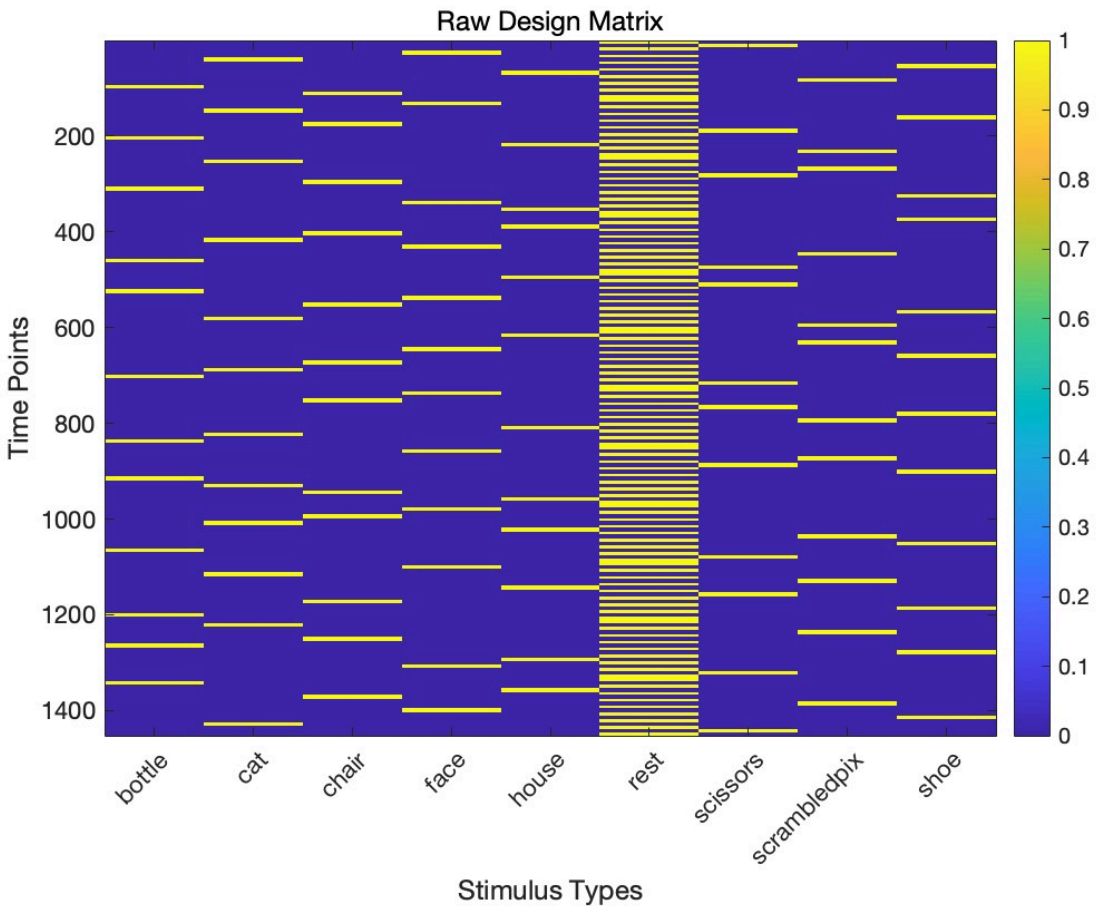
# MDMB Portfolio

## Project & Figures

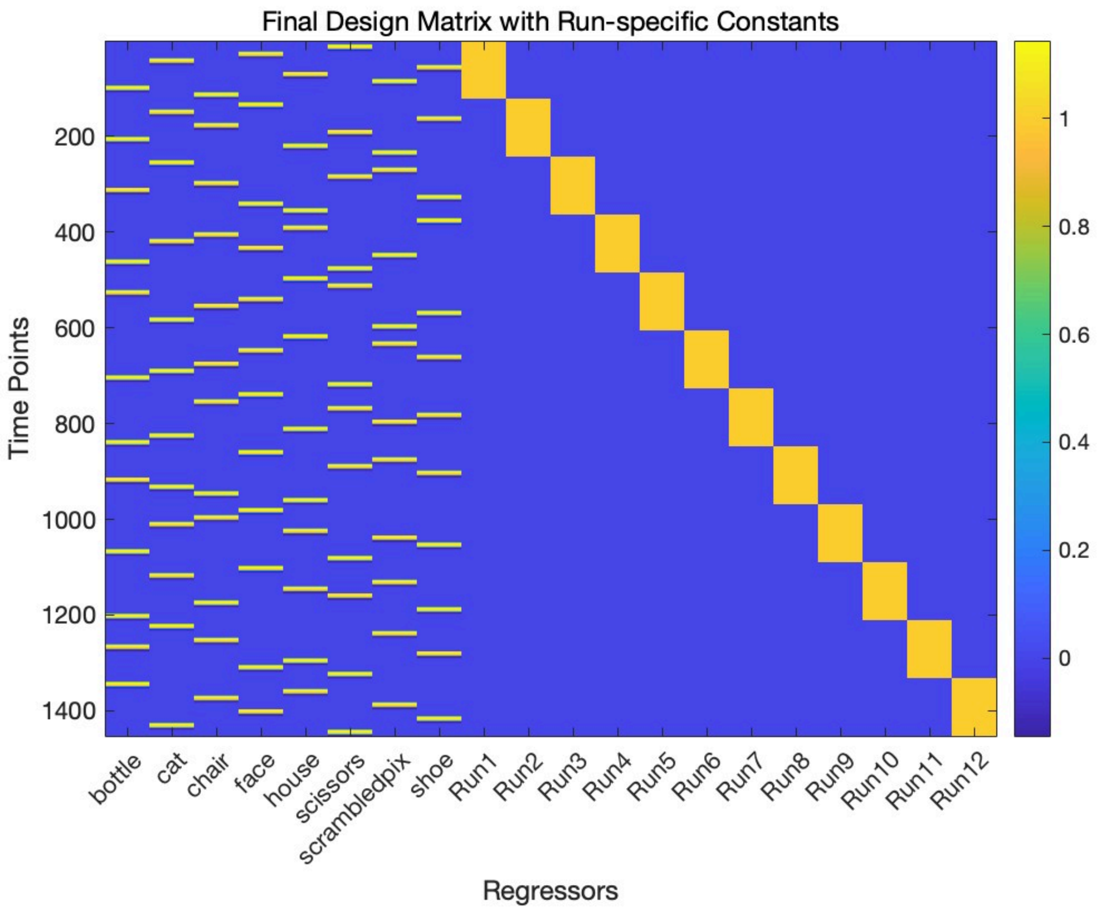
```
%%
% Step1: Load NIfTI data
data = niftiread('path_data');
[num_x, num_y, num_z, num_time_points] = size(data);
fmri_data = reshape(data, [], num_time_points)'; % Time point
s x Voxels

% Load labels file
labels_file = 'path_labels';
labels_data = readtable(labels_file, 'Delimiter', ' ', 'ReadV
ariableNames', false);
stimuli_conditions = labels_data.Var1;
condition_mapping = {'rest', 'scissors', 'face', 'cat', 'sho
e', 'house', 'scrambledpix', 'bottle', 'chair'};

% Every kind of stimuli matches one number
[unique_conditions, ~, condition_indices] = unique(stimuli_co
nditions);

% Condition mapping
fprintf('Condition Mapping:\n');
for i = 1:length(unique_conditions)
    fprintf('%s -> %d\n', unique_conditions{i}, i);
end
%%
% Step2: Construct the Design Matrix
num_conditions = length(unique_conditions);
design_matrix = zeros(num_time_points, num_conditions);

% Assign each condition to the corresponding column
```

```
for t = 1:num_time_points
    condition_id = condition_indices(t);
    design_matrix(t, condition_id) = 1;
end

% Design Matrix Visualization
figure;
imagesc(design_matrix);
xlabel('Stimulus Types');
ylabel('Time Points');
title('Raw Design Matrix');
colorbar;
xticks(1:length(unique_conditions));
xticklabels(unique_conditions);
xtickangle(45);
```



```
%%
% Step3: Get spatial and temporal resolution
nii_info = niftiinfo('path_nii');
voxel_size = nii_info.PixelDimensions;
TR = nii_info.raw.pixdim(5);
fprintf('Voxel size: %.2f x %.2f x %.2f mm\n', voxel_size);
