# Cargando archivos de datos en R

## Sesión 5

Natalie Julian - www.nataliejulian.com

Estadística UC y Data Scientist en Zippedi Inc.

# Big Data

*El Big Data consiste en grandes cantidades de datos complejos y de variada naturaleza, los que requieren de tecnología y técnicas avanzadas para su creación, almacenamiento, distribución, acceso, protección y análisis.*

# Ejemplos de casos en Big Data

- Geolocalización continua para la trazabilidad de COVID-19
- Vigilancia y seguridad a través de cámaras en sucursales de un banco
- Transacciones realizadas mediante tarjetas de crédito
- Seguimiento de pedidos de Uber Eats
- Reconocimiento de voz en llamadas de servicio al cliente

Dependiendo de la naturaleza de la información, existen distintos tipos de datos.

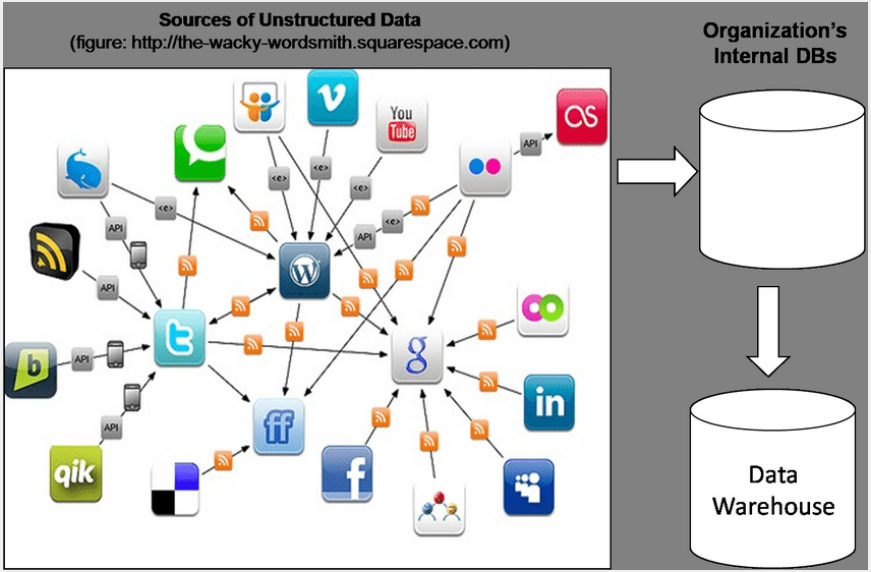
# Unstructured Data

Datos que no tienen un formato o estructura predefinidos. Análisis requerido es de alta complejidad.

**Ejemplos**

- Imágenes de toma continua para análisis metereológico
- Análisis de audios de WhatsApp para la búsqueda de evidencia criminal
- Reconocimiento facial de actores en películas o escenas

# Data Warehouse



# Actual Data

Con la revolución de datos actual, aproximadamente el 80% de los datos corresponden a datos no estructurados.

# Actual Data

Con la revolución de datos actual, aproximadamente el 80% de los datos corresponden a datos no estructurados.

El 20% restante corresponde a datos estructurados.

# Structured Data

Datos que tienen formato y estructura determinada, fácil de almacenar y analizar.

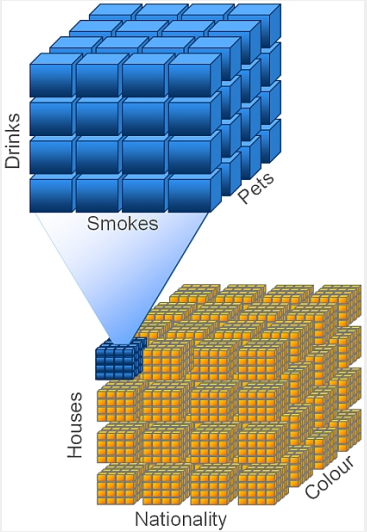
**Ejemplos**

- Historial de índices económicos
- Estadísticas de personajes en League of Legends
- Información de seguimiento de pacientes

