



Online Computation

Moo K. Chung
Department of Biostatistics and Medical Informatics
University of Wisconsin-Madison

www.stat.wisc.edu/~mchung

Iterative
algorithms

Online learning

Real time
prediction

Online
statistical
analysis

Online Computation

Large-scale
computation

Big network data

Big data

Big data in medical imaging: large- p small- n problem p voxels >> n subjects

i-th subject: $\mathbf{x}_i = (x_{i1}, \dots, x_{ip})'$

assumption:

$$\mathbb{E}\mathbf{x}_i = 0 \quad \mathbb{E}\mathbf{x}_i\mathbf{x}_j' = \Sigma$$

The dependency among p voxels is characterized by the covariance matrix.

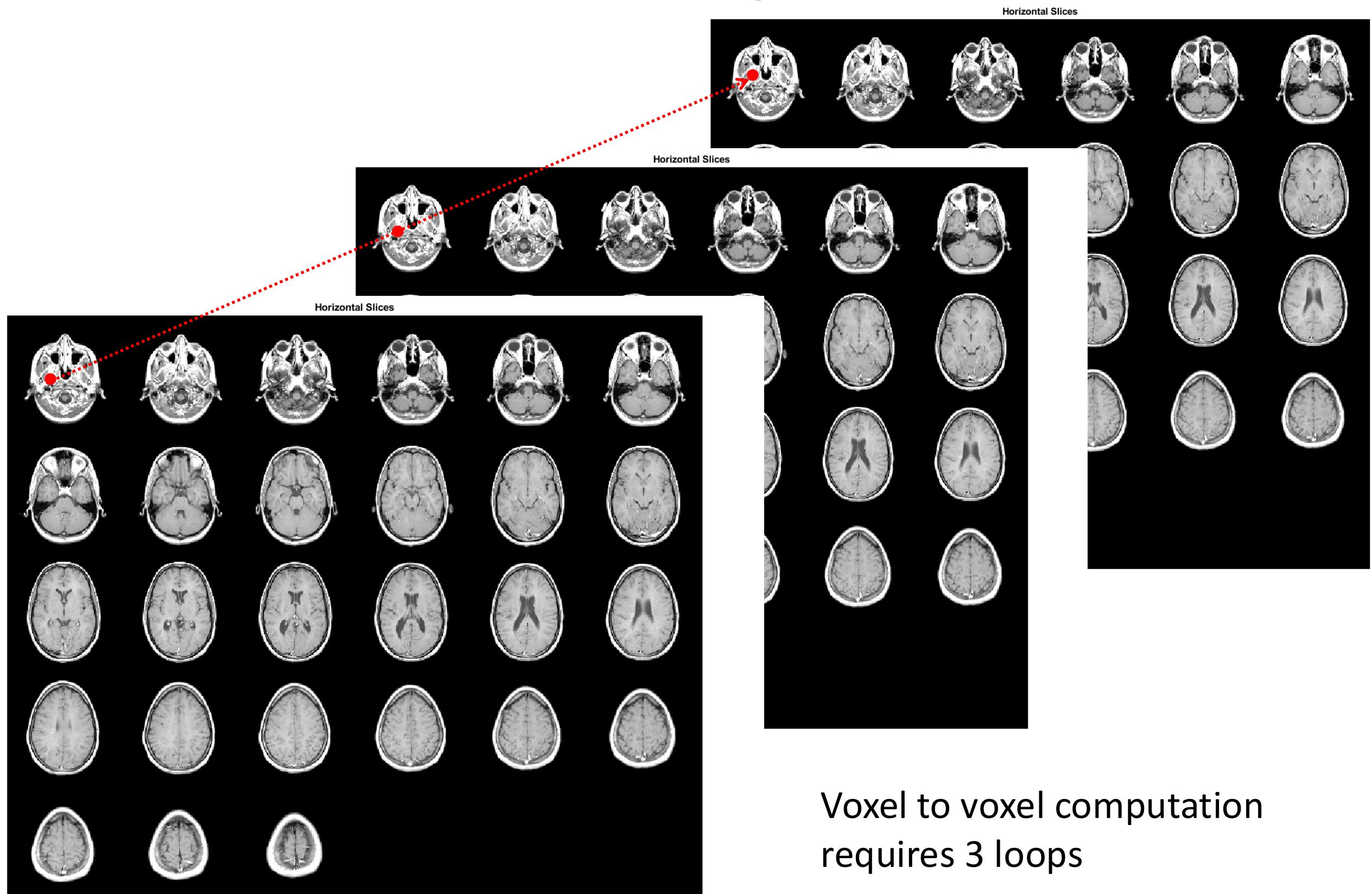
Rank deficiency in large-scale covariance matrix

$$\mathbb{E}\mathbf{x}_i = 0 \quad \mathbb{E}\mathbf{x}_i\mathbf{x}'_j = \Sigma$$

$$\Sigma = U^\top D(\eta_1, \dots, \eta_p) U$$

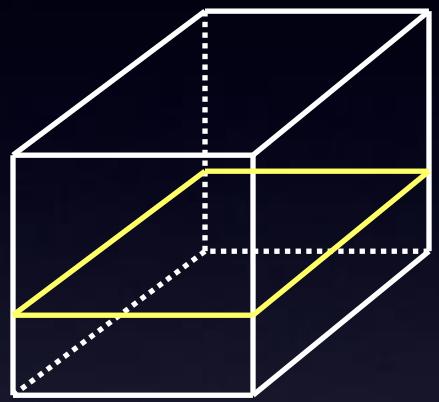
For n subjects data, up to n positive eigenvalues

How brain imaging packages (SPM, AFNI) handles 3D images

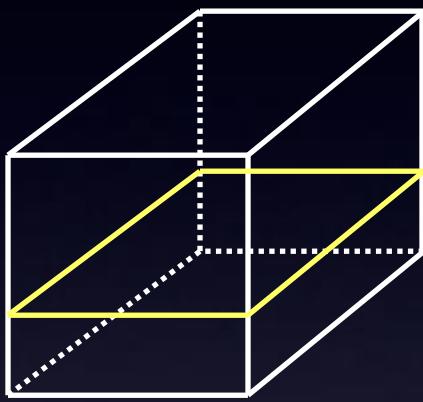


Voxel to voxel computation
requires 3 loops

Existing method for computing large 3D medical images

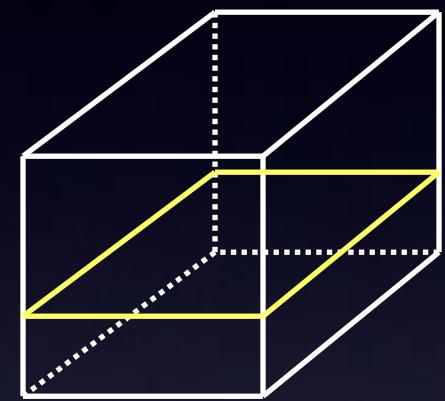


Subject 1



Subject 2

...



Subject n

$$I_1(x, y, \cdot)$$

$$I_2(x, y, \cdot)$$

....

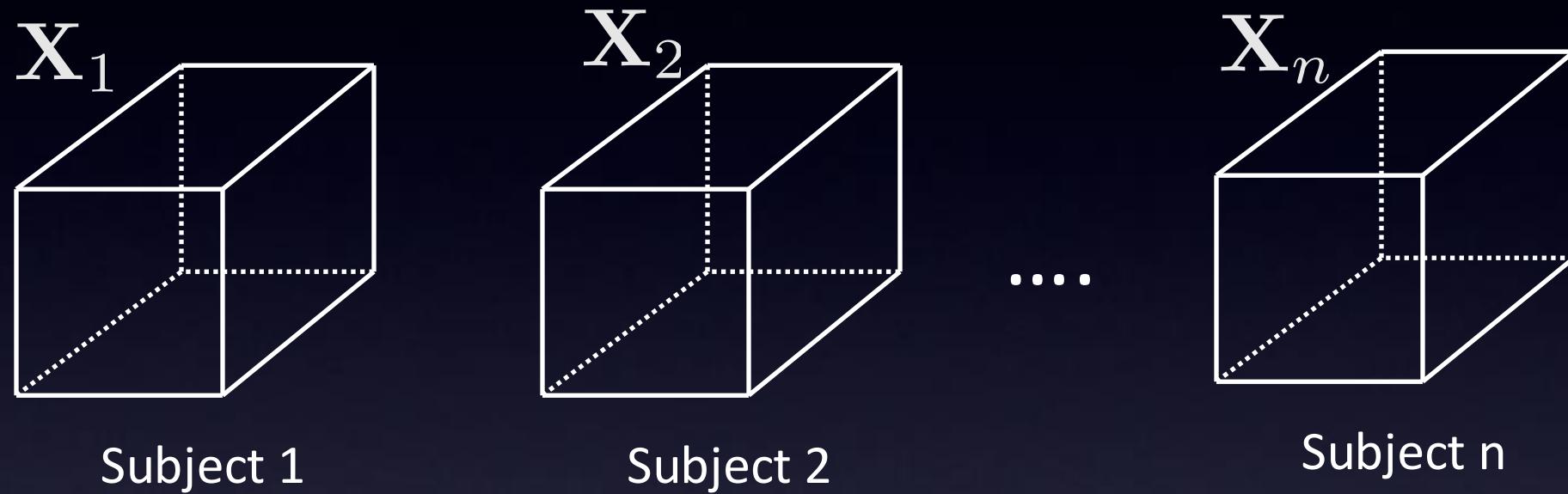
$$I_n(x, y, \cdot)$$

Online computation

An *online algorithm* is one that processes its inputted data in a sequential manner (Karp, 1992).

$$f(I_1, I_2, \dots, I_n), I_{n+1} \rightarrow f(I_1, I_2, \dots, I_{n+1})$$

Basic idea of online computation for large 3D medical images



$$f(I_1, I_2, \dots, I_n), I_{n+1} \rightarrow f(I_1, I_2, \dots, I_{n+1})$$

Online computation for sample mean

Given x_1, \dots, x_{m-1} add x_m

$$\mu_m = \frac{1}{m} \sum_{i=1}^m x_i = \mu_{m-1} + \frac{1}{m} (x_m - \mu_{m-1})$$

Initial value $\mu_1 = x_1$

Online computation for sample variance

$$\sigma_m^2 = \frac{1}{m-1} \sum_{i=1}^m (x_i - \mu_m)^2$$

$$= \frac{m-2}{m-1} \sigma_{m-1}^2 + \frac{1}{m} (x_m - \mu_{m-1})^2$$

Initial value $\sigma_1^2 = 0$

Online computation for two-sample t -stat

x_1, \dots, x_m

y_1, \dots, y_n

Under null

$$T_{m,n} = \frac{\mu_m^1 - \mu_n^2 - (\mu^1 - \mu^2)}{\sqrt{(\sigma^1)_m^2/m + (\sigma^2)_n^2/n}}$$

- (m,n)

- (0,0)

$$T_{1,0} \rightarrow T_{2,0} \rightarrow \dots \rightarrow T_{m,0} \rightarrow T_{m,1} \rightarrow \dots \rightarrow T_{m,n}$$

Online computation for regression

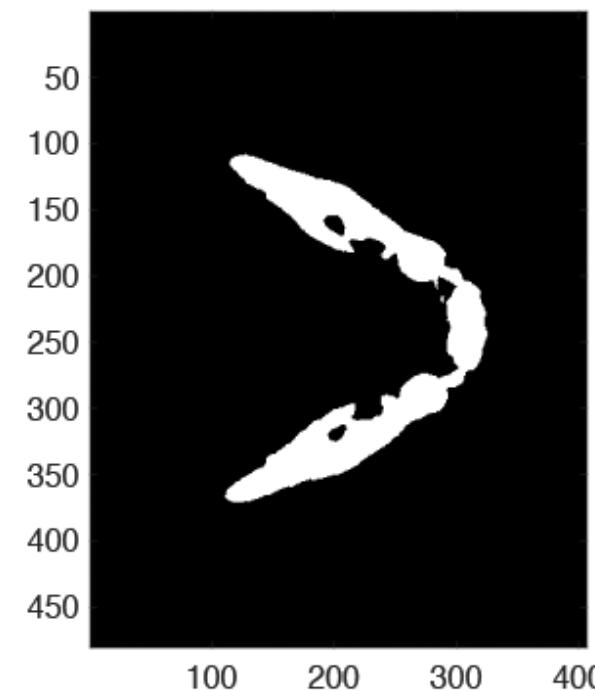
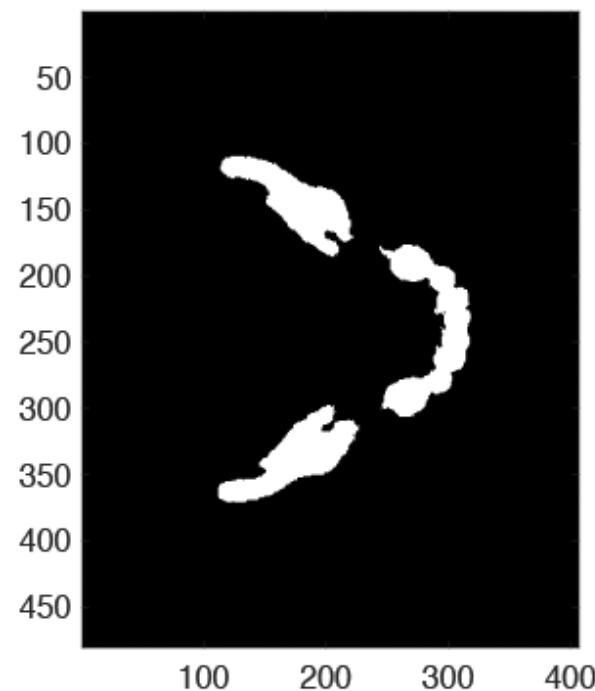
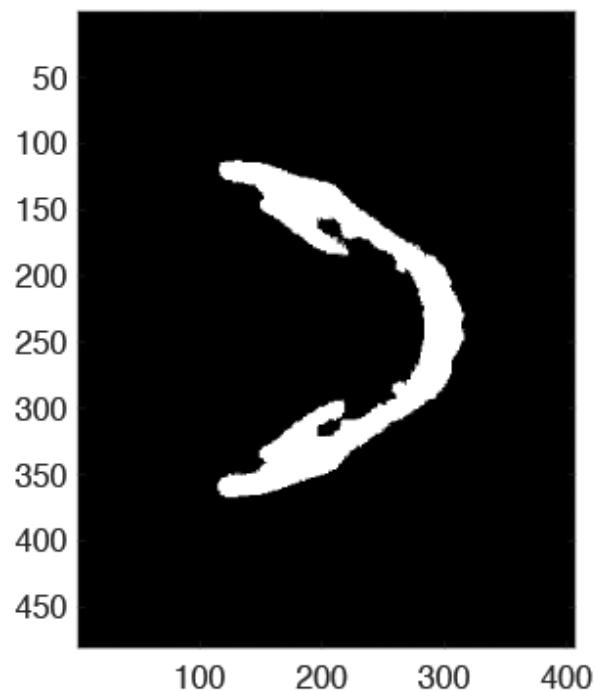
$$\mathbf{y}_{m-1} = Z_{m-1} \boldsymbol{\lambda}_{m-1}$$

$$W_{m-1} = Z'_{m-1} Z_{m-1}$$

Based on Woodbury formula and Schur complement

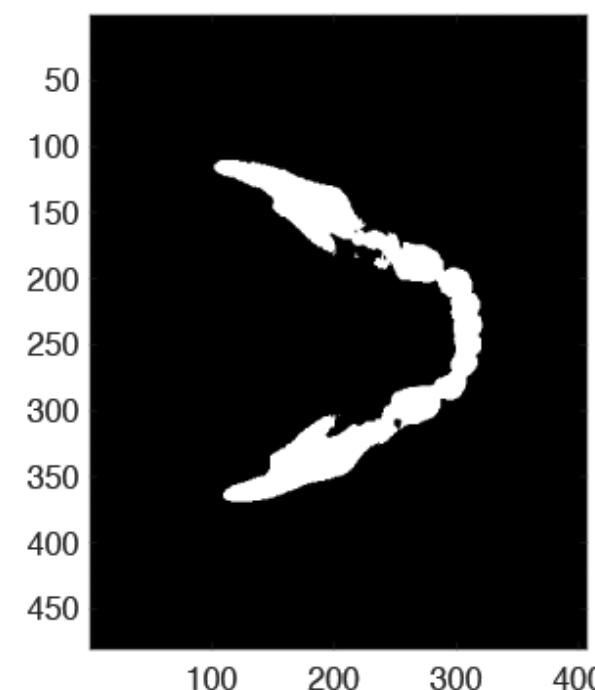
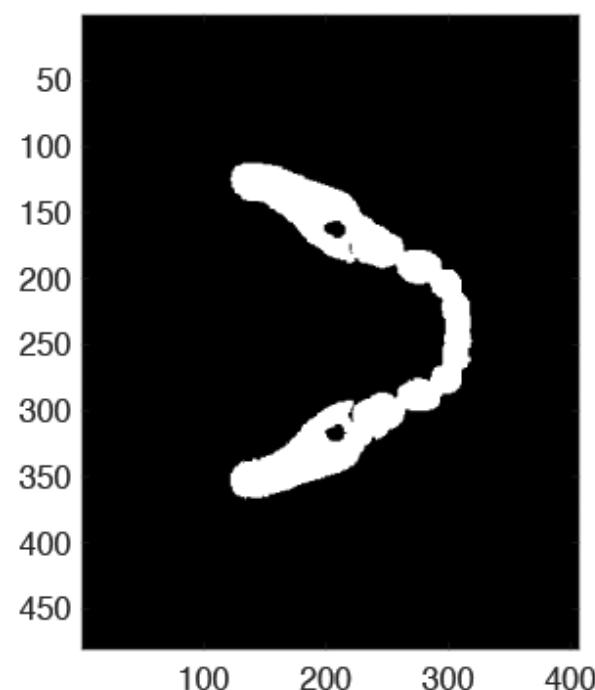
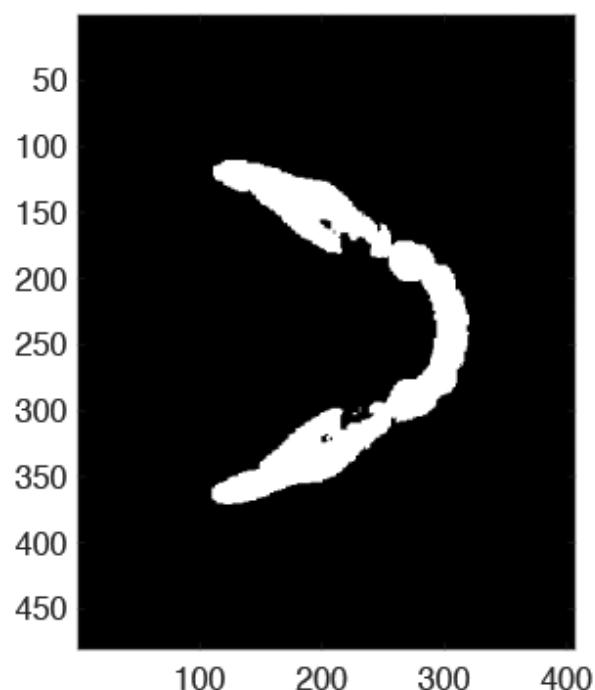
$$\boldsymbol{\lambda}_m = (I - W_{m-1}^{-1} z'_m y_m - c_m W_{m-1}^{-1} z'_m W'_{m-1}) \boldsymbol{\lambda}_{m-1} - c_m W_{m-1}^{-1} z'_m z'_m y_m$$

