Threshold-Free Cluster Enhancement

Luigi Giugliano[1], Marco Mecchia[1]

[1]Universitá degli studi di Salerno

24 giugno 2016

Overview

Overview

Mappa statistica associata ad un esperimento FMRI

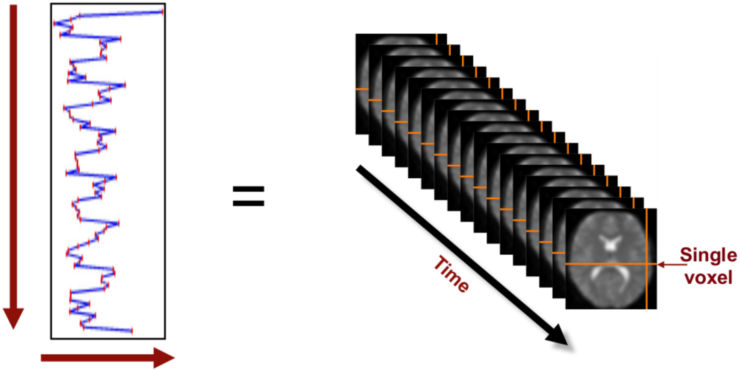
Mappa statistica

Per un dato esperimento di *Risonanza Magnetica Funzionale (FMRI)*, una mappa statistica é un immagine in cui ad ogni voxel corrisponde un valore statistico.

- Solitamente, tali valori rappresentano la *significativitá statistica* di attivazioni neuronali avvenute durante l'esperimento.
- Le attivazioni vengono stimate attraverso il GLM, su cui viene fatta *inferenza* per ottenere i valori statistici.

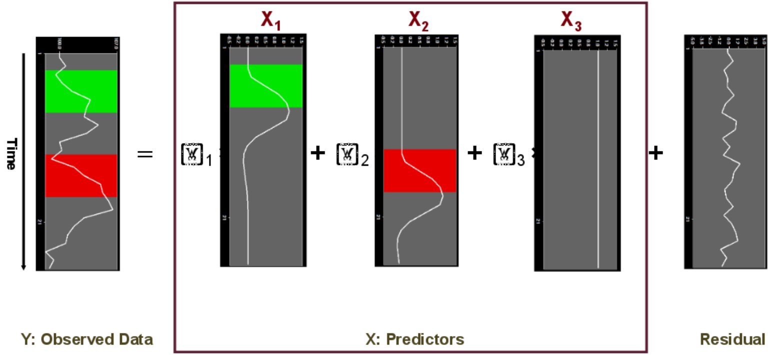
GLM - Aspetti teorici (1/2)

Osservando l'intensitá dei voxel nel tempo, si ottiene il vettore delle osservazioni.



GLM - Aspetti teorici (2/2)

I predittori descrivono la risposta emodinamica *standard* agli stimoli ricevuti durante l'esperimento.



GLM - Riepilogo

Tenendo conto degli aspetti appena visti, il modello completo del GLM diventa:

$$Y = \beta X + \epsilon$$

dove:

- Y é il vettore $N \times 1$ dei dati osservati.
- X é la matrice $N \times r$ a rango pieno dei predittori.
- β é il vettore $r \times 1$ dei coefficienti di regressione.
- ϵ é il vettore $N \times 1$ degli errori casuali.

GLM - Stima dei β

Il numero dei parametri spesso é $<<$ del numero di data point, per cui tra le infinite soluzioni del sistema si sceglie quella che minimizza l'errore residuale, cioé la stima dei minimi quadrati:

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

Nei casi in cui $(X^T X)$ risulti non invertibile, si utilizza la *Pseudoinversa di Monroe*.

GLM - Inferenza statistica(1/2)

- Per poter effettuare inferenza statistica sui valori restituiti dal GLM, occorre stimare la varianza del residuale.
- Si suppone che il rumore abbia una distribuzione *gaussiana*, per cui é possibile stimare la varianza tramite la *distribuzione chi quadrato*:

$$\hat{\sigma}^2 = \frac{\epsilon^T \epsilon}{J - p} \sim \sigma^2 \frac{\chi^2_{J-p}}{J - p}$$

dove:

- ϵ é il rumore.
- J é il numero di data points.
- $p = rank(X)$ é il numero di parametri indipendenti introdotto.
- $J - p$ rappresenta il numero di gradi di libertá del GLM.