# Tablas de datos

| | Rank | Pupil | Total | Subject | Score |
|---|---|---|---|---|---|
| 1 | 3 | Avinash | 300.00 | English | 50.00 |
| 2 | 3 | Avinash | 300.00 | Maths | 60.00 |
| 3 | 3 | Avinash | 300.00 | Science | 70.00 |
| 4 | 3 | Avinash | 300.00 | History | 54.00 |
| 5 | 4 | Ravi | 320.00 | English | 56.00 |
| 6 | 4 | Ravi | 320.00 | Maths | 60.00 |
| 7 | 4 | Ravi | 320.00 | Science | 60.00 |
| 8 | 4 | Ravi | 320.00 | History | 64.00 |
| 9 | 5 | Dipti | 320.00 | English | 56.00 |
| 10 | 5 | Dipti | 320.00 | Maths | 65.00 |
| 11 | 5 | Dipti | 320.00 | Science | 49.00 |
| 12 | 5 | Dipti | 320.00 | History | 80.00 |
| 13 | 7 | Sumedh | 250.00 | English | 56.00 |
| 14 | 7 | Sumedh | 250.00 | Maths | 65.00 |
| 15 | 7 | Sumedh | 250.00 | Science | 49.00 |
| 16 | 7 | Sumedh | 250.00 | History | 80.00 |

# Cubo OLAP




*OLAP:* On-Line Analytical Processing

# Datos estructurados en R

En este curso trabajaremos con datos estructurados en R. Aprenderemos a importar estos datos contenidos en archivos externos.

# Importando datos en R

Ya aprendimos cómo definir manualmente tablas de datos y matrices en R. Sin embargo, no podemos definir manualmente miles de registros (sería una locura).

Algunos formatos de archivos de datos son:

- Archivo de valores de Microsoft Excel *.csv*
- Hoja de cálculo de Microsoft Excel *.xlsx*
- Documento de texto *.txt*
- Bases de datos de otros softwares, por ejemplo: Stata, SPSS, SAS.

# Información nutricial de productos de Mcdonald

El archivo `menu` contiene información de distintos productos de la cadena Mcdonald. Vamos a importar estos datos en R.

# ¿Qué necesitamos para cargar datos en R?

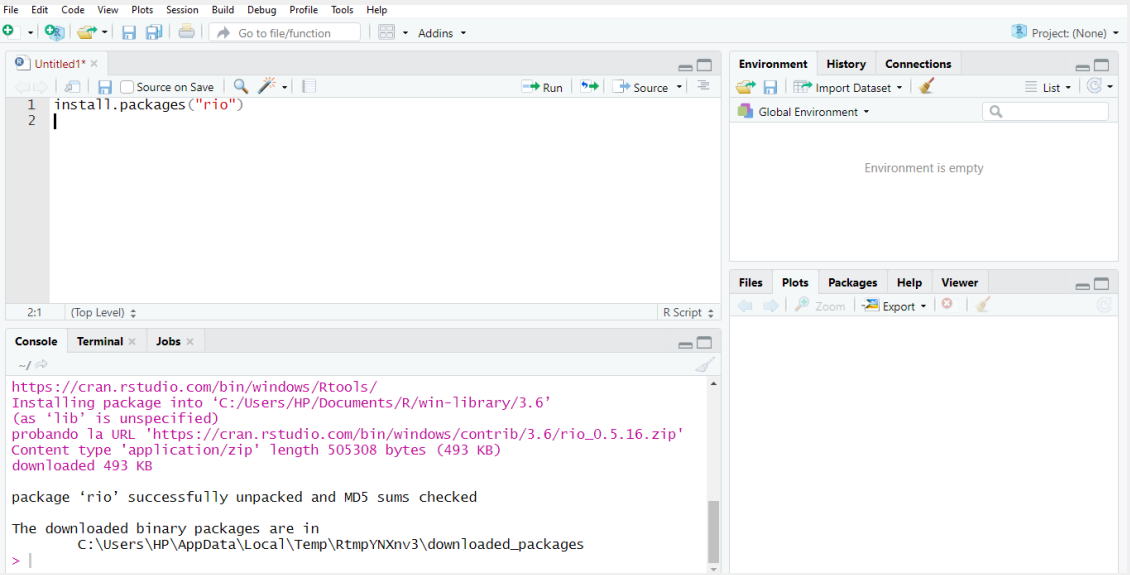
Muchas veces es necesario instalar *paquetes* para acceder a funciones adicionales que no se encuentran por defecto en R. Algunas funciones base de R son *sum()*, *sqrt()*, *data.frame()*, *cbind()*, entre otras que hemos utilizado sin necesidad de instalar nada adicional.

