**LECTURA FUNDAMENTAL**

|  |  |
| --- | --- |
| Título de la unidad | Tipos de datos y sus representaciones |
| Código del recurso | Copie el código aquí |

**Título**

**Introducción a los lenguajes de programación y tipos de datos**

**Contenido**

Contenido

[1 Introducción 2](#_Toc68515157)

[2 Instalación 3](#_Toc68515158)

[2.1 R y Rstudio 3](#_Toc68515159)

[2.2 Python 4](#_Toc68515160)

[3 Iniciando las tareas básicas en R 4](#_Toc68515161)

[3.1 Operaciones aritméticas 4](#_Toc68515162)

[3.2 Creación de objetos 5](#_Toc68515163)

[3.3 Operadores lógicos 5](#_Toc68515164)

[4 Tipos de datos en R 6](#_Toc68515165)

[5 Tibbles 7](#_Toc68515166)

[5.1 Creación de tibbles 8](#_Toc68515167)

[5.2 Subsetting tibbles 8](#_Toc68515168)

[6 Importación de datos 9](#_Toc68515169)

[7 Iteraciones Loops 9](#_Toc68515170)

[8 Declaraciones condicionales (conditional statements) 10](#_Toc68515171)

[9 Funciones 11](#_Toc68515172)

**Palabras clave**

R-Studio, Python, Programación

# Introducción

Es muy probable que, en algún momento de la vida, todos nos enfrentemos a una base de datos estructurada en una hoja de cálculo, como Microsoft Excel o Libre Office. Algunas de las tareas que usualmente se pueden hacer en estos softwares incluyen la gestión de registros, estadísticas descriptivas o modelos financieros y contables sencillos. La utilidad de estas herramientas es innegable. Sin embargo, no son la mejor opción cuando:

* Las bases de datos son demasiado grandes en observaciones o cantidad de variables.
* Sobre las bases de datos originales, se ejecutan tareas repetitivas
* Los procesos de transformación, representación y modelado involucran un proceso extenso con varios pasos.
* Se implementan pruebas estadísticas o modelos de minería de datos que no están incluidos en las herramientas del software o presentan muchas limitaciones.

Si el lector se encuentra con algunas de estas dificultades, puede considerar el uso de un lenguaje de programación de acceso libre como R o Python. Estas dos opciones constituyen una herramienta poderosa, en constante desarrollo. Por ende, son imprescindibles en el análisis cuantitativo en cualquier área de conocimiento. Algunas de las ventajas de estas herramientas son:

* Es fácil gestionar bases de datos grandes. De hecho, los procedimientos se organizan en forma de código y se ejecutan sin tener que manipular los datos directamente.
* Hay varias opciones para la automatización de tareas repetitivas. Por ejemplo, las instrucciones de un procedimiento completo se pueden almacenar en un objeto denominado funciónque ejecuta el mismo código cada vez que se ejecuta. También se pueden programar bucles fácilmente.
* Los procesos más complicados, es decir los que involucran más pasos, quedan bien documentados. A diferencia de las hojas de cálculo en los lenguajes de programación se tiene registro paso a paso, de cada una de las tareas ejecutadas.
* R y Python incluyen un conjunto de funciones nativas. Sin embargo, en todo el mundo la comunidad académica desarrolla y actualiza librerías que incrementan las posibilidades investigativas a través de modelos estadísticos o de minería de datos.

Ahora bien: al César lo que es del César, para ciertas tareas sencillas las hojas de cálculo pueden ser una opción más viable en términos de la relación beneficio/costo. No hay duda de la utilidad de una tabla dinámica, solo para citar un ejemplo.

Este módulo está centrado en el aprendizaje del lenguaje de programación R-studio. Sin embargo, se otorgarán algunos materiales adicionales para que el lector pueda familiarizarse con el lenguaje Python.

# Instalación

## R y Rstudio

Primero debe instalar el lenguaje de programación en su computador. Para completar esa tarea visite la siguiente página web: <https://www.icesi.edu.co/CRAN/> Allí puede descargar el instalador de la base del lenguaje de programación para los siguientes sistemas operativos: Linux, OS X (MAC) y Windows.