GLM - Inferenza statistica(2/2)

Se la matrice X é a rango pieno allora:

$$\hat{\beta} \sim \mathcal{N}(\beta, \sigma^2 (X^T X)^{-1})$$

Segue che qualunque combinazione lineare dei β, cioé un contrasto statistico segue la stessa distribuzione:

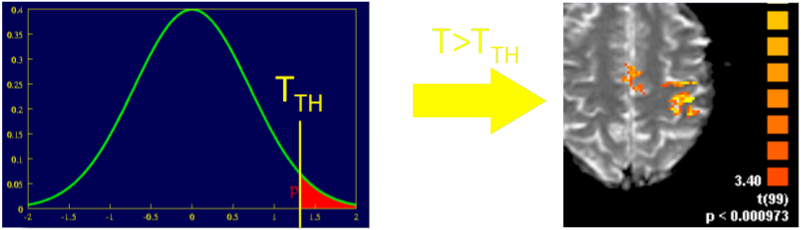
$$c^T \hat{\beta} \sim \mathcal{N}(c^T \beta, c^T \sigma^2 (X^T X)^{-1} c)$$

Pertanto, dopo la stima, si calcola direttamente il parametro T:

$$T = \frac{c^T \hat{\beta} - d}{\sqrt{\hat{\sigma}^2 c^T (X^T X)^{-1} c}}$$

Grafico Inferenza

$$p = \int_{T_{TH}}^{\infty} t(T, v)\,dT$$



Sogliatura delle immagini statistiche

Sogliatura statistica

Generalmente, in statistica la sogliatura é un processo che permette di visualizzare i risultati di un esperimento maggiori di una soglia scelta.

Una soglia ben scelta consente di visualizzare solo i risultati piú significativi di un esperimento, eliminando in parte il rumore.

Spatial information enhancing

Lo spatial information enhancing é una tecnica particolarmente utile per la sogliatura di mappe statistiche derivate da FMRI:

- le informazioni spaziali vengono usate per aumentare l'autenticitá di estese aree di segnale.
- le regioni del segnale sono infatti piú estese del rumore e quindi trovare tali zone aumenta la possibilitá che esse siano segnale e non artefatti.

Cluster-based Thresholding

Il cluster-based thresholding é l'approccio piú comune in neuroimaging:

- Consiste nel visualizzare solo i voxel che fanno parte di aree la cui estensione é $\geq$ di una soglia fissata.

Problemi:

- Necessitá di definire una soglia di clustering.
- Sogliatura di tipo *hard*.
- Difficoltá nel riconoscimento di eventuali *subcluster*.

Overview

TFCE

TFCE tenta di superare i problemi degli approcci precedenti.

- Input: Una mappa statistica di qualsiasi tipo(T, Z, F).
- Output: Una mappa statistica in cui il valore di ogni voxel é un punteggio che rappresenta il contributo spaziale del cluster di cui fa parte.

- Clustering dell'immagine intrinseco.

Assegnazione dei punteggi (1/2)

Il punteggio del voxel p viene stabilito dalla seguente formula:

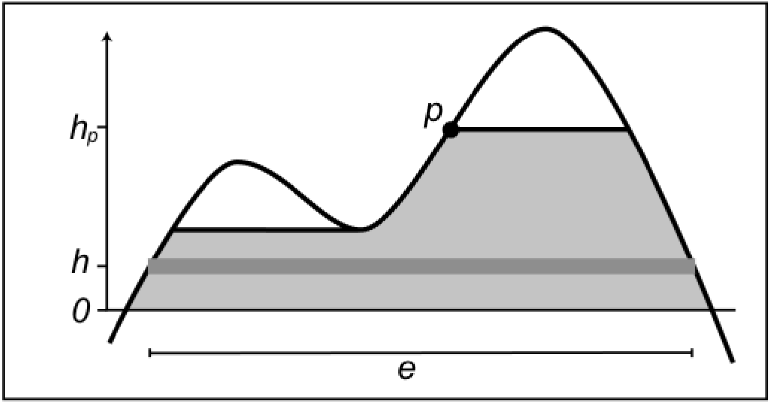
$$TFCE(p) = \int_{h=h_0}^{h_p} e(h)^E h^H dh$$

dove:

- h_p é il valore statistico del voxel p.
- $e(h)$ é l'area del cluster ad altezza h.
- E ed H sono costanti.

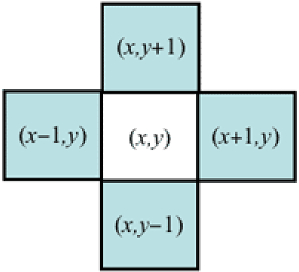
Questo integrale viene calcolato approssimandolo con una sommatoria ponendo $dh = 0.1$.

Assegnazione dei punteggi (2/2)

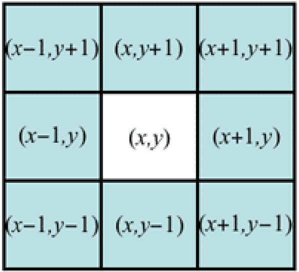


Calcolo estensione cluster (1/2)

- Il calcolo dell'estensione del cluster nel caso di immagini tridimensionali risulta essere piú complesso.
- Occorre controllare il vicinato di ogni voxel in base alla 26 connectivity.