# ¿Cómo instalar un paquete?

Para cargar los datos instalaremos el paquete `rio` corriendo la siguiente línea de código:

```
install.packages("rio")
```

# Vista en R

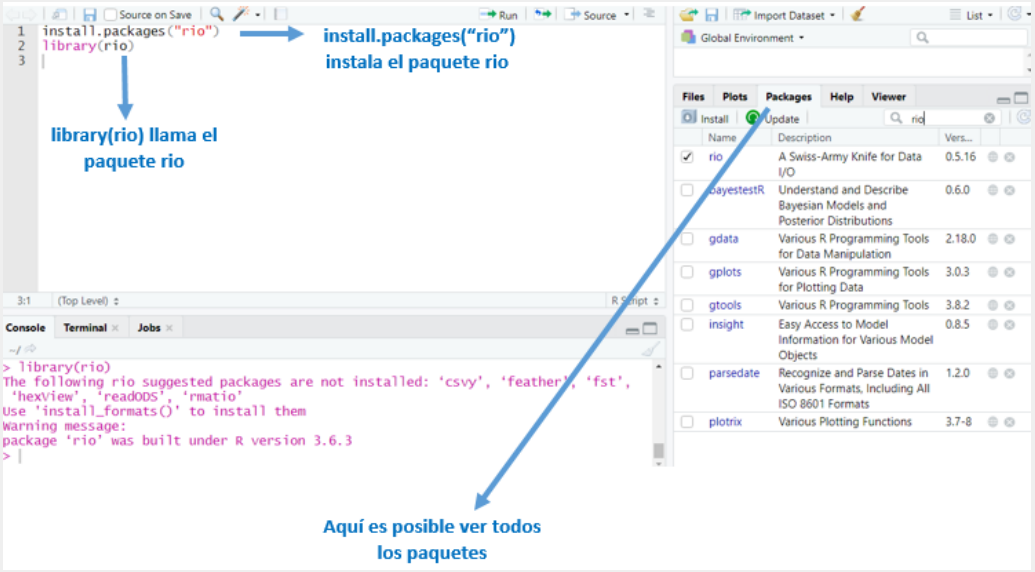

File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Untitled1*
Source on Save
Run
Source
1 install.packages("rio")
2
2:1 (Top Level)
R Script
Environment History Connections
Import Dataset
List
Global Environment
Environment is empty
Files Plots Packages Help Viewer
Zoom
Export
Console Terminal Jobs
~/
https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/HP/Documents/R/win-library/3.6'
(as 'lib' is unspecified)
probando la URL 'https://cran.rstudio.com/bin/windows/contrib/3.6/rio_0.5.16.zip'
Content type 'application/zip' length 505308 bytes (493 KB)
downloaded 493 KB
package 'rio' successfully unpacked and MD5 sums checked
The downloaded binary packages are in
C:\Users\HP\AppData\Local\Temp\RtmpYNXnv3\downloaded_packages
>

# Cargando un paquete

Una vez que se ha instalado el paquete debemos llamarlo o cargarlo, utilizando lo siguiente:

```
library(rio)
```

