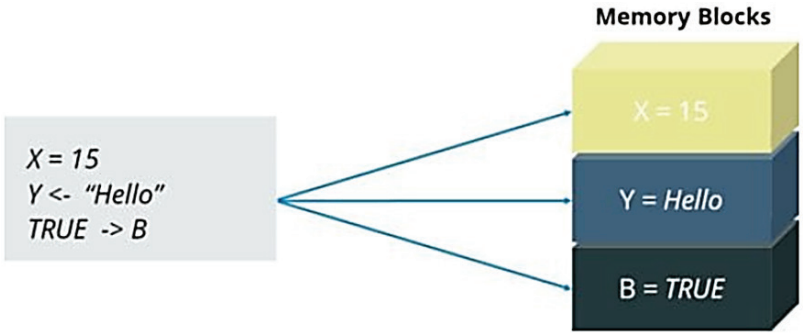
4

VARIABLES, MANEJO Y LIMPIEZA DE DATOS Y BASES DE DATOS

4.1 VARIABLES

Las ***variables*** son componentes fundamentales de la investigación que permiten medir y analizar datos. Pueden definirse como características o propiedades que pueden adoptar distintos valores. En el diseño de una investigación, comprender los tipos de variables y sus funciones es crucial para elaborar hipótesis, diseño de métodos e interpretación de resultados.

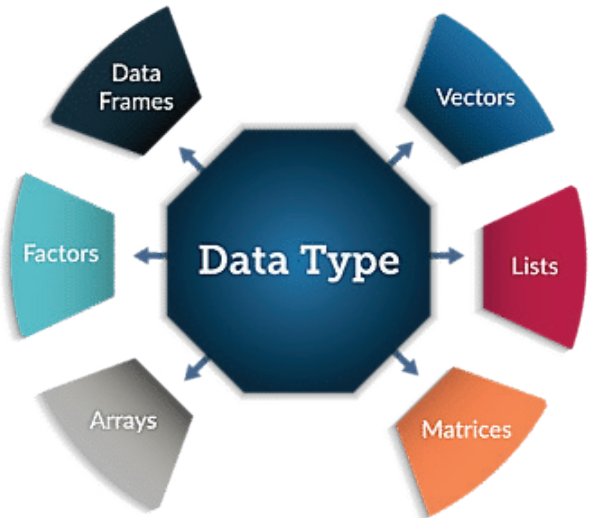
Una variable en R puede almacenar valores numéricos, valores complejos, palabras, matrices e incluso una tabla.



La imagen muestra cómo crear variables y cómo almacenarlas en diferentes bloques de memoria. En R, no tenemos que declarar una variable antes de usarla, a diferencia de otros lenguajes de programación como Java, C, C++, etc.

4.2 TIPOS DE DATOS EN R

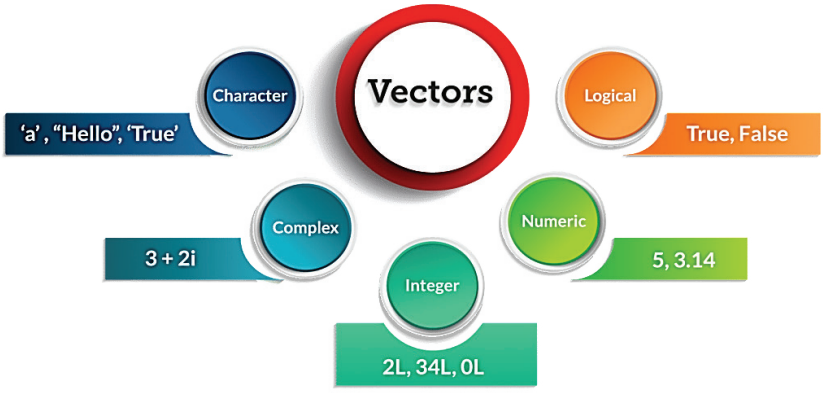
Los ***tipos de datos*** se utilizan para almacenar información. En R, no necesitamos declarar una variable como un tipo de datos. Las variables se asignan con R-Objects y el tipo de datos del R-object se convierte en el tipo de datos de la variable. Hay principalmente seis tipos de datos presentes en R:



- **Vector**: un vector es una secuencia de elementos de datos del mismo tipo básico.

```
> V1 =(1,3,5,7,9)     o     V1 <-(1,3,5,7,9)
```

Hay 5 clases de vectores:



- **List:** las listas son objetos de R que contienen elementos de diferentes tipos como números, cadenas, vectores y otra lista dentro de ella.

```
> n = c(2, 3, 5)
> s = c("aa", "bb", "cc", "dd", "ee")
> x = list(n, s, TRUE)
> x
# Output
  [[1]]
  [1] 2 3 5
  [[2]]
  [1] "aa"  "bb"  "cc" "dd" "ee"
  [[3]]
  [1] TRUE

# Función which()

> which.min(x)  # Posición del valor mínimo en el vector
> which.max(x)  # Posición del valor máximo en el vector
```

 # Ilustración

```
> mvec <- c(5, 3, 2, 1, 2, 0, NA, 0, 9, 6)
```

Posición del valor mínimo '0' en el vector.

```
> which.min(mvec)
[1] 6
```

Posición del valor máximo '9' en el vector.

```
> which.max(mvec)
[1] 9
```

 # Matrices: las matrices son objetos de R en los que los elementos están dispuestos en una disposición rectangular bidimensional. Una matriz se crea con la función matrix(). Sintaxis: **matrix(data, nrow, ncol, byrow, dimnames)**, donde 'data' es el vector de entrada o input que se convierte en los elementos de la matriz.

Generación de una matriz de 4 filas y 4 columnas.

```
> matrix(data=1:16, nrow=4, ncol=4, byrow=F)
      [,1]  [,2]  [,3]  [,4]
[1,]    1     5     9    13
[2,]    2     6    10    14
[3,]    3     7    11    15
[4,]    4     8    12    16
```

 # Arrays o matriz multidimensional: los arrays son objetos de R que pueden almacenar datos en más de dos dimensiones. Toma vectores como entrada o inputs y utiliza los valores del argumento **dim** para crear un array.

```
# Generación de una matriz multidimensional.
> vector1 <- c(5,9,3)
> vector2 <- c(10,11,12,13,14,15)
> result <- array(c(vector1,vector2),dim = c(3,3,2))
# Output
, , 1
     [,1] [,2] [,3]
[1,]    5   10   13
[2,]    9   11   14
[3,]    3   12   15

, , 2
     [,1] [,2] [,3]
[1,]    5   10   13
[2,]    9   11   14
[3,]    3   12   15
```

Factores

Los factores son objetos de R que se utilizan para categorizar los datos y almacenarlos como niveles. Pueden almacenar tanto cadenas como enteros. Son útiles en el análisis de datos y en la modelización estadística.

El término factor se refiere a un tipo de variable, categórica o cualitativa, que almacenan datos estadísticos por niveles. Por definición, las variables categóricas cuentan con un número limitado de categorías. Un factor es una variable categórica con un número finito de niveles. En R los factores se utilizan habitualmente para realizar clasificaciones de los datos, estableciendo su pertenencia a los grupos o categorías determinados por los niveles del factor.

Los niveles de un factor pueden estar codificados como valores numéricos o como caracteres. Independientemente de que el factor sea numérico o carácter, sus valores son siempre almacenados internamente por R como números enteros, con lo que se consigue economizar memoria.

Por ejemplo, la variable sexo solo tendría dos factores o niveles: hombre o mujer. Pero estos factores o categorías también pueden ser numéricos, como el número de cilindros del motor de un vehículo. Incluso, se podrían codificar los dos factores de sexo como 0 y 1.

