University of Wisconsin
SCHOOL OF MEDICINE
AND PUBLIC HEALTH

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: $\mathbf{X}_i = (x_{i1}, \cdots, x_{ip})'$

assumption:

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

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

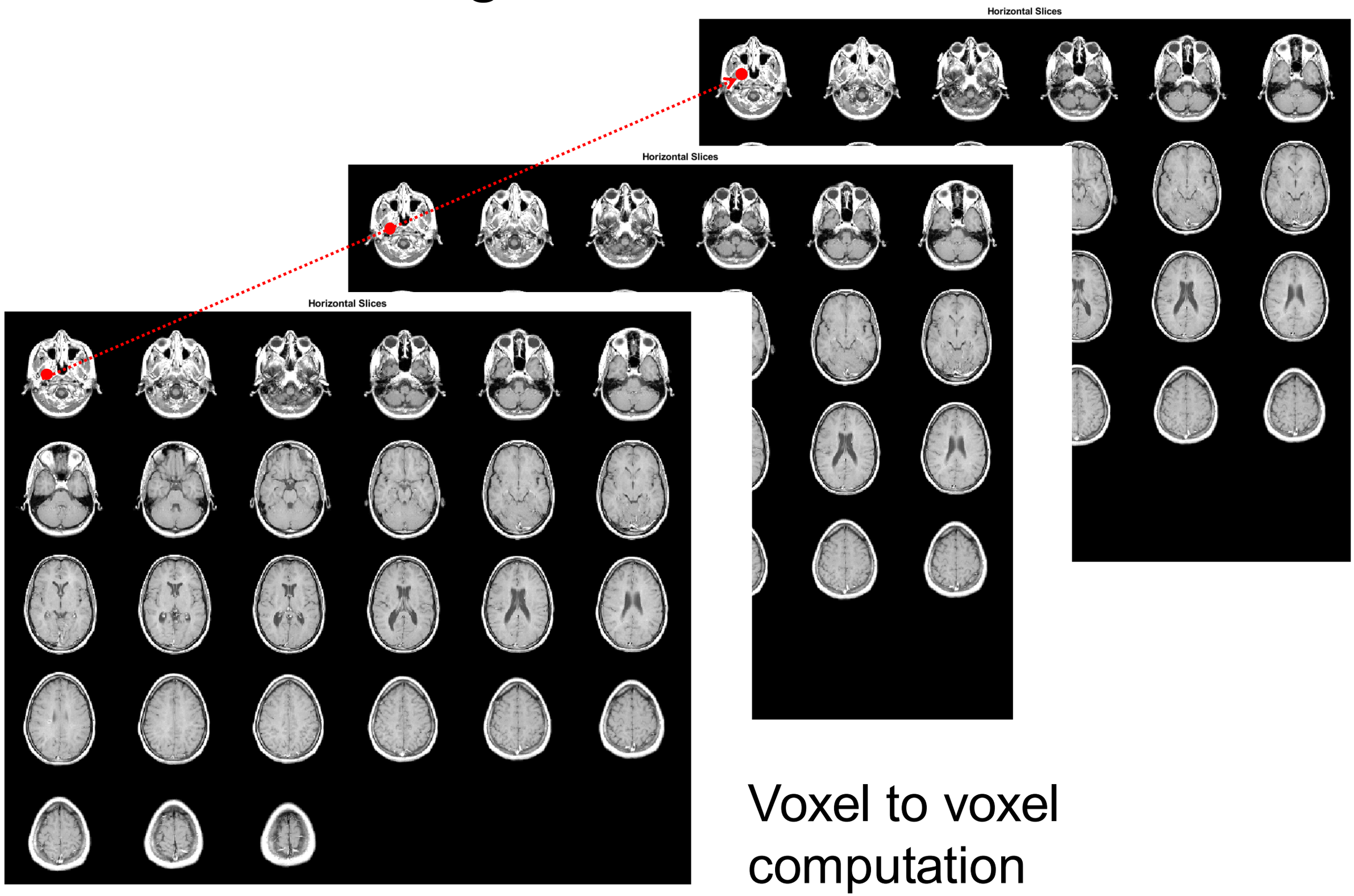
Rank deficiency in large-scale covariance matrix

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

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

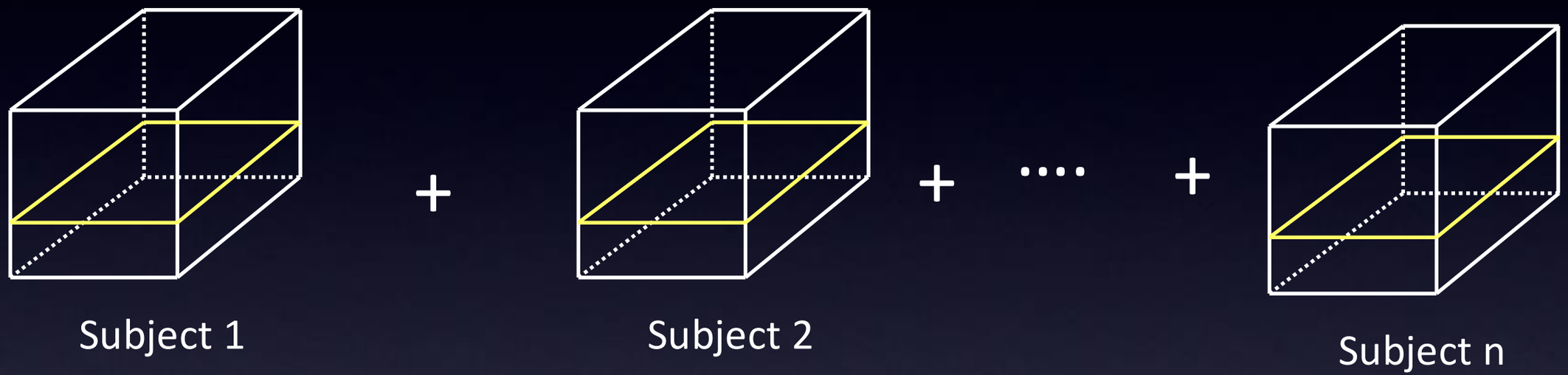
$$\Sigma = U^\top \begin{bmatrix} \eta_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \eta_2 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \eta_n & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{bmatrix} U$$

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



Voxel to voxel computation requires 3 loops

Existing method for computing large 3D medical images



$I_1(x, y, \cdot)$ $I_2(x, y, \cdot)$ $\cdots$ $I_n(x, y, \cdot)$

Load each slide of 3D image volume into computer

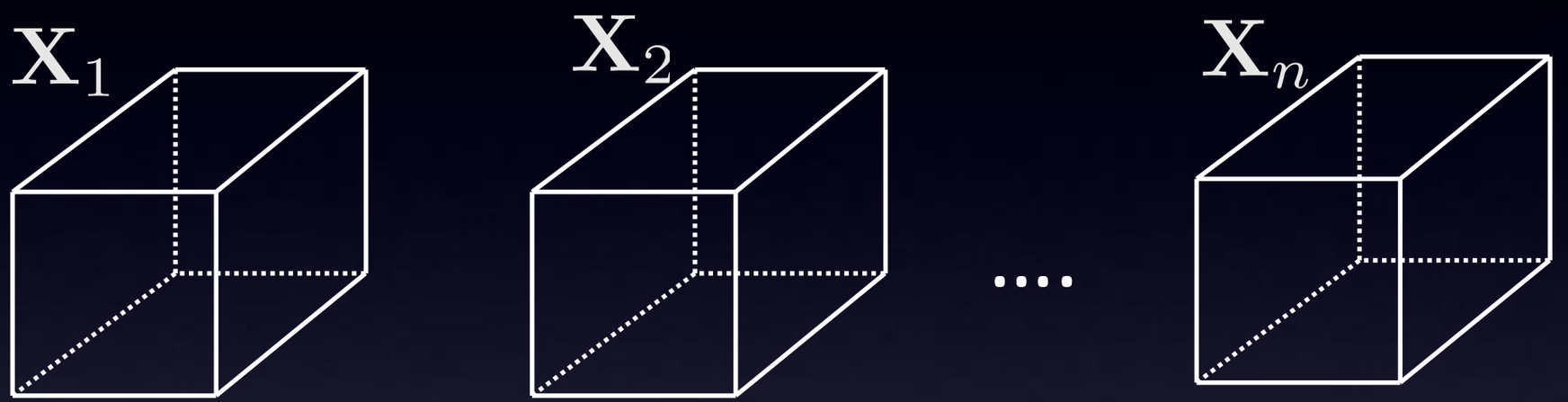
This will *not* work with collection of CT-images.

Online computation

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

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

Basic idea of online computation for large 3D medical images



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

Load one image at a time, compute

Online computation for sample mean

Given $x_1, \cdots, x_{m-1}$ 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

$$x_1, \cdots, x_m \qquad\qquad y_1, \cdots, y_n$$

Under null

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

$$T_{1,0} \to T_{2,0} \to \cdots \to T_{m,0} \to T_{m,1} \to \cdots \to T_{m,n}$$

Online computation for regression

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

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

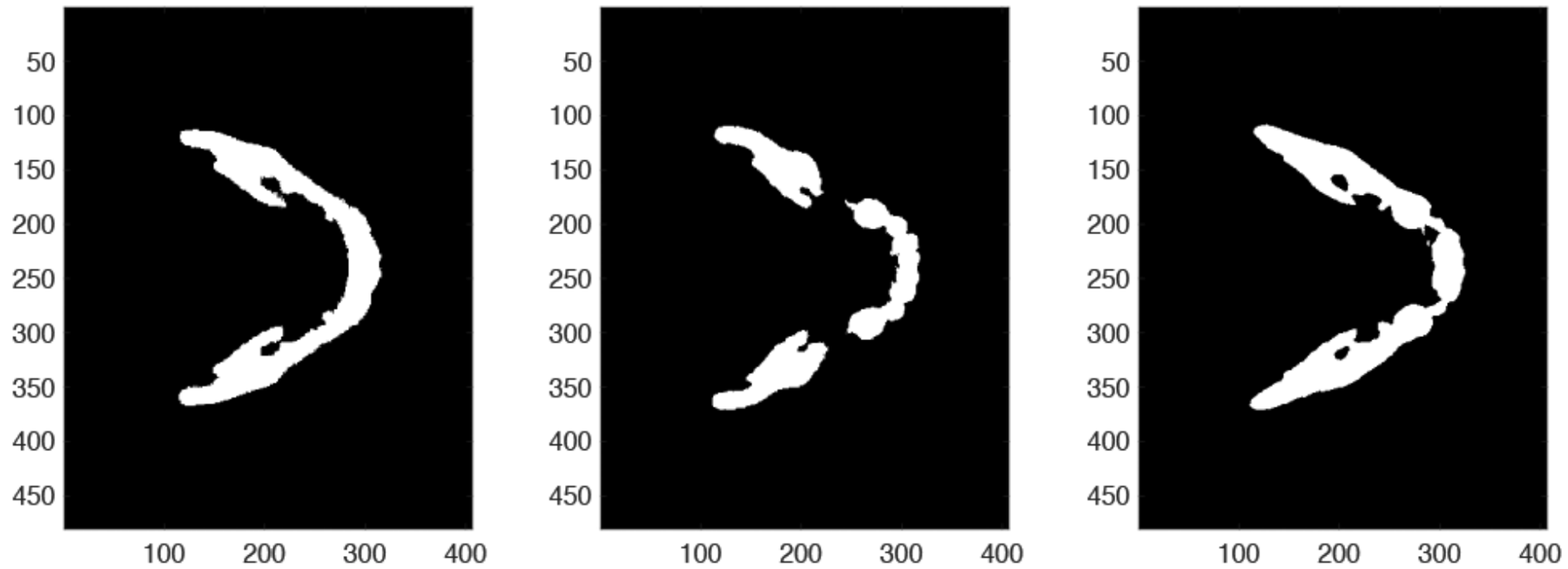
Based on Woodbury formula and Schur complement