fprintf('TR: %.2f seconds\n', TR);

%%
% Step4: HRF and Convolution
% remove 'rest'
condition_indices_no_rest = condition_indices;
rest_idx = find(strcmp(unique_conditions, 'rest'));
design_matrix_no_rest = design_matrix(:, setdiff(1:size(desig
n_matrix,2), rest_idx));

% Load pre-defined HRF from file
hrf_data = load('path_hrf');
disp('Available fields in hrf_data:');
disp(fieldnames(hrf_data));
hrf = hrf_data.hrf_sampled;
disp(['Loaded HRF size: ', num2str(size(hrf))]);

% Convolve each column with HRF
convolved_design_matrix = zeros(size(design_matrix_no_rest));
for c = 1:size(design_matrix_no_rest, 2)
    conv_result = conv(design_matrix_no_rest(:, c), hrf, 'ful
l');
    convolved_design_matrix(:, c) = conv_result(1:size(design
_matrix_no_rest,1));
end

% Add constant terms for each run (121 time points per run)
constants = [];
for ind = 1:12
    constants = [constants; zeros(1,(ind-1)*121) ones(1,121)
zeros(1,1452-ind*121)];
end

% Combine stimulus regressors with constants
final_design_matrix = [convolved_design_matrix constants'];

% Visualize final design matrix
figure;
imagesc(final_design_matrix);
xlabel('Regressors');
ylabel('Time Points');
title('Final Design Matrix with Run-specific Constants');
colorbar;
```

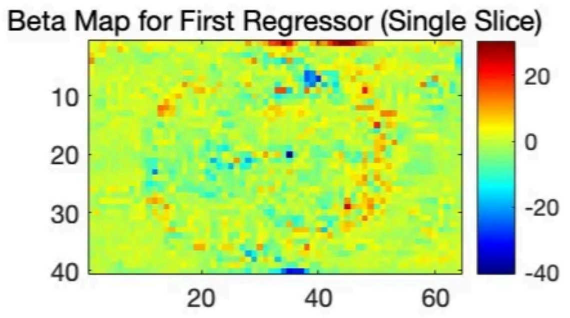
```
% Update condition labels for visualization
conditions_no_rest = unique_conditions(~strcmp(unique_conditi
ons, 'rest'));
if ~iscell(conditions_no_rest)
    conditions_no_rest = cellstr(conditions_no_rest);
end
constant_labels = arrayfun(@(x) sprintf('Run%d', x), 1:12, 'U
