Material Imprimible

Curso Data Analytics

Módulo 4: Análisis de datos con R

Contenido:

- ¿Qué es R?
- Fundamentos de R
- Transformación de datos con dplyr
- Visualización de datos con ggplot2

- **¿Qué es R?**

R es un lenguaje de programación y un entorno para el análisis estadístico y gráfico. R es parte del sistema GNU y se distribuye bajo la licencia GNU GPL; es decir, es software libre y gratuito. Es multiplataforma y se encuentra disponible para ser instalado en Windows, Linux y MAC

Sus principales ventajas son:

- Facilidad de uso
- Grandes colecciones de funciones para análisis de datos
- Integrable fácilmente a procesos de data mining
- De código abierto
- Posee una gran comunidad que lo respalda

Para aprovechar al máximo esta tecnología, instalaremos dos herramientas necesarias que se complementan entre sí.

- R Base: Es el lenguaje propiamente dicho que debe instalarse para ser utilizado. Incluye una simple consola para ejecutar comandos, pero no dispone de interfaz gráfica. Es la herramienta central donde se cargan nuestros métodos y funciones, y pueden ejecutarse. Siempre debe descargarse de la página oficial de R:
- R Studio: es la herramienta para acceder, ejecutar y visualizar resultados en R. Provee una interfaz gráfica amigable que nos permite interactuar con el lenguaje base instalado. Es gratuita, aunque posee versiones comerciales.

- **Fundamentos de R**

El lenguaje R lo introducimos dentro del espacio de trabajo en RStudio, o directamente en la consola. Por razones obvias recomendamos la primera opción, ya que permite armar un set de código más prolijo y documentado.

Las principales funciones que deben conocerse antes de arrancar con R, son las siguientes:

- Declaración de variables y valores globales

En R el concepto de variable hace referencia a valores que pueden ir tomando distintos valores y ser utilizados a lo largo de nuestro código. Dichas variables son palabras que definimos nosotros a medida que programamos.

Por ejemplo:

pais

Es una palabra, que a priori carece sentido, pero asignándole un valor toma relevancia. Para realizar dicha asignación existe el Operador de asignación " <- ", que se utiliza para asignarle a nuestra variable (lado izquierdo) el valor o dato que queramos (lado derecho)

pais <- "Argentina"

Ahora ya sabemos que si invocamos la variable país, ¡será Argentina!

Recuerda que estas variables están pensadas para que varíen según el contexto de nuestro código. Por ejemplo, podríamos definir que cambie según el origen del usuario del análisis, y cuando se loguee alguien de Venezuela, tome ese valor.

- Impresión de resultados en pantalla

Una vez que tenemos datos o valores cargados en nuestro código, puede ser necesario que tengamos que previsualizarlos o generar alguna salida. R nos permite hacer impresiones de pantalla que van a permitir visualizar el resultado en la consola.

print(pais)

Resultado en consola:

“Argentina”

- Realización de operaciones aritméticas

El lenguaje de programación R se puede utilizar para realizar operaciones aritméticas entre números reales y enteros, como por ejemplo la suma, multiplicación, resta y división.

Si introducimos el siguiente código:

Resultado <- 5 + 5

Resultado será 10, una vez que el codigo se ejecute.

Es importante mencionar que las operaciones también pueden ser definidas mediante el uso de paréntesis, como en la aritmética tradicional.

- Operadores lógicos

R nos permite utilizar operadores lógicos de manera similar que en SQL. De esta forma podremos validar condiciones de determinados campos a la hora de realizar nuestros análisis. Los principales operadores son los siguientes:

Operador	Descripción	Ejemplo
> >=	Valida si un dato es mayor o mayor igual que otro	10 > 12 Resultado será FALSO
< <=	Valida si un dato es menor o menor igual que otro	10 < 12 Resultado será VERDADERO
==	Igualación para validar si ambos valores son similares	1 == 1 Resultado será VERDADERO
!=	Valida si los valores son distintos	1 != 1 Resultado será FALSO

- Carga de datos y concepto de dataframe

R nos permite conectarnos a múltiples fuentes de datos, mediante diversos conectores. En este módulo introductorio hemos aprendido a cargar un archivo de valores separados por coma mediante la función read.csv

Sintaxis

```
# Por defecto coma (,) como separador y punto (.) como separador decimal
read.csv(file,                  # Nombre del archivo o ruta completa del archivo
    header = TRUE,                  # Leer el encabezado (TRUE) o no (FALSE)
    sep = ",",                  # Separador de los valores
    quote = "\"",               # Caracter de citaciones
    dec = ".",                  # Punto decimal
    fill = TRUE,                # Rellenar celdas vacías (TRUE) o no (FALSE)
    comment.char = "",              # Carácter de los comentarios o cadenas vacías
    encoding = "unknown")           # Codificación del archivo
```

Recuerda que según el archivo de origen, puede variar el encoding y el separador. Típicamente Excel genere los archivos CSV con separador ";" y "latin1" si está en español latinoamérica nuestro paquete Office.

