



UTIU
UNIVERSITÀ TELEMATICA
INTERNAZIONALE **UNINETTUNO**

Dipartimento di Ingegneria
Corso di Laurea Magistrale in Ingegneria Informatica
Indirizzo: Big Data
A.A. 2022/2023

RELAZIONE SULL'ANALISI DELLA CONTABILITÀ AZIENDALE

Nome dello studente:	<i>Giorgia Nesci</i>
Matricola	<i>685HHHINGINFOR</i>
Data Appello	<i>Dicembre 2022</i>
Esame	<i>Introduzione ai Big Data</i>

Indice

<u>Introduzione</u>	Pag 1
<u>Svolgimento</u>	Pag 3
<u>Conclusioni</u>	Pag 27
<u>Sitografia</u>	Pag 28
<u>Appendice</u>	Pag 29

Elenco delle figure

Figura 1 – Aggiunta intestazioni colonne ed analisi tipologie documenti	3
Figura 2 – Output funzione table(ordini\$Tipo)	3
Figura 3 – Creazione DataFrame “ricavi”	3
Figura 4 – Frammento del DataFrame “ricavi”	4
Figura 5 – Calcolo media di ogni mese di ogni anno	4
Figura 6 – Frammento del DataFrame “media_mesi”	4
Figura 7 – Grafico della media mensile di vendita	5
Figura 8 – Frammento della struttura originaria file Ordini.csv fornita in input	6
Figura 9 – Frammento delle operazioni di mapping e shuffling job1	7
Figura 10 – Frammento dell’operazione di merging job1	7
Figura 11 – implementazione reduce per calcolo media di vendita per ogni mese di ogni anno	8
Figura 12 – Frammento dell’output del paradigma MapReduce a seguito dell’operazione di reducing job1	8
Figura 13 – Confronto del frammento dei risultati job1 dell’implementazione in R e in MapReduce	8
Figura 14 – Calcolo varianza di ogni mese di ogni anno	9
Figura 15 – Frammento del DataFrame “varianza_mesi”	9
Figura 16 – Grafico della varianza mensile di vendita	10
Figura 17 – Implementazione reduce per il calcolo della media di ogni mese di ogni anno	11
Figura 18 – Implementazione map per il calcolo dei risultati parziali della devianza	11
Figura 19 – Implementazione reduce per il calcolo della devianza	12
Figura 20 – Implementazione ultimo passaggio per il calcolo della varianza	12
Figura 21 – Frammento dell’output del paradigma MapReduce a seguito dell’operazione di reducing job2	12
Figura 22 – Confronto del frammento dei risultati job2 dell’implementazione in R e in MapReduce	12
Figura 23 – Calcolo fatturato totale di ogni mese di ogni anno	13
Figura 24 – Frammento dei DataFrame “somma_mesi”	13
Figura 25 – Grafico dell’andamento del fatturato da Gen2016 ad Ago2020	14
Figura 26 – Calcolo del ricavo mensile massimo di ogni anno	14
Figura 27 – DataFrame “mesi_max_importonly”	15
Figura 28 – Creazione DataFrame “mesi_max”	15
Figura 29 – DataFrame “mesi_max”	16
Figura 29 – Grafico del fatturato mensile massimo per ogni anno	16
Figura 30 – Implementazione reduce per il calcolo del ricavo totale di ogni mese job3	17
Figura 31 – Frammento dell’output del paradigma MapReduce a seguito dell’operazione di reducing per il risultato intermedio job3	17
Figura 35 – Frammento delle operazioni di mapping e shuffling job3	19
Figura 36 – Frammento dell’operazione di merging job3	19
Figura 37 – Implementazione reduce per selezione del mese di ogni anno con ricavo maggiore	20
Figura 38 – Output del paradigma MapReduce a seguito dell’operazione di reducing job3	20
Figura 39 – Confronto dei risultati job3 dell’implementazione in R e in MapReduce	20
Figura 40 – Calcolo del ricavo mensile minimo di ogni anno	21
Figura 41 – DataFrame “mesi_min_importonly”	21
Figura 42 – Creazione DataFrame “mesi_min”	22
Figura 43 – DataFrame “mesi_min”	22
Figura 44 – Grafico del fatturato mensile minimo per ogni anno	23
Figura 45 – Implementazione reduce per il calcolo del ricavo totale di ogni mese job4	24
Figura 46 – Frammento delle operazioni di mapping e shuffling job4	25
Figura 47 – Frammento dell’operazione di merging job4	25
Figura 48 – Implementazione reduce per selezione del mese di ogni anno con vendita minore	26

Figura 49 – Output del paradigma MapReduce a seguito dell'operazione di reducing job4.....	26
Figura 50 – Confronto dei risultati job4 dell'implementazione in R e in MapReduce.....	26

Introduzione

La presente relazione ha lo scopo di illustrare lo svolgimento di un'analisi effettuata su un esempio di documento relativo alla contabilità di un'azienda.

Ho scelto di svolgere tale esercitazione poiché avevo il desiderio di apprendere le basi dell'analisi dei dati su un esempio che ritengo sia rispondente alla realtà, ma soprattutto di interesse rilevante per la gestione delle aziende in generale, indipendentemente dal pubblico cui si rivolgono.

La traccia dell'esercitazione è la seguente:

La ditta STECCAPARAPETUTTI S.R.L. ha un sistema di contabilità che memorizza tutti gli ordini fatti (per ordini si intendono Fatture, Buoni Prelievo, Offerte...). In previsione dell'anno successivo, la ditta vorrebbe sapere il suo andamento di vendita.

Scaricato il file Ordini.csv strutturato in questa maniera:

Tipo-documento,data(aaaammgg),costo(€)

FATTURA, 20160104,139.8

FATTURA, 20160104,169.2

FATTURA, 20160104,65.3

.....

Le analisi richieste sono:

1. Calcola la media di vendita per ogni mese di ogni anno
2. Calcola la varianza di vendita per ogni mese di ogni anno
3. Identifica il mese di ogni anno con maggiore vendita
4. Identifica il mese di ogni anno con minore vendita

Per ognuna delle quali è necessario illustrare e documentare:

- Implementazione R
- Implementazione del paradigma mapReduce di HADOOP
(in alternativa tramite l'utilizzo del TOOL)

Per lo svolgimento verrà utilizzato il Linguaggio R sull'ambiente di sviluppo RStudio.

Il Linguaggio R nasce a metà degli anni '90 e si colloca tra i primi 10 linguaggi di programmazione più popolari al mondo.

Tale linguaggio è uno strumento computazionale fondamentale per la ricerca in diversi campi, tra cui statistica, biologia, fisica, matematica, chimica, economia, geologia e medicina.

Le capacità di R spaziano dalla semplice implementazione di metodi statistici ad una enormità di funzionalità con applicazioni che raggiungono l'intelligenza artificiale moderna e lo sviluppo del web.

A supporto dei programmatori vi sono varie IDE ed interfacce utente grafiche (GUI) per implementare, eseguire e testare il proprio lavoro.

Tali IDE mostrano nella stessa finestra: il codice in fase di scrittura, la console, gli output grafici ed altri elementi che mostrano, ad esempio, il file system.

RStudio è uno degli ambienti di sviluppo R più popolari, nato nel 2011, pensato inizialmente come IDE Open Source specifico per R; Successivamente si è evoluto come ambiente bilingue, concentrandosi su R e Python.

È disponibile in versione Desktop, la quale combina una console, un editor di evidenziazione della sintassi con la funzione di completamento delle schede, grafica, cronologia e guida, in un unico ambiente di lavoro.

È inoltre capace di mescolare il codice a documentazione testuale. (1)

Parallelamente, ogni *Job* verrà eseguito tramite l'implementazione del paradigma MapReduce in Linguaggio JavaScript.

Il paradigma MapReduce nasce nel 2004 in seguito all'identificazione delle seguenti necessità:

- Processare grandi quantità di dati
- Utilizzo di molte CPU, nell'ordine di centinaia o migliaia
- Rendere tutto ciò il più semplice che sia possibile

MapReduce è un nuovo modello di programmazione distribuita in grado di risolvere problemi di analisi su grandi insiemi di dati (le stesse istruzioni vengono eseguite su tanti record differenti).

Sostanzialmente è necessario ridurre un codice in due sole funzioni:

- Map Function
- Reduce Function

Ciò rende i programmi intrinsecamente parallelizzabili. (2)

Il tool MapReduce.html, che verrà utilizzato per lo svolgimento dell'analisi qui illustrata, è stato sviluppato dal Prof. Daniele Pirrone, con lo scopo di sopperire alle difficoltà di installazione del sistema Hadoop su PC, e permette di simulare un programma MapReduce attraverso un generico browser internet, poiché sviluppato in JavaScript con interfaccia HTML.

Essendo uno strumento pensato per sole finalità didattiche, presenta alcuni limiti:

1. Permette una simulazione del paradigma di programmazione parallela MapReduce ma in realtà lavora in modalità stand alone, non distribuita.
2. Permette di utilizzare file di input di dimensioni limitate alla capacità del computer su cui gira.
3. Gestisce in input solo file di testo.

Svolgimento

1. Calcola la media di vendita per ogni mese di ogni anno

Preparazione dei dati ed implementazione Job1 in R

Importo il documento Ordini.csv nell'ambiente di sviluppo RStudio.

Tramite l'istruzione `read.csv` lo carico in un `data.frame` chiamato `ordini`.

Da una prima osservazione qualitativa emerge che:

- Il file è composto da 5600 istanze a 3 variabili
- Non è presente l'intestazione delle colonne
- È necessario ispezionare il file per conoscere tutte le tipologie di documento

```
1 # Importo il file csv
2 ordini <- read.csv("/Users/giorgianesci/Documents/Uninettuno/Introduzione ai Big Data/Progetto Nesci/Ordini.csv", sep=";", header= FALSE)
3
4 #Imposto l'intestazione per le colonne
5 colnames(ordini) <- c('Tipo', 'Data', 'Prezzo')
6
7 #Conto la quantità di documenti per ogni tipologia
8 table(ordini$Tipo)
9
10
```

Figura 1 – Aggiunta intestazioni colonne ed analisi tipologie documenti

L'argomento `header=TRUE` (presente nella funzione `read.csv`) specifica che la prima linea del file non contiene i nomi delle variabili, le quali sono state aggiunte grazie alla funzione `colnames`.

In seguito alla chiamata della funzione `table(ordini$Tipo)` avremo il seguente output:

```
> table(ordini$Tipo)
BUONO.PRELIEVO      DDT      FATTURA  INVENTARIO NOTA.DI.CREDITO  OFFERTA  PREVENTIVO
      2146         15       2556           1             9         622         240
RICEVUTA
      11
```

Figura 2 – Output funzione `table(ordini$Tipo)`

Sulla base di tale risultato è possibile capire la tipologia di documenti da selezionare per effettuare le stime relative ai ricavi dell'azienda.