Application to large CT dataset



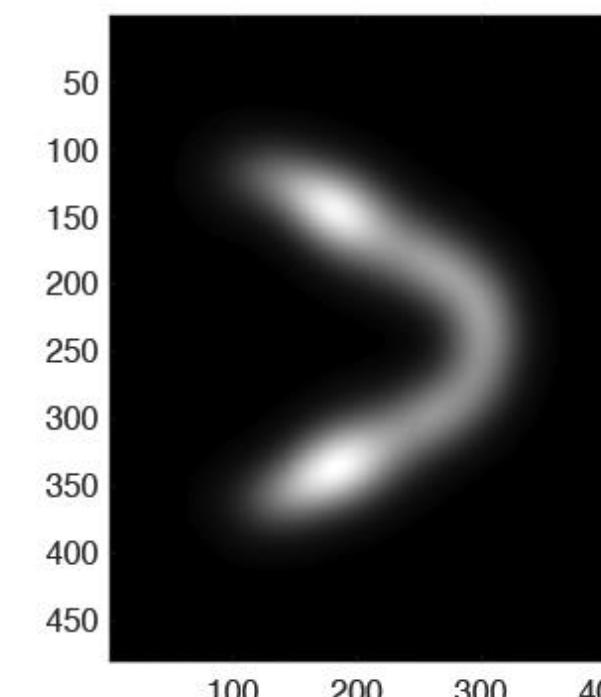
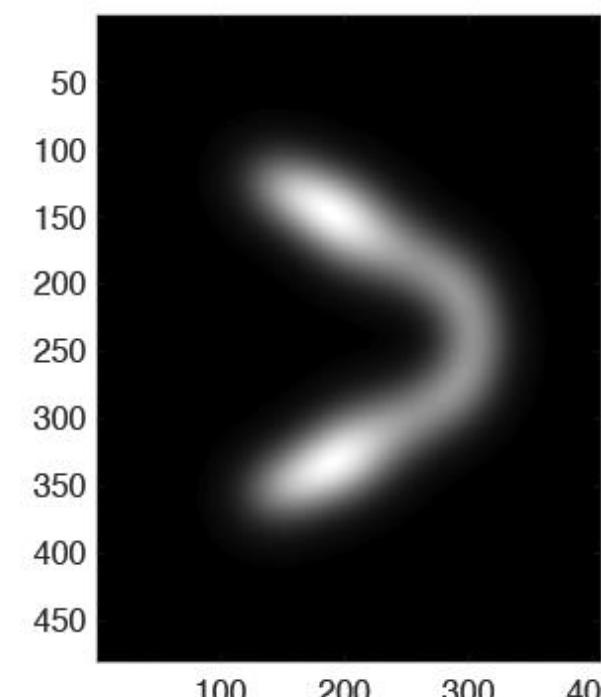
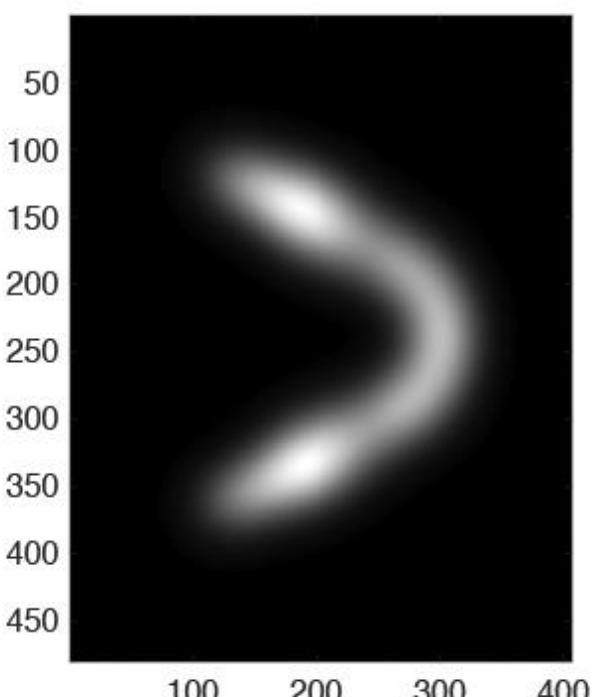
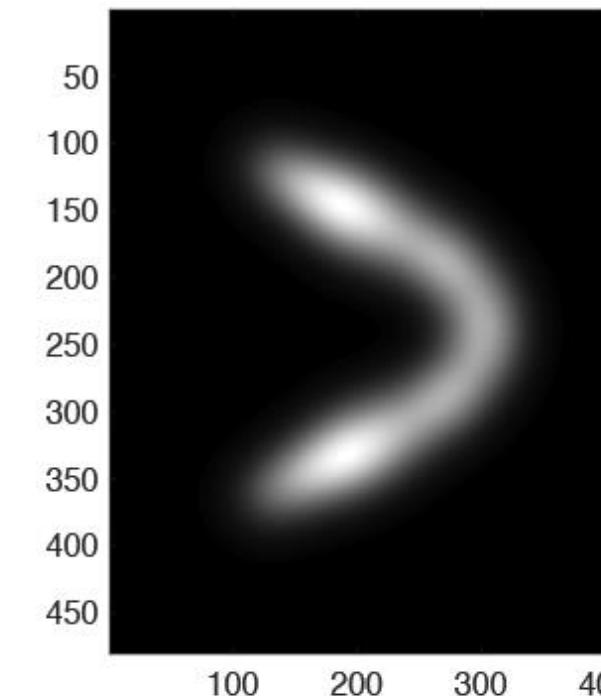
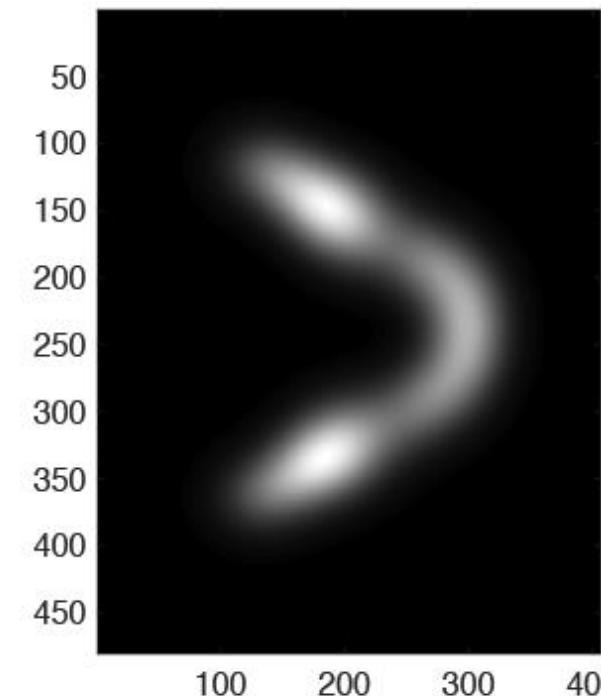
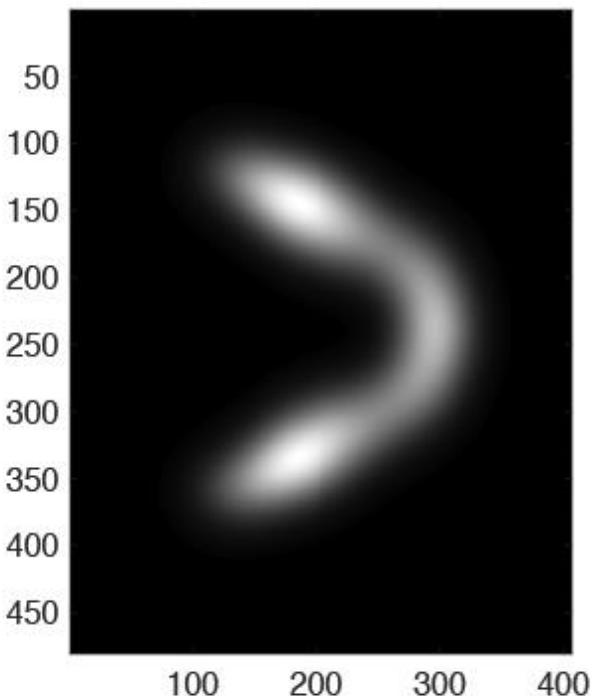
$4^3 = 64$ times
bigger image
than MRI

160 males
130 females

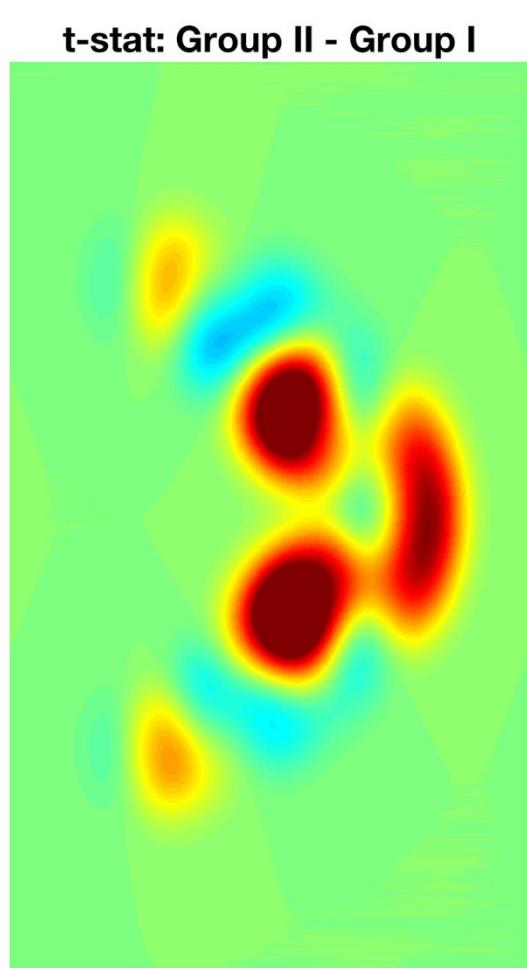


Voxel-based morphometry

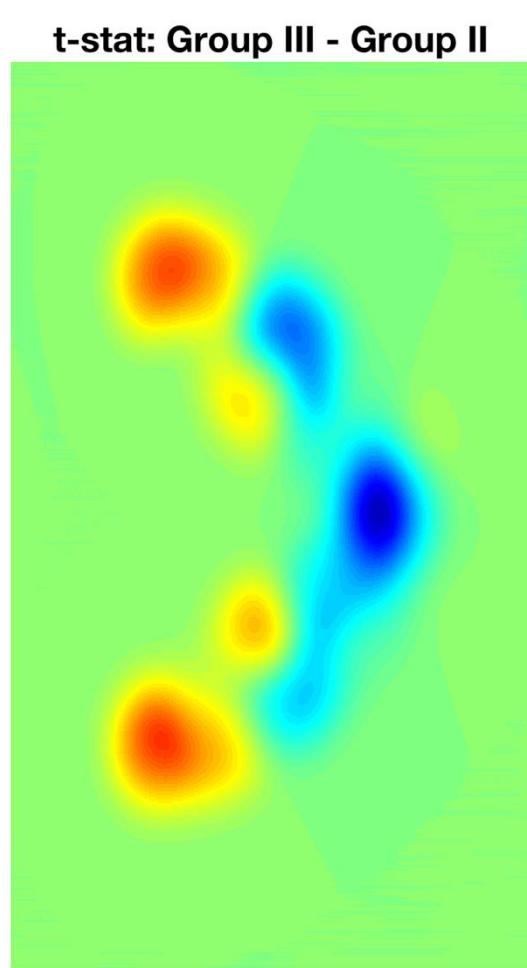
Smoothed image = probability density



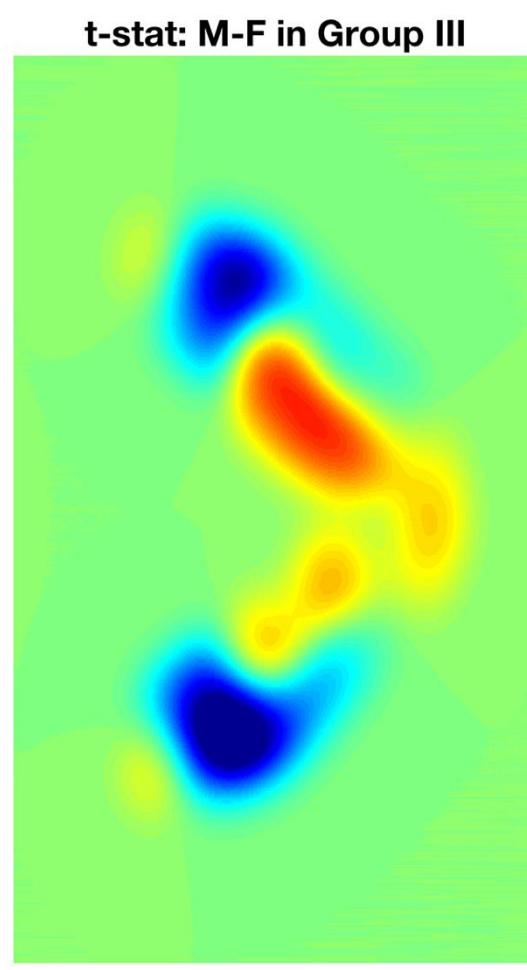
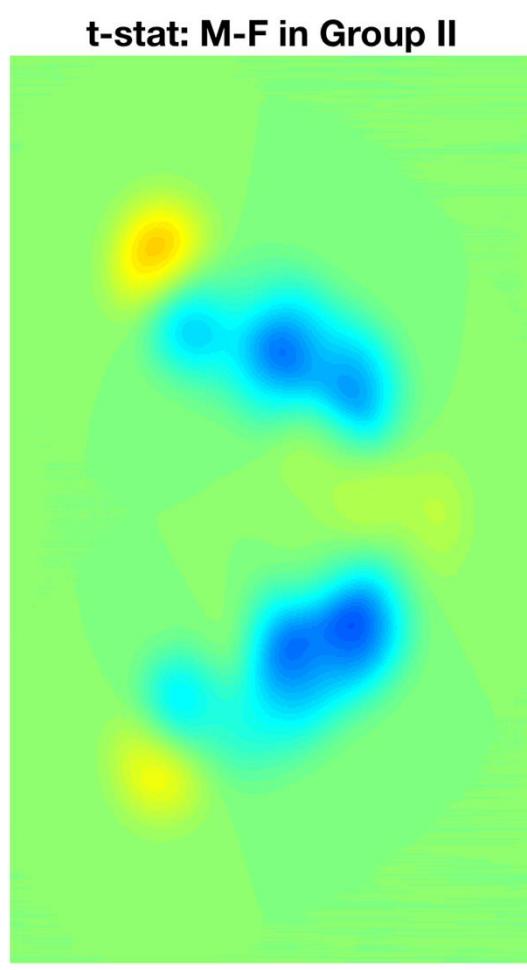
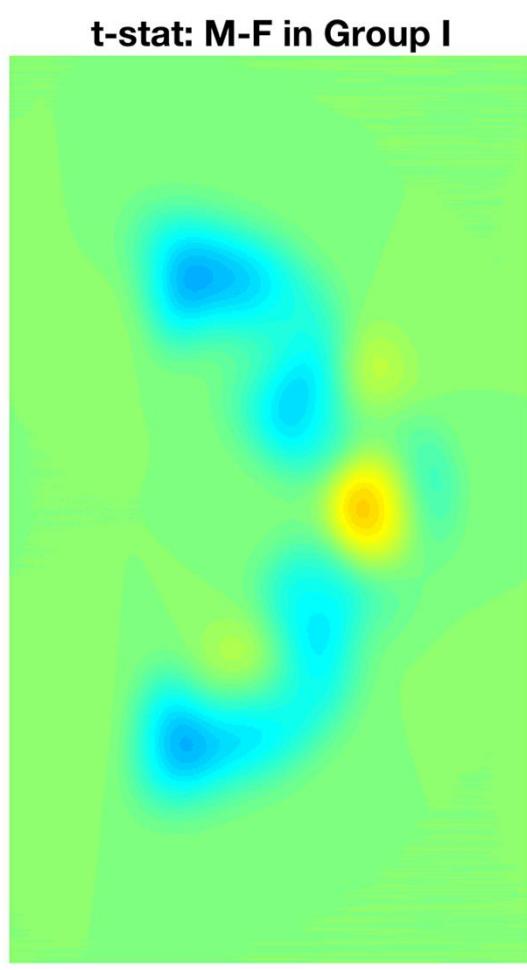
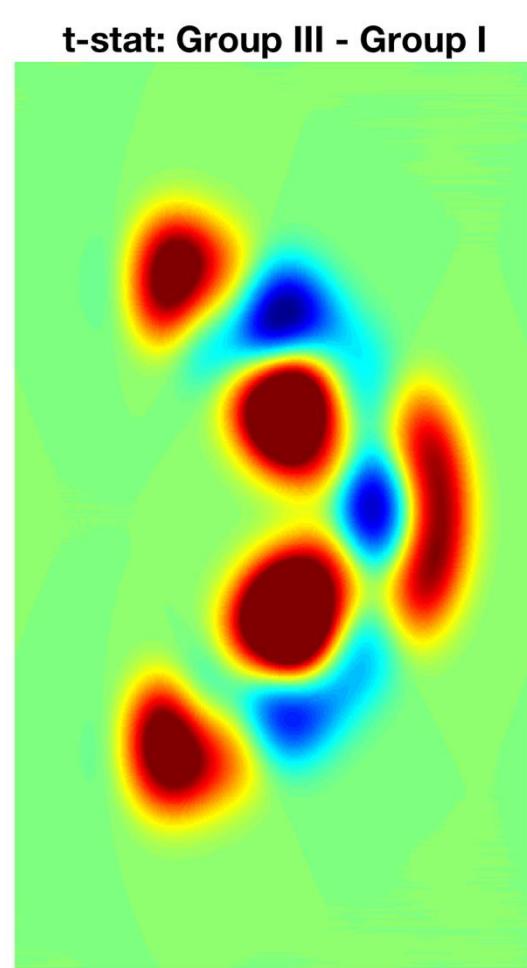
Group I:
-7 years



