MapReduce

(Este documento es por el momento un borrador de trabajo. No hay inconveniente en difundirlo, pero la condición de borrador se ha de tener en cuenta al emitir valoraciones sobre el mismo, y sobre todo se agradecen sugerencias de todo tipo, en mi dirección de email, *cpareja@ucm.es (mailto:cpareja@ucm.es)*.)

Breve recordatorio de map

Queremos calcular los n primeros términos de la sucesión $a_i = \frac{i^2}{i+1}$.

Por ejemplo, para los cinco primeros términos, te ndríamos la lista siguiente:

$$[\frac{1}{2}, \frac{2^2}{3}, \frac{3^2}{4}, \frac{4^2}{5}, \frac{5^2}{6}]$$

Podemos hacerlo con la notación intensional o con la función `map`:

In [1]:

```
def termino(i):
    return i**2/(i+1)

def lista_de_terminos(n):
    return [i**2/(i+1) for i in range(1, n+1)]

print(lista_de_terminos(5))

def lista_de_terminos(n):
    return list(map(lambda i: termino(i), range(1, n+1)))

print(lista_de_terminos(5))
```

```
[0.5, 1.3333333333333333, 2.25, 3.2, 4.166666666666667]
[0.5, 1.3333333333333333, 2.25, 3.2, 4.166666666666667]
```

Breve recordatorio de reduce

Si quisiéramos ahora sumar todos los elementos de una lista, bastaría con insertar la función suma entre sus elementos. Esto se puede hacer con la función `reduce`:

In [2]:

```
from functools import reduce

def suma(lista):
    return reduce(lambda a, b: a+b, lista)

mis_terminos = lista_de_terminos(5)
mi_sumatorio = suma(mis_terminos)

print(mi_sumatorio)
```

```
11.45
```

Primera versión de la técnica map-reduce

La técnica de map-reduce consiste en resolver un problema mediante la combinación sucesiva de estas dos operaciones, `map` y `reduce` :

$$\sum_{i=1}^{n} a_i = \frac{i^2}{i+1}$$

In [*]:

```
def sumatorio(n):
    return suma(lista_de_terminos(n))

print(sumatorio(5))
```

```
11.45
```

Observa que la clave ha sido encontrar las funciones mapeadora

$$\lambda i \rightarrow a_i = \frac{i^2}{i+1}$$

y la función reductora:

$$\lambda a, b \rightarrow a + b$$

Segunda versión de la técnica map-reduce

Queremos contabilizar cuántas veces aparece el carácter `a` en un texto, cuántas, el `b` , etc.

Versión 2 del mapper. En realidad, el mapper toma cada elemento (una línea) de la lista de entrada y devuelve una lista de pares, clave-valor en un dominio diferente:

```
"calabaza" → [("c" ,1),("a" ,1),("l" ,1),("a" ,1),("b" ,1),("a" ,1),("z" ,1),("a"
 ,1)]
"lima" → [("l" ,1),("i" ,1),("m" ,1),("a" ,1)]
```

(Posiblemente, te preguntarás por qué no hemos hecho una función más útil, como la siguiente:

"calabaza" → [("c" ,1),("a" ,4),("l" ,1),("b" ,1),("z" ,1)]

Pero déjame que conteste a esto más tarde.)

Versión 2 del reducer. En realidad, el reducer toma todos los valores de cada clave y los combina:

[("c" ,[1]),("a" ,[1,1,1,1,1]),("l" ,[1,1]),("b" ,[1]),("z" ,[1]),("i" ,[1]),("m" ,[1])] → [("c" ,1),("a" ,5),("l" ,2),("b" ,1),("z" ,1),("i" ,1),("m" ,1)]

Para que todo esto funcione, basta con definir el mapper, que actúa sobre una línea así: for car in línea: yield car, 1

Y el reducer, que actúa sobre una lista de números así:

lista → sum(lista)

Explicación del paréntesis anterior. Completando el paréntesis de antes, vemos que una función que suma las repeticiones de cada carácter, simplemente adelanta una parte del trabajo del reducer, pero no altera el resultado, y es más sencillo para nosotros dar la versión sencilla de una lista de unos.

Ejecución desde la línea de comandos. Para mostrar el comportamiento de estas funciones desde la línea de comandos, realizamos las llamadas desde las celdas siguientes.

