



*Cloud Computing*

---

# Ejercicios HADOOP

Implementación y análisis de  
funciones básicas sobre  
conjuntos de datos BigData

---

# ¿Qué vamos a ver?

---

- ❖ Ejemplos iniciales con Hadoop
- ❖ Conjunto de datos **ECBDL14**:  
10 columnas, 31992921 registros, 1.2 GB.
- ❖ Cálculo de MAX,MIN,AVG para el conjunto de datos **ECBDL14**.
- ❖ ¿Es balanceado o no balanceado el conjunto de datos **ECBDL14**?
- ❖ Cálculo de coeficiente de correlación.

---

# WordCount en Hadoop

---

- ❖ En primer lugar debemos almacenar el fichero de entrada en HDFS:
  - ❖ `hadoop fs -put /tmp/joyce.txt ./`
  - ❖ También podemos usar el que hay en HDFS: `/tmp/BDCC/wordcount/dataset/joyce/joyce.txt`
- ❖ A continuación, aprovechamos los ejemplos que hay por defecto en Hadoop:
  - ❖ `hadoop jar /usr/lib/hadoop-0.20-mapreduce/hadoop-examples.jar wordcount <IN> <OUT>`
    - ❖ `<IN>` es la ruta del fichero de entrada (joyce.txt)
    - ❖ `<OUT>` es la ruta del directorio de salida (joyceWC) [no debe estar creado]



# Listado de mensajes de MapReduce

- ❖ ¿Qué significan todos estos mensajes? Hadoop ha efectuado mucho trabajo y está intentando comunicárselo a usted, incluyendo lo siguiente.
- ❖ Verificó si el archivo de entrada existe.
- ❖ Verificó si el directorio de salida existe, y si existe, aborta el trabajo. No hay nada peor que horas de cómputo duplicado debido a un simple error de digitación.
- ❖ Distribuyó el archivo Java jar hacia todos los nodos responsables por realizar el trabajo. En este caso, este es solo un nodo.
- ❖ Ejecutó la fase mapper del trabajo. Típicamente esto analiza el archivo de entrada y emite un par de valor clave. Observe que la clave y valor pueden ser objetos.
- ❖ Ejecutó la fase de ordenar, que ordena la salida del mapper con base en la clave.
- ❖ Ejecutó la fase de reducción, típicamente esto resume la transmisión de clave-valor y escribe salida hacia HDFS.
- ❖ Creó muchas métricas en el transcurso.

```
$ hadoop jar /usr/lib/hadoop/hadoop-examples.jar wordcount HF.txt HF.out
12/08/08 19:23:46 INFO input.FileInputFormat: Total input paths to process : 1
12/08/08 19:23:47 WARN snappy.LoadSnappy: Snappy native library is available
12/08/08 19:23:47 INFO util.NativeCodeLoader: Loaded the native-hadoop library
12/08/08 19:23:47 INFO snappy.LoadSnappy: Snappy native library loaded
12/08/08 19:23:47 INFO mapred.JobClient: Running job: job_201208081900_0002
12/08/08 19:23:48 INFO mapred.JobClient: map 0% reduce 0%
12/08/08 19:23:54 INFO mapred.JobClient: map 100% reduce 0%
12/08/08 19:24:01 INFO mapred.JobClient: map 100% reduce 33%
12/08/08 19:24:03 INFO mapred.JobClient: map 100% reduce 100%
12/08/08 19:24:04 INFO mapred.JobClient: Job complete: job_201208081900_0002
12/08/08 19:24:04 INFO mapred.JobClient: Counters: 26
12/08/08 19:24:04 INFO mapred.JobClient:   Job Counters
12/08/08 19:24:04 INFO mapred.JobClient:     Launched reduce tasks=1
12/08/08 19:24:04 INFO mapred.JobClient:     SLOTS_MILLIS_MAPS=5959
12/08/08 19:24:04 INFO mapred.JobClient:     Total time spent by all reduces...
12/08/08 19:24:04 INFO mapred.JobClient:     Total time spent by all maps waiting...
12/08/08 19:24:04 INFO mapred.JobClient:     Launched map tasks=1
12/08/08 19:24:04 INFO mapred.JobClient:     Data-local map tasks=1
12/08/08 19:24:04 INFO mapred.JobClient:     SLOTS_MILLIS_REDUCES=9433
12/08/08 19:24:04 INFO mapred.JobClient:   FileSystemCounters
12/08/08 19:24:04 INFO mapred.JobClient:     FILE_BYTES_READ=192298
12/08/08 19:24:04 INFO mapred.JobClient:     HDFS_BYTES_READ=597700
12/08/08 19:24:04 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=498740
12/08/08 19:24:04 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=130218
12/08/08 19:24:04 INFO mapred.JobClient:   Map-Reduce Framework
12/08/08 19:24:04 INFO mapred.JobClient:     Map input records=11733
12/08/08 19:24:04 INFO mapred.JobClient:     Reduce shuffle bytes=192298
12/08/08 19:24:04 INFO mapred.JobClient:     Spilled Records=27676
12/08/08 19:24:04 INFO mapred.JobClient:     Map output bytes=1033012
12/08/08 19:24:04 INFO mapred.JobClient:     CPU time spent (ms)=2430
12/08/08 19:24:04 INFO mapred.JobClient:     Total committed heap usage (bytes)=183701504
12/08/08 19:24:04 INFO mapred.JobClient:     Combine input records=113365
12/08/08 19:24:04 INFO mapred.JobClient:     SPLIT_RAW_BYTES=113
12/08/08 19:24:04 INFO mapred.JobClient:     Reduce input records=13838
12/08/08 19:24:04 INFO mapred.JobClient:     Reduce input groups=13838
12/08/08 19:24:04 INFO mapred.JobClient:     Combine output records=13838
12/08/08 19:24:04 INFO mapred.JobClient:     Physical memory (bytes) snapshot=256479232
12/08/08 19:24:04 INFO mapred.JobClient:     Reduce output records=13838
12/08/08 19:24:04 INFO mapred.JobClient:     Virtual memory (bytes) snapshot=1027047424
12/08/08 19:24:04 INFO mapred.JobClient:     Map output records=113365
```