Si procede creando un `data.frame` composto esclusivamente dai documenti di tipo `Fattura` e `Ricevuta`, poiché sono gli unici rilevanti ai fini della nostra analisi.

Tale `data.frame`, sottoinsieme della tabella `ordini`, verrà denominato `ricavi` e sarà composto da 2567 istanze a 3 variabili.

```
11
12 # Selezione degli oggetti di tipo "FATTURA" e "RICEVUTA"
13 ricavi <- subset(ordini, Tipo=="FATTURA" | Tipo=="RICEVUTA")
14
```

Figura 3 – Creazione DataFrame "ricavi"

A questo punto è stata creata la base su cui procedere.

Per il calcolo della media di vendita di ogni mese di ogni anno, viene in nostro soccorso la funzione `aggregate`.

Tale funzione permette di calcolare statistiche di riepilogo per sottoinsiemi di dati, anche inserendo una formula, e fornisce un output di tipo `data.frame`. (3)

Nel nostro caso la media di vendita va calcolata per ogni mese di ogni anno, i quali rappresentano dei sottoinsiemi di `ricavi`.

	Tipo	Data	Prezzo
1	FATTURA	20160104	139.80
2	FATTURA	20160104	160.92
3	FATTURA	20160104	65.03

Figura 4 – Frammento del DataFrame “ricavi”

[Figura 4] ci mostra come è strutturato il `data.frame ricavi`.

È possibile osservare che la media mensile va calcolata *aggregando* i dati sulla base dei caratteri da 1 a 6 della variabile `Data`, corrispondente alla colonna n. 2.

L'istruzione da utilizzare è la seguente:

```
15  
16 # Fatturato medio di ogni mese di ogni anno  
17 media_mesi = aggregate(ricavi$Prezzo, by = list(substr(ricavi[,2], 1, 6)), FUN=mean)  
18
```

Figura 5 – Calcolo media di ogni mese di ogni anno

L'argomento `FUN` indica la funzione utilizzata per calcolare le statistiche di riepilogo che vengono applicate a tutti i sottoinsiemi di dati (`Mesi`), in questo caso `FUN=mean` (funzione media).

Si produrrà il seguente `data.frame`, dopo aver opportunamente rinominato le colonne:

	Mesi	Fatturato_medio
1	201601	179.93081
2	201602	131.92288
3	201603	222.55800

Figura 6 – Frammento del DataFrame “media_mesi”

che salveremo in un file di testo di output denominato `Ordini_media_mesi.csv`

[Appendice: Contabilità_job_1_2_3_4.R – 23]

È possibile visualizzare in modo grafico il risultato che abbiamo appena salvato nel `data.frame` `media_mesi`. [Appendice: Contabilità_job_1_2_3_4.R – 27]

La funzione `barplot` consente di creare grafici a barre, i quali sono una rappresentazione grafica dei dati che presenta dati categoriali con barre rettangolari con altezze o lunghezze proporzionali ai valori che rappresentano.

Questi set di dati contengono i valori numerici delle variabili che rappresentano l'altezza o la lunghezza. (4)

In [Figura 7] possiamo osservare il grafico a barre rappresentante la media mensile di vendita.

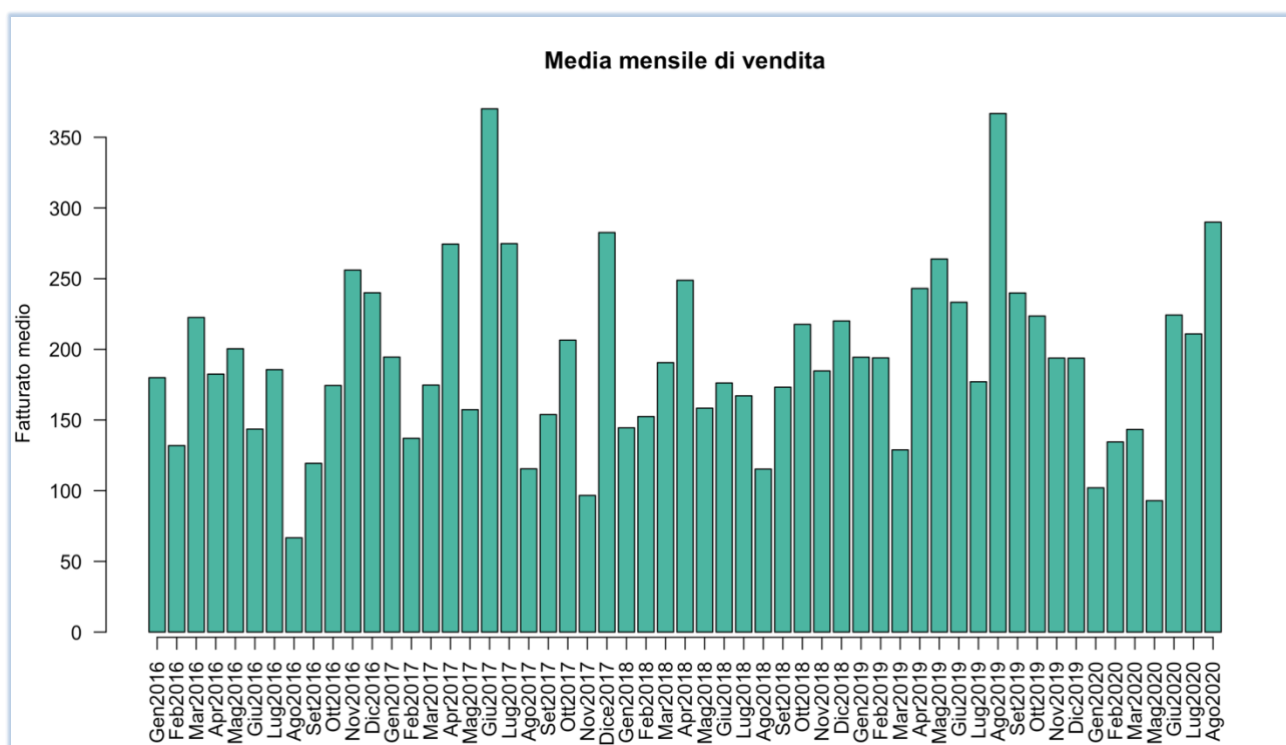


Figura 7 – Grafico della media mensile di vendita

Implementazione del paradigma MapReduce per il Job1

Per quanto riguarda l'implementazione del paradigma MapReduce, le considerazioni per la preparazione dei dati, partendo dal file originale Ordini.csv, sono analoghe, ma in questo caso non abbiamo necessità di disporre delle intestazioni delle colonne.

Va osservata la struttura del file per comprendere in che modo suddividere il testo per effettuare le operazioni in modo ottimale.

Il testo si presenta nel seguente modo:

Upload Text File or drop below	
FATTURA,20160104,139.8	
FATTURA,20160104,160.92	
FATTURA,20160104,65.03	
FATTURA,20160104,535.84	
FATTURA,20160104,75.68	

Figura 8 – Frammento della struttura originaria file Ordini.csv fornita in input

Per poter operare sull'input è necessario suddividere il testo in sottoinsiemi su cui andrà a lavorare in parallelo la funzione Map.

È facile notare che i vari sottoinsiemi sono ricavabili dividendo l'input ogni volta che si presenta il carattere '\n' (comando per “andare a capo”).

È in questo momento che va operata anche la selezione degli elementi di tipo FATTURA e RICEVUTA: divido ulteriormente i sottoinsiemi ogni volta che incontro il carattere ','.

Abbiamo quindi degli array la cui prima posizione è occupata dalla specifica della tipologia del documento registrato (che nel data.frame ordini dell'implementazione in R abbiamo chiamato 'Tipo'), su cui possiamo applicare la funzione filter.

```
[Appendice: MapReduce_Job1.js - function jobInputSplit(input_str)]
```

A questo punto è possibile procedere con le considerazioni per la scrittura della funzione jobMap [Appendice: MapReduce_Job1.js - function jobMap(V_In_Map)]:

V_In_Map è l'input iniziale opportunamente manipolato al punto precedente.

Divido i sottoinsiemi ogni volta che incontro il carattere ',', definendo una variabile chiamata ordini. Avrò così degli array del tipo ['Tipo', 'aaaammgg', 'Prezzo'].

Per calcolare la media di vendita di ogni mese di ogni anno, le variabili key saranno composte dai primi 6 caratteri di ordini[1], quindi key = 'aaaamm'.

Le variabili value saranno costituite da ordini[2], quindi value = 'Prezzo'.

Verrà così creata una mappa `keyVal (key, value)`, opportunamente ordinata secondo valori crescenti (mapping e shuffling):

MAPPING	SHUFFLING
« KEY - VALUE »	« KEY - VALUE »
« 201601 - 139.8 »	« 201601 - 106.6 »
« 201601 - 160.92 »	« 201601 - 108.02 »
« 201601 - 65.03 »	« 201601 - 109.47 »
« 201601 - 535.84 »	« 201601 - 11.13 »
« 201601 - 75.68 »	« 201601 - 11.37 »
« 201601 - 332.28 »	« 201601 - 113.02 »
« 201601 - 343.02 »	« 201601 - 13.81 »
« 201601 - 247.84 »	« 201601 - 132.2 »
« 201601 - 75.85 »	« 201601 - 139.8 »
« 201601 - 244.39 »	« 201601 - 145.85 »
« 201601 - 36.16 »	« 201601 - 15.4 »
« 201601 - 34.53 »	« 201601 - 159.43 »
« 201601 - 51.37 »	« 201601 - 160.65 »

Figura 9 – Frammento delle operazioni di mapping e shuffling job1

La mappa `keyVal (key, value)` verrà fornita come argomento della funzione `jobReduce`
 [Appendice: `MapReduce_Job1.js - function jobReduce(K_In_Reduce_V_In_Reduce)] :`

Vengono ordinate tutte le coppie key-value in modo da individuare i valori che puntano alla stessa chiave (merging):

MERGING
« KEY - VALUE »
« 201601 - 106.6, 108.02, 109.47, 11.13, 11.37, 113.02, 13.81, 132.2, 139.8, 145.85, 15.4, 159.43, 160.65, 160.92, 177.71, 179.19, 180.28, 19.15, 204.55, 22.4, 226.79, 235.42, 24.52, 244, 244.39, 247.21, 247.84, 252.44, 252.7, 2537.6, 26.94, 271.83, 294.92, 32.37, 332.28, 34.53, 343.02, 35.49, 36.16, 36.88, 360.17, 37.05, 42.5, 503.81, 51.37, 52.24, 53.94, 535.84, 55.66, 57.44, 59.62, 635.4, 65.03, 68.99, 7.32, 7.38, 7.88, 75.68, 75.85, 80.15, 96.94, 97.17 »
« 201602 - 104.86, 11.36, 112.98, 12.49, 122.31, 14.38, 14.47, 14.96, 142.06, 143.81, 148.53, »

Figura 10 – Frammento dell'operazione di merging job1

Ora che abbiamo unito tutti i valori che puntavano alla stessa chiave, è necessario implementare la funzione `reduce` in modo congeniale al nostro scopo: fare in modo che calcoli la somma di tutti i valori associati ad una chiave, ed in fine ne calcoli la media.

