
UNIVERSITÀ DI PISA

# Visual Analitics

REPORT

# Infortuni in Toscana 2015

*Michele Tarabella*

**Anno Accademico 2014-2015**

# Indice

# Introduzione

La seguente relazione ha come obiettivo la documentazione del progetto sviluppato per l'esame di Visual Analytics (a.a 2014/2015).
Il progetto presenta, mediante l'utilizzo di layout grafici, la situazione degli infortuni avvenuti in Toscana nel primo semestre 2015.
L'infortunio sul lavoro è un tipo di argomento che viene affrontato in Italia in maniera molto accesa. Un articolo, datato 25 Agosto 2015, del quotidiano socialista *Avanti!* espone che il trend degli infortuni sul lavoro sta decrescendo dal 2014, anno in cui l'Inail presentò ben 663.149 denunce di infortuni (ben il 4,6% in meno rispetto al 2013).[1]
L'Inail mette a disposizione online sul proprio sito banche di open data di ogni tipologia e ovviamente anche quelli riguardanti gli infortuni.
Sembrava interessante quindi utilizzare questi dati per un'analisi statistica degli infortuni avvenuti in Toscana mediante l'utilizzo di tecnologie grafiche presentate durante il corso di studio come la libreria Javascript D3 (Data Driven Documents)[2] per rappresentare i dati attraverso layout accattivanti e coinvolgenti.

# L'idea

L'idea del progetto si doveva anzitutto basare sul dataset a disposizione. La ricerca pertanto dello stesso è stata l'attività principale della prima fase dello sviluppo del progetto
I dati sono stati ricavati dal sito dell'Inail[3] (Istituto Nazionale per l'Assicurazione contro gli Infortuni sul Lavoro) al seguente indirizzo: http://www.inail.it/internet_web/appmanager/internet/home.

## Modello dei dati

Come descritto nella pagina del sito i dati raccolti riguardano gli infortuni avvenuti in Toscana nel 2015 con periodicità di rilevazione mensile.
Gli attributi dei record sugli infortuni nel dataset sono di varie tipologie:

1. **Attributi temporali** (DataRilevazione, DataProtocollo, DataAccadimento, Data Morte);

---

[1] http://www.avantionline.it/2015/08/sicurezza-inail-infortuni-in-calo/#.Vd8YQLztmkp

[2] http://d3js.org/

[3] http://www.inail.it/internet_web/appmanager/internet/home

2. **Attributi spaziali** (LuogoAccadimento, cioè la provincia dove si è verificato l'infortunio);
3. **Attributi sul genere e l'età**;
4. **Attributi sul luogo di nascita** (nazione di provenienza dell'infortunato, viene utilizzato il codice Belfiore della nazione o "ITAL" per l'Italia);
5. **Attributi sulla modalità di accadimento** (se l'infortunio si è verificato "in occasione di lavoro" o "in itinere" cioè se si è verificato sul normale percorso di andata e ritorno dall'abitazione al posto di lavoro;
6. **ConSenzaMezzoDiTrasporto** (variabile booleana, indica l'eventuale coinvolgimento di un mezzo di trasporto nell'infortunio);
7. **Altri**;

## Idee sul design

Il dataset sugli infortuni in Toscana 2015 contiene, purtroppo, una enorme quantità di dati (si contano quasi 26.000 record per il solo primo semestre 2015), pertanto la progettazione del data design doveva operare un gestione che sapesse contenere la grande dimensione del dataset.
Ho pensato che utilizzare un cluster layout (layout a grappoli) avesse contenuto fortemente le grandi dimensioni del dataset, pertanto ho scelto di rappresentare i dati utilizzando questo tipo di layout dopo aver individuato un pattern accattivante: *Cluster Force Layout III* [4] sulla libreria dei *mbostock's blocks* disponibile online.[5] Mi interessava infatti un design sintetico dei dati che presentasse una componente interattiva dell'interfaccia, interattività che però dipendesse sempre dalla quantità dei dati in questione. Posso dire che il *Cluster Force Layout III* sintetizza bene questo insieme di concetti sulla gestione, sintesi e interattività sui dati.
La seconda idea prendeva in esame l'elemento del tempo: dal momento che i record dei dati dispongono di attributi temporali ho pensato di visualizzare la distribuzione degli infortuni per età tramite un semplice *scatter plot*. Inizialmente l'idea era quella di utilizzare il *timeseries layout* della MLVL[6], abbandonata poi per problemi di *cross-browsing*. Anche in questo caso si è fatto ricorso allo *scatter plot* disponibile dai *blocks* di Bostock.

