# Programación y estadística con R

Contenido

[Programación y estadística con R 1](#_Toc75265451)

[Introducción a R y software 1](#_Toc75265452)

[Introducción 1](#_Toc75265453)

[¿R o R Studio? 1](#_Toc75265454)

[Empezando con R Studio 2](#_Toc75265455)

[Mi primer línea de código 2](#_Toc75265456)

[R como calculadora 3](#_Toc75265457)

[¿Cómo escribir números pequeños y grandes? 3](#_Toc75265458)

[Orden de operaciones 4](#_Toc75265459)

[Help y Stackoverflow 4](#_Toc75265460)

[Importar archivos a R 5](#_Toc75265461)

[Asignación de una variable 7](#_Toc75265462)

[Mi primer Script 8](#_Toc75265463)

[Instalar un paquete 8](#_Toc75265464)

[¿Qué es una función? 9](#_Toc75265465)

[Cómo crear una nueva función 10](#_Toc75265466)

[Programación y manipulación de datos en R 11](#_Toc75265467)

[Introducción 11](#_Toc75265468)

[Medidas de tendencia central 11](#_Toc75265469)

[Medidas de posición 13](#_Toc75265470)

[Medidas de dispersión 14](#_Toc75265471)

## Introducción a R y software

### Introducción

Bienvenido a la sesión propedéutica del módulo de programación y estadística con R. En este módulo aprenderás los conceptos básicos de estadística y los aplicarás con R, un lenguaje de programación que tiene un enfoque de análisis estadístico. Esta primera sesión propedéutica te dará las bases necesarias para utilizar el lenguaje R durante todo el módulo a la par que aprendes conceptos estadísticos.

Antes de comenzar, sigue las instrucciones para instalar R y Rstudio. Te compartimos anteriormente estas instrucciones si aún no lo realizas o tienes dudas, puedes consultar los siguientes 3 videos tutoriales que te ayudarán a conocer como instalar R, R Estudio y sus paquetes..

[Instalar R](https://drive.google.com/file/d/1CMLvw_uhYP-wrWeWEnc5UTB8LeQRJGGu/view?usp=sharing" \t "_blank)

[Instalar R Studio](https://drive.google.com/file/d/1N5ckkyxPzjCZwg-cgE4Y2ZLnPILZqQLk/view?usp=sharing" \t "_blank)

[Usar R Studio / Usar paquetes](https://drive.google.com/file/d/1Afz0j07r4Di7wUMlRfvZlpoXpAvg_6em/view?usp=sharing" \t "_blank)

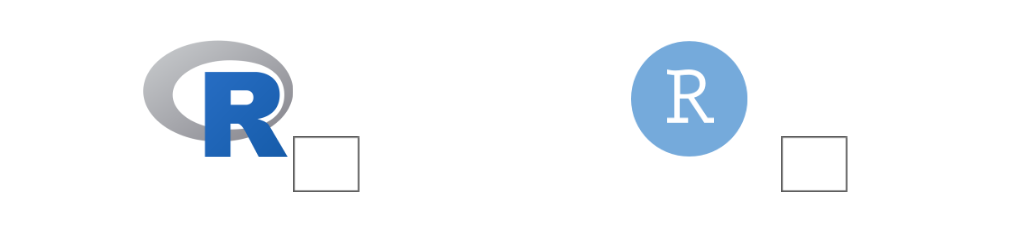
La primera vez que nos enfrentamos a un lenguaje de programación suele ser confuso e intimidante. Existen algunos temas que vale la pena tener claros para, más adelante, evitar confusiones. En este nuevo mundo que estás por descubrir, el mundo de **R**, se pueden crear muchos tipos de análisis, pero antes de dar este salto, es importante reforzar los **fundamentos del lenguaje**.

### ¿R o R Studio?

Lo primero que debemos entender es qué es R y la diferencia con R Studio, los cuáles ya tienes instalados. Como dijimos, R es un lenguaje de programación mientras que R Studio es un Entorno Integrado de Desarrollo (IDE), el cual nos brindará herramientas para poder escribir nuestro código de manera amigable y poder administrar nuestros proyectos muy fácilmente.

#### Actividad 1

Busca en tus programas instalados y completa ¿Cuál icono corresponde a R y cuál a R Studio?



A lo largo de este curso y para hacernos la vida más fácil trabajaremos en **R Studio**. Es importante mencionar que, como el **lenguaje de programación** es **R**, es indispensable tener instalados ambos programas juntos. De hecho, R podría vivir solo, pero R Studio no puede vivir sin que también exista R en nuestra computadora.

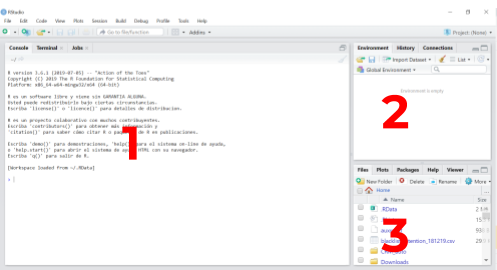
### Empezando con R Studio

Vamos a abrir R Studio por primera vez. Para hacerlo es muy fácil:

* Busca el icono de R Studio en tu computadora
* Dale doble click

Al abrirla, podemos ver tres elementos principales:

* La **consola**
* El entorno e historia
* Files/Plots/Packages/Help/Viewer



¿Cómo sé si R Studio está listo para recibir mis instrucciones? ¡Muy fácil! Busca en la consola el icono > y da clic a la derecha, podrás ver en la imagen 1 cómo debe de verse. Esto nos indica que estamos listos para comenzar.

