POSTERIOR AGREEMENT FOR CONNECTIVITY IN ANATOMICAL CONTEXT

Author: Nico Stephan Gorbach , Institute of Machine Learning, ETHZ, email: nico.gorbach@inf.ethz.ch

Implementation of " **Pipeline Validation for Connectivity-based Cortex Parcellation** " by Nico S. Gorbach, Marc Tittgemeyer and Joachim M. Buhmann

Simulation for validation by posterior agreeement for clustering in an anatomical context

Contents


```
clear all; close all; clc
```

Introduction

This instructional code validates clustering in an anatomical context by simulating connectivity scores from a seed region in the postcentral gyrus. Given a template connectivity structure we can describe the fibers underlying that connectivity structure by Bézier curves. Bézier curves are defined by a set of control points $\mathbf{P}$:

$$\mathbf{B} = \mathbf{A}(\mathbf{t})\,\mathbf{P} + \epsilon, \qquad \epsilon \sim \mathcal{N}(\mathbf{0}, \gamma\mathbf{I}),$$

where $\mathbf{t}$ is a vector of linearly spaced scalar values ranging from 0 to 1 and γ determines the spread of the fiber samples. For a cubic Bézier curve, matrix $\mathbf{A}$ is given by:

$$\mathbf{A}(\mathbf{t}) = (\mathbf{1}-\mathbf{t})^3, 3(\mathbf{1}-\mathbf{t})^2\mathbf{t}, 3(\mathbf{1}-\mathbf{t})\mathbf{t}^2, \mathbf{t}^3).$$

Since Bézier curves are linear in the control points $\mathbf{P}$, we can determine the control points by multiplying the left hand-side of equation \ref{eqn:bezier_curve} by the pseudo-inverse $\mathbf{A}$ (i.e. $\mathbf{P} = \mathbf{A}^{+}(\mathbf{B}-\epsilon)$). Notice that the first and last control points $\mathbf{p}_1, \mathbf{p}_4 \in \mathbf{P}$ mark the start and end of the Bézier curve, respectively. We can therefore determine the connectivity scores by sampling from the marginal distribution of $\mathbf{p}_4$:

$$\mathbf{p}_4 \sim \mathcal{N}((\mathbf{A}^{+}\mathbf{B})_{(4,\cdot)}, (\gamma\mathbf{A}^{+T}\mathbf{A}^{+})_{(4,\cdot)}),$$

We compare the posterior agreement to alternative validation criteria such as BIC and AIC.

The performance of each validation criteria is summarized by the confusion matrices whose entries capture the frequency of estimated clusters (rows) for a range of actual number of clusters (columns).

Input

```
nsamples = 300;
edges = [1:1:800];
K = 2:12;
nClusters = 2:7;
```

```
% interpolate seed coord
seed_coords = importdata('seed_coords.mat'); seed_coords(:,2) = 500-seed_coords(:,2);
xyz = seed_coords';
[~,npts]=size(xyz);
xyzp=[];
for k=1:2
    xyzp(k,:)=fnval(csaps(1:npts,xyz(k,:)),[1:0.1:npts]);
end
tmp = randperm(size(xyzp,2)); tmp = tmp(1:200); tmp = sort(tmp);
seed_coords = xyzp(:,tmp)';
```

Learn spline control points

```
line = importdata('anatomy_lines.mat');
data = [];
for i = 1:length(line)
    line_obs{i} = line{i}; line_obs{i}(:,2) = 500-line_obs{i}(:,2);
    distribution{i} = learn_spline(line_obs{i});
    ctrl_pts(:,:,i) = distribution{i}.ctrl_pts.mean;
end

[ctrl_pts2,ctrl_pts] = rearrange_ctrl_pts(ctrl_pts);

seed_ctrl_pts = reshape(ctrl_pts(1,:,:),2,[])';
```

```
% Initialize confusion matrix
confusion_matrix.PA = zeros(max(nClusters),max(K));
confusion_matrix.BIC = zeros(max(nClusters),max(K));
confusion_matrix.AIC = zeros(max(nClusters),max(K));
confusion_matrix.bayes_evidence1 = zeros(max(nClusters),max(K));
confusion_matrix.bayes_evidence2 = zeros(max(nClusters),max(K));
```

```
% start timer
tic;
```

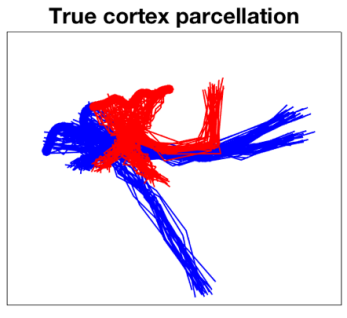
```
n_experiments =  1;
for n = 1:n_experiments
for  ktrue = nClusters
```

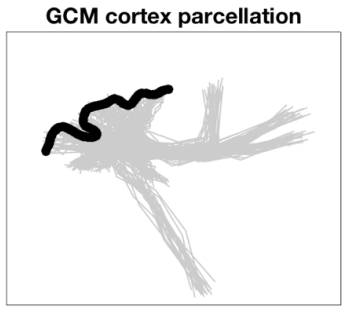
```
    clear fiber_assignment connectivity_matrix target_coord
```

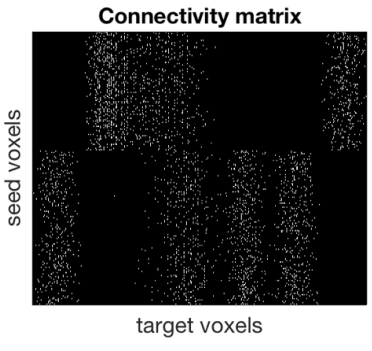
```
tmp0 = round(size(ctrl_pts2,3)*dirichletRnd(1,ones(1,ktrue)./0.1)); j=0;
tmp0(end) = tmp0(end) + (size(ctrl_pts2,3) - sum(tmp0));
for i = 1:ktrue
    for m = 1:tmp0(i)
        j=j+1;
        fiber_assignment{i}(m) = j;
    end
end

tmp = round(size(seed_coords,1)*dirichletRnd(1,ones(1,8)./0.1)); j=0; idx_start = 1; idx_end = 0;
tmp(end) = tmp(end) + (size(seed_coords,1) - sum(tmp));
seed_labels_true = ones(1,size(seed_coords,1));
for i = 1:length(tmp)-1
    idx_start = idx_start+tmp(i);
    idx_end = idx_start+tmp(i+1)-1;
    seed_labels_true(idx_start:idx_end) = i+1;
end
```

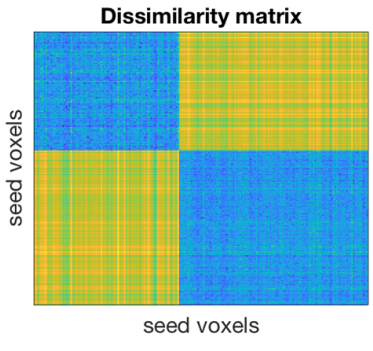
Connectivity matrices

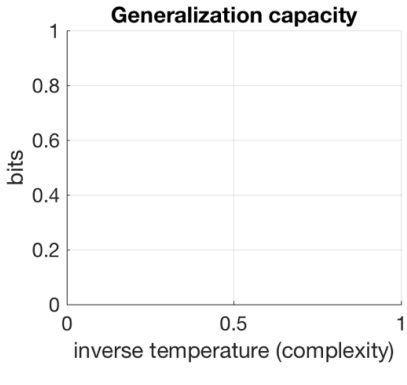
```
h = setup_plots;
colors = distinguishable_colors(length(fiber_assignment));

for m = 1:2
    for i = 1:size(seed_coords,1)
        clear spline_sample
        for j = 1:nsamples
            label(i) = find(cellfun(@(x) ismember(seed_labels_true(i),x),fiber_assignment));
            prob = ones(1,length(fiber_assignment{label(i)}))./length(fiber_assignment{label(i)});
            sample_idx = discreternd(prob,1);
            sample_idx = fiber_assignment{label(i)}(sample_idx);
            seed_coord_ctrl_pts = ctrl_pts2(:,:,sample_idx);
            seed_coord_ctrl_pts(1,:) = seed_coords(i,:);
            spline_sample{j} = sample_spline(5e1,1e4,seed_coord_ctrl_pts,line_obs{sample_idx});
            target_coord{m}(j,:,i) = round(spline_sample{j}(end,:));

            s(i) = sample_idx;
        end
        if m==1
        hold on; p{i} = plot(h{1},spline_sample{j}(:,1),spline_sample{j}(:,2),'LineWidth',1);
        end
        connectivity_matrix{m}(i,:) = reshape(histcn(target_coord{m}(:,:,i),1:600,1:600),1,[]);
    end
end

for i = 1:length(p)
    p{i}.Color = colors(label(i),:);
end
scatter(h{1},seed_coords(:,1),seed_coords(:,2),50,colors(label,:),'filled')

for i = 1:length(p)
    p2{i} = plot(h{2},p{i}.XData,p{i}.YData,'Color',[0.8,0.8,0.8],'LineWidth',1);
    hold on
end
scatter(h{2},seed_coords(:,1),seed_coords(:,2),50,[0,0,0],'filled')

% remove zero columns
siz = size(connectivity_matrix{1},2);
rm_idx = sum(connectivity_matrix{1},1) + sum(connectivity_matrix{2},1) == 0;
connectivity_matrix{1}(:,rm_idx) = [];
connectivity_matrix{2}(:,rm_idx) = [];

% normalize connectivity matrix
% data{1} = sparse(bsxfun(@rdivide,connectivity_matrix{1},sum(connectivity_matrix{1},2)));
% data{2} = sparse(bsxfun(@rdivide,connectivity_matrix{2},sum(connectivity_matrix{2},2)));
data{1} = sparse(connectivity_matrix{1});
data{2} = sparse(connectivity_matrix{2});

% display connectivity matrix
imagesc(h{3},flipdim(logical(connectivity_matrix{1}),1)); colormap(h{3},gray)
h{3}.FontSize = 15; h{3}.XTick = []; h{3}.YTick = []; h{3}.Box = 'on';
h{3}.Title.String = 'Connectivity matrix'; h{3}.XLabel.String = 'target voxels';
h{3}.YLabel.String = 'seed voxels';

% display dissimilarity matrix
%dsim = flipdim(pdist2(data{1},data{2}),1);
dsim = pdist2(data{1},data{2});
imagesc(h{4},flipdim(flipdim(dsim,1),2));
h{4}.FontSize = 15; h{4}.XTick = []; h{4}.YTick = []; h{4}.Box = 'on';
h{4}.Title.String = 'Dissimilarity matrix';
h{4}.XLabel.String = 'seed voxels'; h{4}.YLabel.String = 'seed voxels';
```