---

[4] http://bl.ocks.org/mbostock/7881887

[5] http://bl.ocks.org/mbostock

[6] http://mlvl.github.io/timeseries/

## Pulizia dei dati

In base alle idee precedentemente sviluppate, sono passato alla pulizia del dataset, estraendo solo quei dati che mi sarebbero serviti per il suddetto design.
Inizialmente, avendo bisogno dei dati riguardanti il primo semestre del 2015 ho selezionato un attributo temporale che presentasse la data dell'infortunio. Ho scelto l'attributo DataProtocollo perché volevo che l'infortunio fosse stato realmente registrato ufficialmente dagli uffici dell'Inail, e ho filtrato tutti i dati del 2015 su questa condizione.
Di questa selezione ho poi estratto la lista dei record con attributo "Età", "Genere" e "Luogo Accadimento" e ho copiato tutto in un nuovo database.
Il nuovo database conteneva record sugli infortuni con i soli attributi: "DataProtocollo", "Età", "Genere" e "LuogoAccadimento". I primi due attributi sono stati utilizzati nello *scatter plot* come visualizzazione della distribuzione temporale dei dati per età, i secondi due nel *Cluster Force Layout* come visualizzazione della quantità degli infortuni nelle province per genere (ogni genere, un cluster).

# Stato dell'arte

Non posso certo affermare che sia lo *Scatter Plot* e il *Cluster Force Layout* siano idee originali come *data design* e infatti lo state dell'arte di entrambi è composto da una grande quantità di documentazione sull'uso che ne è stato fatto.
Uno *Scatter Plot* è un semplice grafico a dispersione in cui i dati sono visualizzati come una collezione di punti su d un grafico cartesiano; ho pensato ad ogni modo che se Mike Bostock ne ha pubblicata una sua versione in D3 rimane sempre un design semplice di visualizzazione anche temporale dei dati.
Il *Cluster diagram* è un modo di rappresentare un gruppo di oggetti che presentano attributi di valori molto simili tra loro; è molto utilizzato nella rappresentazione di *network,* rappresentazione di galassie, alberi di dati...
Il Cluster Force Layout è un nuovo tipo di diagramma a grappoli offerto dalla libreria dei *blocks* di Bostock in cui si può interagire con i nodi spostandoli sul canvas nella schermata. La forza di spostamento del grappolo è direttamente proporzionale alla dimensione di ogni oggetto; questa è una feature interattiva che permette una comprensione immediata della dimensione dei dati (informazione già fornita comunque e graficamente dalla dimensione dell'oggetto del *cluster*).

Il concetto di *dynamic graphs* è espresso in un *paper* dal titolo *The state of the art in visualizing dynamic graphs*[7]. "The characteristic difference of a dynamic graph to a static graph is that the structure of the vertices and edges as well as their attributes can change over time. The Figure shows an illustrating example of a dynamic graph and its visualization: a directed graph consisting of four nodes is visualized over three time steps as juxtaposed node-link diagrams. The position of the nodes is the same for all diagrams, which makes it easier to track the nodes over time".
Della categoria di *dynamic graphs* fanno parte quindi i *force-directed-graph* di cui il *Cluster Force Layout* ne è un esempio.