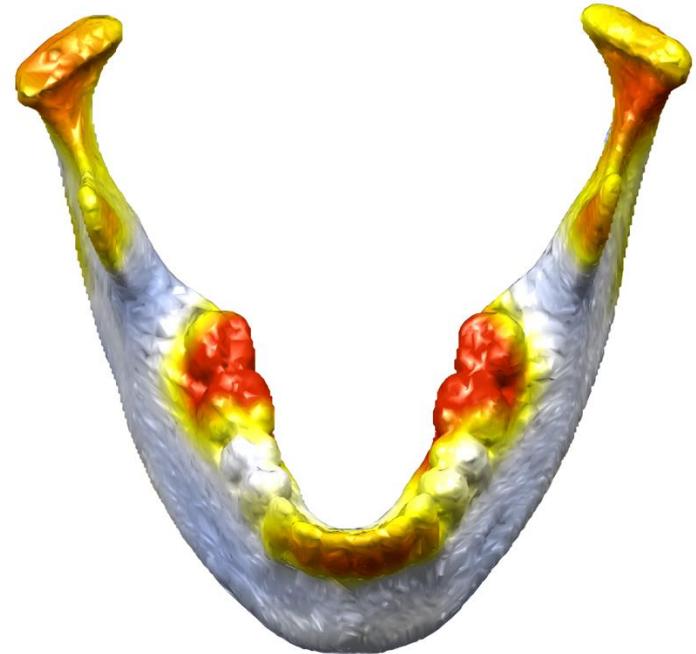
Group II:
7-13 years



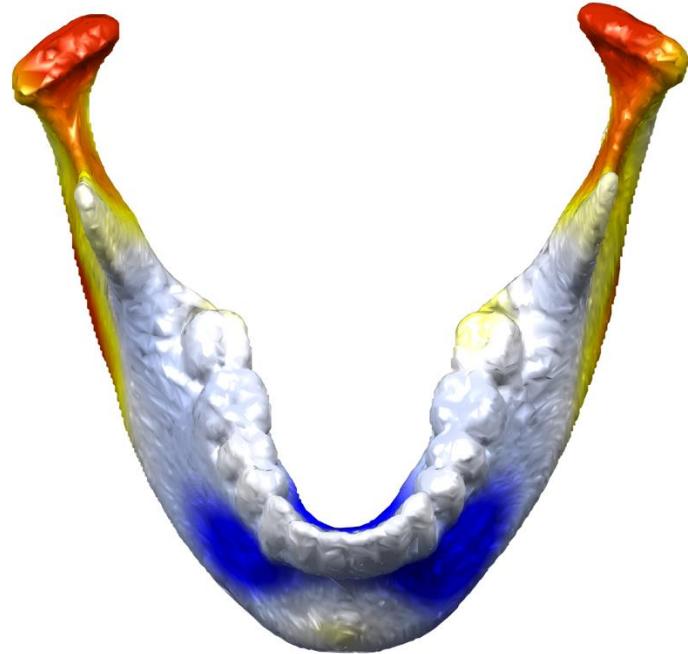
Group III:
13-20 years



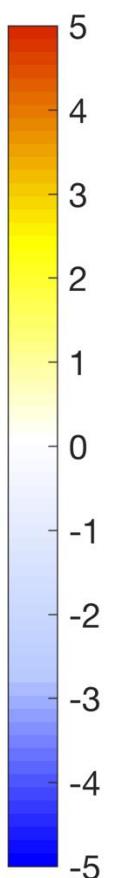
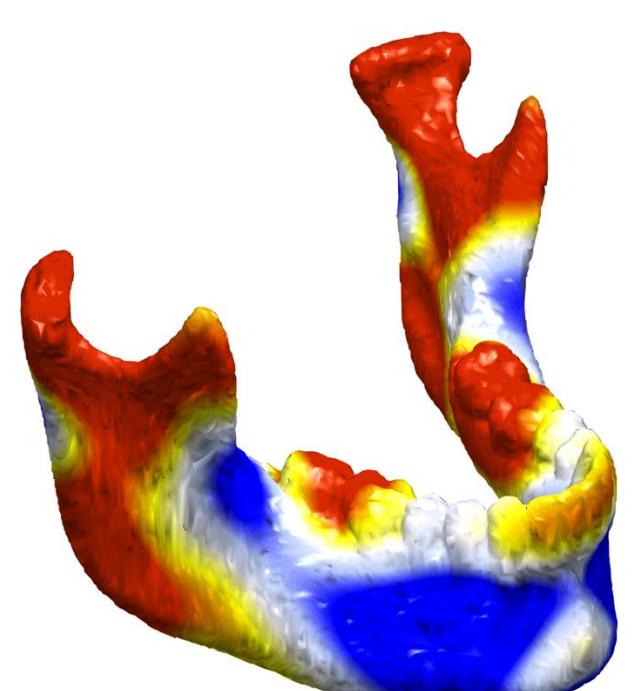
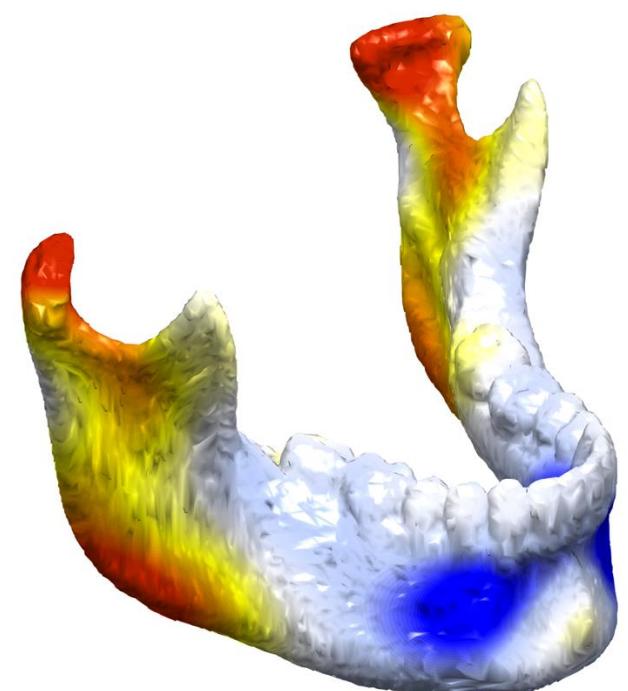
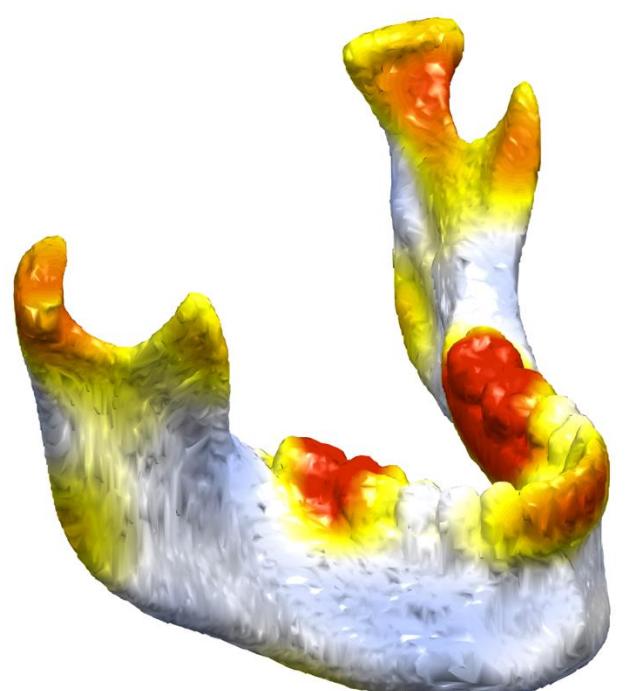
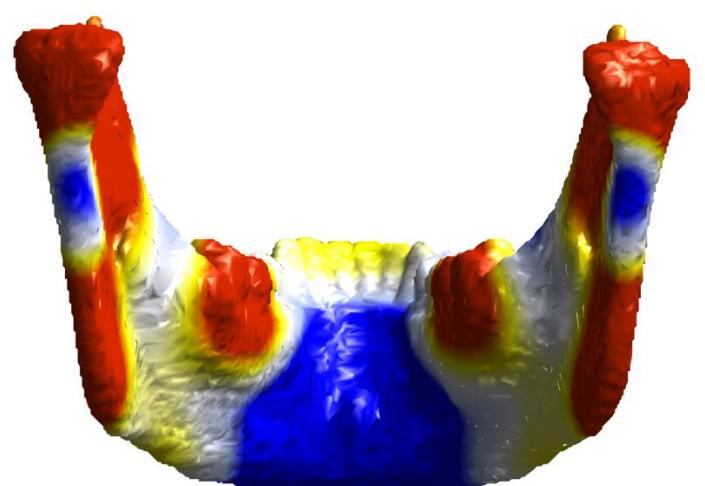
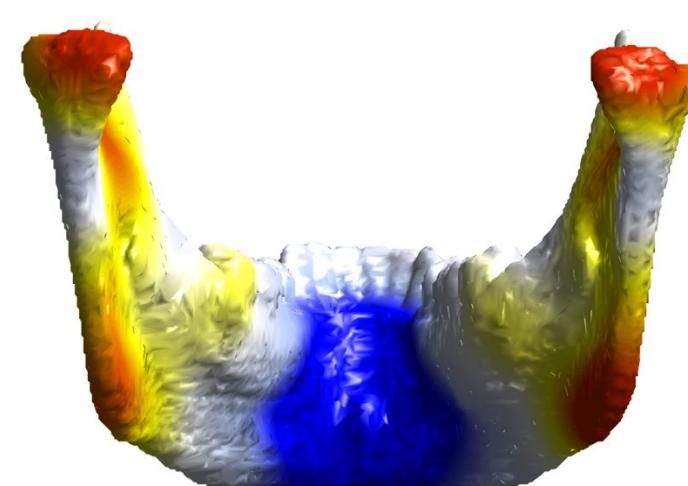
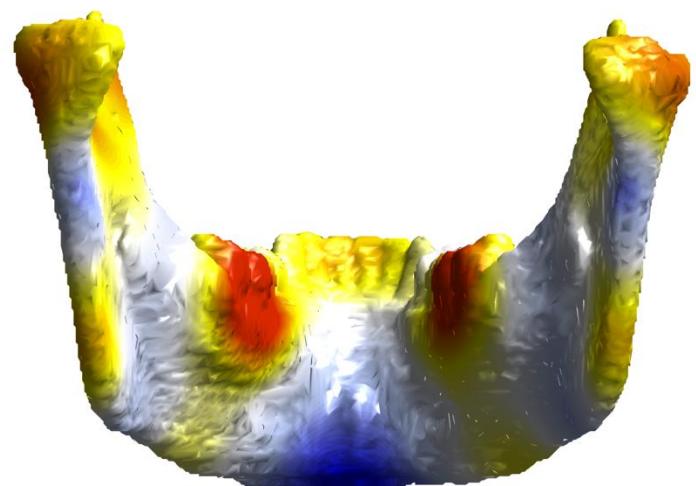
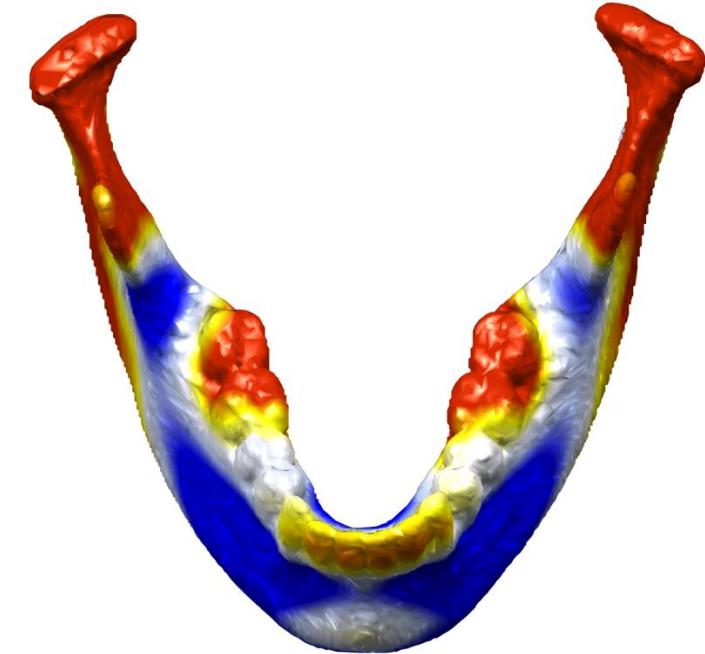
Group II-I



