Università degli studi di Padova
Dipartimento di Scienze Statistiche

Corso di Laurea Triennale in
Statistica e Tecnologie Informatiche

Relazione Finale

Cluster Analysis, creazione di un'interfaccia intuitiva per le aziende d'oggi

Relatore Ombretta Gaggi

Laureando Matteo Aduso
Matricola N. 1007707

Anno Accademico 2014/2015

Indice

Capitolo 1

Introduzione

Oggetto di questa relazione è la partecipazione allo stage formativo all'interno della ditta Zucchetti S.P.A. (filiale di Padova) dove mi sono occupato di sviluppare un applicativo web intuitivo per l'analisi gerarchica di gruppi su una tabella di pivot.

1.1 Analisi delle competenze medie aziendali

Nelle aziende d'oggi si cerca di ricavare sempre maggiori informazioni dai dati che vengono accumulati nei vari comparti (contabilità, vendite, magazzino, etc.).
Molto spesso, specialmente in aziende di medie dimensioni, non si trattano piccole quantità di dati, bensì grosse moli di dati (*big-data*).
Tali analisi richiedono metodi ben delineati, conoscenze ma sopratutto affidabilità e significatività dei risultati ottenuti.
Per quanto riguarda le conoscenze e competenze, si è avuta l'occasione di partecipare a qualche riunione fra Zucchetti e alcuni operatori di *Business-Intelligence* (BI): tali operatori hanno il compito di installare e configurare opportunamente i prodotti Zucchetti per la BI presso aziende che li comprano ed in particolare, a seconda delle esigenze dell'azienda richiedente, di produrre grafici e tabelle di pivot riassuntive per alcuni indici (rotazione delle scorte, ritorno di investimento, ritorno delle vendite, etc.) e alcune analisi (analisi ABC, analisi di fatturato).
Durante queste riunioni sono emerse due cose molto importanti:

- *Competenze aziendali*: nelle aziende molto spesso non vi sono dipendenti capaci di comprendere le analisi create da tali operatori, probabilmente per mancanza di conoscenze specifiche; talvolta questi risultati vengono ritenuti errati in quanto non corrispondono al risultato atteso.

- *Conoscenze degli operatori*: Durante la riunione gli operatori dimostravano di essere competenti nell'uso e nel calcolo delle analisi richieste, senza essere a conoscenza dei fondamenti matematico-statistici delle analisi richieste, come per esempio la regola del 3σ per il Controllo qualità (usata per determinare limiti sulla qualità dei campioni utilizzati), applicata correttamente ma senza sapere le ipotesi di partenza e l'origine statistica.

É necessario precisare che tra i due punti esposti precedentemente, il primo risulta importante e rappresenta un punto cruciale: senza le competenze necessarie un'azienda potrà cercare di riassumere ed analizzare big-data più volte ma non sarà mai capace di riadattare tali analisi alla precisa situazione o ricavare informazioni corrette.

1.2 Zucchetti S.P.A.

Zucchetti è una società per azioni con oltre 2700 dipendenti operante nel settore dell'Information and Comunication Tecnology (ICT). Si occupa di creare e fornire soluzioni software e hardware studiate per aziende operanti su determinati settori quali:

- aziende di qualsiasi settore e dimensione, come banche (Banca popolare di Milano, Fininvest), aziende commerciali (Pompea, Riso Scotti, The Space Cinema) e aziende automobilistiche (Toyota, Automobili Lamborghini);

- professionisti (commercialisti, consulenti del lavoro, avvocati, notai), associazioni di categoria e CAF;

- pubblica amministrazione (comuni, province, regioni, ministeri, società pubbliche, ecc).

fornendo inoltre supporto pre-vendita, assistenza post-vendita, formazione per sfruttare al meglio i prodotti acquistati e aggiornamenti su di essi.
Alcune delle soluzioni create sono:

- Gestione contabile e fiscale: Contabilità e bilancio, dichiarazioni, clienti, fatturazioni
- Gestionali di Enterprise Resource Planning (ERP), per settori quali turismo, moda, sport.
- Gestione del personale (paghe, presenze, note spese, budget del personale, gestione turni, compilazione 730).
- Gestione documentazioni, archiviazione e conservazione sostitutiva.
- Gestionali di Costumer Relationship Management (CRM) pre-vendita e post-vendita.
- Gestione tesoreria, del credito e Business Intelligence (BI).
- Sicurezza: gestione della sicurezza sul lavoro, elaborazione Documento di Valutazione dei Rischi (DVR) e corsi di formazione.
- Strumenti di sviluppo per la creazione di soluzioni desktop e mobile.
- Altre soluzioni quali note spese, firma elettronica, Software per l'agenzia delle entrate, gestione dei fallimenti, trattamento dati personali, software di consulenza legale e studi legali, software per i CAF.

1.3 Scopo del progetto

Nella realtà italiana, come approfondito nella sezione 4.1, vengono usate massivamente tabelle di pivot per riassumere dati ed anche per analizzarli.
Si sottolinea come sia molto difficile ricercare precise anomalie fra righe o colonne, oppure determinati comportamenti comuni, specialmente se la tabella di pivot rappresenta big-data e quindi risulta di notevoli dimensioni (intorno alle mille righe per una cinquantina di colonne o più).
Lo scopo del progetto inizialmente era evidenziare quelle righe che, rispetto alle altre, avevano un comportamento o andamento anomalo rispetto alle altre, una sorta di ricerca di valori anomali, ma esteso a tutta la riga.
Tale tipo di analisi, basata su test statistici di Goodness of Fit (bontà di adattamento) è risultata fallimentare, come spiegato in dettaglio nel capitolo 4 ed ha comportato un evoluzione del progetto: la ricerca di possibili gruppi, sempre tra righe, con comportamento o andamento simile.

Per raggiungere tale scopo si è optato per un approccio Machine-Learning, in particolare di classificazione gerarchica, come spiegato nell'analisi dei requisiti (Capitolo 2) e descritto in maniera approfondita nel capitolo riguardante l'implementazione del progetto (Capitolo 5).

Capitolo 2

Analisi dei requisiti

Come spiegato nella sezione 1.1, spesso le aziende non possiedono personale competente, conoscenze di metodi statistici o di Machine-Learning; è importante quindi effettuare una adeguata analisi dei requisiti descritta in questo capitolo.
La creazione di un'interfaccia grafica e di un metodo di analisi semplice e di facile comprensione, rivolto ad utenti interessati solo al risultato ed usabile senza presupposti di alcuna conoscenza scientifica, rappresenta uno dei punti cardini del progetto.
Durante la prima versione del progetto, dove si prevedeva l'utilizzo di test statistici per identificare righe anomale, non era prevista una vera e propria interfaccia grafica, in quanto doveva essere uno strumento integrato nella visualizzazione di tabelle. Doveva però essere presente una guida che spiegasse all'utente come interpretare le evidenziazioni di riga e dare la possibilità di disabilitarlo.
Per quanto riguarda la seconda versione, invece, risulta fondamentale non solo la creazione di un interfaccia grafica semplice, come detto precedentemente, ma anche la realizzazione di un grafico che sia il più rappresentativo possibile, cioè che sia di facile comprensione, ma anche che dia più informazioni possibili sull'analisi che si sta conducendo.
A tal proposito sono state pensati due diverse tipologie di grafico (illustrate a fine capitolo):

- Force-Based (fig. 2.1): basato sulla sola matrice delle distanze (capitolo 5.3), ogni "riga" è collegata con un altra in base ad un limite numerico, questo limite è variabile e viene scelto dall'utente.

- Cluster-Based (fig. 2.2): basato sulla Herarchical-Clustering, raggruppa le "righe" appartenenti ad uno stesso gruppo, l'utente però deve scegliere un punto di taglio per questi raggruppamenti.

Tutte e due le tipologie di grafico devono quindi cambiare a seconda del limite o punto di taglio scelto dall'utente; inoltre il metodo per scegliere questo valore deve essere semplice. Per raggiungere tale scopo è stata introdotta una barra di scorrimento, detta "barra di Threshold". Oltre alla barra di Threshold, è stata creata un'interfaccia grafica il più minimale possibile, negando quindi la possibilità di scegliere alcuni parametri della cluster analysis, quali tipo di distanza e metodo di raggruppamento, scelti dal sistema in automatico.
Questo approccio non dà la possibilità di personalizzare questa analisi a seconda delle esigenze, ma consente ad un utente qualunque di utilizzare tale analisi senza nessun tipo di conoscenza tecnico-scientifica.
Un altro importante requisito del progetto è la generalizzazione del metodo di analisi: tale metodo dovrebbe poter essere applicato a qualsiasi "insieme" di dati, quantitativo o qualitativo, assicurando all'utente di effettuare qualsiasi analisi esso ritenga opportuno.
Nei prossimi capitoli si illustrano le tecnologie adoperate per sviluppare tale progetto e lo sviluppo del progetto in tutte le sue fasi: la prima versione del progetto (Capitolo 4) e l'evoluzione dello stesso (Capitolo 5).

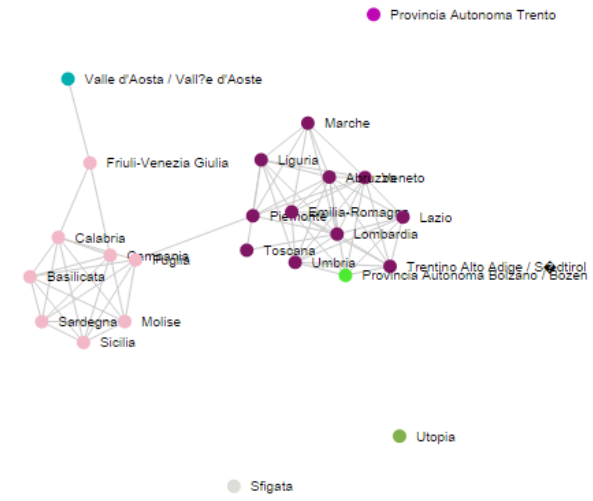

Figura 2.1: Grafico Force-Based

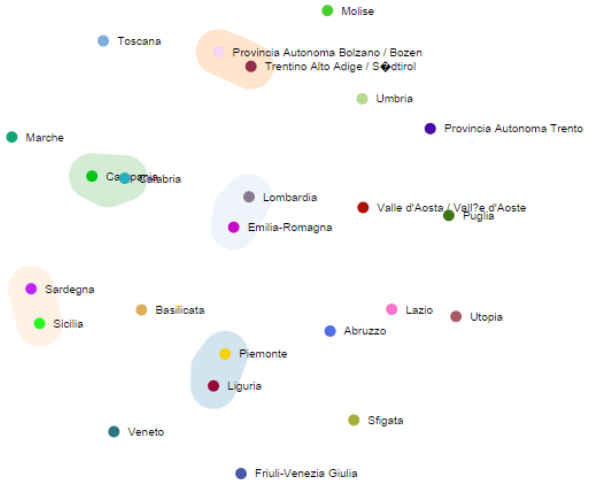

Figura 2.2: Grafico Cluster-Based

Capitolo 3

Tecnologie adoperate

In questo capitolo vengono illustrate (ed eventualmente spiegate) le tecnologie impiegate per la creazione del progetto, motivandone l'adozione rispetto a possibili alternative disponibili sul mercato.
Si sottolinea in particolar modo l'impiego della libreria D3 (sezione 3.3), in quanto essenziale per il progetto.

3.1 HTML5 & CSS3

HTML5 rappresenta l'evoluzione del mondo web con l'obbiettivo di favorire lo sviluppo di applicativi web; vuole estendere lo scopo delle pagine web, che non siano più semplici documenti, articoli o siti in cui si possono reperire informazioni, ma vere e proprie applicazioni utilizzabili dall'utente all'interno di un browser Web.
Viene considerata una tecnologia utilizzata nel progetto in quanto esso è liberamente ispirato a questa nuova concezione di web.
Questa concezione di web viene correntemente applicata ai prodotti moderni della Zucchetti, come il progetto Infinity Analytics (un progetto di BI).
Ai fini della realizzazione del progetto, si potevano usare standard HTML meno recenti come XHTML 1.0 in quanto all'interno delle pagine HTML del progetto non vengono usati tag di HTML5. Viene comunque descritto HTML5 in quanto adoperato dalla Zucchetti su Infinity Analytics, pensando sopratutto ad una possibile integrazione di questo progetto di stage nel prodotto Zucchetti.
CSS invece viene usato per definire lo stile delle pagine HTML del progetto,

come già avviene nelle pagine web che si possono trovare nel web.
É stata adottata la versione 3 di CSS solo per ottenere bordi arrotondati nel riquadro che compare sopra i cerchi all'interno del grafico. Rappresenta comunque un adeguamento delle tecnologie verso la Zucchetti, che lo usa per i nuovi selettori, le animazioni e le media query.
Nelle animazioni delle pagine, però, CSS viene usato solo in parte, in quanto i movimenti di alcune parti della pagina sono realizzati usando SVG, mentre dissolvenza, movimenti di box, variazioni di colore ed altre cose sono ottenute cambiando le proprietà di stile degli elementi interessati usando JavaScript, quindi agendo sugli stili senza usare transizioni in CSS3.

