**Semana 1**

**04. Lección de Swirl 2: Objetos y Tipos de Dtos**

**Lección 3.1 – Operaciones básicas y Números**

Hay varios objetos en R:

* Objetos de Lenguaje
* Expresiones: Conjunto de expresiones correctas aún no evaluadas
* Funciones: Con una lista de argumentos y su código y entorno.

Cuando R se utiliza por primera vez es una calculadora, haciendo con ella todas las operaciones aritméticas y lógicas, como con vectores y números. Las operaciones lógicas se evalúan de izquierda a derecha y con cada elemento de cada vector. El vector es el objeto más simple en R.

Se pueden seleccionar elementos dentro de un vector, y a su vez se pueden manipular valores perdidos y ordenar éstos.

Para operaciones con conjuntos se pueden hacer partes y con secuencias determinísticas y secuecnias aleatorias con funciones.

En R casi todo es un objeto, incluyendo las funciones y estructuras de datos. Con la función *ls()* regresa el contenido del objeto. La función *attach()* carga de nuevo los objetos a la memoria.

R tiene 5 tipos de objetos básicos o atómicos, que en realidad son 3:

* *character*
* *numeric* – Objetos flotantes
  + *integer*
  + *complex –* Representa los números complejos
* *logical*

El objeto más básico es el vector. La lista se puede ver contrario al vector, viéndose como un vector de objetos de diferentes clases, teniendo en cuenta que el vector sólo puede contener objetos de la misma clase. Los números en R se tratan como elementos de tipo numérico, si queremos números enteres tenemos que hacerlo explícitamente poniendo una L detrás del número. Los valor TRUE y FALSE se usan con valores de tipo lógicos.

NaN significa Not a Number, y Inf significa Infinito. Este valor NaN aparece cuando llevamos a cabo alguna operación aritmética indeterminada, e Inf con las operaciones que den infinito.

**Lección 3.2 – Atributos, Entrada y Coerción**

Otra característica de los objetos en R son los atributos, que pueden ser nombres, dimensiones, clases, tamaño y otros atributos definidos con el usuario. Los atributos de un objeto se ven con la función *attributes()* con el nombre del objeto.

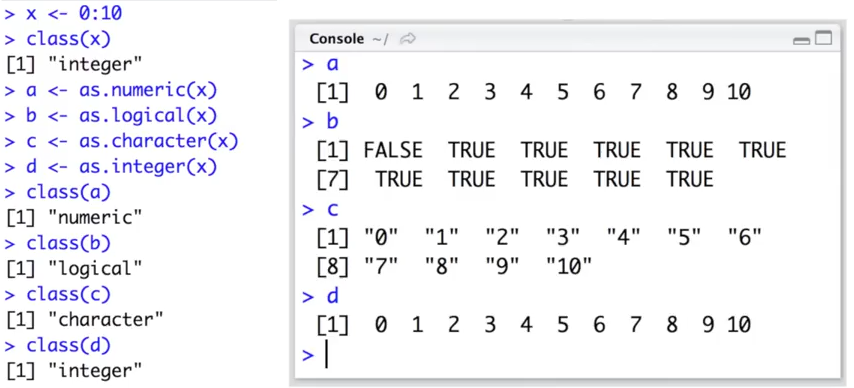
EL símnolo <- es un símbolo de asignación. La gramática del lenguaje determina si una expresión está completa o no, y solo se ve si al pulsar enter funciona o no. Una cosa importante es que con # puedes añadir comentarios al código que estamos escribiendo.

Cuando escribimos una expresión y pulsamos enter, se evalua y da un resultado. Un valor entrr [] indica el tamaño del vector (número de elementos). Si se genera un vector de tamaño 100 en el que se aumenta de 1 en 1 (x<-1:100).

La función *c()* genera vectores. Concatenando los elementos que encuentre dentro de un paréntesis. También sirve la función *vector()*. Se pueden crear vectores numéricos, lógicos, de caracteres, enteros y complejos. Podemos darnos cuenta que cuando creamos estos vectores son del mismo tipo. Cuando usamos la función *vector()* hay que decir el tipo y el tamaño.

Si combinas dos tipos distintos usando la función *c()* se lleva a cabo una función llamada coerción, en la que se genera un tipo común y en el que los elementos que no son de este tipo son convertidos. Los valores de letras no pueden ser convertidos a números, por lo que los números se convierten en caracteres.

La coerción tiene una serie de reglas, y con la coerción explícita se pueden usar funciones como ***as.\**** (*as.numeric(), as.logical(), as.character()* y *as.integer()*). Como su nombre indica, lo que hacen es convertir lo que le estas introduciendo al tipo que tiene en el nombre.