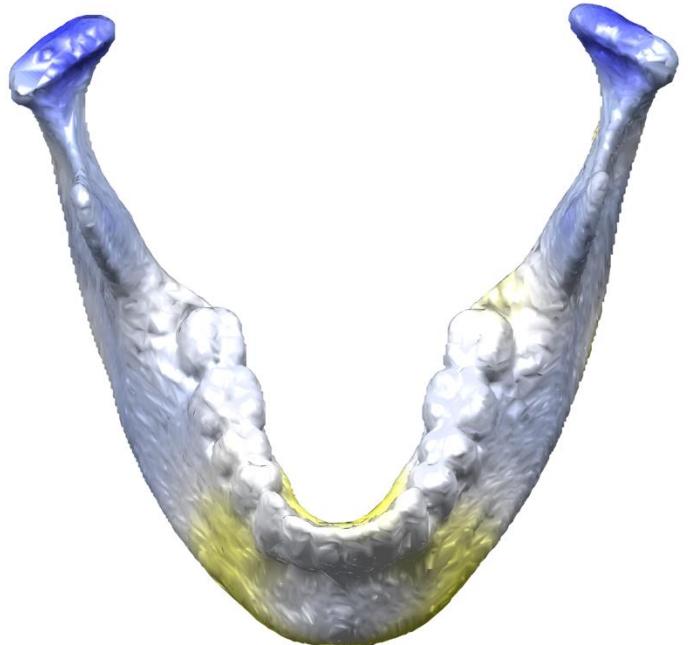
Group III-II



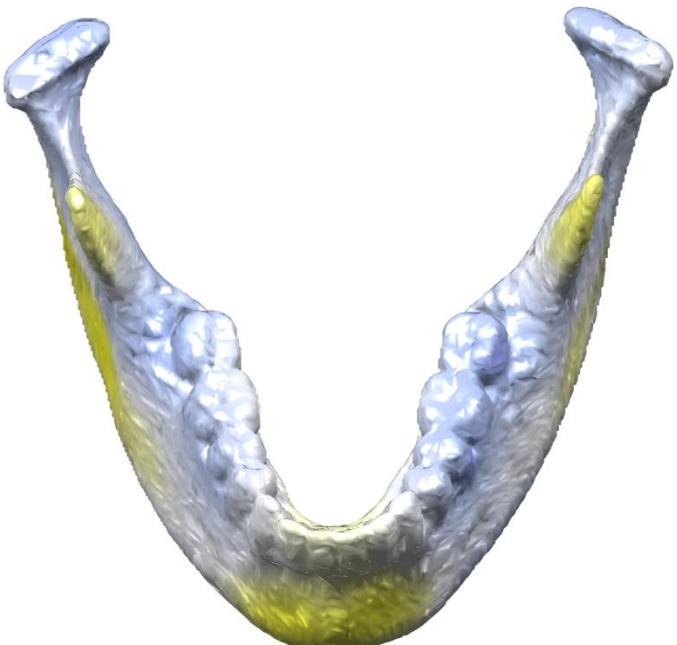
Group III-II



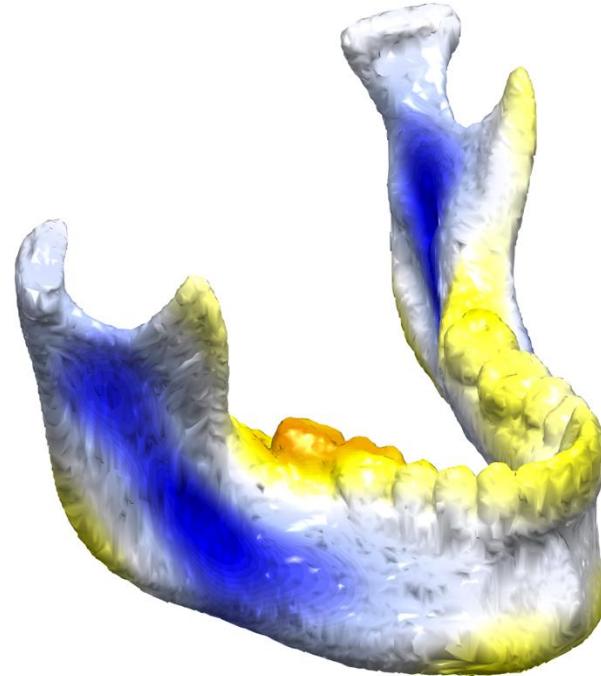
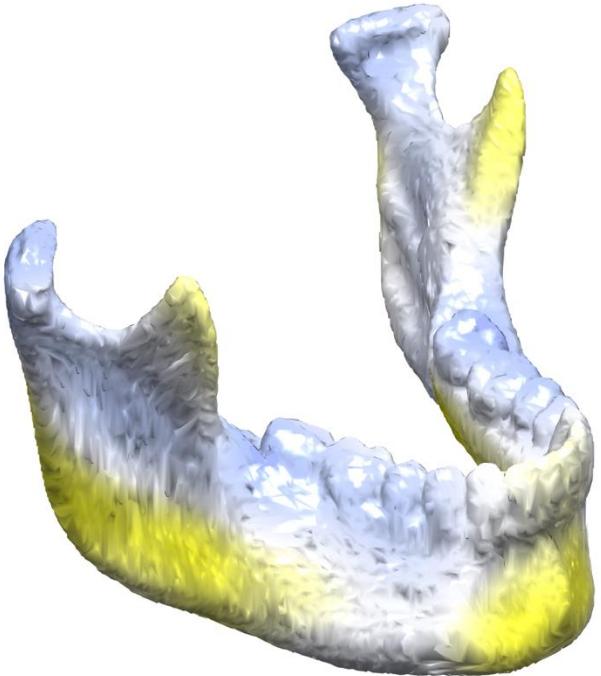
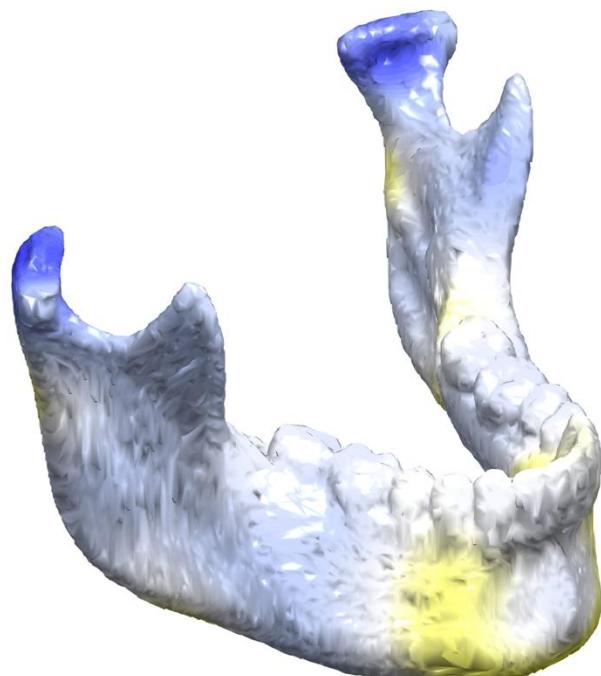
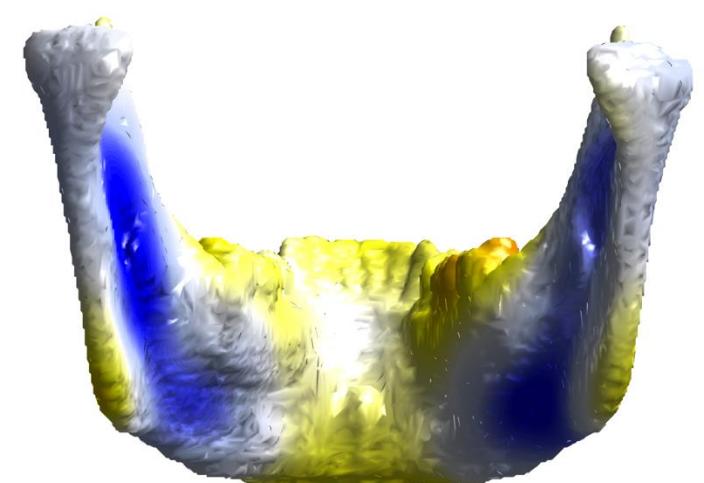
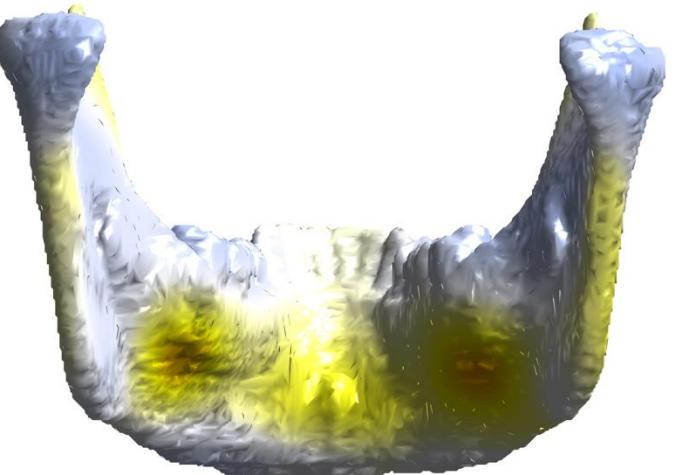
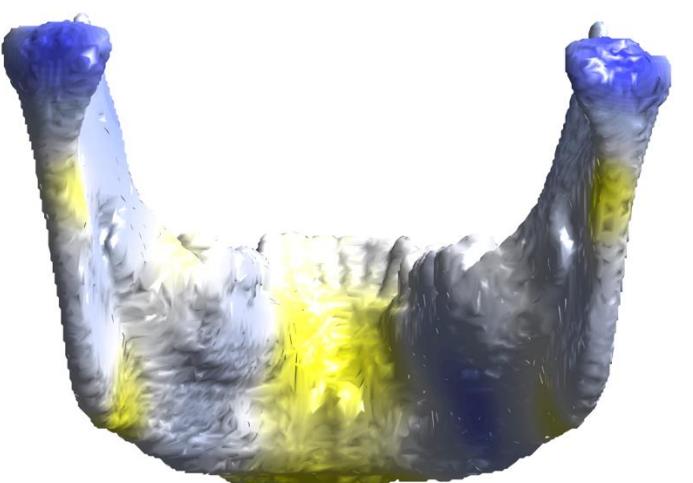
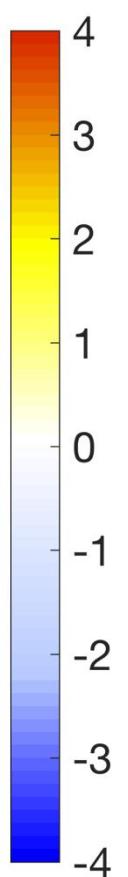
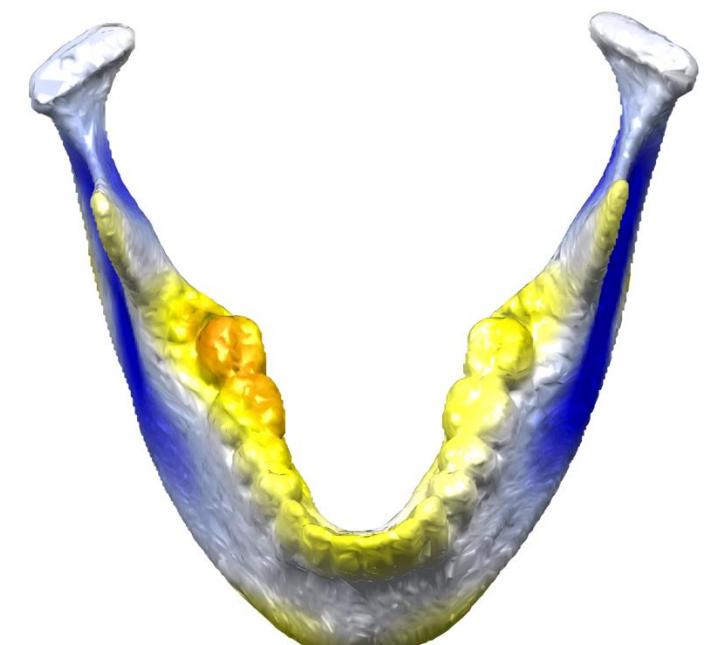
Group I (M-F)



Group II (M-F)



Group III (M-F)



It seems like **online algorithms** require
brute-force algebraic derivations?

Given x_1, \dots, x_m

Is the online algorithm unique?

*Let's solve a more complex
problem!*

Transposition test

Online computation
of permutation test

Permutation test

$$\mathbf{x} = (x_1, x_2, \dots, x_m)$$

$$\mathbf{y} = (y_1, y_2, \dots, y_n)$$

$$(\mathbf{x}, \mathbf{y}) = (x_1, \dots, x_m, y_1, \dots, y_n)$$

$$\pi(\mathbf{x}, \mathbf{y}) \in \mathbb{S}_{m+n}$$

Permutation group of order $m+n$

$$p\text{-value} = \frac{1}{(m+n)!} \sum_{\pi \in \mathbb{S}_{m+n}} \mathcal{I}\left(f(\pi(\mathbf{x}), \pi(\mathbf{y})) \geq f(\mathbf{x}, \mathbf{y})\right)$$

↑
stat on each permutation
observation

Computational bottleneck

Why the permutation test is so slow

Most of computation is on recomputing the statistic function. The actual permutation itself is not computationally Bottleneck.

How to speed up the permutation test

Random transposition on the permutation group

$$\mathbf{x} = (x_1, x_2, \dots, x_{i-1}, \underset{\text{circle}}{x_i}, x_{i+1}, \dots, x_m)$$

transpose i -th and j -th data

$$\mathbf{y} = (y_1, y_2, \dots, y_{j-1}, \underset{\text{circle}}{y_j}, y_{j+1}, \dots, y_n)$$



$$\pi_{ij}(\mathbf{x}) = (x_1, x_2, \dots, x_{i-1}, \underset{\text{circle}}{y_j}, x_{i+1}, \dots, x_m)$$

$$\pi_{ij}(\mathbf{y}) = (y_1, y_2, \dots, y_{j-1}, \underset{\text{circle}}{x_i}, y_{j+1}, \dots, y_n)$$

Online computation over transpositions

Identify
pattern

x_i deleted $\rightarrow y_j$ added

Define

$$\mathbf{x} = (x_1, \dots, x_m)$$

$$\nu(\mathbf{x}) = \sum_{j=1}^m x_j, \quad \omega(\mathbf{x}) = \sum_{j=1}^m \left(x_j - \frac{\nu(\mathbf{x})}{m} \right)^2$$

Apply
the pattern

delete $\nu(\mathbf{x}, a) = \nu(\mathbf{x}) + a$

add $\nu(\mathbf{x}, b) = \nu(\mathbf{x}) + b$

Derive

$$\nu(\pi_{ij}(\mathbf{x})) = \nu(\mathbf{x}) - x_i + y_j$$

$$\omega(\pi_{ij}(\mathbf{x})) = \omega(\mathbf{x}) - x_i^2 + y_j^2 + \frac{\nu(\mathbf{x})^2 - \nu(\pi_{ij}(\mathbf{x}))^2}{m}$$

Online computation over transpositions

$$\nu(\mathbf{x}) = \sum_{j=1}^m x_j, \quad \omega(\mathbf{x}) = \sum_{j=1}^m \left(x_j - \frac{\nu(\mathbf{x})}{m} \right)^2$$

O(m) O(3m+2)

$$\nu(\pi_{ij}(\mathbf{x})) = \nu(\mathbf{x}) - x_i + y_j \quad \text{O}(2)$$

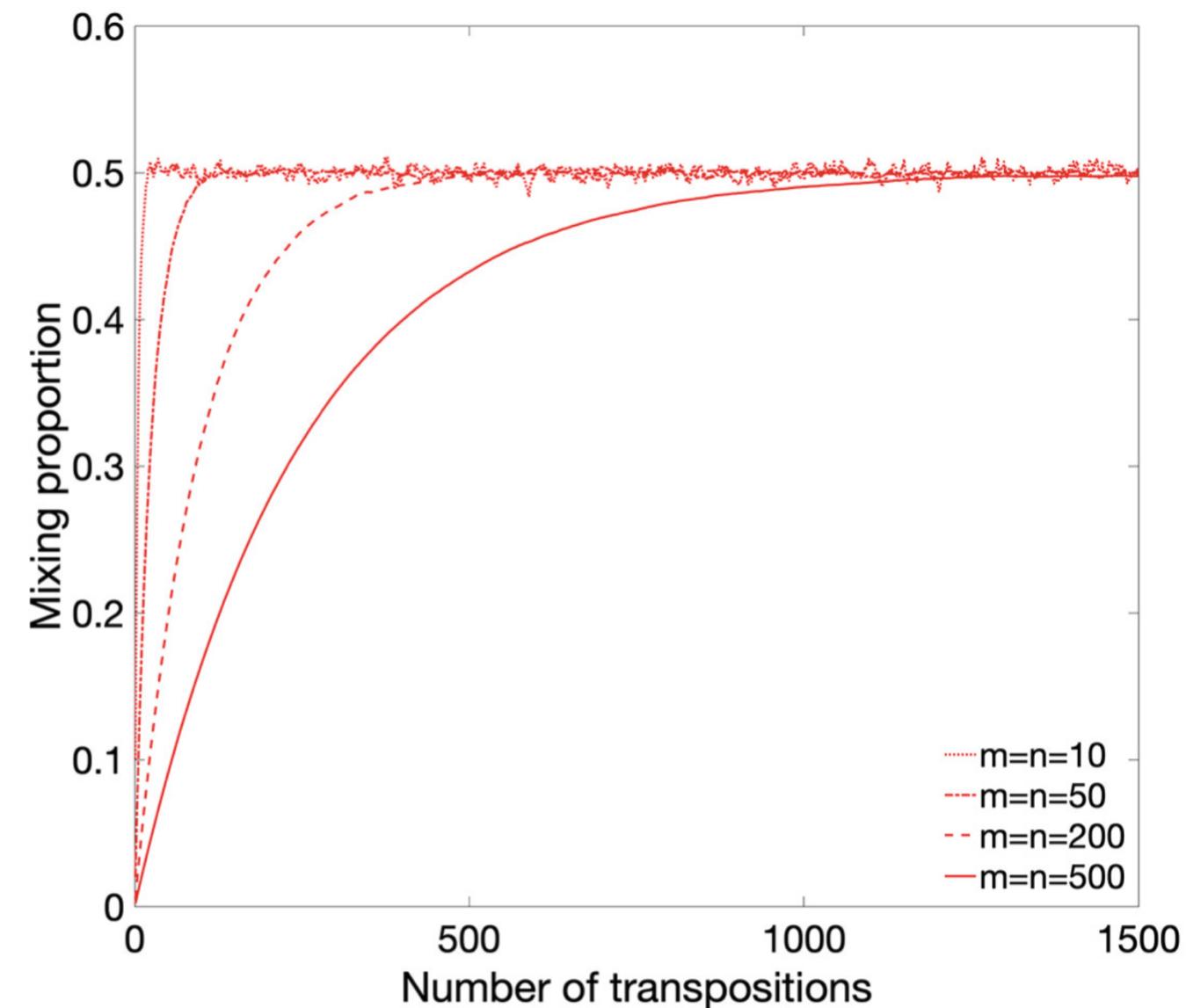
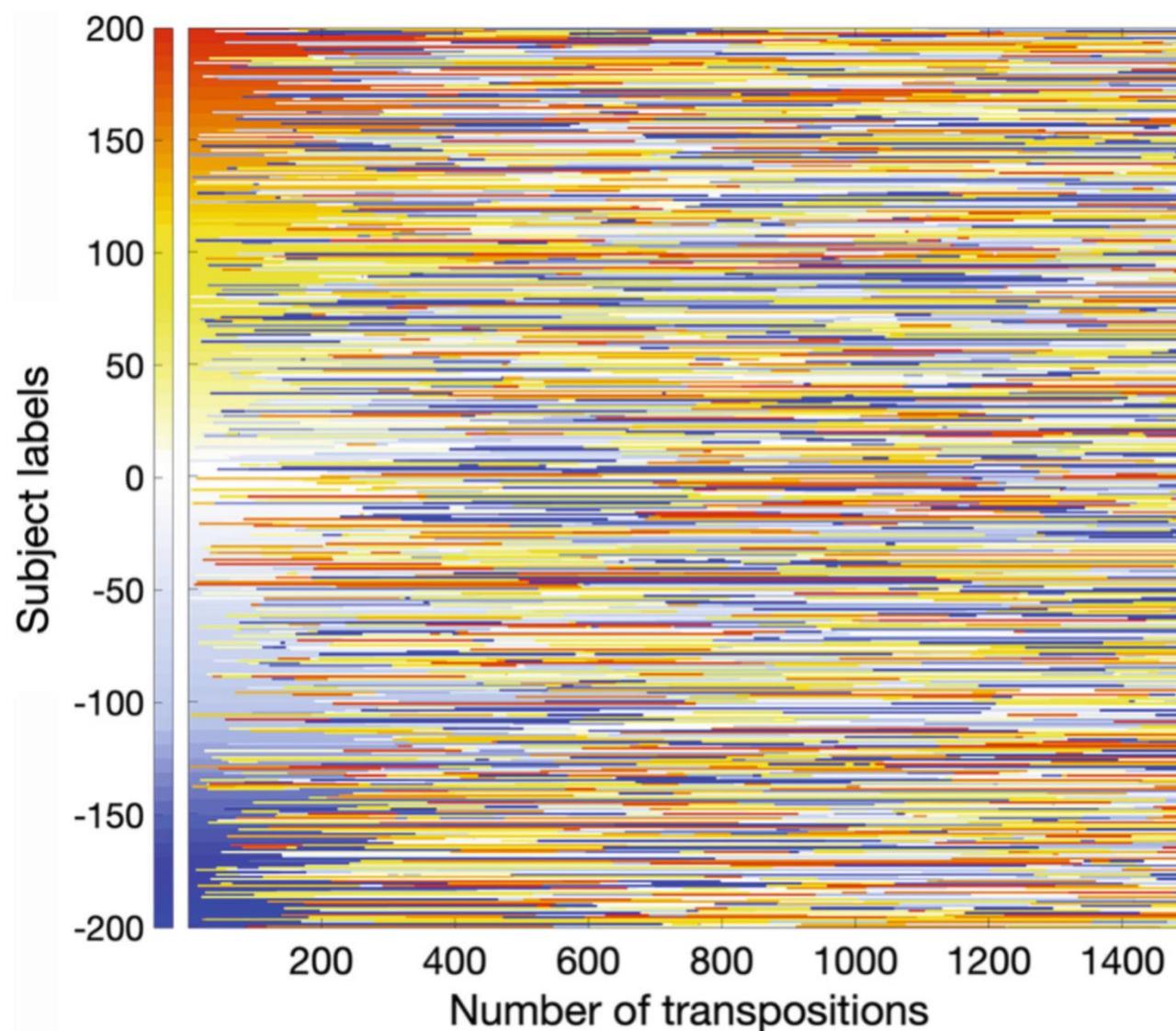
$$\omega(\pi_{ij}(\mathbf{x})) = \omega(\mathbf{x}) - x_i^2 + y_j^2 + \frac{\nu(\mathbf{x})^2 - \nu(\pi_{ij}(\mathbf{x}))^2}{m}$$