niformOutput', false);
xticks(1:size(final_design_matrix,2));
xticklabels([conditions_no_rest(:)', constant_labels]);
xtickangle(45);
```

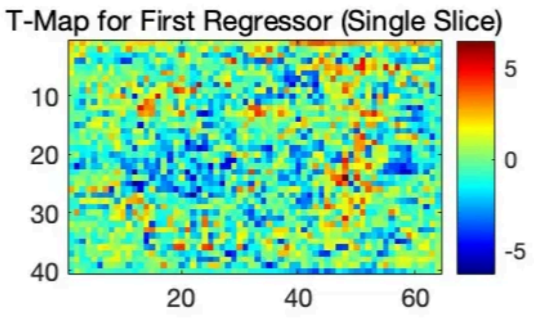


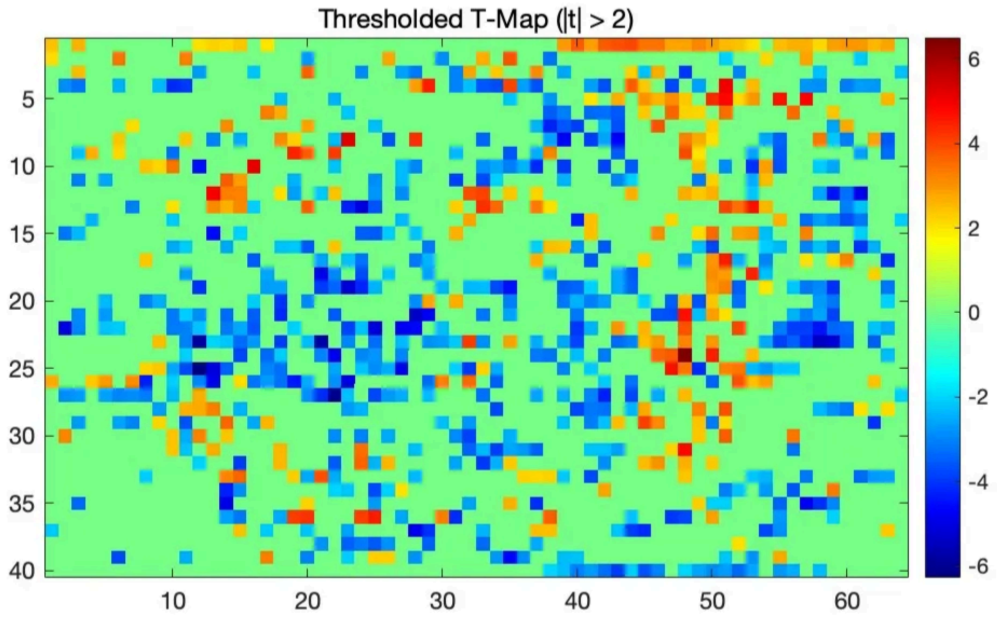
```
%%
% Step5: GLM Analysis
```

```
slice_z = round(num_z/2);
slice_data = data(:,:,slice_z,:);
slice_voxels = reshape(slice_data, [], num_time_points)';

% Make sure data type is double
slice_voxels = double(slice_voxels);
final_design_matrix = double(final_design_matrix);

% Print to make sure it fits
disp(['Size of slice_voxels: ', num2str(size(slice_voxel
s))]);
disp(['Size of final_design_matrix: ', num2str(size(final_des
ign_matrix))]);

% Dimension match
if size(slice_voxels, 1) ~= size(final_design_matrix, 1)
    error('Dimension mismatch: Number of time points does not
match between data and design matrix');
end

% GLM
B = zeros(size(final_design_matrix, 2), size(slice_voxels,
2));
t_stats = zeros(size(final_design_matrix, 2), size(slice_voxe
ls, 2));

for voxel = 1:size(slice_voxels, 2)
    y = slice_voxels(:, voxel);
    if length(y) > size(final_design_matrix, 1)
        y = y(1:size(final_design_matrix, 1));
    end

    [b, ~, residuals, ~, stats] = regress(y, final_design_mat
rix);
    B(:, voxel) = b;
    % t value
    dfe = length(y) - rank(final_design_matrix);  % df
    mse = sum(residuals.^2)/dfe;  % mse

    % se matrix
    Xinv = pinv(final_design_matrix);
    se = sqrt(diag(Xinv*Xinv')*mse);  % se

    % t stat
    t_stats(:, voxel) = b./se;
end

% Visualization
% interested condition
condition_of_interest = 1;