3.2 JavaScript

JavaScript è un linguaggio di scripting dinamico, comunemente usato come parte di pagine web per renderle dinamiche favorendo una maggior interazione con l'utente, ma usato anche lato Server usando Node.js e per creare applicazioni desktop e mobile. Viene definito dinamico in quanto la definizione dei tipi di dato è basata su prototipi, riuscendo quindi a supportare diversi stili di programmazione tra cui il paradigma imperativo e Object-Oriented(OOP).
In particolare, per quanto riguarda il paradigma Object-Oriented, esso non è come Java o C++; infatti questi due linguaggi di programmazione sono OOP class-based, cioè basati sul concetto di classe, dove gli oggetti sono un istanza di una classe. In JavaScript, invece, non esiste una distinzione tra classi e oggetti, ma esiste il concetto di prototipo, cioè un oggetto che funge da modello per altri oggetti; questo tipo di paradigma, OOP prototype-based, viene anche utilizzato in linguaggi di scripting come Ruby e Lua, ma anche in Perl, Python e R tramite l'utilizzo di opportune librerie.
JavaScript è stato formalizzato nello standard di linguaggio ECMAScript dall'ente European Computer Manufacturers Association (ECMA). Questa organizzazione si occupa di creare standard nell'ambito dell'Information and Comunication Tecnology (ICT), tra i più importanti standard creati si menzionano quelli per i linguaggi C++, C#, CLI, JSON.
JavaScript inoltre ricopre un ruolo fondamentale nella tecnica di sviluppo "Asynchronous JavaScipt And XML" (AJAX) permettendo una continua comunicazione tra server e client anche dopo aver caricato una pagina web

sfruttando alcuni oggetti presenti in JavaScript. Questa tecnica di sviluppo viene correntemente usata dalla Zucchetti nei suoi prodotti.
All'interno del progetto questa tecnica viene utilizzata per il caricamento di alcuni dataset di esempio, cioè file csv presenti nel server e scaricati una volta selezionati dall'utente; pensando però ad una futura integrazione nel progetto Nexus, esso farà parte di una applicazione che usa come tecnica di sviluppo AJAX.
Come verrà meglio approfondito nel capitolo 5, il progetto è stato strutturato a oggetti, favorendo cosi una programmazione ben strutturata, facilitandone la gestione ed eventualmente favorendo il riutilizzo del codice usando i prototipi scritti.

3.3 D3.js

D3 è una libreria, sviluppata in JavaScript, che consente la manipolazione di pagine HTML ed elementi Scalable Vector Graphics (SVG) basandosi sui dati; quindi dà la possibilità di creare qualsiasi tipo di grafico e, tramite l'uso di transizioni, permette di creare animazioni efficaci ed accattivanti.
I grafici e le transizioni vengono ottenuti manipolando immagini vettoriali, principalmente gestite da D3, facilitando quindi il compito del programmatore.
Non viene quindi usato un elemento canvas, adottato dallo standard HTML5, ma viene usato un elemento SVG.
SVG descrive le immagini attraverso un insieme di vettori anziché con il metodo raster (descrizione di un immagine attraverso una griglia di pixel). Questo permette di ridimensionare queste immagini senza perdita di qualità, contrariamente al metodo raster dove un ingrandimento può rendere l'immagine sgranata per mancanza di qualità.
La descrizione di immagini vettoriali avviene attraverso l'uso di un metalinguaggio derivante da XML e attualmente facente parte delle raccomandazioni standard del consorzio W3C; questo linguaggio descrive i seguenti elementi grafici:

- forme geometriche, definite come insieme di linee e curve (cerchio, rettangolo, ellisse, linea, poligono o percorsi);
- testi, definite come insieme di glifi;

- immagini raster

A disposizione del programmatore c'è inoltre lo standard SVG Document Object Model (DOM), che consente di accedere agli "oggetti" dell'immagine vettoriale tramite linguaggi di scripting ECMAScript come JavaScript e ActionScript. Viene data anche la possibilità di assegnare event handler agli oggetti grafici facente parte dell'immagine vettoriale descritta.
D3 sfrutta quindi SVG per la creazione di grafici, e usa l'SVG DOM per accedere agli elementi SVG per modificarne le proprietà e creare animazioni o cambiamenti di stili usando CSS.
Esempi di grafici semplici ottenibili utilizzando D3 sono:

- Grafico a Barre, Istogramma
- Grafico ad area, Grafico a torta
- BoxPlot, Scatterplot e BubbleChart
- Alberi
- Grafici dinamici basati sulla forza di gravità
- Mappe

Altri tipi di grafici, meno comuni, sono Parallel Coordinates, Diagramma di Venn, etc.
Le tipologie sopra definite non sono le uniche disponibili: la libreria si presta anche a nuove tipologie di grafici.
Per quanto riguarda il progetto, i grafici che sono stati creati sono basati sulla forza di gravità, come spiegato in dettaglio nella sezione 6: in questo modo si assicura un grafico piacevole e altamente fruibile, quindi di immediata comprensione per l'utente finale.

Capitolo 4

Studio e prove di alcuni test di Goodness of Fit

In questo capitolo verrà spiegato come, nel mondo del lavoro, le tabelle di Pivot, siano quasi essenziali per rilevare informazioni strategiche nonostante spesso vengano usate in maniera impropria; inoltre si approfondirà la prima parte del progetto (la ricerca di righe "fuori controllo" rispetto alle altre) di una tabella di pivot. A tal proposito verranno approfonditi i test statistici presi in esame per la realizzazione di tale scopo.

4.1 L'importanza delle tabelle di Pivot

Durante il periodo di Stage svolto in Zuchetti SPA sono emerse importanti questioni riguardanti l'uso delle tabelle di Pivot.
Una tabella di Pivot è uno strumento di reporting usato in programmi come i fogli elettronici o di Business Intelligence; con questo strumento si riesce a descrivere un numero elevato di variabili in una tabella a doppia entrata, inserendo anche risultati di una qualche funzione.
Una tabella di pivot di solito è costituita da un area per le righe, per le colonne e per i dati; l'utente può quindi trascinare una variabile in una delle tre aree ed ottenere una tabella a doppia entrata. Infine si può scegliere di filtrare le variabili secondo un qualche tipo di criterio e di applicare una funzione riassuntiva ai margini della tabella quale conteggi, somma, media ed altro.

Data	Nome	Prodotto	Tipo	Quantità	Totale
11/11/2014	Foo	Portatile	Bianco	1	100
12/11/2014	Bar	Mouse	Nero	2	50
13/11/2014	Foobar	PC	Bianco	1	200
14/11/2014	Foo	Mouse	Indaco	3	75
15/11/2014	Foo	Portatile	Bianco	2	200
16/11/2014	Foobar	Portatile	Nero	2	200
17/11/2014	Bar	Mouse	Viola	4	100

Tabella 4.1: Dataset di esempio

Ad esempio, avendo un dataset come quello in tabella 4.1 si può ottenere una tabella di pivot che riassuma la quantità di prodotti comprati da ciascun cliente nel seguente modo:

Somma di Quantità	Etichette di colonna			
Etichette di riga	Mouse	PC	Portatile	Totale complessivo
Bar	6			6
Foo	3		3	6
Foobar		1	2	3
Totale complessivo	**9**	**1**	**5**	**15**

Figura 4.1: Esempio di una tabella di Pivot riferita al dataset di esempio in tabella 4.1

Secondo gli operatori della Zucchetti, a diretto contatto con le aziende, una tabella di pivot solitamente viene usata per ricercare informazioni di qualsiasi tipo nonostante la dimensione della tabella sia troppo grande (dell'ordine delle migliaia di righe) e quindi non consenta una buona visione d'insieme dei dati da analizzare.
In particolare si è puntualizzato sul comportamento dell'utente: in questa situazione egli comincia a scorrere la tabella, anche velocemente, alla ricerca di un valore grande (o piccolo) che possa significativamente impressionarlo.
Questa modalità operativa, lunga e poco produttiva, può essere sostituita con l'uso di funzioni apposite messe a disposizione dai software sviluppati da Zucchetti.
Inoltre, domanda fatta molto spesso durante la presentazione di una tabelle pivot nei gestionali Zucchetti, vi è la possibilità di esportare il contenuto in un documento Excel.
Tale richiesta è probabilmente dovuta alla mancata possibilità di cambiare il

contenuto di singole celle nel gestionale Zucchetti; questo perché i risultati delle analisi fatte probabilmente non sono ritenuti soddisfacenti o non piacciono.
I software di Business Intelligence sviluppati da Zucchetti SPA hanno l'obbligo di certificare i dati processati oltre che di fornire ottime analisi utili per far crescere un'azienda; consentire il cambiamento di alcuni valori all'interno di una tabella renderebbe inaffidabile la certificazione e la vera origine del dato.

4.2 Test statistici per la segnalazione di dissimilarità

Lo scopo del progetto è quello di evidenziare all'interno di una tabella di pivot le righe apparentemente anomale, con un comportamento cioè differente rispetto alle altre.
Con diversità di comportamento si intende l'andamento della riga, il range di valori, media e varianza, vale a dire il fatto che le righe provengano dalla stessa popolazione e siano realizzazione di una stessa *variabile casuale* (v.c.).
Per evidenziare righe anomale si sono analizzati alcuni test statistici sulla bontà di adattamento (Goodness of Fit); questi test si basano o sul confronto tra due sample (campioni) di dati o sul confronto tra un sample e un modello probabilistico, a seconda del tipo di test usato.
I test presi in esame sono:

- Pearson's Chi-Square
- Kolmogorov-Smirnov
- Cramer-Von-Mises
- Wilcoxon-Mann-Whitney

Sono tutti test non parametrici, quindi permettono di non fare ipotesi sul tipo di distribuzione delle righe mantenendo un certo grado di generalizzazione.
L'ipotesi da verificare tramite l'utilizzo di questi test è formalizzata come segue:

$$\begin{aligned} H_0 &: F_0(x) = F_1(x) \\ H_1 &: F_0(x) \neq F_1(x) \end{aligned} \tag{4.1}$$

dove $F_i(x)$ rappresenta la funzione di ripartizione empirica dell'i-esimo sample appartenente alla tabella.
Gli assunti dei test verranno trattati nella sezione seguente.

4.2.1 Limitazioni comportate dai test utilizzati

I test utilizzati, come già accennato, permettono di non fare ipotesi sulla distribuzione di provenienza dei campioni; il loro utilizzo però comporta delle limitazioni.
Una prima limitazione riguarda l'andamento dei campioni: utilizzando tali test non è possibile verificare la somiglianza dell'andamento dei campioni in esame. Infatti permettono solo di verificare la distribuzione di provenienza, nel capitolo 5 verrà spiegato come la seconda parte del progetto sia più flessibile al tipo di ricerca richiesta.
Un'altra considerazione importante, oltre che notevolmente limitativa, riguarda le assunzioni dei test di Goodness of Fit, riportate in tabella 4.2 e spiegate sotto:

Assunzione\Test	Chi-Quadrato	Kolmogorov	Wilcoxon	Cramer
Indipendenza	Si	Si	Si	Si
Campionamento C.S.	Si	Si	Si	Si
Continuità	No	Si	Si	Si
Dati ordinali	No	Si	Si	Si
Frequenze	Si	No	No	No

Tabella 4.2: Assunzioni dei test esaminati

- Indipendenza: che vi sia indipendenza tra le osservazioni di un singolo campione;
- Campionamento casuale semplice: i campioni sono estrazioni casuali della variabile aleatoria d'interesse;
- Continuità: si assume che la distribuzione del campione, seppur incognita, sia continua;
- Dati ordinali: presuppone che i dati si possano ordinare, in particolare che prese due osservazioni si riesca sempre a definirne una più grande rispetto ad un'altra;

- Frequenze: che le unità dei campioni siano numeri naturali, rappresentanti una frequenza;

Tali assunti rappresentano un'ulteriore perdita di generalità per il progetto che dovrebbe mantenere un livello di generalità molto alto.
In particolare l'assunzione di indipendenza non si può dire soddisfatta nella maggior parte dei casi, un esempio è dato da una serie storica: è molto probabile che l'andamento di un'annata sia influenzata dalle annate precedenti.
Inoltre anche il campionamento casuale semplice non si può definire sempre soddisfatto in quanto non si conosce il metodo di estrazione dei campioni.
Altre assunzioni limitative verranno trattate durante la spiegazione dei singoli test.
Un altro aspetto importante riguarda la scala dei dati: in presenza di scale differenti non si può normalizzare i dati; un approccio del genere porterebbe ad una perdita di informazione in termini di media e varianza.
Ad esempio avendo due campioni casuali semplici realizzazione di due variabili aleatorie Normali ma con media e varianza diversi, una normalizzazione di queste due variabili porterebbe ad avere media nulla e varianza unitaria; questo risultato porta quindi ad avere due popolazioni di provenienza identiche quando invece le due popolazioni di origine non lo sono.
Di seguito si presentano i test statistici analizzati durante lo stage.

4.2.2 Pearson's Chi-Square

Il test chi-quadrato di Pearson ipotizza che i campioni sia realizzazione di una v.c. $Bin(\pi, n)$, quindi che ogni valore rappresenti il conteggio dei successi di una determinata classe.
La sua statistica test si calcola:

$$X^2 = \sum_{i=1}^{k} \frac{(n_i - E_i)^2}{E_i} = \sum_{i=1}^{k} \frac{n_i^2}{E_i} - n \tag{4.2}$$

dove

n_i è la numerosità dell'i-esimo campione

E_i è il numero di casi attesi nel caso l'ipotesi nulla fosse vera

k è il numero di modalità nella quale si esprime la variabile nominale

La statistica test $X^2 \sim \chi^2_{k-1}$ è quindi confrontabile con una v.c. χ^2 con $k-1$ gradi di libertà.
Il test risulta però molto restrittivo in quanto si applica solo per valori interi,

rappresentanti la frequenza di una determinata classe, ad esempio il numero di persone disoccupate in un determinato anno.
Inoltre il confronto avviene tra un campione e una legge di probabilità discreta, quindi nelle simulazioni fatte in seguito verrà stimata una legge di probabilità basata sull'altro campione in esame.