O(9)

T-stat computation per permutation O(4m+4n+20)

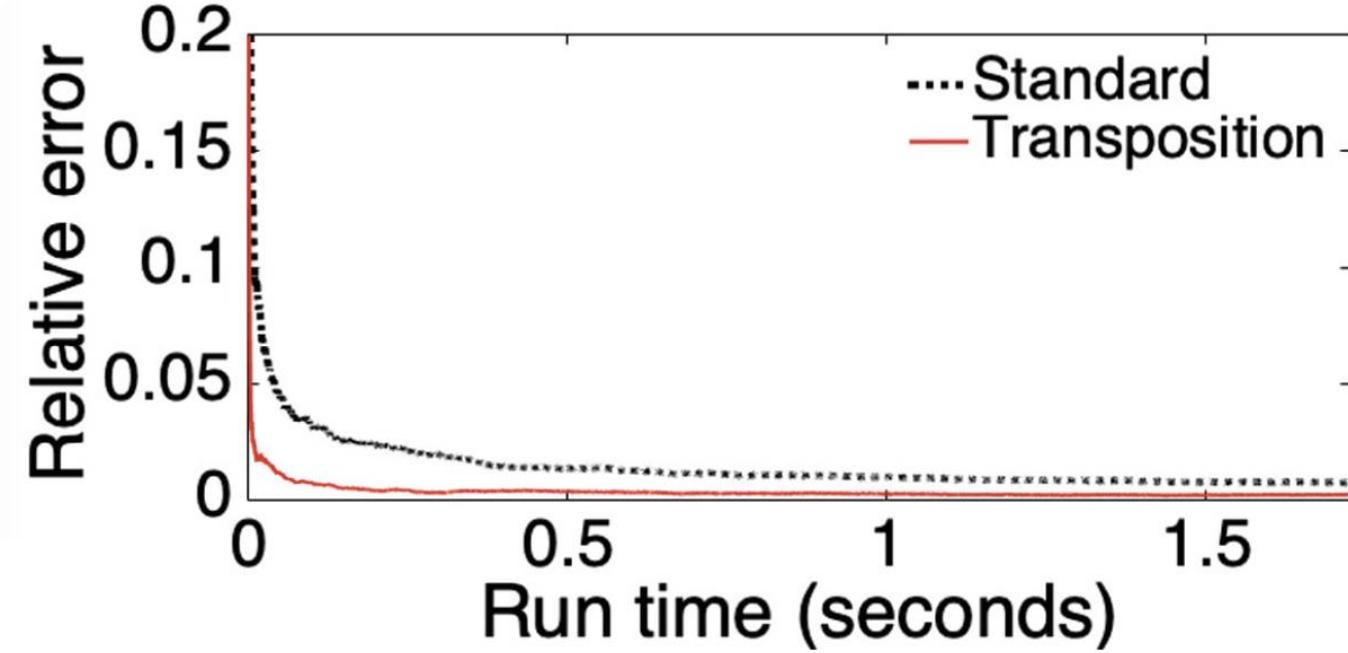
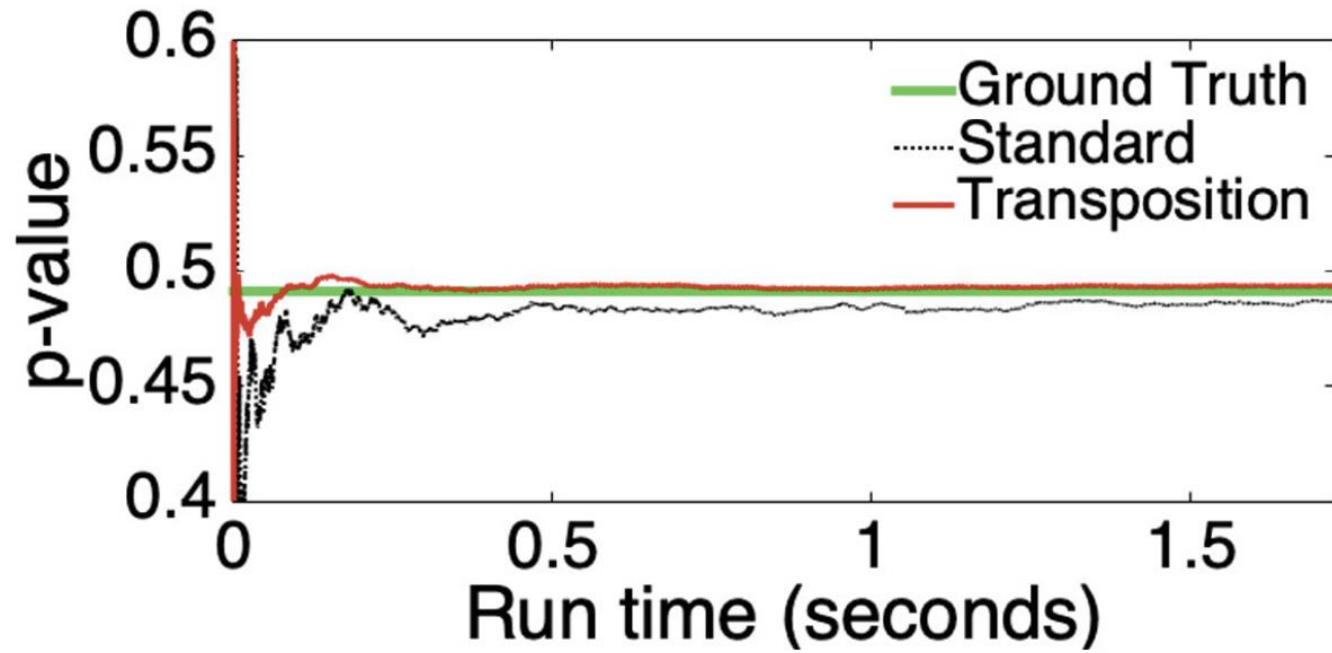
T-stat computation per transpositions O(35)

Simulation: Mixing proportion over transpositions



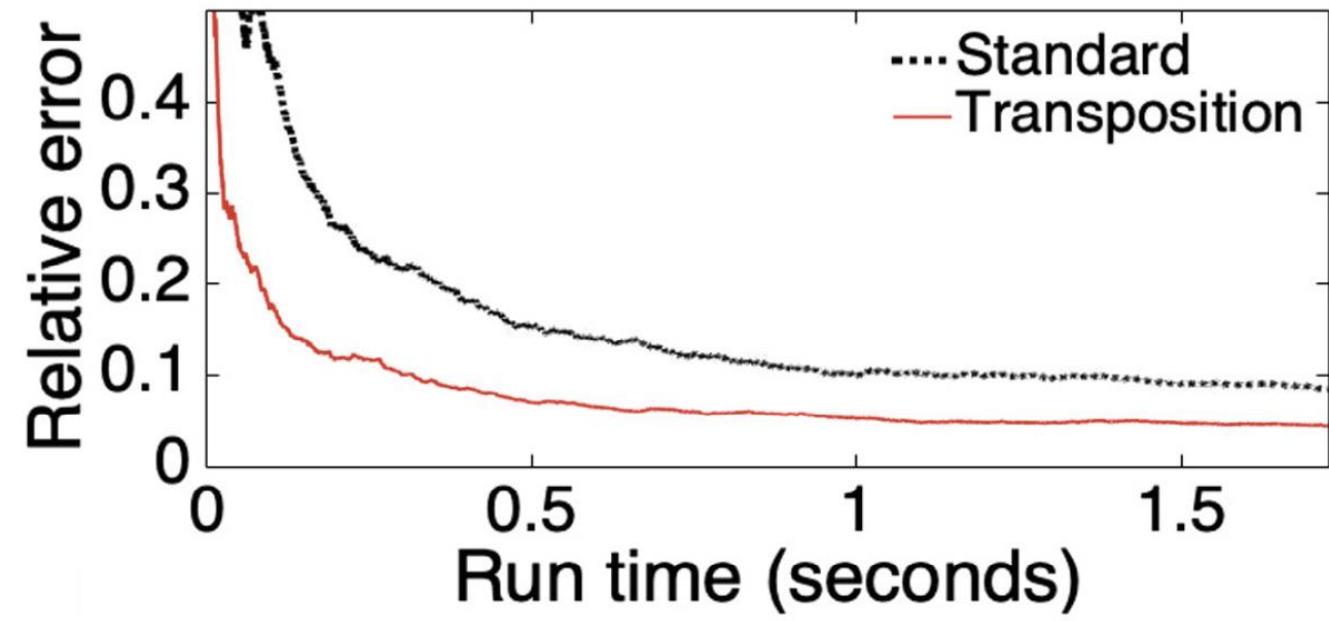
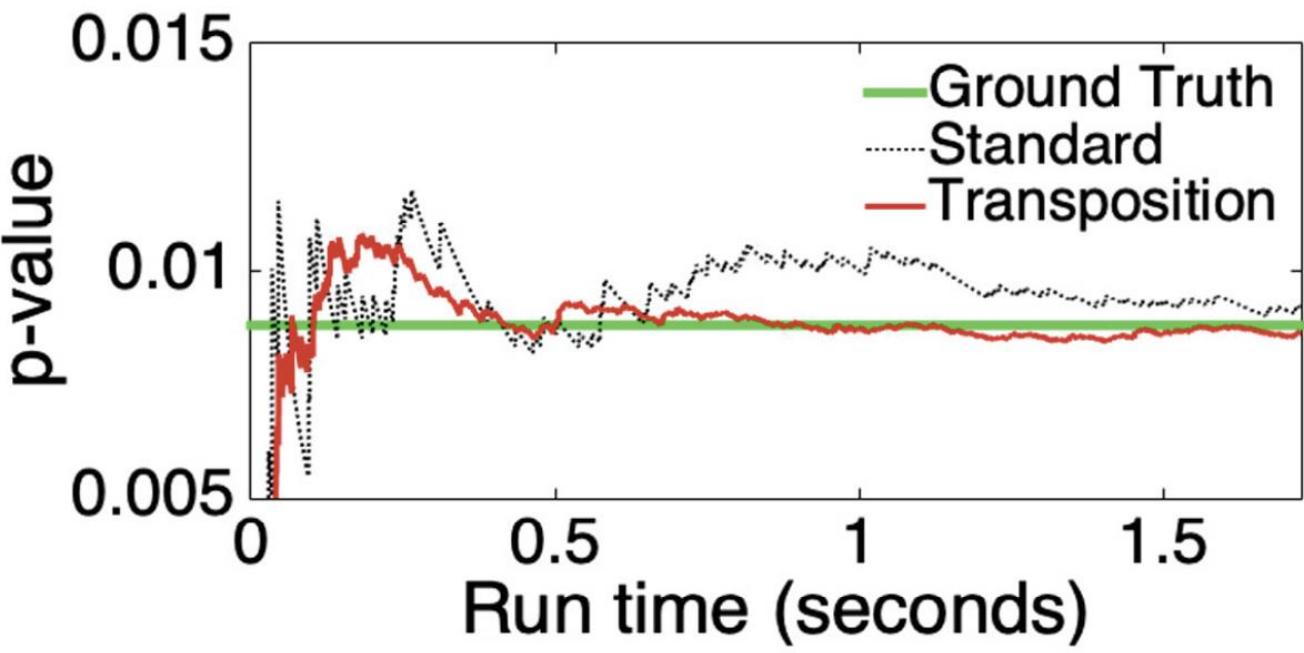
$m=n=10$

$$x_1, \dots, x_m \sim 0.1 + \text{Unif}(0, 1)$$
$$y_1, \dots, y_n \sim \text{Unif}(0, 1)$$



$m=n=100$

$$x_1, \dots, x_m \sim 0.1 + \text{Unif}(0, 1)$$
$$y_1, \dots, y_n \sim \text{Unif}(0, 1)$$



Matlab code

<http://www.stat.wisc.edu/~mchung/transpositions>

<https://github.com/laplcebeltrami/transpositions>

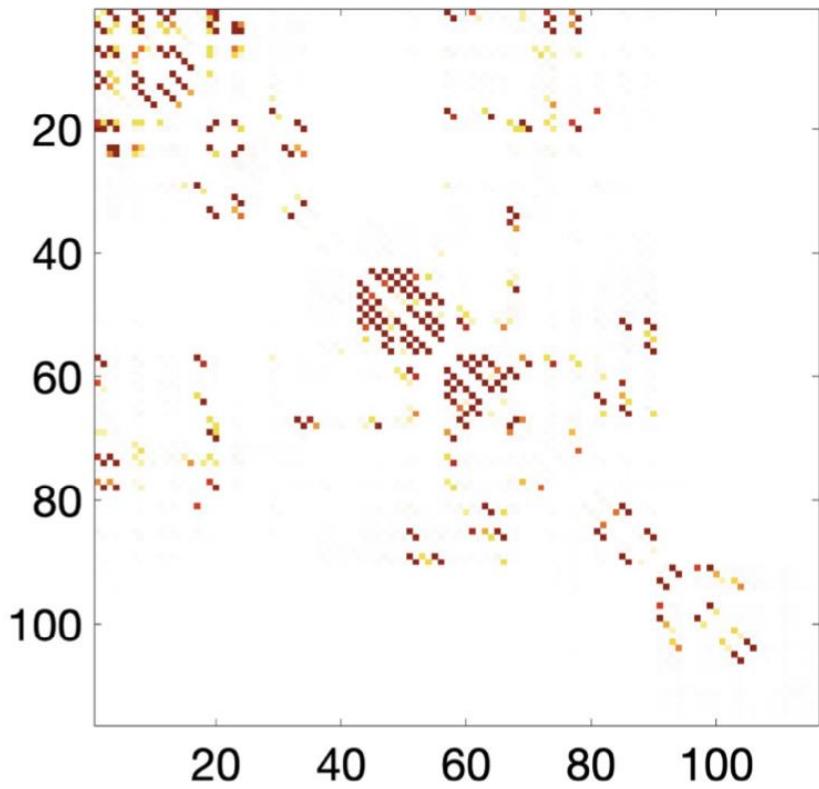
Permutation test

```
[stat_s, time_s] = test_permute (x,  
, y, per_s)
```

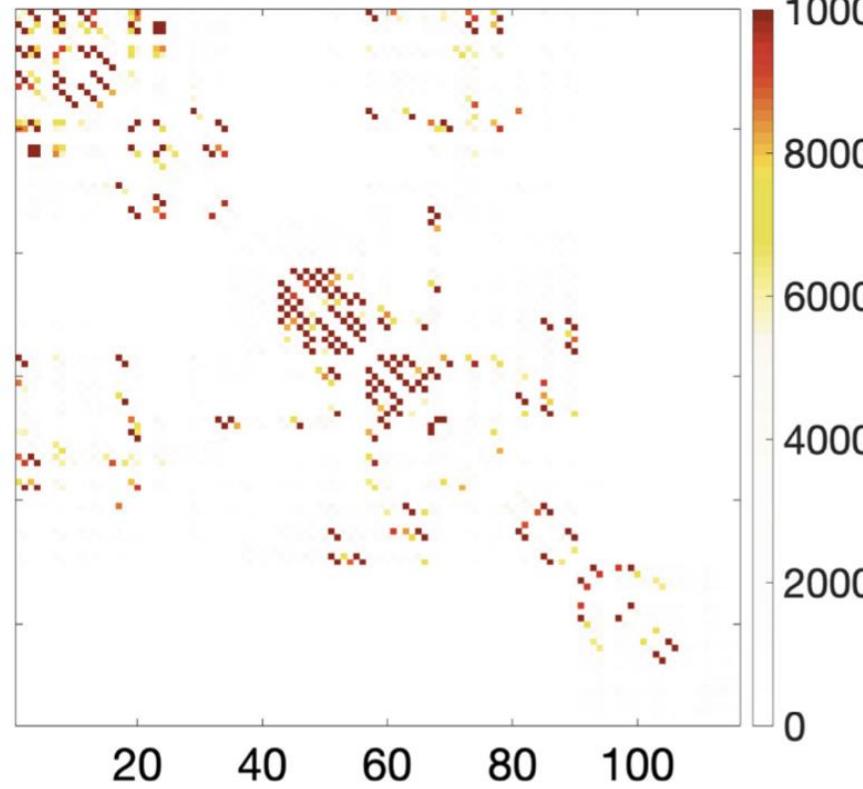
Transposition test

```
stat_t=[];  
for i=1:10000  
    [stat, time] = test_transpose (x, y,  
per_t / 10000);  
    stat_t=[stat_t; stat];  
    time_t=time_t + time;  
end
```