Con `V_In_Reduce` = lista di valori che puntano alla stessa chiave, abbiamo:

```

32.
33. var Reduce = V_In_Reduce.reduce(function (accumulator, item) {
34.     return parseFloat(accumulator) + parseFloat(item);
35. });
36. var media = parseFloat(Reduce/V_In_Reduce.length);
37.
38.

```

Figura 11 – implementazione `reduce` per calcolo media di vendita per ogni mese di ogni anno

La funzione `reduce` andrà a lavorare in parallelo su ogni chiave.

Verrà fornita in output la mappa `keyVal (K_In_Reduce, media.toFixed(5))`.

Con `K_In_Reduce` = lista delle chiavi fornite in input, abbiamo il seguente output:

REDUCING	
« KEY - VALUE »	
« 201601 - 179.93081 »	
« 201602 - 131.92288 »	
« 201603 - 222.55800 »	
« 201604 - 182.44100 »	
« 201605 - 200.41270 »	
« 201606 - 143.58000 »	
« 201607 - 185.62951 »	
« 201608 - 66.66857 »	
« 201609 - 119.33484 »	
« 201610 - 174.44197 »	
« 201611 - 256.04695 »	
« 201612 - 240.04896 »	
« 201701 - 194.52074 »	
« 201702 - 137.06327 »	
« 201703 - 174.72517 »	

Figura 12 – Frammento dell'output del paradigma MapReduce a seguito dell'operazione di `reducing job1`

I valori ottenuti in [Figura 12] possono essere confrontati con quelli di [Figura 6] per constatarne la corrispondenza:

	Mesi	Fatturato_medio	« KEY - VALUE »	
1	201601	179.93081	« 201601 - 179.93081 »	
2	201602	131.92288	« 201602 - 131.92288 »	
3	201603	222.55800	« 201603 - 222.55800 »	
4	201604	182.44100	« 201604 - 182.44100 »	

Figura 13 – Confronto del frammento dei risultati `job1` dell'implementazione in R e in MapReduce

2. Calcola la varianza di vendita per ogni mese di ogni anno

Implementazione Job2 in R

Prima di iniziare, è necessario fornire alcune definizioni di *varianza* per poterne comprendere appieno il significato.

“La varianza fornisce una misura della possibile deviazione dalla media”. (5)

“La varianza identifica la dispersione dei valori della variabile attorno al valor medio. Tanto è più piccola la varianza, tanto più i valori della variabile sono concentrati attorno al valor medio”. (6)

Per eseguire l’analisi richiesta dal Job n. 2, è possibile usufruire del `data.frame` `ricavi` creato nel corso del paragrafo 1, del quale si può ricordare la struttura osservando [Figura 4].

Anche in questo caso, è possibile notare che la varianza mensile va calcolata *aggregando* i dati sulla base dei caratteri da 1 a 6 della variabile `Data`, corrispondente alla colonna n. 2.

L’istruzione da utilizzare è la seguente:

```
31  
32 # Varianza di vendita di ogni mese di ogni anno  
33 varianza_mesi = aggregate(ricavi$Prezzo, by = list(substr(ricavi[,2], 1, 6)), FUN=var)  
34
```

Figura 14 – Calcolo varianza di ogni mese di ogni anno

L’argomento `FUN` indica la funzione utilizzata per calcolare le statistiche di riepilogo che vengono applicate a tutti i sottoinsiemi di dati (`Mesi`), in questo caso `FUN=var` (funzione varianza).

Si produrrà il seguente `data.frame`, dopo aver opportunamente rinominato le colonne:

	Mesi	Varianza
1	201601	110972.691
2	201602	38351.044
3	201603	295872.781

Figura 15 – Frammento del DataFrame “varianza_mesi”

che salveremo in un file di testo di output denominato `Ordini_varianza_mesi.csv`.

[Appendice: Contabilità_job_1_2_3_4.R – 38]

È possibile visualizzare in modo grafico il risultato che abbiamo appena salvato nel `data.frame` `varianza_mesi`. [Appendice: Contabilità_job_1_2_3_4.R – 41]

Utilizziamo la funzione `barplot`, già descritta nel paragrafo 1, per realizzare il grafico a barre rappresentante la varianza mensile di vendita [Figura 16].

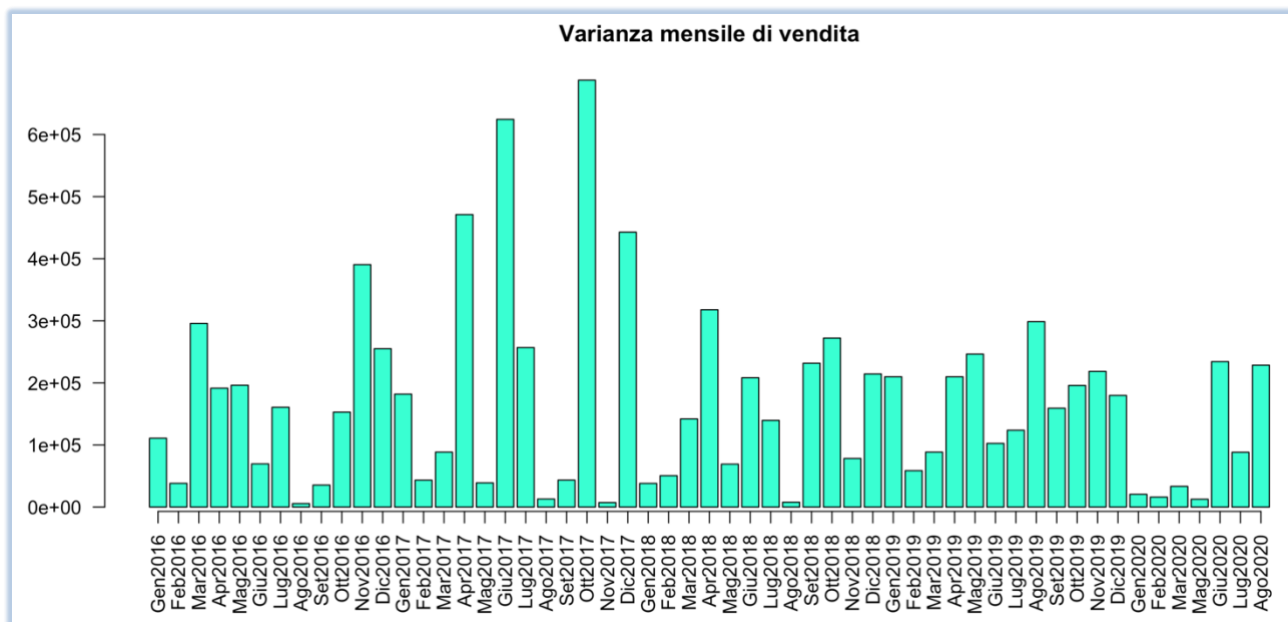


Figura 16 – Grafico della varianza mensile di vendita

Implementazione del paradigma MapReduce per il Job2

Per quanto riguarda l'implementazione del paradigma MapReduce, per il calcolo della varianza possiamo usufruire delle implementazioni delle funzioni `jobInputSplit(input_str)` e `jobMap(V_In_Map)`, relative al Job1, ottenendo i medesimi risultati illustrati in [Figura 9].

La mappa `keyVal(key, value)` verrà fornita come argomento della funzione `jobReduce` [Appendice: MapReduce_Job2.js - `function jobReduce(K_In_Reduce_V_In_Reduce)`]:

Vengono ordinate tutte le coppie key-value in modo da individuare i valori che puntano alla stessa chiave (merging) [Figura 10].

Una volta uniti tutti i valori che puntano alla stessa chiave, è necessario implementare la funzione `reduce` in modo congeniale al nostro scopo.

In questo caso non disponiamo di una funzione preimpostata; risulta quindi necessario uno studio preventivo sulla modalità di calcolo per la varianza.

Quando si lavora su un campione, si utilizza la seguente formula:

$$\sigma^2 = \frac{\sum (x_i - \bar{x})^2}{n - 1}$$

Con:

- σ^2 = la varianza, che viene sempre misurata in unità al quadrato
- x_i = rappresenta un dato del campione
- \bar{x} = il valor medio del campione
- $\sum (x_i - \bar{x})^2$ = devianza
- n = il numero di dati che compongono l'insieme

Si procede, quindi, con i seguenti passaggi:

1. Calcolo la media di ogni mese di ogni anno

```
45 |  
46 | var totale_mesi = V_In_Reduce.reduce(function (accumulator, item) {  
47 |     return parseFloat(accumulator) + parseFloat(item);  
48 | });  
49 | var media = parseFloat(totale_mesi/V_In_Reduce.length);  
50 |
```

Figura 17 – Implementazione `reduce` per il calcolo della media di ogni mese di ogni anno

2. Calcolo i risultati parziali della devianza $= (x_i - \bar{x})^2$

```
51 |  
52 | var devianza_parziali = V_In_Reduce.map(function(item){  
53 |     return parseFloat(Math.pow((item-media), 2));  
54 | });  
55 |
```

Figura 18 – Implementazione `map` per il calcolo dei risultati parziali della devianza

3. Calcolo la devianza

```

56 var devianza = devianza_parziali.reduce(function (accumulator, item){
57   return parseFloat(accumulator) + parseFloat(item);
58 });
59
60

```

Figura 19 – Implementazione reduce per il calcolo della devianza

4. Effettuo il passaggio finale per il calcolo della varianza

```

61
62 var varianza = parseFloat(devianza / (V_In_Reduce.length - 1));
63
64

```

Figura 20 – Implementazione ultimo passaggio per il calcolo della varianza

Verrà fornita in output la mappa keyVal (K_In_Reduce, varianza.toFixed(3)) :

REDUCING	
« KEY - VALUE »	
« 201601 - 110972.691 »	
« 201602 - 38351.044 »	
« 201603 - 295872.781 »	
« 201604 - 191570.685 »	
« 201605 - 196337.279 »	
« 201606 - 69653.870 »	
« 201607 - 160827.843 »	
« 201608 - 5731.737 »	
« 201609 - 35471.358 »	
« 201610 - 152986.941 »	
« 201611 - 390485.050 »	
« 201612 - 255096.324 »	
« 201701 - 181979.971 »	
« 201702 - 43393.861 »	
« 201703 - 88570.205 »	

Figura 21 – Frammento dell'output del paradigma MapReduce a seguito dell'operazione di reducing job2

I valori ottenuti in [Figura 21] possono essere confrontati con quelli di [Figura 15] per constatarne la corrispondenza:

	Mesi	Varianza	« KEY - VALUE »
1	201601	110972.691	« 201601 - 110972.691 »
2	201602	38351.044	« 201602 - 38351.044 »
3	201603	295872.781	« 201603 - 295872.781 »
4	201604	191570.685	« 201604 - 191570.685 »

Figura 22 – Confronto del frammento dei risultati job2 dell'implementazione in R e in MapReduce

3. Identifica il mese di ogni anno con maggiore vendita

Implementazione Job3 in R

Per eseguire l'analisi richiesta dal Job n. 3, è necessario conoscere l'importo totale dei ricavi di ogni mese di ogni anno.