4.2.3 Kolmogorov-Smirnov

Il test di Kolmogorov-Smirnov verifica la forma delle distribuzioni campionarie tra due campioni; prevede che i dati siano almeno ordinali, quindi risulta meno restrittivo rispetto al test Chi-Quadrato di Pearson.
La sua statistica test è calcolata mediante:

$$D_n = \sup_{-\infty<x<+\infty} \left| \hat{F}_n(x) - F_0(x) \right| \tag{4.3}$$

con $\hat{F}_n(x)$ funzione di ripartizione empirica calcolata nel seguente modo:

$$\hat{F}_n(x) = \frac{1}{n} \sum_{i=1}^{n} I_{X(i)\leq x} \tag{4.4}$$

Quindi la statistica test D_n trova la differenza maggiore tra le due funzioni di ripartizione empiriche (fig. 4.2) e viene confrontata con la variabile casuale test di Kolmogorov-Smirnov.
Questa variabile casuale è difficile da calcolare, e viene di solito approssimata per m e n sufficientemente grandi:

$$X = 4 \left(\frac{mn}{m+n} \right) D_{m,n}^2 \sim \chi_2^2 \tag{4.5}$$

Se m e n non sono abbastanza grandi, di solito si fa affidamento sulle tavole di significatività, questo perché risulta complesso calcolare il p-value, specialmente quello esatto.
Questa complessità deriva dalla funzione di probabilità della variabile casuale di Kolmogorov-Smirnov:

$$Pr(K \leq x) = 1 - 2 \sum_{k=1}^{\infty} (-1)^{k-1} e^{-2k^2x^2} = \frac{\sqrt{2\pi}}{x} \sum_{k=1}^{\infty} e^{-(2k-1)^2\pi^2/(8x^2)} \tag{4.6}$$

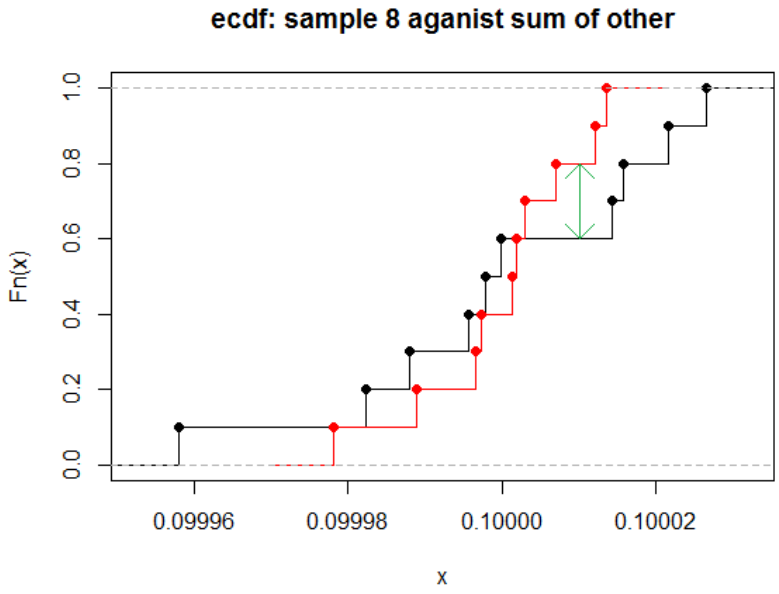

Figura 4.2: Due funzioni di ripartizione messe a confronto

dove K è la variabile casuale $K = \sup_{t\in[0,1]} |B(F(t))|$ e $B(t)$ è la funzione di Brownian Bridge.
Questa variabile è in relazione con la statistica test 4.3, sotto ipotesi nulla, dal seguente limite:

$$\sqrt{n}D_n \overset{n\to\infty}{\to} \sup_t |B(F(t))| \tag{4.7}$$

Per esempio, se si vuole trovare il valore critico della statistica test al 95%, il risultato si otterrà risolvendo la seguente equazione:

$$Pr(K \leq K_\alpha) = 1 - \alpha \tag{4.8}$$

4.2.4 Wilcoxon-Mann-Whitney

Viene anche chiamato Mann-Whitney U test, esso si applica a dati almeno ordinali e l'ipotesi alternativa in realtà è che tra le due popolazioni, una delle due tenda stocasticamente ad avere valori più grandi dell'altra, correggendo così l'ipotesi alternativa del test 4.1:

$$H_1 : F_0(x) < F_1(x)$$

Questo particolare test viene definito l'alternativa non parametrica del t-test. Il calcolo della statistica test U viene calcolato mediante il seguente algoritmo:

1. si dispongono le osservazioni dei due campioni in una singola serie e vi si assegna un rango a ciascuna.

2. si sommano i ranghi per il primo campione, s_1, e per il secondo campione, s_2.

Dopo aver calcolato s_1 e s_2, si procede con il calcolo della statistica test U:

$$U_1 = n_1 n_2 + \frac{n_1(n_1+1)}{2} - R_1 \tag{4.9}$$

$$U_2 = n_1 n_2 + \frac{n_2(n_2+1)}{2} - R_2 \tag{4.10}$$

Viene poi confrontato con le tavole di significatività il più piccolo tra U_1 e U_2, determinando così il risultato del test.

4.2.5 Cramer-Von-Mises

Questo test rappresenta una valida alternativa al test di Kolmogorov-Smirnov e la sua statistica test, per il caso a due campioni, è calcolata nel seguente modo:

$$T = N\omega^2 = \frac{U}{NM(N+M)} - \frac{4MN-1}{6(M+N)} \tag{4.11}$$

dove U e ω^2 è definito nel seguente modo

$$U = N\sum_{i=1}^{N}(r_i - i)^2 + M\sum_{j=1}^{M}(s_j - j)^2 \tag{4.12}$$

$$\omega^2 = \int_{-\infty}^{+\infty} \left(F_n(x) - F^*(x)\right)^2 dF^*(x) \tag{4.13}$$

ed r_i e s_j rappresentano rispettivamente i ranghi per il primo campione (x) e il secondo campione (y), mentre N e M sono le numerosità campionarie dei due campioni.
I ranghi invece vengono assegnati ordinando il campione e assegnandovi rango i, dove i rappresenta la posizione del dato nella serie ordinata.

Con questo metodo si assume che ogni campione sia ordinabile e che ogni osservazione all'interno del proprio campione sia unica, come il suo rango. Se così non fosse, si può applicare un metodo, detto di *midrank*, che prevede l'assegnazione ad ogni valore duplicato di un rango uguale a $(i + j)/2$.

4.3 Risultati dei Test

I test statistici presentati nella precedente sezione sono stati analizzati e provati su **R** nelle due seguenti modalità:

1. Metodo 1: Ogni campione confrontato con la somma di tutti gli altri campioni, riscalato opportunamente in base al campione analizzato (vedi formula 4.14).

2. Metodo 2: Ogni campione confrontato con ogni altro campione, e raccolto il numero totale di casi in cui il test rifiuta l'ipotesi nulla.

Per automatizzare le analisi sono state implementate delle funzioni che, dato un campione di dati in ingresso, fornisce il risultato per ogni test sopra presentato; le funzioni sono: sumTest per il primo metodo, multiTest e simMultiTest per il secondo.
Un utile osservazione su questi metodi è la complessità computazionale: nel primo caso si ha una complessità pari a $O(nC_i)$ dove C_i è la complessità del test usato, mentre nel secondo caso la complessità è pari a $O(C_i * (n + (n-1) + (n-2) + \ldots + 1)) \sim O(n^2 * C_i)$, quindi da una complessità lineare a una quadratica.
I dati presi in analisi sono di due tipologie:

- Dati fittizi: sono dati generati da una Poisson di tasso $\lambda = 5000$, a cui sono stati aggiunti dei campioni provenienti da una Poisson di tasso $\lambda = 10000$.

- Dati reali: sono stati presi dall'**ISTAT** e rappresentano i dati sulla disoccupazione per regione dagli anni 2000 al 2013, a cui sono stati aggiunti dei campioni provenienti da regioni fittizie di nome Utopia e Sfortunata, la prima con un andamento crescente dato dalla formula $x_j = x_{j-1} * 2$ a partire da un valore di 353322, la seconda con un aumento dato dalla formula di Utopia fino al 2005 e una diminuzione data dalla formula $x_j = x_{j-1}/2$ dopo il 2005.

In figura 4.3 è possibile apprezzare un grafico dei dati fittizi, mentre in figura 4.4 è possibile vedere l'andamento dei dati reali analizzati in seguito.
Segue quindi l'analisi dei due metodi: per ognuno di essi verrà fornito il risultato con i due dataset presi in esame, fornendone un'interpretazione.

4.3.1 Primo Metodo

Per ogni test viene mostrato se si accetta l'ipotesi H_0 (valore 0), o se si rifiuta l'ipotesi H_0 a favore dell'ipotesi H_1(valore 1).
La "scalatura" della somma viene effettuata nel seguente modo:

$$x_{\cdot k} = \frac{\sum_{h=0}^{n,h\neq i} x_{hk}}{\sum_{h=0}^{n}\sum_{q=0}^{m} x_{hq}} \sum_{h=0}^{m} x_{ih} \tag{4.14}$$

con n, m rispettivamente righe e colonne della tabella analizzata, i indice della riga da esaminare, k la colonna che sto calcolando.
Risultati dei dati fittizi analizzati usando il primo metodo:

```
> sumTest(sim, figName="sumTestSim.png", colGrap=3, rowGrap=4)

[1] "Pearson's Chi-Square Test results: "
 [1] 0 0 0 0 0 0 0 0 0 0 0 0
[1] "Kolmogorov-Smirnov's Test results: "
 [1] 0 0 0 0 0 1 0 0 0 1 0 1
[1] "Wilcoxon-Mann-Whitney (or Mann-Whitney U) Test results: "
 [1] 0 0 0 0 0 0 0 0 0 0 0 0
[1] "Cramer-Von-Mises's Test results: "
 [1] 1 0 0 0 0 1 0 0 0 1 0 1
```

Il test chi-quadrato di Pearson e il test di Wilcoxon accettano sempre, quindi senza identificare nessuna "riga" fuori controllo. Mentre i test di Kolmogorov-Smirnov e Cramer-Von-Mises hanno gli stessi risultati, cioè rifiutano l'ipotesi nulla per la riga 4, 10 e 12 (eccetto Cramer-Von-Mises che identifica come fuori controllo anche il primo campione).
Si potrebbe quindi dire che il test di Cramer-Von-Mises stia funzionando, a meno di qualche errore (in questo caso la prima riga).
In figura 4.5 si possono osservare le funzioni di ripartizione dei campioni in esame (in nero) e la funzione di ripartizione di tutti gli altri campioni (in rosso).
Risultati dei dati reali analizzati usando il primo metodo:

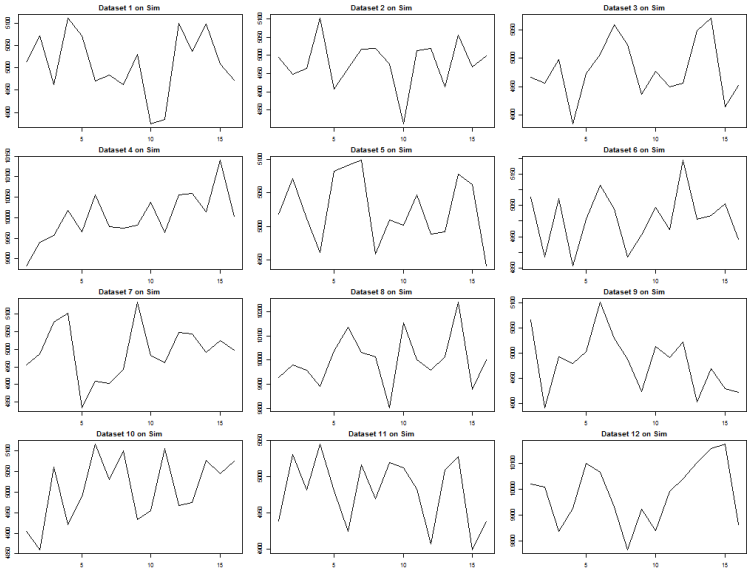

Figura 4.3: Dati fittizi generati in **R**

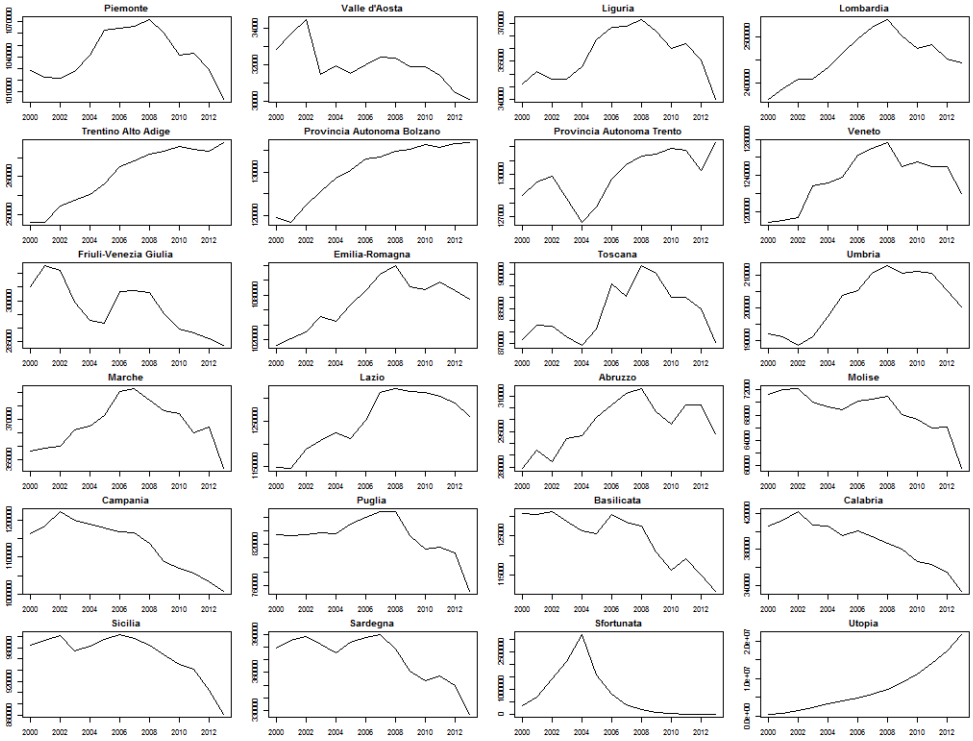

Figura 4.4: Dati sull'occupazione per regione

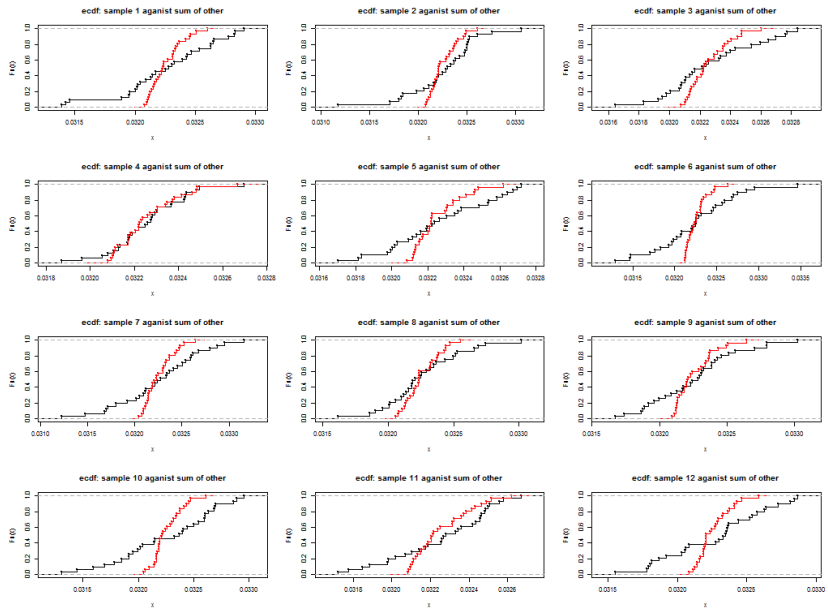

Figura 4.5: funzione di ripartizione empirica del campione in analisi (colore nero) contro la funzione di ripartizione empirica della somma degli altri campioni (rosso)

```
> sumTest(occ, figName="sumTestOcc.png", colGrap=4, rowGrap=6)

[1] "Pearson's Chi-Square Test results: "
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[1] "Kolmogorov-Smirnov's Test results: "
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[1] "Wilcoxon-Mann-Whitney (or Mann-Whitney U) Test results: "
 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[1] "Cramer-Von-Mises's Test results: "
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Si nota subito che i test Chi quadrato di Pearson, Kolmogorov-Smirnov e Cramer-Von-Mises rifiutano sempre l'ipotesi nulla, mentre il test di Wilcoxon-Mann-Whitney accetta sempre l'ipotesi; questi risultati sono superflui ai fini dell'analisi, l'unica informazione che danno è che il metodo adottato sembra non funzionare bene con dati reali.
In figura 4.6 si possono osservare le funzioni di ripartizione delle regioni in

esame (in nero) contro tutte le altre regioni (in rosso).

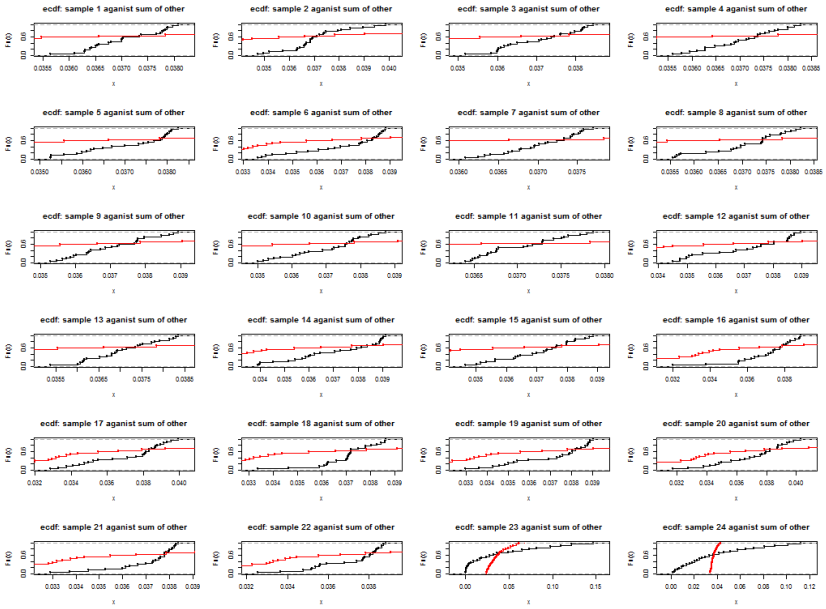

Figura 4.6: funzione di ripartizione empirica del campione in analisi (colore nero) contro la funzione di ripartizione empirica della somma degli altri campioni (rosso)

4.3.2 Secondo Metodo

Nel secondo metodo, per ogni risultato verrà fornito il numero di rifiuti dell'ipotesi H_0 nei test verificati tra l'i-esimo campione e tutti gli altri campioni.
Per i dati fittizi si è fatta una simulazione di 50 prove, raccogliendone i risultati e visualizzando una media generale della simulazione:

```
> simMultiTest(nsim=nSim, row=row, col=col)

[1] "Pearson's Chi-Square Test results: "
 [1] 6.68 6.32 5.84 8.14 6.42 6.68 6.50 6.14 7.20 7.24 7.22 4.34
[1] "Kolmogorov-Smirnov's Test results: "
 [1] 3.36 3.26 3.28 9.08 3.40 3.30 3.24 9.06 3.40 3.14 3.26 9.06
```

```
[1] "Wilcoxon-Mann-Whitney (or Mann-Whitney U) Test results: "
 [1] 3.38 3.50 3.36 9.08 3.36 3.32 3.36 9.08 3.48 3.14 3.38 9.04
[1] "Cramer-Von-Mises's Test results: "
 [1] 3.52 3.42 3.44 9.08 3.52 3.38 3.40 9.10 3.54 3.14 3.44 9.06
```

Si può notare che, al contrario dei risultati del test Chi-quadrato di Pearson, i test Kolmogorov-Smirnov, Wilcoxon-Mann-Whitney e Cramer-Von-Mises evidenziano le righe 4, 8 e 12 con dei "picchi" di rifiuti dell'ipotesi H_0. le altre righe invece hanno valori compresi tra 3 e 4; sembra, quindi, che il metodo riesca a evidenziare le righe con comportamenti anomali rispetto alle altre.
Si sottolinea però che questa è una simulazione di 50 prove, nelle singole prove può essere che qualche riga venisse evidenziata come diversa quando in realtà era simile alle altre.
I risultati che si hanno analizzando il dataset reale, invece, non sono così promettenti.
Risultati dei dati reali analizzati usando il secondo metodo:

```
> multiTest(occ)