4-neighbourhood



8-neighbourhood

Calcolo estensione cluster (2/2)

La nostra implementazione consiste in un semplice algoritmo:

1. Viene generata, a partire dall'immagine statistica, una mappa binaria in base alla soglia *h* corrente.

Calcolo estensione cluster (2/2)

La nostra implementazione consiste in un semplice algoritmo:

1. Viene generata, a partire dall'immagine statistica, una mappa binaria in base alla soglia h corrente.
2. Una visita in ampiezza della mappa binaria etichetta tutti i cluster.

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & 0 & 3 & 3 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 2 & 2 \end{bmatrix}$$

Calcolo estensione cluster (2/2)

La nostra implementazione consiste in un semplice algoritmo:

1. Viene generata, a partire dall'immagine statistica, una mappa binaria in base alla soglia h corrente.
2. Una visita in ampiezza della mappa binaria etichetta tutti i cluster.

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & 0 & 3 & 3 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 2 & 2 \end{bmatrix}$$

3. Per ogni voxel della mappa, il valore $e(h)$ é il numero di elementi presenti nel cluster di cui fa parte.

$$\begin{bmatrix} 1 & 1 & 0 & 3 & 3 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 2 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 5 & 5 & 0 & 2 & 2 \\ 5 & 5 & 0 & 0 & 0 \\ 5 & 0 & 0 & 2 & 2 \end{bmatrix}$$

Scelta dei parametri E ed H (1/2)

Ricordando la formula di assegnazione dei punteggi:

$$TFCE(p) = \int_{h=h_0}^{h_p} e(h)^E h^H dh$$

- La scelta dei parametri E ed H risulta essere cruciale per avere dei punteggi coerenti.
- Tali parametri sono stati scelti in modo da adattarsi ad un ampio set di segnali e rumore.

Scelta dei parametri E ed H (2/2)

La scelta finale é stata $H = 2$ ed $E = 0.5$ poiché:

- Scegliere $H > 1$ ha il risultato di far si che gli score scalino piú che linearmente con l' “altezza” dei cluster.
  - Ció é desirabile in quanto vengono favoriti cluster di intensitá molto alta rispetto a quelli con intensitá piú bassa.
- Scegliere $E < 1$ fa si che il risultati scali meno che linearmente con la “larghezza” dei cluster.
  - Ció é desirabile specie con h molto basso poiché é probabile che ci siano pochi cluster di dimensioni molto grandi, che forniscono poca informazione spaziale.

Sogliatura esplicita

- L'algoritmo TFCE produce un'immagine con gli score calcolati sull'immagine originale.

Sogliatura esplicita

- L'algoritmo TFCE produce un'immagine con gli score calcolati sull'immagine originale.
- Tuttavia, l'immagine prodotta manca di *valenza statistica*.

Sogliatura esplicita

- L'algoritmo TFCE produce un'immagine con gli score calcolati sull'immagine originale.
- Tuttavia, l'immagine prodotta manca di *valenza statistica*.
- Per calcolare la valenza statistica dell'immagine degli score, occorre calcolare i p-value per ogni voxel.

Sogliatura esplicita

- L'algoritmo TFCE produce un'immagine con gli score calcolati sull'immagine originale.
- Tuttavia, l'immagine prodotta manca di *valenza statistica*.
- Per calcolare la valenza statistica dell'immagine degli score, occorre calcolare i p-value per ogni voxel.
- Essendo TFCE una particolare statistica che sfrutta informazioni spaziali, é possibile calcolare i p-value effettuando il test delle permutazioni sull'esperimento originale.

Test delle permutazioni applicato al GLM single study(1/2)

- Il GLM asserisce che il comportamento di ogni voxel puó essere descritto dalla legge $Y = \beta X + \epsilon$.

Test delle permutazioni applicato al GLM single study(1/2)

- Il GLM asserisce che il comportamento di ogni voxel puó essere descritto dalla legge $Y = \beta X + \epsilon$.
- Sotto l'ipotesi nulla $\mathcal{H}_0 : \beta = 0 \implies Y = \epsilon$, cioé i dati sono puro rumore.

Test delle permutazioni applicato al GLM single study(1/2)

- Il GLM asserisce che il comportamento di ogni voxel puó essere descritto dalla legge $Y = \beta X + \epsilon$.
- Sotto l'ipotesi nulla $\mathcal{H}_0 : \beta = 0 \implies Y = \epsilon$, cioé i dati sono puro rumore.
- Sotto questa ipotesi, i voxel osservati Y possono essere quindi permutati.