### Mi primer línea de código

Prueba escribir la siguiente línea de código y da **Enter**. De respuesta, obtendrás algo como lo siguiente:

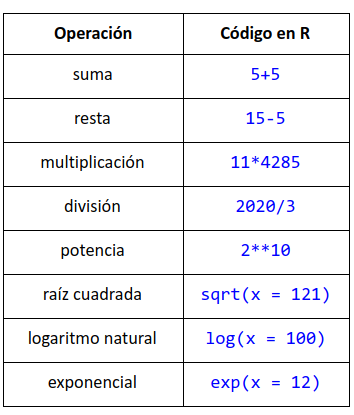


Es oficial, lograste comunicarte con tu computadora por primera vez. Lo podemos pensar como: *“Acabo de decirle a la computadora que me imprima el texto Hola R, después di Enter, que fue la manera de decirle ‘Hazlo ahora’. Entonces, la computadora hizo lo que le pedí: imprimió el texto Hola R.”* Nota: No es necesario utilizar la función print cuando se requiera imprimir una variable, con su nombre es suficiente."

Es de resaltar que en R el uso de **''**, es idéntico al de utilizar **“ ”**, las puedes utilizar del mismo modo cuando quieras escribir e identificar datos de tipo carácter.

### R como calculadora

Una de las maneras más sencillas de utilizar R, es como una calculadora. A continuación tenemos las operaciones básicas y la manera de escribir el código en R.



#### Actividad 2

Ejecuta en la consola cada operación de la tabla para descubrir el resultado. ¿Son los resultados que esperabas?

### ¿Cómo escribir números pequeños y grandes?

Hay que tener en mente que cuando escribimos números pequeños o grandes, R cambia los números a la **notación e**. Podemos ver el siguiente ejemplo:

Forma

Descripción generada automáticamente con confianza media

En esta notación, 1e+07 es lo mismo a 1 × 10ˆ7 que a su vez es lo mismo a 10000000 .

Por otro lado, 1e-07 es lo mismo a 1 × 10ˆ− 7 que a su vez es lo mismo a 0.0000001 .

En resumen, la **notación e** nos dice que cualquier número a puede ser expresado como aey, que representa al número a × 10y .

Pero, R tiene un límite. Al intentar representar un número mucho más grande o mucho más pequeño, entonces lo hace con infinitos. Por ejemplo,

Forma

Descripción generada automáticamente con confianza media

### Orden de operaciones

¿Recuerdas cuando nos enseñaron el orden de las operaciones en matemáticas? Bueno, R las respeta al ejecutar código. Por lo tanto, hay que tenerlas presentes, este orden lo enlistamos a continuación:

* Paréntesis
* Exponentes
* Multiplicación
* División
* Suma
* Resta

Esto quiere decir que, si tenemos una operación con paréntesis, R ejecutará primero lo que esté en paréntesis. Cuando termine, entonces elevará a las potencias que tenga y así sucesivamente.

### Help y Stackoverflow

Cuando empezamos en el mundo de la programación, es normal que no sepamos qué escribir. Imagínate, podrías estar en alguno de los siguientes casos:

* Hace mucho tiempo utilicé una función para obtener el promedio, la cual sé que se llama mean, pero no recuerdo qué tenía que poner de argumento. Solución:

?mean

* Creo que puede existir una función que me de el promedio, pero no sé cómo se llama esa función. Solución:

??mean

* Estoy atorado, ya no sé qué escribir o hacer. Solución:

Seguramente, tu duda alguien más la tuvo anteriormente y la puedes buscar (con solución) en la siguiente página:

[stackoverflow](https://stackoverflow.com/" \t "_blank)

### Importar archivos a R

R puede leer diferentes tipos de archivos. Por lo tanto, es muy importante que sepas cómo importarlos y exportarlos. Esto lo irás aprendiendo poco a poco, sin embargo, el siguiente [recurso](https://support.rstudio.com/hc/en-us/articles/218611977-Importing-Data-with-RStudio) puede ser de mucha ayuda para identificar los diferentes tipos de archivos que puedes usar en R así como la manera en la que los puedes importar.

Como ejercicio adicional puedes importar a Rstudio uno de estos sencillos datasets (base de datos) para principiantes de la plataforma [Kaggle](https://www.kaggle.com/rtatman/fun-beginner-friendly-datasets).

Sigue los pasos del primer recurso sobre importación de archivos y si necesitas más información, indaga en stackoverflow (con tus habilidades de investigación) para lograr importar alguno de estos archivos.

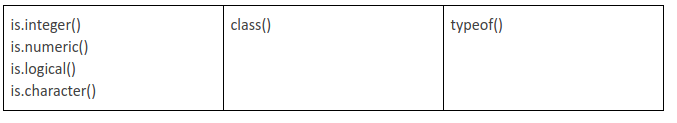
En este prework aprenderás algunos conceptos básicos sobre R. Identificarás los diferentes tipos de datos que existen en este lenguaje; las estructuras como vectores y matrices; las funciones y finalmente la lectura de datos.

Comencemos con los **tipos de datos**. En R hay varios tipos de datos. Sin embargo, vamos a comenzar a identificar los más básicos.