La funzione `aggregate` viene nuovamente in nostro soccorso, usufruendo del `data.frame` `ricavi` creato in precedenza, il quale è un sottoinsieme dell'input.

I ricavi mensili vengono calcolati *aggregando* i dati sulla base dei caratteri da 1 a 6 della variabile `Data`, corrispondente alla colonna n. 2.

L'istruzione da utilizzare è la seguente:

```
46  
47 # Fatturato totale di ogni mese di ogni anno  
48 somma_mesi = aggregate(ricavi$Prezzo, by = list(substr(ricavi[,2], 1, 6)), FUN=sum)  
49
```

Figura 23 – Calcolo fatturato totale di ogni mese di ogni anno

L'argomento `FUN` indica la funzione utilizzata per calcolare le statistiche di riepilogo che vengono applicate a tutti i sottoinsiemi di dati (`Mesi`), in questo caso `FUN=sum` (funzione somma).

Si produrrà il seguente `data.frame`, dopo aver opportunamente rinominato le colonne:

	Mesi	Fatturato_totale
1	201601	11155.71
2	201602	8706.91
3	201603	12240.69

Figura 24 – Frammento del DataFrame "somma_mesi"

che salveremo in un file di testo di output denominato `Ordini_somma_mesi.csv`.

[Appendice: Contabilità_job_1_2_3_4.R – 53]

Grazie alla funzione `barplot` possiamo ottenere una rappresentazione grafica dei risultati appena ottenuti [Figura 25] [Appendice: Contabilità_job_1_2_3_4.R – 57]:

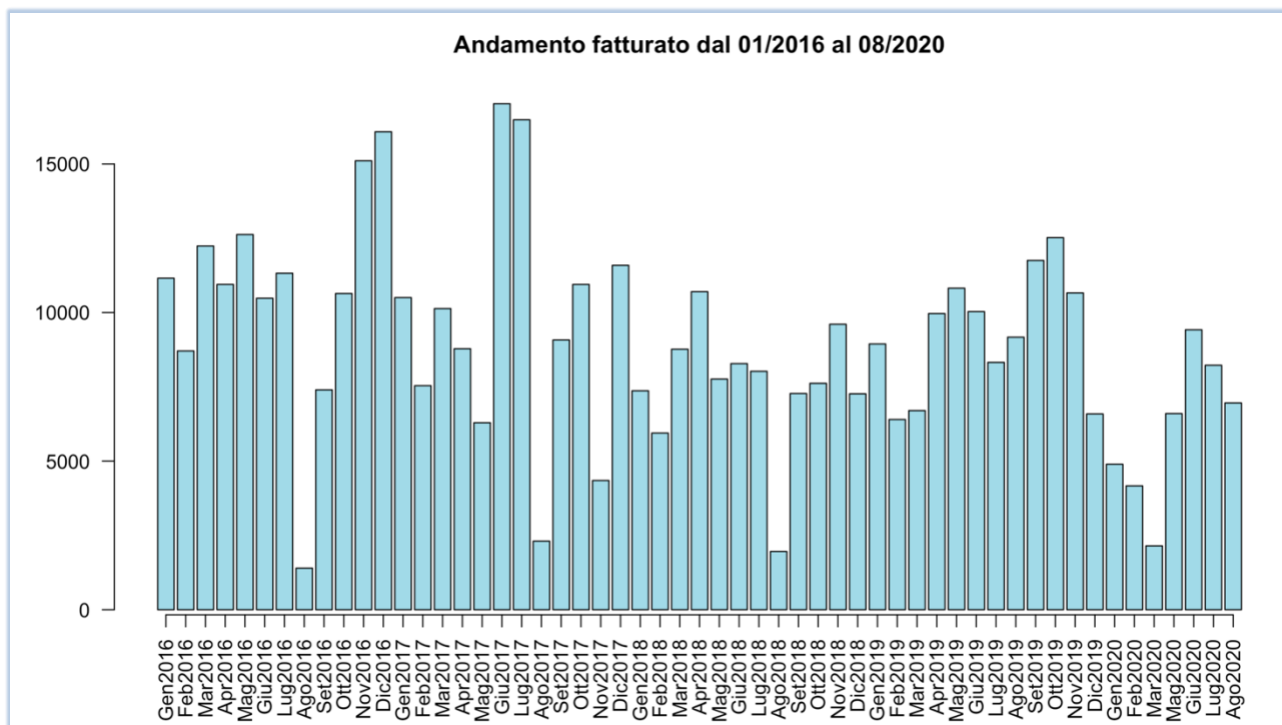


Figura 25 – Grafico dell'andamento del fatturato da Gen2016 ad Ago2020

A questo punto possiamo procedere con l'analisi richiesta dalla consegna.

Per ottenere un `data.frame` in cui viene evidenziato il mese di ogni anno in cui il ricavo è stato massimo, risulta necessario effettuare due passaggi.

1. Nel primo passaggio ci avvaliamo della funzione `aggregate` per selezionare il valore massimo della colonna n. 2 del `data.frame` `somma_mesi`, *aggregando* i dati sulla base dai valori da 1 a 4 della variabile `Mesi`, corrispondente alla colonna n. 1.
L'istruzione da utilizzare è la seguente:

```
77
78 # Importo mensile massimo di ogni anno
79 mesi_max_importoonly = aggregate(somma_mesi$Fatturato_totale, by = list(substr(somma_mesi[,1], 1, 4)), FUN=max)
80
```

Figura 26 – Calcolo del ricavo mensile massimo di ogni anno

L'argomento `FUN` indica la funzione utilizzata per calcolare le statistiche di riepilogo che vengono applicate a tutti i sottoinsiemi di dati (Anno), in questo caso `FUN=max` (funzione valore massimo).

Si produrrà il seguente `data.frame`, dopo aver opportunamente rinominato le colonne:

	Anno	Fatturato_totale
1	2016	16083.28
2	2017	17028.09
3	2018	10701.99
4	2019	12520.92
5	2020	9420.84

Figura 27 – DataFrame “mesi_max_importonly”

Si noti l’intestazione della variabile corrispondente alla colonna n. 2 del `data.frame` mostrato nella figura precedente [Figura 27]: la denominazione `Fatturato_totale` non è corretta in termini di logica, in quanto sarebbe conveniente utilizzare la denominazione `Fatturato_mensile_max`, ma in questo passaggio ci risulta utile il nome effettivamente utilizzato per poter eseguire al meglio il passaggio successivo.

2. Nel secondo passaggio ci avvaliamo per la prima volta della funzione `merge`.
Tale funzione consente di unire due `data.frame` per colonne comuni o per nomi di riga.
Questa funzione consente di eseguire diversi join di database (SQL), come `left join`, `inner join`, `right join` o `full join`. (7)

Vogliamo ottenere un `data.frame` che ci comunichi non solo il valore del ricavo massimo mensile ottenuto, ma soprattutto a quale mese corrisponda.

Per fare ciò, l’istruzione da utilizzare è la seguente:

```
86  
87 mesi_max = merge(mesi_max_importonly, somma_mesi, by="Fatturato_totale")  
88
```

Figura 28 – Creazione DataFrame “mesi_max”

I primi due argomenti indicano i due `data.frame` che vogliamo unire;
l’argomento `by` indica la denominazione della colonna che deve essere utilizzata per l’unione:
individuare i valori comuni tra i due `data.frame` della variabile `Fatturato_totale`, e su questa base effettuare l’operazione `merge`.

Si produrrà il seguente `data.frame` (dopo aver rinominato la prima colonna):

	Fatturato_mensile_max	Anno	Mesi
4	16083.28	2016	201612
5	17028.09	2017	201706
2	10701.99	2018	201804
3	12520.92	2019	201910
1	9420.84	2020	202006

Figura 29 – `DataFrame` “mesi_max”

che salveremo in un file di testo di output denominato `Ordini_mesi_max.csv`
 [Appendice: `Contabilità_job_1_2_3_4.R` – 81]

Grazie alla funzione `barplot` possiamo ottenere il seguente grafico a barre rappresentante i dati contenuti nel `data.frame` `mesi_max` [Appendice: `Contabilità_job_1_2_3_4.R` – 77]:

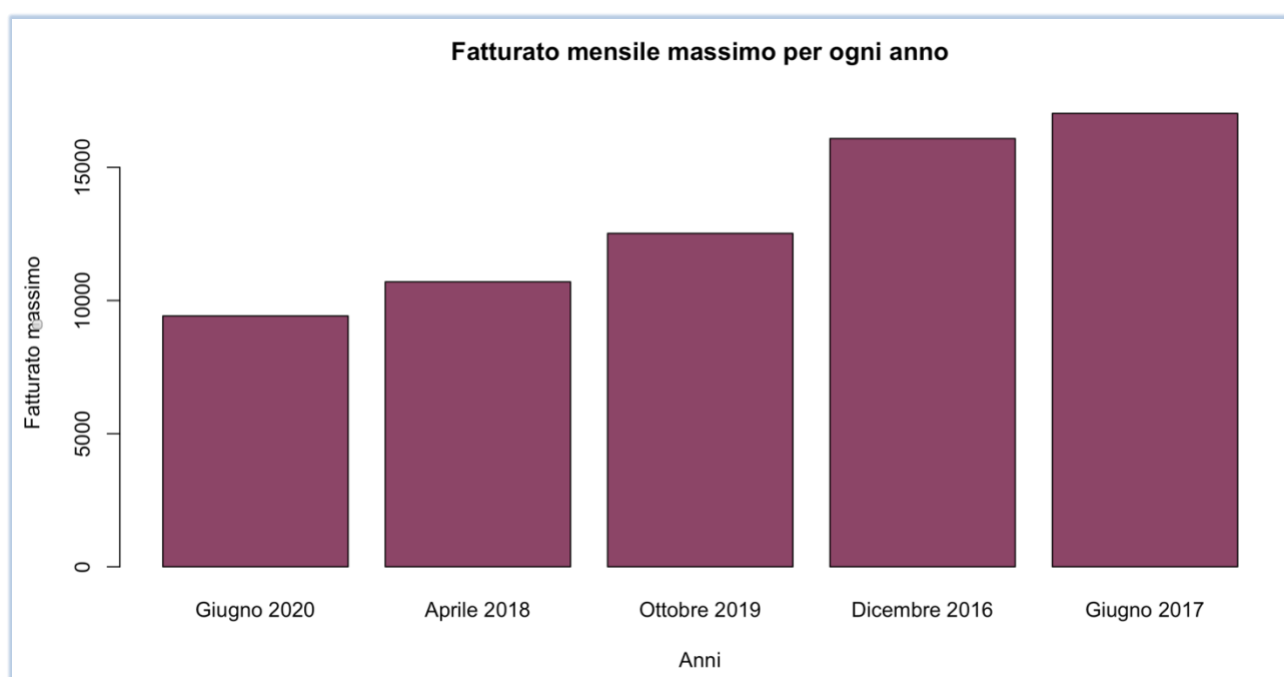


Figura 29 – Grafico del fatturato mensile massimo per ogni anno

Implementazione del paradigma MapReduce per il Job3

Per quanto riguarda l'implementazione del paradigma MapReduce che ci fornisca il mese di ogni anno in cui il ricavo sia stato massimo, abbiamo bisogno, anche in questo caso di produrre un risultato intermedio.