Test delle permutazioni applicato al GLM single study(1/2)

- Il GLM asserisce che il comportamento di ogni voxel puó essere descritto dalla legge $Y = \beta X + \epsilon$.
- Sotto l'ipotesi nulla $\mathcal{H}_0 : \beta = 0 \implies Y = \epsilon$, cioé i dati sono puro rumore.
- Sotto questa ipotesi, i voxel osservati Y possono essere quindi permutati.
- In caso di assenza di variabili di disturbo, permutare i predittori e' equivalente:

$$PY = X\beta + \epsilon \iff Y = P'X\beta + P'\epsilon$$

Test delle permutazioni applicato al GLM single study(2/2)

- Data la *permutabilitá* garantita dall'ipotesi nulla, i dati osservati possono derivare in maniera equiprobabile da qualsiasi condizione sperimentale.

Test delle permutazioni applicato al GLM single study(2/2)

- Data la *permutabilitá* garantita dall'ipotesi nulla, i dati osservati possono derivare in maniera equiprobabile da qualsiasi condizione sperimentale.
- Ció vale anche per la statistica calcolata a partire dai dati.

Test delle permutazioni applicato al GLM single study(2/2)

- Data la *permutabilitá* garantita dall'ipotesi nulla, i dati osservati possono derivare in maniera equiprobabile da qualsiasi condizione sperimentale.
- Ció vale anche per la statistica calcolata a partire dai dati.
- La distribuzione delle permutazioni é quindi l'insieme delle statistiche calcolate dalle possibili permutazioni dei dati di partenza.

Test delle permutazioni applicato al GLM single study(2/2)

- Data la *permutabilitá* garantita dall'ipotesi nulla, i dati osservati possono derivare in maniera equiprobabile da qualsiasi condizione sperimentale.
- Ció vale anche per la statistica calcolata a partire dai dati.
- La distribuzione delle permutazioni é quindi l'insieme delle statistiche calcolate dalle possibili permutazioni dei dati di partenza.
- Tale distribuzione ci consente di formalizzare la probabilitá di un certo risultato.

P-value

Il P-value é la proporzione dei valori statistici nella distribuzione delle permutazioni che sono maggiori o uguali del valore osservato nell'esperimento.

Test delle permutazioni applicato a TFCE(1/2)

La procedura per applicare il test delle permutazioni all'algoritmo TFCE, tenendo conto di quanto visto, é la seguente:

1. Calcolare gli score sull'immagine originale.
2. Trovare lo score massimo.
3. Permutare i predittori e ricalcolare i β e la statistica desiderata.
4. Trovare lo score massimo ottenuto nella permutazione e memorizzarlo.
5. Ripetere dal punto 3 fino a raggiungere il numero di permutazioni desiderato.

Ció ci restituisce la distribuzione T_{max}.

Test delle permutazioni applicato a TFCE(2/2)

Data la distribuzione T_{max}:

- La soglia critica é pari all'elemento $c+1$ piú grande della distribuzione, con $c = \lfloor \alpha N \rfloor$, dove α é il livello di significativitá desiderato e N é il numero delle permutazioni.
- I voxel con statistiche che superano tale soglia sono quelli che hanno evidenza statistica al piú pari ad α.

Il p-value di ogni voxel v é dato da:

$$\text{p-value}(v) = \frac{\#\text{Elementi di}\, T_{max} > T(v)}{T(v)}$$

Dove $T(v)$ é il valore della statistica nell'immagine originale.

Test delle permutazioni - Casi particolari

Il semplice modello appena visto puó essere esteso a diversi casi:

- Il modello di Freedman-Lane tiene conto delle variabili di disturbo, ed é quello utilizzato dall'algoritmo *Randomise*.
- Negli studi di gruppo, il caso é ancora piú semplice: i blocchi di permutazione sono definiti dai pazienti.

Per ulteriori approfondimenti consultare:

Permutation inference for the general linear model

Overview

Suddivisione del codice

I file principali che compongono il plugin sono:

- Tfce.cpp
- Utilities.cpp

Tfce É il core del plugin, dove avviene il calcolo dei punteggi.

Utilities contiene tutte le funzioni di supporto.