1. Numeric (numéricos) Estos tipos de datos son de tipo numérico. Es decir, son números. Dentro de esta categoría hay otras subcategorías:
   * Integer (enteros) → Estos se representan con una L para indicarle a R que son enteros y no decimales. 2L, 97L, 3888L …
   * Float (decimales) 18.9, 0.19 …
2. Complex (complejos) → Números complejos con números imaginarios 1+4i …
3. Logical (lógicos) → Sólo hay dos opciones ya sea TRUE o FALSE
4. Character (carácter) → Son letras, palabras o frases y se representan entre comillas siempre. Pueden ser números entre comillas que serán tomados como caracteres y no como numéricos.
   * “A”, “Hello world”, “34”

Puedes empezar a hacer pruebas con los diferentes tipos de datos dentro de Rstudio. Si utilizas las siguientes funciones puedes:

* identificar el tipo de dato →



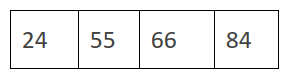
Dentro del paréntesis pon el dato que quieras identificar.

* cambiar el tipo de dato → as.integer() as.numeric() as.logical() as.character()

Por otro lado, existen varias estructuras en R (a estructuras nos referimos a la manera en la que organizamos los diferentes tipos de datos que tenemos). Vamos a revisar un ejemplo de cada una de las estructuras yendo de las más sencillas a las más complejas.

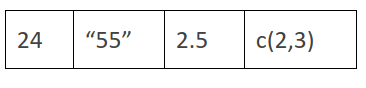
* **Vectores** Estos son valores que sólo están en una dimensión (Esto quiere decir que están en una línea y no forman tablas con columnas ni filas). Normalmente se crean con la función c(). Esto hace que lo que esté dentro del paréntesis se convierta en un vector. Con los vectores se pueden hacer operaciones aritméticas y ÚNICAMENTE pueden ser del mismo tipo de datos.

Ejemplo: x <- c(24, 55,66,84) Esto es un vector numérico de integers y la información se ve así en una línea en una única dimensión.



* **Listas** Las listas son igual a los vectores, pero lo que difiere de ellos es que estas sí pueden guardar diferentes tipos de datos. Estas se pueden crear usando la función list().

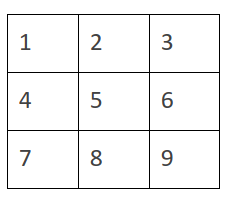
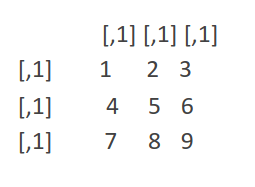
Ejemplo: x <- list(24, “55”,2.5, c(2,3))



Nota que se pueden guardar vectores dentro de listas.

* **Matrix** Las matrices son datos estructurados en dos dimensiones: columnas y filas. Estas se pueden crear con la función matrix() y al igual que los vectores, únicamente pueden guardar 1 mismo tipo de datos.

Ejemplo: x <- matrix(1 : 9, byrow = TRUE, nrow = 3) Cuando imprimes x, este es el resultado:



La función matrix() hace que los números se representen en dos dimensiones: columnas (arriba a abajo) y filas (de un lado a otro). Puedes poner esta línea de código en Rstudio y darle enter para saber cuáles son los argumentos de esta función: ?matrix

* **Data Frames** Los data frames (datos organizados en 2 dimensiones: columnas y filas) son igual que las matrices, pero estos sí pueden guardar diferentes tipos de datos.

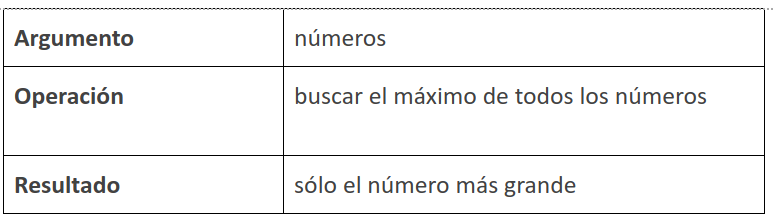
Ahora, ya sabes cuáles son los tipos de datos y las estructuras en R. En la siguiente sección aprenderás lo que es una función y una variable, así como la manera en la que asignas una.

Como ya habrás identificado, existen funciones en R. Por ejemplo, las funciones que ya revisaste previamente como las de identificar tipos de datos typeof(), class() o funciones para convertir estructuras de datos como matrix(), list(), etc…

Las **funciones** en R son aquellas que convierten una entrada o dato, las procesan y arrojan un resultado. Realiza la actividad 3 para poner en práctica tus habilidades con las funciones en R.

#### Actividad 3

Existe una función en R llamada max, la cual tiene las siguientes partes:



Utiliza esta función para encontrar el máximo entre los números 100/4, 108/5, 200/20, sqrt(81), 3\*5. Recuerda que para saber cómo implementar la función mean puedes pedir ayuda.

### Asignación de una variable

Hemos aprendido a hacer algunas operaciones, buscar funciones que ya existen en R y también a pedir ayuda. Pero muchas veces, los **resultados** de las operaciones que hacemos, o de las funciones que utilizamos, las queremos **guardar** para utilizarlas más adelante. Para guardar un elemento, podemos utilizar una **variable**. Aunque tenga un nombre raro, una variable sólo es un objeto al que vamos a “etiquetar” con un nombre que nosotros queramos. Por ejemplo, imagina que el resultado de la siguiente operación la tenemos que utilizar más adelante, por lo que guardamos el resultado en una variable llamada **resultado**.

resultado <- ((43012\*180)+10)/4.1

Al hacerlo, no se imprime nada. Sólo se guarda la nueva variable y en un futuro podemos ver lo que tiene.

* resultado
* [1] 1888334

### Mi primer Script

Como te puedes dar cuenta, la **consola** es muy amigable para correr comandos cortos, pero, en general vamos a preferir trabajar en un **R Script**, pues funciona como un editor de texto y nos dejará visualizar muchas líneas de código a la vez. Para abrir un R Script:

* Windows: ve a File-> New File-> R Script
* Mac: File-> New Document-> R Script en Mac

Esto nos abre un nuevo panel en la parte superior izquierda. Para que una línea de código se ejecute en un R Script (cómo al hacer Enter en la consola) basta con ponerse en la línea deseada y picarle al botón Run.