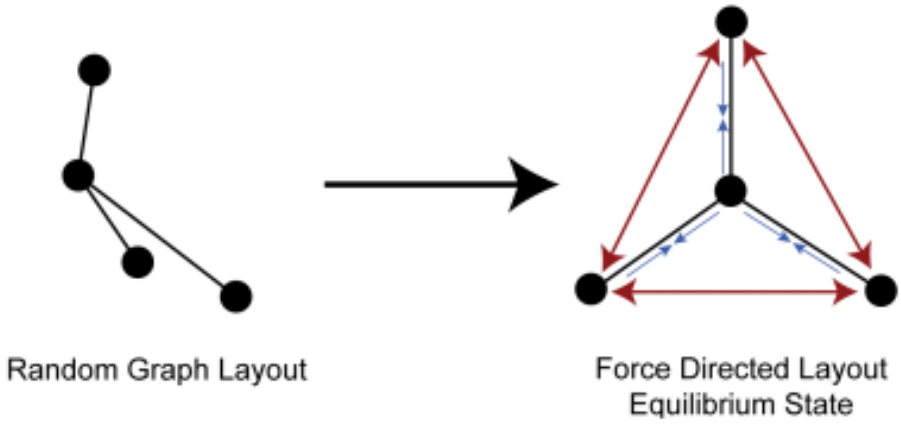


*Figure: Getting the force in the layout*

I *force-directed layout* vengono utilizzati per rappresentare, ad esempio, reti sociali, in particolare le connessioni di ogni nodo: la forza di spostamento del nodo dipende da quante connessioni dipendono da quel nodo. Un esempio applicativo è stato realizzato al CNR di Pisa, un team di ricercatori ha realizzato una rete di nodi di utenti di Tweet, dove i nodi con più *follower* avevano una forza di spostamento, del sottografico. maggiore, rispetto a nodi con meno connessioni.

# Data design

L'idea del progetto si basa su due tipi di analisi del dataset da cui derivano due tipologie di interfacce di presenzione dello stesso: una statica e una dinamica.
Queste interfacce offrono due tipologie di visualizzazioni dei dati: una visualizzazione temporale-distributiva e una visualizzazione quantitativa.
Ho scelto questo dualismo per offrire all'utente la possibilità di cambiare modalità di visualizzazione dei dati all'occorrenza.
Per la parte statica ho rappresentato i dati degli infortuni in Toscana nel mese di Giugno 2015 su un

[7] http://www.vis.uni-stuttgart.de/uploads/tx_vispublications/eurovis14-star.pdf

grafico a dispersione (*scatterplot*).

Lo scatteplot di Bostock in questione è stato modificato nella sua parte visiva aggiungendo tre feature principali:

1) colore (per rappresentare i giorni della settimana, e quindi individuare le settimane);

2) opacità (per rappresentare la concentrazioni di infortuni nel giorno);

3) tooltip (per informazioni su singolo record con etichetta);

Ho deciso di omettere l'asse delle ascisse perché ho pensato che il colore in sé potesse già suddividere i giorni della settimana e quindi di rappresentare graficamente l'andamento del tempo.

L'opacità invece avrebbe svolto un ruolo di quantità e quindi di distribuzione dei dati per età.