Come si è fatto per l'implementazione in R, con la creazione del `data.frame somma_mesi` [Figura 24], andremo ad implementare un paradigma MapReduce in cui la funzione `jobMap` sarà la medesima dei job 1 e 2, e la funzione `jobReduce` si focalizzerà solo sulla restituzione di una mappa in cui le `key`, composte dalla lista dei mesi del periodo di riferimento, saranno associate ad un singolo `value` corrispondente al ricavo totale di ogni mese.

[Appendice: `MapReduce_Job3_Totale_mesi.js` - `function jobReduce(K_In_Reduce_V_In_Reduce)`]

```
43  
44 var totale_mesi = V_In_Reduce.reduce(function (accumulator, item) {  
45     return parseFloat(accumulator) + parseFloat(item);  
46 });  
47 return keyVal(K_In_Reduce, totale_mesi.toFixed(2));  
48
```

Figura 30 – Implementazione `reduce` per il calcolo del ricavo totale di ogni mese `job3`

La mappa `keyVal` fornita in output è la seguente:

REDUCING	
« KEY - VALUE »	
« 201601 - 11155.71 »	
« 201602 - 8706.91 »	
« 201603 - 12240.69 »	
« 201604 - 10946.46 »	
« 201605 - 12626.00 »	
« 201606 - 10481.34 »	
« 201607 - 11323.40 »	
« 201608 - 1400.04 »	
« 201609 - 7398.76 »	
« 201610 - 10640.96 »	
« 201611 - 15106.77 »	
« 201612 - 16083.28 »	
« 201701 - 10504.12 »	
« 201702 - 7538.48 »	
« 201703 - 10134.06 »	

Figura 31 – Frammento dell'output del paradigma MapReduce a seguito dell'operazione di `reducing` per il risultato intermedio `job3`

I risultati ottenuti in [Figura 31] possono essere confrontati con quelli di [Figura 24] per constatarne la corrispondenza:

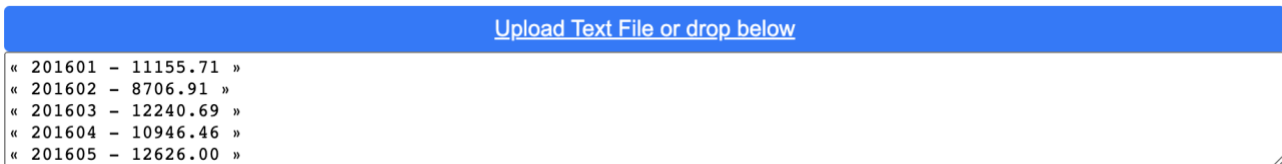
			« KEY - VALUE »
	Mesi	Fatturato_totale	
1	201601	11155.71	« 201601 - 11155.71 » « 201602 - 8706.91 » « 201603 - 12240.69 » « 201604 - 10946.46 »
2	201602	8706.91	
3	201603	12240.69	
4	201604	10946.46	

Figura 32 – Confronto del frammento dei risultati intermedi `job3` dell'implementazione in R e in MapReduce

A questo punto, la mappa generata in output dal paradigma MapReduce può essere fornita in input all'implementazione MapReduce che servirà per produrre i risultati richiesti.

Va osservata la struttura del file per comprendere in che modo suddividere il testo per effettuare le operazioni in modo ottimale.

Il testo si presenta nel seguente modo:



```
« 201601 - 11155.71 »
« 201602 - 8706.91 »
« 201603 - 12240.69 »
« 201604 - 10946.46 »
« 201605 - 12626.00 »
```

Figura 33 – Frammento della struttura originaria del file fornito in input

Per poter operare sull'input è necessario suddividere il testo in sottoinsiemi su cui andrà a lavorare in parallelo la funzione Map.

È facile notare che i vari sottoinsiemi sono ricavabili dividendo l'input ogni volta che si presenta il carattere '\n' (comando per “andare a capo”).

[Appendice: MapReduce_Job3.js - function jobInputSplit(input_str)]

A questo punto è possibile procedere con le considerazioni per la scrittura della funzione jobMap

[Appendice: MapReduce_Job3.js - function jobMap(V_In_Map)] :

V_In_Map è l'input iniziale opportunamente manipolato al punto precedente.

Divido i sottoinsiemi ogni volta che incontro il carattere " - ", definendo una variabile chiamata somma_mesi. Avrò così degli array del tipo ['<< aaaamm', 'Fatturato_totale >>'].

È immediato notare come i primi 3 caratteri di somma_mesi[0] e gli ultimi 3 caratteri di somma_mesi[1] siano superflui e vadano eliminati.

Per trovare il mese di ogni anno con maggiore vendita, le variabili key saranno composte dai caratteri da 2 a 6 di somma_mesi[0], quindi key = 'aaaa'.

Le variabili value saranno composte da somma_mesi[1] cui vengono eliminati gli ultimi 3 caratteri, quindi value = 'Fatturato_totale'.

```
15.
16.   var key = somma_mesi[0].substring(2,6);
17.   var value = somma_mesi[1].slice(0, -2);
18.
```

Figura 34 – Definizione variabili key e value funzione jobMap del job3

Per poter restituire in output, come risultato finale di tutto il paradigma MapReduce, una mappa composta da `key = anno`, `value = mese_max`, la mappa da fornire in input alla funzione `jobReduce` avrà come valori di `key` i valori appena definiti; i `value` saranno così composti: i caratteri da 2 a 8 caratteri di `somma_mesi[0]` + la variabile `value` appena definita.

Verrà così creata una mappa `keyVal (key, somma_mesi[0].substring(2, 8)+value)`, opportunamente ordinata (mapping e shuffling):

MAPPING	SHUFFLING
« KEY - VALUE »	« KEY - VALUE »
« 2016 - 20160111155.71 »	« 2016 - 20160111155.71 »
« 2016 - 2016028706.91 »	« 2016 - 2016028706.91 »
« 2016 - 20160312240.69 »	« 2016 - 20160312240.69 »
»	»
« 2016 - 20160410946.46 »	« 2016 - 20160410946.46 »
»	»
« 2016 - 20160512626.00 »	« 2016 - 20160512626.00 »
»	»
« 2016 - 20160610481.34 »	« 2016 - 20160610481.34 »
»	»
« 2016 - 20160711323.40 »	« 2016 - 20160711323.40 »
« 2016 - 2016081400.04 »	« 2016 - 2016081400.04 »
« 2016 - 2016097398.76 »	« 2016 - 2016097398.76 »
« 2016 - 20161010640.96 »	« 2016 - 20161010640.96 »
»	»
« 2016 - 20161115106.77 »	« 2016 - 20161115106.77 »
« 2016 - 20161216083.28 »	« 2016 - 20161216083.28 »
»	»
« 2017 - 20170110504.12 »	« 2017 - 20170110504.12 »
« 2017 - 2017027538.48 »	« 2017 - 2017027538.48 »
« 2017 - 20170310134.06 »	« 2017 - 20170310134.06 »
»	»

Figura 35 – Frammento delle operazioni di mapping e shuffling job3

La mappa `keyVal` verrà fornita come argomento della funzione `jobReduce`

[Appendice: `MapReduce_Job3.js - function jobReduce(K_In_Reduce_V_In_Reduce)`]:

Vengono ordinate tutte le coppie `key-value` in modo da individuare i valori che puntano alla stessa chiave (merging):

MERGING
« KEY - VALUE »
« 2016 - 20160111155.71, 2016028706.91, 20160312240.69, 20160410946.46, 20160512626.00, 20160610481.34, 20160711323.40, 2016081400.04, 2016097398.76, 20161010640.96, 20161115106.77, 20161216083.28 »
« 2017 - 20170110504.12, 2017027538.48, 20170310134.06,

Figura 36 – Frammento dell'operazione di merging job3

Ora che abbiamo unito tutti i valori che puntavano alla stessa chiave, è necessario implementare la funzione reduce in modo congeniale al nostro scopo: fare in modo che ci comunichi quale sia il mese di ogni anno con maggiore vendita.

Con `V_In_Reduce` = lista di valori che puntano alla stessa chiave, abbiamo:

```
32. let n = 7;
33. var Reduce = V_In_Reduce.reduce (function (max, item){
34.   return parseFloat(1*max.substring(n)) > parseFloat(1*item.substring(n)) ? max : item;
35. });
36.
37. return keyVal( K_In_Reduce, Reduce.substring(0,7));
38.
```

Figura 37 – Implementazione reduce per selezione del mese di ogni anno con ricavo maggiore

La funzione reduce andrà a lavorare in parallelo su ogni chiave, confrontando i valori che puntano alla stessa chiave, cui sono stati eliminati i primi 6 caratteri (corrispondenti al mese di appartenenza), i quali verranno però selezionati per comporre i value della mappa `keyVal` fornita in output.

Verrà fornita in output la mappa `keyVal(K_In_Reduce, Reduce.substring(0, 7))`:

REDUCING
« KEY - VALUE »
« 2016 - 201612 »
« 2017 - 201706 »
« 2018 - 201804 »
« 2019 - 201910 »
« 2020 - 202006 »

Figura 38 – Output del paradigma MapReduce a seguito dell'operazione di reducing job3

I risultati ottenuti in [Figura 38] possono essere confrontati con quelli di [Figura 29] (colonne 2 e 3) per constatarne la corrispondenza:

Anno ▲	Mesi ▼	« KEY - VALUE »
2016	201612	
2017	201706	« 2016 - 201612 »
2018	201804	« 2017 - 201706 »
2019	201910	« 2018 - 201804 »
2020	202006	« 2019 - 201910 »
		« 2020 - 202006 »

Figura 39 – Confronto dei risultati job3 dell'implementazione in R e in MapReduce

3. Identifica il mese di ogni anno con minore vendita

Implementazione Job4 in R

Anche per eseguire l'analisi richiesta dal Job n. 4, è necessario conoscere l'importo totale dei ricavi di ogni mese di ogni anno, quindi possiamo avvalerci dei risultati ottenuti in precedenza, descritti da [Figura 23], [Figura 24] e [Figura 25].