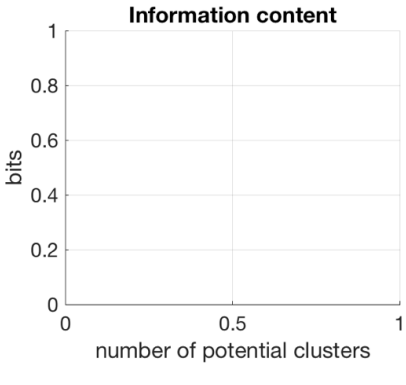

True cortex parcellation

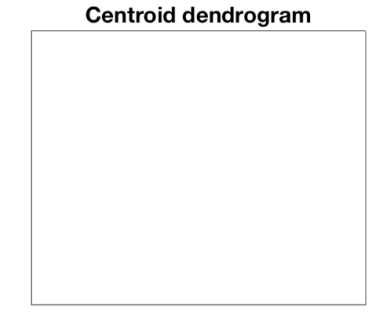

GCM cortex parcellation

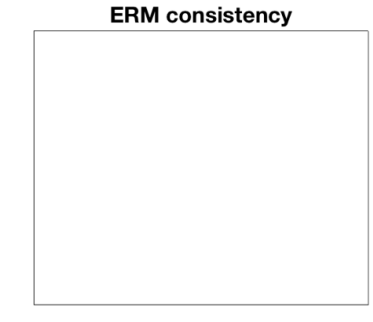

Connectivity matrix
seed voxels
target voxels

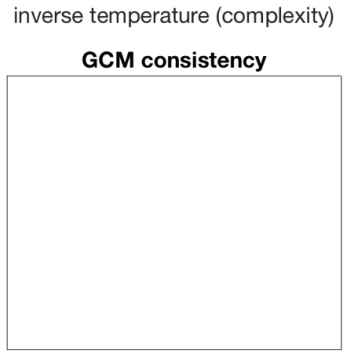

Dissimilarity matrix
seed voxels
seed voxels

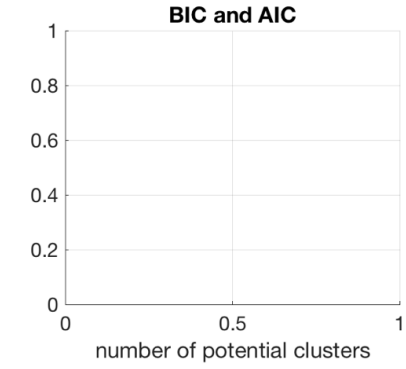

Generalization capacity
bits
inverse temperature (complexity)

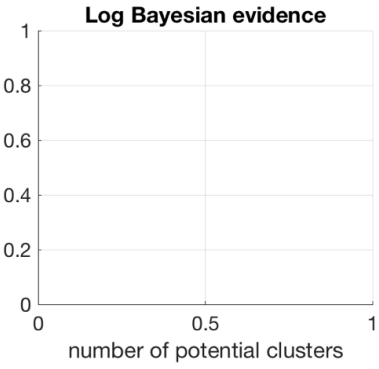

Information content
bits
number of potential clusters

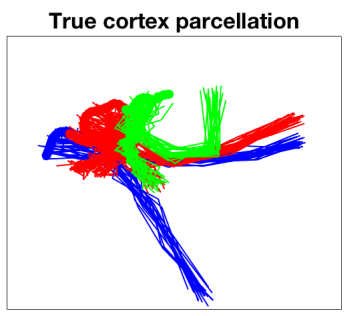

Centroid dendrogram

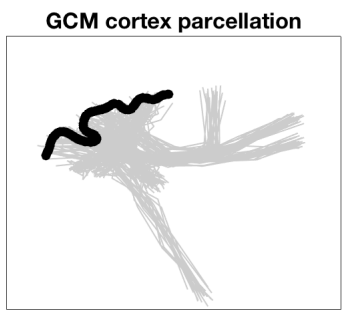

ERM consistency

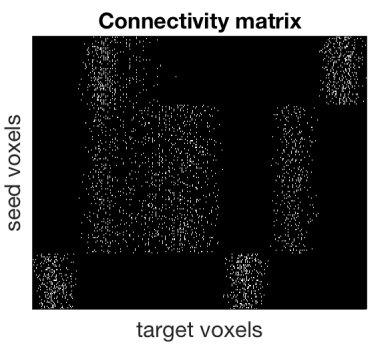

GCM consistency

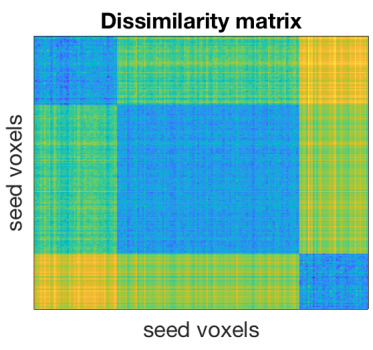

BIC and AIC
number of potential clusters

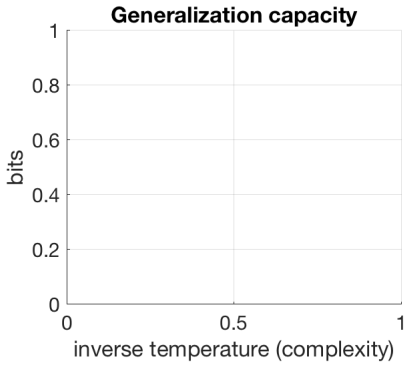

Log Bayesian evidence
number of potential clusters

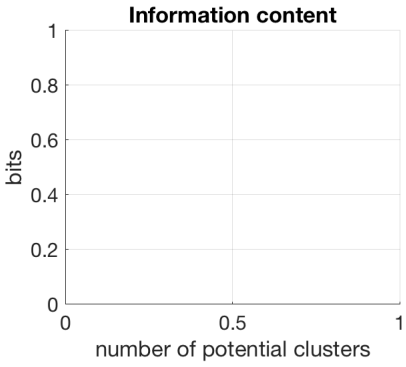

True cortex parcellation

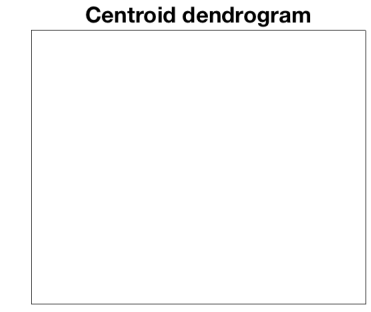

GCM cortex parcellation

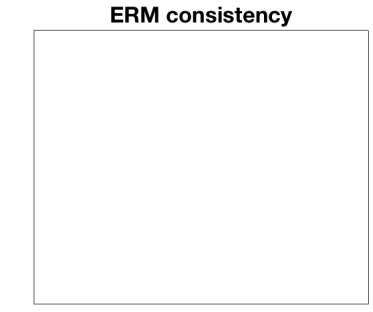

Connectivity matrix
seed voxels
target voxels

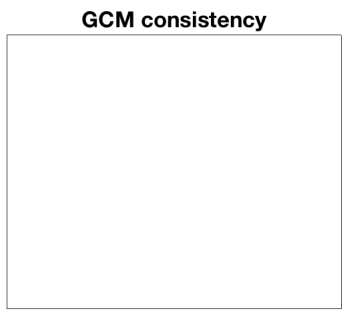

Dissimilarity matrix
seed voxels
seed voxels

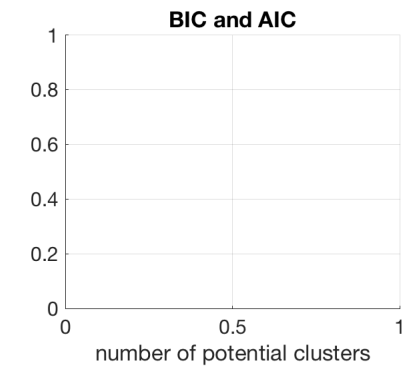

Generalization capacity
bits
inverse temperature (complexity)