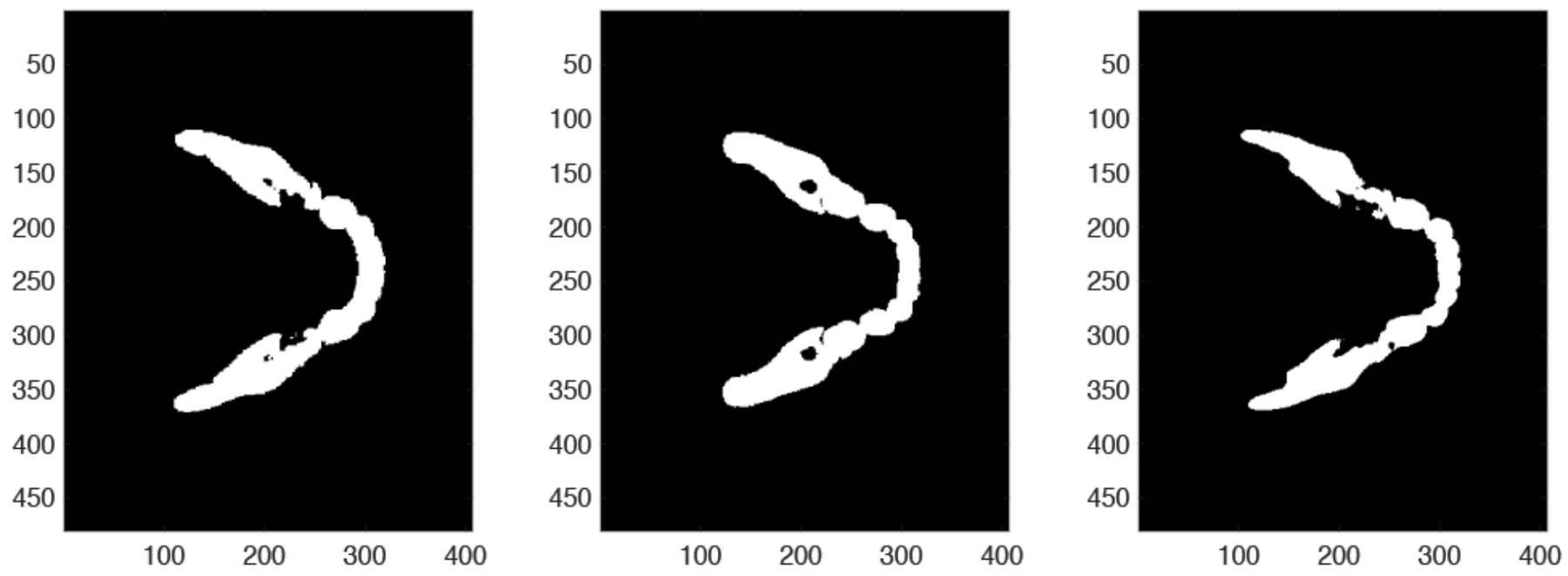
$$\boldsymbol{\lambda}_m = (I - W_{m-1}^{-1}z'_m y_m - c_m W_{m-1}^{-1}z'_m W'_{m-1})\boldsymbol{\lambda}_{m-1} - c_m W_{m-1}^{-1}z'_m z'_m y_m$$