# Salida de MapReduce

```
# way too much typing, create aliases for hadoop commands
$ alias hput="hadoop fs -put"
$ alias hcat="hadoop fs -cat"
$ alias hls="hadoop fs -ls"
$ alias hrmr="hadoop fs -rmr"

# first list the output directory
$ hls /user/cloudera/HF.out
Found 3 items
-rw-r--r-- 1 cloudera supergroup 0 2012-08-08 19:38 /user/cloudera/HF.out/_SUCCESS
drwxr-xr-x - cloudera supergroup 0 2012-08-08 19:38 /user/cloudera/HF.out/_logs
-rw-r--r-- 1 cl... sup... 138218 2012-08-08 19:38 /user/cloudera/HF.out/part-r-00000

# now cat the file and pipe it to the less command
$ hcat /user/cloudera/HF.out/part-r-00000 | less

# here are a few lines from the file, the word elephants only got used twice
elder, 1
eldest 1
elect 1
elected 1
electronic 27
electronically 1
electronically, 1
elegant 1
elegant!--'deed 1
elegant, 1
elephants 2
```



---

# Código WordCount: Driver

---

```
public static void main(String[] args) throws Exception {

    JobConf conf = new JobConf(WordCount.class);
    conf.setJobName("wordcount");

    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(IntWritable.class);

    conf.setMapperClass(Map.class);
    conf.setCombinerClass(Reduce.class);
    conf.setReducerClass(Reduce.class);

    conf.setInputFormat(TextInputFormat.class);
    conf.setOutputFormat(TextOutputFormat.class);

    FileInputFormat.setInputPaths(conf, new Path(args[0]));
    FileOutputFormat.setOutputPath(conf, new Path(args[1]));

    JobClient.runJob(conf);
}
```