A questo punto possiamo procedere con l'analisi richiesta dalla consegna.

Per ottenere un `data.frame` in cui viene evidenziato il mese di ogni anno in cui il ricavo è stato minimo, risulta necessario effettuare due passaggi:

1. Nel primo passaggio ci avvaliamo della funzione `aggregate` per selezionare il valore minimo della colonna n. 2 del `data.frame` `somma_mesi`, aggregando i dati sulla base dai valori da 1 a 4 della variabile `Mesi`, corrispondente alla colonna n. 1.

L'istruzione da utilizzare è la seguente:

```
103  
104 # Importo mensile minimo di ogni anno  
105 mesi_min_importonly = aggregate(somma_mesi$Fatturato_totale, by = list(substr(somma_mesi[,1], 1, 4)), FUN=min)  
106
```

Figura 40 – Calcolo del ricavo mensile minimo di ogni anno

L'argomento `FUN` indica la funzione utilizzata per calcolare le statistiche di riepilogo che vengono applicate a tutti i sottoinsiemi di dati (`Anno`), in questo caso `FUN=min` (funzione valore minimo).

Si produrrà il seguente `data.frame`, dopo aver opportunamente rinominato le colonne:

	Anno	Fatturato_totale
1	2016	1400.04
2	2017	2309.31
3	2018	1959.88
4	2019	6400.68
5	2020	2150.03

Figura 41 – DataFrame "mesi_min_importonly"

Si noti l'intestazione della variabile corrispondente alla colonna n. 2 del `data.frame` mostrato nella figura precedente [Figura 41]: la denominazione `Fatturato_totale` non è corretta in termini di logica, in quanto sarebbe conveniente utilizzare la denominazione `Fatturato_mensile_min`, ma in questo passaggio ci risulta utile il nome effettivamente utilizzato per poter eseguire al meglio il passaggio successivo.

2. Nel secondo passaggio ci avvaliamo della funzione `merge`.
Vogliamo ottenere un `data.frame` che ci comunichi non solo il valore del ricavo minimo mensile ottenuto, ma soprattutto a quale mese corrisponda.
Per fare ciò, l'istruzione da utilizzare è la seguente:

```
112  
113 mesi_min = merge(mesi_min_importonly, somma_mesi, by = 'Fatturato_totale')  
114
```

Figura 42 – Creazione DataFrame “mesi_min”

I primi due argomenti indicano i due `data.frame` che vogliamo unire;
l'argomento `by` indica la denominazione della colonna che deve essere utilizzata per l'unione: individuo i valori comuni tra i due `data.frame` della variabile `Fatturato_totale`, e su questa base effettuo l'operazione `merge`.

Si produrrà il seguente `data.frame` (dopo aver rinominato la prima colonna):

	Fatturato_mensile_min	Anno	Mesi
1	1400.04	2016	201608
4	2309.31	2017	201708
2	1959.88	2018	201808
5	6400.68	2019	201902
3	2150.03	2020	202003

Figura 43 – DataFrame “mesi_min”

che salveremo in un file di testo di output denominato `Ordini_mesi_min.csv`
[Appendice: Contabilità_job_1_2_3_4.R – 105].

Grazie alla funzione `barplot` possiamo ottenere il seguente grafico a barre rappresentante i dati contenuti nel `data.frame mesi_min`
[Appendice: Contabilità_job_1_2_3_4.R – 99]:

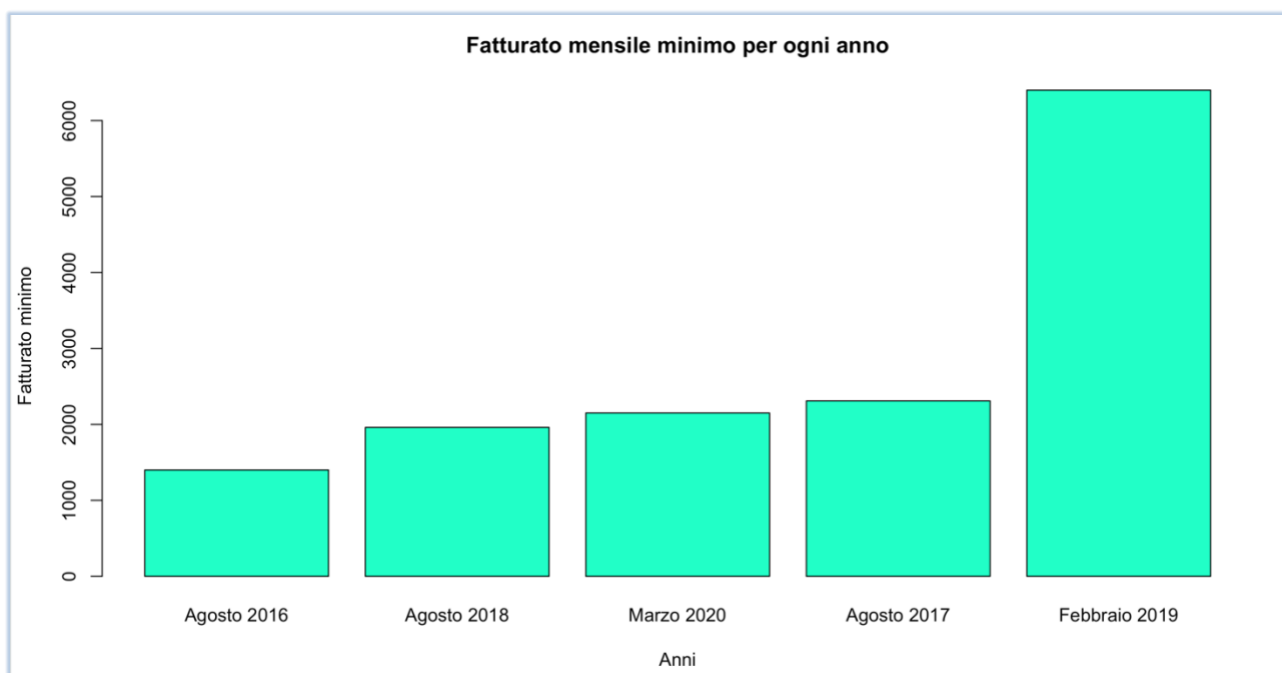


Figura 44 – Grafico del fatturato mensile minimo per ogni anno

Implementazione del paradigma MapReduce per il Job4

Per quanto riguarda l'implementazione del paradigma MapReduce che ci fornisca il mese di ogni anno in cui il ricavo sia stato minimo, abbiamo bisogno del risultato intermedio, da generare in modo simile all'implementazione osservabile in [Figura 30] anche se, in questo caso, per comodità elimino la parte decimale dai value della mappa fornita in output.

[Appendice: MapReduce_Job4_Totale_mesi.js - function jobReduce(K_In_Reduce_V_In_Reduce) - 30]

```
43.
44. var totale_mesi = V_In_Reduce.reduce(function (accumulator, item) {
45.     return parseFloat(accumulator) + parseFloat(item);
46. });
47. return keyVal(K_In_Reduce, totale_mesi.toFixed(0));
48.
```

Figura 45 – Implementazione reduce per il calcolo del ricavo totale di ogni mese job4

La mappa keyVal fornita in output è analoga a quella mostrata in [Figura 31], con value composti solo dalla parte intera del numero.

A questo punto, la mappa appena descritta può essere fornita in input all'implementazione MapReduce che servirà per produrre i risultati richiesti.

Restano validi i ragionamenti relativi alle modalità di divisione del testo di input per la generazione dei sottoinsiemi su cui andrà a lavorare in parallelo la funzione Map; possiamo quindi usufruire dell'implementazione della funzione `jobInputSplit(input_str)` relativa al Job 3.

Per la funzione `jobMap(V_In_Map)`, utilizziamo l'implementazione effettuata per il Job 3, fino al momento in cui vengono definite le variabili `key` e `value` [Figura 34].

[Appendice: MapReduce_Job4.js - function jobMap(V_In_Map) - 16]:

Per poter restituire in output, come risultato finale di tutto il paradigma MapReduce, una mappa composta da `Key = anno`, `Value = mese_min`, la mappa da fornire in input alla funzione `jobReduce` avrà come valori di `key` i valori appena definiti;

i value saranno così composti: la variabile `value` appena definita + i caratteri da 2 a 8 di `somma_mesi[0]`.

I value saranno quindi composti in modo inverso rispetto a quanto avvenuto per la funzione `jobMap(V_In_Map)` relativa al Job 3.

Verrà così creata una mappa keyVal (key, value+somma_mesi[0].substring(2,8)), opportunamente ordinata (mapping e shuffling):

MAPPING	SHUFFLING
« KEY - VALUE »	« KEY - VALUE »
« 2016 - 11156201601 »	« 2016 - 10481201606 »
« 2016 - 8707201602 »	« 2016 - 10641201610 »
« 2016 - 12241201603 »	« 2016 - 10946201604 »
« 2016 - 10946201604 »	« 2016 - 11156201601 »
« 2016 - 12626201605 »	« 2016 - 11323201607 »
« 2016 - 10481201606 »	« 2016 - 12241201603 »
« 2016 - 11323201607 »	« 2016 - 12626201605 »
« 2016 - 1400201608 »	« 2016 - 1400201608 »
« 2016 - 7399201609 »	« 2016 - 15107201611 »
« 2016 - 10641201610 »	« 2016 - 16083201612 »
« 2016 - 15107201611 »	« 2016 - 7399201609 »
« 2016 - 16083201612 »	« 2016 - 8707201602 »
« 2017 - 10504201701 »	« 2017 - 10134201703 »
« 2017 - 7538201702 »	« 2017 - 10504201701 »
« 2017 - 10134201703 »	« 2017 - 10946201710 »

Figura 46 – Frammento delle operazioni di mapping e shuffling job4

La mappa keyVal verrà fornita come argomento della funzione jobReduce

[Appendice: MapReduce_Job4.js - function jobReduce(K_In_Reduce_V_In_Reduce)] :

Vengono ordinate tutte le coppie key-value in modo da individuare i valori che puntano alla stessa chiave (merging):

MERGING
« KEY - VALUE »
« 2016 - 10481201606, 10641201610, 10946201604, 11156201601, 11323201607, 12241201603, 12626201605, 1400201608, 15107201611, 16083201612, 7399201609, 8707201602 » « 2017 - 10134201703, 10504201701, 10946201710,

Figura 47 – Frammento dell'operazione di merging job4

Ora che abbiamo unito tutti i valori che puntavano alla stessa chiave, è necessario implementare la funzione reduce in modo congeniale al nostro scopo: fare in modo che ci comunichi quale sia il mese di ogni anno con minore vendita.