[1] "Pearson's Chi-Square Test results: "
 [1] 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
 [21] 23 23 23 23
[1] "Kolmogorov-Smirnov's Test results: "
 [1] 23 23 23 23 23 23 23 23 22 23 23 23 23 23 22 23 23 23 23 23 23
 [21] 23 23 23 23
[1] "Wilcoxon-Mann-Whitney (or Mann-Whitney U) Test results: "
 [1] 23 23 22 23 22 22 22 22 21 23 23 22 21 22 21 23 23 23 23 22 23
 [21] 23 21 15 23
[1] "Cramer-Von-Mises's Test results: "
 [1] 23 23 23 23 23 23 23 23 22 23 23 23 23 23 22 23 23 23 23 23 23
 [21] 23 23 23 23
```

É evidente che tutti i test tendono a rifiutare qualsiasi campione paragonato agli altri, senza evidenziare maggiori rifiuti in corrispondenza delle righe inventate.
Si nota inoltre che nel test di Wilcoxon la regione Sfortunata, una regione fittizia, fallisce solo 15 volte nonostante la regione non sia compatibile con le altre.

4.3.3 Conclusioni sui test statistici di GoF

Per concludere, i due metodi presi in esame sono stati insoddisfacenti e i test statistici sfruttati non hanno dato i risultati sperati.
In particolare sembra che tali test non rispondano bene in presenza di un grande numero di dati o con valori particolarmente alti.

Capitolo 5

L'evoluzione del progetto: la Cluster Analysis

Dopo la sperimentazione descritta nel capitolo 4, ci si è chiesti se un analisi dei gruppi non supervisionata potesse dare lo stesso tipo di informazioni che poteva dare la segnalazione di dissimilarità fra le righe.

5.1 Motivazioni del passaggio da Test statistici al Machine-Learning

Le motivazioni principali di questo cambiamento sono in primis lo scarso risultato ottenuto nelle analisi di vari test statistici e anche le restrizioni comportate dall'utilizzo di essi.
Inoltre, una informazione che i test statistici analizzati non sfruttano è il confronto tra osservazioni "appaiate", cioè la possibilità di confrontare i valori presenti nelle diverse righe usando le stesse colonne.
Si prenda come esempio il caso dei dati reali usati come dataset di esempio nella sezione 4.3: questi dati sono delle serie storiche, quindi è interessante confrontare le regioni analizzando le stesse annate, e non mettendo a confronto annate distinte.
Non è stato sviluppato un adattamento dei test statistici utilizzati o cercato un altro test che sfruttasse tale informazione facendogli fare questa tipologia di confronto. La motivazione è dovuta alle assunzioni di tali test, più o meno simili ai test già analizzati.
Viceversa la cluster analysis permette una certa dinamicità dovuta al metodo

con cui viene calcolata la distanza tra le righe.
Un'altra informazione che la Cluster Analysis può dare è la suddivisione in gruppi, come si vedrà nella successiva sezione, che da quindi la possibilità all'utente di ricavare informazioni in più.
Anche questa possibilità ha contribuito notevolmente alla scelta di questo passaggio.
Nel prossimo capitolo si illustra la Cluster Analysis usata, e successivamente verrà descritta anche la sua implementazione e l'interfaccia grafica creata.

5.2 Herarchical Clustering

Il tipo di Cluster Analysis usata per il progetto è la **classificazione gerarchica** (Herarchical Clustering): è un tipo di classificazione non supervisionata in quanto non è prefissato il numero di cluster da trovare.
La classificazione gerarchica consiste nell'individuazione di un insieme di raggruppamenti ordinati in senso crescente; inizialmente ogni elemento, nel nostro caso ogni riga, costituisce un cluster e ad ogni iterazione dell'algoritmo si uniscono i due cluster più vicini, basandosi sulle distanze calcolate tra i cluster, fino ad ottenere un unico cluster.
Il metodo appena descritto appartiene alla categoria dei **metodi agglomerativi**, cioè quei metodi che partono da tanti cluster quante sono le unità da analizzare e procedono unendo ogni unità di volta in volta.
Vi sono anche metodi che funzionano in maniera contraria; questi metodi si chiamano **scissori** e partono da un unico cluster per poi dividerlo in due cluster distinti fino ad ottenere n cluster.
Nel progetto sono stati usati solo metodi agglomerativi in quanto comportano un minor rischio di pervenire a suddivisioni delle unità che non rispecchiano l'effettiva struttura dei dati.
Vi sono quindi tre parametri da scegliere in un'analisi di questo tipo:

- la **distanza**: la formula usata per determinare, con un risultato numerico, quanto diverse sono due unità;
- il **metodo agglomerativo**: il metodo usato per determinare la nuova serie o la nuova distanza dopo l'unione di due cluster;
- la **scalatura** applicata ai dati: il calcolo usato per uniformare le diverse unità, se usato.

La scelta di questi tre parametri è cruciale per l'analisi in quanto ognuno di essi preso singolarmente può determinare risultati totalmente differenti. Verrà spiegato, nel corso di questo capitolo, il peso che ha ognuno dei tre parametri nell'analisi e cosa comporta la scelta di un metodo o formula rispetto ad un altro.
Il risultato di questa analisi, cioè l'insieme di raggruppamenti in ordine crescente, viene usualmente rappresentato in un dendrogramma.
Questo grafico riporta sull'asse delle ordinate una scala delle distanze mentre sull'asse delle ascisse un identificativo delle unità. All'interno invece mostra l'unione delle unità al relativo livello di distanza. In questo modo il dendrogramma riesce a rappresentare ogni livello di aggregazione indicando anche il valore della distanza al momento dell'unione.
Il passo finale nell'analisi gerarchica consiste nella scelta di un punto di taglio: basterà tracciare una linea orizzontale in corrispondenza del livello di distanza prescelto; apparterranno a uno stesso gruppo tutte le unità congiunte da tratti orizzontali che si trovano al di sotto della linea tracciata e non incluse in eventuali gruppi precedenti.
In figura 5.1 si può osservare un dendrogramma per i dati sull'occupazione, con il punto di taglio evidenziato dalla linea rossa. Il punto di taglio scelto forma quindi i seguenti 6 gruppi di regioni:

1. Sfortunata;
2. Valle d'Aosta e Fruili;
3. Puglia, Calabria, Campania, Basilicata, Sardegna, Molise e Sicilia;
4. Prov. autonoma di Trento;
5. Utopia;
6. Umbria, Emilia-Romagna, Lombardia, Abruzzo, Veneto, Lazio, Prov. autonoma di Bolzano e Trentino;
7. Toscana, Marche, Liguria e Piemonte.

Un analisi di questo tipo può suggerire all'utente che vi sia una certa distinzione tra regioni del nord e regioni del sud, indicando anche raggruppamenti non prevedibili ma che potrebbero rispecchiare la verità.

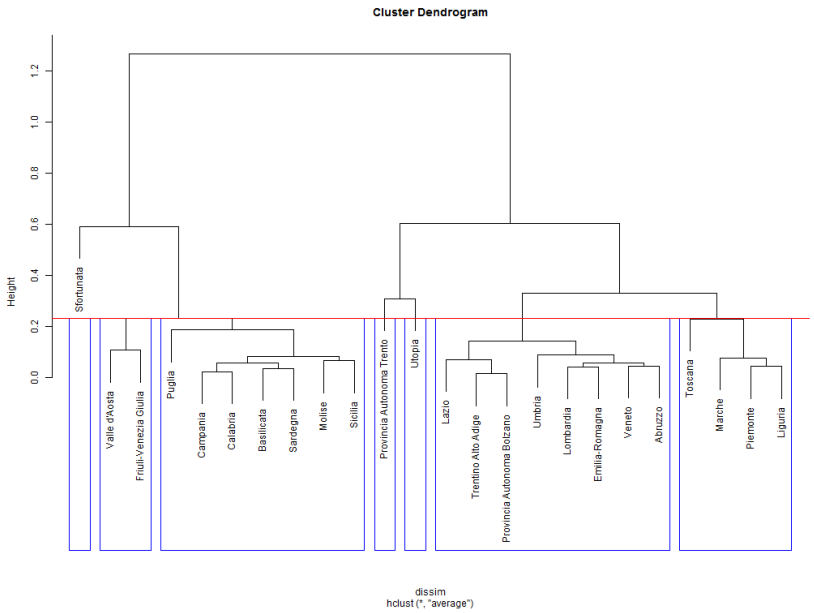

Figura 5.1: Dendrogramma dei dati sull'occupazione, in rosso il punto di taglio, in blu le evidenziazzioni dei gruppi che crea tale taglio

5.2.1 Misure di dissimilarità

Le misure di dissimilarità nell'analisi gerarchica servono a quantificare la similarità o la distanza di un unità rispetto ad un'altra, secondo una determinata regola che può privilegiare determinati aspetti delle unità (massimo, minimo, media, varianza, andamento, ecc.).
Come accennato nella sezione precedente, questa misura ricopre un ruolo fondamentale nella classificazione gerarchica in quanto determina i futuri raggruppamenti.
Una 'buona' misura dovrà essere rappresentativa delle caratteristiche di interesse e sintetizzare in maniera adeguata la similarità tra le due unità a confronto.

Vi sono quindi alcune proprietà che caratterizzano misure diverse:

$$\begin{array}{lcr} (P1) & \text{non negatività} & d(x_1, x_2) \geq 0 \forall x_1, x_2 \in X \\ (P2) & \text{identità} & d(x_1, x_2) \Leftrightarrow x_1 = x_2 \\ (P3) & \text{simmetria} & d(x_1, x_2) = d(x_2, x_1) \forall x_1, x_2 \in X \\ (P4) & \text{triangolare} & d(x_1, x_2) \leq d(x_1, x_3) + d(x_2, x_3) \forall x_1, x_2, x_3 \in X \\ (P5) & \text{ultrametrica} & d(x_1, x_2) \leq \max\{d(x_1, x_3), d(x_2, x_3)\} \forall x_1, x_2, x_3 \in X \end{array} \tag{5.1}$$

Le prime quattro proprietà caratterizzano le distanze metriche, invece le prime tre e la quinta caratterizzano le ultrametriche. Vengono definite distanze se valgono le proprietà P1, P2, P3; se invece valgono le proprietà P1, P3 si definiscono indici di dissimilarità.
In generale queste distanze hanno bisogno di una scalatura opportuna dei dati per funzionare bene: infatti senza portare le variabili alla stessa scala queste misure potrebbero classificare come distanti due variabili che in realtà non lo sono.
Basti pensare a delle provette di sangue in cui si sono rilevati parametri quali emoglobina, ferro ecc; per poter analizzare al meglio questi dati occorre prima portarli alla stessa scala (ml^3) e poi effettuare l'analisi.
Esistono molti tipi di misure, per variabili quantitative, qualitative o entrambe; verranno descritte solo quelle implementate nel progetto e una di futura implementazione.

Distanza Euclidea

La distanza Euclidea rappresenta una normale distanza tra due punti in uno spazio euclideo. La distanza viene calcolata tramite il teorema di Pitagora:

$$L_2(x_i, x_j) = \left[\sum_{t=1}^{q} |x_{it} - x_{jt}|^2 \right]^{1/2} \tag{5.2}$$

Distanza di Canberra

La distanza di Canberra viene definita la versione pesata della distanza di Manhattan, cioè la minor distanza da percorrere per un taxi in una città suddivisa in isolati quadrati; la distanza fra due vettori x_i, x_j in uno spazio

vettoriale n-dimensionale è data dalla seguente formula:

$$d(x_i, x_j) = \sum_{k=1}^{n} \frac{|x_{ik} - x_{jk}|}{|x_{ik}| + |x_{jk}|} \tag{5.3}$$

Viene solitamente usata come metrica per comparare liste con priorità o per rilevare intrusioni nel campo della sicurezza sui computer.

Indice di correlazione di Pearson

Questo indice prende in considerazione la sola correlazione e serve ad esprimere una relazione di linearità tra due variabili statistiche; è definito come:

$$\rho_{XY} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y} \tag{5.4}$$

dove σ_{XY} è la covarianza tra X e Y, mentre σ_X e σ_Y sono le deviazioni standard delle due variabili statistiche.
Questo indice varia tra -1 a 1, dove per $\rho < 0$ si dice che le due variabili sono correlate, per $\rho > 0$ si dice che sono inversamente correlate, mentre per $\rho = 0$ che sono incorrelate.
Viene quindi considerata una sua riscalatura secondo la formula $d_{XY} = 1 - \rho_{XY}$.
Si noti che questo tipo di distanza non necessita di una scalatura dei dati prima dell'analisi in quanto usa solo le covarianze e varianze.

Indice di similarità di Gower

Per quanto riguarda il caso misto, cioè in presenza di variabili sia qualitative che quantitative, può essere usato l'indice di similarità di Gower. Tale indice si presenta come una sorta di media pesata di misure di tipo diverso, le quali risultano adatte alle variabili che stanno misurando.
Viene formalizzato nella maniera seguente:

$$s_G(x_i, x_j) = \frac{\sum_{h=1}^{g} \delta_{ij}(h) s_{ij}(h)}{\sum_{h=1}^{g} \delta_{ij}(h)} \tag{5.5}$$

con

$$s_{ij}(h) = \begin{cases} 1 - \dfrac{|x_{ih} - x_{jh}|}{range(x_h)}, & \text{se } x_h \text{ quantitativa} \\ I(x_{ih} = x_{jh}), & \text{se } x_h \text{ dicotomica} \\ 1 - \left|x''_{ih} - x''_{jh}\right|, & \text{se } x_h \text{ ordinale} \end{cases} \tag{5.6}$$

$$\delta_{ij}(h) = \begin{cases} 1, & i, j \text{ confrontabili rispetto a } x_h \\ 0, & i, j \text{ non confrontabili rispetto a } x_h \end{cases} \tag{5.7}$$

La 5.7 assume valore 0 se nelle due unità a confronto manca un valore o entrambi i valori. Mentre per quanto riguarda il sistema 5.6, si noti che si possono sostituire le metriche usate per le variabili quantitative o qualitative con altre metriche più adatte al caso.
Questa distanza non è stata implementata nel progetto, ma viene descritta in quanto usando questa distanza si potrebbe ampliare il tipo di analisi anche per i casi dove vi sono variabili quantitative e qualitative.

5.2.2 Metodi agglomerativi

Come detto precedentemente, i metodi agglomerativi consistono nell'unione di due cluster, calcolando la distanza del nuovo cluster verso gli altri basandosi su un qualche metodo.
Quindi non vengono uniti i valori dei cluster, ma viene solo aggiornata la matrice delle distanze applicando uno dei metodi descritti di seguito.
Vengono illustrati i metodi più usati ed implementati nel progetto.

Single Linkage

Questo metodo prevede di prendere il minimo delle distanze quando viene creato un nuovo cluster:

$$\delta(C_1, C_2) = \min\{d(x_i, x_j) : x_i \in R_1, x_j \in R_2\} \tag{5.8}$$

Questo metodo però può comportare un effetto a cascata: si unisce un cluster, e questo nuovo cluster si unisce subito con un altro cluster e così via. In figura 5.2 si può osservare un dendrogramma in cui si è verificato questo effetto nella parte destra.

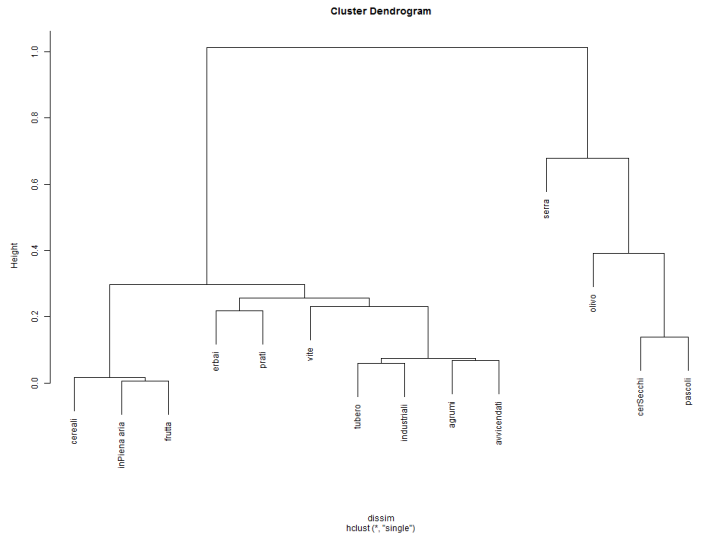

Figura 5.2: Esempio di effetto a cascata nella parte a destra (dataset: Agricoltura per tipologia)

Complete Linkage

Al contrario del Single Linkage, questo metodo prende il massimo delle distanze:

$$\delta(C_1, C_2) = \max\{d(x_i, x_j) : x_i \in R_1, x_j \in R_2\} \tag{5.9}$$

Con questo metodo però si tende a creare cluster compatti, cioè chiusi verso l'unione con altri in quanto un nuovo cluster prende sempre la massima distanza verso gli altri.

Average Linkage

É un metodo intermedio rispetto al single e il complete linkage; prevede di usare la media delle distanze tra i due cluster da unire:

$$\delta(C_1, C_2) = \frac{1}{n_1 n_2} \sum_{x_i \in R_1} \sum_{x_j \in R_2} d(x_i, x_j) \tag{5.10}$$

Distanza delle medie

La distanza delle medie prevede di ricalcolare la distanza tra il nuovo cluster e gli altri, assegnando al nuovo cluster valori che sono la media dei

valori dei due cluster:

$$\delta(C_1, C_2) = \frac{1}{n_1 n_2} d(\bar{x}_1, x_2) \tag{5.11}$$

con R_1 nuovo cluster avente valori $\bar{x}_1$ uguali a

$$\bar{x}_1^T = \left(\sum_{x_i \in R_1} x_{i1} \cdots \sum_{x_i \in R_1} x_{iq} \right)$$

Anche questo metodo, come l'Average Linkage, rappresenta una via di mezzo rispetto al Single e il Complete Linkage.
A differenza degli altri, però, questo metodo necessita di un parziale ricalcolo della matrice delle distanze.

5.2.3 Tipi di scalature

La scalatura dei dati è una delle scelte cruciali nell'analisi gerarchica. Come accenato precedentemente, senza un adeguata scalatura le distanze non saranno in grado di dare risultati veritieri per la realtà d'interesse. La scalatura dei dati serve quindi a portare variabili con diversi "misure" ad una stessa in modo tale da renderla confrontabile dalla distanza scelta.

Standardizzazione

Il classico metodo statistico che porta variabili con media e varianza distinta ad averla unitaria e pari a 0 per la media, 1 per la varianza.
Il calcolo consiste nel trovare la media e la varianza del campione, e di applicare la seguente formula per trovare la nuova j-esima osservazione:

$$x_j^{'} = \frac{(x_j - \bar{x})}{\sigma_x} \tag{5.12}$$

La perdita di parametri come media e varianza può essere sbagliata nel caso in cui la distanza le usi per confrontare due unità.

Min-Max

Un tipo di scalatura semplice senza presupposti statistici è la scalatura "Min-Max"; consente di portare i dati in una scala da 0 a 1, riducendo media

e varianza dei dati ma senza portarli ad averle unitarie come avviene nella standardizzazione.
La scalatura per un elemento x_j del vettore x si ottiene:

$$x_j^{'} = \frac{(x_j - \min x)}{\max x} \tag{5.13}$$

5.3 Un grafico force-direct sfruttando solo la matrice delle distanze

Nell'analisi dei requisiti (capitolo 2) si è accennato alle tipologie di grafico create.
Il grafico che si vuole approfondire nel corso di questa sezione è quello basato sulla sola matrice delle distanze.
Il metodo usato differisce in maniera sostanziale rispetto a quello dell'analisi gerarchica, non c'è il meccanismo di aggregazione e viene usata solo la matrice delle distanze, creando un collegamento tra due nodi solo se la loro distanza è minore o uguale al punto di taglio scelto dall'utente.
Il nome del grafico, force-based, si riferisce alla gestione delle forze in D3: nel grafico viene creato solo un collegamento tra i due nodi ed aggiunto un vettore di forza tra i due nodi, di conseguenza D3 elabora i movimenti di ogni singolo nodo basandosi sulle leggi della fisica e i vincoli.
La barra di threshold varia in base alla dimensione della matrice delle distanze, consentendo di scegliere un valore di questa matrice partendo dalle distanze piccole a quelle grandi.
Il calcolo della matrice delle distanze comporta comunque una scelta, quella della scalatura e del tipo di distanza. Come descritto nel corso del capitolo, si potrebbe scegliere l'indice di Gower, permettendo cosi di analizzare qualsiasi tipo di dato in ingresso.
Nel nostro caso é stato scelto di analizzare delle serie storiche e di conseguenza di usare l'indice di similarità di Pearson, non come ulteriore perdita di generalità, ma per il poco tempo che era rimasto per l'implementazione. Inoltre questo indice ci consente di non applicare nessun tipo di scalatura in quanto viene usata la correlazione come distanza.
Per la creazione di questo tipo di grafico si disegnano per primi i nodi, etichettati con il nome della loro serie, e poi si vanno a inserire tutti i collegamenti che rispettano il vincolo $d(x_i, x_j) \leq T$ dove T rappresenta il valore di threshold scelto dall'utente.

Questa procedura si può tradurre nel pseudo-codice contenuto nell'algoritmo 1.
Questo algoritmo non corrisponde all'attuale implementazione presente nel