En el proceso de crear algo nuevo con R muy seguramente necesitaremos funciones mucho más específicas a las que tenemos precargadas en R. Por ejemplo, si tuvieras que graficar algunos puntos en un mapa de la Ciudad de México, ¿tienes idea de cómo harías una tarea tan específica?

* **PREWORK - PARTE 2**

R, a pesar de ser un lenguaje relativamente nuevo, tiene una **comunidad** muy grande de personas que continuamente **contribuyen** a través de **paquetes**. Dichos paquetes incluyen **funciones y datasets** relacionados a un tema específico, y nos ahorra el trabajo de tener que programar muchas cosas desde cero.

#### Actividad 1

Abre el [link](https://rstudio.com/products/rpackages/) y descubre para qué sirve el paquete ggplot2.

### Instalar un paquete

Como te pudiste dar cuenta, ggplot2 es un paquete para visualización de datos. Entonces, si queremos crear visualizaciones, nos conviene instalar el paquete de ggplot2 y así poder utilizar todas las funcionalidades que se instalan. Para instalar un paquete, tenemos que hacer:

install.packages(“ggplot2”)

Basta con hacer esto una vez en la vida para que quede instalado el paquete. Es decir, lo instalo hoy y en dos meses va a seguir instalado sin necesidad de volver a ejecutar el comando anterior. El detalle es que, cada vez que queramos utilizarlo, tenemos que hacerle un llamado. Este se hace de la siguiente manera:

library(ggplot2)

Al hacer esto, estamos avisando que necesitamos utilizar funciones de la librería ggplot2. Esto se tiene que hacer cada que iniciemos un script nuevo, si no, no será capaz de ejecutar las funciones que vienen cargadas con ggplot2, aunque ya esté instalado.

#### Actividad 2

Abre el [link](https://rstudio.com/products/rpackages/) y descubre para qué sirve el paquete dplyr. Después, instala y llama el paquete. Por último, busca en Google ‘dplyr cheat sheet’ y mira la información que tiene.

### ¿Qué es una función?

Una persona que vende galletas de 3 diferente tamaños (la chica cuesta $5, la mediana $7 y la grande $9) se enteró que ya estás aprendiendo a programar y que además, eres muy bueno utilizando R como una calculadora. Por eso, te quiere contratar para que le ayudes a calcular el precio final para cada cliente. Esta persona que vende las galletas tiene una manera muy peculiar de cobrar: tienes que pagar solamente el promedio de todas las galletas que llevas, y puedes llevar a lo más 2 galletas por persona.



La única opción que tenemos con lo que sabemos hasta ahora, es estar en la tienda y para cada cliente, calcular el promedio de lo que lleva.

El cliente 1 lleva una galleta chica y una mediana, entonces hacemos:

* (5+7)/2

[1] 6

El cliente 2 lleva dos galletas chicas, entonces hacemos:

* (5+5)/2

[1] 5

El cliente 3 lleva una galleta grande, entonces hacemos:

* 9/1

[1] 9

y tendríamos que hacer este proceso para cada cliente que entre a la tienda.

Por suerte, hay mucha gente que está utilizando R en el mundo que nos ayuda a hacer nuestra tarea más fácil. En particular, existe una función llamada mean la cual calcula el promedio de los números, sin que nosotros tengamos que hacer la operación manual. Podríamos utilizarla de la siguiente manera para los mismos casos que antes.

* mean(c(5,7))

[1] 6

* mean(c(5,5))

[1] 5

* mean(c(9))

[1] 9

Esta es la función mean. Existen muchas funciones precargadas en R, las cuales tienen como objetivo recibir uno o más datos, hacer una operación intermedia y dar un resultado. En este caso, la función mean recibe números. Después, y aunque nosotros no lo vemos, calcula el promedio, como lo hicimos nosotros antes y al final solo nos regresa el resultado.

### Cómo crear una nueva función

Las funciones que ya existen en R son muy fáciles de utilizar. Por ejemplo, la función mean() recibe como argumento una serie de números y regresa el promedio. En realidad, casi todas las funciones generales están programadas en R. Por lo tanto, nos limitaremos a crear nuevas funciones solo en los casos donde necesitamos crear una serie de instrucciones muy particulares.

Para poder crear una función, hay que conocer su estructura:

* Nombre de la función
* Argumentos
* Cuerpo
* Resultado

Para poder escribir una nueva función, se tiene que utilizar la sintaxis:

Texto

Descripción generada automáticamente

En donde, en este caso, el nombre de la función es nombre.funcion y tiene n argumentos llamados argumento.1, argumento.2, ... argumento.n. El cuerpo de la función depende de las operaciones que queramos hacer utilizando los argumentos y esta función regresa la var.resultado.