A su vez, no todos los procesos de carga requieren que definamos todos los parámetros de la sintaxis mencionada. Sí es obligatorio definir el archivo a cargar y su ubicación "C:/Carpeta/Archivo.csv" o "Archivo.csv" si se encuentra en el mismo directorio que nuestro script R.

Una vez que entendemos cómo funciona la sintaxis de nuestro cargador de archivos csv, debemos guardar el mismo en nuestro entorno R. Para ello, generamos una variable y le asignamos nuestro conjunto de datos:

datosCargados <- read.csv("archivo.csv", sep = ";")

De ahora en más, todos nuestros datos estarán disponibles como datosCargados, que será identificado como un dataframe, un almacenamiento lógico tabular mediante filas y columnas en memoria.

Con los dataframe, podemos empezar a explorar nuevas funciones de análisis y transformación de datos.

- Paquetes en R

Antes de comenzar nuestra travesía en el análisis de datos con R, es importante mencionar que no todas las funciones se encuentran disponibles, y según diferentes necesidades, hay que instalarlas.

Un paquete es una colección de funciones que la comunidad dispone para que todos podamos usar. El universo de funciones para el análisis de datos es Tidyverse, y dentro de él se encuentran los principales paquetes para la ciencia de datos.

Nosotros en el curso aprenderemos a usar “dplyr” para la manipulación y análisis, y “ggplot2” para las visualizaciones.

Los paquetes deben instalarse desde la consola mediante el comando:

install.packages(“NombrePaquete”)

Ejemplo:
install.packages(“dplyr”)
install.packages(“ggplot2”)

Siempre recomendamos revisar bien que instalar, y hacerlo desde la consola de R que se conecta a los repositorios oficiales y seguros, lo cuál nos evita instalar archivos maliciosos que nos descarguemos de otras páginas.

Una vez instalados los paquetes, deben invocarse desde nuestra hoja de trabajo en R, ya que de esta forma habilitamos a que nuestro script pueda utilizar las funciones que dichos paquetes contienen.

library(ggplot2)

- **Transformación de datos con dplyr**

El paquete dplyr proporciona un conjunto de funciones extremadamente útiles para manipular data frames y así reducir el número de repeticiones, la probabilidad de cometer errores y el número de caracteres que hay que escribir.

El paquete dplyr utiliza el comando pipes %>% para combinar y generar un código más legible al momento de combinar funciones.

Las más utilizadas son:

- El valor “Datos Base” puede ser reemplazado por cualquier nombre, siempre y cuando se haga referencia a un dataframe cargado previamente.

Mutate:

Esta función permite generar nuevas columnas y ratios.

```
data_mutate <- mutate(DatosBase, ValorItem = Total / Cantidad,
                      Valor2 = Total*2)
```

En este caso, estamos generando un data frame nuevo denominado “data mutate” donde adicionamos dos columnas nuevas (ValorItem y Valor2) al dataframe “DatosBase”. El contenido de estas columnas refleja el cálculo indicado en las operaciones.

IdProducto	Total	CiudadTienda	PaisTienda	ValorItem	Valor2
MAT-AL-10002116	19187	Veracruz	México	3197.8333	38374
MOB-MO-10004698	29369	Veracruz	México	2097.7857	58738
MAT-AR-10001651	3276	São Paulo	Brasil	273.0000	6552
MAT-GR-10000916	29864	São Paulo	Brasil	3733.0000	59728
TEC-TE-10001721	26460	São Paulo	Brasil	1102.5000	52920
MAT-PA-10000211	22003	São Paulo	Brasil	2750.3750	44006
MOB-LI-10000060	12966	San Luis Potosí	México	2161.0000	25932

Select:

Esta función nos permite seleccionar solo las columnas que necesitemos para realizar nuestros análisis

```
data_select <- select(DatosBase, PaisTienda,
                      Total,
                      ValorItem)
```

Del dataframe inicial “DatosBase” selecciono los campos “Pais Tienda”, “Total” y “Valor Item” solamente.