B_map = reshape(B(condition_of_interest,:), [num_x, num_y]);
t_map = reshape(t_stats(condition_of_interest,:), [num_x, num
_y]);

figure;
subplot(1, 2, 1);
imagesc(B_map);
colorbar;
title('Beta Map for First Regressor (Single Slice)');
axis image;
colormap('jet');

subplot(1, 2, 2);
imagesc(t_map);
colorbar;
title('T-Map for First Regressor (Single Slice)');
axis image;
colormap('jet');

% t-map threshold
figure;
```

```
threshold = 2;  % threshold
t_map_thresholded = t_map;
t_map_thresholded(abs(t_map) < threshold) = 0;
imagesc(t_map_thresholded);
colorbar;
title(sprintf('Thresholded T-Map (|t| > %g)', threshold));
axis image;
colormap('jet');
```



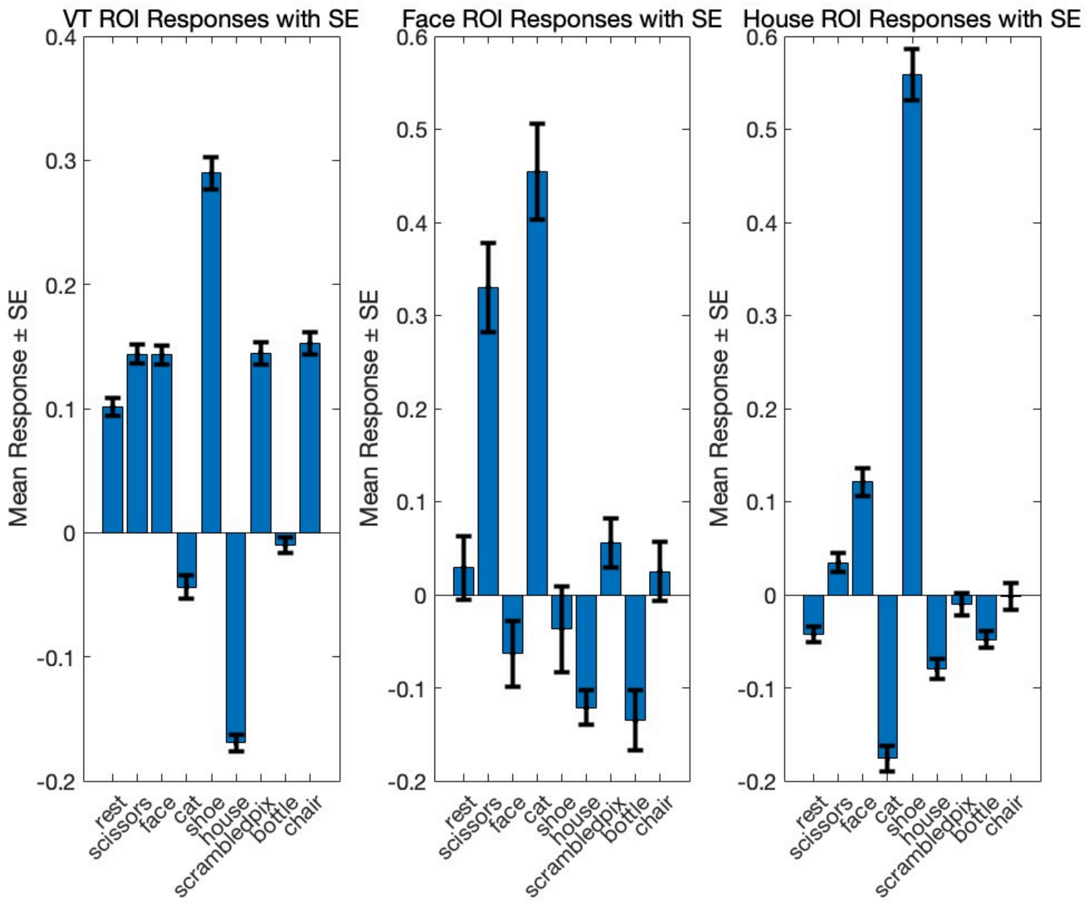




```
%%
% Step7: ROI Analysis
```

```matlab
% Convert fMRI data to double and Z-score
fmri_data = double(fmri_data);
fmri_data = zscore(fmri_data);

% Load ROI masks
mask_vt = niftiread('path_mask4_vt.nii.gz');
mask_face = niftiread('path_mask8_face_vt.nii.gz');
mask_house = niftiread('path_mask8_house_vt.nii.gz');

% Convert masks to double and extract voxel indices
mask_vt_voxels = find(double(mask_vt(:)) > 0);
mask_face_voxels = find(double(mask_face(:)) > 0);
mask_house_voxels = find(double(mask_house(:)) > 0);

% Extract BOLD signals for each ROI
time_series_vt = fmri_data(:, mask_vt_voxels);
time_series_face = fmri_data(:, mask_face_voxels);
time_series_house = fmri_data(:, mask_house_voxels);

% Compute Beta coefficients for each ROI
B_vt = pinv(design_matrix) * time_series_vt;
B_face = pinv(design_matrix) * time_series_face;
B_house = pinv(design_matrix) * time_series_house;

% Calculate mean responses and standard errors