#### Ejemplo

Vamos a crear una función que tenga:

* Nombre: saludo
* Argumento: nombre
* Body: utilizar el nombre de la función para crear 'Hola -nombre-'
* Resultado: Que al llamar esta función, nos regrese el mensaje

Texto

Descripción generada automáticamente

Vamos a crear otra función que tenga:

* Nombre: operacion.personalizada
* Argumento: tres números llamados a,b,c
* Body
  + (a+b)4 + 2a
  + al resultado de lo anterior, multiplicarlo por dos y sumarle c/2
  + al resultado de lo anterior, multiplicarlo por tres
* Resultado: Que al llamar esta función, nos regrese el último resultado

Texto

Descripción generada automáticamente

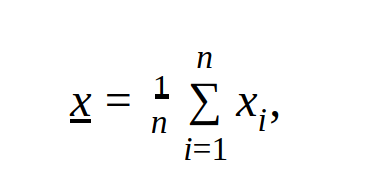
## Programación y manipulación de datos en R

### Introducción

En este prework, estudiarás las medidas de tendencia central más conocidas, algunas medidas de posición muy útiles como los cuartiles y medidas de dispersión. Es muy importante que conozcas estos conceptos básicos cuando estudias estadística, posteriormente en el work, trabajarás con datos, y los cálculos serán muy rápidos y fáciles de hacer con algunas funciones de R, pero primero necesitas entender el significado de estas medidas, para que, en el work, te puedas enfocar únicamente en cálculos fáciles e interpretaciones.

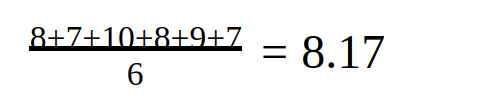
### Medidas de tendencia central

Dado un conjunto de datos, las llamadas medidas de tendencia central son números alrededor de los cuales se concentran los datos. La **media** o promedio es quizás la medida de tendencia central más conocida. Dado un conjunto de datos {x1, ⋯, xn} que representa una muestra de alguna población, la media del conjunto se define como:

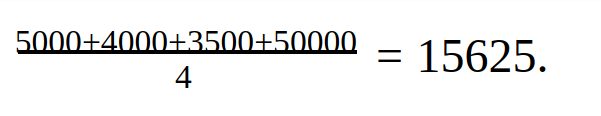


es decir, el promedio de los datos.

**Ejemplo**. Un estudiante cursó seis materias en el semestre, obteniendo calificaciones de 8, 7, 10, 8, 9 y 7. Su calificación media o promedio semestral es de



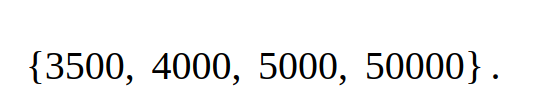
**Ejemplo**: Una pequeña compañía consultora tiene una secretaria, un empleado de limpieza, un mensajero y un economista. Sus salarios mensuales respectivos son de $5000, $4000, $3500 y $50000. El salario promedio de la compañía es de



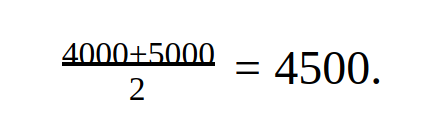
El lector podría pensar que este salario no es una medida representativa de los salarios del personal. ¿Qué sucede? La razón es que existe un dato, el salario del economista, que está totalmente fuera del rango de los demás salarios. Así, al realizar el promedio, este dato jala a los demás.

Para evitar problemas con datos alejados de los demás, como en el ejemplo anterior, se utiliza otra medida de tendencia central llamada **mediana**. La mediana es el valor que parte al conjunto de datos ordenados en dos. Para encontrar la mediana, los datos se ordenan de menor a mayor y, si el conjunto de datos es impar, la mediana es el valor que se encuentra a la mitad del conjunto, Si el conjunto de datos es par, la mediana es el promedio de los dos datos intermedios.

**Ejemplo**. Encontrar la mediana del conjunto de salarios {$5000, $4000, $3500 y $50000}. Los ordenamos de menor a mayor obteniendo,



Este es un conjunto par de datos de manera que la mediana es el promedio de los dos datos intermedios, es decir la mediana es



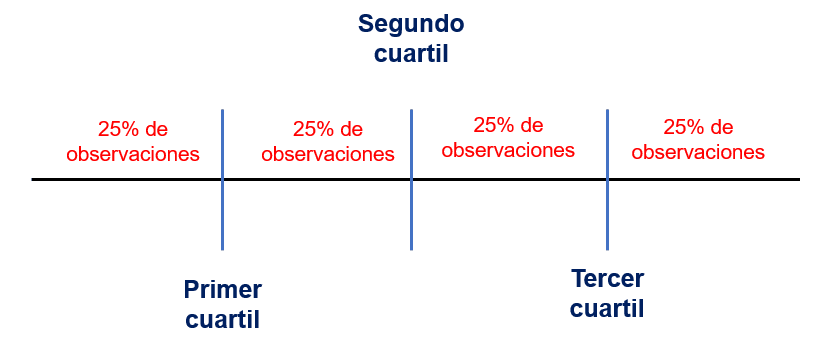
En ocasiones, este dato puede ser una mejor representación de los salarios de la empresa que el promedio obtenido anteriormente en el ejemplo.

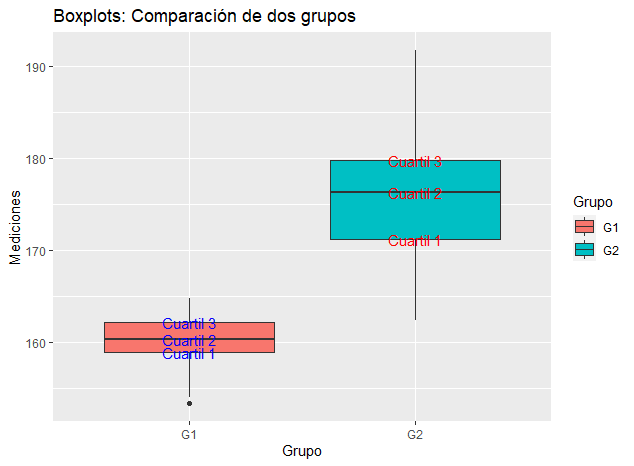
La media y la mediana no toman en cuenta la repetición de los datos, se define para este efecto, la **moda**. Ésta es simplemente el valor o categoría que ocurre con mayor frecuencia en un conjunto de datos. Es claro que puede haber más de una moda ya que puede haber más de un dato que se repita con la misma frecuencia. Para el caso de dos modas, decimos que la distribución de los datos es bimodal. Cuando existen más de dos modas, la distribución se denomina multimodal.