```
   Dati: tabellaPivot
   Risultato: Oggetto SVG
 1 /* Inizializzazione                                          */
 2 svg ← D3.creaSvg();
 3 NomiRighe ← getNomiRighe(tabellaPivot);
 4 per ciascun elemento t in NomiRighe fai svg.addNodo(t);
 5 matriceDistanze ← lib.getMatriceDistanze(tabellaPivot, "Pearson");
 6 /* Aggiunta collegamenti e forze                             */
 7 limiteThreshold ←
   ordina(matriceDistanze.inArray())[barraThreshold.getValore()];
 8 per i ← 0 a (nRighe(matriceDistanze)-1) fai
 9 |   per j ← (i + 1) a nRighe(matriceDistanze) fai
10 |   |   se matriceDistanze[i,j] < limiteThreshold allora
11 |   |   |   svg.addCollegamento(i, j);
12 |   |   |   svg.addForze(i, j);
13 |   fine
14 fine
   Uscita: svg
```

Algorithm 1: Creazione di un grafico Force-Based

progetto, ma riassume i passaggi fondamentali che effettua il codice per creare tale tipo di grafico, tralasciando le funzioni scritte come il calcolo della matrice delle distanze.
In sintesi, per prima cosa si crea un oggetto SVG dove inserire gli elementi del grafico (riga 2), e subito si prendono i nomi delle righe presenti nella tabella di pivot e si inseriscono nel grafico creando dei nodi opportunamente etichettati (riga 3 e 4).
Creati i nodi, si calcola la matrice delle distanze (riga 5) e si calcola il punto di Threshold selezionato dall'utente (riga 7). Infine si scorre una parte della matrice delle distanze (in questo caso la parte triangolare superiore, essendo simmetrica la matrice delle distanze non occorre scorrerla tutta) e si creano i collegamenti e i vettori di forza, ma solo nel caso in cui la distanza sia minore da quella scelta dall'utente (righe 10, 11 e 12).

Per l'aggiornamento del grafico dopo una variazione del punto di taglio scelto, basta ripetere dalla riga 7 in poi l'algoritmo 1, aggiungendo però un blocco **Altrimenti** per la condizione in riga 10 che permetta la cancellazione di collegamenti e forze nel caso esistano. Il blocco dalla riga 10 alla 12 si tradurrà quindi nell'algoritmo 2.
Un utile accorgimento su questo tipo di grafico è la sua forte somiglianza al-

```
se (matriceDistanze[i,j] < limiteThreshold) ∧
(¬svg.contieneCollegamento(i, j)) allora
    svg.addCollegamento(i, j);
    svg.addForze(svg, i, j);
allora
    se svg.contieneCollegamento(i, j) allora
        svg.removeCollegamento(i, j);
        svg.removeForze(i, j);
fine
```

Algorithm 2: Variazione del blocco di codice dalla riga 10 alla 12 dell'algoritmo 1

l'analisi gerarchica; i collegamenti che crea questo grafico, e quindi i gruppi, corrispondono al risultato di una analisi gerarchica con metodo agglomerativo di tipo single linkage.
É possibile notarlo guardando i collegamenti fatti dal grafico descritti nella tabella sotto (i collegamenti sono in successione a partire dalla prima colonna, poi la seconda e così via) e confrontando i risultati con il dendrogramma riportato sopra la tabella.

Figura 5.3: Dendrogramma relativo ai dati sull'agricoltura effettuato con l'indice di correlazione di Pearson e il metodo di agglomerazione single-Linkage

inPiena aria > frutta	cereali > inPiena aria
cereali > frutta	tubero > industriali
agrumi > avvicendati	tubero > agrumi
tubero > avvicendati	industriali > agrumi
cerSecchi > pascoli	industriali > avvicendati
erbai > prati	tubero > vite
erbai > avvicendati	agrumi > erbai
cereali > industriali	industriali > erbai
inPiena aria > industriali	industriali > frutta
industriali > vite	agrumi > prati
vite > avvicendati	avvicendati > prati
cereali > tubero	tubero > erbai
olivo > pascoli	tubero > frutta
agrumi > vite	tubero > inPiena aria
vite > erbai	industriali > prati
cerSecchi > olivo	cereali > agrumi
frutta > vite	inPiena aria > vite
tubero > prati	cereali > vite
frutta > agrumi	inPiena aria > agrumi
pascoli > serra	cereali > avvicendati
frutta > avvicendati	inPiena aria > avvicendati
frutta > erbai	vite > prati
inPiena aria > erbai	cerSecchi > serra
cereali > erbai	olivo > serra
inPiena aria > serra	

Tabella 5.1: Unioni compiute dal grafico Force-based (ordinati per colonna) per la raffigurazione dei dati sull'agricoltura

5.4 Un grafico per l'analisi gerarchica, il Cluster-Based

Questo grafico, al contrario del Force-Based, é una rappresentazione completa di una analisi gerarchica.
Come per il precedente, si parte con il disporre i nodi, e in base al punto di taglio scelto dall'utente, si vanno a raggruppare i nodi appartenenti ai cluster formati, evidenziando il cluster con uno sfondo colorato e applicando una forza fra di loro che li vincoli a stare vicini.
Per ottenere tale vincolo si è scelto di creare, per ogni cluster formato da più di un elemento, un nodo di pivot invisibile all'utente. Su tale nodo si creano i collegamenti verso gli altri cluster; in questo modo i cluster rappresentati avranno una dimensione perfettamente circolare.
I parametri scelti per l'analisi gerarchica sono gli stessi usati nel grafico Force-Based, con l'aggiunta del metodo agglomerativo, non necessario per l'altro tipo di grafico.
I metodi agglomerativi implementati nel progetto sono quelli descritti nel capitolo, però si vuole escludere all'utente la possibilità di scegliere un metodo, quindi si è scelto come metodo predefinito l'average linkage.
L'algoritmo per la creazione di questo grafico si può tradurre nel modo descritto dall'algoritmo 3.
L'inizializzazione è simile a quella del grafico Force-based, eccetto per la riga 5: il risultato della funzione *getAnalisiGerarchica* è un albero in formato JSON che, tramite le diramazioni, descrive un dendrogramma aggiungendo anche ad ogni nodo non foglia quando è stato unito.
Una volta scelto il punto di taglio dall'utente, viene esaminato l'albero (riga 8) dalla funzione *esaminaAlbero*; questa funzione è ricorsiva, ad ogni nodo esamina se il numero del raggruppamento corrisponde al punto di taglio, e se minore o uguale aggiunge il pezzo di albero che sta esaminando all'array che ritornerà, sennò per ogni elemento di quell'albero invoca se stessa.
In questo modo *clusterDaCreare* sarà popolato di nodi o frammenti di albero che corrispondono a tutti i cluster da creare.
Infine nel ciclo dalla riga 9 alla 18 si creano i nodi di pivot, i collegamenti, le forze e si applicano gli sfondi (delle forme geometriche in secondo piano).
L'aggiornamento di questo grafico si può realizzare eliminando tutti i nodi pivot e tutti gli sfondi creati in precedenza prima dell'inizio del ciclo (riga 9).

Dati: dataset
Risultato: Oggetto SVG

```
1  /* Inizializzazione                                          */
2  svg ← D3.creaSvg();
3  NomiRighe ← getNomiRighe(dataset);
4  per ciascun elemento t in NomiRighe fai svg.addNodo(t);
5  alberoAnalisi ← lib.getAnalisiGerarchica(dataset, "Pearson",
   "Average");
6  /* Aggiunta collegamenti, forze e nodi di pivot              */
7  limiteThreshold ← barraThreshold.getValore();
8  clusterDaCreare = esaminaAlbero(alberoAnalisi, limiteThreshold);
9  per i ← 0 a lunghezza(clusterDaCreare) fai
10 |   nodiCluster = getNodi(clusterDaCreare[i]);
11 |   pivotNode = svg.addNodo();
12 |   svg.applicaSfondo(pivotNode);
13 |   per j ← 0 a lunghezza(nodiCluster) fai
14 |   |   svg.aggiungiCollegamento(nodiCluster[j], pivotNode);
15 |   |   svg.aggiungiForze(nodiCluster[j], pivotNode);
16 |   |   svg.applicaSfondo(nodiCluster[j]);
17 |   fine
18 fine
```

Uscita: svg

Algorithm 3: Creazione di un grafico Cluster-Based

Non serve eliminare anche i collegamenti e le forze in quanto quando le si ricreano D3 rimuove tutti i collegamenti prima di crearne altri.
Nel capitolo successivo verrà discussa l'implementazione dei due grafici e la struttura del progetto.

Capitolo 6

Implementazione del progetto

Il progetto è stato suddiviso in tre file JavaScript e un file html per la presentazione e l'utilizzo degli oggetti JavaScript.
Per ogni file verrà presentato un elenco delle funzioni, indicando i parametri in ingresso e il parametro di ritorno, soffermandoci sull'oggetto designer che si occupa di disegnare i due tipi di grafici.
Nel corso del capitolo non si approfondirà l'implementazione del metodo di analisi ne delle singole distanze, ma ci si soffermerà in particolar modo sul disegnatore.

6.1 StatLibrary.js

Questo oggetto contiene una libreria di funzioni, tra cui quelle per il calcolo della distanza, la scalatura dei dati e altre funzioni d'appoggio quali media e varianza.
Le funzioni implementate e utilizzabili da chiunque importi lo script sono:

- mean ([] sample): restituisce la media di sample;
- variance ([] sample): restituisce la varianza di sample;
- normGauss ([] sample): restituisce l'array sample standardizzato secondo la regola descritta nella sezione 5.2.3;
- normMinMax ([] sample): restituisce l'array sample scalato secondo il metodo Min-Max (sezione 5.2.3);

- euclideanDistance ([] sampleA, [] sampleB): restituisce il calcolo della distanza tra l'array sampleA e l'array sampleB secondo la distanza Euclidea (sezione 5.2.1);

- canberraDistance ([] sampleA, [] sampleB): come per *euclideanDistance*, ma usando la distanza di Canberra;

- pearsonIndex ([] sampleA, [] sampleB): restituisce l'indice di correlazione di Pearson tra i due array sampleA e sampleB.

6.2 ClusterHelper.js

Questo prototipo sfrutta le funzioni scritte in StatLibrary per creare la matrice delle distanze e per effettuare l'analisi gerarchica dei vari dataset. Questo oggetto quando viene costruito richiede un parametro *options*, un oggetto in JSON del tipo:

```
1    method      CL_** metodo agglomerativo da applicare
2    typeDistance  DS_** tipo di distanza da usare per la
        analisi
3    norm      NR_** tipo di scalatura
4    reDistance    boolean se vero ricalcola ad ogni nuovo
        cluster la distanza verso gli altri
```

Codice 6.1: Descrizione parametro *options* per l'oggetto Cluster

I tipi indicati con CL_**, DS_** e NR_** sono delle costanti definite nel prototipo per la scelta dei metodi di aggregazione, di distanza e di scalatura. L'oggetto implementa i seguenti metodi:

- normalize ([] row, type): esegue la normalizzazione per l'array row secondo il tipo scelto;

- getDistanceMatrix (dataset, options): ritorna la matrice delle distanze per l'oggetto dataset secondo le opzioni contenute dentro l'oggetto options;

- getHerarchicalClustering (dataset, options): ritorna un oggetto JSON ad albero contenente il risultato di un'analisi gerarchica per il dataset in input secondo le opzioni scelte;

- jsonJoin (a, b, nIter, dist): usato quando si crea un cluster per creare un oggetto JSON che rispecchi questa unione salvando anche a che iterazione è avvenuta l'unione e che valore della distanza aveva.

I metodi *normalize* e *jsonJoin* vengono utilizzati solo all'interno dell'oggetto, non vengono usati dal disegnatore che si occuperà di chiamare il metodo *getDistanceMatrix* quando deve creare un grafico di tipo Force-Based, o il metodo *getHerarchicalClustering* quando dovrà creare un grafico di tipo Cluster-Based.

6.3 Designer.js

Questo oggetto si occupa della creazione e dell'implementazione dei grafici di tipo Force-Based e Cluster-Based.
Come per il prototipo Cluster.js, il disegnatore implementa le stesse costanti ed utilizza parametri in più nell'oggetto options. I parametri del "costruttore" quindi sono:

```
1  idPos, id del tag HTML dove posizionare il grafico
2  options, oggetto JSON con:
3    graphStyle    GS_**   Tipo di grafico (force-based,
       cluster-based)
4    widthSvg    integer   larghezza del grafico
5    heightSvg   integer   altezza del grafico
6    widthTip    integer   larghezza del popup
7    heightTip   integer   altezza del popup
8    thresholdBar  DOM object  oggetto DOM riferito alla barra
       di Threshold
9    method      CL_**   metodo agglomerativo da applicare
10   typeDistance  DS_**   tipo di distanza da usare per la
       analisi
11   norm      NR_**   tipo di scalatura
12   reDistance    boolean   se vero ricalcola ad ogni nuovo
       cluster la distanza verso gli altri
```

Codice 6.2: Descrizione dei parametri per il costruttore di Designer

Alla creazione dell'oggetto, il disegnatore setta alcune variabili riguardanti l'oggetto JSON di input, e crea l'oggetto SVG su cui andrà poi a creare il grafico implementando anche lo stile dei componenti del grafico.
Una volta creato l'oggetto, il disegnatore rimane in attesa della prima chiamata al metodo *loadData*.

Questo metodo si occupa solo del caricamento dei dati e di effettuare un eventuale pulizia dell'SVG da elementi precedentemente creati. Una volta che il caricamento dei dati è stato effettuato, il disegnatore rimane nuovamente in attesa della chiamata al metodo *showGraph()* o il metodo *updateOptions(options)*.
Il metodo *updateOptions* si occupa solo dell'aggiornamento delle opzioni scelte, e se il grafico non è ancora stato disegnato chiama *showGraph*.
Questo metodo, invece, parte con la creazione dei nodi, setta alcuni parametri riguardanti la barra di threshold e aggiunge l'ascoltatore agli eventi *change* e *input* alla barra, mentre per i nodi aggiunge l'ascoltatore agli eventi *mouseover*, *mouseout* e *dbclick*.