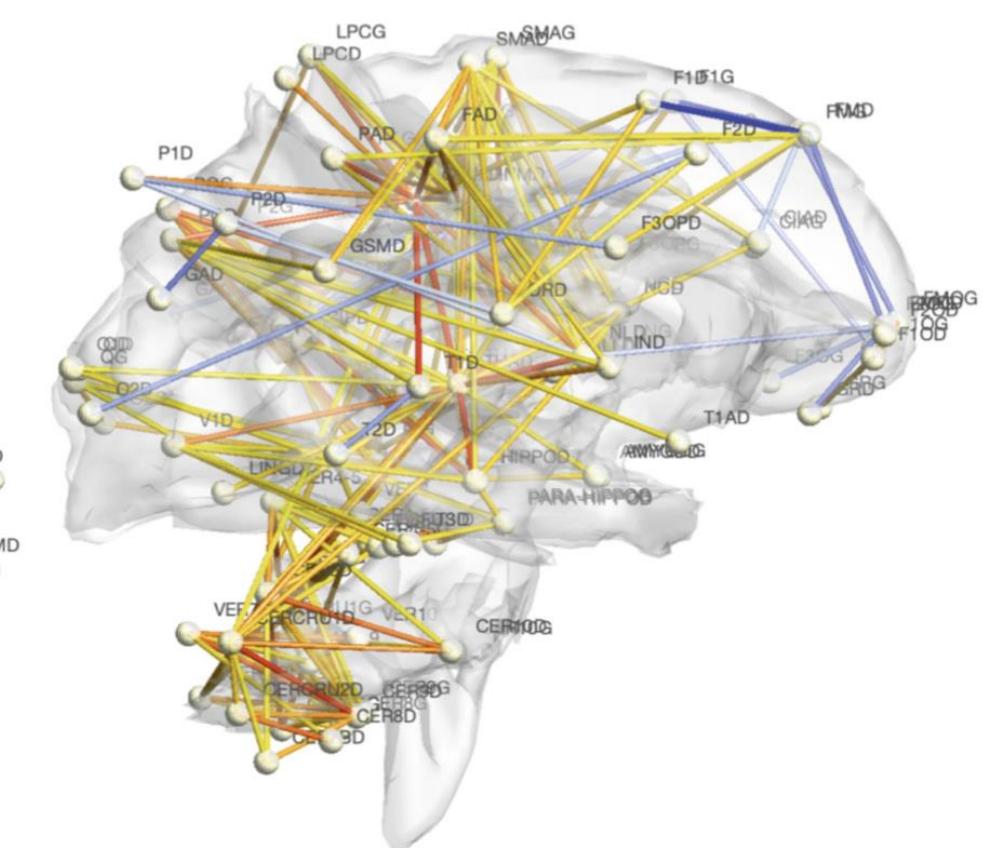
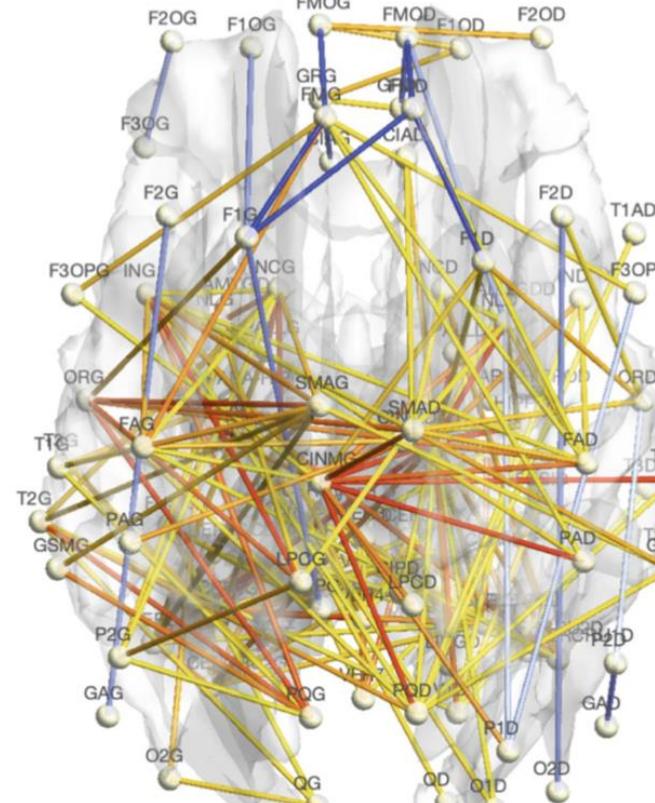
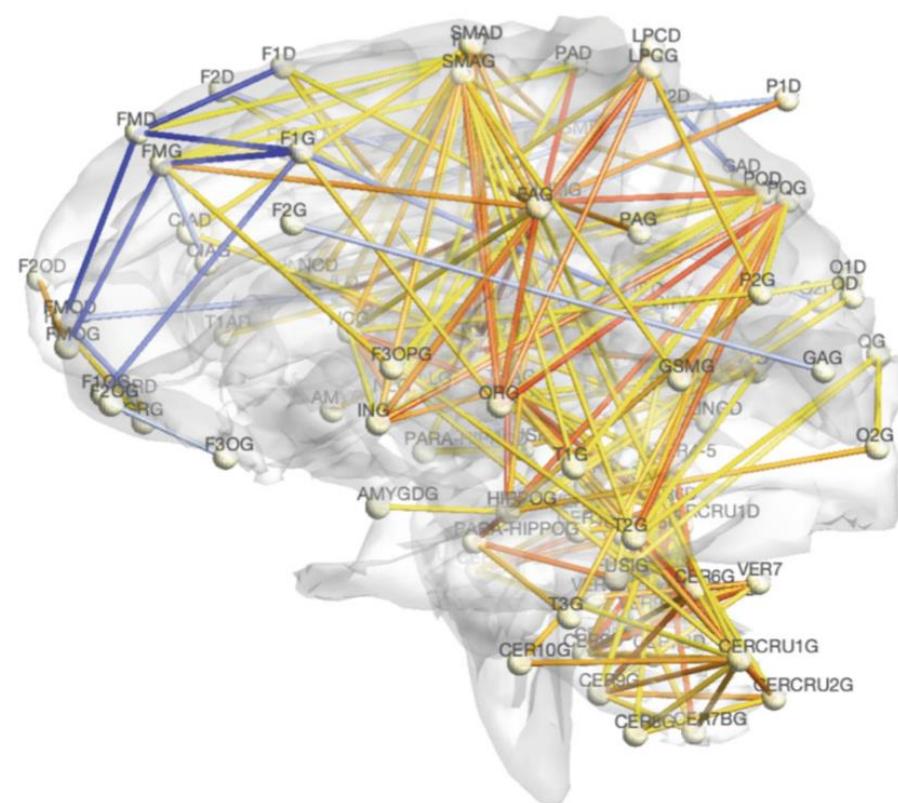
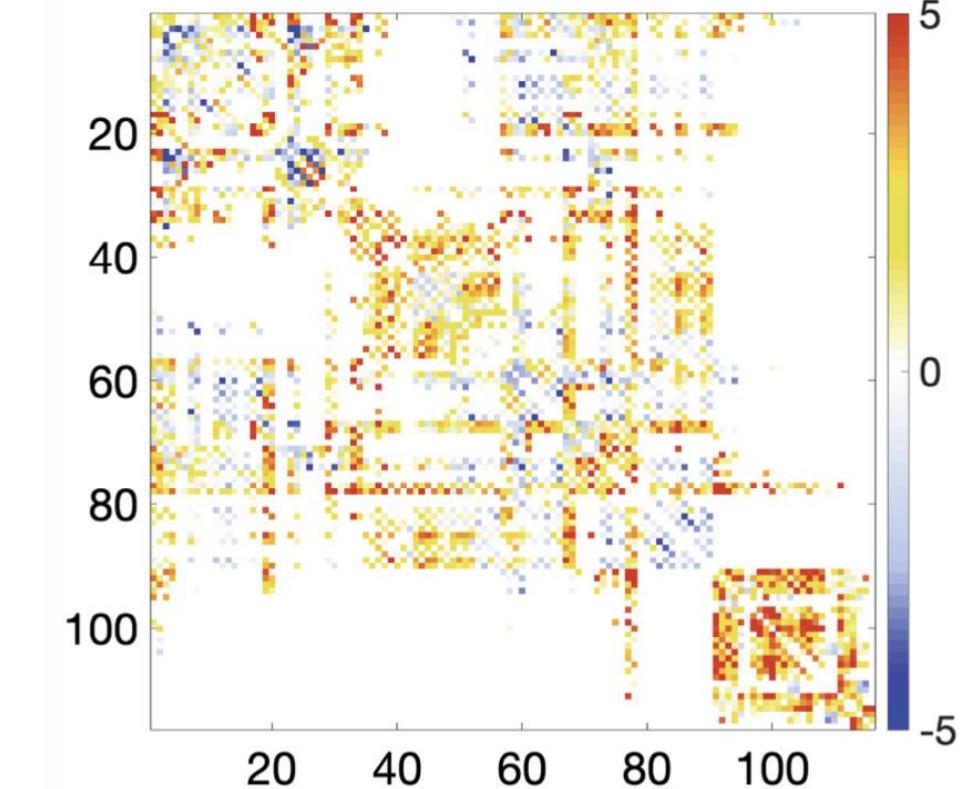
Female



Male



t-stat.



t-stat (202 females – 154 males)

Transposition test on Wasserstein distance

Songdechakraiwut and Chung 2023 Annals of Applied Statistic

2-Wasserstein distance between persistent diagrams

Random variables:

$$X \sim f_1 \quad Y \sim f_2$$

2-Wasserstein distance:

$$\mathcal{D}(X, Y) = \left(\inf \mathbb{E} \|X - Y\|^2 \right)^{1/2}$$

Persistent diagrams

$$P_1 = \{x_1, \dots, x_q\} \subset \mathbb{R}^2$$

$$P_2 = \{y_1, \dots, y_q\} \in \mathbb{R}^2$$

Empirical distributions

$$f_1(x) = \frac{1}{q} \sum_{i=1}^q \delta(x - x_i)$$

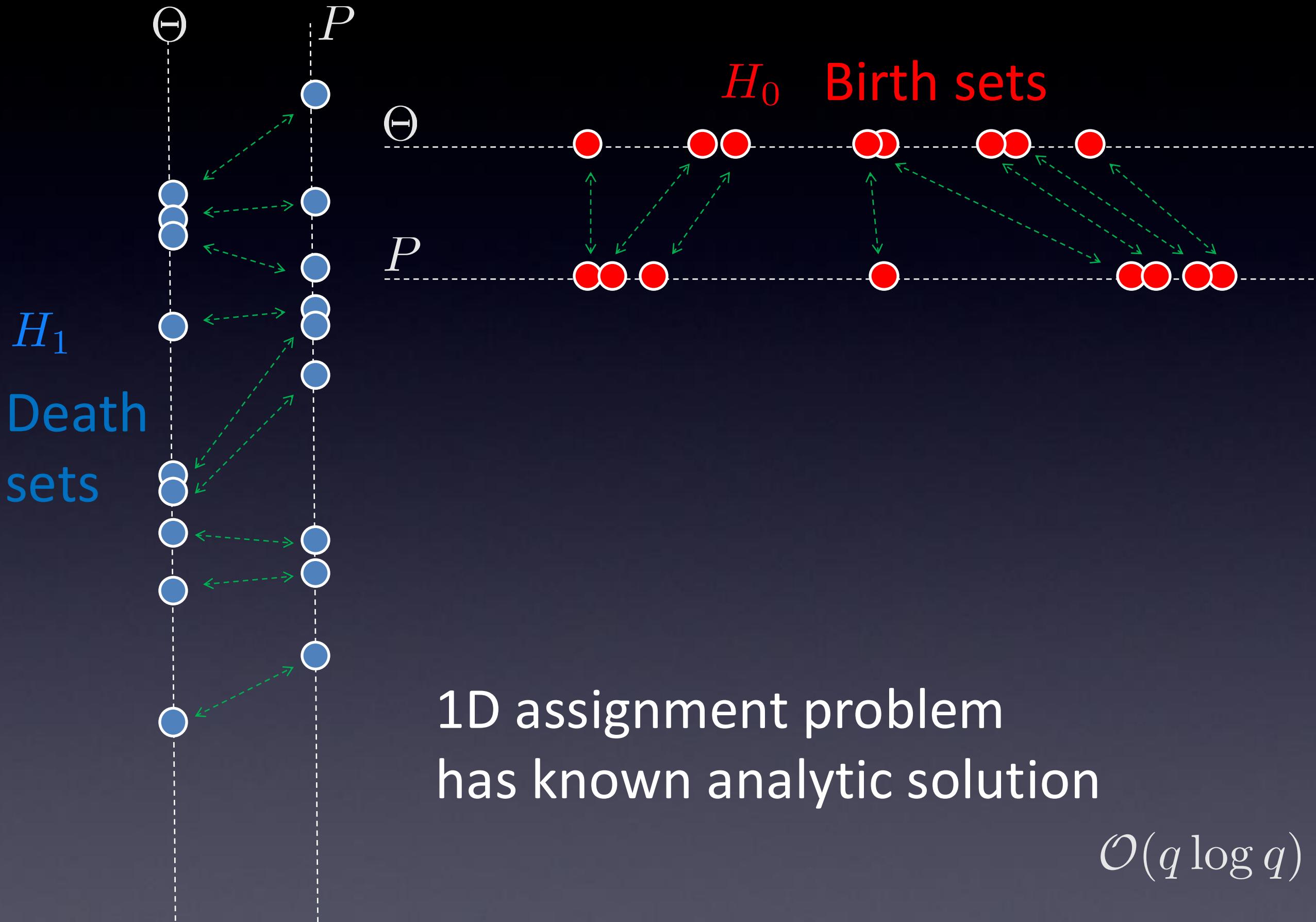
$$f_2(y) = \frac{1}{q} \sum_{i=1}^q \delta(y - y_i)$$



$$\mathcal{L}(P_1, P_2) = \inf_{\psi: P_1 \rightarrow P_2} \left(\sum_{x \in P_1} \|x - \psi(x)\|^2 \right)^{1/2}$$

Assignment problem: Hungarian algorithm $\mathcal{O}(q^3)$

Wasserstein distance for graph filtrations



Theorem: Wasserstein distance on graph filtrations

$$\begin{aligned}\mathcal{L}_{0D}(\Theta, P) &= \min_{\tau} \sum_{b \in E_0} [b - \tau(b)]^2 \\ &= \sum_{b \in E_0} [b - \tau_0^*(b)]^2\end{aligned}$$

τ_0^* :The i -th smallest birth value to the i -th smallest birth value

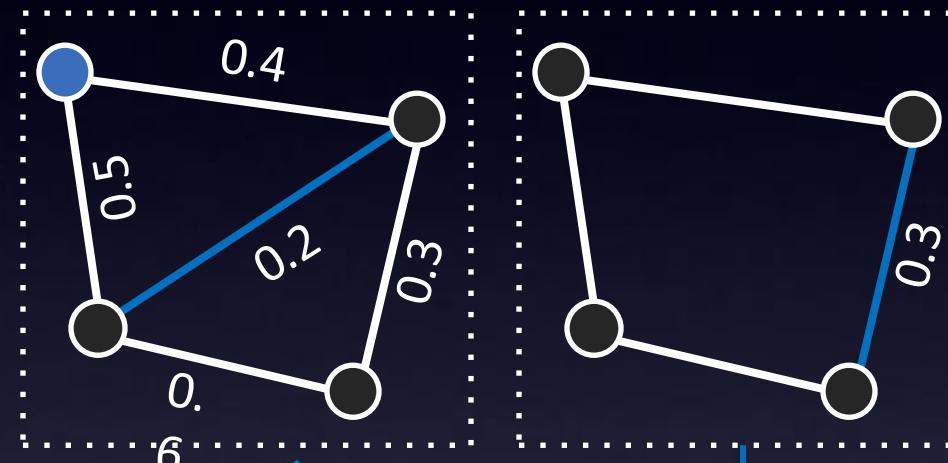
$$\begin{aligned}\mathcal{L}_{1D}(\Theta, P) &= \min_{\tau} \sum_{d \in E_1} [d - \tau(d)]^2 \\ &= \sum_{d \in E_1} [d - \tau_1^*(d)]^2\end{aligned}$$

τ_1^* :The i -th smallest death value to the i -th smallest death value

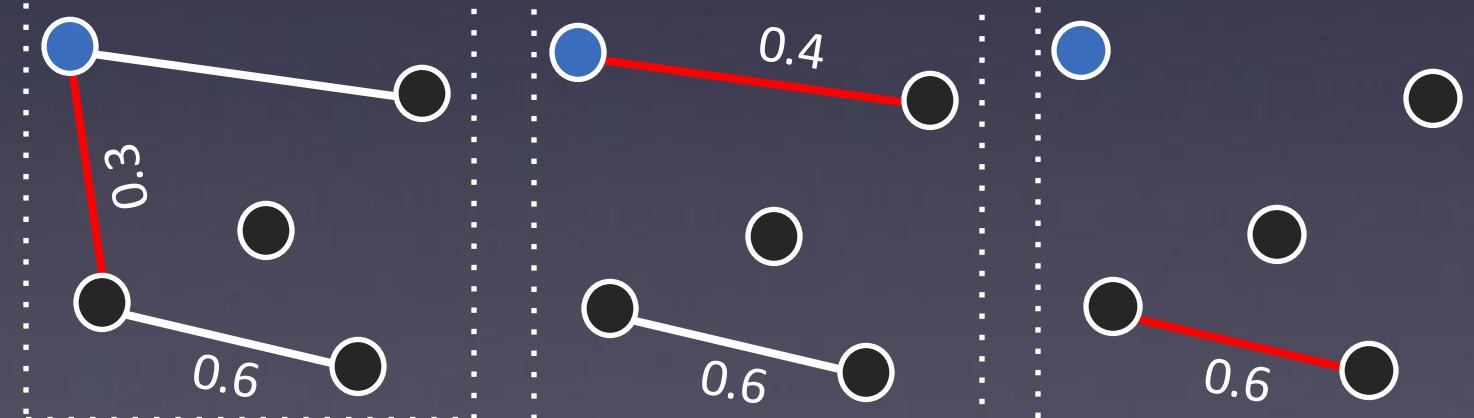
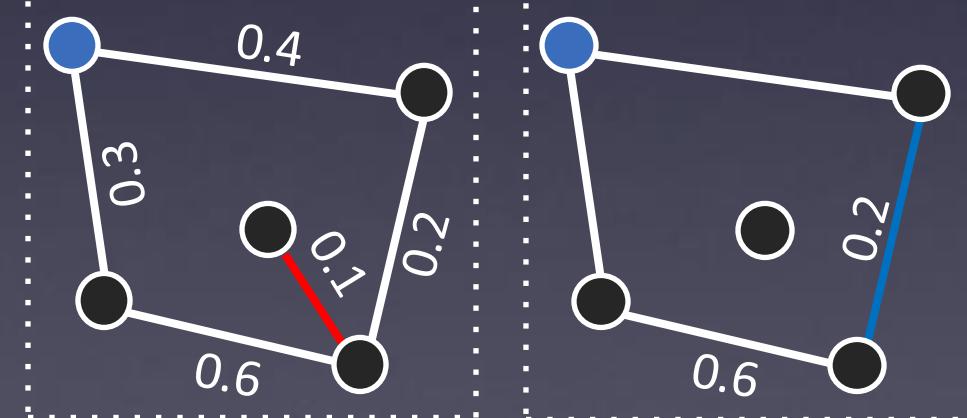
Graph matching by minimizing Wasserstein distance

$$\mathcal{L}(\Theta, P) = \mathcal{L}_{0D}(\Theta, P) + \mathcal{L}_{1D}(\Theta, P)$$