### Medidas de posición

De manera un poco informal pero útil para la práctica, podemos decir que el cuantil de orden p (0 < p < 1) de un conjunto de mediciones, es un número que deja una proporción p de valores del conjunto por debajo de él. Por ejemplo, el cuantil de orden 0.43 dejaría un 43% de las observaciones por debajo de él.

**Cuartiles**. Los cuartiles son 3 números que dividen al conjunto de datos en cuatro partes iguales, es decir, debajo del primer cuartil se encuentra el 25% de las observaciones, el segundo cuartil es la mediana y el tercer cuartil es un número que tiene el 75% de las observaciones por debajo de él.





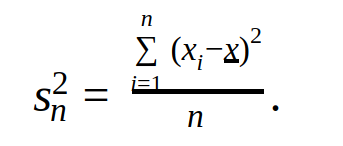
**Deciles**. Los deciles son 9 valores que dividen el conjunto de datos en 10 partes iguales.

### Medidas de dispersión

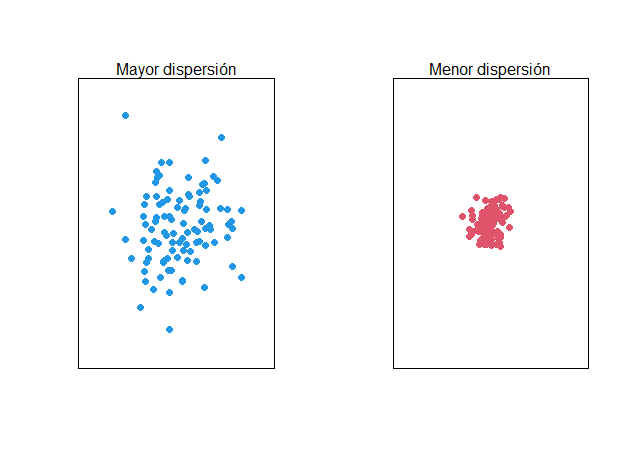
Consideremos a los conjuntos {5, 6, 7, 8, 9} y {1, 2, 7, 12, 13}. Ambos tienen media y mediana iguales a 7 y, sin embargo, nuestra intuición nos dice que los datos del segundo conjunto están más dispersos. ¿Cómo formalizar este concepto de dispersión? Una forma de hacerlo es considerando el rango o extensión de los datos que se define como la diferencia entre el dato más grande y el más pequeño.

**Ejemplo**. El rango del conjunto de datos {5, 6, 7, 8, 9} es de 9 - 5 = 4 y el rango del conjunto {1, 2, 7, 12, 13} es de 13 – 1 = 12. Observemos que el conjunto con el rango más grande es más disperso.

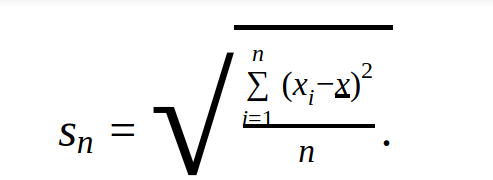
Una forma muy utilizada para medir la dispersión de un conjunto de datos es la llamada **desviación estándar**. Esta mide qué tanto los datos se desvían de la media y se denota comúnmente por sn. Esta desviación se construye a partir de su cuadrado conocido como varianza, como sigue: dado un conjunto de datos {x1, ⋯xn} con media x, la varianza se define como



Intuitivamente, xi-x es la distancia a la media del dato xi, ésta se eleva al cuadrado para tener siempre un valor positivo y se divide entre n para obtener el promedio de estas desviaciones.

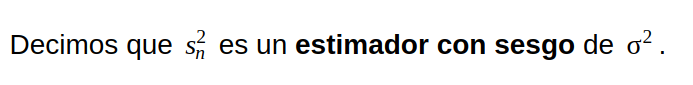


Por definición, la varianza da un valor numérico para el promedio de los cuadrados de las distancias. Para que el número conserve las unidades originales de la variable, se toma la raíz cuadrada y se tiene así la desviación estándar

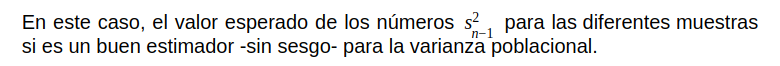
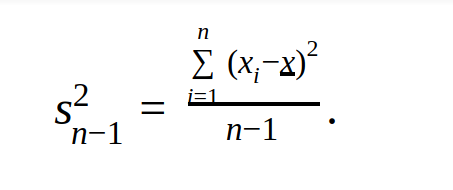
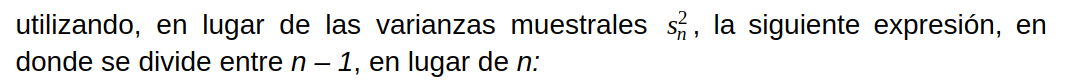


Esta desviación nos da una idea de que tan alejados están los datos de la media.

A la varianza de una población se le denota por 02 y a su desviación estándar por 0. Al igual que en el caso de la media, la varianza de una población puede inferirse a partir de las varianzas de las muestras. Desgraciadamente, en este caso se tiene una complicación: la varianza de la población no se aproxima bien por el valor esperado de las varianzas de las muestras y tiende a subestimarse.