Posteriormente, puede instalar un editor de código. Le recomendamos utilizar R-Studio. La versión gratuita es ideal para el alcance de este módulo. El instalador puede descargarse en la siguiente página Web: <https://www.rstudio.com/products/rstudio/download/#download>

## Python

Existen varias maneras de instalar y trabajar con Python. Para las aplicaciones relacionadas con ciencia de datos, puede ser útil instalar la distribución Anaconda. Usted puede completar esa tarea descargando el instalador desde la siguiente página web: <https://www.anaconda.com/products/individual>

Una de las herramientas complementarias que instalará Anaconda es el notebook Jupyter en el cual se incluirán algunos materiales adicionales.

Nota flotante: el mejor manual de R y Python es Google. El lector puede pensar en alguna pregunta sobre un tema básico y luego googlearla. En muchos casos será redireccionado a los foros de stackoverflow. Vale la pena detenerse un poco en la lectura para inscribirse en ese portal y revisar materiales como libros, blogs de universidades y cursos gratuitos en línea que le ayudaran a reforzar su proceso de aprendizaje.

# Iniciando las tareas básicas en R

## Operaciones aritméticas

Usted puede iniciar haciendo algunas operaciones aritméticas básicas. Para eso, puede guiarse por la siguiente tabla de operadores:

|  |  |
| --- | --- |
| Operador Aritmético | Operación |
| + | Suma |
| - | Resta |
| / | División |
| %/% | División: solo retorna la parte entera de la respuesta |
| ^ | Exponente |
| %% | División: Devuelve el remanente de la división (módulo) |

## Creación de objetos

Para crear objetos (visibles en la sección de global evironment en R studio) puede usar el operador **<-** . Por ejemplo:

sum**<-**3**+**8

Tenga en cuenta que los nombres de los objetos creados solo pueden contener letras, números, guíon al piso (recomendable para separar palabras en vez de espacios).

## Operadores lógicos

A medida que se programa, es decir se construye un flujo de trabajo en R, se hará necesario utilizar operador lógicos, de acuerdo con la siguiente tabla:

|  |  |
| --- | --- |
| Operator | Description |
| < | less than |
| <= | less than or equal to |
| > | greater than |
| >= | greater than or equal to |
| == | exactly equal to |
| != | not equal to |
| !x | Not x |
| x | y | x OR y |
| x & y | x AND y |
| isTRUE(x) | test if X is TRUE |

# Tipos de datos en R

En R podemos trabajar con al menos 6 tipos de datos:

1. **Numéricos:**  si una variable contiene números será almacenada como numérica. Por ende, con ella se pueden hacer operaciones aritméticas.
2. **Enteros:** En R se pueden configurar variables que almacenan datos numéricos como enteros, cuando estos no contienen decimales. Un ejemplo común puede ser una variable que almacena el resultado de un conteo en el que no hacen sentido las partes decimales.
3. **Carácter:** Este tipo de dato se refiere al texto. Siempre que se usen letras, R almacenará datos como tipo carácter. Los números también se pueden forzar para que sean de tipo carácter. Por ejemplo, el número de cédula de una persona está definido en números, pero no representa una cantidad. En ese caso, es mejor configurar la variable como tipo carácter.
4. **Factor:** Ciertas variables categóricas solo pueden asumir valores definidos en un conjunto. Por ejemplo, en una base de datos de clientes de Bogotá, se genera una variable denominada localidad. En esta ciudad contamos con 20 localidades. Por ende, al momento de definir el tipo de variable en R se pueden establecer los posibles valores (levels) que ella puede asumir.
5. **Lógico:**  Una variable lógica puede asumir dos valores: Verdadero o Falso.
6. **Complejo:** En R se pueden crear expresiones con una parte real y una compleja.

Usted puede practicar abriendo un script en R-Studio para correr el siguiente código

# Numeric series:

numeric\_series**<-**C**(**3.5,2.9,6.8**)**

# Inspeccionar el tipo de variable

class**(**numeric\_series**)**

# Integer

integer\_vector**<-**c**(**4,8,6**)**

# Definir el tipo de variable (entero) con la función as.integer

integer\_vector**<-**as.integer**(**integer\_vector**)**

# Inspeccionar el tipo de variable

class**(**integer\_vector**)**

# carácter

character\_feature**<-**"creating my first character"