H_1 Edges destroy cycles

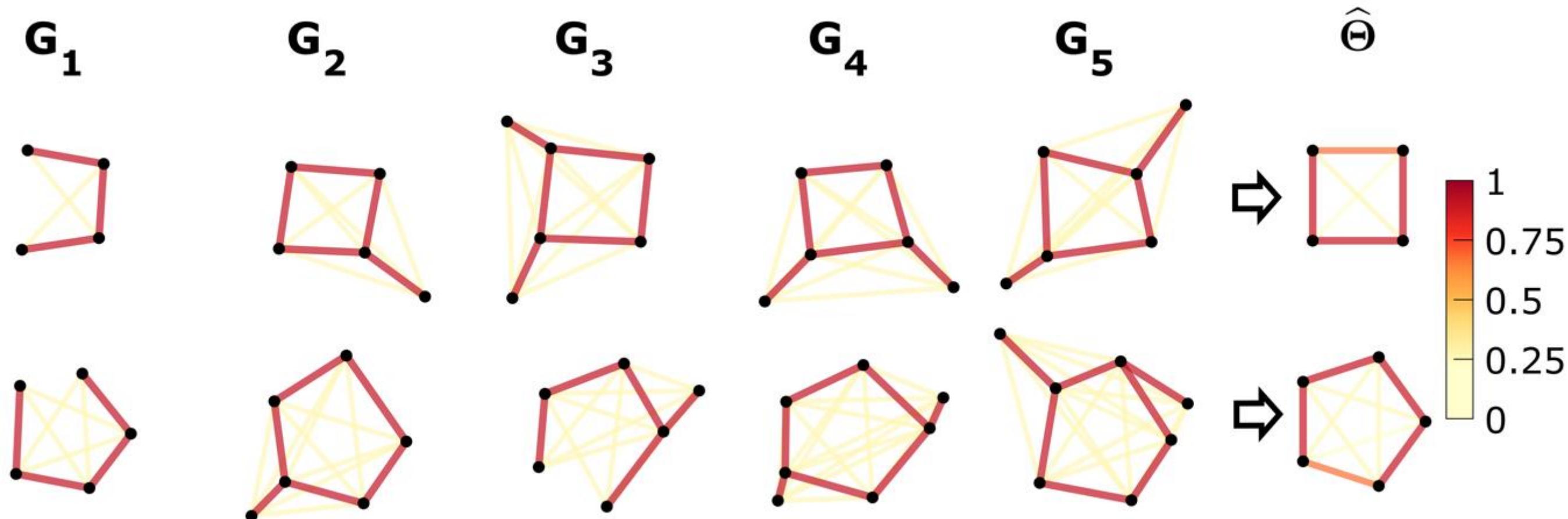


H_0 Edges create components



Graph matching → Topological mean of graphs

Death values of Θ are given by averaging the sorted k death values of all the networks G_k .

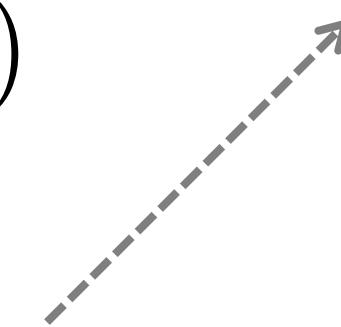


Topological averaging: $G_1 + G_2 + \dots + G_n$

ANOVA-like stat for Wasserstein distances

Between-group distance

$$l_B \propto \sum_{i \in C_1, j \in C_2} \mathcal{L}(G_i, G_j)$$

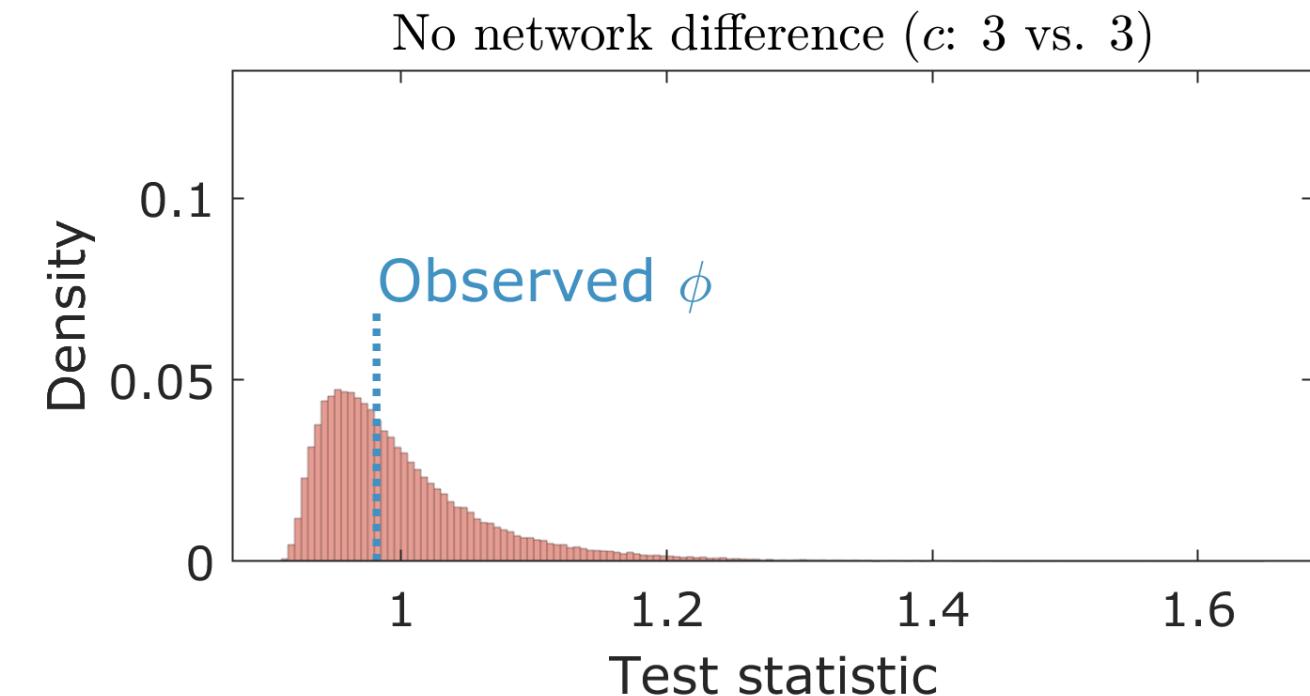
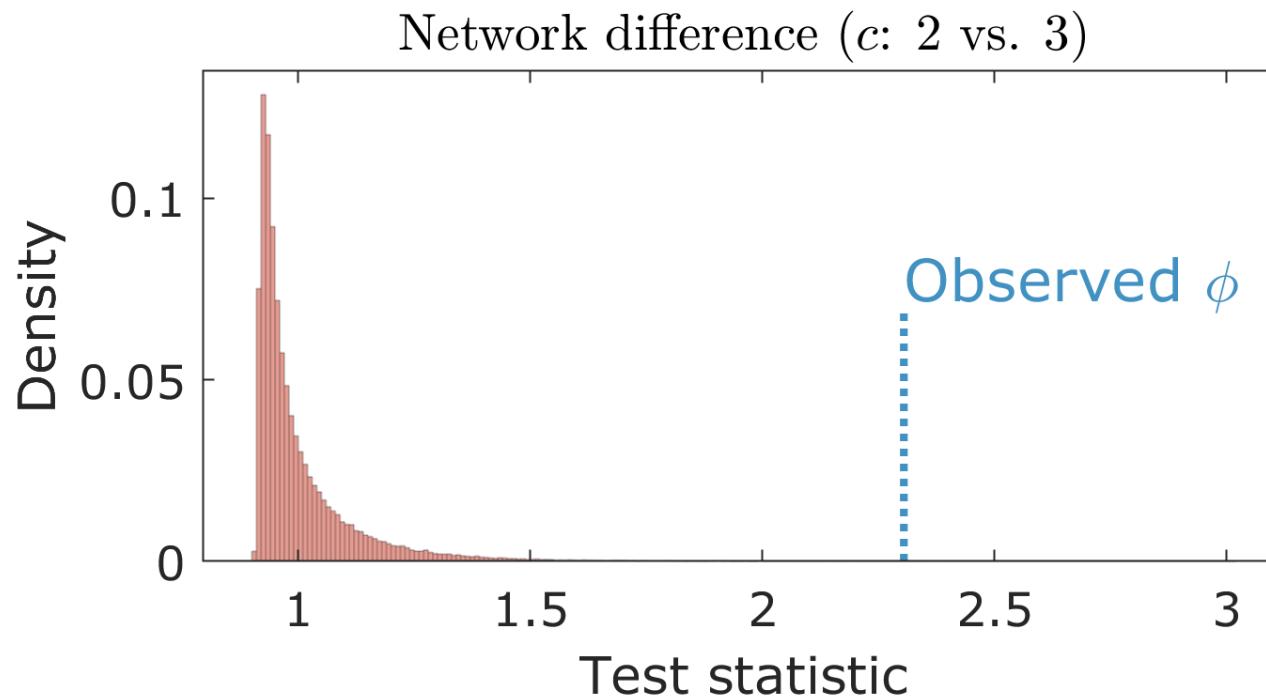


Statistic $\phi = \frac{l_B}{l_W}$

Within-group distance

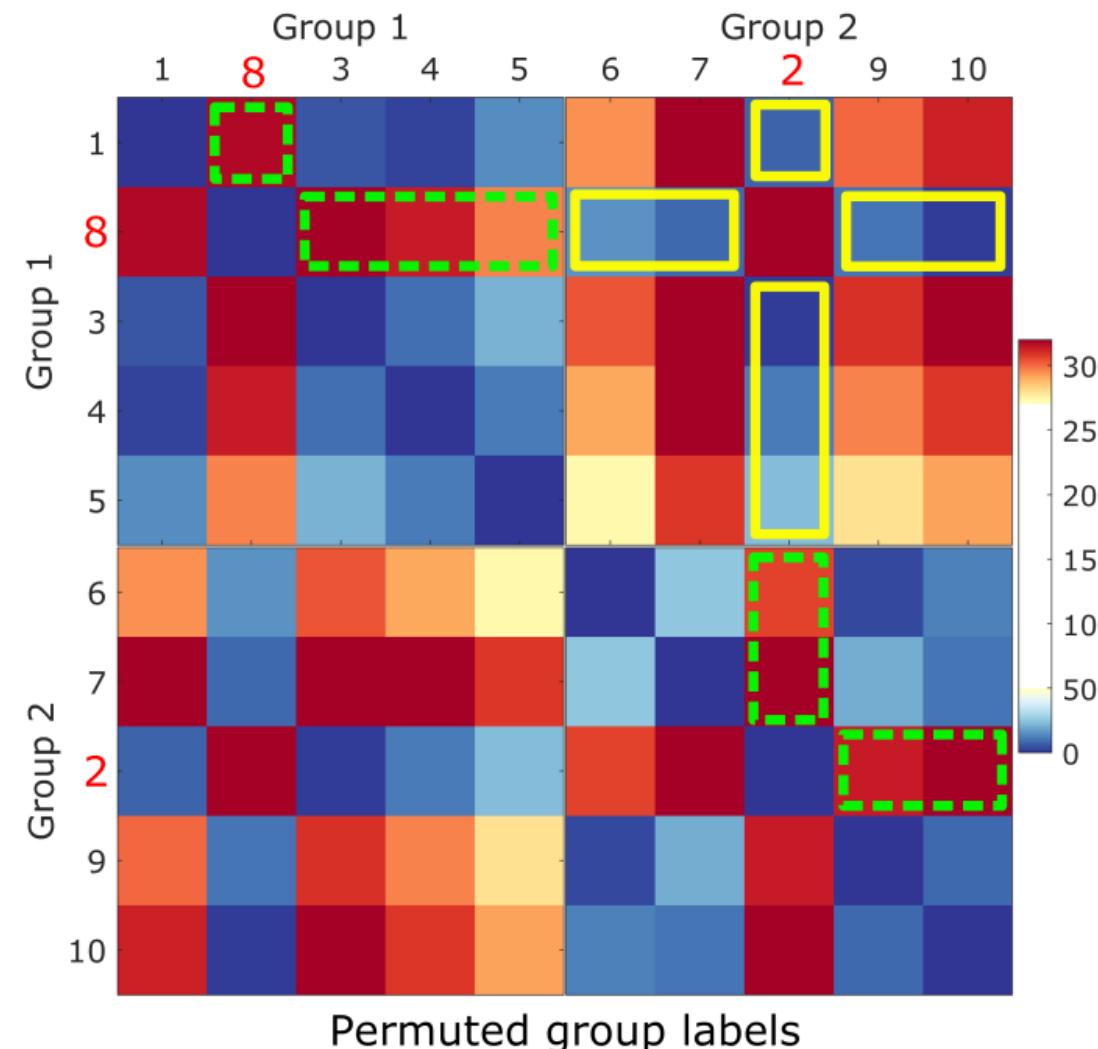
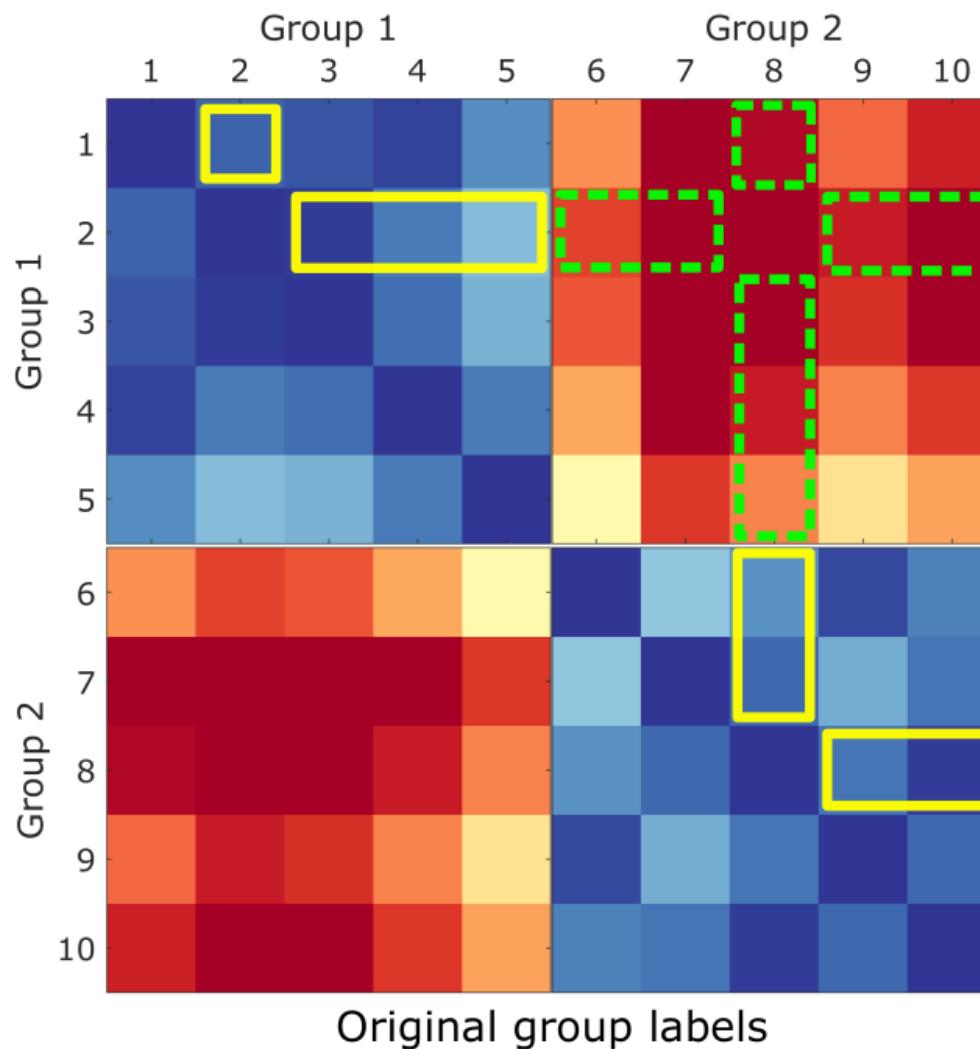
$$l_W \propto \sum_k \sum_{i, j \in C_k} \mathcal{L}(G_i, G_j) \longrightarrow$$