Funzioni pubbliche (1/3)

L'unica funzione che viene esposta dal file **Tfce.h** é:

```
float * tfce_score(float * map, int dim_x, int dim_y,
    int dim_z, float E, float H, float dh);
```

Funzioni pubbliche (2/3)

Le funzioni che espone **Utilities.h** sono:

```
void findMinMax(float *map, int n, float *min, float
    *max, float * range);

int * getBinaryVector(float * map, int n, int
    (*confront)(float, float), float value, int *
    numOfElementsMatching);
```

Funzioni pubbliche (3/3)

```
float * fromBinaryToRealVector(float * map, int n, int
    * binaryVector);

float * fill0(int n);

void apply_function(float * vector, int n, float (*
    operation) (float a, float b), float argument);

int linearIndexFromCoordinate(int x, int y, int z, int
    max_x, int max_y);

void coordinatesFromLinearIndex(int index, int max_x,
    int max_y, int * x, int * y, int * z);

float * copyAndConvertIntVector(int * vector, int n);
```

Funzione tfce score

```
float * tfce_score(float * map, int dim_x, int dim_y,
    int dim_z, float E, float H, float dh){
  findMinMax(map, n, &minData, &maxData, &rangeData);
  precision = rangeData/dh;
  if (precision > 200) {
    increment = rangeData/200;
  } else{
    increment = rangeData/precision;
  }
  steps = ceil(rangeData / (increment));
  #pragma omp parallel for
  for (i = 0; i < steps; i++) {
      computeTfceIteration(minData + i*increment, map,
          n, dim_x, dim_y, dim_z, E, H, increment,
          toReturn);
  }
  return toReturn;
}
```

Funzione computeTfceIteration (1/3)

```
void computeTfceIteration(float h, float * map, int n,
    int dim_x, int dim_y, int dim_z, float E, float H,
    float dh, float * toReturn){
  int * indexMatchingData = getBinaryVector(map, n,
      moreThan, h, &numOfElementsMatching);
  clustered_map = find_clusters_3D(indexMatchingData,
      dim_x, dim_y, dim_z, n, &num_clusters);
  extent_map = new int[n];
  for (j = 0; j < n; ++j){
    extent_map[j] = 0;
  }
  delete [] indexMatchingData;
```

Funzione computeTfceIteration (2/3)

```
for (i = 1; i <= num_clusters; ++i) {
    numOfElementsMatching = 0;
    for (j = 0; j < n; ++j){
        if(clustered_map[j] == i)
            numOfElementsMatching++;
    }
    for (j = 0; j < n; ++j) {
       if(clustered_map[j] == i)
           extent_map[j] = numOfElementsMatching;
    }
}
```

Funzione computeTfceIteration (3/3)

```
clustered_map_float =
    copyAndConvertIntVector(extent_map, n);
apply_function(clustered_map_float, n, elevate, E);
apply_function(clustered_map_float, n, multiply,
    pow(h, H));
apply_function(clustered_map_float, n, multiply, dh);
for (i = 0; i < n; ++i) {
#pragma omp atomic
    toReturn[i] += (clustered_map_float[i]);
}
delete[] clustered_map_float;
delete[] clustered_map;
delete[] extent_map;
```

Funzione getBinaryVector

Questa funzione emula il risultato del costrutto Matlab (matrice <condizione> valore).

```
int * getBinaryVector(float * map, int n, int
    (*confront)(float, float), float value, int *
    numOfElementsMatching){
    int * binaryVector = new int [n];
    (*numOfElementsMatching) = 0;
    int i;
    for (i = 0; i < n; ++i) {
        if(confront(map[i],value)){
            binaryVector[i] = 1;
            (*numOfElementsMatching)++;
        }
        else
            binaryVector[i] = 0;
    }
    return binaryVector;
}
```

Calcolo dell'estensione dei cluster

La funzione **find_cluster_3D**:

```
int * find_clusters_3D(int * binaryVector, int dim_x,
    int dim_y, int dim_z, int n, int * num_clusters)
```

restituisce la mappa dei cluster trovati utilizzando la **26-connectivity** nell'immagine binaria fornita in input.

E' stato utilizzata la specifica OpenMP per rendere il calcolo degli score piú veloce.

OpenMP (Open Multiprocessing) é un API multipiattaforma per la creazione di applicazioni parallele su sistemi a memoria condivisa.

Il comando:
#pragma omp parallel for
viene utilizzato per rendere un for parallelo.

Il comando:
#pragma omp atomic
invece viene utilizzato per rendere un istruzione atomica.

Abbiamo deciso di utilizzare, OMP perché l'effort per utilizzarlo é praticamente nullo, e le prestazioni sono ottime.

Inoltre essendo che l'implementazione dei *Thread* in *C* cambia tra Windows e Linux, si sarebbe reso necessario modificare il codice per renderlo funzionante su entrambe le piattaforme.

Overview

Metodo di montecarlo per la sogliatura basata su cluster

Il metodo prende in input una soglia di significativitá α ed una mappa da sogliare e restituisce tale mappa sogliata.