Chung et al. 2017 MICCAI, Online statistical inference

Problem: online algorithm for least squares

Application to large CT dataset





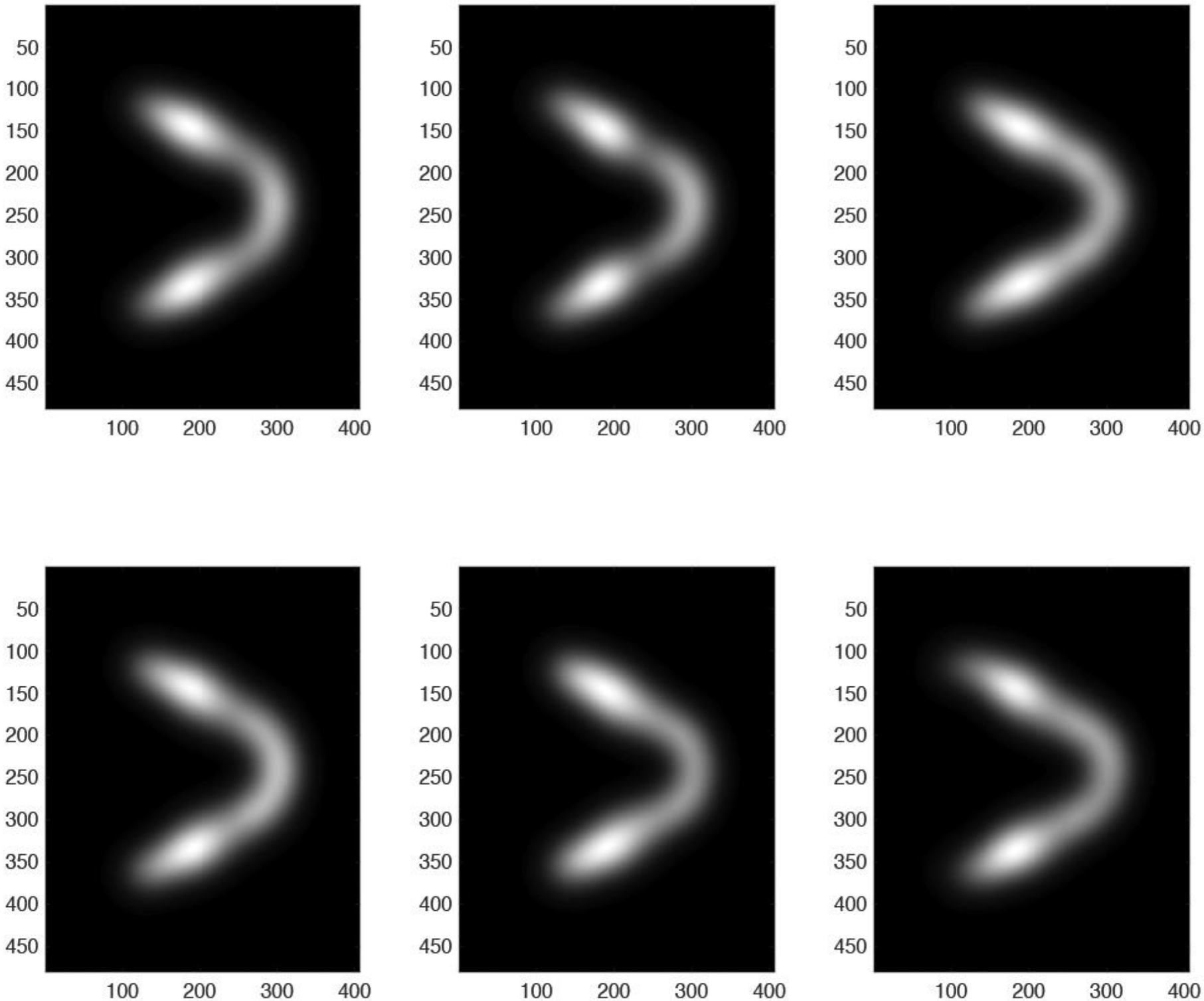
4^3=64 times bigger image than MRI

160 males

130 females

Voxel-based morphometry

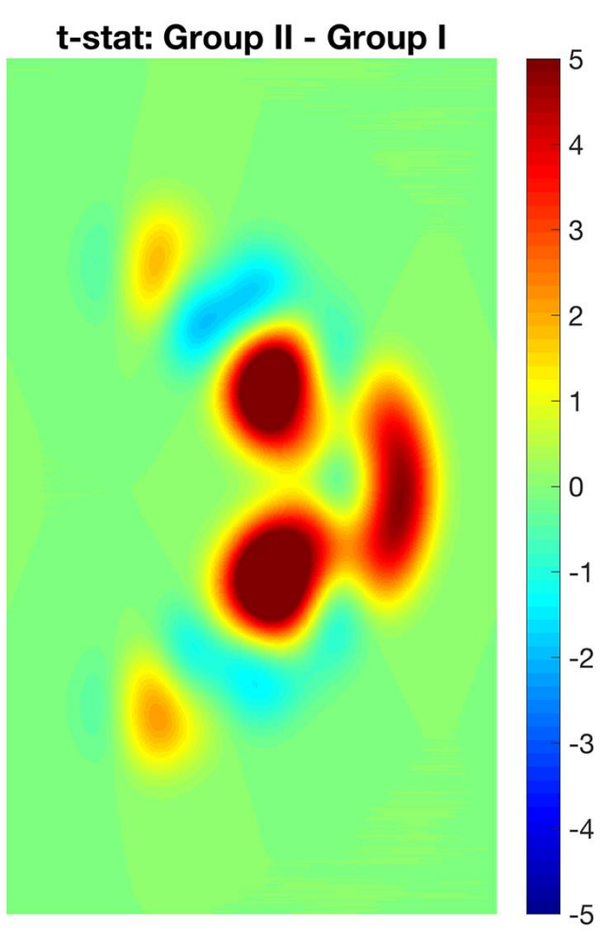
Smoothed image = probability density

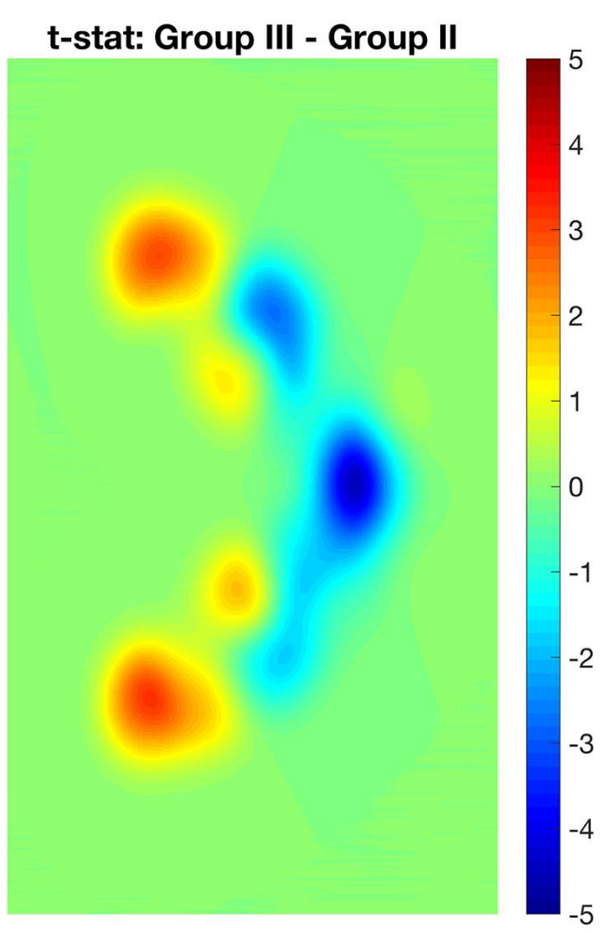


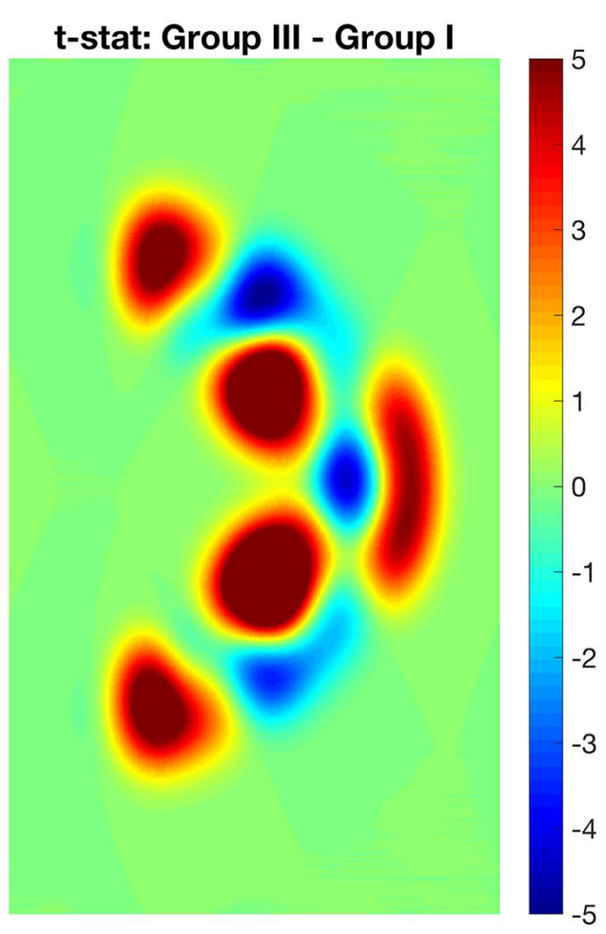
Group I:
-7 years

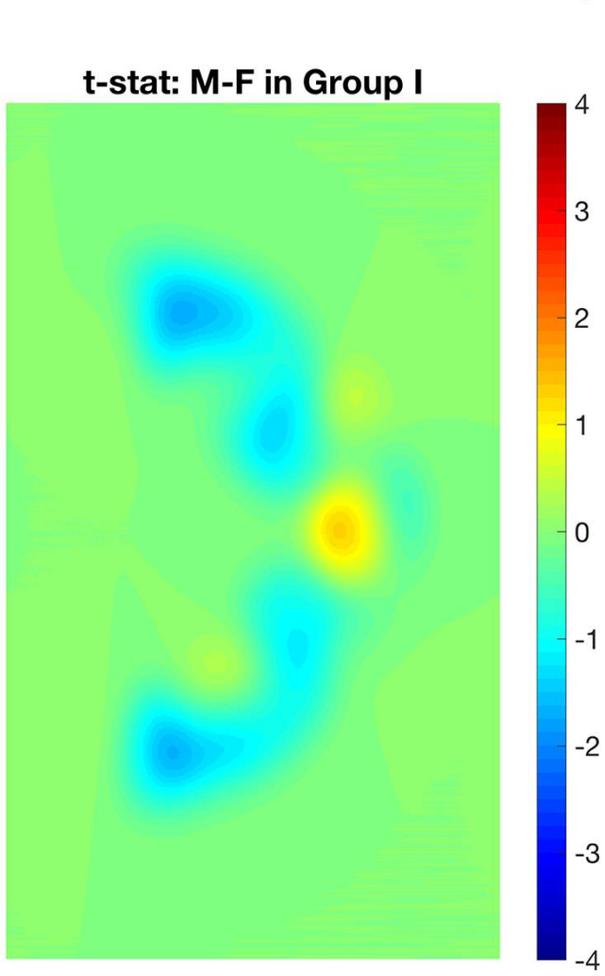
Group II:
7-13 years

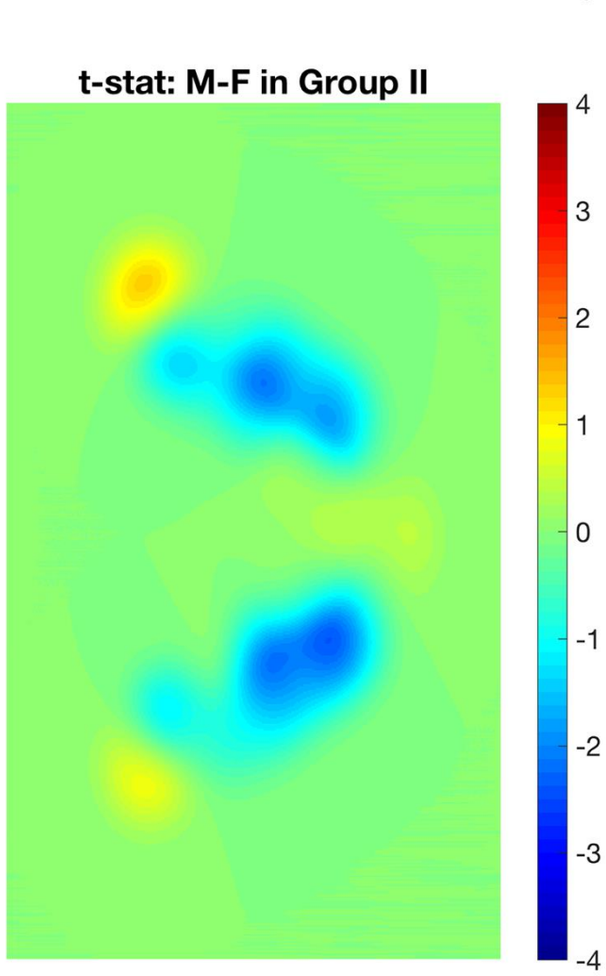
Group III:
13-20 years

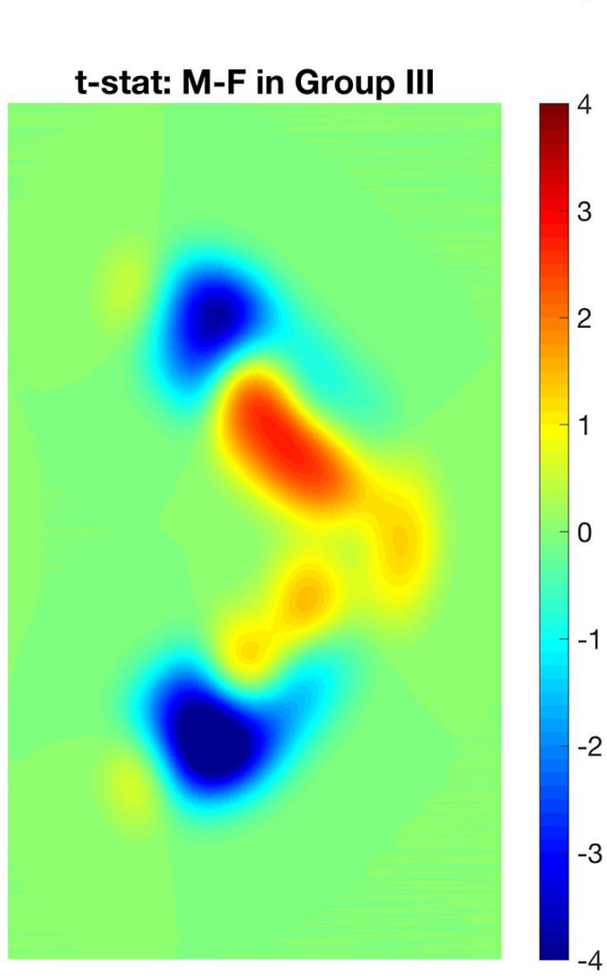








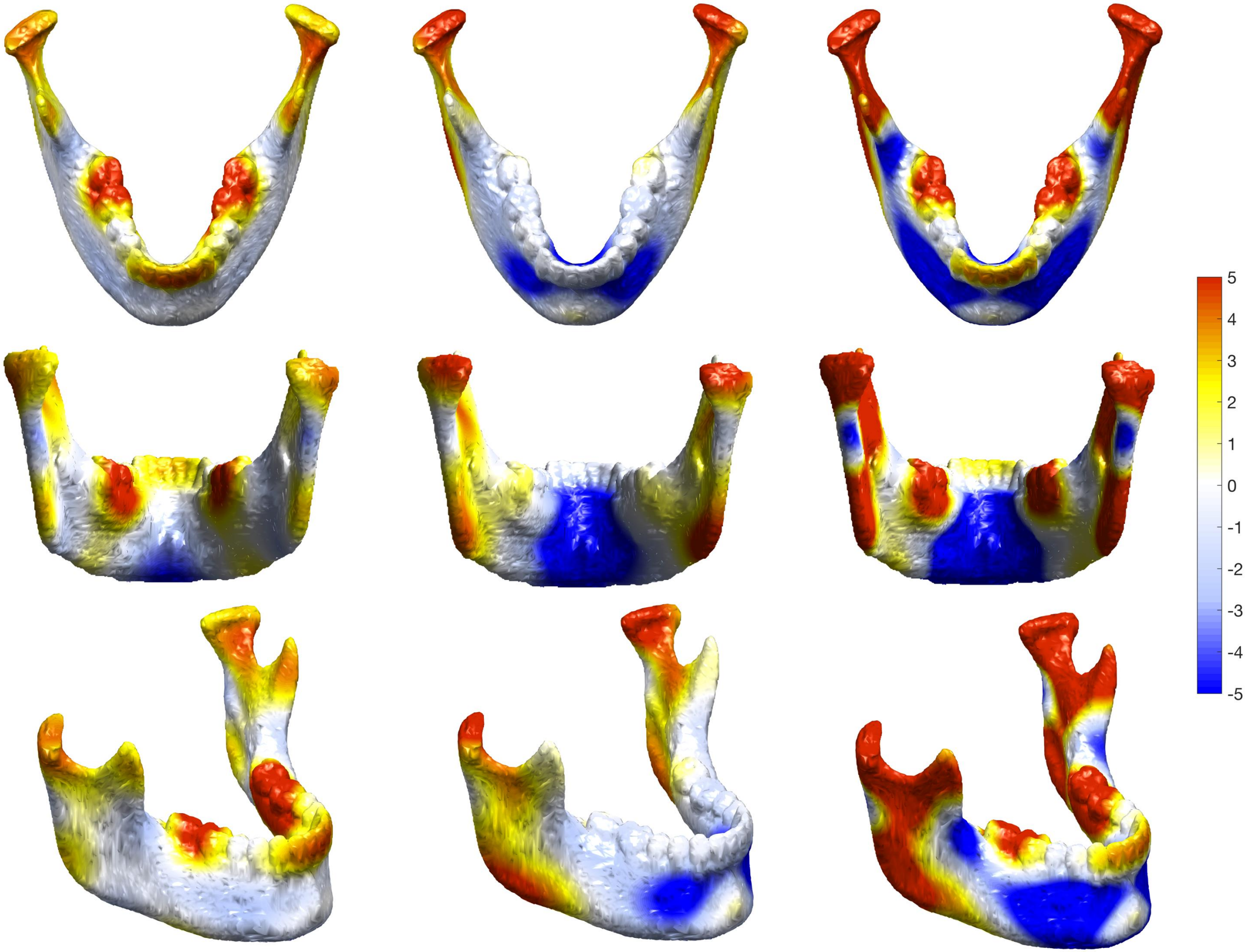




Group II-I

Group III-II

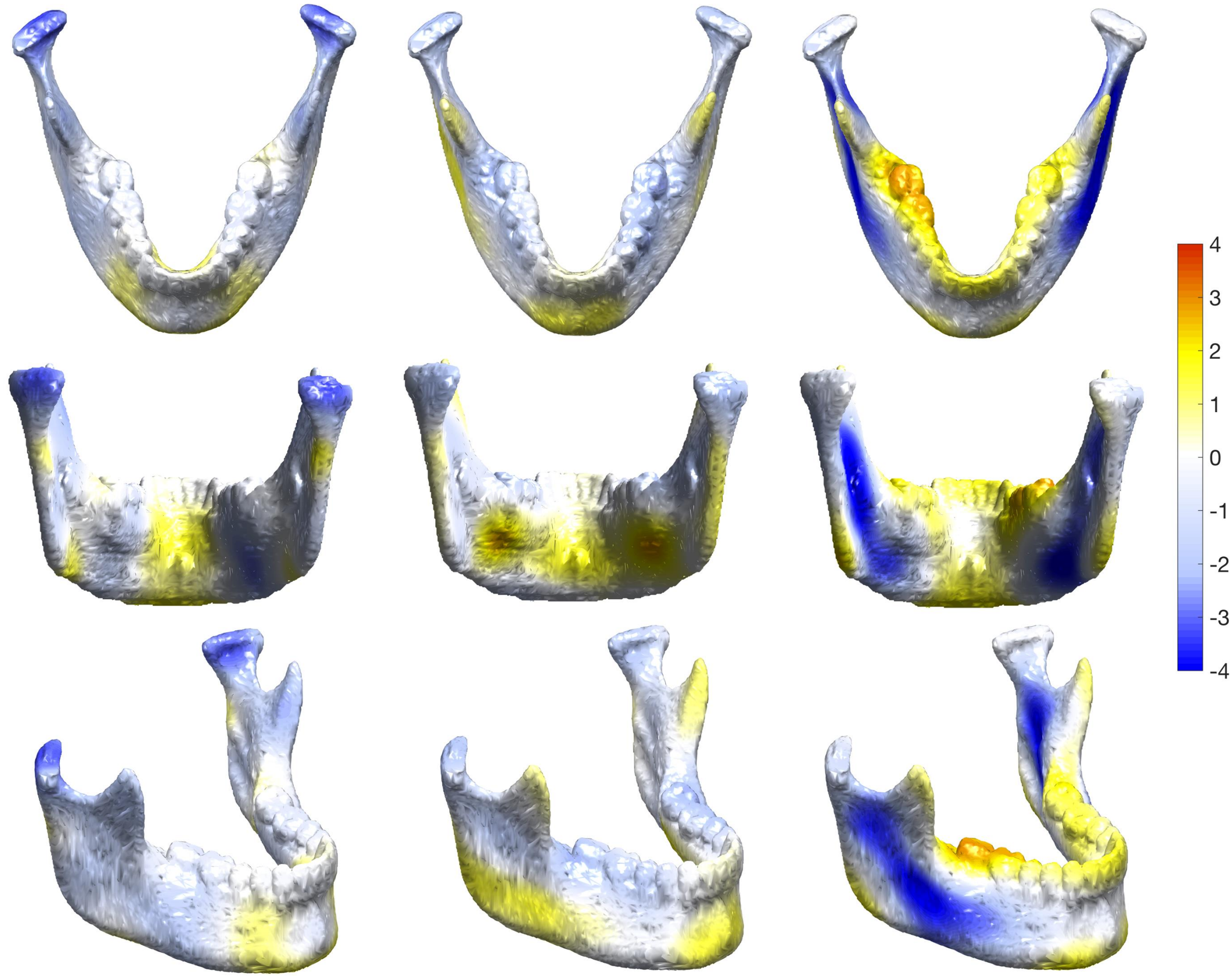
Group III-II



Group I (M-F)

Group II (M-F)

Group III (M-F)



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

Given $\mathcal{L}_1, \ldots, \mathcal{L}_m$

Is the online algorithm unique?

Let's solve a more complex problem!

Online iterative algorithms are not unique

Exercise: Design *different* ways of computing mean and variance using the online algorithm. Compare computational run time.

Transposition test

Online computation of permutation test

Permutation test

$$\mathbf{x} = (x_1, x_2, \cdots, x_m)$$
$$\mathbf{y} = (y_1, y_2, \cdots, y_n)$$