PaisTienda	Total	ValorItem
México	19187	3197.8333
México	29369	2097.7857
Brasil	3276	273.0000
Brasil	29864	3733.0000
Brasil	26460	1102.5000
Brasil	22003	2750.3750
México	12966	2161.0000
México	3279	1639.5000

Filter:

Esta función nos permite filtrar filas de nuestro dataframe, pudiendo definir condiciones a cumplir a los datos

data_filter <- filter(DatosBase, Total > 10000)

Todas las filas que retorna dicha acción, contienen el campo Total con un valor superior a 10000

Fecha.del.pedido	IdProducto	Total	CiudadTienda	PaisTienda
01/01/2017 0:00	MAT-AL-10002116	19187	Veracruz	México
01/01/2017 0:00	MOB-MO-10004698	29369	Veracruz	México
01/12/2016 0:00	MAT-GR-10000916	29864	São Paulo	Brasil
01/12/2016 0:00	TEC-TE-10001721	26460	São Paulo	Brasil
01/12/2016 0:00	MAT-PA-10000211	22003	São Paulo	Brasil
21/12/2016 0:00	MOB-LI-10000060	12966	San Luis Potosí	México
27/12/2016 0:00	MOB-MO-10001228	19761	Panamá	Panamá

Summarise:

Esta función nos permite generar resúmenes de datos, y se complementa con funciones de agregación, como el contar (count), sumar (sum) o mediana (median)

data_summ <- summarise(DatosBase, ValorSumado = sum(Total))

	ValorSumado
1	180342107

De esta forma, resumimos nuestros datos a un nuevo único campo (“ValorSumado”) donde tendremos sumarizado todos los valores existentes en el campo “Total”

Group By:

Permite definir criterios de agregación para analizar los resúmenes realizados. Por ejemplo, si defino el país como agrupador y luego se realiza un resumen con la sumatoria de los totales, tendré como resultado la Cantidad Total por país.

agrupado <- DatosBase%>% group_by(PaisTienda) %>% summarise(ValorSumado = sum(Total))

Como se puede observar, el group by funciona con el operador pipes presentado anteriormente.

A su vez, al combinarse con las otras funciones, se define el dataframe inicialmente (En este ejemplo, llamado “DatosBase”) y en las distintas funciones no es necesario volver a mencionarlo.

PaisTienda	ValorSumado
Argentina	7289136
Barbados	457495
Bolivia	876002
Brasil	27660717
Chile	2348657
Colombia	6174527
Cuba	12397877
Ecuador	916326
El Salvador	12993286
Guatemala	9055549
Haití	1806176
Honduras	12453460
Jamaica	585634
México	46501649
Nicaragua	10952708

Se pueden combinar todas estas funciones en una sólo función, mediante el operador pipes mencionado anteriormente:

```
agrupado <- data %>% group_by(PaisTienda) %>% summarise(ValorSumado = sum(Total))
%>% filter(PaisTienda == "Argentina")
```

En este caso, la agrupación resultante de la sumarización está acotada solamente a "Argentina":

	PaisTienda	ValorSumado
1	Argentina	7289136

- **Visualización de datos con ggplot2**

A diferencia de los gráficos con el paquete base donde creamos un gráfico a base de pasos sucesivos, ggplot2 se basa en una gramática de gráficos, añadiendo elementos a un graphical device , donde distintos componentes independientes se pueden combinar de muchas maneras diferentes.

Se compone de tres elementos centrales:

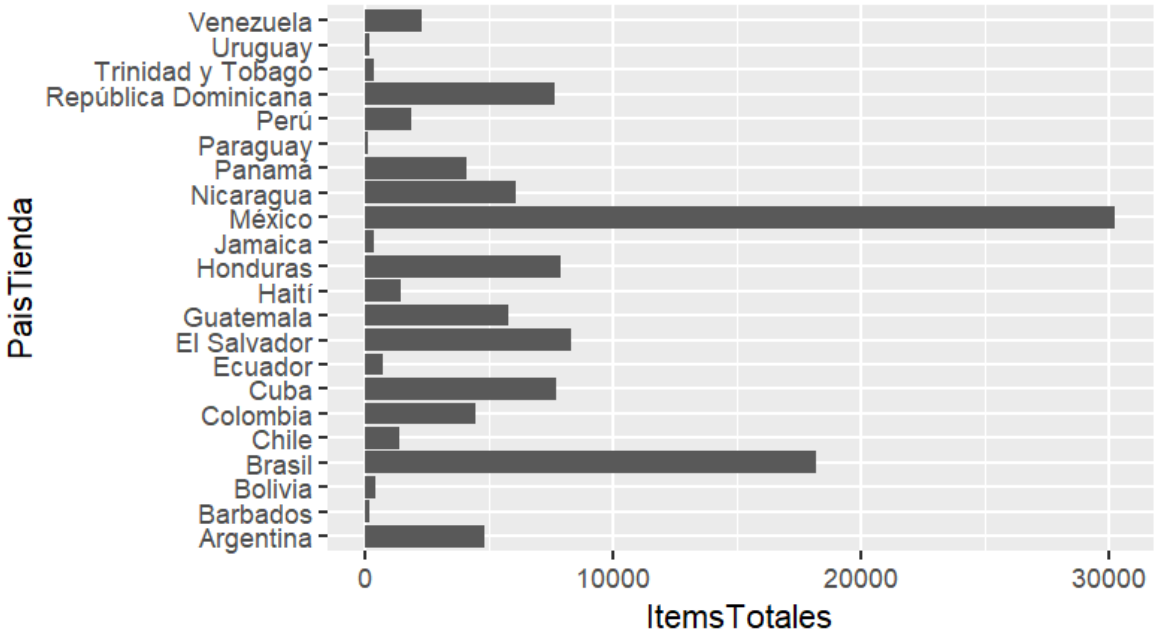
- data: es el data frame que utilizaremos para ingestar los datos
- aesthetics o aes: son los parámetros fundamentales de definición de estructura y composición, como los ejes, escalas, etc.
- geometry o geom: es la definición de la figura geométrica a visualizar (barras, histogramas, líneas, etc)