- Vengono generate *n* mappe statistiche random con il metodo di Montecarlo
- Od ogni mappa viene applicato uno smoothing spaziale calcolato sulla mappa di input
- Si sceglie la taglia *t* tale che la probabilitá di osservare cluster generati casualmente di taglia maggiore di *t* sia uguale ad α

Il valore cosí ottenuto viene utilizzato per la sogliatura basata su cluster.

Bonferroni(1/2)

In statistica la **correzione di Bonferroni** viene utilizzata per contrastare il problema dei confronti multipli.

Cercando di mantenere il **familywise error rate**(FWER) all'interno di una determinata soglia.

L'FWER é la probabilitá di effettuare errori di Tipo 1 "falsi positivi" su tutte le ipotesi quando si effettuano test multipli.

Bonferroni(2/2)

Se si sta svolgendo l'esperimento con *m* ipotesi, un modo per mantenere l'FWER é quello di testare ogni ipotesi individualmente con una significatività statistica di $1/m$ moltiplicato per il livello massimo desiderato.

Quindi, se vogliamo un *p-value* totale di α, la correzione di Bonferroni testerá ogni singolo esperimento con un valore di α/m e rifiuterá l'ipotesi nulla se il p-value di quell'esperimento é minore di tale valore.

False Discovery Rate (1/2)

Il **False Discovery Rate** come la correzione di Bonferroni si prefigge l'obiettivo di contrastare il problema dei confronti multipli.

La procedura per il controllo FDR é stata creata per gestire la proporzione attesa di rifiuto dell'ipotesi nulla, che peró sarebbe stato sbagliato rifiutare ("false discoveries").

La procedura FDR fornisce un controllo meno stringente sugli errori di *Tipo 1* rispetto a Bonferroni.

False Discovery Rate (2/2)

Sia:

- V il numero di falsi positivi (Errori di Tipo 1)
- S il numero di veri positivi
- $R = V + S$

La FDR é definita:

$$FDR = E\left[\frac{V}{S+V}\right] = E\left[\frac{V}{S+V}\right]$$

dove $\frac{V}{R} = 0$ quando $R = 0$.

Avendo $H_1 \dots H_m$ test sull'ipotesi nulla e $p_1 \dots p_m$ *p-value* corrispondenti. Ordiniamo i *p-value* in ordine crescente; il *p-value* piú piccolo corrisponde al test con valore statistico piú alto.

La procedura Benjamini-Hochberg controlla il false discovery rate (al livello α) con i seguenti passi:

1. Per un dato α, trova il piú grande k per cui: $p_k \leq \frac{k}{m}\alpha$
2. Rifiuta l'ipotesi nulla (accetta come discovery vere) tutti i test $H_1 \dots H_k$

La procedura di Benjamini-Hochberg é valida quando i m test sono indipendenti e soddisfa anche la seguente equazione:

$$FDR \leq \frac{m_0}{m}\alpha \leq \alpha$$

Overview

Presentazione dei risultati

I risultati che seguono enfatizzano i due principali obiettivi preposti:

1. Il plugin sviluppato produce risultati confrontabili con l'implementazione in FSL.
2. La tecnica TFCE produce risultati confrontabili con altri metodi di correzione presenti in Brainvoyager (introdotti precedentemente).

FSL TFCE

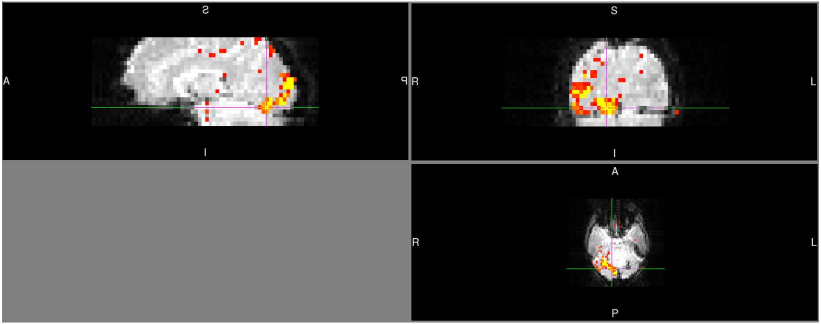


Figura: Tfce calcolato su FSL.