# Vista en R


install.packages("rio")
library(rio)
install.packages("rio")
instala el paquete rio
library(rio) llama el paquete rio
Aquí es posible ver todos los paquetes
> library(rio)
The following rio suggested packages are not installed: 'csvy', 'feather', 'fst', 'hexView', 'readODS', 'rmatio'
Use 'install_formats()' to install them
Warning message:
package 'rio' was built under R version 3.6.3
Name
Description
rio
A Swiss-Army Knife for Data I/O
BayestestR
Understand and Describe Bayesian Models and Posterior Distributions
gdata
Various R Programming Tools for Data Manipulation
gplots
Various R Programming Tools for Plotting Data
gtools
Various R Programming Tools
insight
Easy Access to Model Information for Various Model Objects
parsedate
Recognize and Parse Dates in Various Formats, Including All ISO 8601 Formats
plotrix
Various Plotting Functions

# Importar los datos menu

```
install.packages("rio") #Instala el paquete o librería rio
library(rio) #Carga o llama el paquete rio

?import #Usaremos la función import del paquete rio

#Los datos se guardaran en un objeto llamado datos

datos<-import(file.choose())

#file.choose abre una ventana para seleccionar la ubicación del archivo

View(datos) #Vemos una vista previa de los datos
```

# Vista de los datos

Untitled1* × datos ×

Filter

| | Category | Item | Serving Size | Calories | Calories from Fat | Total Fat | Total Fat (% Daily Value) | Sat Fat |
|---|---|---|---|---|---|---|---|---|
| 1 | Breakfast | Egg McMuffin | 4.8 oz (136 g) | 300 | 120 | 13 | 20 | 5.0 |
| 2 | Breakfast | Egg White Delight | 4.8 oz (135 g) | 250 | 70 | 8 | 12 | 3.0 |
| 3 | Breakfast | Sausage McMuffin | 3.9 oz (111 g) | 370 | 200 | 23 | 35 | 8.0 |
| 4 | Breakfast | Sausage McMuffin with Egg | 5.7 oz (161 g) | 450 | 250 | 28 | 43 | 10.( |
| 5 | Breakfast | Sausage McMuffin with Egg Whites | 5.7 oz (161 g) | 400 | 210 | 23 | 35 | 8.0 |
| 6 | Breakfast | Steak & Egg McMuffin | 6.5 oz (185 g) | 430 | 210 | 23 | 36 | 9.0 |
| 7 | Breakfast | Bacon, Egg & Cheese Biscuit (Regular Biscuit) | 5.3 oz (150 g) | 460 | 230 | 26 | 40 | 13.( |

# Análisis exploratorio

a) ¿De cuántos productos se tiene información en los datos?

b) ¿Cuántas categorías de productos hay? ¿Cómo se distribuyen los platos por categoría? ¿Cuáles son las categorías con más productos ofrecidos en la carta?

c) ¿Cómo se distribuye el tamaño de la porción en la categoría *Breakfast*? ¿Cuáles son los productos de desayuno con mayor y menor tamaño de porción?

# a) ¿De cuántos productos se tiene información?

```
length(unique(datos$Item)) #Cuenta los nombres de productos que hay
[1] 259

head(unique(datos$Item), 12) #Muestra el nombre de 12 platos
 [1] "Egg McMuffin"
 [2] "Egg White Delight"
 [3] "Sausage McMuffin"
 [4] "Sausage McMuffin with Egg"
 [5] "Sausage McMuffin with Egg Whites"
 [6] "Steak & Egg McMuffin"
 [7] "Bacon, Egg & Cheese Biscuit (Regular Biscuit)"
 [8] "Bacon, Egg & Cheese Biscuit (Large Biscuit)"
 [9] "Bacon, Egg & Cheese Biscuit with Egg Whites (Regular Biscuit)"
[10] "Bacon, Egg & Cheese Biscuit with Egg Whites (Large Biscuit)"
[11] "Sausage Biscuit (Regular Biscuit)"
[12] "Sausage Biscuit (Large Biscuit)"
```