---

# Código WordCount: Mapper

---

```
public static class Map extends MapReduceBase
    implements Mapper<LongWritable, Text, Text, IntWritable> {

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value, OutputCollector<Text,
IntWritable> output, Reporter reporter) throws IOException {

        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
            word.set(tokenizer.nextToken());
            output.collect(word, one);
        }
    }
}
```

---

# Código WordCount: Reducer

---

```
public static class Reduce extends MapReduceBase
    implements Reducer<Text, IntWritable, Text, IntWritable> {

    public void reduce(Text key, Iterator<IntWritable> values,
        OutputCollector<Text, IntWritable> output, Reporter reporter)
        throws IOException {

        int sum = 0;
        while (values.hasNext()) {
            sum += values.next().get();
        }
        output.collect(key, new IntWritable(sum));
    }
}
```



# Visualizar estado via Web (8088)



Logged in as: dr.who

## Nodes of the cluster

### Cluster

[About](#)  
[Nodes](#)  
[Applications](#)  
NEW  
NEW\_SAVING  
SUBMITTED  
ACCEPTED  
RUNNING  
REMOVING  
FINISHING  
FINISHED  
FAILED  
KILLED  
[Scheduler](#)

### Tools

### Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
0	0	0	0	0	0 B	8 GB	0 B	1	0	0	0	0

Show 20 entries

Search:

Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Mem Used	Mem Avail
/default-rack	RUNNING	localhost:38490	localhost:8042	24-Feb-2014 00:17:38		0	0 B	8 GB

Showing 1 to 1 of 1 entries

First [Previous](#) 1 [Next](#) Last

# Visualizar estado via Web (8088)

## Application Overview

**User:** sergio  
**Name:** PigLatin:DefaultJobName  
**Application Type:** MAPREDUCE  
**Application Tags:**  
**State:** FINISHED  
**FinalStatus:** SUCCEEDED  
**Started:** Thu Mar 12 12:01:02 +0100 2015  
**Elapsed:** 15sec  
**Tracking URL:** [History](#)  
**Diagnostics:** No of maps and reduces are 0 job\_1426156216705\_0005

## Application Metrics

**Total Resource Preempted:** <memory:0, vCores:0>  
**Total Number of Non-AM Containers Preempted:** 0  
**Total Number of AM Containers Preempted:** 0  
**Resource Preempted from Current Attempt:** <memory:0, vCores:0>  
**Number of Non-AM Containers Preempted from Current Attempt:** 0  
**Aggregate Resource Allocation:** 24598 MB-seconds, 12 vcore-seconds

## ApplicationMaster

Attempt Number	Start Time	Node	Logs
	Thu Mar 12 12:01:12 +0100 2015	<a href="#">nodeh05.local:8042</a>	<a href="#">logs</a>