Brainvoyager TFCE

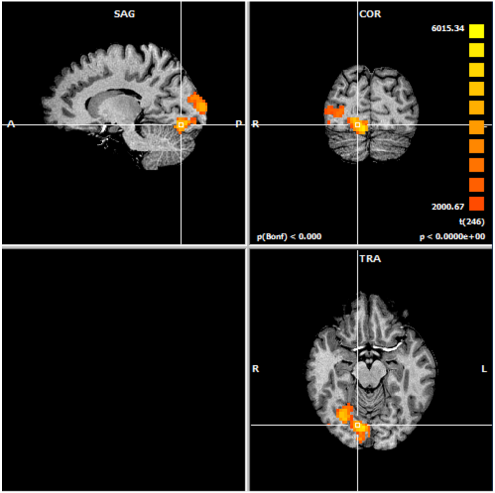
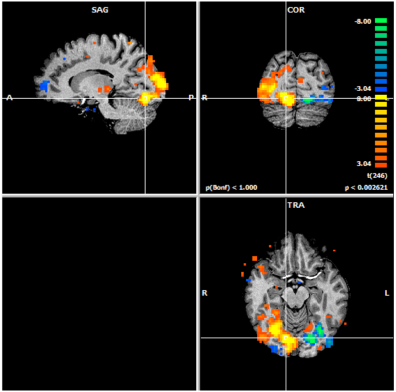
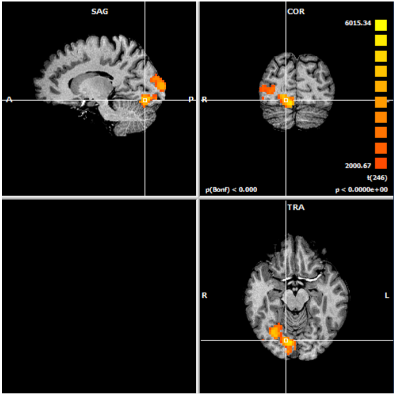


Figura: Tfce calcolato su Brainvoyager tramite il plugin.