Crea cuatro vectores de distintos tipos en función al primero (integer es entero). Si intentamos meter caracteres de letras en un vector de tipo lógico, como no se pueden interpretar como valores de tipo verdadero o falso, lo que hace es dar valores NA (Not Available).

**Lección 3.3 – Matrices**

Nuestro siguiente tipo de objeto que vamos a aprender son las matrices, un vector particular con un atributo especial llamado dimensión, que indica el número de filas y columnas. Las matrices son creadas con la función *matrix()*, teniendo el primer argumento como argumentos y luego las dimensiones de filas y columnas.

*>m<-matrix(1:6, nrow=2, ncol=3)*

Si utilizamos la función *dim()* vemos la dimensión y si vemos *class()* vemos el tipo de vector. Las matrices se van rellenando columna a columna y de izquierda a derecha. Es importante, porque cuando queramos convertir un vector a matriz tenemos que tener en cuenta como se rellena.

Las matrices se pueden crear modificando el atributo de dimensión de un vector. Dado un vector de tamaño 1:10, si luego lo obligamos a convertirse en matriz con *dim()* y con c(2,5) se convierte en una matriz de 2 filas y 5 columnas. Otra manera de generar matrices en R es con las funciones es *cbind()* que pega vectores a manera de columnas y *rbind()* en este caso como filas. Uno de los requerimentos para crear una matriz es que este tenga el mismo número de elementos.

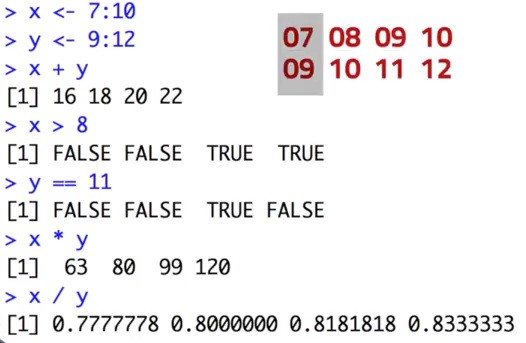
**Lección 3.4 – Precedencia Operaciones Vectoriales**

En esta lección aprenderemos las reglas de precedencia y las operaciones vectoriales. La división entera se representa con %%. Cuando introducimos paréntesis es más fácil ver el orden en el que se hacen las operaciones. Cuando metemos operaciones sin paréntesis, unas se llevarán a cabo antes que otras. Las reglas de precedencia son:

* Llamadas a funciones y expresiones agrupadas
* Operaciones de indexado y búsqueda
* Aritméticas
* Comparación
* Fórmulas
* Asignación
* Ayuda

Si utilizamos la ayuda de R para buscar la sintaxis, usamos el comando *?base:Sintax,* y da información de las reglas de preferencia. Como el vector es un elemento básico del lenguaje, hay operaciones sobre ellos que hace el código más conciso y fácil de leer.

Las operaciones vectoriales se hacen para comprender que se pueden hacer de manera paralela.



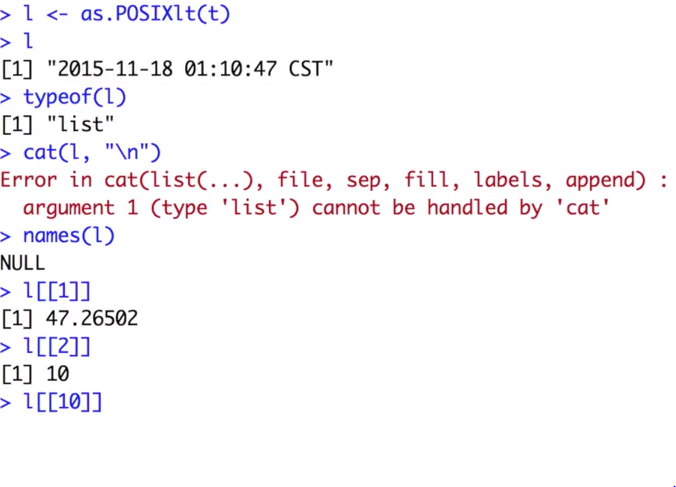
Podemos llevar a cabo operaciones vectoriales de un vector con un valor único, y lo mismo pasa si usamos el resto de operadores.

Las operaciones vectoriales se pueden hacer con matrices, teniendo el mismo atributo de dimensión, pudiendo sumarse, restarse, multiplicarse o dividirse elemento a elemento. La multiplicación matricial se puede llevar a cabo con el operador **%\*%**. Esto está definido muy diferente a multiplicar elemento a elemento, se multiplican aquellas con las mismas dimensiones cuadradas o cuando las columnas de la primera matriz coindicen con la segunda.