mean_response_vt = mean(B_vt, 2);
mean_response_face = mean(B_face, 2);
mean_response_house = mean(B_house, 2);

se_vt = std(B_vt, [], 2) / sqrt(size(B_vt, 2));
se_face = std(B_face, [], 2) / sqrt(size(B_face, 2));
se_house = std(B_house, [], 2) / sqrt(size(B_house, 2));

% Plot results with error bars
figure;
subplot(1,3,1);
```

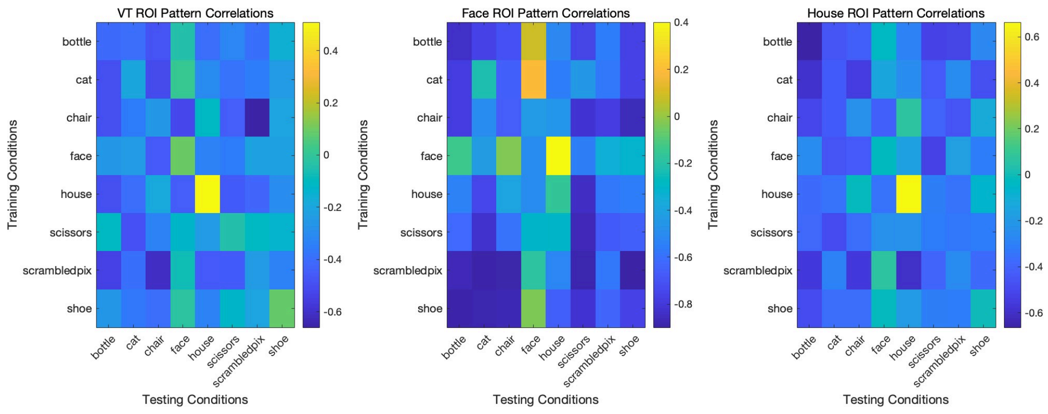
```
bar(mean_response_vt);
hold on;
errorbar(1:length(mean_response_vt), mean_response_vt, se_vt,
'k.', 'LineWidth', 2);
title('VT ROI Responses with SE');
set(gca, 'XTick', 1:length(condition_mapping), 'XTickLabel',
condition_mapping);
xtickangle(45);
ylabel('Mean Response ± SE');

subplot(1,3,2);
bar(mean_response_face);
hold on;
errorbar(1:length(mean_response_face), mean_response_face, se
_face, 'k.', 'LineWidth', 2);
title('Face ROI Responses with SE');
set(gca, 'XTick', 1:length(condition_mapping), 'XTickLabel',
condition_mapping);
xtickangle(45);
ylabel('Mean Response ± SE');

subplot(1,3,3);
bar(mean_response_house);
hold on;
errorbar(1:length(mean_response_house), mean_response_house,
se_house, 'k.', 'LineWidth', 2);
title('House ROI Responses with SE');
set(gca, 'XTick', 1:length(condition_mapping), 'XTickLabel',
condition_mapping);
xtickangle(45);
ylabel('Mean Response ± SE');
```

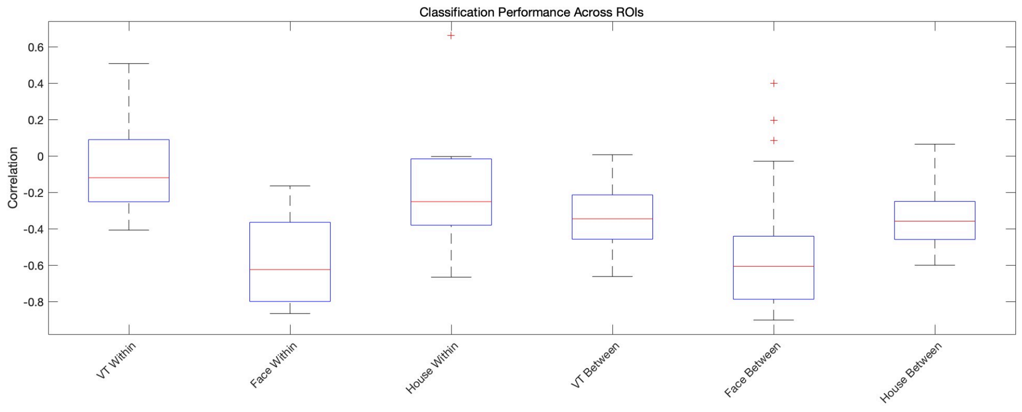


```
%%
% Step8: Classification Analysis

% Even & odd
runs = reshape(1:12, [], 1);  % runs
odd_runs = runs(1:2:end);
even_runs = runs(2:2:end);

% 121 time points per run
time_per_run = 121;
odd_indices = [];
even_indices = [];
for run = 1:12