Original vs TFCE

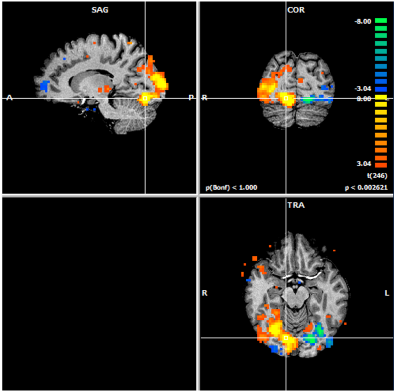


(a) Original

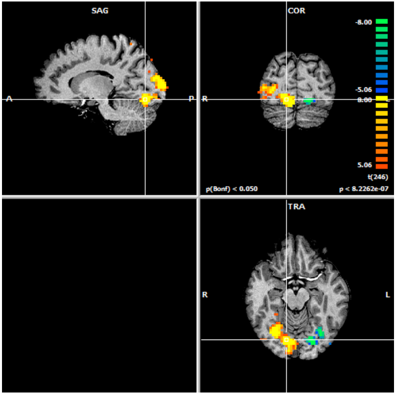


(b) TFCE

Original vs Bonferroni

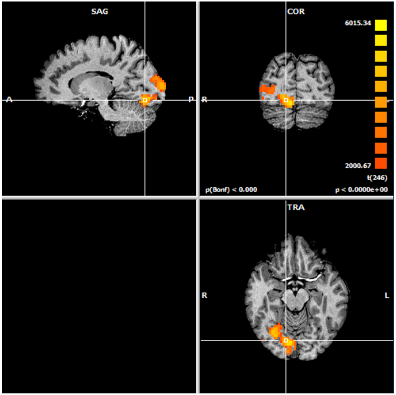


(a) Original

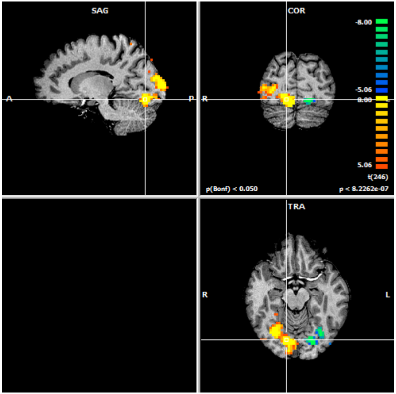


(b) Bonferroni

TFCE vs Bonferroni

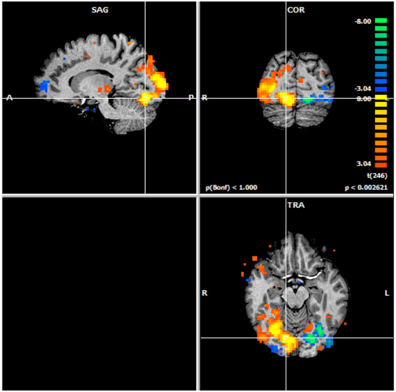


(a) TFCE

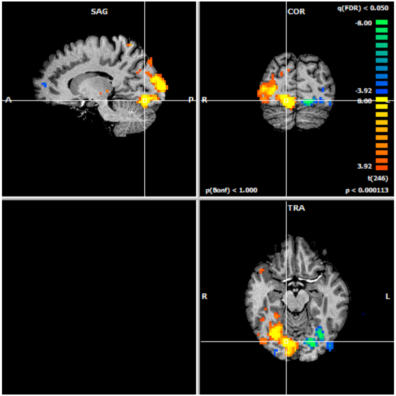


(b) Bonferroni

Original vs FDR

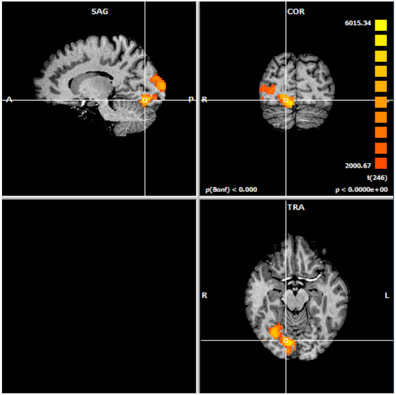


(a) Original

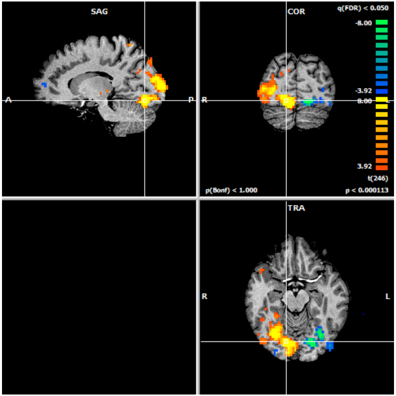


(b) FDR

TFCE vs FDR

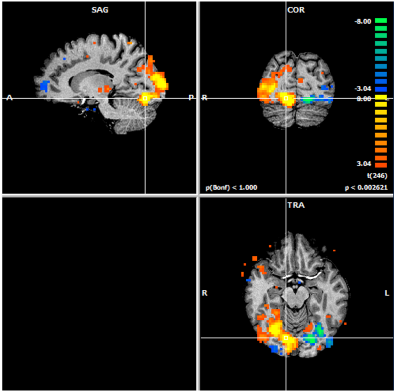


(a) TFCE

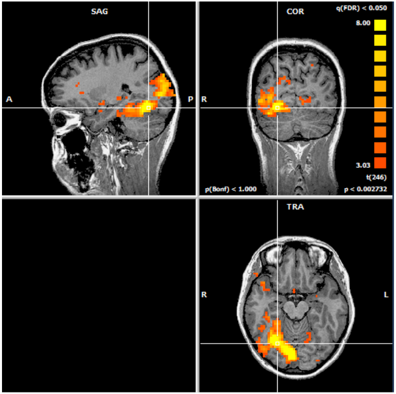


(b) FDR

Original vs Montecarlo Simulations

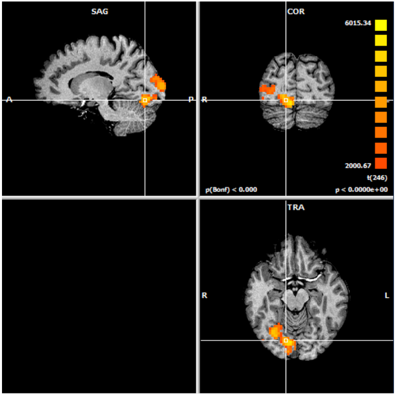


(a) Original

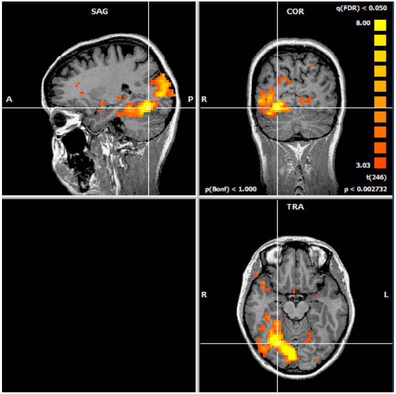


(b) Montecarlo

TFCE vs Montecarlo Simulations



(a) TFCE



(b) Montecarlo

Overview

Conclusioni

Il plugin sviluppato risulta essere efficace nella correzione delle mappe generate tramite contrasti statistici sui risultati di un GLM.

Tuttavia, l'implementazione corrente produce mappe prive di valore statistico, in quanto la parte relativa al test delle permutazioni non é stata sviluppata.

Sviluppi futuri

A partire dal lavoro svolto, futuri sviluppi prevedono:

- Sviluppo del framework randomise ed inclusione del plugin in esso.
- Calcolo congiunto degli score positivi e negativi.
- Estendere i risultati ad altre mappe statistiche.
- Confrontare la tecnica TFCE con Gaussian Field mapping implementato in SPM.

Grazie per l'attenzione