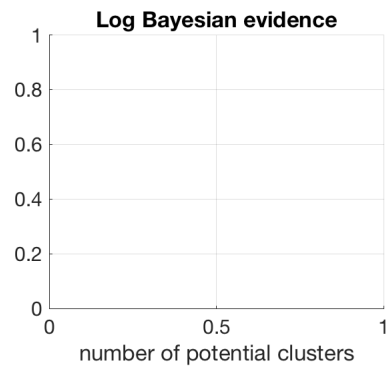

Information content
bits
number of potential clusters

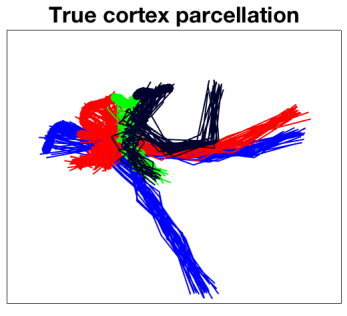

Centroid dendrogram

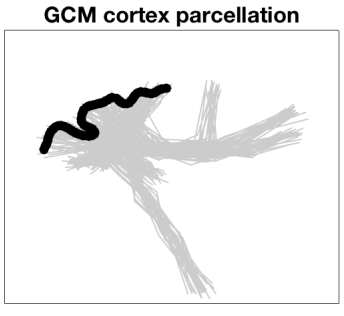

ERM consistency

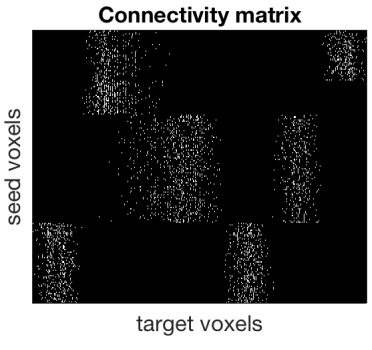

GCM consistency

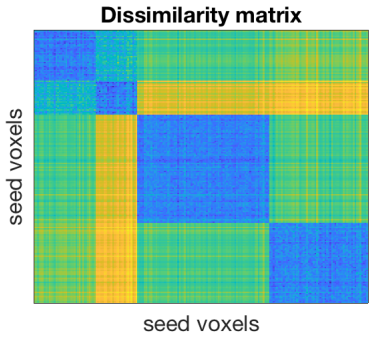

BIC and AIC
number of potential clusters

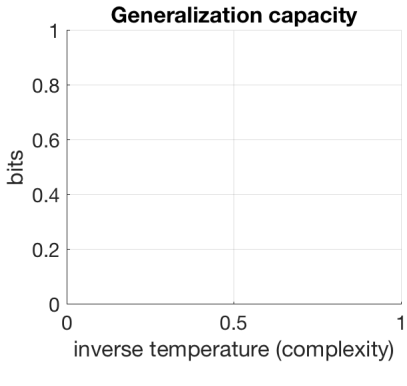

Log Bayesian evidence
number of potential clusters

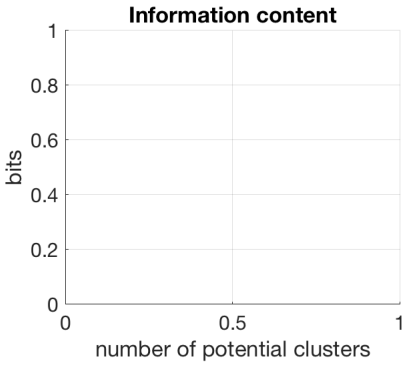

True cortex parcellation

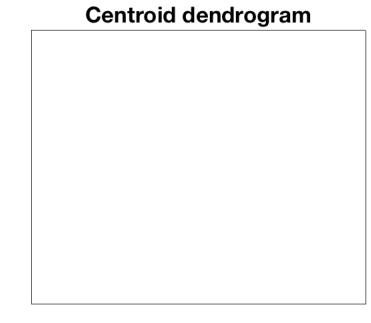

GCM cortex parcellation

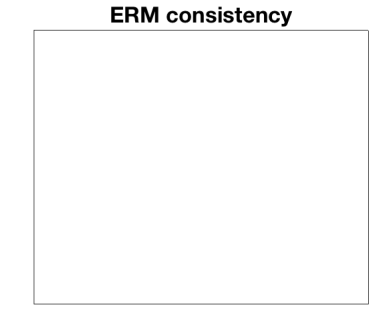

Connectivity matrix
seed voxels
target voxels

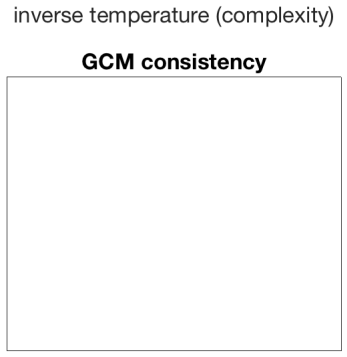

Dissimilarity matrix
seed voxels
seed voxels

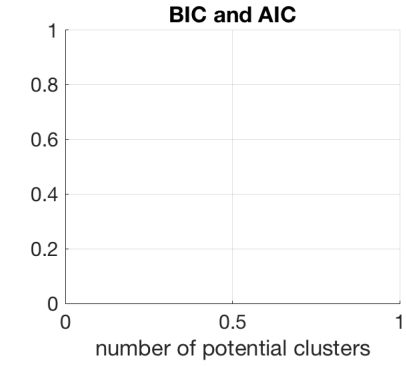

Generalization capacity
bits
inverse temperature (complexity)

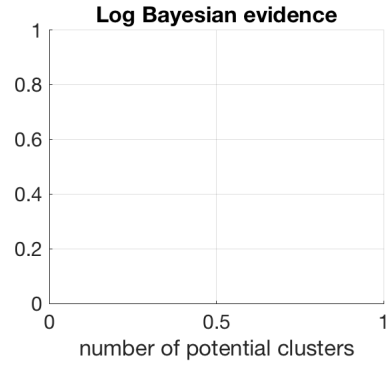

Information content
bits
number of potential clusters

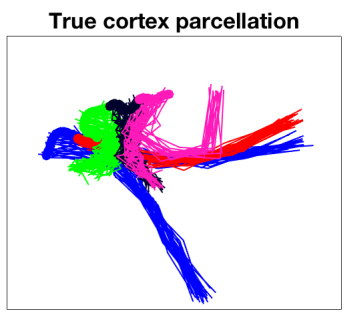

Centroid dendrogram

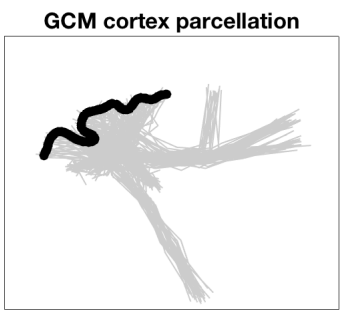

ERM consistency

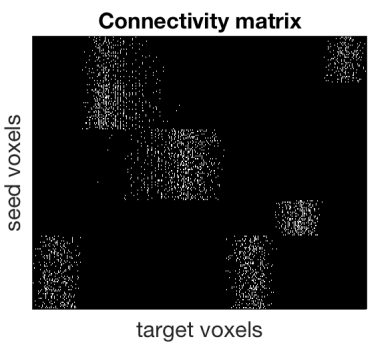

GCM consistency

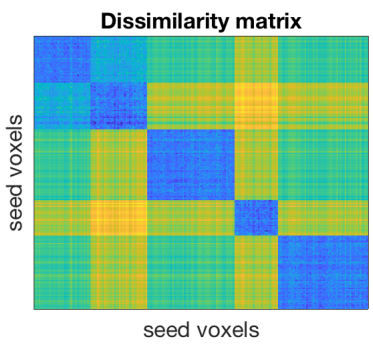

BIC and AIC
number of potential clusters

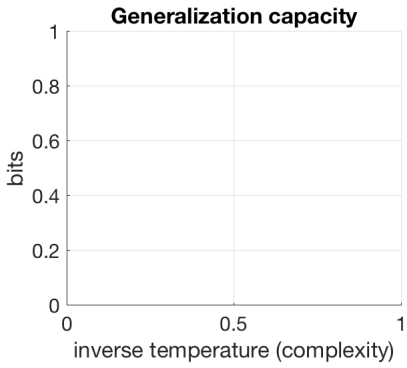

Log Bayesian evidence
number of potential clusters

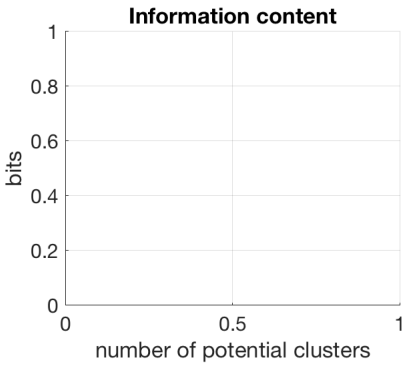

True cortex parcellation

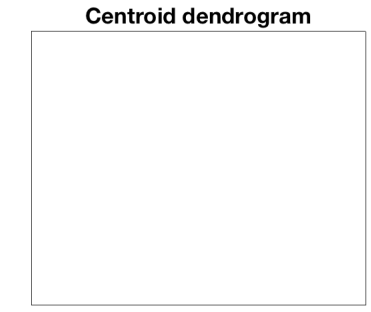

GCM cortex parcellation

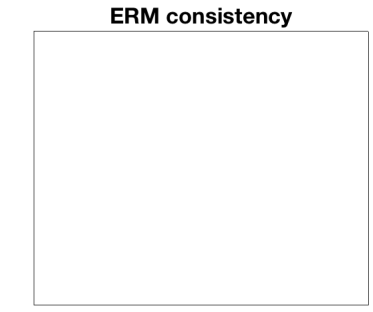

Connectivity matrix
seed voxels
target voxels

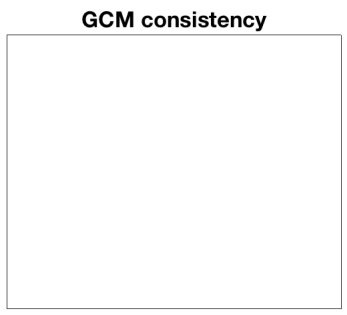

Dissimilarity matrix
seed voxels
seed voxels

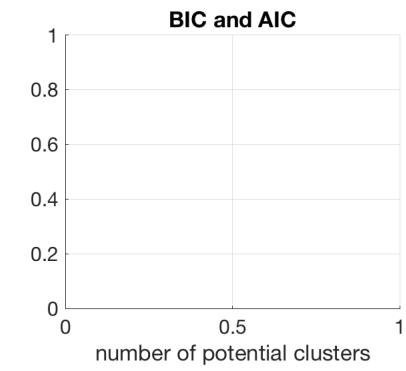

Generalization capacity
bits
inverse temperature (complexity)