# Visualizar estado via Web (8088)

Show 20 entries <span>Search:</span>											
Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Mem Used	Mem Avail	VCores Used	VCores Avail	Version
/default-rack	RUNNING	nodeh11.local:42693	<a href="#">nodeh11.local:8042</a>	Thu Mar 12 13:25:31 +0100 2015		0	0 B	54.69 GB	0	24	2.5.0-cdh5.3.1
/default-rack	RUNNING	nodeh15.local:51071	<a href="#">nodeh15.local:8042</a>	Thu Mar 12 13:25:31 +0100 2015		0	0 B	54.69 GB	0	24	2.5.0-cdh5.3.1
/default-rack	RUNNING	nodeh06.local:51414	<a href="#">nodeh06.local:8042</a>	Thu Mar 12 13:25:31 +0100 2015		0	0 B	54.69 GB	0	24	2.5.0-cdh5.3.1
/default-rack	RUNNING	nodeh12.local:51953	<a href="#">nodeh12.local:8042</a>	Thu Mar 12 13:25:31 +0100 2015		0	0 B	54.69 GB	0	24	2.5.0-cdh5.3.1
/default-rack	RUNNING	nodeh05.local:34173	<a href="#">nodeh05.local:8042</a>	Thu Mar 12 13:25:31 +0100 2015		0	0 B	54.69 GB	0	24	2.5.0-cdh5.3.1
/default-rack	RUNNING	nodeh02.local:37593	<a href="#">nodeh02.local:8042</a>	Thu Mar 12 13:25:31 +0100 2015		0	0 B	54.69 GB	0	24	2.5.0-cdh5.3.1
/default-rack	RUNNING	nodeh14.local:45120	<a href="#">nodeh14.local:8042</a>	Thu Mar 12 13:25:31 +0100 2015		0	0 B	54.69 GB	0	24	2.5.0-cdh5.3.1
/default-rack	RUNNING	nodeh07.local:54001	<a href="#">nodeh07.local:8042</a>	Thu Mar 12 13:25:31 +0100 2015		0	0 B	54.69 GB	0	24	2.5.0-cdh5.3.1
/default-rack	RUNNING	nodeh13.local:37267	<a href="#">nodeh13.local:8042</a>	Thu Mar 12 13:25:31 +0100 2015		0	0 B	54.69 GB	0	24	2.5.0-cdh5.3.1
/default-rack	RUNNING	nodeh08.local:43208	<a href="#">nodeh08.local:8042</a>	Thu Mar 12 13:25:31 +0100 2015		0	0 B	54.69 GB	0	24	2.5.0-cdh5.3.1
/default-rack	RUNNING	nodeh03.local:47698	<a href="#">nodeh03.local:8042</a>	Thu Mar 12 13:25:31 +0100 2015		0	0 B	54.69 GB	0	24	2.5.0-cdh5.3.1
/default-rack	RUNNING	nodeh10.local:58548	<a href="#">nodeh10.local:8042</a>	Thu Mar 12 13:25:31 +0100 2015		0	0 B	54.69 GB	0	24	2.5.0-cdh5.3.1
/default-rack	RUNNING	nodeh04.local:35646	<a href="#">nodeh04.local:8042</a>	Thu Mar 12 13:25:31 +0100 2015		0	0 B	54.69 GB	0	24	2.5.0-cdh5.3.1
/default-rack	RUNNING	nodeh09.local:45014	<a href="#">nodeh09.local:8042</a>	Thu Mar 12 13:25:31 +0100 2015		0	0 B	54.69 GB	0	24	2.5.0-cdh5.3.1
Showing 1 to 14 of 14 entries										First Previous 1 Next Last	

# Cálculo de MAX, MIN y AVG con BigData

- ❖ Vamos a trabajar con un conjunto de datos Big Data real.
- ❖ El conjunto ECBDL corresponde a un problema biológico.
- ❖ Presenta la siguiente estructura: 10 columnas, 31992921 registros, 1.2 GB.
- ❖ Trabajaremos solo con un 10% del total.

```
0.217,0.045,-5,0,-4,0,-4,9,-7,3,0
0.221,0.076,-2,-2,1,-2,1,-4,-3,1,0
0.152,0.055,-4,-8,-3,0,-6,-2,1,-4,0
0.247,0.045,-2,2,-1,1,-3,-1,-1,0,0
0.227,0.077,-4,-1,-4,-4,-1,0,-2,0,0
0.289,0.100,-1,0,-1,-2,-6,-8,1,-8,0
0.198,0.043,-2,-3,-6,-1,-4,1,-4,-2,0
0.191,0.106,2,-2,-3,-3,2,-5,3,-5,0
0.395,0.065,-4,1,2,-1,-1,-4,3,-1,0
0.225,0.070,-1,-3,-3,0,-3,-5,4,4,0
```

Separador de columnas

Variable de clase



---

# Cálculo de MIN, MAX y AVG con Big Data

---

- ❖ Descargamos el código Java a nuestra carpeta:

```
mkdir stat
```

```
cp /tmp/Min* ./stat/
```

- ❖ Comprobamos que la ruta de los datos de entrada:

```
hdfs dfs -ls /tmp/BDCC/datasets/ECBDL14/
```