Con `V_In_Reduce` = lista di valori che puntano alla stessa chiave, abbiamo:

```

30.
31. var Reduce = V_In_Reduce.reduce (function (max, item){
32.   return 1*max.slice(0, -6) < 1*item.slice(0, -6) ? max : item;
33. });
34. return keyVal( K_In_Reduce, Reduce.substr(-6));
35.

```

Figura 48 – Implementazione reduce per selezione del mese di ogni anno con vendita minore

La funzione reduce andrà a lavorare in parallelo su ogni chiave, confrontando i valori che puntano alla stessa chiave, cui sono stati eliminati gli ultimi 6 caratteri (corrispondenti al mese di appartenenza), i quali verranno però selezionati per comporre i value della mappa `keyVal` fornita in output.

Verrà fornita in output la mappa `keyVal(K_In_Reduce, Reduce.substr(-6))`:

REDUCING
« KEY - VALUE »
« 2016 - 201608 »
« 2017 - 201708 »
« 2018 - 201808 »
« 2019 - 201902 »
« 2020 - 202003 »

Figura 49 – Output del paradigma MapReduce a seguito dell'operazione di reducing job4

I risultati ottenuti in [Figura 49] possono essere confrontati con quelli di [Figura 43] (colonne 2 e 3) per constatarne la corrispondenza:

Anno ▲	Mesi ▼	« KEY - VALUE »
2016	201608	
2017	201708	« 2016 - 201608 »
2018	201808	« 2017 - 201708 »
2019	201902	« 2018 - 201808 »
2020	202003	« 2019 - 201902 »
		« 2020 - 202003 »

Figura 50 – Confronto dei risultati job4 dell'implementazione in R e in MapReduce

Conclusioni

Sono state effettuate alcune analisi relative alle vendite di una azienda nel periodo che parte da gennaio 2016 ad agosto 2020.

Dai grafici sviluppati in R è possibile effettuare le seguenti considerazioni:

- Da [Figura 7] si evince che:
 - Le medie mensili sono per la maggior parte stabili su valori compresi tra 150 e 250.
 - Vi sono stati dei picchi nel mese di giugno 2017 e di agosto 2019
 - Agosto 2016 risulta essere il mese con media di vendita minore
 - Nel mese di aprile 2020 non si sono registrate vendite
- Da [Figura 16] si evince che:
 - I mesi in cui gli importi delle vendite sono stati pressoché costanti sono agosto 2016, 2017 e 2018, novembre 2017 ed i mesi tra gennaio e maggio del 2020.
 - I mesi in cui gli importi delle vendite sono stati molto distanti dal valore medio sono marzo 2016, novembre 2016, aprile 2017, dicembre 2017, aprile 2018 e agosto 2019, con due picchi nei mesi di giugno e ottobre 2017.
- Da [Figura 25] si evince che:
 - Il mese del periodo di riferimento in cui le vendite sono state massime è giugno 2017, risultato riscontrabile anche da [Figura 29].
 - Il mese del periodo di riferimento in cui le vendite sono state minime è agosto 2016, risultato riscontrabile anche da [Figura 44].

Sitografia

- (1). Tratto da <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9148156/>
- (2). Tratto da <https://lia.deis.unibo.it/Courses/som1516/materiale/mapreduceSOM.pdf>
- (5). Tratto da <https://economia.uniroma2.it/cdl/triennio/clef/corso/asset/YTo0OntzOjI6ImlkIjtzOjQ6IjEyMTkiO3M6MzoiaWRhIjtzOjU6IjQ3Mzc0IjtzOjI6ImVtIjtzOjM6MToiYyI7czo1OiJjZmNkMiI7fQ==>
- (6). Tratto da <http://www.edutecnica.it/calcolo/varianza/varianza.htm>
- (3). Tratto da <https://r-coder.com/aggregate-r/>
- (4). Tratto da <https://www.geeksforgeeks.org/r-bar-charts/>
- (7). Tratto da <https://r-coder.com/merge-r/>

Appendice

MapReduce_Job1.js

```
1. var S = ["|","|","|"];
2. // => WRITE YOUR CODE HERE <=
3.
4. function jobInputSplit(input_str){
5. //input_str=>document.getElementById('input_text_area').value;
6. return input_str.split('\n').filter(function (lines){
7. return lines.split(',')[0] === 'FATTURA' ||
   lines.split(',')[0] === 'RICEVUTA';
8. })
9. }
10.
11. function jobMap(V_In_Map){
12. return V_In_Map.map(function(item){
13. var ordini = item.split(",");
14. var key = ordini[1].substring(0,6);
15. var value = ordini[2];
16. return keyVal(key, value);
17. });
18. }
19.
20. function jobReduce(K_In_Reduce V_In_Reduce){
21. return K_In_Reduce_V_In_Reduce.map(function (items){
22.     var K_In_Reduce = items.split(S[0])[0];
23.     var V_In_Reduce = items.split(S[0])[1].split(S[1]);
24.
25.     var Reduce = V_In_Reduce.reduce(function(accumulator, item) {
26.     return parseFloat(accumulator) + parseFloat(item); });
27.
28.     var media = parseFloat(Reduce/V_In_Reduce.length);
29.
30.     return keyVal(K_In_Reduce, media.toFixed(5));
31. });
32. }
```

MapReduce_Job2.js

```
1. var S = ["|", ",", ""];
2. // => WRITE YOUR CODE HERE <=
3.
4. function jobInputSplit(input_str){
5. // input_str => document.getElementById('input_text_area').value;
6. return input_str.split('\n').filter(function (lines){
7. return lines.split(',')[0] === 'FATTURA' ||
   lines.split(',')[0] === 'RICEVUTA';
8. })
9. }
10.
11. function jobMap(V_In_Map){
12. return V_In_Map.map(function(item){
13. var ordini = item.split(",");
14. var key = ordini[1].substring(0,6);
15. var value = ordini[2];
16. return keyVal(key, value);
17. });
18. }
19.
20. function jobReduce(K_In_Reduce V_In_Reduce){
21. return K_In_Reduce_V_In_Reduce.map(function (items){
22. var K_In_Reduce = items.split(S[0])[0];
23. var V_In_Reduce = items.split(S[0])[1].split(S[1]);
24.
25. var totale_mesi = V_In_Reduce.reduce(function (accumulator, item){
26. return parseFloat(accumulator) + parseFloat(item);
27. });
28.
29. var media = parseFloat(totale_mesi/V_In_Reduce.length);
30.
31. var devianza_parziali = V_In_Reduce.map(function(item){
32. return parseFloat(Math.pow((item-media),2));});
33.
34. var devianza = devianza_parziali.reduce(function (accumulator, item){
35. return parseFloat(accumulator) + parseFloat(item);});
36.
37. var varianza = parseFloat(devianza/(V_In_Reduce.length - 1));
38.
39. return keyVal(K_In_Reduce, varianza.toFixed(3));
40. });
41. }
```

MapReduce_Job3_Totale_mesi.js

```
1. var S = ["|", ",", "|"];
2. // => WRITE YOUR CODE HERE <=
3.
4. function jobInputSplit(input_str){
5. // input_str => document.getElementById('input_text_area').value;
6. return input_str.split('\n').filter(function (lines){
7. return lines.split(',')[0] === 'FATTURA' ||
   lines.split(',')[0] === 'RICEVUTA';
8. })
9. }
10.
11. function jobMap(V_In_Map){
12. return V_In_Map.map(function(item){
13. var ordini = item.split(",");
14.
15. var key = ordini[1].substring(0,6);
16. var value = ordini[2];
17. return keyVal(key, value);
18. });
19. }
20.
21. function jobReduce(K_In_Reduce_V_In_Reduce){
22. return K_In_Reduce_V_In_Reduce.map(function (items){
23. var K_In_Reduce = items.split(S[0])[0];
24. var V_In_Reduce = items.split(S[0])[1].split(S[1]);
25.
26. var totale_mesi = V_In_Reduce.reduce(function (accumulator, item) {
27. return parseFloat(accumulator) + parseFloat(item);
28. });
29. return keyVal(K_In_Reduce, totale_mesi.toFixed(2));
30. });
31. }
```

MapReduce_Job3.js

```
1. var S = ["|", ",", ""];
2. // => WRITE YOUR CODE HERE <=
3.
4. function jobInputSplit(input_str){
5. // input_str => document.getElementById('input_text_area').value;
6. return input_str.split('\n')
7. }
8.
9. function jobMap(V_In_Map){
10. return V_In_Map.map(function(item){
11. var somma_mesi = item.split(" - ");
12.
13. var key = somma_mesi[0].substring(2,6);
14. var value = somma_mesi[1].slice(0, -2);
15.
16. return keyVal(key, somma_mesi[0].substring(2,8)+value);
17. });
18. }
19.
20. function jobReduce(K_In_Reduce_V_In_Reduce){
21. return K_In_Reduce_V_In_Reduce.map(function (items){
22. var K_In_Reduce = items.split(S[0])[0];
23. var V_In_Reduce = items.split(S[0])[1].split(S[1]);
24.
25. let n = 7;
26. var Reduce = V_In_Reduce.reduce (function (max, item){
27. return parseFloat(1*max.substring(n)) > parseFloat(1*item.substring(n))
28. ? max : item;
29. });
30. return keyVal( K_In_Reduce, Reduce.substring(0,7));
31. });
32. }
```

MapReduce_Job4_Totale_mesi.js

```
1. var S = ["|", "|", "|"];
2. // => WRITE YOUR CODE HERE <=
3.
4. function jobInputSplit(input_str){
5. // input_str => document.getElementById('input_text_area').value;
6. return input_str.split('\n').filter(function (lines){
7. return lines.split(',')[0] === 'FATTURA' ||
   lines.split(',')[0] === 'RICEVUTA';
8. })
9. }
10.
11. function jobMap(V_In_Map){
12. return V_In_Map.map(function(item){
13. var ordini = item.split(",");
14.
15. var key = ordini[1].substring(0,6);
16. var value = ordini[2];
17.
18. return keyVal(key, value);
19. });
20. }
21.
22. function jobReduce(K_In_Reduce_V_In_Reduce){
23. return K_In_Reduce_V_In_Reduce.map(function (items){
24. var K_In_Reduce = items.split(S[0])[0];
25. var V_In_Reduce = items.split(S[0])[1].split(S[1]);
26.
27. var totale_mesi = V_In_Reduce.reduce(function (accumulator, item) {
28. return parseFloat(accumulator) + parseFloat(item);
29. });
30. return keyVal(K_In_Reduce, totale_mesi.toFixed(0));
31. });
32. }
```