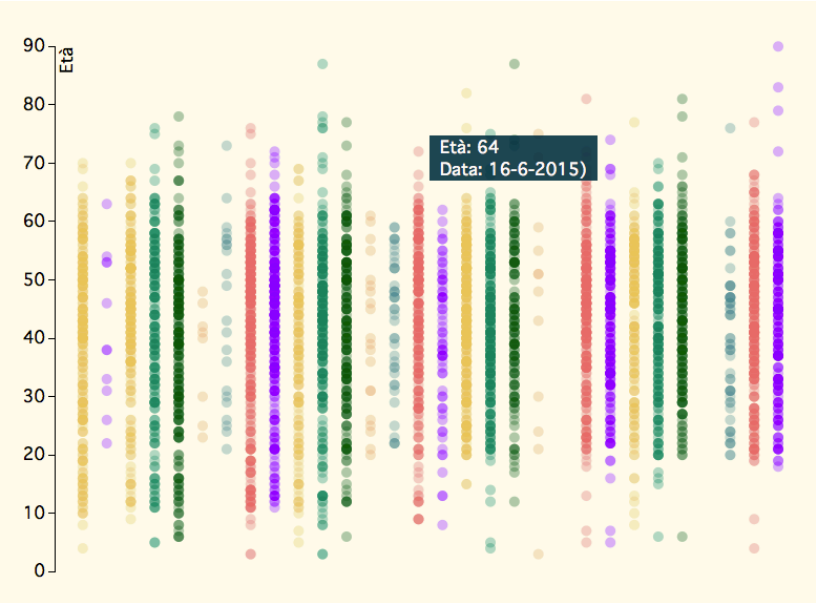


*Figura 2: scatterplot (DataInfortunio, Età)*

Per il *dynamic-graph*, vale a dire il *Clusterd force Layout* ho raccolto i dati riguardanti il genere e la provincia toscana dove è avvenuto l'infortunio.

L'idea era di realizzare due "grappoli" di dati (uno per ogni genere, maschile e femminile) in cui ogni elemento del grappolo (un cerchio) rappresentava una provincia e il raggio di ogni pallina era dipendente dalla quantità degli infortuni in quella provincia.

Dal momento che le province Toscane possono essere considerate come elementi che contengono attributi di valori vicini tra di loro, ho pensato che un cluster layout fosse un buon modo di rappresentare la situazione, in questo caso, degli infortuni.

Due grappoli di due colori distinti (per rappresentare i generi)

In questo caso gli elementi su cui ho lavorato maggiormente sono stati:

1) elaborazione dei dati in un array di oggetti;

2) creazione dell'oggetto nodo (con particolare attenzione all'attributo del *radius* che doveva dipendere dalla quantità degli infortuni);
3) colori e altre variabili dimensionali;

Il concetto di *cluster-force* ("forza del grappolo") serve per conferire una caratteristica dinamica e interattiva al grafico e quindi di comprensione dei dati stessi. Se sposto un nodo di piccole dimensioni, questo avrà una forza di spostamento dell'intero cluster minore rispetto a un nodo di maggiori dimensioni dello stesso cluster. Quindi la forza di spostamento è proporzionale alla dimensione del nodo.

Questo è un caso di applicazione pratica di una caratteristica tipica dei *dynamic-graphs* descritti nella sezione dello stato dell'arte.

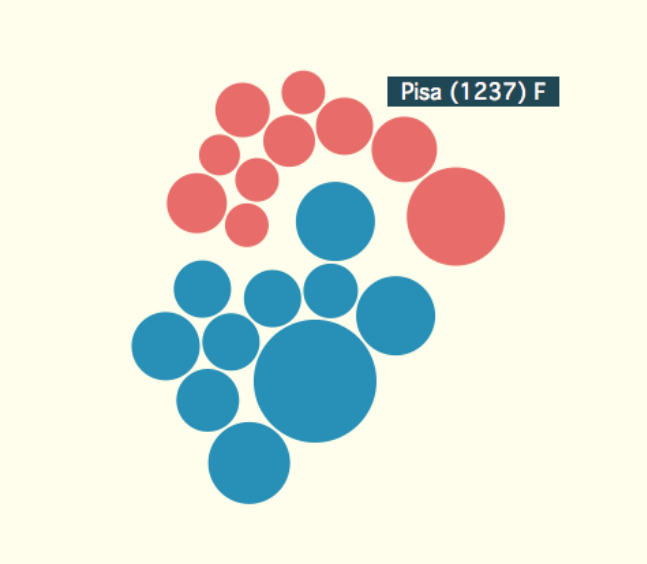


*Figura 3: Cluster con etichetta su nodo della provincia di Pisa per dati riguardanti infortuni di genere femminile*

## Colori

I colori utilizzati nel progetto (sia per la struttura HTML e CSS che per colorare i grafici) sono stati prelevati dal sito *Palletton.com*.

Il sito offre un interfaccia in cui è possibile selezionare tre colori e di ricavarne tutte le sfumature associate.

La cosa interessante è che per ogni paletta customizzata viene resa disponile la sua versione in file css con colori espressi in formato esadecimale.

Questo mi ha reso molto semplice il copia incolla durante la fase di progettazione del sito mantenendo una coerenza nel design dal punto di vista dei colori.