$$(\mathbf{x}, \mathbf{y}) = (x_1, \cdots, x_m, y_1, \cdots, y_n)$$
$$\pi(\mathbf{x}, \mathbf{y}) \in \mathbb{S}_{m+n}$$

Permutation group of order *m*+*n*

stat on each permutation

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

observation

Computational bottleneck

Why the permutation test is so slow

Most of computation is on **recomputi**
the statistic function. The actual
permutation itself is *not* computati
Bottleneck.

How to speed up the permutation te

Random transposition on the permutation group

$$\mathbf{x} = (x_1, x_2, \cdots, x_{i-1}, x_i, x_{i+1}, \cdots, x_m)$$

transpose *i*-th and *j*-th data

$$\mathbf{y} = (y_1, y_2, \cdots, y_{j-1}, y_j, y_{j+1}, \cdots, y_n)$$

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

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

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(3*m*+2)

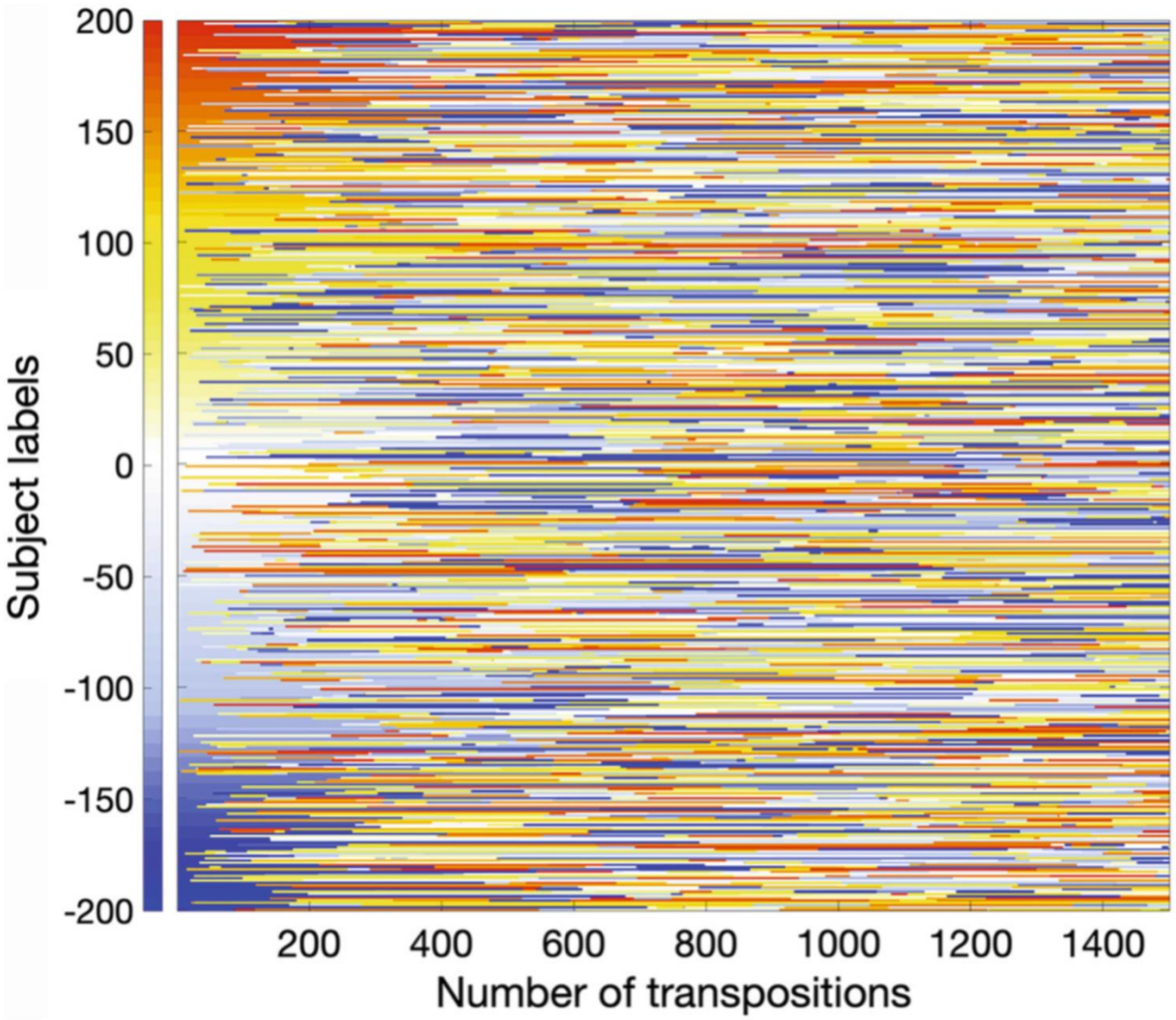
$$\nu(\pi_{ij}(\mathbf{x})) = \nu(\mathbf{x}) - x_i + y_j \quad \text{O(2)}$$

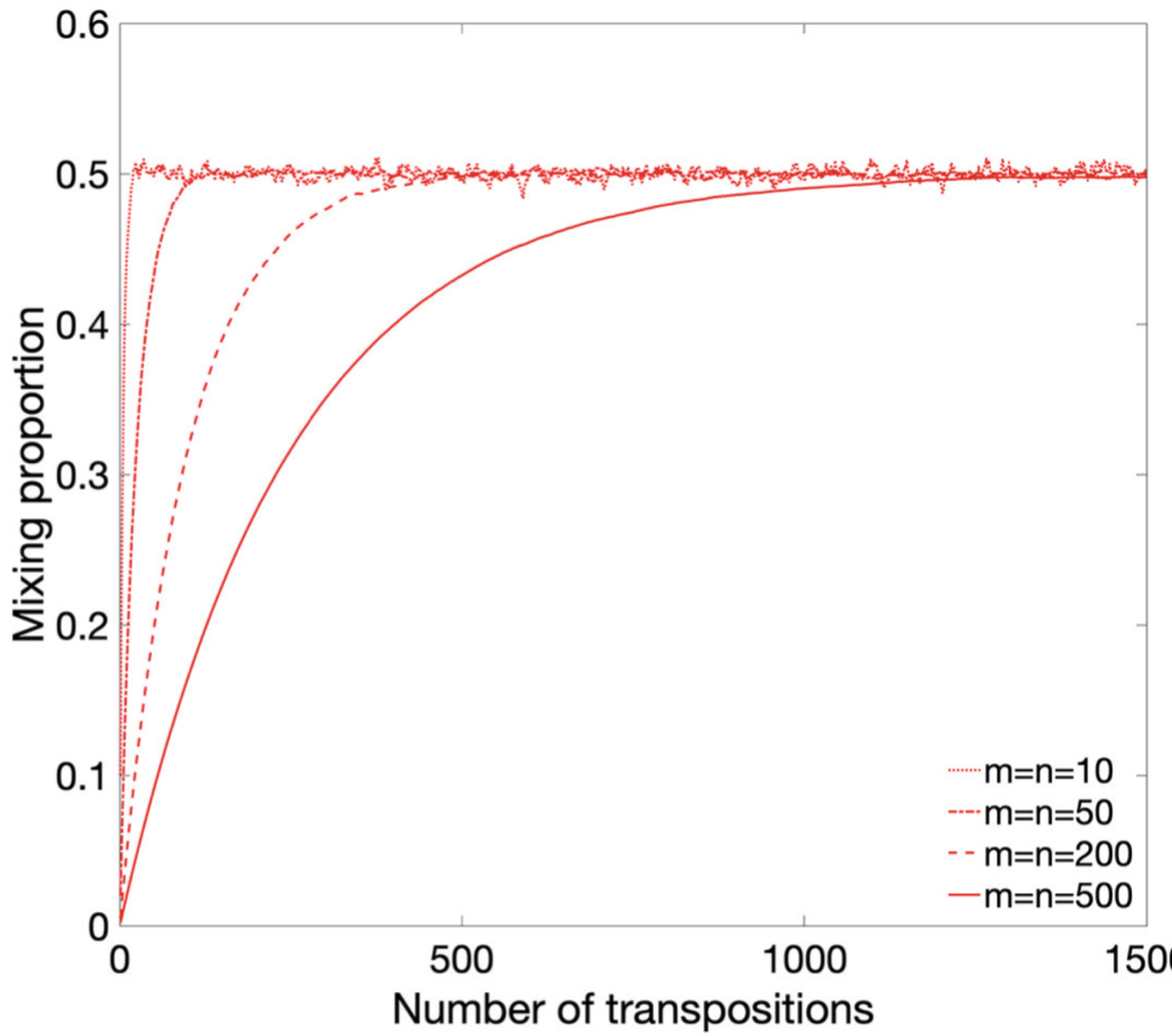
$$\omega(\pi_{ij}(\mathbf{x})) = \omega(\mathbf{x}) - x_i^2 + y_j^2 + \frac{\nu(\mathbf{x})^2 - \nu(\pi_{ij}(\mathbf{x}))^2}{m} \quad \text{O(9)}$$

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

T-stat computation per transpositions O(35)