In [4]:

```
# He aquí el archivo de datos

!type dos_palabras.txt
```

```
calabaza
lima
```

In [5]:

```
# Nuestro programa, contador de caracteres:

!type char_count.py
```

```
from mrjob.job import MRJob

class MRCharCount(MRJob):

    def mapper(self, _, line):
        for car in line:
            yield car, 1

    def reducer(self, key, values):
        yield key, sum(values)

if __name__ == '__main__':
    MRCharCount.run()
```

In [6]:

```
#Y ahora vemos el programa en acción sobre el archivo de datos:

! python char_count.py dos_palabras.txt -q
```

```
"a"     5
"b"     1
"c"     1
"i"     1
"l"     2
"m"     1
"z"     1
```

Nota. La opción `-q` significa *quiet*. Elimina mensajes superfluos del sistema operativo en la consola de comandos. Para ver su efecto, ejecuta la orden anterior suprimiendo esta opción.

Dos ejercicios básicos

Ejercicio 1 Hemos diseñado un contador de letras. ¿Sabrías diseñar tú un contador de palabras?

Para una línea dada, la función mapper debería generar, cada palabra acompañada con un 1.

Ejercicio 2 ¿Sabrías diseñar tú un programa que cuenta cuántas líneas hay en un archivo?

Para una línea dada, la función mapper debería generar un 1, con una etiqueta cualquiera.

In [7]:

```
#Y ahora vemos el programa en acción sobre el archivo de datos:

! type pi_cuarteto.txt
```

```
Soy y seré a todos definible
mi nombre tengo que daros
cociente diametral siempre inmedible
soy de los redondos aros

y seré también todos los aros cuadrados
y soy definible y cociente siempre de los aros cuadrados
```

In [8]:

```
#Y ahora vemos el contador de palabras en acción sobre el archivo de datos:

! python word_count.py pi_cuarteto.txt -q
```

```
"Soy"   1
"a"     1
"aros"  3
"cociente"      2
"cuadrados"     2
"daros" 1
"de"    2
"definible"     2
"diametral"     1
"inmedible"     1
"los"   3
"mi"    1
"nombre"        1
"que"   1
"redondos"      1
"ser\u00e9"     2
"siempre"       2
"soy"   2
"tambi\u00e9n"  1
"tengo" 1
"todos" 2
"y"     4
```

In [9]:

```
#Y ahora vemos el contador de líneas en acción sobre el archivo de datos:

! python lines_count.py pi_cuarteto.txt -q
```

```
"lines" 7
```

Un ejemplo más completo:

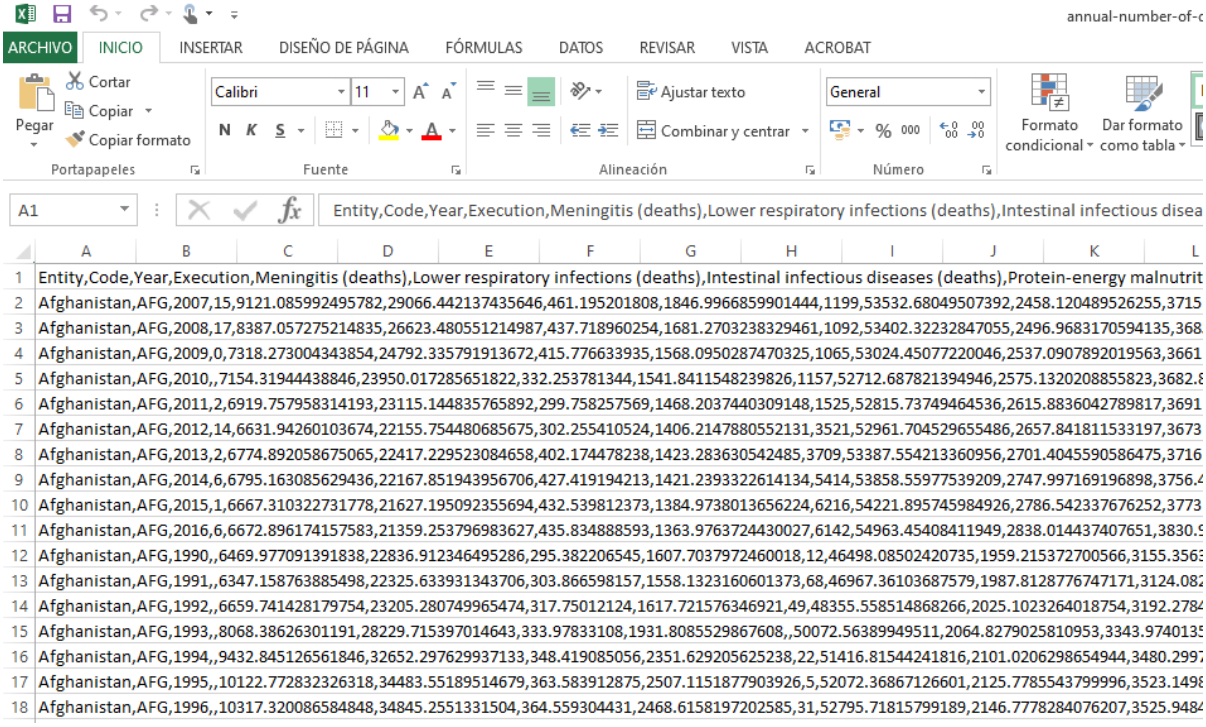
Tenemos un archivo de datos con información sobre las causas de muerte en cada país del mundo según la causa del deceso. He aquí unas cuantas líneas de dicho archivo:

In [10]:

```
! type "annual-number-of-deaths-by-cause.csv"
```

```
Entity,Code,Year,Execution,Meningitis (deaths),Lower respiratory infection
s (deaths),Intestinal infectious diseases (deaths),Protein-energy malnutri
tion (deaths),Terrorism (deaths),Cardiovascular diseases (deaths),Dementia
(deaths),Kidney disease (deaths),Respiratory diseases (deaths),Liver disea
ses (deaths),Digestive diseases (deaths),Hepatitis (deaths),Cancers (death
s),Parkinson disease (deaths),Fire (deaths),Malaria (deaths),Drowning (dea
ths),Homicide (deaths),HIV/AIDS (deaths),Drug use disorders (deaths),Tuber
culosis (deaths),Road injuries (deaths),Maternal disorders (deaths),Neonat
al disorders (deaths),Alcohol use disorders (deaths),Natural disasters (de
aths),Diarrheal diseases (deaths),Heat (hot and cold exposure) (deaths),Nu
tritional deficiencies (deaths),Suicide (deaths),Conflict (deaths),Diabete
s (deaths),Poisonings (deaths)
Afghanistan,AFG,2007,15,9121.085992495782,29066.442137435646,461.19520180
8,1846.9966859901444,1199,53532.68049507392,2458.120489526255,3715.2775921
059497,6896.159472690725,1930.0471197273198,4922.241281315872,1050.2894758
720836,14253.281223073709,400.81106488371825,498.3921466537136,1118.396508
3681703,2494.128056651257,3029.127923045497,144.5037469282267,209.68336203
058993,4859.483790971534,8099.404949919545,4810.833683712643,28477.0811066
7426.96.02759236080246.296.00002613563987.10366.124291797547.160.930121431
```

Mejor es que lo veas por ti mismo, abriendo el archivo, aunque te proporciono una imagen a continuación de un fragmento:



Los campos están separados por comas. La primera línea es la cabecera. Queremos saber cuántas muertes ha habido en Estados Unidos (la abreviatura es USA, en el segundo campo de la tabla) debidas a ejecuciones.

In [11]:

```
! python deadh_cause.py "annual-number-of-deaths-by-cause.csv" -q
```

```
Belgium,BEL,2002,,67.7372038836438,5326.496481573126,0.483599636578,112.22
012158130893,0,35922.723743033814,7923.168599702642,1462.0555603432356,704
9.8500818641305,1807.9392712348674,4748.459102228866,47.114076165298904,28
503.28267660546,938.1820809142874,184.72808704368794,0,107.0877108052443,2
18.58667782496758,95.71422230532018,195.41023172700164,132.17686540592928,
1645.9846752297087,8.93705631505045,236.85350994557314,368.4985984883272,
2.999999967308866,284.68561555083573,70.67269475791721,168.83536031302077,
2311.0691580828056,0,1621.9213982756607,56.38020908092849
Belgium,BEL,2003,,65.35439634098012,5425.466552555146,0.475825243326,122.2
3659828714455,0,35479.41415558417,8041.6801721555685,1508.7091654429541,70
30.380388526124,1805.4752207119132,4750.987355434725,41.606284358912816,28
176.28696607491,968.3770385111236,184.25555960487856,0,104.18196660394969,
202.36013729630338,93.72503421179435,196.00726006575934,128.0242503875303
4,1536.3489453718175,10.910939820572384,234.25631363667304,370.69971688356
867,0,340.1802717637355,111.74601167401292,183.35018164743312,2269.6656391
036895,0,1612.2441108802493,52.77524360683966
Belgium,BEL,2004,,59.86543158850268,5240.622994951065,0.463645495596,124.9
7249921297657,,33921.19383688878,8009.8631677692865,1496.4153409215542,675
0.913744658176.1792.6453368978734.4699.168436413891.36.50108246662096.2803
```

