% arrange(Sex,W.Hnd,Fold,Smoke,Wr.Hnd)
> d[107:126,]
```

```
       Sex Wr.Hnd NW.Hnd W.Hnd    Fold Pulse    Clap Exer Smoke
107 Female   19.5   19.2 Right  R on L    70   Right Some Never
108 Female   20.8   20.7 Right  R on L    NA Neither Freq Never
109 Female   14.0   13.5 Right  R on L    87 Neither Freq Occas
110 Female   16.2   16.4 Right  R on L    NA   Right Freq Occas
111 Female   16.2   15.8 Right  R on L    61   Right Some Occas
112 Female   16.5   16.9 Right  R on L    60 Neither Freq Occas
113 Female   17.5   17.0 Right  R on L    83 Neither Freq Occas
114 Female   18.0   17.6 Right  R on L    60   Right Some Occas
115 Female   18.5   18.0 Right  R on L    76   Right None Occas
116 Female   19.1   19.0 Right  R on L    80   Right Some Occas
117 Female   20.5   20.5 Right  R on L    NA    Left Freq Regul
118 Female   13.0   13.0  <NA>  L on R    70    Left Freq Never
119   Male   23.0   22.0  Left  L on R    83    Left Some Heavy
120   Male   17.5   17.0  Left  L on R    97 Neither None Never
121   Male   19.8   20.0  Left  L on R    59   Right Freq Never
122   Male   20.5   19.5  Left  L on R    80   Right Some Occas
123   Male   20.6   21.0  Left  L on R    NA    Left Freq Occas
124   Male   18.1   18.2  Left Neither    NA   Right Some Never
125   Male   19.4   19.2  Left  R on L    74   Right Some Never
126   Male   19.5   19.5  Left  R on L    66    Left Some Never
    Height      M.I  Age
107    170   Metric 18.2
108    172   Metric 18.5
109    165 Imperial 17.1
110    172   Metric 17.0
111    167   Metric 19.2
112    169   Metric 29.1
113    168   Metric 17.1
114    168   Metric 18.4
115     NA     <NA> 41.6
116    170   Metric 19.2
117     NA     <NA> 19.2
118    180 Imperial 17.4
119    193 Imperial 18.9
```

```
120    165   Metric 19.5
121    180   Metric 17.4
122    183 Imperial 18.7
123    175 Imperial 18.4
124    168   Metric 21.2
125    183 Imperial 18.3
126     NA     <NA> 16.8
```

2.5.4 Selección de variables: select

Podemos seleccionar alguna variables de una base de datos con la funcion select[4]:

```
> d <- survey %>% dplyr::select(Sex,W.Hnd,Fold)
> head(d)
```

```
     Sex W.Hnd    Fold
1 Female Right  R on L
2   Male  Left  R on L
3   Male Right  L on R
4   Male Right  R on L
5   Male Right Neither
6 Female Right  L on R
```

2.5.5 Selección de casos: slice

En algunos análisis, puede ser conveniente seleccionar unos casos concretos por sus índices. Por ejemplo, si queremos los casos del 1 al 4, del 12 al 17, el caso 20,23, y el último de los obtenidos en el ejemplo de la sección anterior podemos hacer:

```
> d %>% slice(1:4,12:17,c(20,23),n())
```

```
      Sex W.Hnd   Fold
1  Female Right R on L
2    Male  Left R on L
3    Male Right L on R
4    Male Right R on L
5    Male Right R on L
6  Female Right L on R
7  Female Right L on R
8    Male Right R on L
9  Female Right R on L
10 Female Right L on R
11   Male Right R on L
12   Male Right L on R
13 Female Right R on L
```

[4]la función select aparece en diversos paquetes. Para indicar que utilizamos la función de dlpy debemos indicarla como dplyr::select

2.5.6 Añadir variables: mutate

Vamos a ver ahora cómo podemos añadir variables.En el ejemplo con los datos **Orange**, vamos a agruparlos por age y añadiremos la media por cada edad y cómo se desvia cada árbol de la media. Primero, ordenaremos por age y a continuación añadimos la media del grupo y calculamos la diferencia de cada árbol respecto de la media del grupo:

```
> Orange %>% arrange(age) %>% group_by(age) %>%
+   mutate(m=mean(circumference),dif=circumference-mean(circumference))
```

```
# A tibble: 35 x 5
# Groups:   age [7]
   Tree    age circumference     m   dif
   <ord> <dbl>         <dbl> <dbl> <dbl>
 1 1       118            30  31   -1
 2 2       118            33  31    2
 3 3       118            30  31   -1
 4 4       118            32  31    1
 5 5       118            30  31   -1
 6 1       484            58  57.8  0.2
 7 2       484            69  57.8 11.2
 8 3       484            51  57.8 -6.80
 9 4       484            62  57.8  4.2
10 5       484            49  57.8 -8.80
# ... with 25 more rows
```

La diferencia entre **sumarise** y **mutate** es que la primera función crea una nueva tabla sólo con los cálculos indicados. La segunda función añade dichos cálculos a la tabla original. Poemos hacer todo tipo de cálculos. Por ejemplo, vamos a calcular el radio, el diametro y la superficie de cada tronco:

```
> d <- Orange %>% mutate(radio=round(circumference/(2*pi),2),
+                   diametro=round(radio*2,2),
+                   superficie=round(pi*radio*radio,2))
> head(d)
```

```
  Tree  age circumference radio diametro superficie
1    1  118            30  4.77     9.54       71.5
2    1  484            58  9.23    18.46      267.6
3    1  664            87 13.85    27.70      602.6
4    1 1004           115 18.30    36.60     1052.1
5    1 1231           120 19.10    38.20     1146.1
6    1 1372           142 22.60    45.20     1604.6
```

2.5.7 Ejemplos

Base de datos fat: Definir subgrupos de BMI y edad

Definiremos 5 grupos de BMI (nueva variable BMI) y selecionaremos las variable BMI y gBMI

```
> data(fat)
> fat %>% mutate(gBMI=cut_interval(BMI,n=5)) %>%
+   dplyr::select(BMI,gBMI) %>% head()
```

```
   BMI        gBMI
1 23.7 [18.1,24.3]
2 23.4 [18.1,24.3]
3 24.7 (24.3,30.4]
4 24.9 (24.3,30.4]
5 25.6 (24.3,30.4]
6 26.5 (24.3,30.4]
```

Ahora definiremos 5 grupos de BMI y 5 de edad y obtendremos una tabla:

```
> d <- fat %>% mutate(gBMI=cut_interval(BMI,n=5),
+                     gage=cut_interval(age,n=5))
> with(d, table(gBMI,gage))
```

```
             gage
gBMI          [22,33.8] (33.8,45.6] (45.6,57.4] (57.4,69.2] (69.2,81]
  [18.1,24.3]        23          35          33           9         2
  (24.3,30.4]        24          47          36          16         7
  (30.4,36.6]         2           6           4           5         0
  (36.6,42.7]         0           1           1           0         0
  (42.7,48.9]         0           0           1           0         0
```

Podemos hacer que el resultado sea un poco más elegante. Pondremos nombres a cada intervalo definido de acuerdo a un criterio:

```
> d <- fat %>% mutate(gBMI=cut(BMI,breaks=c(0,18.5,25,30,200),
+                              labels=c('Underweight','Normal',
+                                       'Overwight','Obese')),
+                     gage=cut(age,breaks=c(0,30,50,70,200),
+                              labels=c('Under 30','30-50',
+                                       '50-70','Over 70'))
+                     )
> with(d, table(gBMI,gage))
```

```
             gage
gBMI          Under 30 30-50 50-70 Over 70
  Underweight        0     1     0       0
```