Los factores, que pueden ser ordenados o no ordenados, se utilizan para representar variables de naturaleza categórica.

```
# Ordenar los factores por orden alfabético.
> factor_nominal <- factor(rep(c("Ford","Seat","Renault"),10))
levels(factor_nominal)
 [1] "Ford"    "Renault" "Seat"

# Reordenación de factores.
> nuevo_factor_nominal <- factor(factor_nominal, levels=c("Seat","Renau
lt","Ford"))
> levels(nuevo_factor_nominal)
 [1] "Seat"    "Renault" "Ford"
```

Ilustración

```
# Vamos a cargar la base de datos iris, que se encuentra en el paquete datasets().
Iris contiene información sobre longitud y anchura de pétalos y sépalos y especies
de un total de 150 lirios.
data("iris")
str(iris)
'data.frame':    150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width: num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width: num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species: Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1
1 1 1 ...
```

Como podemos ver, la columna especies (*Species*) es una variable categórica (o factor) que tiene tres niveles (*levels*): setosa, versicolor, virginica. Vamos a ver la distribución del tipo de especie con una tabla.

```
# Visualización de los niveles.
>iris1<-factor(iris$Species
> head(iris1)
[1] setosa setosa setosa setosa setosa setosa
Levels: setosa versicolor virginica

> levels(iris$Species)
 [1] "setosa"     "versicolor" "virginica"

> table(iris$Species)
     setosa versicolor  virginica
         50         50         50
```

En ocasiones, cuando cargamos variables que son carácter se crean como factores. Por ejemplo, si vamos a realizar un análisis de regresión es conveniente guardar las variables categóricas como factores (R codificará internamente los distintos niveles del factor como enteros). Además, puede que sea de nuestro interés cambiar el orden de los niveles.

4.3 TIPOS DE VARIABLES CATEGÓRICAS O FACTORIALES EN R

Podemos distinguir 2 tipos de variables factoriales en R:

1. Variables categóricas ***nominales***. No implica ningún orden. Por ejemplo, el sexo.
2. Variables categóricas ***ordinales***. Implica orden o gradación. Por ejemplo, la altura de una persona podría categorizarse, en este orden: Bajo, Medio, Alto.
3. En el contexto de R, los datos nominales se tipifican como carácter y las ordinales se tipifican como factor.

En R las variables de tipo Factor son las ideales a la hora de trabajar con variables de tipo nominal u ordinal. Esto se debe a dos motivos: en primer lugar, su compatibilidad con librerías como **`sjPlot`**, **`sjmisc`** o **`ggplot2`**, que nos permiten presentar de manera cómoda estos datos; así como su compatibilidad con funciones dedicadas a la generación de distintos modelos estadísticos, como modelos de análisis de clases latentes o de regresión. En segundo término, porque cabe la posibilidad de establecer un orden (distinto al alfabético) entre las categorías de la variable, lo cual es fundamental si trabajamos con variables ordinales, como el nivel educacional de las/os informantes.

Variables categóricas nominales

La escala nominal es el nivel más elemental de medición y consiste en clasificar los objetos de estudio según las categorías de una variable. El alcance de definición y medición de variables de esta escala es el conteo, que permite la aplicación de técnicas estadísticas como la distribución de frecuencia y la moda. Para la elaboración de esta escala se determinan las categorías de la variable donde el valor numérico a las categorías de una variable se le denomina codificar con el objetivo de categorizar e identificar.

Consideremos el siguiente ejemplo de variable nominal. La variable sexo.

```
> sexo <- c("M","V","M","M","M","V","M","M","V","V")
> sexo

 [1] "M" "V" "M" "M" "M" "V" "M" "M" "V" "V"
```

La variable sexo puede ser considerada un factor porque cada sujeto pertenece a una de las dos categorías definidas por la variable: "Varón "o "Mujer".

Para que R reconozca la variable sexo como factor, una vez introducidos los datos, utilizamos la función:

```
> factor_sexo <- factor(sexo)
> factor_sexo
Output
[1] M V M M M V M M V V
Levels: V M
```

Nota

Hemos convertido la variable 'sexo' en un factor con dos niveles M y V.

En muchas situaciones, los niveles del factor son poco ilustrativos en términos de su significado. La siguiente sintaxis especifica explícitamente los niveles del factor (levels) y asigna etiquetas (labels) a cada uno de ellos:

```
> sexo <- factor(sexo, levels=c("V","M"), labels=c("Varón", "Mujer"))
> sexo
 Output
 [1] Mujer Varón Mujer Mujer Mujer Varón Mujer Mujer Varón Varón
  Levels: Hombre Mujer
```

Existen algunas funciones en R que requieren que la variable de entrada sea necesariamente un factor (aun cuando la variable esté codificada numéricamente). Para ello basta recodificar la variable original como factor. Por ejemplo, supongamos que se ha registrado la producción de tres máquinas (identificadas como 27, 32 y 55) durante cinco días sucesivos, dando como resultado los siguientes datos:

```
> produccion=c(120,100,132,112,95,164,172,183,155,176,110,
115,122,108,120)
> maquina=c(27,27,27,27,27,32,32,32,32,32,55,55,55,55,55)
> dia=c(1,2,3,4,5,1,2,3,4,5,1,2,3,4,5)
> cbind(maquina,dia,produccion)
Output
        Maquina dia  produccion
  [1,]      27   1         120
  [2,]      27   2         100
  [3,]      27   3         132
  [4,]      27   4         112
  [5,]      27   5          95
  [6,]      32   1         164
  [7,]      32   2         172
  [8,]      32   3         183
  [9,]      32   4         155
 [10,]      32   5         176
 [11,]      55   1         110
 [12,]      55   2         115
 [13,]      55   3         122
 [14,]      55   4         108
 [15,]      55   5         120
```

Si se pretende evaluar la producción de estas tres máquinas a lo largo de estos días, es evidente que sus números de identificación (27, 32 y 55) son simples

etiquetas sin que su valor intrínseco tenga ningún sentido en el problema. En este caso resulta razonable (y, como veremos, en el ajuste de modelos de análisis de la varianza es además necesario) convertir esta variable en factor. Para ello simplemente ejecutamos:

```
> maquina=factor(maquina)
> maquina
  [1] 27 27 27 27 27 32 32 32 32 32 55 55 55 55 55
  Levels: 27 32 55
```

Para identificar a qué clase pertenece la variable 'maquina' escribimos:

```
> class(maquina)
  [1] "factor"
```

Variables categóricas ordinales

En la escala ordinal se establecen categorías con dos o más niveles que implican un orden inherente entre sí. Este tipo de escala se utiliza para clasificar los objetos, hechos o fenómenos en forma jerárquica, según el grado que posea una característica determinada, sin proporcionar información sobre la magnitud de las diferencias entre los casos así clasificados. Por ejemplo: excelente, bueno, malo. La escala de medición ordinal es cuantitativa porque permite ordenar a los eventos en función de la mayor o menor posesión de un atributo o característica. Por ejemplo, si tomamos la variable peso y utilizamos las categorías de obeso, gordo, normal, bajo peso, sabremos que los obesos pesan más que todos, seguidos por los gordos y así sucesivamente. Sin embargo, no sabremos cuánto más pesan los obesos que los gordos, o los normales que los de bajo peso.

En el contexto de R añadimos dos argumentos dentro de la función factor() que indiquen que hay un orden en los factores y cuál es ese orden.