    run_indices = ((run-1)*time_per_run + 1):(run*time_per_ru
n);
    if ismember(run, odd_runs)
        odd_indices = [odd_indices, run_indices];
    else
        even_indices = [even_indices, run_indices];
    end
end

% Design matrices
X_odd = final_design_matrix(odd_indices, 1:8);
X_even = final_design_matrix(even_indices, 1:8);

% ROI analysis
rois = {time_series_vt, time_series_face, time_series_house};
roi_names = {'VT', 'Face', 'House'};
classification_results = struct();

% All roi matrices
figure('Position', [100 100 1200 400]);
subplot(1,3,1);

for roi_idx = 1:length(rois)
    roi_data = rois{roi_idx};
    roi_name = roi_names{roi_idx};

    % Separate odd and even
    Y_odd = roi_data(odd_indices, :);
    Y_even = roi_data(even_indices, :);

    % beta maps
    B_odd = pinv(X_odd) * Y_odd;
    B_even = pinv(X_even) * Y_even;

    % correlation
    correlation_matrix = zeros(8, 8);
    for i = 1:8
```

```
        for j = 1:8
            correlation_matrix(i,j) = corr(B_odd(i,:)', B_eve
n(j,:)');
        end
    end

    % within-category & between-category
    within_corr = diag(correlation_matrix)';
    between_corr = correlation_matrix(~eye(size(correlation_m
atrix)));
    between_corr = between_corr(:)';
    classification_results.(roi_name).correlation_matrix = co
rrelation_matrix;
    classification_results.(roi_name).within_corr = within_co
rr;
    classification_results.(roi_name).between_corr = between_
corr;

    % matrices presentation
    subplot(1,3,roi_idx);
    imagesc(correlation_matrix);
    colorbar;
    title([roi_name ' ROI Pattern Correlations']);