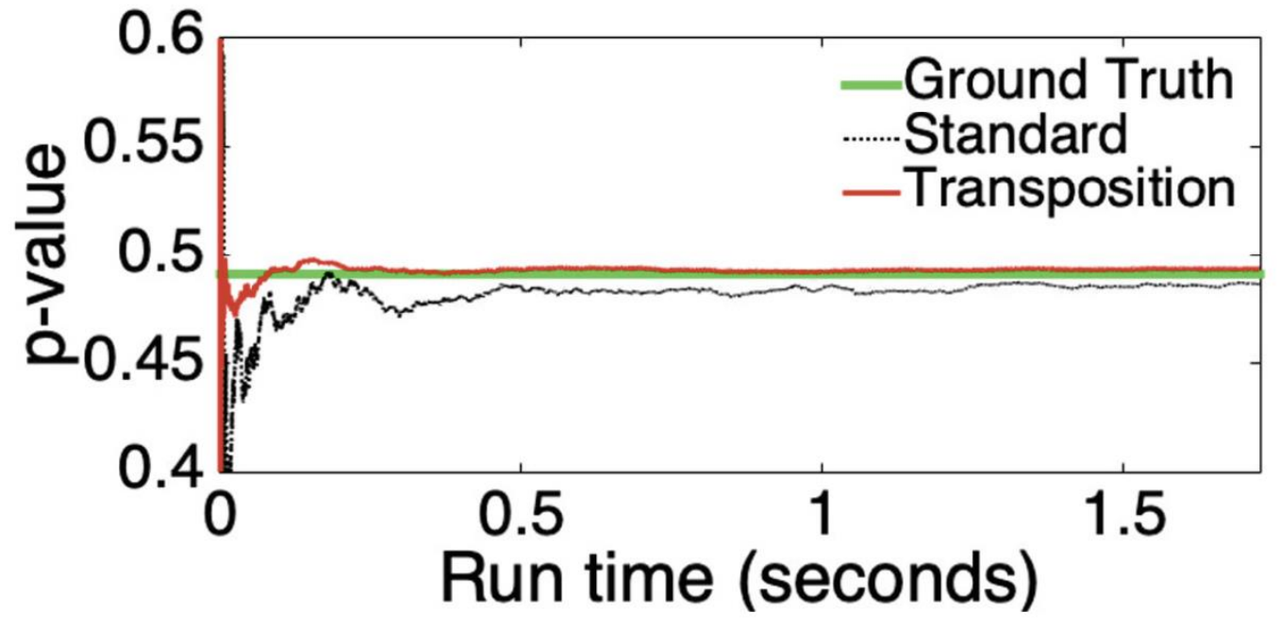
Simulation: Mixing proportion over transpositions

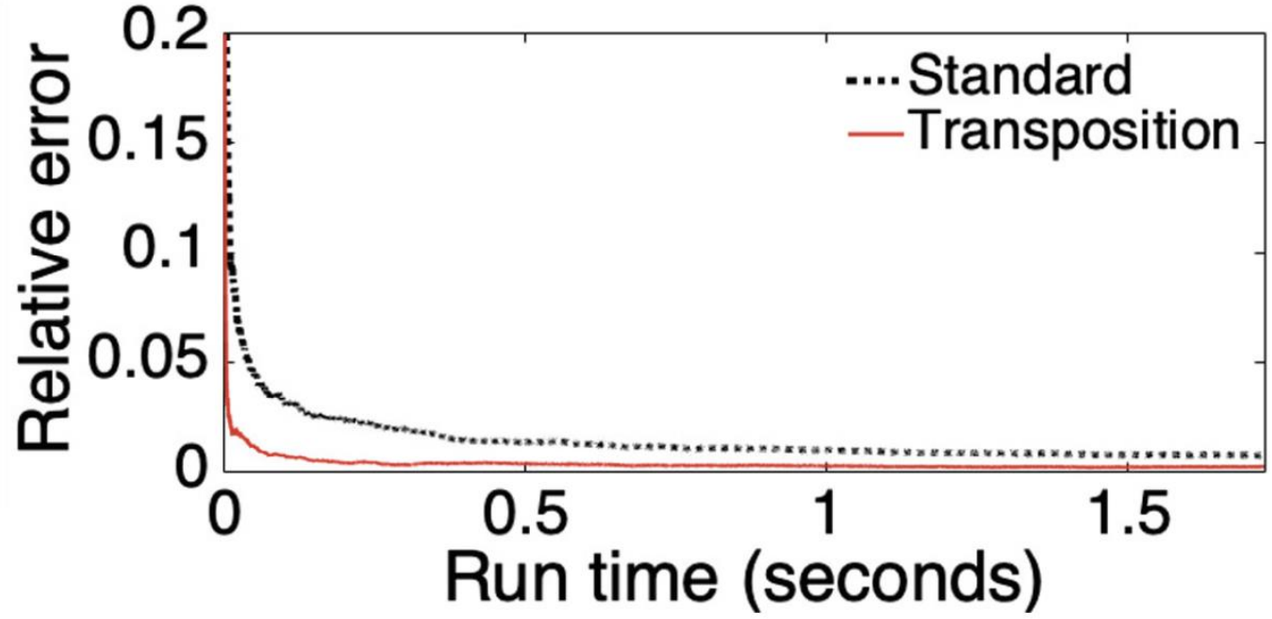




m=*n*=10

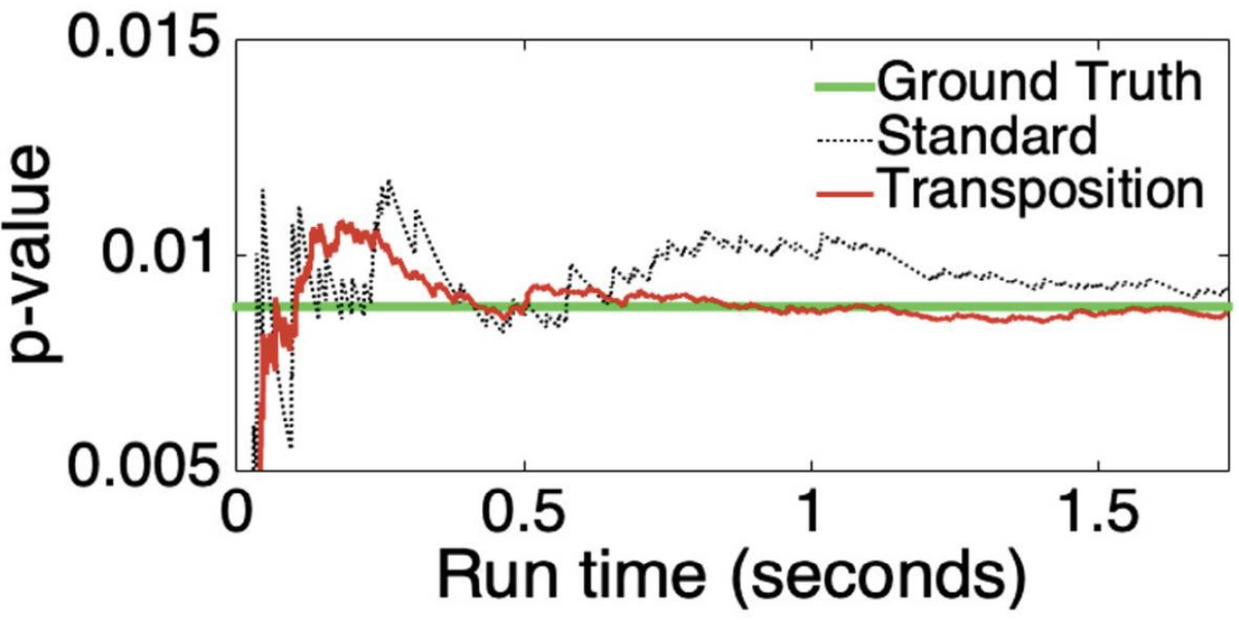
$$x_1, \cdots, x_m \sim 0.1 + Unif(0,1)$$
$$y_1, \cdots, y_n \sim Unif(0,1)$$