 # Ordenación de los niveles de un factor

Ejemplo 1. Supongamos que leemos la siguiente variable, que corresponde al nivel de cierto contaminante en 10 muestras de agua:

```
>    pct    <-    c("alto","bajo","bajo","medio","alto","medio","bajo",
"alto","alto","alto")
```

Si la convertimos en factor, obtenemos una tabla de frecuencias de sus valores:

```
> fpct=factor(pct)
> table(fpct)
  fpct
  alto bajo medio
    5   3    2
```

Ejemplo 2. Categorización ordinal del peso de un conjunto de niños.

```
> peso <- c("Alto", "Bajo", "Bajo", "Medio", "Bajo", "Medio", "Alto")
```

Lo convertimos en una variable categórica añadiendo en la función **factor** que hay un orden.

```
> factor_peso <- factor(peso, ordered = TRUE, levels = c("Bajo", "Medio", "Alto"))
```

Observamos que en el objeto "factor_peso" las categorías ya están ordenadas con el símbolo "<".

```
> factor_peso
[1] Alto  Bajo  Bajo  Medio Bajo  Medio Alto
Levels: Bajo < Medio < Alto
```

Ejemplo 3. Categorización ordinal de las tallas de camisetas.

```
> tallas <- c('m', 'g', 'S', 'S','m', 'M')
> tallas_factor<-factor(tallas)
> tallas_factor
[1] m g S S m M
Levels: g m M S
```

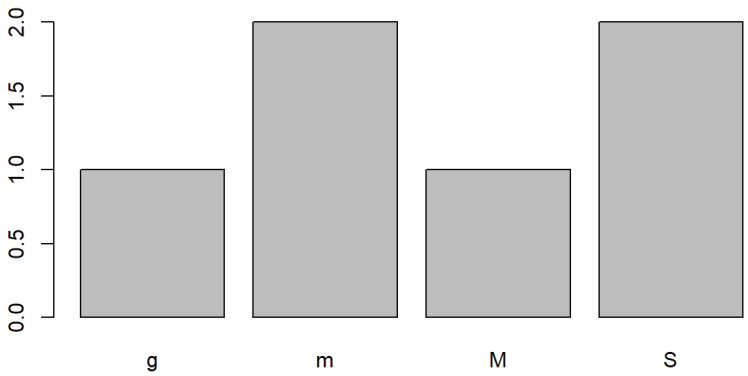
Visualización gráfica de las tallas.

```
> plot(tallas_factor)
Error in plot.new() : figure margins too large
```

Una solución posible a esta incidencia es hacer uso de la función **par()**, que determina los márgenes del plot con el argumento **mar**.

```
> par(mar=c(2,2,2,2))
> plot(tallas_factor)
```

Por defecto: Bottom margin: 5.1, Left margin: 4.1,Top margin: 4.1,Right margin: 2.1



 # Data-frame: un Data-frame de datos es una tabla o una estructura bidimensional similar a una matriz en la que cada columna es independiente de otra y contiene valores de una variable y cada fila contiene un conjunto de valores de cada columna.

```
> std_id = c (1:5)
> std_name = c("Rick","Dan","Michelle","Ryan","Gary")
> marks = c(623.3,515.2,611.0,729.0,843.25)
> std.data <- data.frame(std_id, std_name, marks)
> std.data
Output:
   std_id  std_name     marks
1   1     Rick        623.30
2   2     Dan         515.20
3   3     Michelle    611.00
4   4     Ryan        729.00
5   5     Gary        843.25
```

ⓘ Nota

Codificación de caracteres en un archivo (fileEncoding): las funciones de lectura o carga de datos incluyen como argumento la opción fileEncoding="UTF-8" para que R interprete correctamente los caracteres típicos del español: la ñ, las tíldes, diéresis, etc. Ello es debido a que el archivo de las tortugas se codificó originalmente en un sistema operativo Linux, que usa esa codificación específica de caracteres; si no se especifica "UTF8" y el archivo se lee en un sistema Windows, los caracteres citados no serían legibles. Si la lectura se hiciera en un Mac no habría problemas, ya que Mac utiliza también codificación UTF8. Si un usuario de un Mac quisiera leer archivos que contienen caracteres codificados en Windows, debería usar la opción fileEncoding="latin1".

4.4 TIPOS DE VARIABLES CUANTITATIVAS EN R

Las variables cuantitativas pueden ser de dos tipos: ***Variables cuantitativas continuas***, si admiten tomar cualquier valor dentro de un rango numérico determinado (edad, peso, talla). ***Variables cuantitativas discretas***, si no admiten todos los valores intermedios en un rango.

En R. los datos con la etiqueta de "numeric" nos indica que se trata de variables cuantitativas o nivel de medición cuantitativo. Por ejemplo, calcular la media de ingresos por sueldos y salarios.

Variables de Escala de Intervalo

La medición de intervalo posee las características de las mediciones nominal y ordinal. Establece la distancia entre una medida y otra, siendo igual a la distancia entre dos puntos o valores de un continuo. La escala de intervalo se aplica a variables continuas, pero carece de un punto cero absoluto. Sin embargo el punto cero es arbitrario y convencional, por lo que no se pueden establecer razones o proporciones, ni comparar dos escalas sin definir el mismo punto de partida. Las variables inteligencia, rendimiento académico y temperatura, son ejemplos que utilizan escalas de intervalo, debido a que el punto cero es arbitrario. Tomando la temperatura como ejemplo, se puede decir que el cero no representa la ausencia de calor, sin embargo, la distancia entre dos puntos de la escala es igual, o sea, que el cambio de temperatura entre 36°C y 37°C es igual al cambio entre 40°C y 41°C. Al medir temperatura, no se puede decir que 20°C es el doble de 10°C. Esto es debido a que cuando el termómetro marca 0 grados, en realidad la temperatura es de 273 grados. Por esto, solo podemos decir que una temperatura de 20ºC es 10 grados más que una de 10ºC.

Variables de Escala de Razón

Una escala de medición de razón incluye las características de los tres anteriores niveles de medición anteriores (nominal, ordinal e intervalo). Este tipo de escala constituye el nivel más alto de medición para las variables cuantitativas. Contiene las características de una escala de intervalo con la ventaja adicional de poseer el cero absoluto, lo cual permite determinar la proporción conocida de valores de la escala. Determina la distancia exacta entre los intervalos de una categoría. El peso, talla y número de alumnos son ejemplos de variables de razón o proporción, en las que el cero representa la nulidad o ausencia de lo que se estudia. Por esta propiedad de la escala se puede establecer razones tales como se dan en la variable peso, en la cual se dice que un peso de 50 kg es el doble que uno de 25 kg, o que uno de 100 kg es 4 veces mayor que uno de 25 kg. Con este nivel de medición también se puede decir que 100 kg es mayor que 25, o que 100 kg es 75 kg más que 25 kg. También se pueden hacer estas aseveraciones a la inversa.

El nivel de medición con que se define una variable es lo que determina posteriormente el alcance del análisis de los datos, razón por la cual, en términos generales se recomienda medir las variables al mayor nivel posible. Por ejemplo, una variable como edad, la cual es del nivel de razón o proporción, debe ser medida con una escala de este nivel. No debe bajarse a nivel ordinal agrupando los datos en clases (0<5; 5< 10; 10El nivel de medición de razón se aplica tanto a variables continuas como discretas.

En R las variables cuantitativas continuas se tipifican como numeric o doublé, mientras que las discretas se tipifican como integer.

4.5 GENERACIÓN DE DATOS, MUESTRAS Y MUESTREO

En este apartado nuestro objetivo principal es introducir al usuario en el muestreo simple y complejo en un entorno de programación en R. Los temas tratados son los siguientes:

1. La selección del tamaño de muestra simple (base y dplyr).
2. Selección sistemática (SamplinUtil).
3. Cálculo de tamaños de muestra (SamplinUtil).
4. Cálculo de intervalos de confianza (DescTools).
5. Selección de muestras con PPT (pps).
6. Muestreo complejo con el paquete survey.

Es posible realizar una comparación de los resultados arrojados en R con la información oficial de la Encuesta de Hogares del INEC.

ⓘ Nota

Hacemos uso de las funciones **`head()`** para mostrar las primeras 6 filas, **`tail()`** para mostrar las ultimas 6 filas, **`str()`** para mostrar la estructura de la base de datos.

Generación de datos

 # Función secuencia – seq()

```
> seq(from=1, to=1, by, length.out)
```

Los argumentos de esta función son:

- from: valor de inicio de la secuencia.
- to: valor de fin de la secuencia, no siempre se alcanza.
- by: incremento de la secuencia.
- length.out: longitud deseada de la secuencia.