```
  Normal            23    69    30        3
  Overweight        12    56    31        3
  Obese              3    12     8        1
```

La tabla procesada en latex produce:

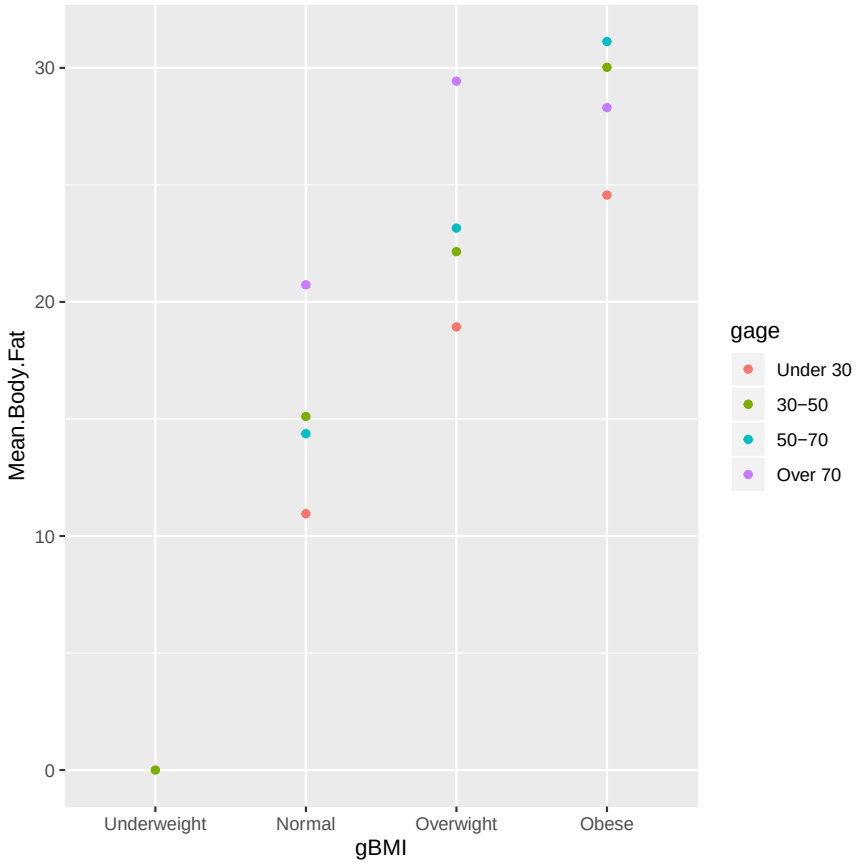
	Under 30	30-50	50-70	Over 70
Underweight	0	1	0	0
Normal	23	69	30	3
Overwight	12	56	31	3
Obese	3	12	8	1

Gráficas para grupos de edad y BMI

```
> d.new <- d %>% group_by(gage,gBMI) %>%
+   summarise(n=n(),
+             Mean.Body.Fat=mean(body.fat),
+             Mean.Neck=mean(neck))
> d.new
```

```
# A tibble: 13 x 5
# Groups:   gage [4]
   gage     gBMI            n Mean.Body.Fat Mean.Neck
   <fct>    <fct>       <int>         <dbl>     <dbl>
 1 Under 30 Normal         23          11.0      36.5
 2 Under 30 Overwight      12          18.9      39.0
 3 Under 30 Obese           3          24.6      40.3
 4 30-50    Underweight     1           0        33.8
 5 30-50    Normal         69          15.1      36.3
 6 30-50    Overwight      56          22.1      39.2
 7 30-50    Obese          12          30.0      41.8
 8 50-70    Normal         30          14.4      36.9
 9 50-70    Overwight      31          23.2      38.9
10 50-70    Obese           8          31.1      40.9
11 Over 70  Normal          3          20.7      38
12 Over 70  Overwight       3          29.4      40.2
13 Over 70  Obese           1          28.3      38.9
```

```
> ggplot(d.new,aes(x=gBMI,y=Mean.Body.Fat,color=gage))+geom_point()
```


30
20
10
0
Mean.Body.Fat
Underweight
Normal
Overwight
Obese
gBMI
gage
Under 30
30−50
50−70
Over 70

Índice