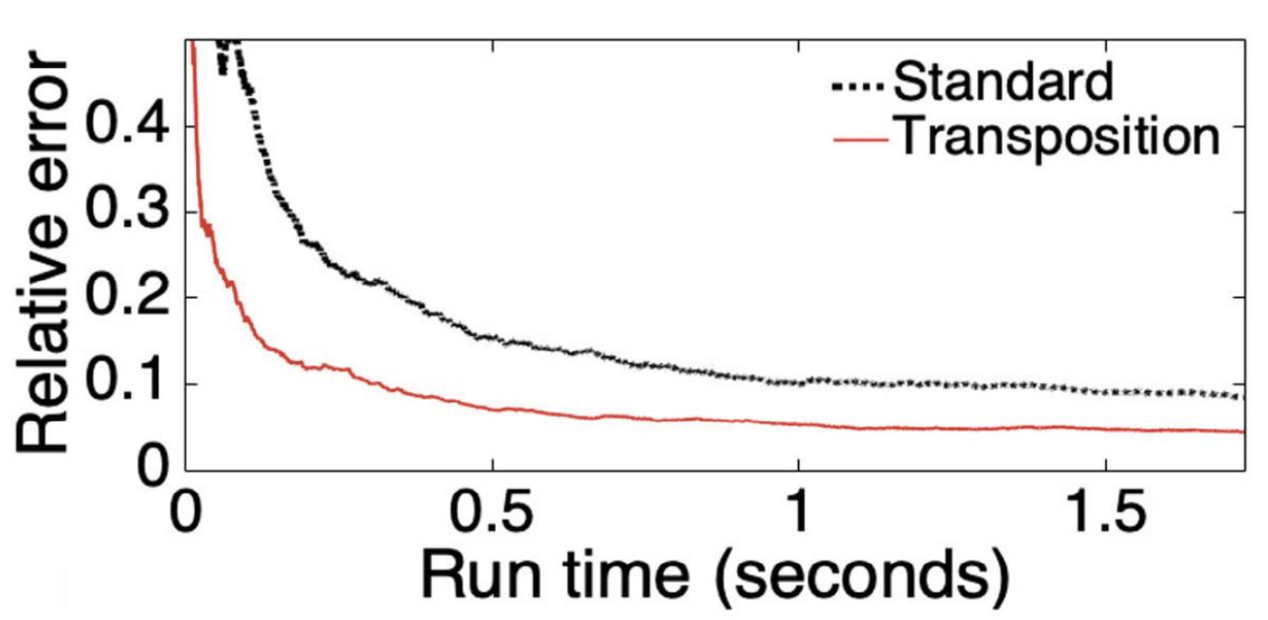




m=*n*=100

$$x_1, \cdots, x_m \sim 0.1 + Unif(0,1)$$
$$y_1, \cdots, y_n \sim Unif(0,1)$$





Matlab code

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

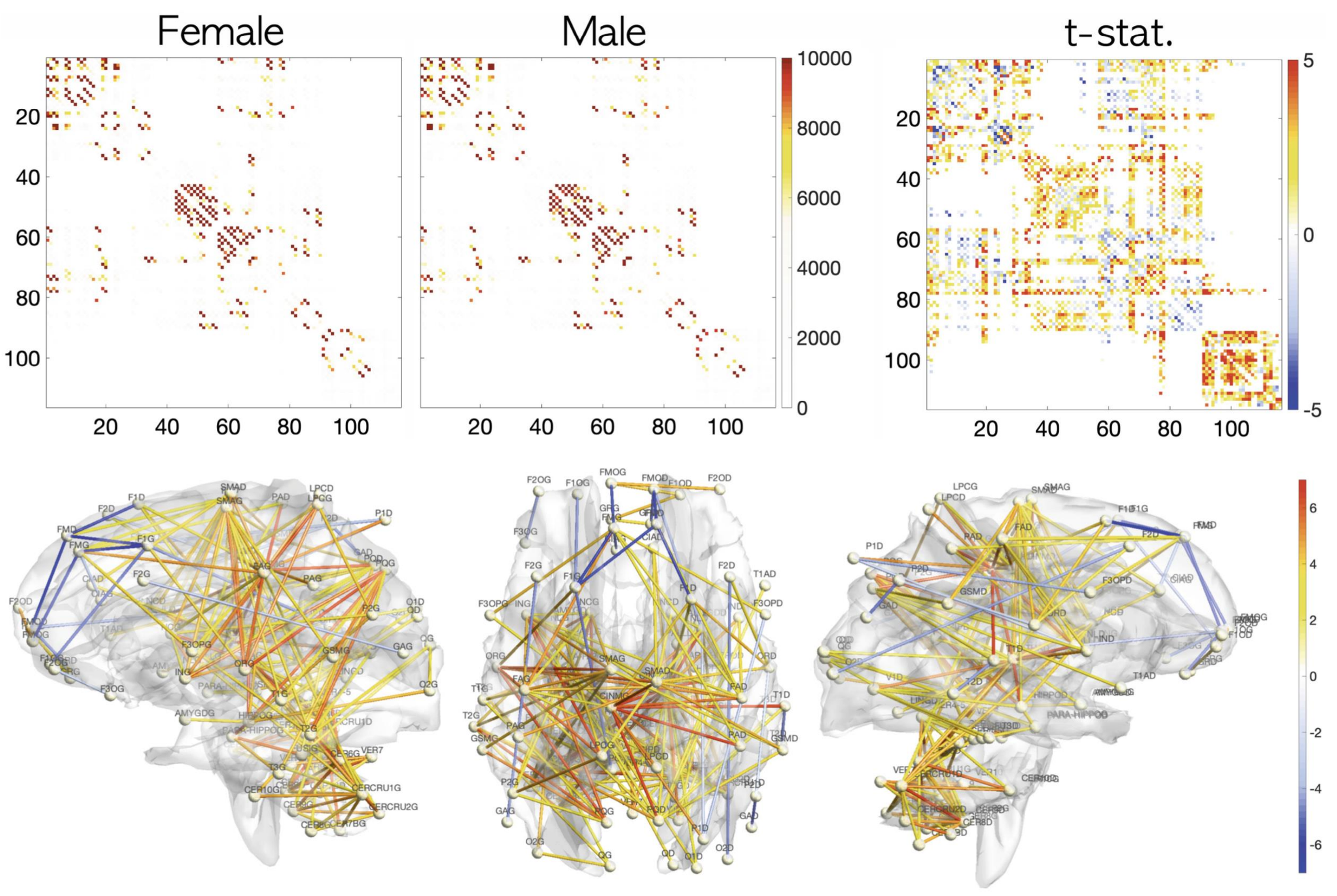
https://github.com/laplcebeltrami/transpositions

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
```



t-stat (202 females – 154 males)

Accelerating online permutation test

Modify transposition test