 # Ilustración

```
> seq(from=0, to=1, length.out = 11)
[1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

> seq(from=1, to=9, by=2)  # matches 'end'
[1] 1 3 5 7 9
```

 # Función repetir – rep()

Crear repeticiones usando la función **`rep`**, la sintaxis de esta función es.

```
> rep(x, times=1, length.out=NA, each=1)
```

Los argumentos de esta función son:

- x: vector con los elementos a repetir.
- times: número de veces que el vector x se debe repetir.
- length.out: longitud deseada para el vector resultante.
- each: número de veces que cada elemento de x se debe repetir.

```
# Ilustración
> rep(x=1:4, times=2)
[1] 1 2 3 4 1 2 3 4

> rep(x=1:4, times=c(2,2,2,2))
[1] 1 1 2 2 3 3 4 4

> rep(x=1:4, each=2)
[1] 1 1 2 2 3 3 4 4

> rep(x=1:4, each=2, len=10)  # 8 integers plus two recycled 1's.
[1] 1 1 2 2 3 3 4 4 1 1

> rep(x=1:4, each=2, times=3)  # length 24, 3 complete replications.
[1] 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4
```

```
# Función sort()
> sort(x, decreasing = FALSE)
> sort(x, decreasing=TRUE)
```

```
# Ejemplo.
x <- c(2, 3, 6, 4, 9, 5)
sort(x)    # Por defecto asume que decreasing es FALSE.
[1] 2 3 4 5 6 9
```

Muestreo Aleatorio Simple

La función **sample** se utiliza para obtener una muestra aleatoria de tamaño **n** de una población, y se encuentra alojada dentro del paquete base de R. El muestreo puede ser con reemplazo y sin reemplazo. Es importante resaltar que, cada vez que llamamos a la función **sample()**, se generan muestras aleatorias diferentes. Pero, si antes fijamos una semilla de R (con **set.seed**), con un número específico como argumento, obtendremos los mismos resultados cada vez que se ejecute el comando.

 # Función Sample() de R para Generar Números Enteros

Vamos a comenzar primero generando números enteros aleatoriamente, esto es, números que no contienen decimales.

```
> sample(1:30, n=10, replace=F)
[1] 5 3 19 10 28 4 11 23 16 22
```

Los atributos de la estructura: **`sample(1:30, n=10, replace=F)`**.

- **1:30:** significa que el intervalo de datos va a estar comprendido entre ambos números (1 y 30 inclusive).
- **n = 10**: significa el número de números aleatorios que se quiere extraer del intervalo anterior.
- **replace=F/T** = si se quiere repetir los números aleatorios (sin repetición o con repetición). Para obtener, por ejemplo, dos veces el 3, dejamos el valor T (TRUE). Si no queremos repetición, ponemos el valor F (FALSE).

 # Función runif() de R para generar números racionales

Ahora generamos números racionales o números con decimales. Esto se puede conseguir con la función **`runif()`** de R, que está incluida dentro del paquete de R llamado **`stats`**.

```
> runif(5,3,4)
> runif(5, min=3, max=4)
[1] 3.537344 3.629892 3.362016 3.860888 3.930647
```

La estructura **`runif(5, min=3, max=4)`** se compone de tres atributos.

- 5 = es el número de números racionales (números con decimales) que queremos generar aleatoriamente. En este caso queremos que nos devuelva 5 números aleatoriamente seleccionados.
- min=3 y max=4 representan el intervalo de donde se van a extraer los números aleatorios en R.

En nuestro ejemplo, nuestros números aleatorios estarán comprendidos entre el 3 y el 4. Es importante mencionar que no es obligatorio poner min= y max= en esta función. Si los omitiéramos como se ve a continuación seguirían extrayéndose los números aleatorios del intervalo marcado.

La función **`sample_n`** es útil para extraer muestras de bases de datos y requiere instalar el paquete **`dplyr`**.

```
> install.packages("dplyr")
> library(dplyr)
# dplyr::simple_n
> sample_n(mtcars, 5)
                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
Ford Pantera L    15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
Merc 280C         17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Porsche 914-2     26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
```

Análogamente, extraemos una muestra de la base de datos "crime".

```
> muestra3<-crime %>% sample_n(size=n,replace=F)
> head(muestra3)
  Var1   Var2 Freq
1 10.7  152.4    4
2 11.4 160.02   29
3 10.1 175.26    0
4 10.9 144.78    0
5 13.5 142.24    0
6 10.2 157.48    0
```

Ejemplo 1. Muestra aleatoria simple (MAS) de la dataset "Estudiantes".

```
> install.packages("repmis")
> library(repmis)
> url.dat<- "http://bit.ly/Database-Estudiantes"
> datos_estudiantes <- read.delim(url.dat)
> head(datos_estudiantes)
  Observacion               ID      Sexo ……................
1           1 SB11201910010435  Femenino ……..............
2           2 SB11201910004475 Masculino ……..............
3           3 SB11201910011427 Masculino
4           4 SB11201910041975 Masculino
5           5 SB11201910013623  Femenino
6           6 SB11201910038122  Femenino ……..............
```

```
> names(datos_estudiantes)
 [1] "Observacion"  "ID"           "Sexo"
 [4] "SexoNum"      "Edad"         "Fuma"
 [7] "Estatura"     "Colegio"      "Estrato"
[10] "Financiacion" "Acumulado"    "P1"
[13] "P2"           "P3"           "Final"
[16] "Definitiva"   "Gastos"       "Ingreso"
[19] "Gas"          "Clases"       "Ley"
[22] "PandemiaCat"  "PandemiaNum"  "Likert1"
[25] "Likert2"      "Likert3"      "Likert4"
[28] "Likert5"      "AGPEQ1"       "AGPEQ2"
[31] "AGPEQ3"       "SATS1"        "SATS2"
[34] "SATS3"        "SATS4"        "IDARE1.1"
[37] "IDARE1.2"     "IDARE1.3"     "IDARE1.4"
[40] "IDARE1.5"     "IDARE2.6"     "IDARE2.7"
[43] "IDARE2.8"     "IDARE2.9"     "IDARE2.10"
[46] "Puntaje"
 "AGPEQ2"
[31] "AGPEQ3"       "SATS1"        "SATS2"
```

```
[34] "SATS3"        "SATS4"        "IDARE1.1"
[37] "IDARE1.2"     "IDARE1.3"     "IDARE1.4"
[40] "IDARE1.5"     "IDARE2.6"     "IDARE2.7"
[43] "IDARE2.8"     "IDARE2.9"     "IDARE2.10"
[46] "Puntaje"
```

 # Nombres de las variables
SexoNum, Edad, Fuma, Estatura, Colegio, Estrato, Financiacion, Acumulado, P1, P2, P3, Final, Definitiva, Gastos, Ingreso, Gas, Clases, Ley, PandemiaCat, PandemiaNum, Likert1, Likert2, Likert3, Likert4, Likert5, AGPEQ1, AGPEQ2, AGPEQ3, SATS1, SATS2, SATS3, SATS4, IDARE1.1, IDARE1.2, IDARE1.3, IDARE1.4, IDARE1.5, IDARE2.6, IDARE2.7, IDARE2.8, IDARE2.9, IDARE2.10, Puntaje.