El problema de estimación de la varianza poblacional puede solucionarse



**Rango intercuartílico**. Otra medida de dispersión muy útil es el **rango intercuartílico**. Consulta en diversas fuentes acerca del rango intercuartílico y cuáles son sus ventajas como medida de dispersión.

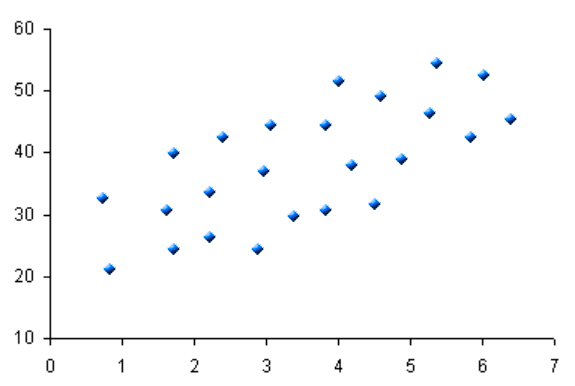
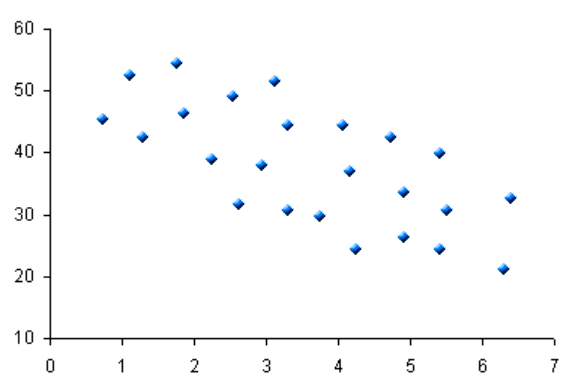
## Análisis Exploratorio de Datos (AED o EDA) con R

### Introducción

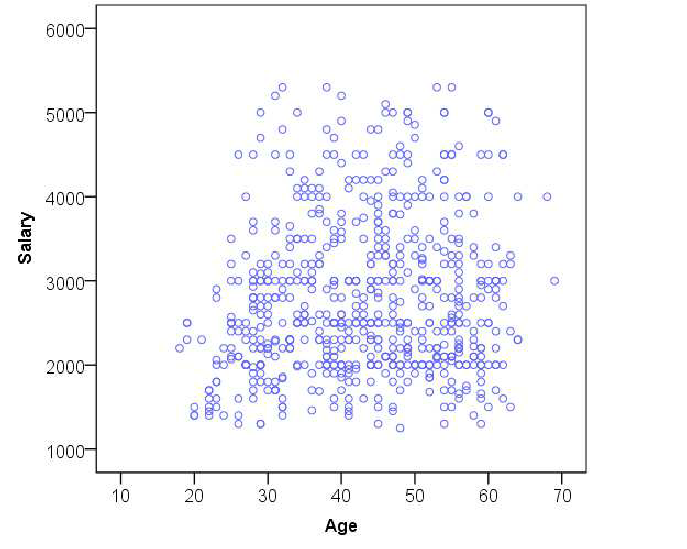
Un diagrama de dispersión, gráfica de dispersión o gráfico de burbujas es un tipo de diagrama matemático que utiliza las coordenadas cartesianas para mostrar los valores de dos variables para un conjunto de datos.

### Gráficos de dispersión

Los diagramas de dispersión son útiles para reconocer tendencias en datos; cuando estos son graficados en puntos, se deben tomar dos variables de tipo cuantitativo. A continuación, se presenta una imagen donde se pueden observar dos tipos de correlación que generalmente son de interés. Cuando se trata de una alta dispersión se puede suponer que la correlación es cercana a cero, y cuando tenemos poca dispersión se puede deber a correlaciones cercanas a 1 o a -1, sin embargo puede existir poca dispersión si los datos se aglomeran en clusters de información, es decir los datos están muy “compactados”, la correlación puede obtenerse mediante el comando cor().



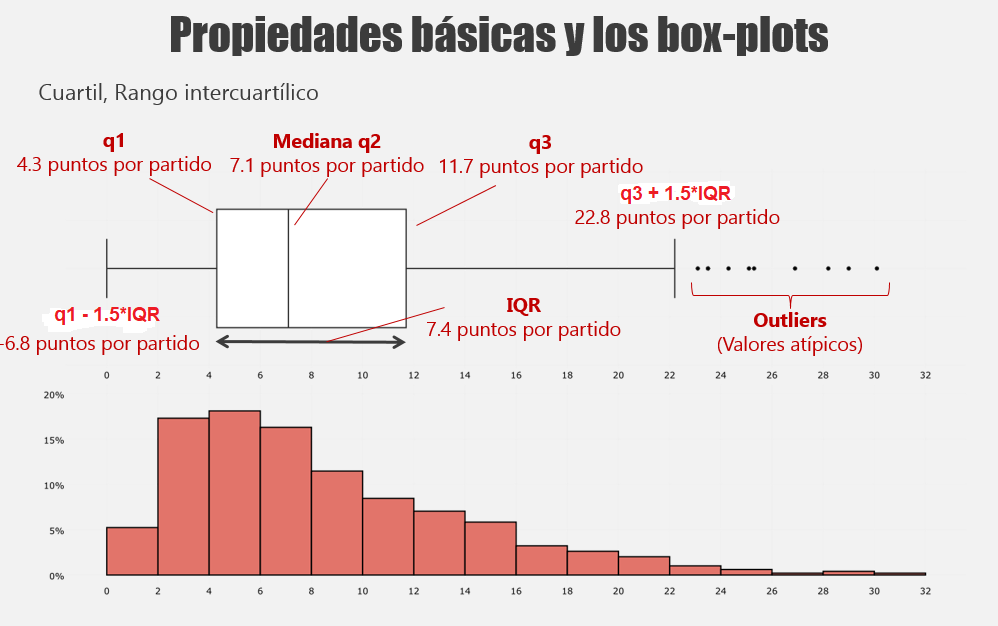
La ilustración 1 es una correlación negativa, esto sucede cuando una variable crece (en este caso eje de las “x”) y la otra disminuye (eje de la “y”). Para el caso de la ilustración 2, se representa una correlación positiva, cuando una variable crece (eje “x”), la otra también lo hace (eje “y”). Puede suceder que no se pueda reconocer un patrón específico, esto también es útil ya que indica que las variables no tienen una correlación o que esta no es tan fácil de determinar visualmente debido a que su coeficiente de correlación es muy bajo.