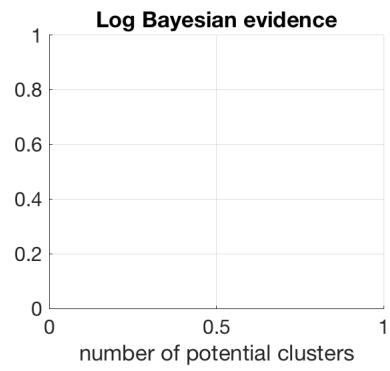

Information content
bits
number of potential clusters

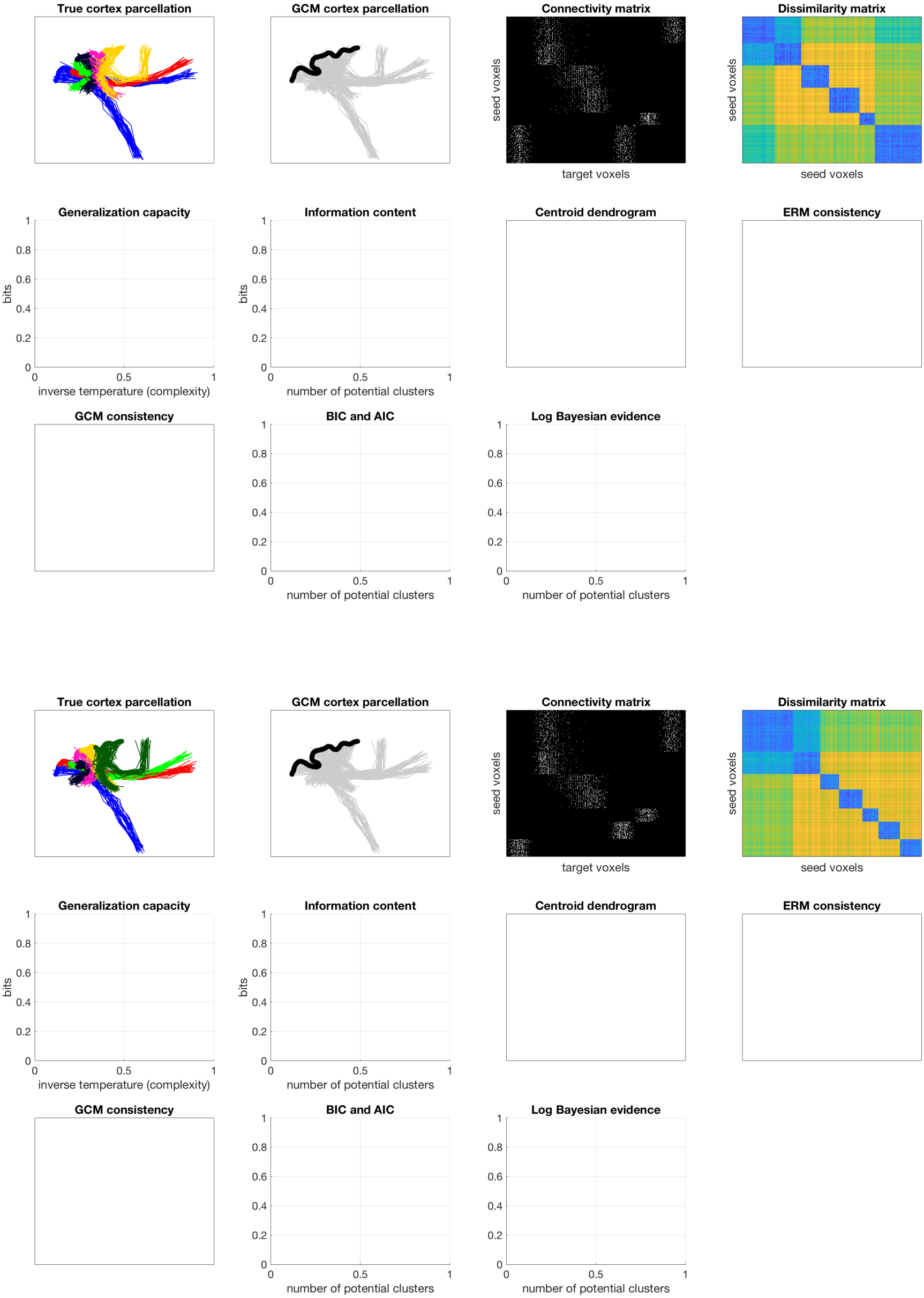

Centroid dendrogram


ERM consistency


GCM consistency


BIC and AIC
number of potential clusters


Log Bayesian evidence
number of potential clusters




```
% Bayesian evidence hyperparameters

alpha{1} = full(sum(data{1},1));
alpha{2} = full(sum(data{1},1))/100;
```

Deterministic annealing

Determine global minimizer.

```
% Annealing settings
beta_init = 1e-4;     % starting inverse temperature
```

```
beta_step = 1.1;      % inverse temperature step
beta_stop = 1e-1;      % stopping inverse temperature
perturb_sd = 1e-4;    % centroid perturbation

% Initialization of information content
info_content = zeros(1,max(K));

for k = K
```

```
    % Initialization of Gibbs distributions
    gibbs_dist1 = ones(size(data{1},1),k) ./ k;
    gibbs_dist2 = ones(size(data{2},1),k) ./ k;

    % Initialization of centroids
    centroid1 = gibbs_dist1'*data{1};
    centroid1 = bsxfun(@rdivide,centroid1,sum(centroid1,2));
    centroid1(centroid1==0) = eps;  % smoothing
    centroid2 = gibbs_dist2'*data{2};
    centroid2 = bsxfun(@rdivide,centroid2,sum(centroid2,2));
    centroid2(centroid2==0) = eps;  % smoothing

    j = 0; beta = beta_init; inv_temp = []; %subplot(2,3,4); cla
    while beta <= beta_stop
```

Perturb centroids

Avoid local minimum by perturbing centroids: $\phi_{kj}^{(\cdot)} = \phi_{kj}^{(\cdot)} + \epsilon$

```
        centroid1 = centroid1 + perturb_sd * rand(size(centroid1));
        centroid1 = bsxfun(@rdivide,centroid1,sum(centroid1,2)); % normalize
        centroid2 = centroid2 + perturb_sd * rand(size(centroid2));
        centroid2 = bsxfun(@rdivide,centroid2,sum(centroid2,2)); % normalize
```

Expectation maximization

Iterate between determining Gibbs distributions and maximzing variational lower bound w.r.t. centroids.

```
        for iter = 1:10
```

Costs for histogram clustering given instance 1

KL divergence between empirical probabilities (data) and centroid probabilities (up to proportionality constant): $R_{ik}^{(1)} = -\sum_j x_{ij}^{(1)} \log(\phi_{kj}^{(1)})$

```
            potential1 = -data{1} * log(centroid1)';
```

Costs for histogram clustering given instance 2

KL divergence between empirical probabilities (data) and centroid probabilities (up to proportionality constant): $R_{ik}^{(2)} = -\sum_j x_{ij}^{(2)} \log(\phi_{kj}^{(2)})$

```
            potential2 = -data{2} * log(centroid2)';
```

Gibbs distribution 1

Maximum entropy distribution: $p_{ik}^{(1)} = \exp\left(-\beta R_{ik}^{(1)}\right)/Z$

```
            gibbs_dist1 = exp(-beta * potential1);
            partition_sum1 = sum(gibbs_dist1,2);
            gibbs_dist1 = bsxfun(@rdivide,gibbs_dist1,partition_sum1);

            % avoid underflow
            idx = find(partition_sum1==0);
            if ~isempty(idx)
                [~,min_cost_idx] = min(potential1(idx,:),[],2);
                max_ind = sub2ind(size(gibbs_dist1),idx,min_cost_idx);
                gibbs_dist1(idx,:) = zeros(length(idx),k);
                gibbs_dist1(max_ind) = 1;
            end
```

Gibbs distribution 2

Maximum entropy distribution: $p_{ik}^{(2)} = \exp\left(-\beta R_{ik}^{(2)}\right)/Z$

```
            gibbs_dist2 = exp(-beta * potential2);
            partition_sum2 = sum(gibbs_dist2,2);
            gibbs_dist2 = bsxfun(@rdivide,gibbs_dist2,partition_sum2);

            % avoid underflow
            idx = find(partition_sum2==0);
            if ~isempty(idx)
                [~,min_cost_idx] = min(potential2(idx,:),[],2);
                max_ind = sub2ind(size(gibbs_dist2),idx,min_cost_idx);
                gibbs_dist2(idx,:) = zeros(length(idx),k);
                gibbs_dist2(max_ind) = 1;
            end
```

Joint Gibbs distribution

Maximum entropy distribution: $p_{ik}^{(1,2)} = \exp\left(-\beta(R_{ik}^{(1)} + R_{ik}^{(2)})\right)/Z$

```
            dist_joint = exp(-beta * (potential1 + potential2));
            joint_partition_sum = sum(dist_joint,2);
```

Centroids for instance 1

Probability prototype: $\phi_{kj}^{(1)} = \frac{\sum_i p_{ik}^{(1)} x_{ij}^{(1)}}{\sum_j \sum_{i'} p_{i'k}^{(1)} x_{i'j}^{(1)}}$

```
            centroid1 = gibbs_dist1'*data{1};
            centroid1 = bsxfun(@rdivide,centroid1,sum(centroid1,2));
            centroid1(centroid1==0) = eps;
```

Centroids for instance 2

Probability prototype: $\phi_{kj}^{(2)} = \frac{\sum_i p_{ik}^{(2)} x_{ij}^{(2)}}{\sum_j \sum_{i'} p_{i'k}^{(2)} x_{i'j}^{(2)}}$

```
            centroid2 = gibbs_dist2'*data{2};
            centroid2 = bsxfun(@rdivide,centroid2,sum(centroid2,2));
            centroid2(centroid2==0) = eps;
        end
```

```
        % increase inverse temperature:
        beta = beta * beta_step;
```

Match clusters across data instances

Use Hungarian algorithm to match clusters.

```
match_clusters_idx = munkres(pdist2(centroid1,centroid2));
potential2=potential2(:,match_clusters_idx);
```

Log partition sum for instance 1

Determine log partition sum while avoiding underflow: $\log Z_1 = \sum_i \log \sum_k \exp\left(-\beta R_{ik}^{(1)}\right)$

```
scaled_cost1 = -beta * potential1;
% log-sum-exp trick to prevent underflow
max_scaled_cost1 = max(scaled_cost1,[],2);
log_partition_sum1 = max_scaled_cost1 + log(sum(exp(scaled_cost1-max_scaled_cost1),2));
```

Log partition sum for instance 2

Determine log partition sum while avoiding underflow: $\log Z_2 = \sum_i \log \sum_k \exp\left(-\beta R_{ik}^{(2)}\right)$

```
scaled_cost2 = -beta * potential2;
% log-sum-exp trick to prevent underflow
max_scaled_cost2 = max(scaled_cost2,[],2);
log_partition_sum2 = max_scaled_cost2 + log(sum(exp(scaled_cost2-max_scaled_cost2),2));
```

Joint log partition sum

Determine joint log partition sum while avoiding underflow: $\log Z_{12} = \sum_i \log \sum_k \exp\left(-\beta (R_{ik}^{(1)} + R_{ik}^{(2)})\right)$

```
joint_scaled_cost = -beta * (potential1 + potential2);
gibbs_dist_joint = bsxfun(@rdivide,joint_scaled_cost,sum(joint_scaled_cost,2));
% log-sum-exp trick to prevent underflow
max_scaled_cost3 = max(joint_scaled_cost,[],2);
log_joint_partition_sum = max_scaled_cost3 + log(sum(exp(joint_scaled_cost-max_scaled_cost3),2));
```

```
j = j+1;
```