- ❖ Debe aparecer el fichero ECBDL14\_10tst.data

---

# Cálculo del MÍNIMO

---

- ❖ Creamos el directorio de clases en local

```
cd stat  
mkdir java_classes
```

- ❖ Compilamos y ejecutamos:

```
javac -cp /usr/lib/hadoop/*:/usr/lib/hadoop-mapreduce/* -d  
java_classes Min*
```

```
jar -cvf stat.jar -C java_classes / .
```

```
hadoop jar stat.jar oldapi.Min /tmp/BDCC/datasets/ECBDL14/  
ECBDL14_10tst.data ./stat/output/
```



---

# Resultados MÍNIMO

---

- ❖ Comprobamos el resultado:

```
hdfs dfs -cat stat/output/*
```


- ❖ Debe aparecer el valor del **Mínimo de la columna seleccionada.**

```
1 -11.0
```

# Veamos el código: MAPPER

```
public class MinMapper extends MapReduceBase implements  
Mapper<LongWritable, Text, Text, DoubleWritable> {  
    private static final int MISSING = 9999;  
    public static int col=5;
```

```
    public void map(LongWritable key, Text value,  
OutputCollector<Text, DoubleWritable> output, Reporter reporter)  
throws IOException {  
        String line = value.toString();  
        String[] parts = line.split(",");  
        output.collect(new Text("1"), new  
DoubleWritable(Double.parseDouble(parts[col])));  
    }  
}
```



¿Por qué usamos en KEY un 1?



---

# Veamos el código: REDUCER

---

```
public class MinReducer extends MapReduceBase implements Reducer<Text,  
DoubleWritable, Text, DoubleWritable> {
```

```
    public void reduce(Text key, Iterator<DoubleWritable> values,  
OutputCollector<Text, DoubleWritable> output, Reporter reporter)  
throws IOException {
```

```
        Double minValue = Double.MAX_VALUE;
```

```
        while (values.hasNext()) {
```

```
            minValue = Math.min(minValue, values.next().get());
```

```
        }
```

```
        output.collect(key, new DoubleWritable(minValue));
```

```
    }
```

```
}
```

---

# Permitir reescribir directorio

---

```
Configuration conf = new Configuration();
String outputPath = HDFSLocation+" / "+Path+"_TMP";
FileOutputFormat.setOutputPath(job, new Path(outputPath));
...
fs = FileSystem.get(new URI(HDFSLocation,conf);
Path textPath = new Path(HDFSLocation+" / "+Path);
BufferedWriter bwText = new BufferedWriter(new OutputStreamWriter(fs.create(textPath,true)));
FileStatus[] status = fs.listStatus(new Path(outputPath));
Writer writer = SequenceFile.createWriter(Mediator.getConfiguration(),
Writer.file(new Path(HDFSLocation()+Path()))),
Writer.keyClass(ByteWritable.class), Writer.valueClass(FloatWritable.class)); // Tipos de datos reconocidos por Hadoop
for (FileStatus fileStatus:status){
    reader = new Reader(conf, Reader.file(fileStatus.getPath()));
    while (reader.next(X,X)) { // utilizar los mismos tipos de datos
        writer.append(X, X); } // utilizar los mismos tipos de datos
    reader.close();
    bwText.close(); writer.close();
}
fs.delete(new Path(outputPath),true);
```



---

# ¿Cómo se calcularía el Máximo?

---

- ❖ Utilizamos el mismo ejemplo de bigdata, cambiando los datos relativos y modificamos el algoritmo para que calcule el máximo.
- ❖ ¿Que habría que cambiar?
- ❖ Reproducimos los pasos anteriores. ¿Qué cambiaría en el REDUCE?

---

# ¿Cómo se calcularía la Media?

---

- ❖ El Mapper es idéntico a los casos anteriores
- ❖ ¿Qué habría que cambiar en REDUCER?