La sintaxis para construir un gráfico de barras es la siguiente:

```
DatosBase %>% group_by(PaisTienda) %>% summarise(ItemsTotales = sum(Cantidad))
%>% ggplot(aes(x=ItemsTotales, y=PaisTienda)) + geom_col()
```

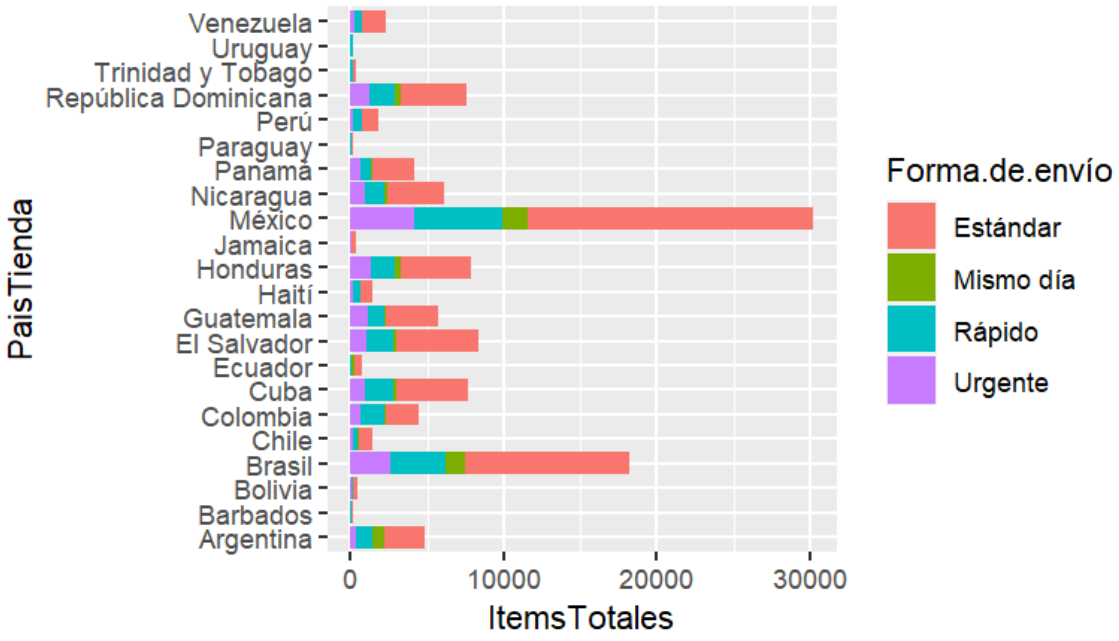
Usamos dplyr para manipular nuestro dataframe inicial, y definimos una agrupación de ItemsTotales por PaisTienda. Luego, mediante el operador pipes invoco a la funcion ggplot que me permite realizar la visualización.

El parámetro data ya esta definido, dado que se obtiene mediante la definición inicial del dataframe. En la aesthetic defino los ejes y luego, coloco un + geom_Col para identificar el tipo de figura que quiero visualizar, en este caso un gráfico de barras.



También, podemos alterar los colores o realizar una doble agrupación para poder convertir nuestras barras simples en barras apiladas:

DatosBase %>% group_by(PaisTienda, Forma.de.envío) %>% summarise(ItemsTotales = sum(Cantidad)) %>% ggplot(aes(x=ItemsTotales, y=PaisTienda, fill = Forma.de.envío)) + geom_col()



En este caso, hemos utilizado el parámetro fill para definir cuál es el campo que utilizaremos para segmentar nuestras barras. No nos olvidemos de incluirlo en el group by, para que dplyr lo tenga en cuenta a la hora de realizar las agregaciones.