Generalization capacity

Resolution of the hypothesis space: $GC(\beta) = \log(k) + \frac{1}{n}(\log Z_{12} - \log Z_1 - \log Z_2)$

```
gc{k}(j) = log(k) + sum(log_joint_partition_sum - log_partition_sum1 ...
    - log_partition_sum2) ./ size(partition_sum1,1);
```

```
% pack
inv_temp(j) = beta;
gibbs_dist_packed1{k}(:,:,j) = gibbs_dist1;
gibbs_dist_packed2{k}(:,:,j) = gibbs_dist2(:,match_clusters_idx);
```

```
end
```

Number of equivariant transformations

Richness of the hypothesis space: $\frac{1}{n}\log|\{\tau\}| = H(n_1/n,\ldots,n_k/n)$

```
gibbs_dist1 = round(gibbs_dist1);
d = sum(gibbs_dist1,1); d = d./sum(d); d(d==0) = 1;
nTransformations = -d * log(d)';

% correct generalization capacity
gc{k} = gc{k}-log(k)+nTransformations;

gc{k} = gc{k} * log2(exp(1));  % transforming units from nats to bits
```

Information content

Quality of algorithm: $\mathcal{I} = \max_\beta GC(\beta)$

```
[info_content(k),max_gc_idx(k)] = max(gc{k});
```

Bayesian Information Criterion (BIC)

$BIC := m \times k \times ln(n) + 2R(\mathbf{X},\hat{\theta})$ where m is the number of bins and $c^\perp$ is the empirical risk minimizer

```
cost = sum(sum(gibbs_dist1 .* (-data{1} * log(centroid1)')));
BIC(k) = size(data{1},2) * k * log(size(data{1},1)) + 2 * cost;
```

Akaike Information Criterion (AIC)

$AIC := 2 \times m \times k + 2R(\mathbf{X},\hat{\theta})$ where m is the number of bins and $c^\perp$ is the empirical risk minimizer

```
AIC(k) = 2 * size(data{1},2) * k + 2 * cost;
```

Bayesian evidence

$p(\mathbf{X} \mid \alpha) = \int p(\mathbf{X} \mid \phi) p(\theta \mid \alpha) d\phi = \prod_k \frac{B(\alpha_k + \mathbf{n}_k)}{B(\alpha_k)}$ where $\mathbf{n}_k := (n_{k1}, n_{k2}, \ldots, n_{kd})$ and $n_{kj} := \sum_{i:c(i)=k} x_{ij}$

```
[~,labels] = max(gibbs_dist1,[],2);
log_bayes_evidence{1}(k) = 0;
log_bayes_evidence{2}(k) = 0;
for c = unique(labels)'
    ncj = full(sum(data{1}(labels==c,:),1));
    alpha_tmp{1} = alpha{1}/k; alpha_tmp{1}(ncj==0) = [];
    alpha_tmp{2} = alpha{2}; alpha_tmp{2}(ncj==0) = [];
    ncj(ncj==0) = [];

    log_bayes_evidence{1}(k) = log_bayes_evidence{1}(k) + sum(gammaln(alpha_tmp{1} + ncj)) ...
        + gammaln(sum(alpha_tmp{1})) - gammaln(sum(alpha_tmp{1} + ncj)) - sum(gammaln(alpha_tmp{1}));
    log_bayes_evidence{2}(k) = log_bayes_evidence{2}(k) + sum(gammaln(alpha_tmp{2} + ncj)) ...
        + gammaln(sum(alpha_tmp{2})) - gammaln(sum(alpha_tmp{2} + ncj)) - sum(gammaln(alpha_tmp{2}));
end
```

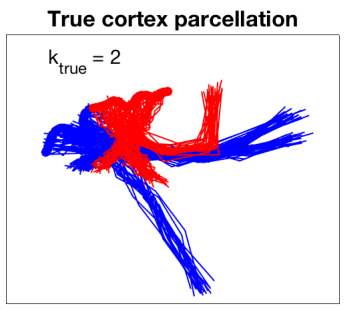
```
end
```

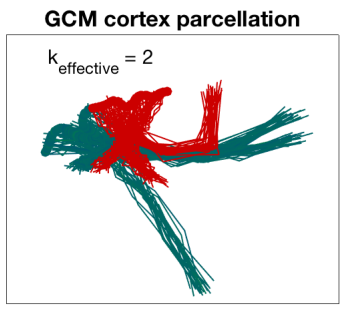
Results

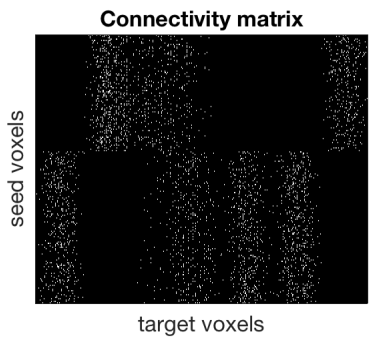
```
[ktrue2,centroid_labels] = display_result(info_content,gc,inv_temp,BIC,AIC,log_bayes_evidence,...
    gibbs_dist_packed1,gibbs_dist_packed2,K,p2,h,seed_coords,data,fiber_assignment);
```

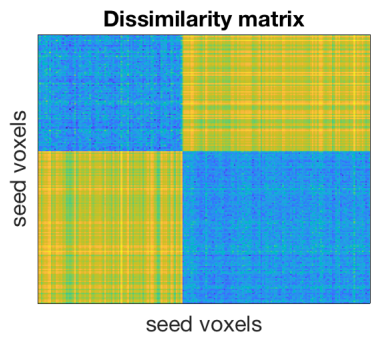
Update confusion matrices

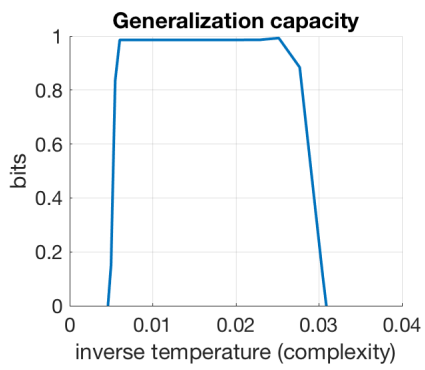
```
    confusion_matrix.PA(ktrue2,length(unique(centroid_labels))) = confusion_matrix.PA(ktrue2,length(unique(centroid_labels))) + 1;
    confusion_matrix.BIC(ktrue2,find(min(BIC(2:end))==BIC(2:end))+1) = confusion_matrix.BIC(ktrue2,find(min(BIC(2:end))==BIC(2:end))+1) + 1;
    confusion_matrix.AIC(ktrue2,find(min(AIC(2:end))==AIC(2:end))+1) = confusion_matrix.AIC(ktrue2,find(min(AIC(2:end))==AIC(2:end))+1) + 1;
    confusion_matrix.bayes_evidence1(ktrue2,find(max(log_bayes_evidence{1}(2:end))==log_bayes_evidence{1}(2:end))+1) = confusion_matrix.bayes_evidence1(ktrue,find(max(log_bayes_evidence{1}(2:end))==log_bayes_evidence{1
}(2:end))+1) + 1;
    confusion_matrix.bayes_evidence2(ktrue2,find(max(log_bayes_evidence{2}(2:end))==log_bayes_evidence{2}(2:end))+1) = confusion_matrix.bayes_evidence2(ktrue,find(max(log_bayes_evidence{2}(2:end))==log_bayes_evidence{2
}(2:end))+1) + 1;

    drawnow
```