# Inspeccionar el tipo de variable

class**(**character\_feature**)**

# Factor

cities**<-**factor**(**c**(**"Bogotá","Medellín","Cali","Salento"**))**

# Inspeccionar el tipo de variable

class**(**cities**)**

# Inspeccionar las categorias (levels)

levels**(**cities**)**

# Lógico

sum**<-**4**+**8

res**<-**15**-**2

logi**<-**sum**<**res

# Inspeccionar el tipo de variable

class**(**logi**)**

# Complejo

complex\_number**<-**complex**(**real**=**7,imaginary**=**2)

# Inspeccionar el tipo de variable

class**(**complex\_number**)**

# Tibbles

Una base de datos se puede crear como un **dataframe**. Una función de la librería **tidyverse** permite crear la base de datos en un objeto denominado **tibble** que soluciona algunos inconvenientes comunes relacionados con la función data.frame.

**Nota flotante**: Tidyverse es una librería muy importante en el lenguaje de programación R. En realidad, es una colección de varias librerías para importación, transformación, limpieza y visualización. En este punto vale la pena pausar la lectura y revisar la documentación en la siguiente URL: <https://www.tidyverse.org/>

Para una exploración adicional de los tibbles, usted puede correr este código en R:

vignette**(**"tibble"**)**

## Creación de tibbles

Usted puede coercer un data frame a un tibble con la función as\_tibble(), incluida en el paquete tidyverse. Puede intentar este procedimiento, utilizando una base de datos de prueba denominada **mpg:**

cars\_info**<-**as\_tibble**(**mpg**)**

# Inspeccionar tipos de variables y primeras filas de datos

str**(**cars\_info**)**

# Inspeccionar tipos de variables,número de observaciones y encabezados de las variables

str**(**cars\_info**)**

# También puede intentarlo con la función glimpse

glimpse**(**cars\_info**)**

## Subsetting tibbles

Las bases de datos creadas s como tibbles en R, se pueden filtrar con los operadores: [$](https://rdrr.io/r/base/Extract.html), para extraer una sola variable por medio de su nombre. El corchete doble [[[](https://rdrr.io/r/base/Extract.html) puede extraer por nombre o posición.

database**<-**tibble**(**times**=**rexp**(**100**)**,capacities**=**rnorm**(**100,mean**=**100,sd**=**20**))**

#Extract one feature by using its name

time\_feature**<-**database**$**times

#Using double bracket

capcity\_feature**<-**database**[[**"capacities"**]]**

#Extracting by position

capacity\_feature2**<-**database**[[**2**]]**

Para filtrar por filas, se puede usar la función **fillter** que es parte del paquete de tidyverse.

library**(**nycflights13**)**

data\_planes**<-**planes

# Filtrar la base para incluir solo los aviones producidos por EMBRAER

data\_planes\_filter1**<-**data\_planes %>% filter**(**manufacturer**==**"EMBRAER"**)**

# Filtrar la base de datos para incluir los aviones producidos por EMBRAER O BOEING

selection**<-**c**(**"EMBRAER","BOEING"**)**

data\_planes\_filter2**<-**data\_planes %>% filter**(**manufacturer**==**selection**)**

# Filtrar la base de datos para incluir los aviones producidos por EMBARAER o BOEING con capacidad de 100 sillas o más

data\_planes\_filter3**<-**data\_planes %>% filter**(**manufacturer**==**selection **&** seats**>=**100**)**

En algunos casos, puede ser necesario hacer agrupaciones por los posibles valores de una variable categórica y, posteriormente, resumir en la cuenta el promedio o la desviación estándar. Este tipo de resúmenes son similares a los que se obtienen en las tablas dinámicas de las hojas de cálculo. Para completar esa tarea, es recomendable utilizar la función group\_by() que pertenece a la librería dplyr. Le recomendamos revisar el siguiente tutorial de la página oficial de tidyverse: <https://dplyr.tidyverse.org/reference/group_by.html>

# Importación de datos

Algunas de las funciones que el lector debe explorar a detalle son:

library **(**tidyverse**)** # este paquete incluye a otro específico para la importación conocido como readr()

library **(**readxl**)**

read\_csv**():** Lee archivos delimitados por coma.

read\_delim**()** Lee archivos separados por cualquier delimitador

read\_log **():** Lee archivos Apache

read\_excel**():** Lee archivos en Excel

Algunas de las ventajas de usar el paquete tidyverse para la importación, en comparación con las funciones de base de R son: rapidez de carga, información en la consola acerca del proceso de carga. Los documentos cargados se convierten automáticamente en un formato especial de *data frame* denominado *tibble* y finalmente, se debe tener en cuenta que este procedimiento facilita la “reproducibilidad”.