```
1  this.showGraph = function() {
2    svg.selectAll("g").remove();
3    nodeColor = [];
4    force.nodes(nodes);
5
6    node = svg.selectAll(".node")
7      .data(nodes)
8      .enter().append("g")
9      .attr("class", "node")
10     .call(force.drag);
11
12   (function(_this, svg){
13     svg.selectAll(".node").on("mouseover", function(d){
           _this.showTip(d)});
14     svg.selectAll(".node").on("mouseout", function(){_this.
           hideTip()});
15     svg.selectAll(".node").on("dblclick", function(d){
16       d.fixed = !d.fixed;
17       if(!d.fixed)
18         force.tick();
19     });
20   })(this, svg);
21
22   node.append("svg:circle")
23     .attr("r", rCircle)
24     .style("fill", function() {
25       var r = Math.random() * 255;
26       var g = Math.random() * 255;
27       var b = Math.random() * 255;
28       var color = d3.rgb(r,g,b);
29       nodeColor.push(color);
```

```
30          return color;
31        }); //colore a caso!!
32
33      node.append("text")
34        .attr("dx", 12)
35        .attr("dy", ".35em")
36        .text(function(d) { return d.name });
37
38      if(this.options.graphStyle == this.GS_CB){
39        clusterResult = maClu.getHerarchicalClustering(
            newSamples, this.options).children[0];
40        max = clusterResult.nIter;
41      } else {
42        distMatrix = maClu.getDistanceMatrix(newSamples, this.
            options);
43        distValue = [];
44
45        for(var i = 0; i < (distMatrix.length - 1); i++)
46          for(var j = (i + 1); j < distMatrix.length; j++){
47            distValue.push(distMatrix[i][j]);
48          }
49        distValue.sort(function(a,b){ return a-b;});
50        max = distValue.length-1;
51      }
52
53      if(this.thresholdBar){
54        this.thresholdBar.min = 0;
55        this.thresholdBar.max = max;
56        if(graphRecentChange){
57          this.thresholdBar.value = parseInt(max/4);
58          graphRecentChange = false;
59        }
60
61        (function(_this){
62          _this.thresholdBar.addEventListener("change",
              function(){_this.updateGraph});
63          _this.thresholdBar.addEventListener("input", function
              (){_this.updateGraph});
64        })(this);
65      }
66      this.updateGraph();
67    }
```

Codice 6.3: Metodo *showGraph* dell'oggetto Designer

Escludendo le righe dalla 13 alla 21, la prima parte di questo frammento di codice (fino alla riga 37) si occupa in successione di creare dei tag *g* per ogni nodo (righe 7-11) e di dire a D3 che questi elementi saranno i diretti interessati nel calcolo delle forze. Poi per ogni elemento *g* inserisce dentro un tag *circle* colorato in maniera casuale, e inserisce un tag *text* contenente l'etichetta del nodo (righe dalla 23 alla 37).
Si può notare l'uso dei metodi *selectAll*, *data*, *enter*, *append*: tali metodi sono forniti da D3 e permettono una più agevole manipolazione del DOM SVG. In particolare usando il metodo *data* si assegnano i dati da utilizzare, ed utilizzando poi *enter* si potranno eseguire operazioni per ogni elemento dei dati utilizzati.
Le righe dalla 13 alla 21 aggiungono l'event listener degli eventi *mouseover*, *mouseout* e *dbclick*. I primi due eventi servono per la visualizzazione di un popup informativo relativo al nodo selezionato quando si passa sopra con il puntatore del mouse, mentre l'evento *dbclick* serve a vincolare un nodo alla posizione in cui si trova facendo un doppio click su di esso.
Necessitano però del passaggio dello scope, quindi si è ricorsi alla creazione di una funzione apposita per passare il contesto *this* in modo tale che gli eventi generati riescano a chiamare i metodi dell'oggetto Designer *showTip*, *hideTip* e la funzione scritta nelle righe dalla 16 alla 20.
Mentre le righe dalla 39 in poi effettuano i calcoli per l'analisi chiamando i relativi metodi dell'oggetto Cluster.js e modificano la barra di threshold a seconda del grafico utilizzato. In particolare, quando si sceglie il grafico Force-Based, si converte l'intera matrice delle distanze in un array e lo si ordina in senso crescente (righe 45-51); questo viene fatto in modo tale da usare la barra di threshold come indice di questo array quando l'utente deve scegliere la distanza.
Infine, nelle righe dalla 55 alla 70, vengono riscritti i valori min e max dell'oggetto DOM relativo alla barra di Threshold, settando anche un valore di default alla barra se il grafico è stato appena disegnato. Inoltre vengono riscritti i due eventi *change* e *input* assegnando il metodo *updateGraph* del Designer.
Una volta che il metodo è stato eseguito, viene chiamato il metodo *updateGraph* per la creazione degli altri elementi del grafico.
Prima della spiegazione del metodo *updateGraph*, verranno spiegate delle funzioni di uso interno al metodo: i metodi *updateFB* e *updateCB*.

```
1  var updateFB = function(threshold){
2    for(var i = 0; i < (distMatrix.length - 1); i++) {
3      for(var j=(i+1); j < distMatrix.length; j++){
4        if(distMatrix[i][j] < distValue[threshold]){
5          links.push({
6            source : i,
7            target : j,
8            value : distMatrix[i][j]
9          });
10       }
11     }
12   }
13 };
```

Codice 6.4: Metodo *updateFB* per l'aggiornamento di un grafico Force-Based

Il ciclo annidato nelle righe 7-19 controlla se instaurare un collegamento tra due nodi confrontando la distanza dei due con l'array ordinato di tutte le distanze usando il valore intero dato dalla barra di threshold. Se la distanza è minore aggiunge quindi il collegamento all'array links creando un nuovo oggetto indicante i due elementi da collegare e il valore della distanza.
Nella funzione *updateCB* si vanno invece a creare i nodi di pivot e i relativi collegamenti, creando inoltre l'array *groups*, usato da *updateGraph* per definire gli sfondi dei cluster.

```
1  var updateCB = function(threshold){
2    clusterToPrint = [];
3    var esaminate = function(obj, thres) {
4      if(obj.nIter){
5        if(obj.nIter < (thres+1)){
6          clusterToPrint.push(obj);
7        } else {
8          for(var key in obj.children)
9            esaminate(obj.children[key], thres);
10       }
11     } else {
12       clusterToPrint.push(obj);
13     }
14   }
15   esaminate(clusterResult, threshold);
16   var getMDistIndex = function(obj) {
17     if(!obj.children){
18       return [obj.mDistIndex];
19     } else {
```

```
20          return getMDistIndex(obj.children[0]).concat(
                getMDistIndex(obj.children[1]));
21        }
22      }
23      p = []; groups = [];
24      var plus = 0; var groupN = 0;
25      for(var i = 0; i < clusterToPrint.length; i++, groupN++)
            {
26        if(clusterToPrint[i].nIter){
27          var points = getMDistIndex(clusterToPrint[i]);
28          var tmp = []
29          var sumX = 0, sumY = 0;
30          points.forEach(function(index){
31            links.push({
32              source : index,
33              target : nodes.length + plus,
34              value : 1,
35              nChild : points.length
36            });
37            sumX += nodes[index].x;
38            sumY += nodes[index].y;
39            nodes[index].group = groupN;
40            tmp.push(nodes[index]);
41          });
42          p.push({name:"pivot" + plus, group:groupN, nChild:
                points.length, x: sumX/points.length, y: sumY/
                points.length});
43          tmp.push(p[plus]);
44          groups.push({key:plus, values:tmp});
45          plus++;
46        } else {
47          nodes[clusterToPrint[i].mDistIndex].group = groupN;
48        }
49      }
50      force.nodes(nodes.concat(p));
51    };
```

Codice 6.5: Metodo *updateCB* per l'aggiornamento di un grafico Cluster-Based

Il metodo *esaminate* dalla riga 384 alla 394 corrisponde alla funzione *getNodi* descritta nella sezione 5.4, mentre la funzione *getMDistIndex* è a sua volta una funzione ricorsiva con lo scopo di ritornare tutti gli identificativi dei nodi partecipanti ad un cluster.
Quindi, nel ciclo alla riga 405, per ogni cluster che ha la proprietà *nIter* (proprietà inesistente nei nodi foglia dell'albero generato dall'analisi gerarchica)

si cercheranno innanzitutto gli id dei partecipanti al cluster; per ognuno di essi quindi si creerà il collegamento al nodo pivot, e si aggiungeranno le coordinate ($sumX$, $sumY$) dove è correntemente posizionato il nodo in modo da inserire il nodo di pivot centrato rispetto agli altri, altrimenti causerebbe molto movimento.
Infine si crea e si aggiunge il nodo pivot all'array p e all'array *groups* si aggiunge la coppia identificatore, nodo di pivot.
Una volta finito il ciclo si segnalano a D3 i nodi da usare come oggetti di forza nel grafico tramite la riga 430 (*force* è un oggetto creato durante l'inizializzazione del disegnatore con il comando `d3.layout.force()`).
Segue il metodo completo *updateGraph*:

```
1  this.updateGraph = function(thresValue) {
2    if(!node)
3      return;
4
5    links = [];
6    svg.selectAll(".pivot").remove();
7    svg.selectAll("path").remove();
8    if(this.thresholdBar){
9      var threshold = parseInt(thresValue || this.
           thresholdBar.value);
10   } else {
11     var threshold = thresValue || 25;
12     threshold = parseInt(max*threshold/100);
13   }
14
15   switch(this.options.graphStyle){
16     case this.GS_FB:
17       updateFB(threshold);
18       break;
19     case this.GS_CB:
20       updateCB(threshold);
21       break;
22   }
23   force.links(links);
24
25   svg.selectAll(".link").remove();
26
27   link = svg.selectAll(".link")
28     .data(links)
29     .enter().insert("line", "g")
30     .attr("class", "link");
31
```

```
32      if(this.options.graphStyle == this.GS_CB){
33        pivotNode = svg.selectAll(".node")
34          .data(nodes.concat(p))
35          .enter().insert("g", ":first-child")
36          .attr("class", "node pivot")
37          .call(force.drag);
38
39        svg.selectAll("path")
40          .data(groups)
41          .attr("d", groupPath)
42          .enter().insert("path", "g")
43          .style("fill", groupFill)
44          .style("stroke", groupFill)
45          .style("stroke-width", 40)
46          .style("stroke-linejoin", "round")
47          .style("opacity", .2)
48          .attr("d", groupPath);
49        (function(_this, svg){
50          svg.selectAll("path").on("mouseover", function(d){
                _this.showTip(d)});
51          svg.selectAll("path").on("mouseout", function(){_this
                .hideTip()});
52        })(this, svg);
53
54        force.on("tick", function(e) {
55          svg.selectAll("path")
56            .data(groups)
57            .attr("d", groupPath);
58          link.attr("x1", function(d) { return d.source.x; })
59            .attr("y1", function(d) { return d.source.y; })
60            .attr("x2", function(d) { return d.target.x; })
61            .attr("y2", function(d) { return d.target.y; });
62          node.attr("transform", moveNode);
63          pivotNode.attr("transform", moveNode);
64        });
65      } else {
66        svg.selectAll(".link")
67          .style("stroke", "#ccc");
68        force.on("tick", function() {
69          link.attr("x1", function(d) { return d.source.x; })
70            .attr("y1", function(d) { return d.source.y; })
71            .attr("x2", function(d) { return d.target.x; })
72            .attr("y2", function(d) { return d.target.y; });
73
74          node.attr("transform", moveNode);
```

```
75        });
76      }
77      force.start();
78      svg.transition()
79        .duration(1000)
80        .style("opacity", 1);
81    };
```

Codice 6.6: Metodo *updateGraph* del disegnatore

Il metodo comincia eliminando gli sfondi e i nodi di pivot precedentemente creati, se vi sono, e salvandosi il nuovo valore di Threshold (righe 5-16). A seconda del grafico scelto esegue il metodo specifico per l'aggiornamento descritto precedentemente (righe 18-25) e ne aggiunge i collegamenti al sistema di forze (riga 23).
Dopo queste aggiunte c'è un ulteriore scelta basata sul tipo di grafico:

Force-based nel caso di questo tipo di grafico, si assegna una grandezza ai collegamenti che altrimenti sarebbero nascosti; infine si assegna una funzione che determini le "future" posizioni dei nodi all'evento "tick", la funzione *moveNodo* la si può osservare nel frammento di codice 6.7.

Cluster-based nel grafico basato sui cluster si creano innanzitutto i nodi di pivot nel grafico (righe 33-37); si procede quindi alla creazione di uno sfondo per ogni cluster usando il tag *path* assegnandogli un colore tramite la funzione *groupFill* e assegnandogli l'estensione tramite la funzione *groupPath* (visibili nel frammento di codice 6.8). Inoltre ad ogni *path* vengono aggiunte le mappature agli eventi *mouseover* e *mouseout* verso le funzioni *showTip* e *hideTip* (spiegate successivamente) per la gestione del popup informativo. Infine, come per l'altro grafico, si assegnano le funzioni per il calcolo delle future posizioni ai collegamenti, nodi, pivot e sfondi.

In conclusione si avvia il sistema di forze e una dissolvenza per far apparire il grafico (righe 77-80).

```
1   var moveNode = function(d) {
2     return "translate(" + Math.max(Math.min(d.x, width -
          rCircle), rCircle) + "," + Math.max(Math.min(d.y,
          height - rCircle), rCircle) + ")";
3   }
```

Codice 6.7: Funzione *moveNode* usata per determinare le future posizioni dei nodi

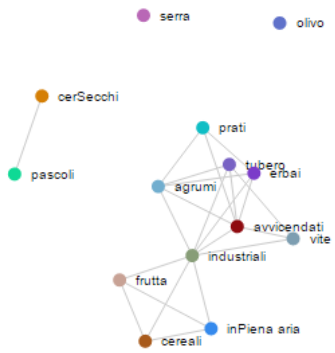

Figura 6.1: Grafico Force-Based sul dataset Agricoltura

Il metodo *moveNode* (visibile nel codice 6.7) impedisce ai nodi di uscire dall'area visibile del grafico tenendo conto anche dell'ampiezza del nodo.
Nel codice 6.8 *groupFill* ritorna un colore, con la possibilità di assegnare fino a 20 colori diversi, usando la funzione *fill* di D3. Mentre *groupPath* ritorna una stringa contenente l'estensione dello sfondo per il relativo cluster.