Pista:

Suma de valores y división entre el total de registros



---

# Cálculo de los estadísticos descriptivos

---

- ❖ Nuestro objetivo ahora es actualizar el código para realizar las siguientes tareas:
  1. Parametrizar la columna sobre la que se quiere calcular el estadístico
  2. Combinar el cálculo de todos los estadísticos en una única función
  3. Calcular los estadísticos sobre todas las columnas
  4. Repite el proceso sobre un conjunto de mayor volumen (Ej: /user/isaac/datasets/higgs...) ¿Hay grandes diferencias de tiempo?
  5. Acelera el proceso de cómputo descargando al Reducer de parte de la tarea.

---

# Procesos Setup y Cleanup

---

- ❖ Se pueden incluir en las tareas Map y Reduce
  - ❖ setup -> map -> cleanup
  - ❖ setup -> reduce -> cleanup
- ❖ SETUP()
  - ❖ Se llama una única vez al comienzo de la tarea
  - ❖ Se suelen leer los parámetros del objeto “Configuration” para adaptar la lógica de procesamiento.
- ❖ CLEANUP():
  - ❖ Se llama una vez al final de la tarea
  - ❖ Normalmente se usa para eliminar recursos. También para la agregación de valores que se han realizado durante el Map



---

# Ejemplo de uso (Driver)

---

```
package com.setup.mcis;

import org.apache.hadoop.conf.Configuration; import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path; import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text; import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.Tool; import org.apache.hadoop.util.ToolRunner;

public class Stopwords extends Configured implements Tool {

    public static void main(String args[]) throws Exception{
        int res = ToolRunner.run(new Configuration(), new Stopwords(), args);
        System.exit(res);
    }

    public int run(String args[])throws Exception{

        Configuration conf = new Configuration();
        conf.set("words", "the, is");
        //conf.set("stopword", "the, an, a, is");
        Job job = new Job(conf, "stopword");
        job.setJarByClass(Stopwords.class);
        job.setMapperClass(Mapstop.class);
        job.setReducerClass(Reducestop.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        return job.waitForCompletion(true) ? 0:1 ;

    }

}
```

---

# Ejemplo de uso (Mapper)

---

```
package com.setup.mcis;

import java.io.IOException; import java.util.HashSet; import java.util.Set; import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration; import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable; import org.apache.hadoop.io.Text; import org.apache.hadoop.mapreduce.Mapper;

public class Mapstop extends Mapper<LongWritable, Text, Text, IntWritable> {
    private final IntWritable one = new IntWritable(1);
    private Text word = new Text();
    private Set<String> stopWords;
    protected void setup(Mapper<LongWritable, Text, Text, IntWritable>.Context context){
        Configuration conf = context.getConfiguration();
        String sw = conf.get("words");
        stopWords = new HashSet<String>();
        System.out.println(sw);
        String[] rec = sw.split(",");
        for(String word : rec) {
            stopWords.add(word);
        }
    }
    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
            String token = tokenizer.nextToken();
            if(stopWords.contains(token)) {
                continue;
            }
            word.set(token);
            context.write(word, one);
        }
    }
}
```



---

# Ejemplo de uso (Reducer)

---

```
package com.setup.mcis;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class Reducestop extends Reducer<Text, IntWritable, Text,
IntWritable>{
    public void reduce(Text key, Iterable<IntWritable> values, Context
context)
        throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        context.write(key, new IntWritable(sum));
    }
}
```

---

# ¿Es un conjunto de datos balanceado?

---

- ❖ Para comprobar si el conjunto de datos es balanceado o no-balanceado, necesitamos saber cuantos registros hay de cada variable de clase.
- ❖ La variable de clase es la última columna de los datos y puede tener los valores de 0 y 1.

Pista: La llave (key del mapper) debe ser el valor de la columna de clase (11).