    xlabel('Testing Conditions');
    ylabel('Training Conditions');
    conditions_no_rest = unique_conditions(~strcmp(unique_con
ditions, 'rest'));
    set(gca, 'XTick', 1:length(conditions_no_rest), 'XTickLab
el', conditions_no_rest);
    set(gca, 'YTick', 1:length(conditions_no_rest), 'YTickLab
el', conditions_no_rest);
    xtickangle(45);

    % statistical test
    [h,p] = ttest2(within_corr(:), between_corr(:));
    classification_results.(roi_name).ttest_p = p;
```

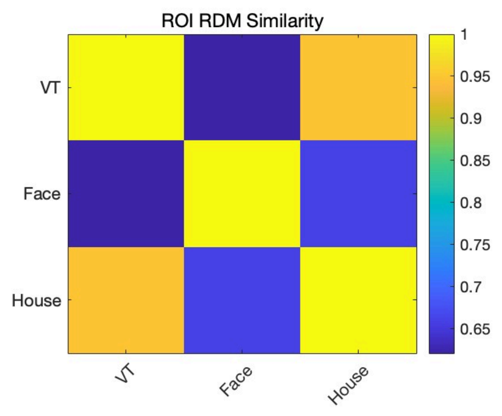
```
end

figure('Position', [100 100 1200 400]);
subplot(1,1,1);
all_within = [];
all_between = [];
roi_labels_within = {};
roi_labels_between = {};

% all ROI
for roi_idx = 1:length(rois)
    roi_name = roi_names{roi_idx};
    within_corr = classification_results.(roi_name).within_co
rr;
    between_corr = classification_results.(roi_name).between_
corr;

    all_within = [all_within; within_corr(:)];
    all_between = [all_between; between_corr(:)];

    roi_labels_within = [roi_labels_within; repmat({[roi_name
' Within']}, length(within_corr(:)), 1)];
    roi_labels_between = [roi_labels_between; repmat({[roi_na
me ' Between']}, length(between_corr(:)), 1)];
end

% all data and labels
all_data = [all_within; all_between];
all_labels = [roi_labels_within; roi_labels_between];

% boxplot
boxplot(all_data, all_labels);
title('Classification Performance Across ROIs');
ylabel('Correlation');
xtickangle(45);
```

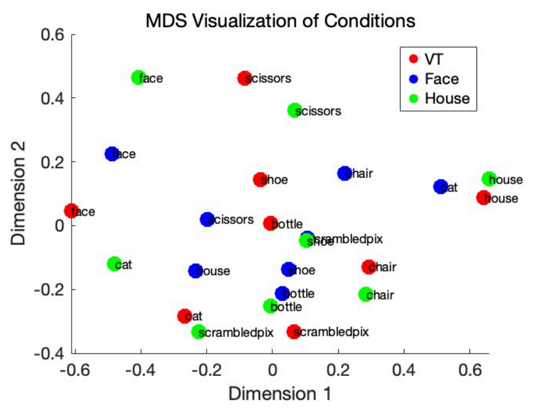


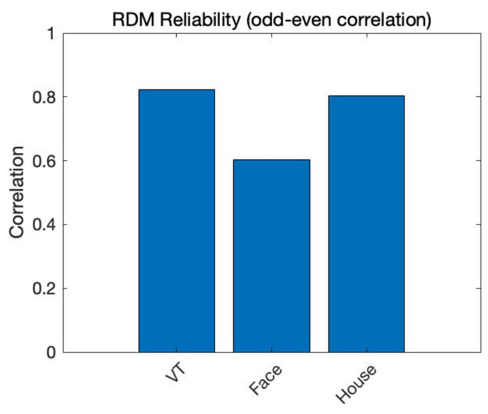


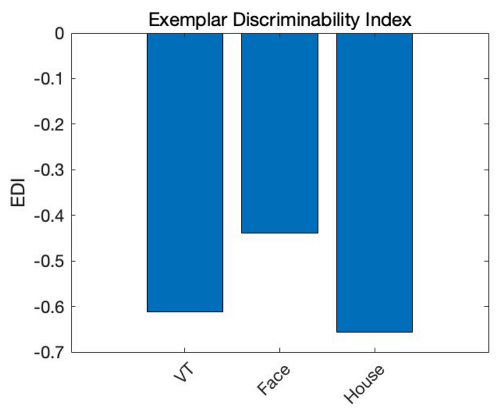
```
%%
% Step9: RDM Analysis