Supongamos que necesitamos un data-frame que contenga solo las observaciones de 1 a 6, con las columnas 2 a 7. Para ello, utilizamos los corchetes [] y los dos puntos ':'.

```
> muestra1<-datos_estudiantes[1:6,2:7]
> muestra1
                ID      Sexo SexoNum  Edad Fuma Estatura
1 SB11201910010435  Femenino       0 21.36   No     Alta
2 SB11201910004475 Masculino       1 21.07   Si     Baja
3 SB11201910011427 Masculino       1 20.92   Si     Alta
4 SB11201910041975 Masculino       1 18.41   Si     Alta
5 SB11201910013623  Femenino       0 16.64   Si     Alta
6 SB11201910038122  Femenino       0 16.02   No     Baja
```

Supongamos que se desea tomar una muestra aleatoria (sin reemplazo) de tamaño n=3 del data-frame muestra1.

```
> f<-nrow(muestra1)            # Tamaño del data-frame de referencia = 6.
> n<-3                         # Tamaño de la nueva muestra aleatoria = 3.
> i<-sample(1:f,n,replace=F) # Posiciones de las observaciones aleatorias.
> muestra2<-muestra1[i,]       # Extraer la nueva muestra aleatoria.
> muestra2                     # Muestra la muestra.
                ID      Sexo SexoNum  Edad Fuma Estatura
1 SB11201910010435  Femenino       0 21.36   No     Alta
4 SB11201910041975 Masculino       1 18.41   Si     Alta
3 SB11201910011427 Masculino       1 20.92   Si     Alta
```

Supongamos que se desea tomar una muestra aleatoria (sin reemplazo) de tamaño n=2 del data-frame "muestra1", pero solo con las 3 últimas variables (Edad, Fuma y Estatura).

```
set.seed(1)                          # Fijando la semilla.
f <- nrow(muestra1)                  # Tamaño del data frame de referencia = 6.
n <- 2                               # Tamaño de la nueva muestra aleatoria = 2.
i <-sample(1:f,n,replace=FALSE)# Posiciones de las observaciones aleatorias
= 1 4.
muestra3  <- muestra1[i,4:6]         # Extraer la nueva muestra aleatoria.
muestra3                             # Mostrando esa muestra.
   Edad Fuma Estatura
1 21.36   No     Alta
2 21.07   Si     Baja
```

 # Ejemplo2. Dataset crime y MAS

```
> data("crimtab")
> crime<-data.frame(crimtab)
> dim(crime)
[1] 924   3
> head(crime,3)
  Var1   Var2 Freq
1  9.4 142.24    0
2  9.5 142.24    0
3  9.6 142.24    0
```

Seleccionamos una muestra de 30 casos sin reemplazo.
Sintaxis de la selección de la muestra y tamaño de la muestra.

```
> n<-30
> muestra1<- sample(1:nrow(crime), size=n, replace=FALSE)
> muestra1       # Nos muestra las posiciones seleccionadas.
 [1] 657 323  51 780 749 160  53
 [8] 825 423 726 908 898 372 470
[15] 425 387 834 697  37 694 488
[22] 887 559 484 492 337 687 167
[29] 753 347
```

Asignamos los elementos de la muestra al data-frame de datos.

```
> crime1<-crime[muestra1,]
> head(crime1)
    Var1   Var2 Freq
657   12 180.34    2
323 12.2 160.02    2
51  10.2 144.78    0
780 11.7 187.96    0
749 12.8 185.42    1
160 12.7 149.86    0
```

 # Selección de una MAS con la liberaría dplyr

Una forma más sencilla de obtener una muestra es con el paquete **dplyr**. Este paquete es sumamente útil para el tratamiento de datos, adicionalmente contiene una función para obtener muestras simples de un data-frame.

```
> library(dplyr)
```

Extracción de una muestra sin reemplazo.

```
> crime2<-crime %>% sample_n(size=n, replace= F)
> head(crime2)
  Var1   Var2 Freq
1 12.6 162.56    0
2 12.1 154.94    0
3 13.2 182.88    0
4 12.4  152.4    0
5 10.8  165.1    7
6 11.4 193.04    0
```

Obtención de muestras con pesos o ponderaciones.

```
> n<-30
> crime3<-crime %>%sample_n(size=n,weight=Freq)
> head(crime3)
  Var1   Var2 Freq
1 11.8 172.72   29
2 11.5 167.64   38
3 11.7 172.72   24
4   11 170.18   10
5 11.3 162.56   26
6 10.9 160.02   24
```

Obtención de muestras con una proporción definida de casos.

```
> crime4<-crime %>%sample_frac(0.05)
> head(crime4);dim(crime4)
  Var1   Var2 Freq
1 11.1 185.42    0
2  9.5  165.1    0
3 13.5 172.72    0
4 12.2  190.5    0
5   13 144.78    0
6 11.2 193.04    0
[1] 46  3
```

 # Otros procedimientos de muestreo

```
> library(magrittr)
> library(dplyr)
> install.packages("devtools")
> library(devtools)
```

Consideraremos el conjunto de datos iris —de dimensión 150 x 5— y extraeremos 60 filas con distintos procedimientos.

 # Muestreo Aleatorio Simple sin Repetición

```
> data(iris)
> indice<-sample(1:nrow(iris), 60)
> iris.muestra1<-iris[indice,]
> head(iris.muestra1, 4)
    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
131          7.4         2.8          6.1         1.9 virginica
94           5.0         2.3          3.3         1.0 versicolor
17           5.4         3.9          1.3         0.4 setosa
99           5.1         2.5          3.0         1.1 versicolor
```

 # Muestreo Aleatorio Simple con Repetición

```
> indice1 <- sample( 1:nrow( iris ), 60, replace = TRUE )
> iris.muestra2<-iris[indice1,]
> head(iris.muestra2,4)
    Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
40           5.1         3.4          1.5         0.2    setosa
131          7.4         2.8          6.1         1.9 virginica
111          6.5         3.2          5.1         2.0 virginica
24           5.1         3.3          1.7         0.5    setosa
```

Muestreo Estratificado con o sin Reemplazo

La manera más sencilla de obtenerlos consiste en usar el paquete sampling.

```
> install.packages("sampling")
> library(sampling)
```

Consideraremos el conjunto de datos iris —de dimensión 150 x 5— y extraeremos 60 filas.

Muestreo sin reemplazamiento y estratificado respecto a iris$Species —que es un factor con tres niveles de 50 elementos cada uno.

```
> library(sampling)
>  stratos<-strata(iris,  stratanames=c("Species"),  size=  c(20,20,20),
method="srswor")
> iris.muestra3<-getdata(iris,stratos)
> head(iris.muestra3)
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species ID_unit Prob
Stratum
6     5.4         3.9       1.7         0.4    setosa    6    0.4    1
7     4.6         3.4       1.4         0.3    setosa    7    0.4    1
9     4.4         2.9       1.4         0.2    setosa    9    0.4    1
15    5.8         4.0       1.2         0.2    setosa   15    0.4    1
16    5.7         4.4       1.5         0.4    setosa   16    0.4    1
19    5.7         3.8       1.7         0.3    setosa   19    0.4    1
```

 # Muestreo con reemplazamiento y estratificado respecto a iris$Species.

```
> stratos1<-strata(iris, stratanames=c("Species"),size= c(20,20,20),
method="srswr")
> iris.muestra4<-getdata(iris, stratos1)
> head(iris.muestra4)
    Sepal.Length Sepal.Width Petal.Length Petal.Width Species ID_unit     Prob Stratum
4            4.6         3.1          1.5         0.2  setosa       4 0.332392       1
4.1          4.6         3.1          1.5         0.2  setosa       4 0.332392       1
5            5.0         3.6          1.4         0.2  setosa       5 0.332392       1
10           4.9         3.1          1.5         0.1  setosa      10 0.332392       1
10.1         4.9         3.1          1.5         0.1  setosa      10 0.332392       1
17           5.4         3.9          1.3         0.4  setosa      17 0.332392       1

> summary(iris.muestreado3)
 Sepal.Length   Sepal.Width    Petal.Length   Petal.Width          Species
 Min.   :4.60   Min.   :2.200   Min.   :1.000   Min.   :0.100   setosa    :20
 1st Qu.:5.10   1st Qu.:2.975   1st Qu.:1.575   1st Qu.:0.300   versicolor:20
 Median :5.75   Median :3.100   Median :4.500   Median :1.500   virginica
:20
 Mean   :5.89   Mean   :3.097   Mean   :3.792   Mean   :1.235
 3rd Qu.:6.50   3rd Qu.:3.325   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.90   Max.   :4.200   Max.   :6.600   Max.   :2.500
    ID_unit            Prob           Stratum
 Min.   :  4.00   Min.   :0.3324   Min.   :1
 1st Qu.: 38.50   1st Qu.:0.3324   1st Qu.:1
 Median : 72.00   Median :0.3324   Median :2
 Mean   : 74.37   Mean   :0.3324   Mean   :2
 3rd Qu.:107.75   3rd Qu.:0.3324   3rd Qu.:3
 Max.   :147.00   Max.   :0.3324   Max.   :3
```