```
1  var groupFill = function(d, i) { return fill(d.key % 20);
       };
2  var groupPath = function(d) {
3    return "M" +
4      d3.geom.hull(d.values.map(function(i) { return [i.x, i.
         y]; }))
5      .join("L")
6    + "Z";
7  };
```

Codice 6.8: Funzioni *groupFill* e *groupPath* usate per definire colore e ampiezza degli sfondi nei vari cluster

Escludendo il popup, il grafico risultante calcolato sul dataset Agricoltura sarà quello in figura 6.1 per il Force-Based, mentre per il Cluster-Based quello in figura 6.2.

Il popup informativo compare solo quando il mouse si trova sopra un nodo o uno sfondo; lo scopo è mostrare un grafico relativo a quel nodo, oppure di mostrare un grafico di tutti i campioni partecipanti ad un cluster.

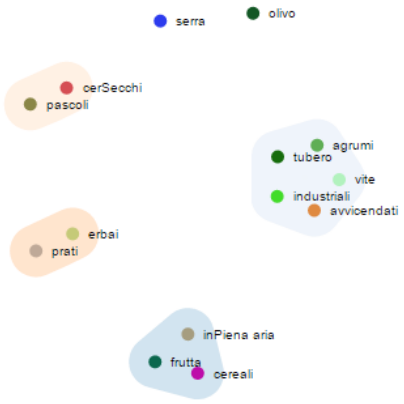

Figura 6.2: Grafico Cluster-Based sul dataset Agricoltura

Il tipo di grafico che crea è un grafico a linee, con le ascisse rappresentanti il

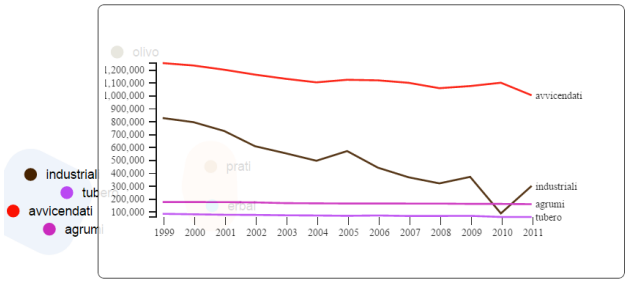

Figura 6.3: Popup su un cluster di un grafico Cluster-Based (dataset Agricoltura)

tempo e le ordinate il valore dei campioni; nel caso in cui il mouse sia sopra un nodo non collegato a nessun altro sarà presente una sola serie, ma se è sopra lo sfondo di un gruppo o sopra un nodo collegato ad un altro verranno mostrate tutte le serie dei nodi colorate secondo il colore di essi (figura 6.3). Per realizzare questo popup si è aggiunto ad ogni nodo e ad ogni sfondo una chiamata ai metodi *showTip, hideTip* al verificarsi degli eventi *mouseover* e *mouseout*.
Mentre *hideTip* applica solo una transizione al contenitore del popup, il metodo *showTip* invece contiene tutti i passaggi per la visualizzazione del grafico (codice 6.9).

```
this.showTip = function(d){
  divOver.transition()
    .duration(300)
    .style("opacity", .9);
  var cluster4Tip = [];
  if(d.values){
    for(var i = 0;i < d.values.length; i++)
      if(d.values[i].index < newSamples.length)
        cluster4Tip.push(JSON.parse(JSON.stringify(
          newSamples[d.values[i].index])));
  } else if(this.options.graphStyle == this.GS_FB) {
    cluster4Tip = [JSON.parse(JSON.stringify(newSamples[d.
      index]))];
    for(var i = 0;i < links.length; i++) {
      if(links[i].source.index == d.index)
        cluster4Tip.push(JSON.parse(JSON.stringify(
          newSamples[links[i].target.index])));
      if(links[i].target.index == d.index)
        cluster4Tip.push(JSON.parse(JSON.stringify(
          newSamples[links[i].source.index])));
    }
  } else {
    cluster4Tip = [JSON.parse(JSON.stringify(newSamples[d.
      index]))];
  }
  if(this.options.norm != 1){
    for(var i = 0; i < cluster4Tip.length; i++){
      cluster4Tip[i].dataset = maClu.normalize(cluster4Tip[
        i].dataset, this.options.norm);
    }
  }

  x = d3.scale.ordinal()
    .domain(xInfo)
    .rangeRoundBands([0, widthLines], .1);
  y = d3.scale.linear()
    .range([heightLines, 0]);
  xAxis = d3.svg.axis()
    .scale(x)
    .orient("bottom");
  yAxis = d3.svg.axis()
    .scale(y)
    .orient("left");
  lineInterpol = d3.svg.line()//interpolatore
    .interpolate("linear")
```

```
40        .x(function(dd, i) { return x(xInfo[i]); })
41        .y(function(dd) { return y(dd); });
42
43      color = function(dd) { return nodeColor[dd.index]; };
44
45      y.domain([
46        d3.min(cluster4Tip, function(c) { return d3.min(c.
              dataset); }),
47        d3.max(cluster4Tip, function(c) { return d3.max(c.
              dataset); })
48      ]);
49
50      divOver.selectAll("svg").remove();
51      svgOver = divOver.append("svg")
52        .attr("width", widthLines + marginLines.left +
              marginLines.right)
53        .attr("height", heightLines + marginLines.top +
              marginLines.bottom)
54        .append("g")
55        .attr("transform", "translate(" + marginLines.left + ",
              " + marginLines.top + ")");
56
57      svgOver.append("g")
58        .attr("class", "yAxis")
59        .call(yAxis)
60        .append("text")
61        .attr("transform", "rotate(-90)")
62        .attr("y", 6)
63        .attr("dy", ".71em")
64        .style("text-anchor", "end");
65
66      svgOver.append("g")
67        .attr("class", "xAxis")
68        .attr("transform", "translate(-10," + heightLines + ")"
              )
69        .call(xAxis);
70
71      var series = svgOver.selectAll(".series")
72        .data(cluster4Tip)
73        .enter().append("g")
74        .attr("class", "series");
75
76      series.append("path")
77        .attr("class", "line")
```

```
78        .attr("d", function(dd) { return lineInterpol(dd.
              dataset); })
79        .transition()
80        .delay(50)
81        .duration(1500)
82        .ease("linear-in-out")
83        .style("stroke", function(dd){return color(dd);})
84        .attrTween("stroke-dasharray", function(d){
85          var l = this.getTotalLength(), j = d3.
                interpolateString("0," + l, l + "," + l);
86          return function(call) { return j(call); };
87        });
88
89      series.append("text")
90        .attr("transform", function(dd) { return "translate(" +
              x(xInfo[xInfo.length-1]) + "," + y(dd.dataset[dd.
              dataset.length-1]) + ")"; })
91        .attr("x", 3)
92        .attr("dy", ".35em")
93        .text(function(dd) { return dd.name; });
94    };
```

Codice 6.9: Funzione *showTip* per la visualizzazione di un popup informativo sopra i nodi e gli sfondi

Il metodo inizia facendo partire un animazione di apertura del contenitore del popup (l'oggetto *divOver*) ed crea un array (*cluster4Tip* conterrà tutti i nodi da visualizzare).
Dalla riga 6 alla riga 20 si gestiscono 3 casi che possono verificarsi:

Cluster-Based Avviene quando si seleziona uno sfondo; si può distinguere dal parametro di input *d* che conterrà una proprietà *values* contenente tutti gli elementi partecipanti al cluster. In questo caso si procede a copiare tutti i campioni dentro l'array *cluster4Tip*, tenendo presente che gli elementi con un id superiore alla dimensione della tabella di pivot (*newSamples*) saranno i nodi di pivot creati.

Force-Based Nella visualizzazione Force-Based, quando si seleziona un nodo viene visualizzato un grafico contenente anche i campioni collegati con esso. Dalla riga 12 alla 17 si scorre quindi tutto l'array *links* contenente i collegamenti, si cercano gli indici degli oggetti collegati con il nodo *d* selezionato e li si inserisce nell'array *cluster4Tip*.

Nodo Costituisce il caso più semplice, si copia l'elemento corrispondente su *newSamples* basandosi sull'indice ricevuto (*d.index*).

Nelle righe 21-25 si normalizzano i cluster nel caso sia stato scelto nell'oggetto *options* secondo la scalatura scelta, quindi si comincia a creare le varie componenti del grafico.
Per prima cosa vengono create delle scale di misura segnalandone il dominio e l'ampiezza (*xInfo* è un array contenente le etichette dei vari anni); poi si creano gli assi basati su queste unità di misura, segnalando che avranno una posizione in basso e una a sinistra del grafico (righe 32-37) e si crea un interpolatore con lo scopo di associare il primo valore dell'array newSamples al primo elemento di xInfo e cosi via.
In riga 43 si è creata una funzione che restituisca il colore del nodo passato in parametro, mentre in riga 45 si è ristretto il dominio delle ordinate rispetto al massimo e il minimo delle serie che si visualizzeranno.
Preparati tutti gli oggetti che compongono il grafico, si precede ad inserirle su di esso: si rimuovono i precedenti grafici selezionando i tag *svg* e chiamando il metodo *remove*, si crea un nuovo oggetto *svg* settandone la dimensione e si aggiungono al grafico i due assi. Con le righe 71-74 si aggiungono invece le serie e le si disegna usando l'interpolatore già definito (righe 76-87). Inoltre viene avviata una animazione (righe 79-81) sul colore definito nella riga 83 e sulla linea (righe 84-87) in modo da vedere una linea che progressivamente avanza e cambia colore.
In conclusione, con le righe dalla 89 alla 93, vengono aggiunti nomi delle serie in prossimità della fine delle serie.

Capitolo 7

Conclusioni

In conclusione, si è visto l'intero progetto di stage, comprendendo anche la prima parte, risultata inappropriata per lo scopo del progetto.
In particolare, per poter comparare due righe di una tabella di pivot risulta poco utile usare un test statistico per la bontà di adattamento per i seguenti motivi:

- Gli assunti dei test vincolano in maniera pesante le tabelle di pivot, infatti dovranno rispettare l'assunto di indipendenza e di campionamento casuale semplice;
- Usando questi test non è possibile scalare i dati in quanto si influirebbe sul risultato del test;
- Non è possibile analizzare campioni con variabili di diverso tipo (dati ordinali e quantitativi).

Al contrario l'analisi gerarchica risulta più efficace e flessibile per questo tipo di studio: scegliendo una distanza adeguata, come l'indice di correlazione di Pearson nel caso di serie storiche, le informazioni che può dare sono di più rispetto a quanto può dare un test.
In particolare si può fornire all'utente l'informazione di gruppi di righe, oltre ad indicare se una determinata riga si comporta in maniera simile ad un'altra.
Il caso delle serie storiche che è stato usato durante il progetto è comunque limitativo, ma usando come misura di dissimilarità l'indice di similarità di gower si può estendere l'analisi anche a tabelle di pivot che hanno colonne eterogenee (ordinali e quantitative), cambiando opportunamente il grafico

del popup che per colonne eterogenee fornirebbe una visualizzazione poco sensata.
Il codice sorgente prodotto, inoltre, permette una facile implementazione del grafico. Per creare un grafico di tipo Cluster-Based basta effettuare i seguenti passaggi (si veda il codice 7.1 come riferimento):

- creare un oggetto *Designer* passando i parametri voluti;
- chiamare il metodo *loadData* passando i dati da visualizzare;
- chiamare il metodo *showGraph*.

Infine, creando questo tipo di grafico e muovendo la barra di threshold, è possibile vedere come le varie righe si raggruppano, permettendo quindi ad un utente senza conoscenze tecnico-scientifiche di tale ambito di effettuare un analisi di questo tipo, che era lo scopo del progetto.

```
1  designer = new Designer("#result", {
2    thresholdBar: document.getElementById("threshold"),
3    graphStyle: Designer.GS_CB,
4    method: Designer.CL_AVERAGE,
5    norm: Designer.NR_NONE,
6    typeDistance: Designer.DS_PEARSON});
7
8  designer.loadData(dataset);
9
10 showGraph();
```

Codice 7.1: Comandi da eseguire per creare un grafico Cluster-Based sulla variabile dataset dentro il *div result*, usando come distanza l'indice di correlazione di Pearson e come metodo agglomerativo l'average linkage

Bibliografia

Michael Alexander Bill Jelen. *Pivot table data crunching*. Que Publishing, 2005.

Mike Bostock. *D3 API Reference*. URL `https://github.com/mbostock/d3/wiki/API-Reference`.

World Wide Web Consortium. *CSS 2.1 standard definition*. a. URL `http://www.w3.org/TR/CSS21/`.

World Wide Web Consortium. *CSS 3 Overview*. b. URL `http://www.w3.org/TR/2001/WD-css3-roadmap-20010523/`.

World Wide Web Consortium. *HTML5 standard definition*. c. URL `http://www.w3.org/TR/html5/`.

World Wide Web Consortium. *Scalable Vector Graphics Reference*. d. URL `http://www.w3.org/TR/SVG/`.

Luigi Fabbris. *Statistica Multivariata*. MCGraw Hill, 2011.

Alfredo Rizzi Mary Fraire. *Analisi dei dati per il Data Mining*. Carocci editore, 2011.

Mozilla Developer Network. *JavaScript API Reference*. URL `https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference`.

David J. Sheskin. *Parametric and Nonparametric Statistical Procedures*. Chapman and Hall/CRC, 2003.

Antero Taivalsaari. Classes vs. prototypes: Some philosophical and historical observation. *Journal of Object-Oriented Programming*, 1996.