```
stat_t=[];
for i=1:10000
    [stat, time] = test_transpose (x, y, per_t / 10000);
    stat_t=[stat_t; stat];
    if ~mod(i, 1000)
       [stat_t, time_t] = test_permute (x, y, 1);
    end
    time_t=time_t + time;
end
```

Add perturbation(standard permutation) from time to time.

Transposition test on Wasserstein distance

Songdechakraiwut and Chung 2023 Annals of Applied St

2-Wasserstein distance between point cloud data

Random variables: $X \sim f_1 \quad Y \sim f_2$

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

Point cloud data

$$P_1 = \{x_1, \cdots, x_q\} \subset \mathbb{R}^2 \qquad P_2 = \{y_1, \cdots, y_q\} \in \mathbb{R}^2$$

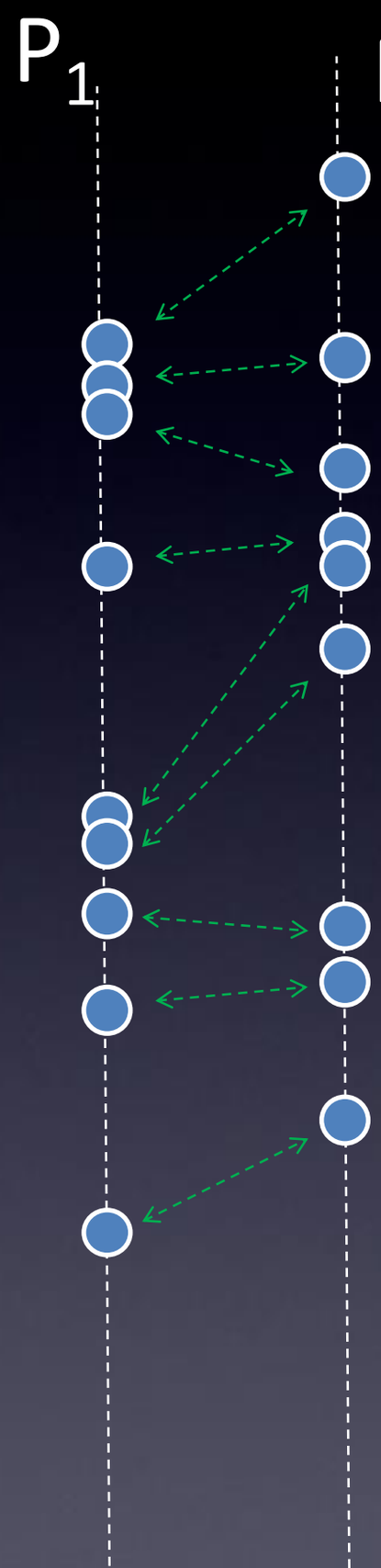
Empirical distributions

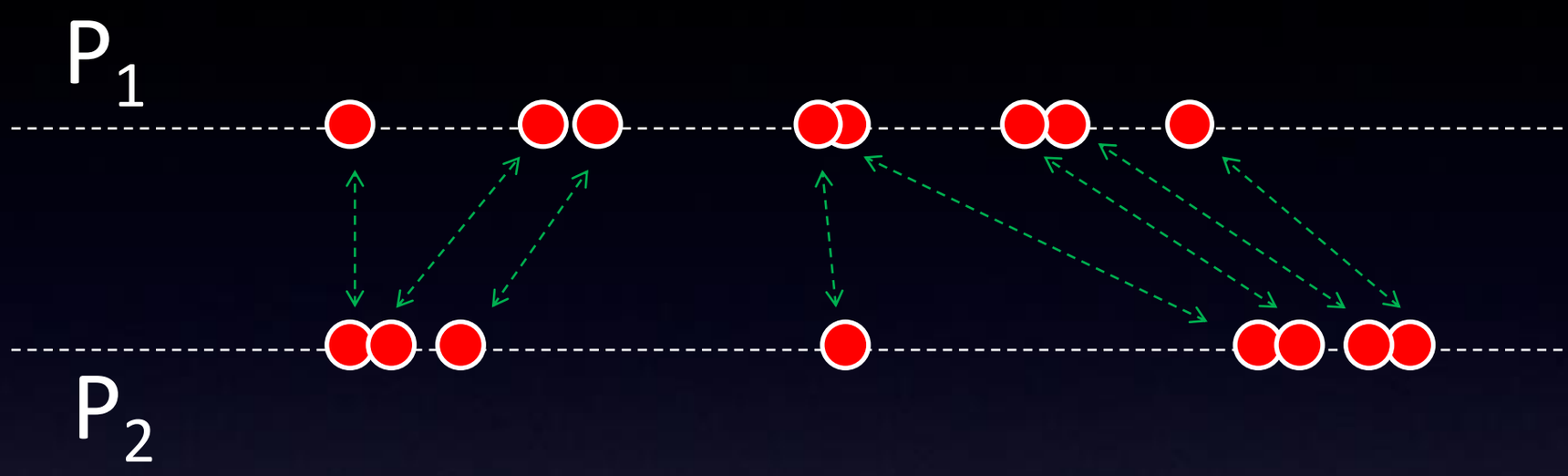
$$f_1(x) = \frac{1}{q}\sum_{i=1}^{q} \delta(x - x_i) \qquad f_2(y) = \frac{1}{q}\sum_{i=1}^{q} \delta(y - y_i)$$

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

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

Wasserstein distance for 1D data





1D linear assignment problem
has known analytic solution

$\mathcal{O}(q \log q)$

Match i-th smallest value to the i-th smallest value

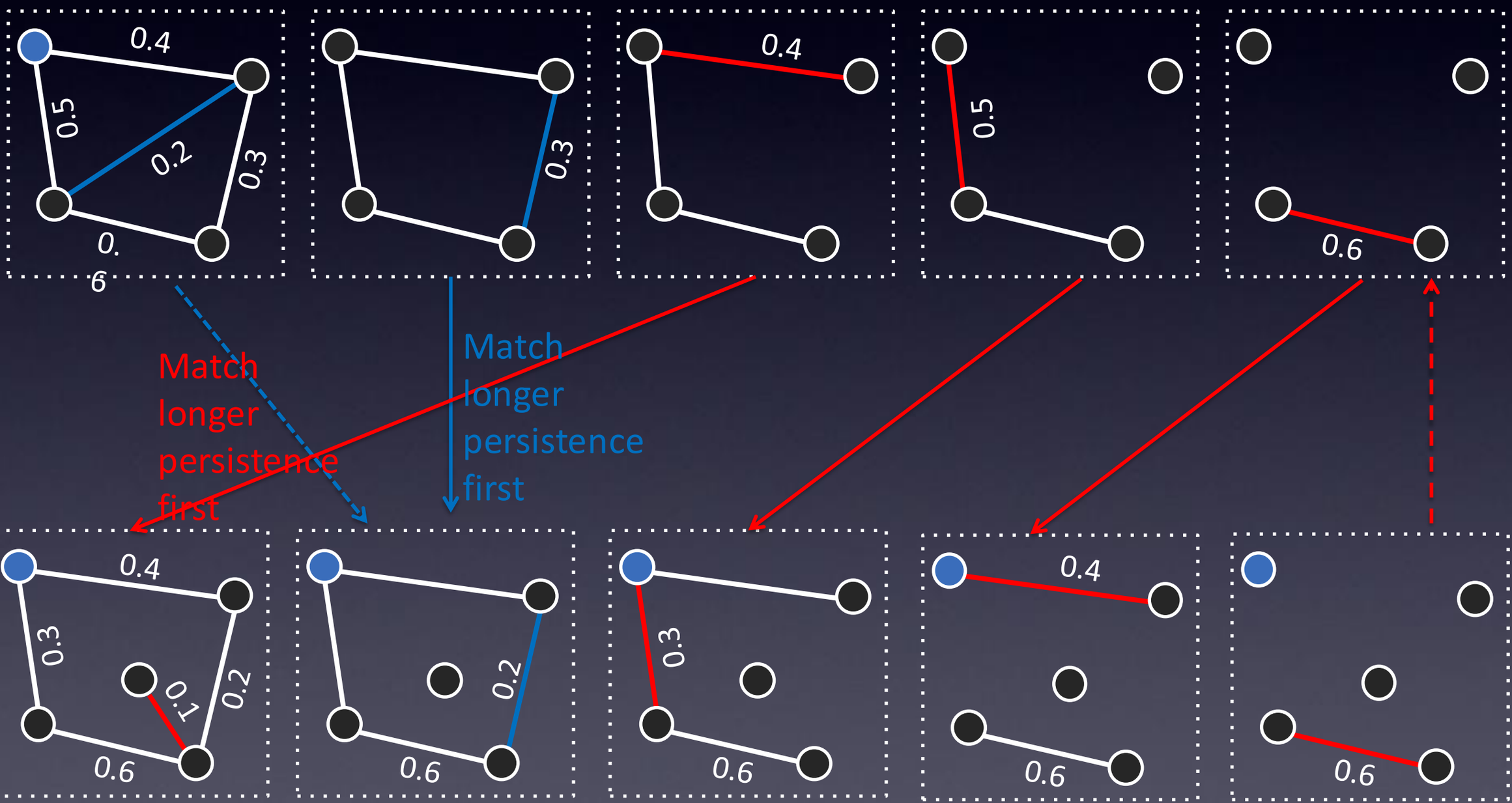
Songdechakraiwut et al. 2021, MICCAI 116-

Graph matching by minimizing Wasserstein distance on edge weight

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

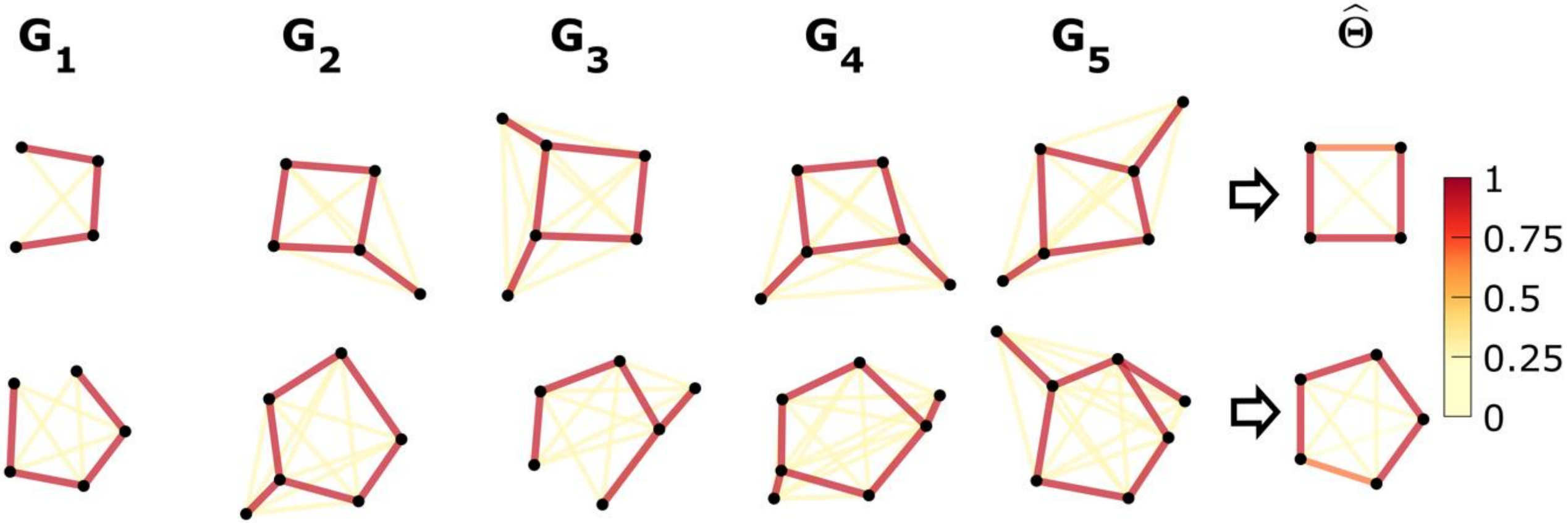
H_1 Edges destroy cycles

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: $G_1 + G_2 + \cdots + G_n$

Songdechakraiwut et al. 2023 Annals of Applied Stat arXiv: 2012.00675

ANOVA-like stat for Wasserstein distances

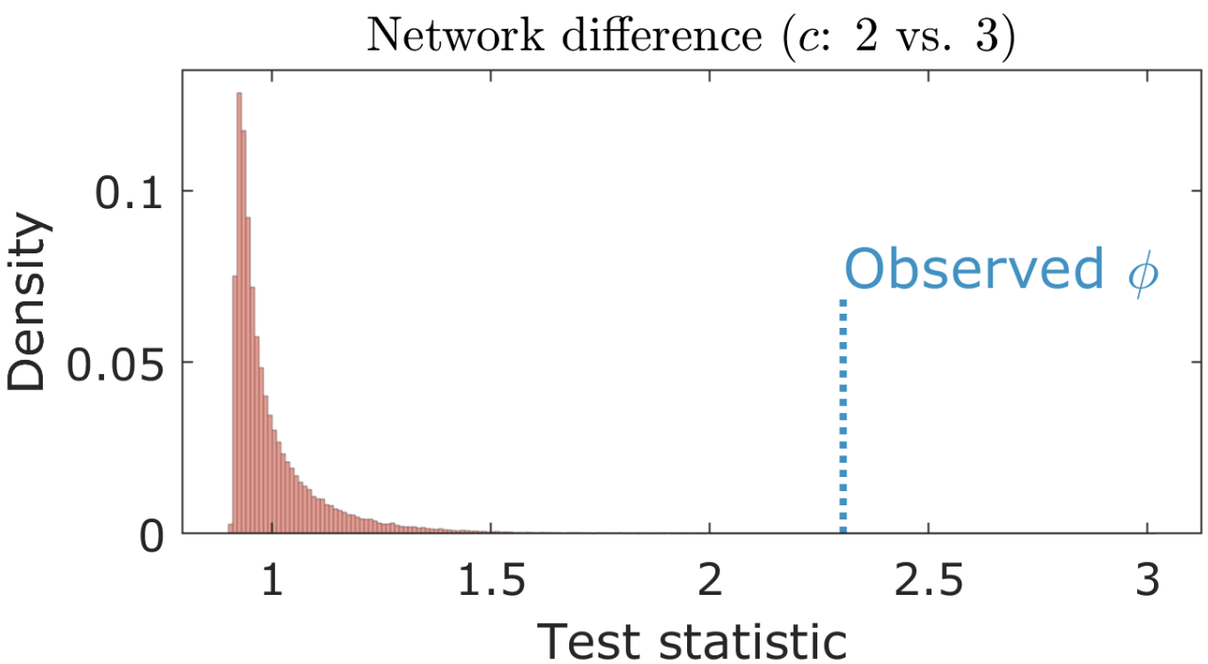
Between-group distance

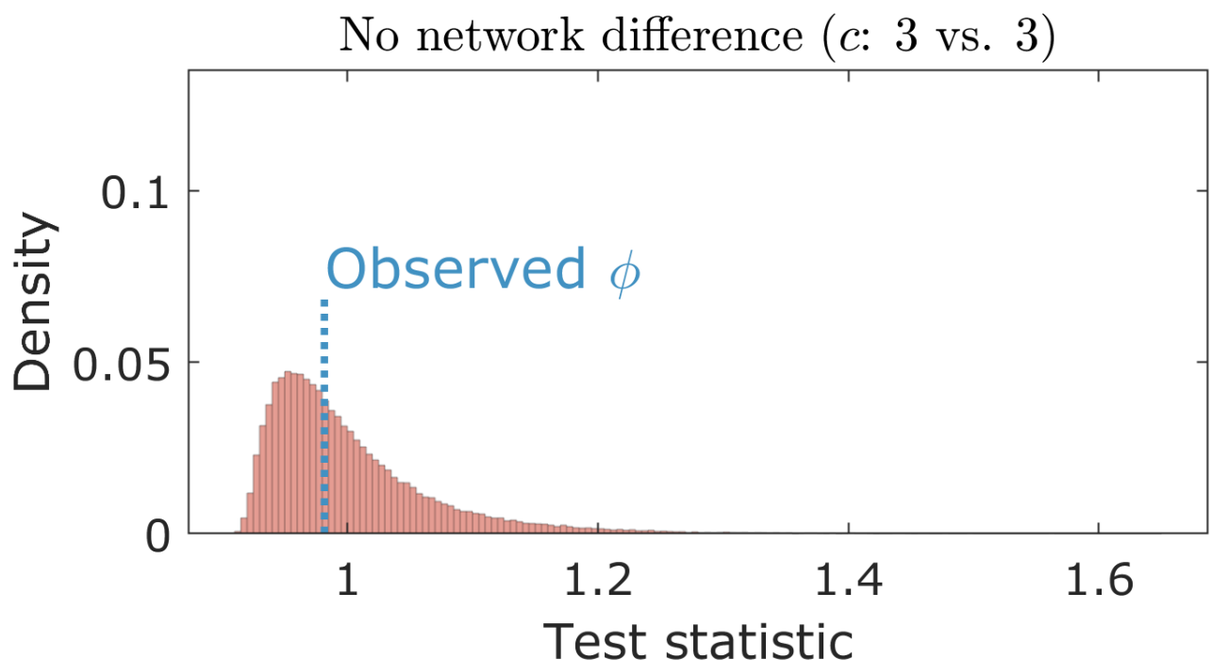
$$l_B \propto \sum_{i \in C_1, j \in C_2} \mathcal{L}(G_i, G_j)$$

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

Within-group distance

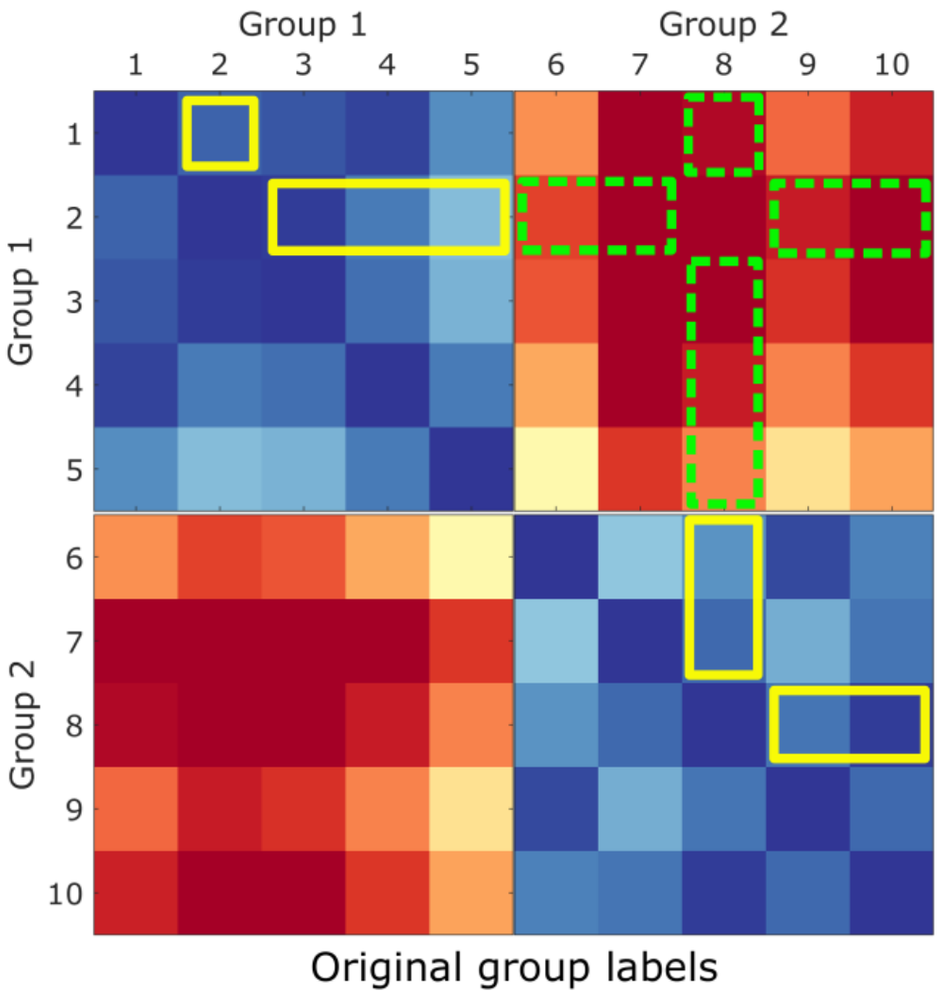
$$l_W \propto \sum_{k} \sum_{i,j \in C_k} \mathcal{L}(G_i, G_j)$$

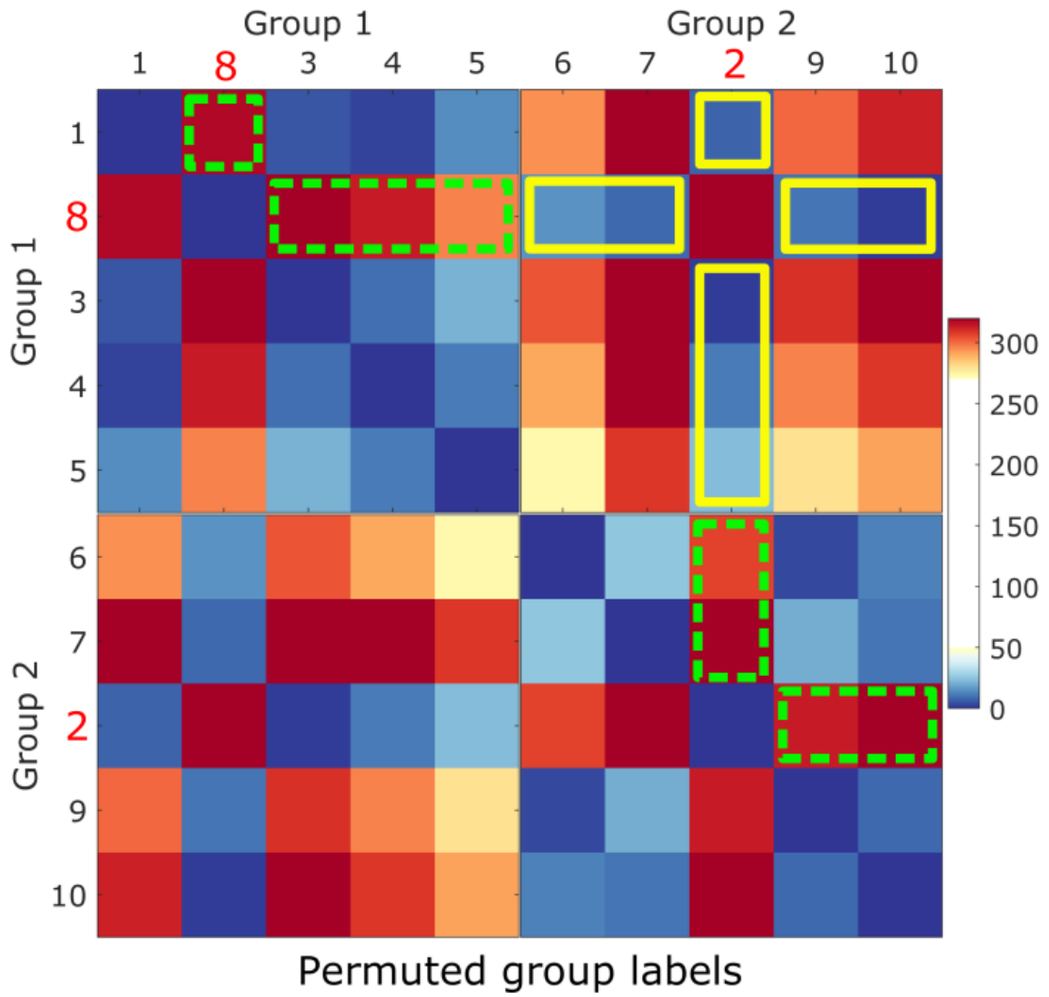




Transposition test on Wasserstein distance

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

$$\overline{\mathcal{L}}_W \rightarrow \overline{\mathcal{L}}_W + \Delta(tranposition)$$

$$\overline{\mathcal{L}}_B \rightarrow \overline{\mathcal{L}}_B + \Delta(tranposition)$$