 # Observación

```
iris.df <- data.frame(iris)
```

Extracción del 75% de las observaciones de la base de datos iris.

```
sample.index <- sample(1:nrow(iris.df), nrow(iris) * 0.75, replace =
FALSE)
head(iris[sample.index, ])
```

Probemos ahora el muestreo estratificado haciendo uso del paquete **splitstackshape** y la función **stratified()**.

```
library('MASS')
```

```
library(splitstackshape)
> dfiris<-data.frame(iris)
```

Muestreo estratificado centrado en los estratos de Sepal.Length. Luego, el código solicita una muestra del 70%.

```
> summary(stratified(dfiris,"Sepal.Length",0.7))
  Sepal.Length    Sepal.Width    Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.00   Min.   :1.000   Min.   :0.100   setosa    :37
 1st Qu.:5.100   1st Qu.:2.80   1st Qu.:1.600   1st Qu.:0.300   versicolor:35
 Median :5.800   Median :3.00   Median :4.300   Median :1.300   virginica
:36
 Mean   :5.867   Mean   :3.07   Mean   :3.762   Mean   :1.194
 3rd Qu.:6.425   3rd Qu.:3.40   3rd Qu.:5.125   3rd Qu.:1.825
 Max.   :7.900   Max.   :4.40   Max.   :6.900   Max.   :2.500
```

Muestreo Sistemático

Para el ejemplo del muestreo sistemático utilizaremos la función **sys.sample** del paquete **SamplingUtil()**. Para instalar este paquete, se debe inicialmente instalar el paquete **devtools()** ya que no se encuentra en CRAN. La sintaxis es la siguiente:

Instalar devtools: **install.packages("devtools")**

Cargar librería: **library(devtools)**

Instalar SamplingUtils: **install_github("DFJL/SamplingUtil")**

El ***muestreo sistemático*** se puede escribir como una función simple que seleccione cada k-ésima fila secuencialmente dado un número de inicialización aleatorio:

```
> sys.sample = function(N, n) {
k = ceiling(N/n)
r = sample(1:k, 1)
sys.samp = seq(r, r + k * (n - 1), k)
}
> systematic.index <- sys.sample(nrow(iris), nrow(iris) * 0.75)
> summary(iris[systematic.index, ])
```

El código define la función de muestreo sistemático y luego la ejecuta sobre los datos de iris.

```
> sys.sample<-function(N,n){
+ k=ceiling(N/n)
+ r=sample(1:k,1)
+ sys.samp=seq(r,r+k*(n-1),k)
+ }
```

```
> systematic.index<-sys.sample(nrow(iris),nrow(iris)*0.75)
> summary(iris[systematic.index,])
 Sepal.Length   Sepal.Width    Petal.Length   Petal.Width          Species
 Min.   :4.40   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :25
 1st Qu.:5.15   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.250   versicolor:25
 Median :5.80   Median :3.000   Median :4.500   Median :1.300   virginica
:25
 Mean   :5.84   Mean   :3.064   Mean   :3.776   Mean   :1.219   NA's      :37
 3rd Qu.:6.45   3rd Qu.:3.400   3rd Qu.:5.100   3rd Qu.:1.850
 Max.   :7.70   Max.   :4.100   Max.   :6.900   Max.   :2.500
 NA's   :37     NA's   :37      NA's   :37      NA's   :37

> library(SamplingUtil)
> msistematic<-sys.sample(N=nrow(crime),n=30)
> msistematic
 [1]   5  35  65  95 125 155 185 215 245 275 305
[12] 335 365 395 425 455 485 515 545 575 605 635
[23] 665 695 725 755 785 815 845 875
```

Asignar los elementos de la muestra al data.frame.

```
> crimsis<-crime[msistematica,]
> head(crimsis)
    Var1   Var2 Freq
5    9.8 142.24    0
35  12.8 142.24    0
65  11.6 144.78    0
95  10.4 147.32    1
125 13.4 147.32    0
155 12.2 149.86    0
```

 # Selección de una submuestra de datos (Subsetting)

Vamos a aprovechar que hemos cargado los datos de *airquality* para recordar algunas ideas sobre la selección de datos (observaciones y/o variables) en un data frame e introducir algunas otras.

Cargamos la base de datos y visualizamos su estructura.

```
> dairq<-airquality
> str(dairq)
Observations: 153 , Variables: 6
$ Ozone   <int> 41, 36, 12, 18, NA, 28, 23, 19, 8, NA, 7, 16, 11, 14, ...
$ Solar.R <int> 190, 118, 149, 313, NA, NA, 299, 99, 19, 194, NA, 256,...
$ Wind    <dbl> 7.4, 8.0, 12.6, 11.5, 14.3, 14.9, 8.6, 13.8, 20.1, 8.6...
$ Temp    <int> 67, 72, 74, 62, 56, 66, 65, 59, 61, 69, 74, 69, 66, 68...
$ Month   <int> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, ...
$ Day     <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,...
```

Nota

num = **dbl** = variable numérica con decimal, **int** = variable numerica, **entero** = variable lógica (Booleano)

Medición diaria de la calidad del aire en New York de mayo a septiembre de 1973

 # Variables:

A data frame with 153 observations on 6 variables.

- Ozone : numeric Ozone (ppb)
- Solar.R : numeric Solar R (lang)
- Wind : numeric Wind (mph)
- Temp : numeric Temperature (degrees F)
- Month : numeric Month (1–12)
- Day : numeric Day of month (1–31)

 # Detalles

- Ozone: ozono medio en partes por mil millones entre las 13:00 y las 15:00 horas en Roosevelt Island.
- Solar.R: radiación solar en Langleys en la banda de frecuencia 4000-7700 Angstroms de 08:00 a 12:00 horas en Central Park.
- Wind: velocidad media del viento en millas por hora a las 07:00 y a las 10:00 horas en el Aeropuerto LaGuardia.
- Temp: temperatura máxima diaria en grados Fahrenheit en el Aeropuerto La Guardia.

Fuente

Chambers, J. M., Cleveland, W. S., Kleiner, B. and Tukey, P. A. (1983) Graphical Methods for Data Analysis. Belmont, CA: Wadsworth.