# b) ¿Cuántas categorías de productos hay? ¿cuántos productos caen en ada una?

```
length(unique(datos$Category)) #Cuenta los nombres de categorías de hay
[1] 9

table(datos$Category) #Cuenta cuántos productos caen en las categorías

       Beef & Pork          Beverages          Breakfast     Chicken & Fish
                15                 27                 41                 27
      Coffee & Tea           Desserts             Salads Smoothies & Shakes
                95                  7                  6                 28
    Snacks & Sides
                13
```

Si queremos guardar la información de esta tabla en un objeto más manejable podemos hacer lo siguiente...

# Guardando una tabla de frecuencias en una dataframe

```
df<-data.frame(table(datos$Category)) #Guarda los datos de la tabla de manera vectorizada

head(df) #El formato es claramente diferente
             Var1 Freq
1    Beef & Pork   15
2      Beverages   27
3      Breakfast   41
4 Chicken & Fish   27
5   Coffee & Tea   95
6       Desserts    7

df[which.max(df$Freq),] #¿Cuál categoría tiene más productos?
          Var1 Freq
5 Coffee & Tea   95

df[which.min(df$Freq),] #¿Cuál categoría tiene menos productos?
    Var1 Freq
7 Salads    6

df[order(df$Freq, decreasing=TRUE),] #Muestra las categorías ordenadas de forma decreciente por cantidad de productos
                Var1 Freq
5       Coffee & Tea   95
3          Breakfast   41
8 Smoothies & Shakes   28
2          Beverages   27
4     Chicken & Fish   27
1        Beef & Pork   15
9     Snacks & Sides   13
6           Desserts    7
7             Salads    6
```

# c) ¿Cómo se distribuyen los tamaños de porciones en la categoría Breakfast?

Notemos primero, el formato de la variable *Serving Size* (tamaño de porción):

```
datos[which(datos$Category=="Breakfast"), 'Serving Size'] #Tamaño de la porcion codificada
 [1] "4.8 oz (136 g)"  "4.8 oz (135 g)"  "3.9 oz (111 g)"  "5.7 oz (161 g)"
 [5] "5.7 oz (161 g)"  "6.5 oz (185 g)"  "5.3 oz (150 g)"  "5.8 oz (164 g)"
 [9] "5.4 oz (153 g)"  "5.9 oz (167 g)"  "4.1 oz (117 g)"  "4.6 oz (131 g)"
[13] "5.7 oz (163 g)"  "6.2 oz (177 g)"  "5.9 oz (167 g)"  "6.4 oz (181 g)"
[17] "5 oz (143 g)"    "5.5 oz (157 g)"  "7.1 oz (201 g)"  "6.1 oz (174 g)"
[21] "6.3 oz (178 g)"  "5 oz (141 g)"    "7.1 oz (201 g)"  "7.2 oz (205 g)"
[25] "6.9 oz (197 g)"  "7.1 oz (201 g)"  "8.5 oz (241 g)"  "9.5 oz (269 g)"
[29] "10 oz (283 g)"   "9.6 oz (272 g)"  "10.1 oz (286 g)" "14.8 oz (420 g)"
[33] "15.3 oz (434 g)" "14.9 oz (423 g)" "15.4 oz (437 g)" "5.3 oz (151 g)"
[37] "6.8 oz (192 g)"  "3.9 oz (111 g)"  "4 oz (114 g)"    "9.6 oz (251 g)"
[41] "9.6 oz (251 g)"
```

No está aún trabajable del todo. Lo que podemos hacer es extraer el tamaño de la porción en gramos con la función `substr`, sólo debemos especificar bien el `start` y `stop` de la extracción...