El comando en R para realizar un gráfico de dispersión es: plot(var1, var2, ...)

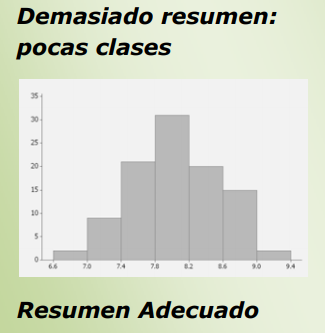
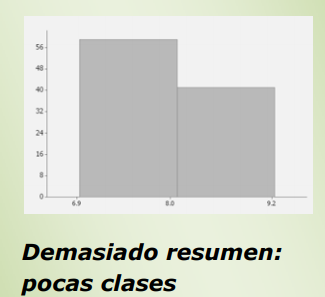
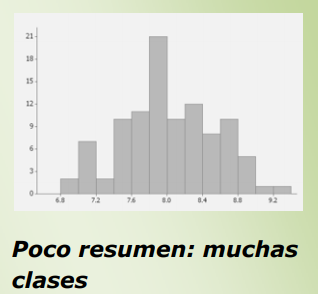
### Boxplots y outliers

También conocido como diagrama de caja y bigote, box plot, box-plot o boxplot. Es un método estandarizado para representar gráficamente una serie de datos numéricos a través de sus cuartiles (1ero; 25%, 2do; 50%, 3ro; 75%) . El máximo interés del box-plot es visualizar la distribución de una variable numérica de la manera más simplificada posible. Sólo utiliza los valores de los cuartiles, los extremos (q1 - 1.5IQR y q3 + 1.5IQR) y valores raros o outliers. No depende de valores ponderados como la media, simplemente se fija en las características de la posición. El diagrama siguiente será de mucha utilidad para comprenderlos. En R se utiliza el comando boxplot para graficarlos.



### Histogramas

Es una gráfica de la distribución de un conjunto de datos. Es un tipo especial de gráfica de barras, en la cual una barra va pegada a la otra, es decir no hay espacio entre las barras. Cada barra representa un subconjunto de los datos. Un histograma muestra la acumulación o tendencia, la variabilidad o dispersión y la forma de la distribución. Un histograma es una gráfica adecuada para representar variables continuas, aunque también se puede usar para variables discretas. Es decir, mediante un histograma se puede mostrar gráficamente la distribución de una variable cuantitativa o numérica. Los datos se deben agrupar en intervalos de igual tamaño, llamados clases.



Se grafican en el eje de las X las clases y en el eje Y las frecuencias de nuestros datos entonces de ese modo obtenemos el histograma, que es la representación visual de la distribución de frecuencias.

Para realizar un boxplot se utiliza el comando en R: Boxplot(var1, var2,...)

### Series de tiempo

Es un conjunto de valores observados durante una serie de periodos temporales, secuencialmente ordenada. Son variables estadísticas bidimensionales en donde el tiempo es la variable independiente, y la otra es la variable dependiente.

Se construyen modelos de series de tiempo para:

Obtención del mecanismo

Estudio de su evolución futura o predicción.

Se realiza:

Analizando los componentes o factores que determinan los resultados de la información.

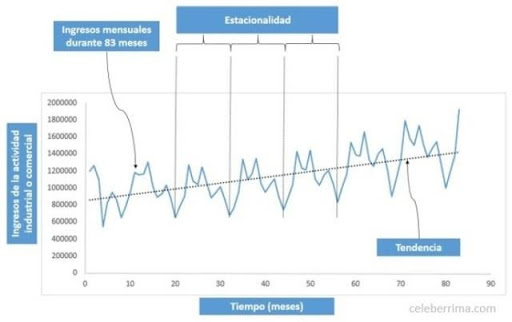
El método clásico para el análisis de series de tiempo identifica cuatro componentes:

TENDENCIA (T).- El movimiento general a largo plazo de los valores de la serie de tiempo (Y) sobre un extenso periodo de años.

FLUCTUACIONES CÍCLICAS (C).- Movimientos ascendentes y descendentes respecto de las tendencias recurrentes, con una duración de varios años.

VARIACIONES ESTACIONALES (E).- Movimientos ascendentes y descendentes respecto de la tendencia que se consuman en el término de un año y se repiten anualmente, estas variaciones suelen identificarse con base en datos mensuales o trimestrales.

VARIACIONES IRREGULARES (I).- Las variaciones erráticas respecto de la tendencia que no puedan atribuirse a las influencias cíclicas o estacionales. A continuación, se muestran las partes de una serie de tiempo.



Existe la descomposición aditiva y multiplicativa las cuales ayudan a entender el comportamiento de la serie de tiempo.

Para realizar las series de tiempo se utilizará el comando ts()