En primer lugar, seleccionamos las variables de interés: *Ozone*, *Solar.R* y *Wind*.

```
> datos1 <- dairq[,1:3]
```

Análogamente, es posible seleccionar del objeto dairq las variables: *Ozone, Temp* y Month, con la siguiente sintaxis:

```
> datos2 <- dairq[,c(1,4,5)]
```

Si en lugar de seleccionar variables (columnas) estamos interesados en seleccionar individuos/observaciones (filas):

```
> datos3 <- dairq[1:6,]
```

```
# ¿Qué observaciones son las seleccionadas?
> datos4 <- dairq[seq(1,nrow(dairq),5),]

# Visualización de los datos.
> datos4
    Ozone Solar.R Wind Temp Month Day
1      41     190  7.4   67     5   1
6      28      NA 14.9   66     5   6
11      7      NA  6.9   74     5  11
16     14     334 11.5   64     5  16
21      1       8  9.7   59     5  21
26     NA     266 14.9   58     5  26
31     37     279  7.4   76     5  31
36     NA     220  8.6   85     6   5
41     39     323 11.5   87     6  10
```

Extraemos las observaciones en intervalos de 5 posiciones. Por ejemplo, extraemos la observación 1 y luego la observación 6 (1 +5 = 6) y a continuación la 11 (6+5 = 11), y así sucesivamente.

```
# Para seleccionar tanto observaciones como variables no tenemos más que
combinar las estrategias anteriores:
> datos5 <- dairq[seq(1,nrow(dairq),5), c(1,2,4)]

# En ocasiones estamos interesados en seleccionar los casos para los que cierta
variable toma determinado valor. Por ejemplo, si queremos seleccionar los valores
de las variables Ozone y Temp para todas las observaciones en las que la variable
Wind satisfaga un valor específico:
> datos6 <- dairq[dairq$Wind<=4, c(1,2)]
> datos7 <- dairq[dairq$Wind>=2 & datos$Wind<=5.1, c(1,2)]
> datos8 <- dairq[dairq$Wind==4 , c(1,2)]

> datos8
   Ozone Solar.R
99   122     255

# Para seleccionar subconjuntos de datos en un data frame también podemos
utilizar la función subset().
>datos9<-subset(dairq,Month==5&Day<=15,select=c(Ozone,Solar.R, Temp))
> head(datos9)
  Ozone Solar.R Temp
1    41     190   67
2    36     118   72
3    12     149   74
4    18     313   62
5    NA      NA   56
6    28      NA   66
```

```
> datos10 <- subset(dairq, Month !=5 & Day <=15)
> head(datos10)
   Ozone Solar.R Wind Temp Month Day
32    NA     286  8.6   78     6   1
33    NA     287  9.7   74     6   2
34    NA     242 16.1   67     6   3
35    NA     186  9.2   84     6   4
36    NA     220  8.6   85     6   5
  NA     264 14.3   79     6   6
```

Muestreo de bola de nieve

El muestreo bola de nieve es una técnica no probabilística donde los participantes iniciales, seleccionados por cumplir criterios específicos, recomiendan a otros miembros de la población, expandiendo la muestra de forma gradual, como una "bola de nieve" rodando cuesta abajo.

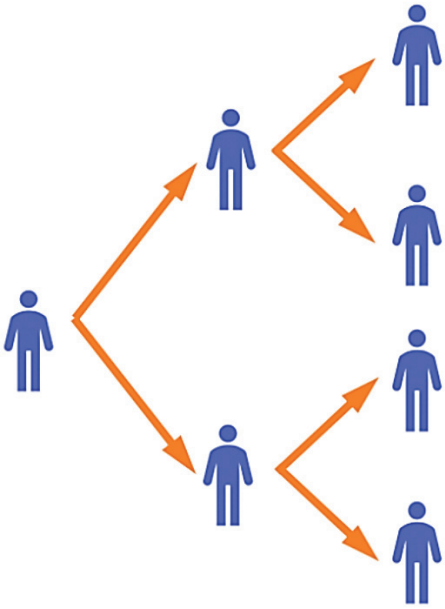
Se aplica en poblaciones difíciles de encontrar o de acceso limitado, como grupos ocultos o especializados, debido a que se basa en las redes de contactos de los participantes. Aunque, permite acceder a este tipo de poblaciones, presenta limitación de subjetividad (la selección de participantes no es aleatoria, lo que significa que la muestra no es representativa de toda la población) y de posibles sesgos (al depender de las redes sociales y de la voluntad de los participantes, el muestreo bola de nieve puede introducir sesgos).

Este método es común en estudios cualitativos y exploratorios, ya que permite llegar a individuos específicos en situaciones donde es difícil construir un marco muestral (No tiene fórmula).

 # Tipos de muestreo de bola de nieve

1. Muestreo de Bola de Nieve Lineal. Los participantes identifican a un solo individuo adicional por etapa.

2. Muestreo de Bola de Nieve Exponencial. Los participantes identifican a más de una persona adicional, lo que permite un crecimiento más rápido de la muestra.



3. Muestreo de Bola de Nieve Controlado. Los investigadores limitan el número de referencias por persona o el número de etapas, para controlar el tamaño de la muestra.

```
# Lista inicial de individuos para el muestreo
> personas_iniciales <- c("Persona1", "Persona2")

# Función para expandir la muestra mediante referencias
> bola_de_nieve <- function(personas, num_iteraciones) {
  muestra <- personas
  for (i in 1:num_iteraciones) {
   nuevos_participantes <- sapply(muestra, function(x) paste(x, "_Ref",
                           i, sep = ""))
  muestra <- unique(c(muestra, nuevos_participantes))
  }
  return(muestra)
  }

# Ejemplo de uso.
> muestra <- bola_de_nieve(personas_iniciales, 3)
> print(muestra)
  [1] "Persona1"                "Persona2"
  [5] "Persona1_Ref2"           "Persona2_Ref2"
  [7] "Persona1_Ref1_Ref2"      "Persona2_Ref1_Ref2"
  [9] "Persona1_Ref3"           "Persona2_Ref3"
  [11] "Persona1_Ref1_Ref3"     "Persona2_Ref1_Ref3"
  [13] "Persona1_Ref2_Ref3"     "Persona2_Ref2_Ref3"
```

```
[15] "Persona1_Ref1_Ref2_Ref3" "Persona2_Ref1_Ref2_Ref3"
```

En este ejemplo, `num_iteraciones` determina cuántas "etapas" tiene el muestreo, simulando cómo una persona en la muestra inicial podría identificar a otros. El resultado es una muestra que crece en cada etapa, como ocurre en el muestreo de bola de nieve.

4.6 TAMAÑOS DE MUESTRA

Tamaño de muestra 'n' de proporciones como MAS

Consideremos primero el caso de calcular una muestra de tamaño 'n' de una población de N sujetos en base al Muestreo Aleatorio Simple (MAS), para lo cual se usa la ecuación del tamaño de la muestra de proporciones que se obtiene fijando el error máximo admisible y el nivel de confianza asociado a la estimación. Según algunos autores como Vivanco (2005, p.77), "el tamaño de la muestra de proporciones presenta las mismas características analíticas que el tamaño de muestra de medias".

$$n = \frac{Z^2_{\alpha/2} Npq}{e^2(N-1) + Z^2_{\alpha/2} pq}$$

Donde:

- N es la población o universo.
- Zα/2 es el valor tabulado del coeficiente de confianza, el coeficiente de confianza es la probabilidad que los resultados de nuestro estudio sean ciertos. El valor Zα/2 es una constante que depende del coeficiente de confianza elegido, la Tabla 1 muestra los valores de Zα/2 asociados a los niveles de confianza.
- pq es la varianza de las proporciones.
- p es la proporción que presenta el atributo de interés.
- q = (1 – p) es el complemento de p. Dado que p es la proporción de individuos que poseen en la población la característica bajo estudio, q es la proporción de individuos que no poseen en la población las características bajo estudio.
- e2 es el error máximo admisible, en tanto por ciento, cuando se desconoce su valor, entonces el investigador fija un criterio que puede variar entre el 1% (0.01) y 9% (0.09).

Normalmente el valor de la varianza (producto pq) es desconocido y, por lo general, se le asigna el valor p=q=0.5 que garantiza la varianza máxima y por ende maximiza el valor de n.

Valores $Z_{\alpha/2}$ por nivel de confianza

$Z_{\alpha/2}$	1.595	1.645	1.755	1.885	1.960	2.170	2.325	2.575
Nivel de confianza	89%	90%	92%	94%	95%	97%	98%	99%

Tabla de valores de Zα/2

 # Ilustración en R: calcular el tamaño n de una muestra aleatoria simple de una población N=10,000 habitantes de una comunidad, fijando un error máximo admisible del 4%, un nivel de confianza del 90% y varianza máxima de pq=0.25.

```
Z=1.645; p=0.5; q=1-p; N=10000; e=0.04
> n=(Z^2*N*p*q)/(e^2*(N-1)+Z^2*p*q)
> n
[1] 405.7032
```

Redondeando a cero decimales.

```
round(n,0)
[1] 406
```

Selección del tamaño de muestra simple (base y dplyr)

1. Muestreo irrestricto Aleatorio (MIA)

R Base contiene la función simple() con la que se puede obtener muestras aleatorias con o sin reemplazo de manera sencilla.