# c) ¿Cómo se distribuyen los tamaños de porciones en la categoría Breakfast?

```
substr(datos[which(datos$Category=="Breakfast"), 'Serving Size'],
start = nchar(datos[which(datos$Category=="Breakfast"), 'Serving Size'])-5,
stop=nchar(datos[which(datos$Category=="Breakfast"), 'Serving Size'])-3)
 [1] "136" "135" "111" "161" "161" "185" "150" "164" "153" "167" "117" "131" "163" "177"
[15] "167" "181" "143" "157" "201" "174" "178" "141" "201" "205" "197" "201" "241" "269"
[29] "283" "272" "286" "420" "434" "423" "437" "151" "192" "111" "114" "251" "251"

#Extramos las medidas en gramos, pero... no están en formato numérico!!


sizebreakfast<-as.numeric(substr(datos[which(datos$Category=="Breakfast"), 'Serving Size'],
start = nchar(datos[which(datos$Category=="Breakfast"), 'Serving Size'])-5,
stop=nchar(datos[which(datos$Category=="Breakfast"), 'Serving Size'])-3))

#Ahora sí! con as.numeric() lo pasamos a formato numérico

range(sizebreakfast) #Rango de valores en gramos
[1] 111 437
```

# ¿Cuáles son los productos con porciones más pequeñas y más grandes?

Tenemos la información del tamaño de las porciones, unifiquemosla con la información del nombre de los productos:

```
productosbreakfast<-data.frame(datos$Item[which(datos$Category=="Breakfast")], sizebreakfast)

names(productosbreakfast)<-c("Producto", "Porcion") #Le añado nombres a las columnas

head(productosbreakfast)
                          Producto Porcion
1                     Egg McMuffin     136
2                Egg White Delight     135
3                 Sausage McMuffin     111
4        Sausage McMuffin with Egg     161
5 Sausage McMuffin with Egg Whites     161
6             Steak & Egg McMuffin     185

#¿Cuál es el producto en el desayuno con menor porcion en gramos?

productosbreakfast[which.min(productosbreakfast$Porcion),] #Sausage McMuffin con 111 gramos

          Producto Porcion
3 Sausage McMuffin     111

#¿Cuál es el producto en el desayuno con mayor porcion en gramos?

productosbreakfast[which.max(productosbreakfast$Porcion),] #Big Breakfast with Hotcakes and Egg Whites con 437 gramos

Producto Porcion
35 Big Breakfast with Hotcakes and Egg Whites (Large Biscuit)     437
```

# Estadísticas de las porciones

```
#¿Cuáles son las estadísticas del tamaño de la porcion de estos productos?

summary(productosbreakfast$Porcion)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  111.0   151.0   177.0   204.7   241.0   437.0

#¿Qué productos están bajo la mediana en tamaño de porcion en gramos?

productosbreakfast$Producto[which(productosbreakfast$Porcion<median(productosbreakfast$Porcion))]
[1] Egg McMuffin
 [2] Egg White Delight
 [3] Sausage McMuffin
 [4] Sausage McMuffin with Egg
 [5] Sausage McMuffin with Egg Whites
 [6] Bacon, Egg & Cheese Biscuit (Regular Biscuit)
 [7] Bacon, Egg & Cheese Biscuit (Large Biscuit)
 [8] Bacon, Egg & Cheese Biscuit with Egg Whites (Regular Biscuit)
 [9] Bacon, Egg & Cheese Biscuit with Egg Whites (Large Biscuit)
[10] Sausage Biscuit (Regular Biscuit)
[11] Sausage Biscuit (Large Biscuit)
[12] Sausage Biscuit with Egg (Regular Biscuit)
[13] Sausage Biscuit with Egg Whites (Regular Biscuit)
[14] Southern Style Chicken Biscuit (Regular Biscuit)
[15] Southern Style Chicken Biscuit (Large Biscuit)
[16] Bacon, Egg & Cheese McGriddles
[17] Sausage McGriddles
[18] Hotcakes
[19] Sausage Burrito
[20] Cinnamon Melts
```