In [12]:

```
#Veamos el código:

! type deadh_cause.py
```

```
from mrjob.job import MRJob

def to_int(cadena):
    try:
        return int(cadena)
    except:
        return 0

class MRCauseOfDeadth(MRJob):

    def mapper(self, _, line):
        campos = line.split(",")
        passcodigo = campos[1]
        num_ejecs = to_int(campos[3])
        if passcodigo == "USA":
            yield "ejecs", num_ejecs

    def reducer(self, key, values):
        yield key, sum(values)
```

Parámetros en la línea de comandos

*** Faltan explicaciones ***

In [13]:

```
! python deadh_cause_country.py --country=USA "annual-number-of-deaths-by-cause.csv" -q
```

```
"USA"	385
```

In [14]:

```
! python deadh_cause_country.py --country=ESP "annual-number-of-deaths-by-cause.csv" -q
```

```
"ESP"	0
```

In [15]:

```
#Veamos el código:

! type deadh_cause_country.py
```

```
from mrjob.job import MRJob

def to_int(cadena):
    try:
        return int(cadena)
    except:
        return 0

class MRCauseOfDeadth(MRJob):

    def configure_args(self):
        super(MRCauseOfDeadth, self).configure_args()
        self.add_passthru_arg(
            '--country', default='Spain',
            help="Indica el cÃ³digo del paÃs.")

    def mapper(self, _, line):
        campos = line.split(",")
        passcodigo = campos[1]
```

Propiedades matemáticas supuestas

Podemos calcular la suma de una lista de varias maneras, alterando el orden y agrupamiento de los sumandos:

```
[1, 2, 3, 4, 5] -> [1, 2, 3], [4, 5] -> [6, 9] -> 15
[1, 2, 3, 4, 5] -> [1, 2], [3, 4, 5] -> [3, 12] -> 15
...
```

Siempre obtenemos el mismo resultado, porque la suma es conmutativa y asociativa.

Hagamos lo mismo con la media:

```
[1, 2, 3, 4, 5] -> [1, 2, 3], [4, 5] -> [2, 4.5] -> 3.25
[1, 2, 3, 4, 5] -> [1, 2], [3, 4, 5] -> [1.5, 4] -> 2.75
```

Se obtienen resultados distintos agrupando en distintos órdenes porque la media no es asociativa.

El modelo *map-reduce* está pensado para procesar grandes volúmenes de datos en un orden arbitrario y agrupando los resultados de forma arbitraria, según vayan siendo generados los paquetes de trabajo por distintas máquinas o procesadores. Esto puede no percibirse con datos pequeños o cuando el procesamiento es en un solo ordenador, pero conviene saber que la función `reduce` tiene que estar basada en una operación binaria asociativa y conmutativa. Y la media no lo es.

Pongamos ahora que deseamos calcular la media de una gran cangtidad de números. Ponemos una solución inicial errónea, aunque se trata de un error muy frecuente y que pasa desapercibido con cantidades pequeñas de datos, y luego comentamos una solución correcta.

Nota. Esta solución está explicada y con ejemplos en la carpeta aparte siguiente: `ej3 - media - postprocesamiento`

Bibliografía

- En la siguiente url se puede encontrar

  `https://mrjob.readthedocs.io/en/latest/guides/writing-mrjobs.html`

- En particular, los detalles técnicos sobre el paso de opciones (como `--country=ESP`) puede completarse con el apartado *Defining command line options*.