# Iteraciones Loops

Puede el lector pensar en un *tibble* en el que cada columna es una variable numérica diferente; y se requiere hacer una operación sobre cada una. En este caso con n variables, una iteración es una orden que repite el mismo procedimiento desde la variable i hasta j.

Cada iteración (loop), tiene tres componentes:

**Una salida (output):** es un elemento, usualmente vector, matriz, data frame o tibble, en el que se almacenara el conjunto de datos procesado o los resultados de algunos cálculos (por ejemplo, se calcula el promedio de cada variable o columna y se almacena en un vector).

**La secuencia,** que determina sobre qué elementos de la *data frame* o *tibble* operan las instrucciones que se van a implementar reiterativamente. Por ejemplo, en una iteracción que opera sobre columnas, la secuencia puede avanzar de a una columna o hacer saltos más grandes (cada dos o tres columnas).

**El cuerpo**: Esta es la parte del código que realiza el trabajo; es decir, se trata de las órdenes que se ejecutara a lo largo de un elemento en cuyos componentes involucrados en la secuencia se aplicarán las mismas funciones.

A continuación se presenta un ejemplo de una iteración usando un *for*. Lo que hace esta instrucción es repetir la misma instrucción; en este caso calculando la media

# usando loop for

flores**<-**datasets**::**iris

promedios**<-**vector**(**"numeric"**)**

length**(**flores**)**

**for** **(**i **in** seq**(**from**=**1,to**=**4**))** **{**

promedios**[**i**]<-**mean**(**flores**[**,i**])**

**}**

# Declaraciones condicionales (conditional statements)

Los operadores lógicos y relacionales se pueden usar para programar tareas condicionadas. Esto se puede hacer mediante la sentencia *if else.* Esta programación ordena que se ejecute una tarea siempre que se cumpla una condición.

**if** **(**condición**){**

expresión

**}**

Usando una base de datos de prueba, el siguiente código establece un tipo de flor diferente dependiendo de la relación que existe entre el largo y el ancho del sépalo en las 150 observaciones:

flores**<-**datasets**::**iris

flores**$**Sepal.Length\_mid**=**flores**$**Sepal.Length**/**1.5

flores**$**type**<-**ifelse**(**flores**$**Sepal.Length\_mid**>=**flores**$**Sepal.Width,"Type1","Type2"**)**

# Funciones

La mayor parte del trabajo de código en R consiste en usar funciones, muchas veces provenientes de paquetes adicionales a la base. Estas funciones simplifican el código, en tanto sintetizan un conjunto de tareas en una sola instrucción, que abarca pocas líneas de código.

Adicional a las funciones que están en la base de R o en los paquetes complementarios, el programador usualmente crea sus propias funciones: esta opción se debe considerar cuando se escriben más de dos veces el mismo código. Es decir, cuando se está haciendo una tarea en forma repetitiva. La función se asigna a un elemento de R y cada vez que se requiera hacer la misma tarea, en vez de reescribir todo el código, se escribe el nombre de la función indicando en cada caso cuál sería el argumento, ósea el objeto (data frame, tibble, list) sobre el que se van a aplicar las tareas.

# definir la función

potencia **<-** **function(**x, y**)** **{**

# function to print x raised to the power y

resultado **<-** x**^**y

print**(**paste**(**x,"elevado a la", y, "es", resultado**))**

**}**

#aplicar la función

**>**potencia**(**8, 2**)**

**[**1**]** "8 elevado a la 2 is 64"

**>** potencia**(**2, 8**)**

**[**1**]** "2 elevado a la 8 is 256"

*Nota: la función paste, se utiliza para unir textos.*

**Referencias**

[Son todas las bases bibliográficas o de información utilizadas dentro del texto (vídeos, películas, casos judiciales, libros, artículos, noticias entre otros). La citación y las referencias deben ir en normas APA. ]

**Referencias de figuras**

[En el caso de que haya utilizado figuras de libre uso debe referenciarla según el documento *Uso de figuras y tablas*.]

**Referencias de tablas**

[En el caso de que haya utilizado tablas de libre uso debe referenciarla según el documento *Uso de figuras y tablas*.]