MapReduce_Job4.js

```
1. var S = ["|", ",", ""];
2. // => WRITE YOUR CODE HERE <=
3.
4. function jobInputSplit(input_str){
5. // input_str => document.getElementById('input_text_area').value;
6. return input_str.split('\n')
7. }
8.
9. function jobMap(V_In_Map){
10.   return V_In_Map.map(function(item){
11.     var somma_mesi = item.split(" - ");
12.
13.     var key = somma_mesi[0].substring(2,6);
14.     var value = somma_mesi[1].slice(0, -2);
15.
16.     return keyVal(key, value+somma_mesi[0].substring(2,8));
17.   });
18. }
19.
20. function jobReduce(K_In_Reduce_V_In_Reduce){
21.   return K_In_Reduce_V_In_Reduce.map(function (items){
22.     var K_In_Reduce = items.split(S[0])[0];
23.     var V_In_Reduce = items.split(S[0])[1].split(S[1]);
24.
25.     var Reduce = V_In_Reduce.reduce (function (max, item){
26.       return 1*max.slice(0, -6) < 1*item.slice(0, -6) ? max : item;
27.     });
28.     return keyVal( K_In_Reduce, Reduce.substr(-6));
29.   });
30. }
```

Contabilità_job_1_2_3_4.R

```
1. # Importo il file csv
2. ordini <- read.csv("/Users/giorgianesci/Documents/Uninettuno/Introduzione
   ai Big Data/Progetto Nesci/Ordini.csv", sep=";", header= FALSE)
3.
4. #Imposto l'intestazione per le colonne
5. colnames(ordini) <- c('Tipo', 'Data', 'Prezzo')
6.
7. #Conto la quantità di documenti per ogni tipologie
8. table(ordini$Tipo)
9.
10.
11. # Selezione degli oggetti di tipo "FATTURA" e "RICEVUTA"
12. ricavi <- subset(ordini, Tipo=="FATTURA" | Tipo=="RICEVUTA")
13.
14.
15. # Fatturato medio di ogni mese di ogni anno
16. media_mesi = aggregate(ricavi$Prezzo, by = list(substr(ricavi[,2], 1,
   6)), FUN=mean)
17.
18. # Rinomino intestazioni colonne media_mesi
19. colnames(media_mesi)[1] <- 'Mesi'
20. colnames(media_mesi)[2] <- 'Fatturato_medio'
21.
22. # File di output job1
23. write.table(media_mesi,
   "/Users/giorgianesci/Documents/Uninettuno/Introduzione ai Big Data/Progetto
   Nesci/Ordini_media_mesi.csv", row.names=FALSE, sep=";" )
24.
25.
26. # Grafico andamento guadagni in base alla media di ogni mese
27. barplot(height=media_mesi$Fatturato_medio, main = 'Media mensile di
   vendita', cex.names = 1, axis.lty=1, las=2,xlab = '', ylab = "Fatturato
   medio", names.arg=c("Gen2016","Feb2016","Mar2016","Apr2016","Mag2016",
   "Giu2016", "Lug2016", "Ago2016", "Set2016", "Ott2016", "Nov2016",
   "Dic2016", "Gen2017","Feb2017","Mar2017","Apr2017","Mag2017", "Giu2017",
   "Lug2017", "Ago2017", "Set2017", "Ott2017", "Nov2017",
   "Dic2017","Gen2018","Feb2018","Mar2018","Apr2018","Mag2018", "Giu2018",
   "Lug2018", "Ago2018", "Set2018", "Ott2018", "Nov2018", "Dic2018",
   "Gen2019","Feb2019","Mar2019","Apr2019","Mag2019", "Giu2019", "Lug2019",
   "Ago2019", "Set2019", "Ott2019", "Nov2019", "Dic2019",
   "Gen2020","Feb2020","Mar2020","Mag2020", "Giu2020", "Lug2020", "Ago2020"),
   col="#69b3a2")
28.
29.
30. # Varianza di vendita di ogni mese di ogni anno
31. varianza_mesi = aggregate(ricavi$Prezzo, by = list(substr(ricavi[,2], 1,
   6)), FUN=var)
32.
33. # Rinomino intestazione colonne varianza_mesi
34. colnames(varianza_mesi)[1] <- 'Mesi'
35. colnames(varianza_mesi)[2] <- 'Varianza'
36.
37. # File di output job2
38. write.table(varianza_mesi,"/Users/giorgianesci/Documents/Uninettuno/Intr
   oduzione ai Big Data/Progetto Nesci/Ordini_varianza_mesi.csv",
   row.names=FALSE, sep=";" )
39.
40. # Grafico varianza mesi
41. barplot(height=varianza_mesi$Varianza, main = 'Varianza mensile di
   vendita', cex.names = 1, axis.lty=1, las=2,xlab = '', ylab = "",
```

```

names.arg=c("Gen2016","Feb2016","Mar2016","Apr2016","Mag2016", "Giu2016",
"Lug2016", "Ago2016", "Set2016", "Ott2016", "Nov2016", "Dic2016",
"Gen2017","Feb2017","Mar2017","Apr2017","Mag2017", "Giu2017", "Lug2017",
"Ago2017", "Set2017", "Ott2017", "Nov2017",
"Dic2017","Gen2018","Feb2018","Mar2018","Apr2018","Mag2018", "Giu2018",
"Lug2018", "Ago2018", "Set2018", "Ott2018", "Nov2018", "Dic2018",
"Gen2019","Feb2019","Mar2019","Apr2019","Mag2019", "Giu2019", "Lug2019",
"Ago2019", "Set2019", "Ott2019", "Nov2019", "Dic2019",
"Gen2020","Feb2020","Mar2020","Mag2020", "Giu2020", "Lug2020", "Ago2020"),
col="#7FFFD4")
42.
43.
44. # Fatturato totale di ogni mese di ogni anno
45. somma_mesi = aggregate(ricavi$Prezzo, by = list(substr(ricavi[,2], 1,
6)), FUN=sum)
46.
47.
48. # Rinomino intestazione colonne somma_mesi
49. colnames(somma_mesi)[1] <- 'Mesi'
50. colnames(somma_mesi)[2] <- 'Fatturato_totale'
51.
52. # File Output somma_mesi
53. write.table(somma_mesi,"/Users/giorgianesci/Documents/Uninettuno/Introdu
zione ai Big Data/Progetto Nesci/Ordini_somma_mesi.csv", row.names=FALSE,
sep=";" )
54.
55.
56. # Andamento fatturato
57. barplot(height=somma_mesi$Fatturato_totale, main = 'Andamento fatturato
dal 01/2016 al 08/2020', cex.names = 1, axis.lty=1, las=2,xlab = '', ylab =
"", names.arg=c("Gen2016","Feb2016","Mar2016","Apr2016","Mag2016",
"Giu2016", "Lug2016", "Ago2016", "Set2016", "Ott2016", "Nov2016",
"Dic2016", "Gen2017","Feb2017","Mar2017","Apr2017","Mag2017", "Giu2017",
"Lug2017", "Ago2017", "Set2017", "Ott2017", "Nov2017",
"Dic2017","Gen2018","Feb2018","Mar2018","Apr2018","Mag2018", "Giu2018",
"Lug2018", "Ago2018", "Set2018", "Ott2018", "Nov2018", "Dic2018",
"Gen2019","Feb2019","Mar2019","Apr2019","Mag2019", "Giu2019", "Lug2019",
"Ago2019", "Set2019", "Ott2019", "Nov2019", "Dic2019",
"Gen2020","Feb2020","Mar2020","Mag2020", "Giu2020", "Lug2020", "Ago2020"),
col="#ADD8E6")
58. par(mar=c(5, 5, 5, 5))
59.
60.
61. # Importo mensile massimo di ogni anno
62. mesi_max_importoonly = aggregate(somma_mesi$Fatturato_totale, by =
list(substr(somma_mesi[,1], 1, 4)), FUN=max)
63.
64.
65. # Rinomino intestazione colonne mesi_max_importoonly
66. colnames(mesi_max_importoonly)[1] <- 'Anno'
67. colnames(mesi_max_importoonly)[2] <- "Fatturato_totale"
68.
69.
70. mesi_max = merge(mesi_max_importoonly, somma_mesi,
by="Fatturato_totale")
71.
72. # Rinomino la prima colonna di mesi_max
73. colnames(mesi_max)[1] <- 'Fatturato_mensile_max'
74.
75.
76. # Rappresentazione grafica "Fatturato mensile massimo per ogni anno"

```



```

77.  barplot (height = mesi_max$Fatturato_mensile_max, main = 'Fatturato
mensile massimo per ogni anno', xlab = 'Anni', ylab = 'Fatturato massimo',
names.arg=c("Giugno 2020","Aprile 2018","Ottobre 2019","Dicembre
2016","Giugno 2017"), col = '#834966')
78.  par(mar=c(5, 5, 5, 5))
79.
80.  # File di output job3
81.  write.table(mesi_max,"/Users/giorgianesci/Documents/Uninettuno/Introduzi
one ai Big Data/Progetto Nesci/Ordini_mesi_max.csv", row.names=FALSE,
sep=";" )
82.
83.
84.  # Importo mensile minimo di ogni anno
85.  mesi_min_importoonly = aggregate(somma_mesi$Fatturato_totale, by =
list(substr(somma_mesi[,1], 1, 4)), FUN=min)
86.
87.
88.  # Rinomino intestazione colonne mesi_min_importoonly
89.  colnames(mesi_min_importoonly)[1] <- 'Anno'
90.  colnames(mesi_min_importoonly)[2] <- "Fatturato_totale"
91.
92.
93.  mesi_min = merge(mesi_min_importoonly, somma_mesi, by =
'Fatturato_totale')
94.
95.  # Rinomino la prima colonna di mesi_min
96.  colnames(mesi_min)[1] <- 'Fatturato_mensile_min'
97.
98.  # Rappresentazione grafica "Fatturato mensile minimo per ogni anno"
99.  barplot (height = mesi_min$Fatturato_mensile_min, main = 'Fatturato
mensile minimo per ogni anno', xlab = 'Anni', ylab = 'Fatturato minimo',
names.arg=c("Agosto 2016","Agosto 2018","Marzo 2020","Agosto
2017","Febbraio 2019"), col = '#79fcca')
100. par(mar=c(5, 5, 5, 5))
101.
102.
103.
104.  # File di output job4
105.  write.table(mesi_min,"/Users/giorgianesci/Documents/Uninettuno/Introduzi
one ai Big Data/Progetto Nesci/Ordini_mesi_min.csv", row.names=FALSE,
sep=";" )

```