# Struttura del progetto

I grafici sono contenuti dentro a una classica struttura di pagine HTML collegate tra di loro come mostrato in figura:

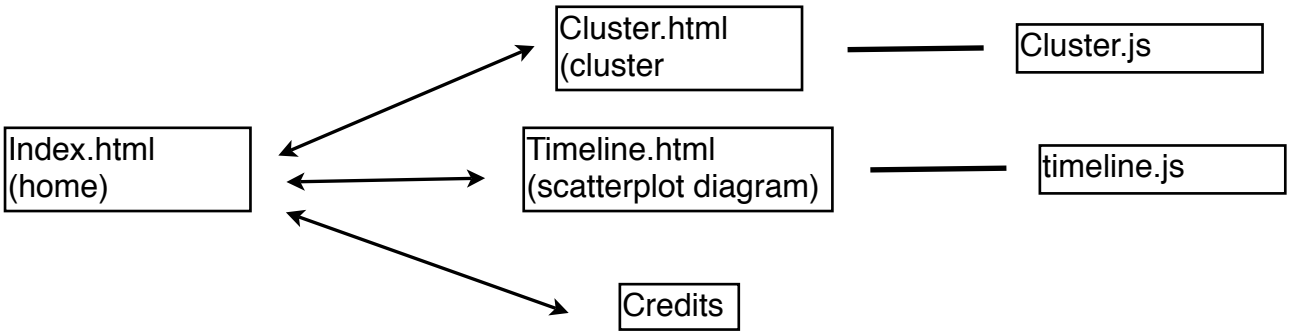


Figura 4: struttura delle pagine HTML

Nelle pagine contenenti i grafici, cioè il cluster diagram e lo scatterplot layout sono stati creati due elementi  appositi a cui poi, nel file Javascript tramite funzioni di libreria, verranno “appesi” dei canvas SVG (*Scalable Vector Graphics*) per il rendering di entrambi i grafici.
Nel folder del progetto quindi sono contenute due programmi in Javascript ognuno dedicato al diagramma di cui vogliamo realizzarne il rendering.
I programmi in Javascript si occupano principalmente di tre cose:
1) lettura del database in formato CSV;
2) elaborazione dei dati e di trasformazione degli stessi in strutture dati come array di oggetti;
3) rendering del grafico;

# I programmi

I programmi contenuti nel folder *script* sono pertanto due, uno per ogni grafico: *cluster.js* e *timeline.js*

## *cluster.js*

Questo programma, dedicato alla realizzazione del diagramma cluster-force layout (o diagramma a grappoli) è suddiviso in varie sezioni:
1) metodi getter, per recuperare in forma di array i dati;

2) metodi createNodes, per realizzare la struttura dati contenente i nodi di entrambi i grappoli;

3) parsing del database;

4) funzioni per la gestione della forza dei nodi;

## metodi getter

Quattro funzioni principali: getGender, getProv, getProvObj, getProvLen. Vediamone una ad una: Dato un array di oggetti (nel nostro caso, fornito dalla funzione di parsing di D3, d3.csv) restituisce un sottoarray filtrato per genere. Questa prima operazione è fondamentale perché si doveva fare un filtraggio dei dati per genere, dal momento che la prima suddivisione è fatta su due cluster principali:

```
function getGender(arr, gend)
```

Questa seconda funzione invece filtra l'array di oggetti principale in base alla provincia, al luogo di accadimento dell'infortunio.

```
function getProv(arr, prov)
```

La terza funzione è la composizione delle prime due: dato un array filtrato per genere, *getProvObj* ritorna un array di oggetti contenenti per ogni provincia il numero di infortuni per genere (dal momento che l'array in input è già filtrato per genere).

```
function getProvObj(arrG, g)
```

Infine, l'ultima funzione: dato un array di oggetti, ritorna un array contenente solo l'attributo dei numeri degli infortuni per ogni provincia; questo dato è fondamentale per poi elaborare il raggio di ogni nodo: `function getProvLen(arrPObj).`