**Lección 3.5 – Manejo de Fechas de Tiempos**

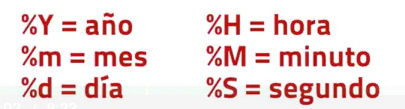
Ahora aprenderemos a manejar las fechas y tiempos en R, que puede ser bastante útil. Es importante saber que hay distintos archivos, que son los de tipo de dato POSIXct, que es el tiempo en segundos de 1970 (positivo para lo ocurrido después y negativo para lo de antes) o POSIXlt, que pueden ser ordenados de 1 a 9; segundos, minutos, horas, día del mes, mes del año, años desde 1900, día de la semana, día del año e indicador de horario de verano.

Una de las grandes dificultades es que la manera en la que se representa internamente es diferente a como se muestra en pantalla, lo que puede hacer que algunas conversiones sean complejas, teniendo siempre cuidado de que lo que estamos haciendo sea lo que queramos. Para obtener el tiempo actual podemos usar la función *Sys.time()*, que la podemos guardar en un objeto y verlo posteriormente con la función *typeof()* (Esta función es de tipo *double,* pero cuando se imprime se imprime como una cadena de texto). Usando las funciones *as.POSIXct()* y *as.POSIXlt()* podemos convertir el valor de tiempo en los diferentes formatos.



Podemos tratar de acceder a los nombres de la lista, pero estos son nulos, porque en la definición no existen los nombres de esta, pero si usamos dos corchetes podemos acceder a sus valores.

Si tuviéramos un conjunto de datos de tiempo y quisiéramos convertirlos a cadena para guardar la cadena en un archivo, podemos usar la función *strftime()*. Esta recibe como primer parámetro un objeto de tipo tiempo y la descripción de la cadena que queremos general, usando una anotación con porcentajes seguido del dato (año, mes…).



Si queremos hacer lo contrario, es decir, si de una cadena de caracteres se crea un objeto de tipo tiempo usamos la función *strptime()*, similar a la anterior, recibiendo un objeto, que es el que queremos convertir, y de segundo dato la cadena, ¿cómo está?, usando la misma especificiación entre comillas. Podemos hacer operaciones entre los archivos, como la diferencia en días o segundos. Primero se crea una fecha y se obtiene el tiempo después con *Sys.time()* seguido de la operación de los dos de manera decimal.

Además, también tenemos los datos Date, con la única diferencia de que se toman los datos en días y no en segundos como en los dos anteriores casos. Para convertir una cadena de caracteres en un tipo Date se usa la función *as.Date()*. Se puede usar el parámetro origin para definir una fecha de origen y poder llevar a cabo operaciones.

*PREGUNTA Si tuvieras la cadena “23-12-1977”, ¿Cuál es la forma para leerla como fecha?*

*Strptime(“23-12-77”, “%d-%m-%Y”)*

En este caso en el ejemplo la función *as.Date* y usa *origin* para definir una fecha de origen.

*>cumple<-as.Date(0,origin=”1977-12-25”)*

*>hoy<-Sys.Date()*

*>hoy – cumple*

Nos devuelve un texto “Time difference of 13842 days”, dando la diferencia en días. La función *format()* recibe un archivo de tipo fecha/tiempo y como segundo formato le indicamos como queremos que nos devuelva el formato de tipo carácter.

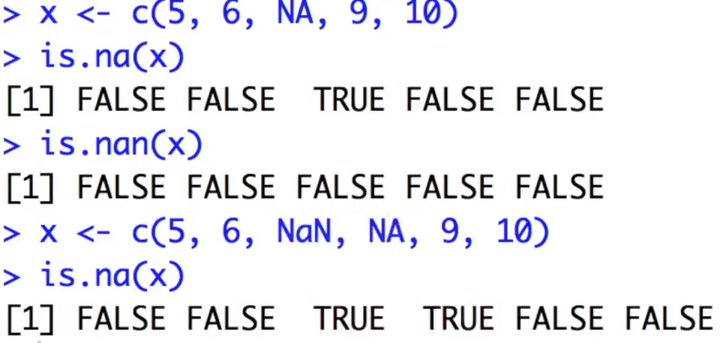
Se pueden hacer comparaciones entre distintos objetos de tipo Date cuando están en formato R. Por ejemplo, la comparación mayor o menor que tiene sentido cuando se usan dos fechas. Creamos dos fechas, una de ellas usando la fecha actual y luego sumándole un periodo de tiempo usando la función *as.difftime()*.