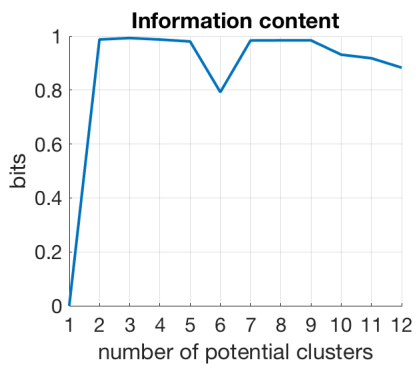


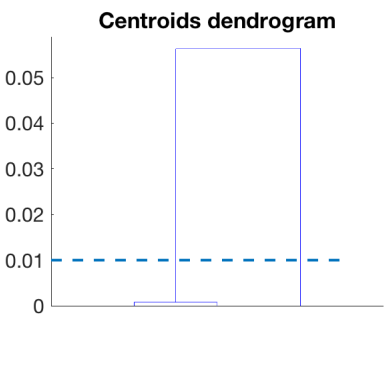


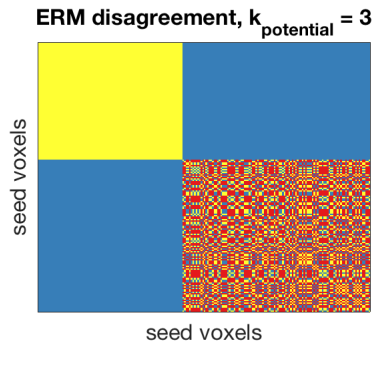


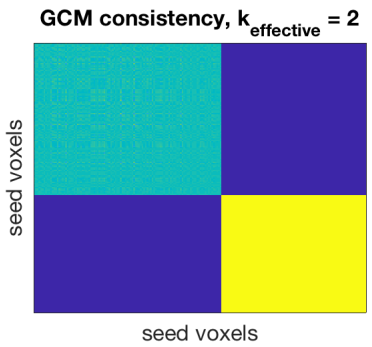


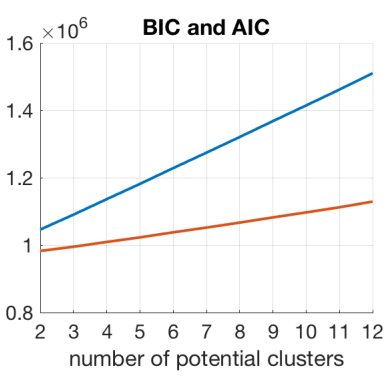


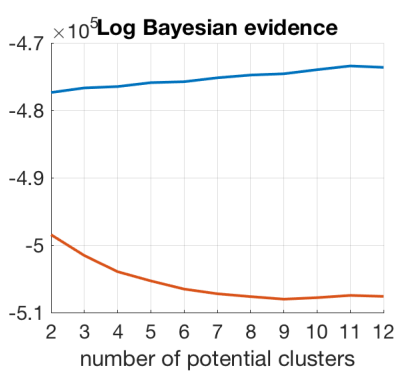


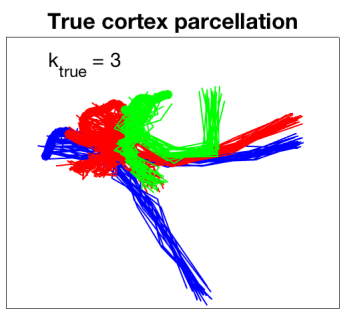


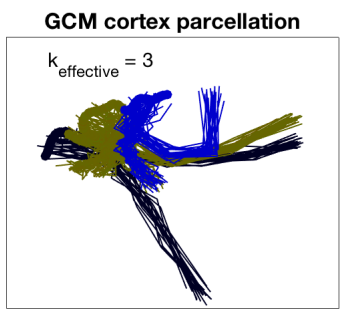


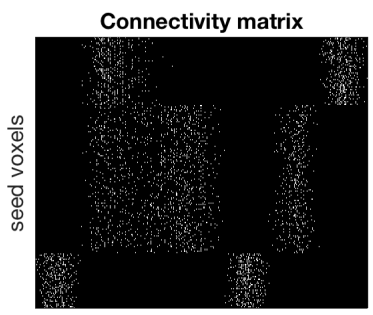


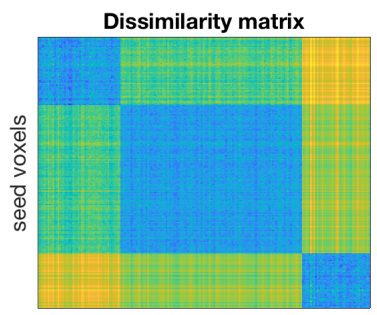


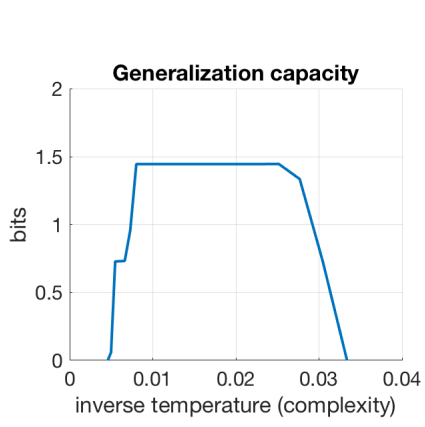


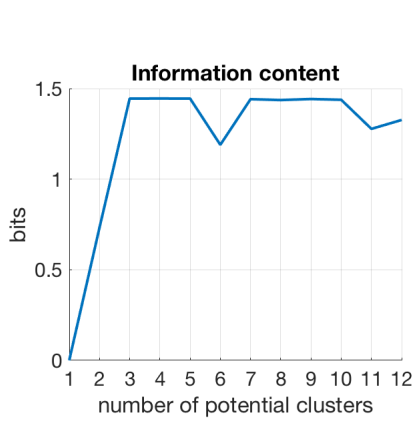


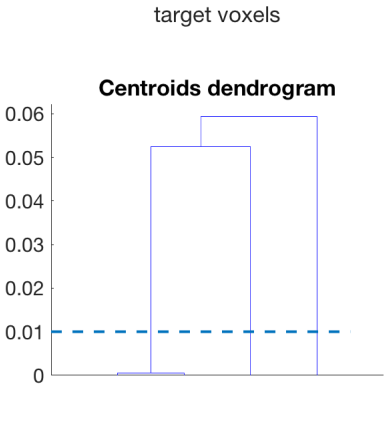


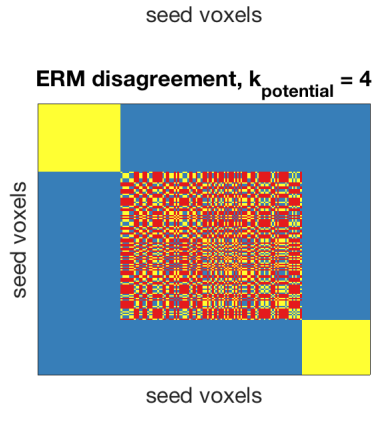


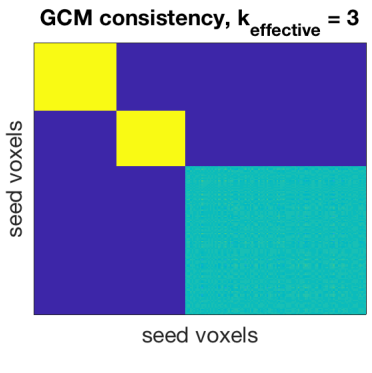


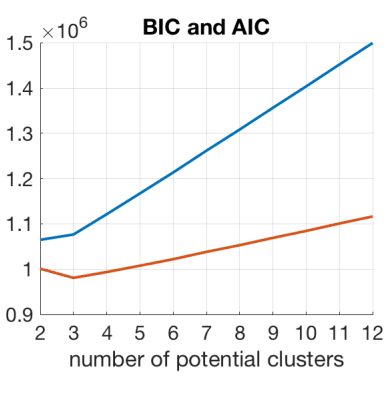


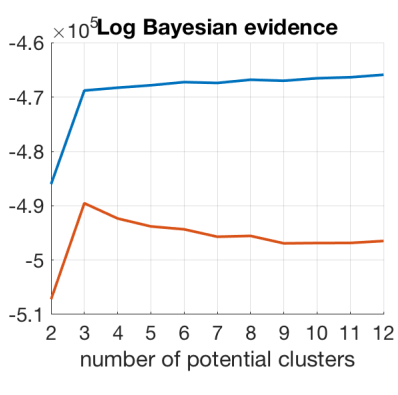


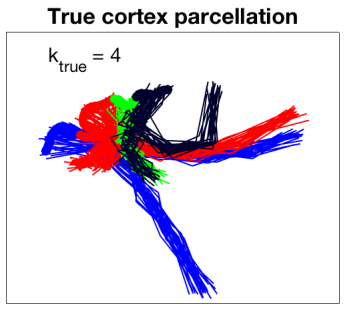


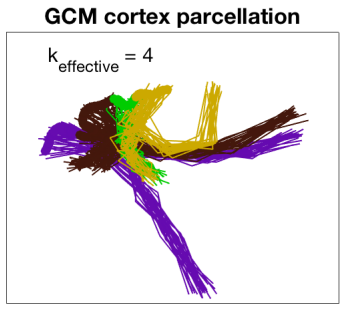


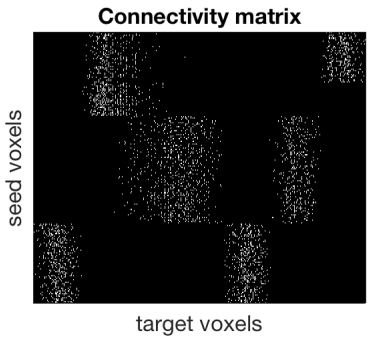


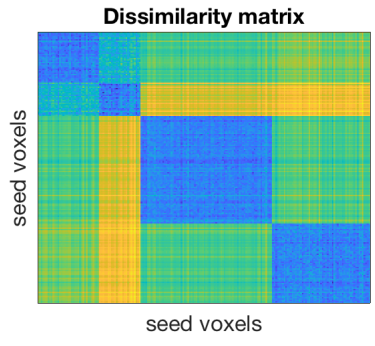


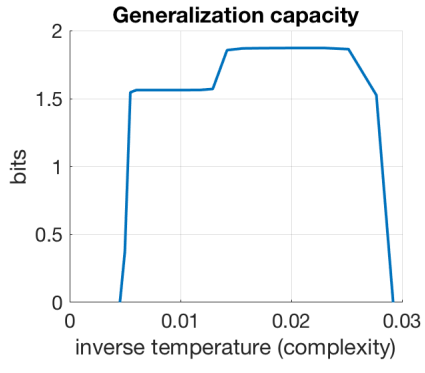


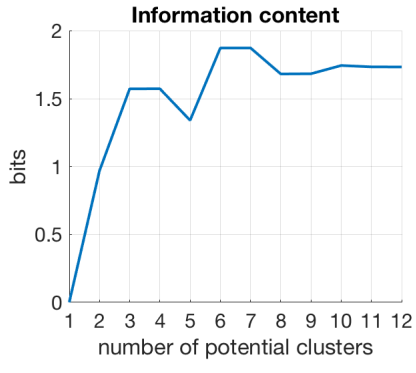

True cortex parcellation
k_true = 4

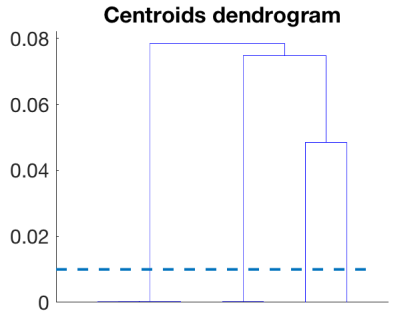

GCM cortex parcellation
k_effective = 4

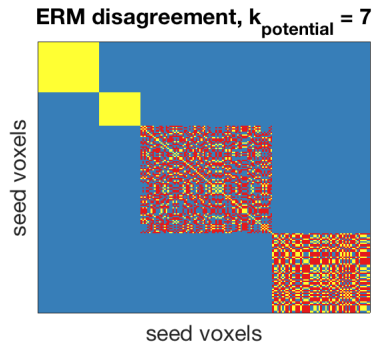

Connectivity matrix
seed voxels
target voxels

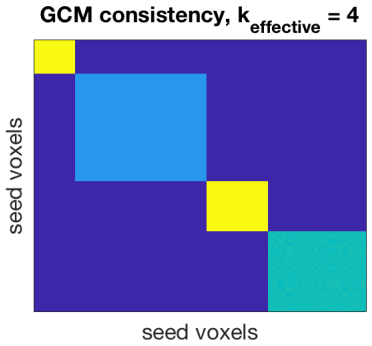

Dissimilarity matrix
seed voxels
seed voxels

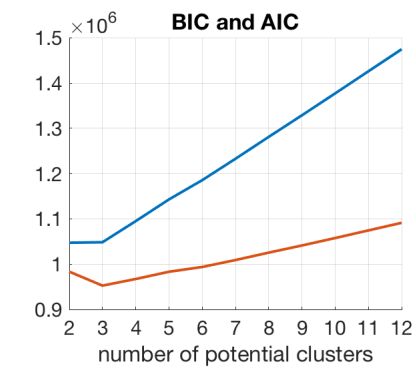

Generalization capacity
bits
inverse temperature (complexity)