## metodi createNodes

Il metodo più importante nella creazione del cluster. In questo metodo viene definito il raggio in funzione della quantità di infortuni:

```
function createCluster(arr)
{
  var nodeArr=[];
  for(key in arr)
  {
    var i = 0;
    var r = Math.floor((Math.sqrt(arr[key].Inf))*0.7);
```

La funzione prende in input un array di oggetti provincia filtrati sul genere (forniti dalla funzione *getProvObj(arr, gender)* e ritorna un array di oggetti nodo (un cluster appunto) in cui è incluso fra gli altri anche l'attributo *radius* dell'oggetto nodo dipendente dal numero degli infortuni avvenuti in quella provincia.
Gli altri attributi di ogni oggetto che viene creato sono:

***cluster***: il numero del cluster (0 maschio, 1 femmina)

***radius***: il raggio di ogni nodo

***tooltip***: etichetta per informazioni a comparsa con evento *mouseover*

***quantity***: numero degli infortuni per provincia

***gender***: M of F

***x, y***: posizione random dei nodi fluttuanti sul canvas SVG

Le funzioni di rendering vengono realizzate tramite funzioni di libreria di D3 insieme ad altre funzioni di gestione della forza di attrazione fra i nodi disponibili dal layout stesso di Bostock da cui ho preso spunto modificandolo secondo le mie esigenze.

## *timechart.js*

Questo programma, dedicato alla realizzazione del diagramma *scatterplot layout* (o diagramma a dispersione) è suddiviso in varie sezioni:
1) elaborazione dei dati
2) dimensioni (canvas e assi)
3) color scales

Dovendo lavorare con due tipologie di dati (età e dataInfortunio, quindi un numero e un oggetto data) la parte relativa all'elaborazione dei dati consiste nel leggere il file CSV (dei dati riguardanti il mese di Giugno 2015) e di cambiare il formato della data da *string* a oggetto *Date()*.
Una volta creato l'array di oggetti degli infortuni per età e dataInfortunio, e dopo avere definito le dimensioni del canvas e l'orientamento degli assi si è passati alla definizione della color scale: sono stati scelti sette colori per evidenziare le settimane del mese di Giugno.

```
var color = d3.scale.ordinal()
              .domain(d3.range(7))
```

```
            .range(["#EC6C6C", "#8F00FF","#ECC46C", "#00845F",
"#005B1C", "#ECA86C", "#438990"]);
```

Altre feature come l'opacità e il tooltip (etichetta di informazioni per ogni dato) sono state modificate durante la fase di rendering del grafico.

# Tecnologie utilizzate

Per realizzare il progetto ho fatto uso della triade tecnologica per sviluppo web: HTML5/CSS3 e Javascript.
In particolare, ed era uno degli obiettivi del corso, ho fatto uso delle librerie di D3 (Data Driven Documents) per la realizzazione dei grafici e mi sono ispirato a determinati layout che secondo me poteva soddisfare l'idea di progetto iniziale.
Altra tecnologia importante è stato il browser, Firefox, che tramite continue funzioni di *console.log()* mi ha permesso un debug chiaro ed efficiente nel comunicare ogni tipo di errore.
Infine, il supporto della comunità di Internet, sia la documentazione stessa di D3 che i forum dedicati alla programmazione in generale (nel mio caso funzioni di libreria di Javascript.

# Conclusioni

Ho acquisito, durante questa esperienza di programmazione, maggiore sicurezza e maggiori conoscenze su Javascript in particolare e sul pensiero algoritmico in generale. Ho imparato a utilizzare D3 a lezione e ho integrato con svariati tutorial disponibili online su YouTube.
Durante le lezioni del corso avevo inoltre fatto esperienza con un altro tipo di linguaggio dedicato al rendering di oggetti grafici, *Processing*, e questo mi ha senz'altro dato motivazione e conoscenze per affrontare D3.
Mi ritengo pienamente soddisfatto del lavoro svolto, delle conoscenze che ho acquisito, ma soprattutto della motivazione e della sicurezza sempre maggiori in questo settore dell'informatica.

***Michele Tarabella***