*>diff<-as.difftime(“00:30:00”, “%H:%M:%S”,units=”hour”)*

**Lección 3.6 – Listas, Factores, Valores Faltantes y Dataframes**

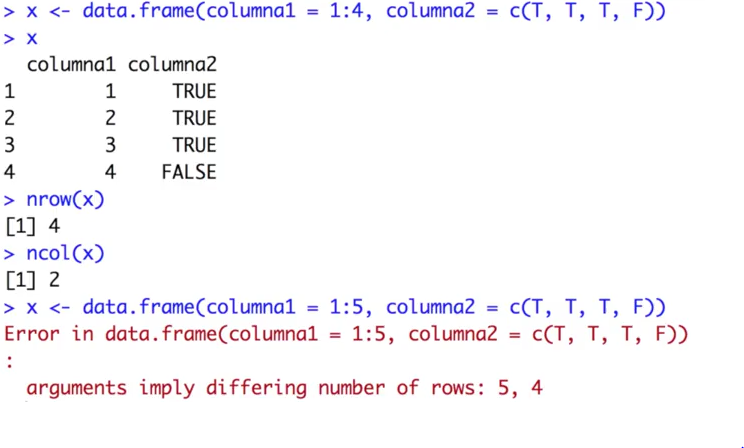
Otro de los objetos importantes en R son las listas, que son tipos especiales de vectores que pueden contener elementos de diferentes clases. La función *list()* recibe los elementos de esta lista. Los factores son un tipo especial de vectores de enteros, en la que cada entero está asignado a una etiqueta nominal. Podemos crear factores con la función *factor()*, siendo los niveles los distintos componentes. Otra cosa de este tipo de datos es que si usas la función *table()* los tabula con sus frecuenicas, y con la función *unclass()* a cada uno de los niveles va asociado un número entero, que va asociado a una etiqueta. El orden se puede establecer con el parámetro *level*, indicando cuál va primero y cuál va después.

Cuando hay un valor faltante, se usa un valor particular, que se indica con el valor NA o NaN dependiendo del tipo de valor que falta. Podemos probar si estos valores existen en nuestros objetos utilizando la función *is.na()*  o *is.nan()*, siendo importante conocer que NA no necesariamente tiene que ser un NaN, siendo ambos valores faltantes. Es importante que un valor NaN es un valor NA pero no al revés.



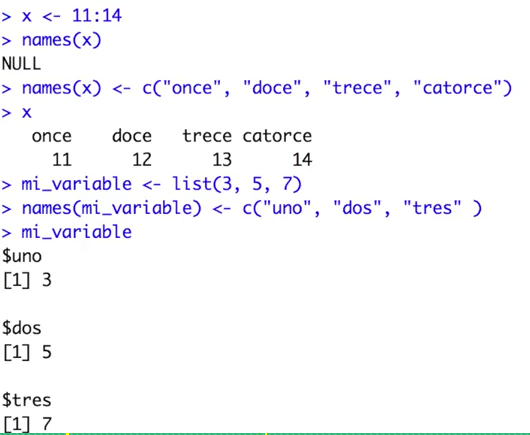
En resumen, NA y NaN son ambos valores faltantes, pero NaN viene de una operación matemática.

Los dataframes se representan en R como una lista particular, en la que cada elemento tiene un tamaño determinado. La forma tabular puede recordar a una matriz, pero en un data frame no tiene que tener el mismo número de columnas. Cada una de las columnas puede tener un nombre, al igual que cada uno de los renglones. Estos son conocidos como *row.names* y *col.names*. Ahora, para generar un dataframe, usamos la función *data.frame()* o las funciones *read.csv()* o *read.table()*, leyendo de una conexión a un archivo los datos y los mete de manera tabula. Se puede convierte una matriz en un data frame con la función *data.matrix()*.



El valor de la columna va seguido del valor de las filas. Podemos preguntar el número de renglones y las columnas. El tamaño de cada una de las columnas debe ser el mismo. En la parte inferior podemos ver el error que pasa si no son del mismo tamaño.

Podemos poner nombres a los objetos, un atributo que permite definir de manera exacta que es un renglón o una columna. Esto se hace con la función *names()*



Las matrices también pueden tener nombres, pero en este caso con la función *dimnames(m).* Esto se hace modificando un atributo asignando una lista a cada atributo en la que cada uno d elos elementos van a ser vectores de los tamaños de la matriz.

**Parte Práctica RStudio**

En R también se puede asignar del lado izquierdo: (180/6) – 15 -> mi\_variable

La variable se queda con la expresión calculada, es decir, un 15. Hay que tener en cuenta que todo lo que introduces en la consola es interpretado como un vector, por eso siempre da un valor después de [1]. **La función *print()* nos muestra el contenido de la variable**.

Para obtener la raíz cuadrada se usa la función *sqrt()* y para obtener el valor absoluto, se usa la función *abs()*. Como operadores lógicos, están el AND lógico (& y &&) y el OR lógico (| y ||).

En los dataframes, cada columna de éste puede tener un tipo diferente, pero cada fila debe tener la misma longitud.