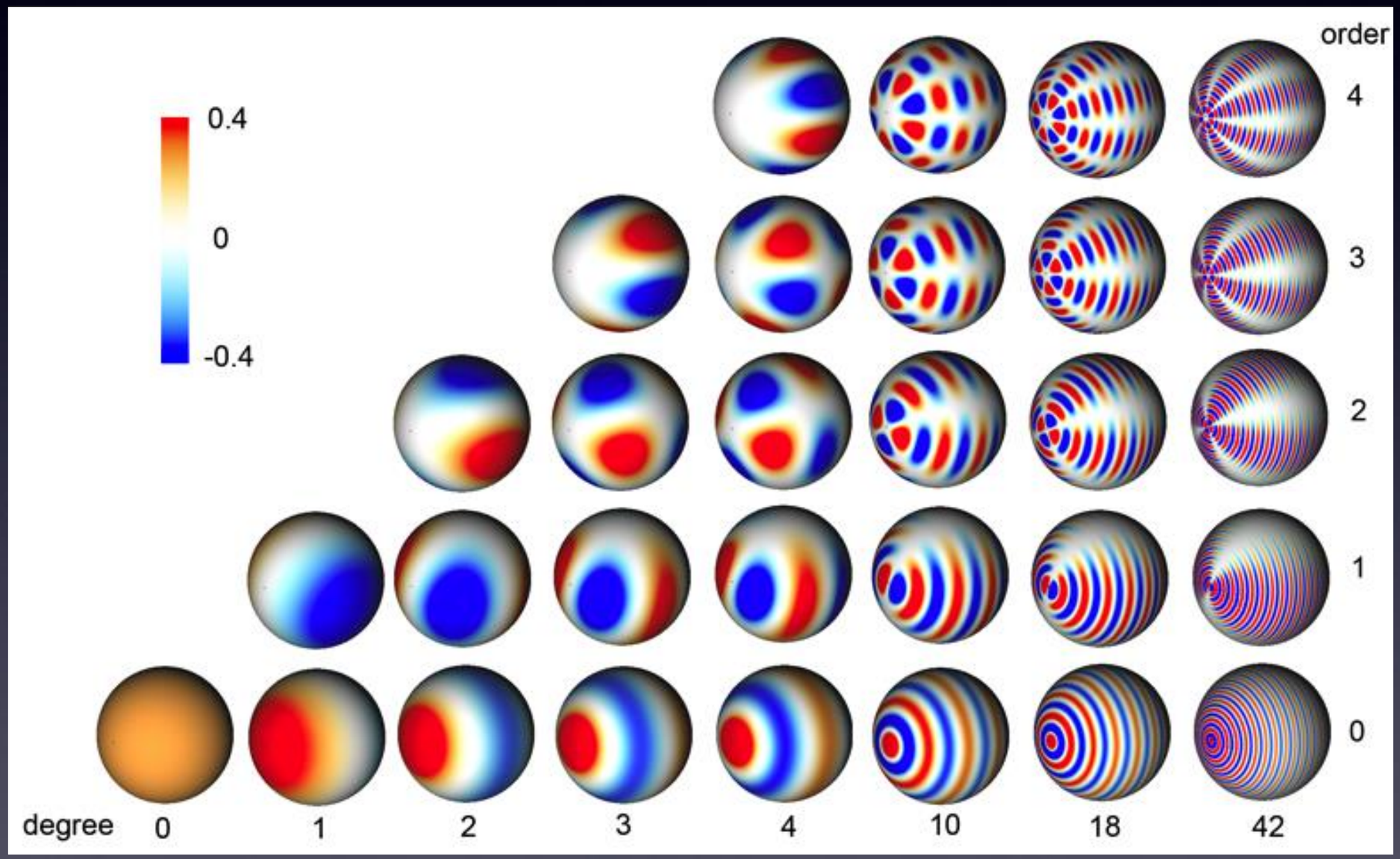
Matlab code:
https://www.stat.wisc.edu/~mchung/dynami

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 series expansion

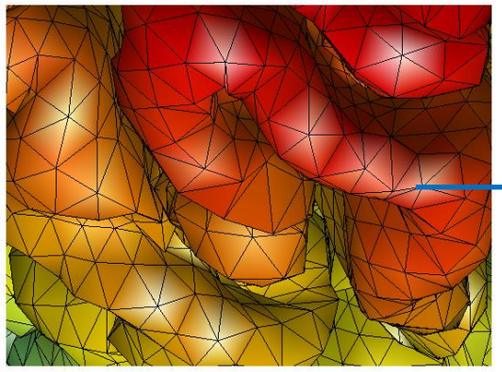
Spherical harmonic of degree *l* and order *m*

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



Coarse detail ⟷ Fine detail

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



i-th mesh vertex

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

- Matrix form:

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

300000 vertices x 10000 basis

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

Chung et al. 2007 IEEE TMI

Eigenfunction expansion

i-th vertex

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

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

$$\beta = (\beta_0, \cdots, \beta_k)'$$

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

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

Parameter Estimatio in more general case

i-th vertex

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

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

$$\beta = (\beta_0, \cdots, \beta_k)'$$

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

$$\mathbf{f} = \mathbf{Y}\beta$$

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

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

Iterative residual fitting (IRF) algorithm

Step 1. measurements $f(p_1), \cdots, f(p_n)$

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

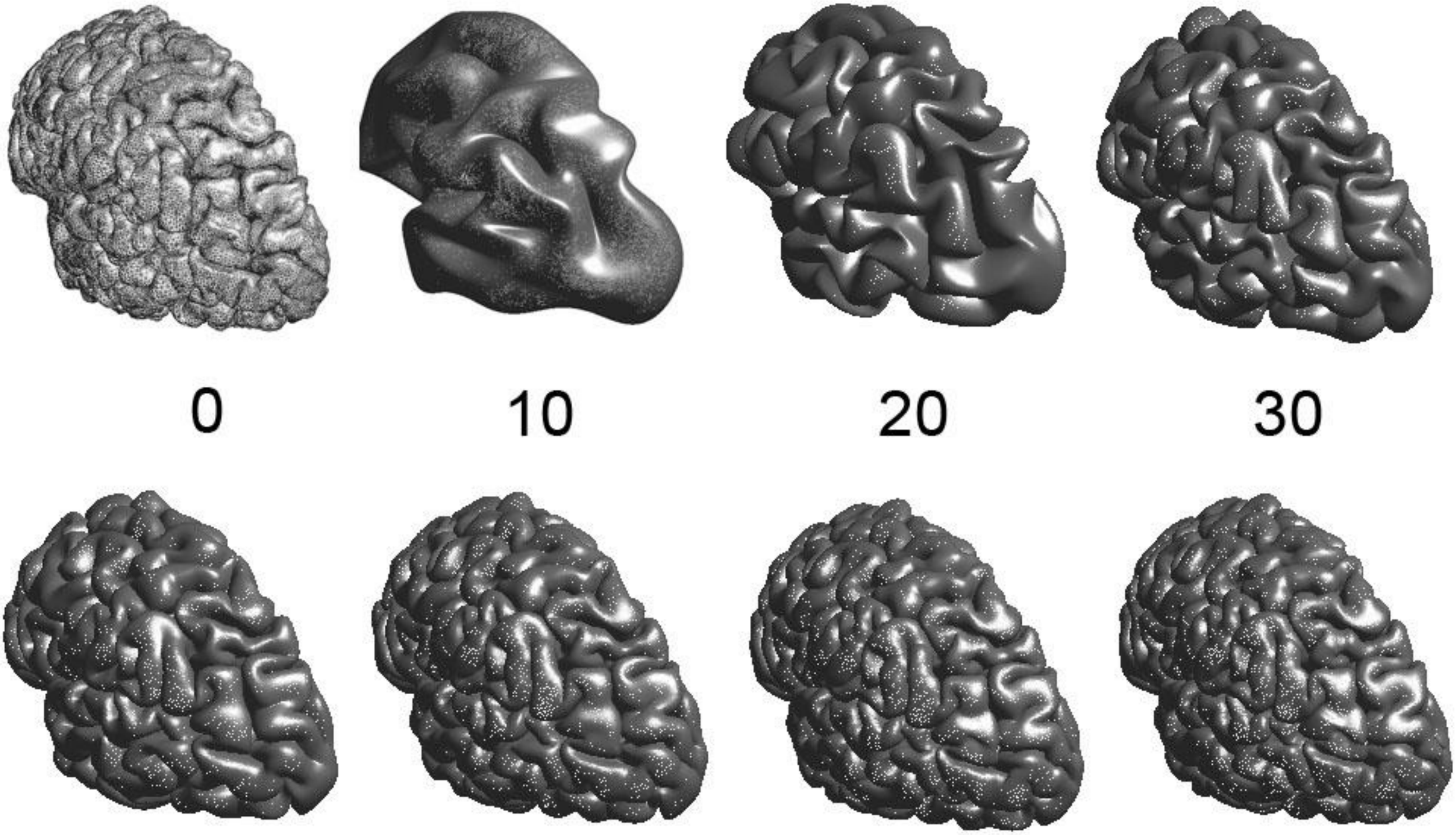
Step 3.5. $f \leftarrow f - \hat{f}$ Once low frequency parts are estimated, we throw them away

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

Iterate

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

We recycle the previous estimates and improves the fit



Learning outcome

1. Understand the philosophy of online algorithm development.

2. Can code few online algorithms.

3. Understand when not to apply.

Self assessment question

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

$$x_1, x_2, \cdots, x_m \qquad\qquad y_1, y_2, \cdots, y_n$$

$$\downarrow \qquad\qquad \downarrow$$

$$\rho_1 \quad \xleftrightarrow[\text{test}]{\text{Permutation}} \quad \rho_2$$

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*