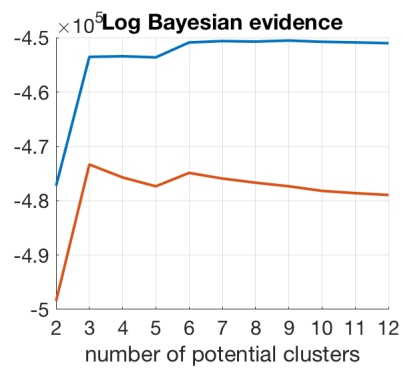

Information content
bits
number of potential clusters

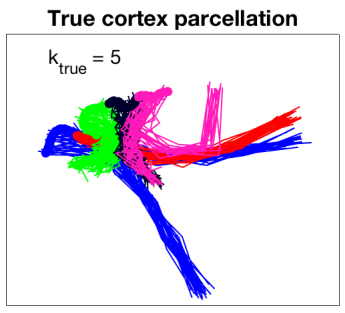

Centroids dendrogram

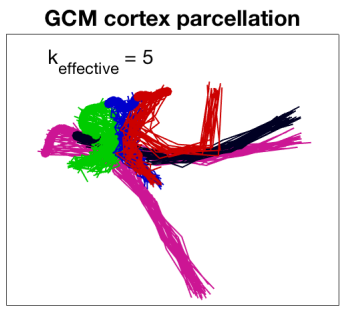

ERM disagreement, k_potential = 7
seed voxels
seed voxels

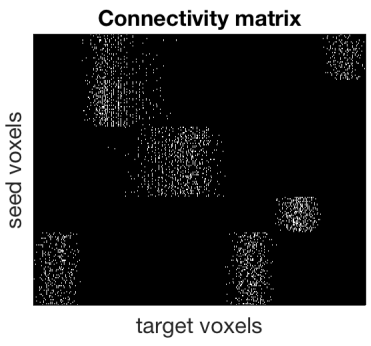

GCM consistency, k_effective = 4
seed voxels
seed voxels

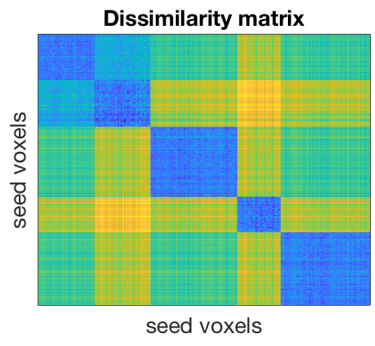

BIC and AIC
number of potential clusters

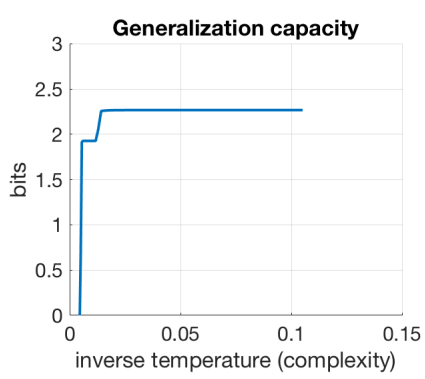

Log Bayesian evidence
number of potential clusters

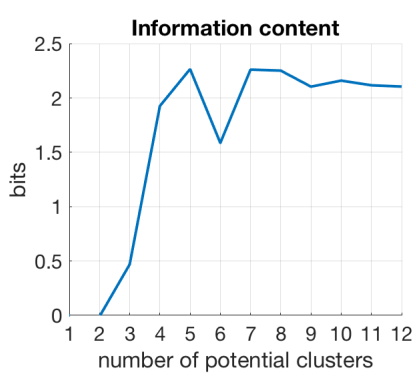

True cortex parcellation
k_true = 5

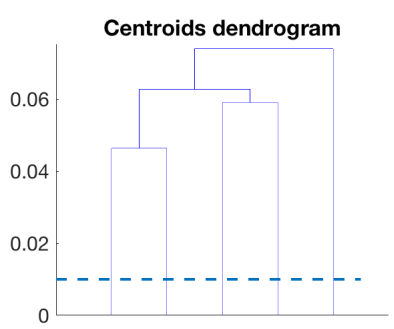

GCM cortex parcellation
k_effective = 5

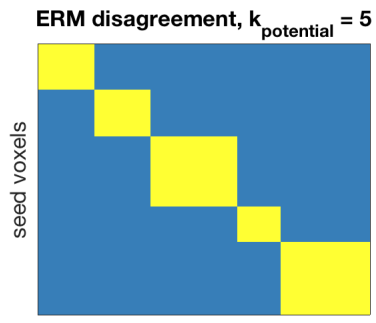

Connectivity matrix
seed voxels
target voxels

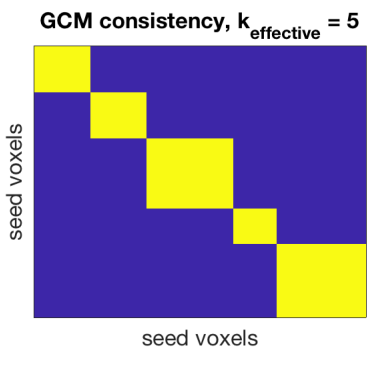

Dissimilarity matrix
seed voxels
seed voxels

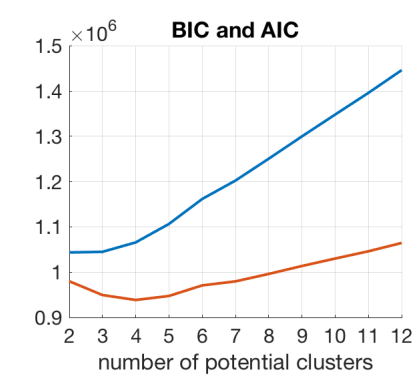

Generalization capacity
bits
inverse temperature (complexity)