Para ilustrar vamos a cargar un dataset y cuantificar sus dimensiones.
Cargamos la dataset "crimtab" del paquete base.

```
> dcrim<-data.frame(crimtab)
> dim(dcrim)
[1] 924   3
```

La dataset "dcrim" tiene 924 observaciones y variables.

Seleccionamos una muestra de n = 30.
Tamaño de la muestra.

```
> n<-30
> mmia<- sample(1:nrow(crime),size=n,replace=FALSE)
> mmia
 R output
[1]  64 658 551 887 546 509 639 654 328 659 453 258 279 550 876 676 121
[18] 170 695 610  55 824 138  96   3 266 256 776 445 638
```

Asignar los elementos de la muestra al data frame de datos.

```
> mcrimmia<- dcrim[mmia,]
> head(mcrimmia)
 R output
```

```
    Var1  Var2   Freq
64  11.5 144.78    0
658 12.1 180.34    4
551  9.8 175.26    0
887  9.8 195.58    0
546 13.5 172.72    0
.8 172.72    0
```

2. **Selección de muestras simples con dplyr**

```
> library(dplyr)
```

Muestra sin reemplazo.

```
> mcrimmia2<- dcrim %>% sample_n(size=n,replace=FALSE)
> head(mcrimmia2)
    Var1  Var2   Freq
252 13.5 154.94    0
537 12.6 172.72    5
203 12.8  152.4    0
119 12.8 147.32    0
702 12.3 182.88    0
649 11.2 180.34    0
```

Muestra ponderadas con la frecuencia.

```
> mcrimmia3<- dcrim %>% sample_n(size=n,weight=Freq)
> head(mcrimmia3)
    Var1  Var2   Freq
314 11.3 160.02   24
402 11.7  165.1   37
177 10.2  152.4    2
616 12.1  177.8   10
486 11.7 170.18   45
```

Muestra con una proporción de casos.

```
> mcrimmia4<- dcrim %>% sample_frac(0.05)
> head(mcrimmia4)
     Var1   Var2 Freq
260 10.1 157.48    0
409 12.4  165.1    2
137 10.4 149.86    1
473 10.4 170.18    0
259   10 157.48    2
279   12 157.48    1

> dim(mcrimmia4)
[1] 46  3
```

Selección sistemática de una muestra (SamplinUtil)

Para el muestreo sistemático utilizaremos la función **sys.sample()** del paquete **SamplingUtil**. Para instalar este paquete, se debe inicialmente instalar el paquete **devtools** ya que no se encuentra en CRAN. Las instrucciones son las siguientes:

```
# Instalar las librerías devtools y SamplingUtil.
> install.packages("devtools")
> library(devtools)

# Instalar SamplingUtils.
install_github("DFJL/SamplingUtil")
library(SamplingUtil)

# Extracción de una muestra por el método de muestreo sistemático haciendo
uso de la dataset "dcrim". Comenzamos identificando las posiciones de los datos
extraídos.
> msistematico<- sys.sample(N=nrow(dcrim),n=30)
> msistematico
[1]  12  42  72 102 132 162 192 222 252 282 312 342 372 402 432
[16] 462 492 522 552 582 612 642 672 702 732 762 792 822 852 882

# Asignar los elementos de la muestra al data frame de datos.
> msistcrime<- dcrim[msistematico, ]
> head(msistcrime)
    Var1   Var2 Freq
12  10.5 142.24    0
42  13.5 142.24    0
72  12.3 144.78    0
102 11.1 147.32    0
132  9.9 149.86    0
.9 149.86    0
```

4.7 CÁLCULO DEL TAMAÑO DE MUESTRA SIMPLE (SAMPLINUTIL)

En este apartado veremos tres variantes para cálculo de tamaño muestral asumiendo muestreo irrestricto aleatorio (MIA). Para ello, utilizaremos la data de la ENAHO 2013 y la función **nsize()** incluída en el paquete **SamplingUtil**.

Como variable continua se utilizará la "Cantidad de personas del hogar" (TamHog), para el ejemplo de proporciones, se creará la variable "Proporción de Hogares Unipersonales".

Error relativo para variables continuas

Supongamos que estamos interesados en conocer el tamaño de muestra para que el error relativo no supere el 5%, con un alpha de 5%.

```
# Generando df de ENAHO2013 a nivel de hogar.
df<-ENAHO2013 %>%  # Define nuevo data frame.
mutate(TamHog=as.numeric(TamHog),  # Convierte TamHog a numérica.
phu=ifelse(TamHog>1,0,1)) %>%  # Crea variable de prop Hog.Unipersonales.
select(FACTOR:ZONA,TamHog,phu,-REGION) %>% # Selecciona variables de interés.
group_by(SEGMENTO,CUESTIONARIO,HOGAR,ZONA) %>%  # Genera esquema de agrupación.
summarise(TamHog=mean(TamHog),  # Cálculo de variables sumarizadas
phu=mean(phu))%>%mutate(id=paste0(SEGMENTO,CUESTIONARIO,HOGAR,ZONA))
# Genera id único para uso posterior.

# Cálculo del tamaño de la muestra con error relativo.
# Error relativo.
> r<-0.05
> nsizeR<- nsize(x=df$TamHog,r=r,alpha=0.05)
> nsizeR
 $n0
 [1] 782

 $n_adjust
 [1] 731
```

Cálculo del tamaño de la muestra con error absoluto

```
# Error absoluto para variables continuas.
# Igualar el error absoluto con el error relativo.
> abs<-mean(df$TamHog)*r
> nsizeA<- nsize(x=df$TamHog,abs=abs,alpha=0.05)
> nsizeA
  $n0
  [1] 782

  $n_adjust
  [1] 731
```

Cálculo del tamaño de la muestra para la proporción

```
# Error relativo para variables dicotómicas.
> nsizeP<- nsize(x=df$phu,abs=0.02,alpha=0.05)
> nsizeP
  $n0
  [1] 924

  $n_adjust
  [1] 854
```

4.8 CÁLCULO DEL TAMAÑO DE MUESTRAS ESTRATIFICADAS

Asignación proporcional

Cálculo de las proporciones por estrato.

```
> Estratos<-df%>%select(ZONA,TamHog)%>%group_by(ZONA)%>% summarise(n=n(),
s=sd(TamHog)) %>% mutate(p=n/sum(n))
> Estratos
```

Output

```
  Source: local data frame [2 x 4]
   ZONA    n      s         p
1 Urbana 4790 3.149710 0.4269543
2 Rural  6429 1.945832 0.5730457
```

Asignación de la muestra proporcional a los estratos.

```
>nsizeProp<-nstrata(n=nsizeR[[2]],wh=Estratos[,4],method="proportional")
> nsizeProp
Output
     p
 1  313
 2  419
```

Asignación óptima de la muestra a los estratos

Coste de las entrevistas por estrato.

```
> ch<-c(5,10)
>nsizeOpt<-nstrata(n=nsizeR[[2]],wh=Estratos[,4],sh=Estratos[,3],ch,
method="optimal")
> nsizeOpt
Output
     p
 1  461
 2  271
```

Asignación de Neyman de la muestra óptima a los estratos(asume costes iguales)

```
>nsizeNeyman<-nstrata(n=nsizeR[[2]],wh=Estratos[,4],sh=Estratos[,3],
method="neyman")
> nsizeNeyman
Output
     p
 1  400
 2  332
```