# Calculo del coeficiente de correlación

- ❖ ¿Cómo se calcula el coeficiente de correlación entre dos variables?

$$r = \frac{S_{XY}}{S_X S_Y}$$

Covarianza

Desviación típica

# Calculo del coeficiente de correlación

<b>2</b>	<b>1</b>
<b>3</b>	<b>3</b>
<b>4</b>	<b>2</b>
<b>4</b>	<b>4</b>
<b>5</b>	<b>4</b>
<b>6</b>	<b>4</b>
<b>6</b>	<b>6</b>
<b>7</b>	<b>4</b>
<b>7</b>	<b>6</b>
<b>8</b>	<b>7</b>
<b>10</b>	<b>9</b>
<b>10</b>	<b>10</b>

<b><math>x_i</math></b>	<b><math>y_i</math></b>	<b><math>x_i \cdot y_i</math></b>	<b><math>x_i^2</math></b>	<b><math>y_i^2</math></b>
<b>2</b>	<b>1</b>	<b>2</b>	<b>4</b>	<b>1</b>
<b>3</b>	<b>3</b>	<b>9</b>	<b>9</b>	<b>9</b>
<b>4</b>	<b>2</b>	<b>8</b>	<b>16</b>	<b>4</b>
<b>4</b>	<b>4</b>	<b>16</b>	<b>16</b>	<b>16</b>
<b>5</b>	<b>4</b>	<b>20</b>	<b>25</b>	<b>16</b>
<b>6</b>	<b>4</b>	<b>24</b>	<b>36</b>	<b>16</b>
<b>6</b>	<b>6</b>	<b>36</b>	<b>36</b>	<b>36</b>
<b>7</b>	<b>4</b>	<b>28</b>	<b>49</b>	<b>16</b>
<b>7</b>	<b>6</b>	<b>42</b>	<b>49</b>	<b>36</b>
<b>8</b>	<b>7</b>	<b>56</b>	<b>64</b>	<b>49</b>
<b>10</b>	<b>9</b>	<b>90</b>	<b>100</b>	<b>81</b>
<b>10</b>	<b>10</b>	<b>100</b>	<b>100</b>	<b>100</b>
<b>72</b>	<b>60</b>	<b>431</b>	<b>504</b>	<b>380</b>



# Calculo del coeficiente de correlación

**1º** Hallamos las **medias aritméticas**.

$$\bar{x} = \frac{72}{12} = 6$$

$$\bar{y} = \frac{60}{12} = 5$$

**2º** Calculamos la **covarianza**.

$$\sigma_{xy} = \frac{431}{12} - 6 \cdot 5 = 5.92$$

**3º** Calculamos las **desviaciones típicas**.

$$\sigma_x = \sqrt{\frac{504}{12} - 6^2} = 2.45$$

$$\sigma_y = \sqrt{\frac{380}{12} - 5^2} = 2.58$$

**4º** Aplicamos la fórmula del **coeficiente de correlación lineal**.

$$r = \frac{5.92}{2.45 \cdot 2.58} = 0.94$$

# Calculo del coeficiente de correlación

**Posible solución:**

MAPPER —> Realiza cálculos por línea

$x_i$	$y_i$	$x_i \cdot y_i$	$x_i^2$	$y_i^2$
2	1	2	4	1

REDUCER —> Opera los pasos 1, 2, 3, 4 de la diapositiva anterior.



---

# Referencias

---

<http://sci2s.ugr.es/sites/default/files/files/Teaching/OtherPostGraduateCourses/CienciaDatosBigData/Bloque%20III.zip>

<https://github.com/geftimov/MapReduce/blob/master/readme/MedianStdDev.md>

<https://github.com/geftimov/MapReduce/blob/master/readme/MedianAndStandardDeviationCommentLengthByHour.md>

<https://github.com/geftimov/MapReduce/blob/master/readme/MedianAndStandardDeviationCommentLengthByHour.md>

<https://svn.apache.org/repos/asf/hadoop/common/trunk/hadoop-mapreduce-project/hadoop-mapreduce-examples/src/main/java/org/apache/hadoop/examples/WordStandardDeviation.java>

<http://www.hadooptpoint.com/hadoop-setup-method-cleanup-method-example-in-mapreduce/>

<https://www.ibm.com/developerworks/ssa/data/library/techarticle/dm-1209hadoopbigdata/>