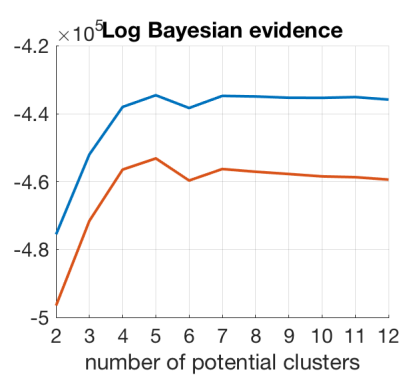

Information content
bits
number of potential clusters

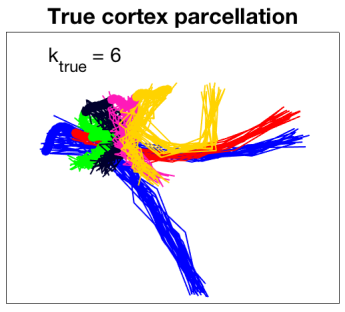

Centroids dendrogram

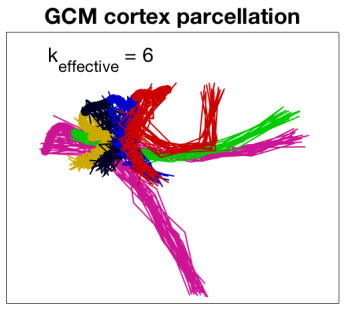

ERM disagreement, k_potential = 5
seed voxels
seed voxels

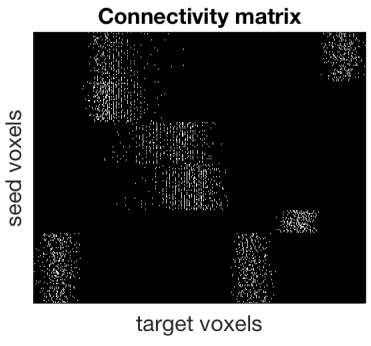

GCM consistency, k_effective = 5
seed voxels
seed voxels

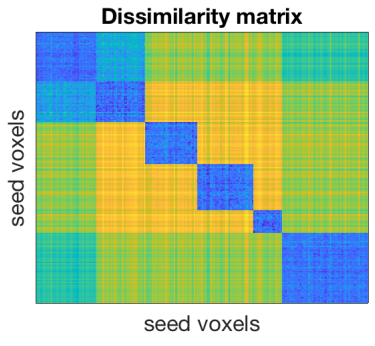

BIC and AIC
number of potential clusters

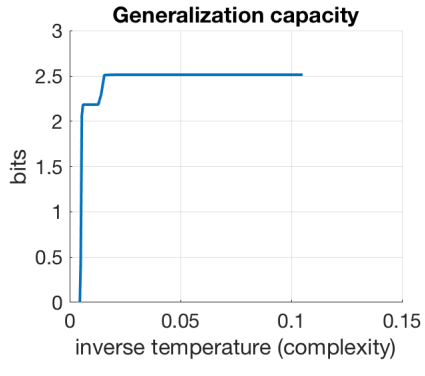

Log Bayesian evidence
number of potential clusters

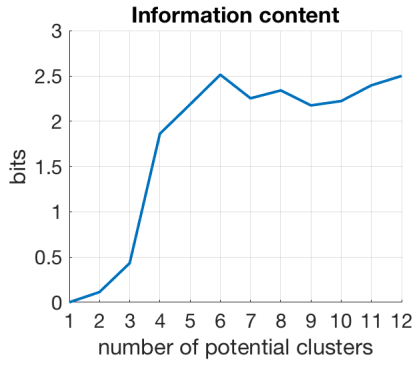


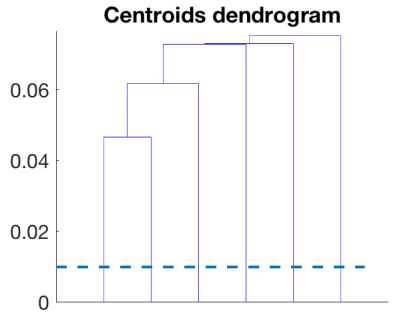


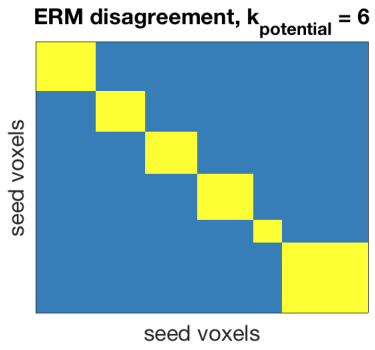


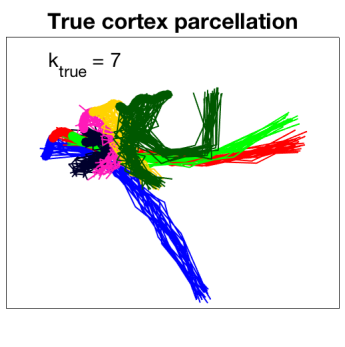


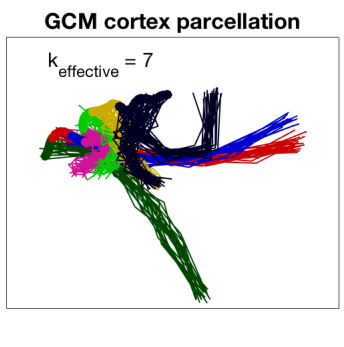


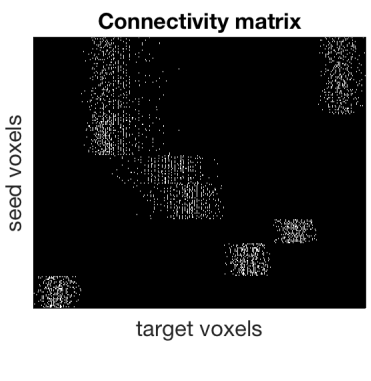


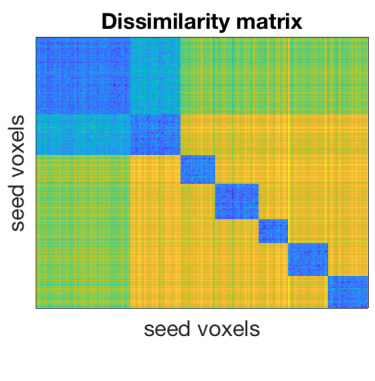


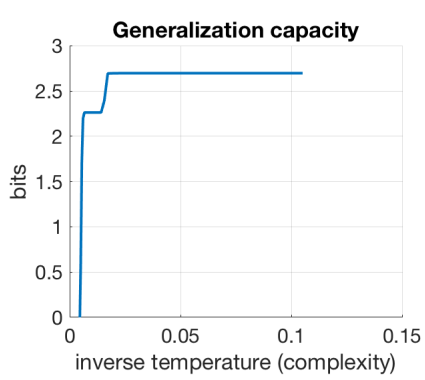


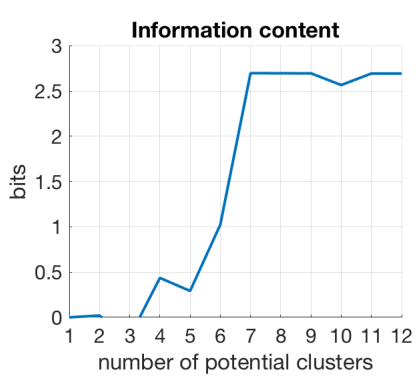


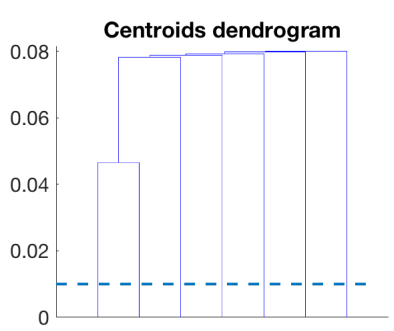


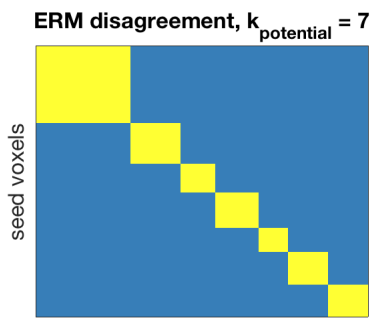


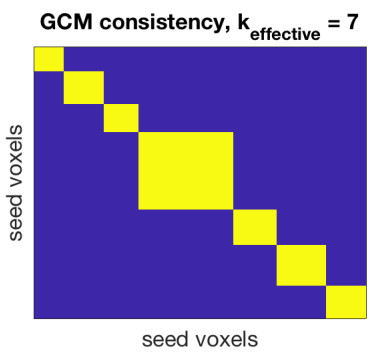


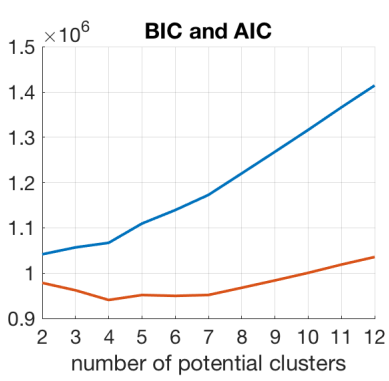


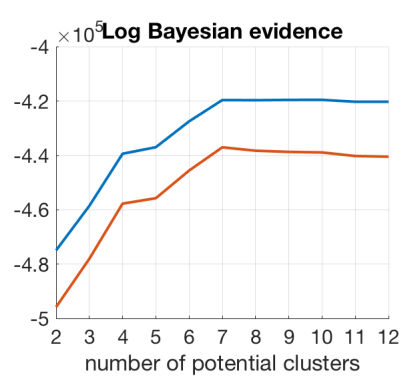













```
end
end
```

Confusion Matrix

```
disp('Confusion Matrices'); disp(' ');
T.PA = array2table(confusion_matrix.PA(2:end,2:end),'VariableNames',{'k2','k3','k4','k5','k6','k7','k8','k9','k10','k11','k12'},'RowNames',{'k2','k3','k4','k5','k6','k7'});
disp('Posterior Agreement:'); disp(T.PA);
T.BIC = array2table(confusion_matrix.BIC(2:end,2:end),'VariableNames',{'k2','k3','k4','k5','k6','k7','k8','k9','k10','k11','k12'},'RowNames',{'k2','k3','k4','k5','k6','k7'});
disp('BIC:'); disp(T.BIC);
T.AIC = array2table(confusion_matrix.AIC(2:end,2:end),'VariableNames',{'k2','k3','k4','k5','k6','k7','k8','k9','k10','k11','k12'},'RowNames',{'k2','k3','k4','k5','k6','k7'});
disp('AIC:'); disp(T.AIC);
```

```
T.bayes_evidence1 = array2table(confusion_matrix.bayes_evidence1(2:end,2:end),'VariableNames',{'k2','k3','k4','k5','k6','k7','k8','k9','k10','k11','k12'},'RowNames',{'k2','k3','k4','k5','k6','k7'});
disp('Bayes_evidence 1:'); disp(T.bayes_evidence1);
T.bayes_evidence2 = array2table(confusion_matrix.bayes_evidence2(2:end,2:end),'VariableNames',{'k2','k3','k4','k5','k6','k7','k8','k9','k10','k11','k12'},'RowNames',{'k2','k3','k4','k5','k6','k7'});
disp('Bayes_evidence 2:'); disp(T.bayes_evidence2);
```

```
Confusion Matrices

Posterior Agreement:
          k2    k3    k4    k5    k6    k7    k8    k9    k10    k11    k12
          __    __    __    __    __    __    __    __    ___    ___    ___

    k2    1     0     0     0     0     0     0     0     0      0      0
    k3    0     1     0     0     0     0     0     0     0      0      0
    k4    0     0     1     0     0     0     0     0     0      0      0
    k5    0     0     0     1     0     0     0     0     0      0      0
    k6    0     0     0     0     1     0     0     0     0      0      0
    k7    0     0     0     0     0     1     0     0     0      0      0

BIC:
          k2    k3    k4    k5    k6    k7    k8    k9    k10    k11    k12
          __    __    __    __    __    __    __    __    ___    ___    ___

    k2    1     0     0     0     0     0     0     0     0      0      0
    k3    1     0     0     0     0     0     0     0     0      0      0
    k4    1     0     0     0     0     0     0     0     0      0      0
    k5    1     0     0     0     0     0     0     0     0      0      0
    k6    1     0     0     0     0     0     0     0     0      0      0
    k7    1     0     0     0     0     0     0     0     0      0      0

AIC:
          k2    k3    k4    k5    k6    k7    k8    k9    k10    k11    k12
          __    __    __    __    __    __    __    __    ___    ___    ___

    k2    1     0     0     0     0     0     0     0     0      0      0
    k3    0     1     0     0     0     0     0     0     0      0      0
    k4    0     1     0     0     0     0     0     0     0      0      0
    k5    0     0     1     0     0     0     0     0     0      0      0
    k6    0     0     0     1     0     0     0     0     0      0      0
    k7    0     0     1     0     0     0     0     0     0      0      0

Bayes_evidence 1:
          k2    k3    k4    k5    k6    k7    k8    k9    k10    k11    k12
          __    __    __    __    __    __    __    __    ___    ___    ___

    k2    0     0     0     0     0     0     0     0     0      1      0
    k3    0     0     0     0     0     0     0     0     0      0      1
    k4    0     0     0     0     0     0     0     1     0      0      0
    k5    0     0     0     1     0     0     0     0     0      0      0
    k6    0     0     0     0     1     0     0     0     0      0      0
    k7    0     0     0     0     0     0     0     0     1      0      0

Bayes_evidence 2:
          k2    k3    k4    k5    k6    k7    k8    k9    k10    k11    k12
          __    __    __    __    __    __    __    __    ___    ___    ___

    k2    1     0     0     0     0     0     0     0     0      0      0
    k3    0     1     0     0     0     0     0     0     0      0      0
    k4    0     1     0     0     0     0     0     0     0      0      0
    k5    0     0     0     1     0     0     0     0     0      0      0
    k6    0     0     0     0     1     0     0     0     0      0      0
    k7    0     0     0     0     0     1     0     0     0      0      0
```

```
% display runtime
disp(['runtime = ' num2str(toc/60) ' minutes']);
```

```
runtime = 10.7506 minutes
```

Published with MATLAB® R2017a