Wasserstein clustering
Chung et al. 2022



Transposition test on Wasserstein distance

Subject 2 in group 1 transposed with subject 8 in group 2



pdist2.m
computes
pairwise
Euclidean
distance

Simply increment changes of loss functions over transposition

$$\bar{\mathcal{L}}_W \rightarrow \bar{\mathcal{L}}_W + \Delta(\text{transposition})$$

$$\bar{\mathcal{L}}_B \rightarrow \bar{\mathcal{L}}_B + \Delta(\text{transposition})$$

Matlab code:

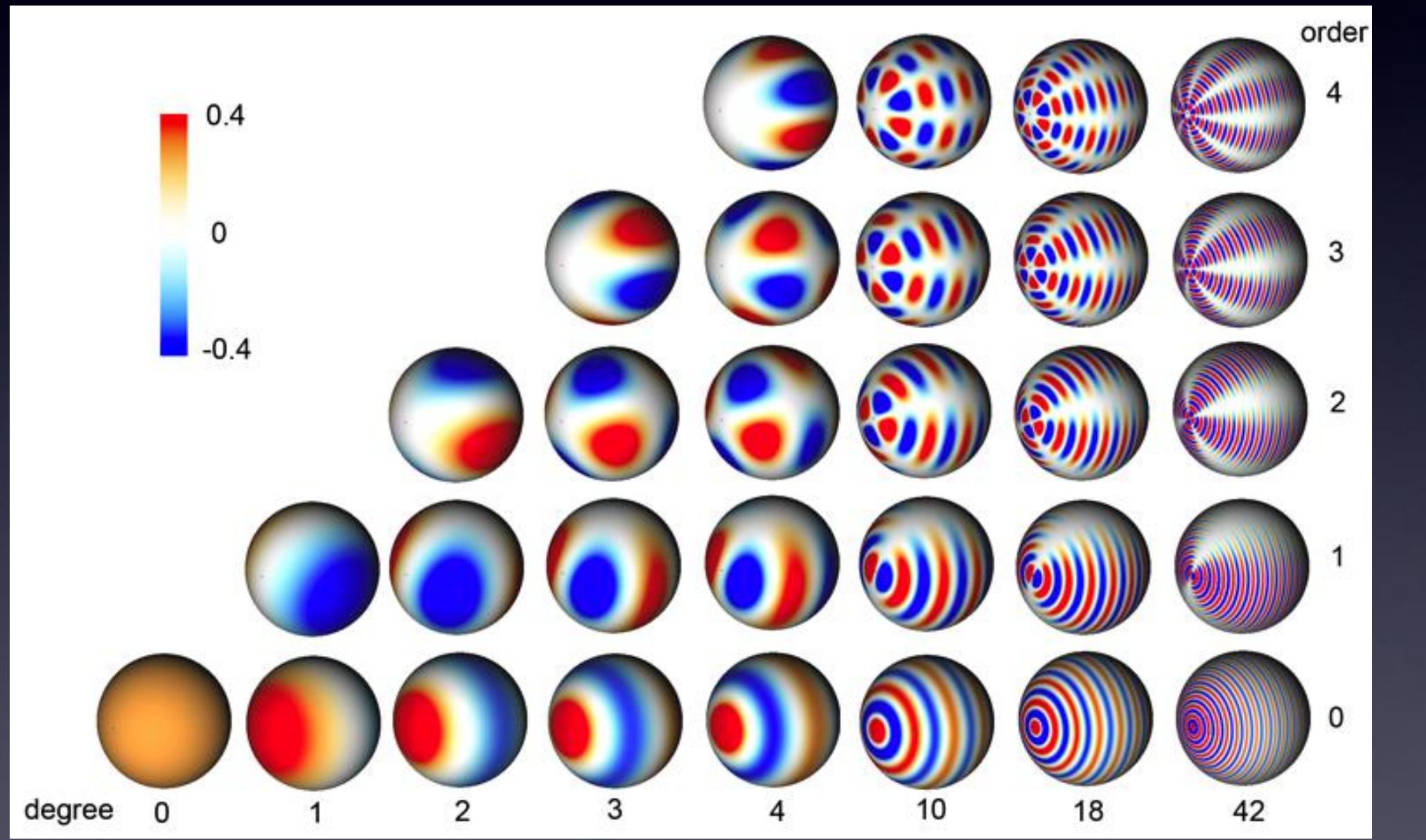
<https://www.stat.wisc.edu/~mchung/dynami>

Iterative residual fitting algorithm

This is well known method in data science with many different names and variations. All use the orthogonal properties of Fourier expansion

Spherical harmonic of degree l and order m

$$Y_{lm} = \begin{cases} c_{lm} P_l^{|m|}(\cos \theta) \sin(|m|\varphi), & -l \leq m \leq -1, \\ \frac{c_{lm}}{\sqrt{2}} P_l^0(\cos \theta), & m = 0, \\ c_{lm} P_l^{|m|}(\cos \theta) \cos(|m|\varphi), & 1 \leq m \leq l, \end{cases}$$



Coarse detail

Fine detail

- For measurements $f(p_1), f(p_2), \dots, f(p_n)$, ($n > 46,000$), we set up normal equations:



$$f(p_i) = \sum_{l=0}^k \sum_{m=-l}^l \beta_{lm} Y_{lm}(p_i).$$

i-th mesh vertex

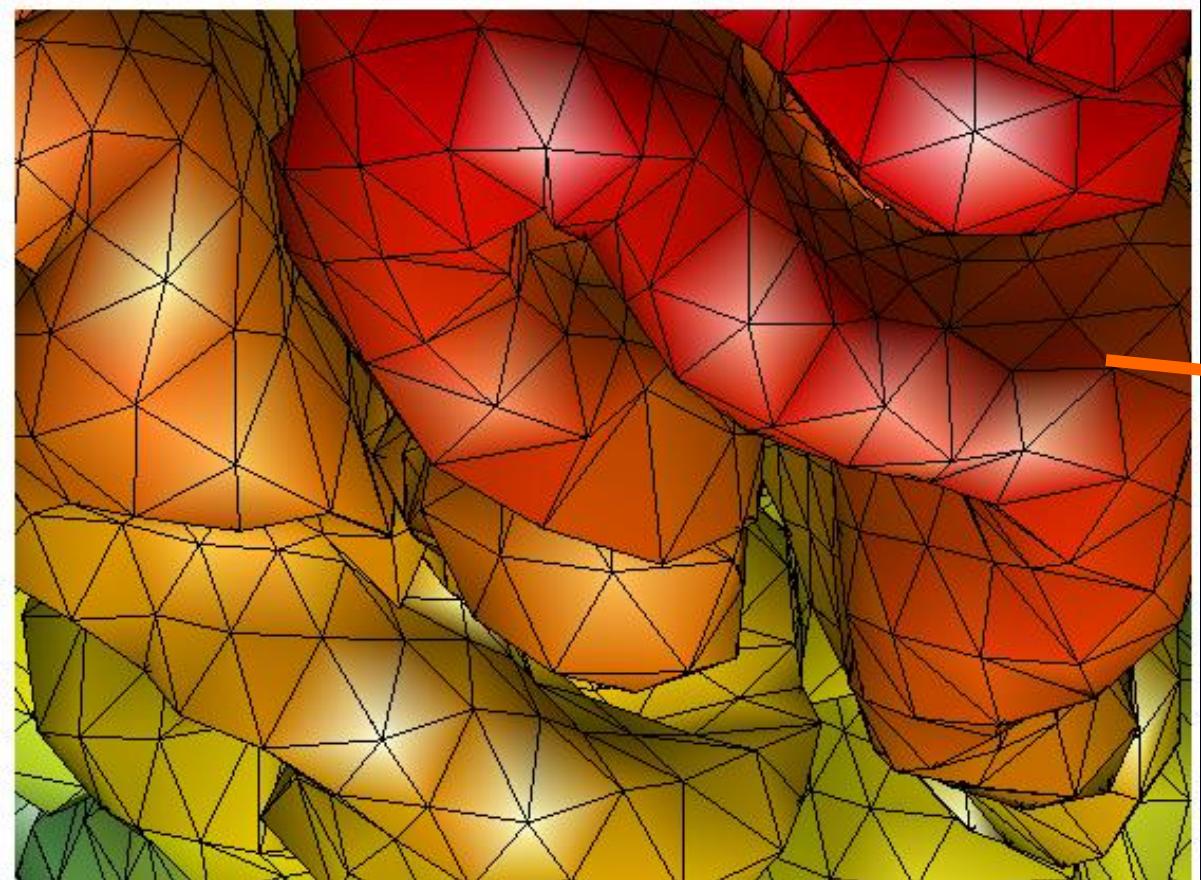
- Matrix form:

$$\underbrace{\begin{pmatrix} f(p_1) \\ f(p_2) \\ \vdots \\ f(p_n) \end{pmatrix}}_{\mathbf{F}} = \underbrace{\begin{pmatrix} Y_{00}(p_1) & Y_{1-1}(p_1) & Y_{10}(p_1) & \cdots & Y_{kk}(p_1) \\ Y_{00}(p_2) & Y_{1-1}(p_2) & Y_{10}(p_2) & \cdots & Y_{kk}(p_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ Y_{00}(p_n) & Y_{1-1}(p_n) & Y_{10}(p_n) & \cdots & Y_{kk}(p_n) \end{pmatrix}}_{\mathbf{Y}} \underbrace{\begin{pmatrix} \beta_{00} \\ \beta_{1-1} \\ \vdots \\ \beta_{kk} \end{pmatrix}}_{\boldsymbol{\beta}}$$

300000 vertices x 10000 basis

Estimation: $\hat{\boldsymbol{\beta}} = (\mathbf{Y}'\mathbf{Y})^{-1}\mathbf{Y}'\mathbf{F}$.

LB-eigenfunction expansion



i-th vertex



$$f(p_i) = \sum_{j=0}^k \beta_j \psi_j(p_i)$$



$$\mathbf{f} = (f(p_1), \dots, f(p_n))'$$

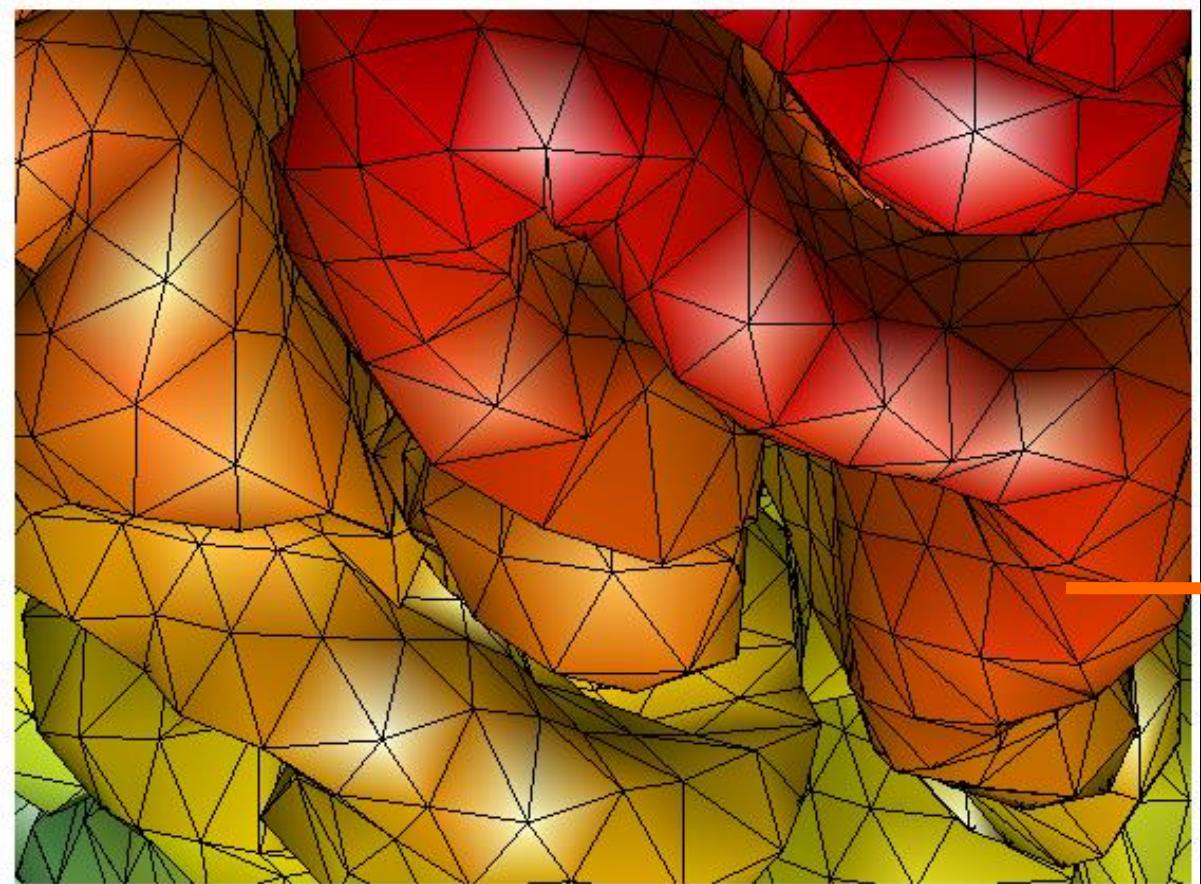
$$\boldsymbol{\beta} = (\beta_0, \dots, \beta_k)'$$

$$\mathbf{Y} = \begin{bmatrix} \psi_0(p_1) & \cdots & \psi_k(p_1) \\ \vdots & \ddots & \vdots \\ \psi_0(p_n) & \cdots & \psi_k(p_n) \end{bmatrix}$$



$$\boldsymbol{\beta} = (\mathbf{Y}' \mathbf{Y})^{-1} \mathbf{Y}' \mathbf{f} \quad \leftarrow \quad \mathbf{f} = \mathbf{Y} \boldsymbol{\beta}$$

Parameter Estimation in more general case



i-th vertex



$$f(p_i) = \sum_{j=0}^k \beta_j \psi_j(p_i)$$



$$\mathbf{f} = (f(p_1), \dots, f(p_n))'$$

$$\boldsymbol{\beta} = (\beta_0, \dots, \beta_k)'$$

$$\mathbf{Y} = \begin{bmatrix} \psi_0(p_1) & \cdots & \psi_k(p_1) \\ \vdots & \ddots & \vdots \\ \psi_0(p_n) & \cdots & \psi_k(p_n) \end{bmatrix}$$



For $n > 300000$
and $k > (80+1)^2$,
it is not so easy to do
this
in a laptop with small
memory.

$$\boldsymbol{\beta} = (\mathbf{Y}' \mathbf{Y})^{-1} \mathbf{Y}' \mathbf{f}$$

Iterative residual fitting (IRF) algorithm

Step 1.
measurements

$$f(p_1), \dots, f(p_n)$$

Step 2. Set initial degree=0 $k = 0$

Step 3. Solve $f(p_i) = \sum_{m=-k}^k \beta_{km} Y_{km}(p_i)$

Project data
into a finite
subspace

Iterate

Step 3.5. $f \leftarrow f - \hat{f}$

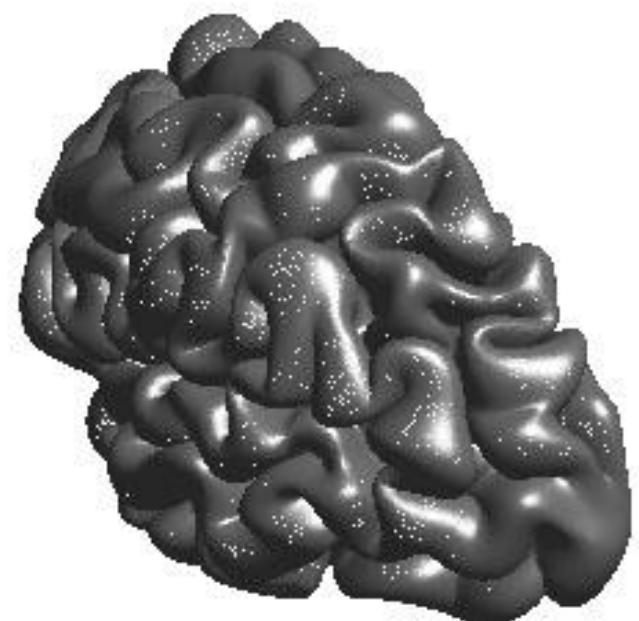
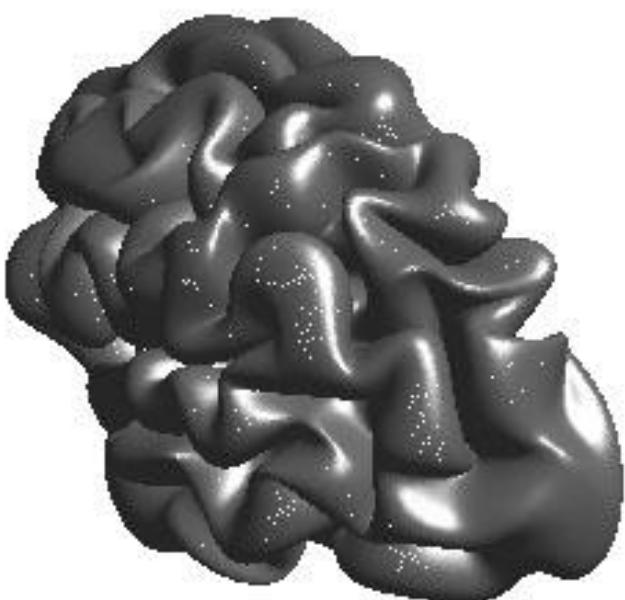
Once low frequency parts are
estimated, we throw them
away

Step 4. Set degree $k \leftarrow k + 1$

MATLAB:

www.stat.wisc.edu/~mchung/softwares/weighted-SPHARM/weighted-SPHARM.html

We recycle the previous estimates and improves the fit



0

10

20

30



40

50

60

70

Self assessment questions → Project

Suppose there are two groups of data and resulting Pearson correlations.



Perform the permutation test through by designing an online algorithm. Compare the result against the parametric approach (Fisher transform + standard normal) and standard permutation test

Solution given in *Chung et al. 2020*