% Extract the activation pattern
rdm_results = struct();

% All ROI's RDM
figure('Position', [100 100 1500 400]);

for roi_idx = 1:length(rois)
    roi_data = rois{roi_idx};
    roi_name = roi_names{roi_idx};

    % separate odd and even
    Y_odd = roi_data(odd_indices, :);
    Y_even = roi_data(even_indices, :);

    % beta maps
    B_odd = pinv(X_odd) * Y_odd;
    B_even = pinv(X_even) * Y_even;

    % construction of RDM
    % pdist & squareform - dissimilarity
    rdm_odd = squareform(pdist(B_odd, 'correlation'));
    rdm_even = squareform(pdist(B_even, 'correlation'));
    rdm_results.(roi_name).rdm_odd = rdm_odd;
    rdm_results.(roi_name).rdm_even = rdm_even;

    % RDM Visualization
    subplot(1,3,roi_idx);
    imagesc(rdm_odd);
    colorbar;
    title([roi_name ' ROI RDM (Odd runs)']);
    xlabel('Conditions');
    ylabel('Conditions');
    conditions_no_rest = unique_conditions(~strcmp(unique_con
ditions, 'rest'));
    set(gca, 'XTick', 1:length(conditions_no_rest), 'XTickLab
el', conditions_no_rest);
    set(gca, 'YTick', 1:length(conditions_no_rest), 'YTickLab
el', conditions_no_rest);
    xtickangle(45);
end

% Comparing RDM similarity
figure('Position', [100 100 800 600]);
roi_similarity = zeros(length(rois));
for i = 1:length(rois)
    for j = 1:length(rois)
        roi1_rdm = rdm_results.(roi_names{i}).rdm_odd(:);
        roi2_rdm = rdm_results.(roi_names{j}).rdm_odd(:);
        roi_similarity(i,j) = corr(roi1_rdm, roi2_rdm);
    end
end

% Similarity matrix
subplot(2,2,1);
imagesc(roi_similarity);
colorbar;
title('ROI RDM Similarity');
set(gca, 'XTick', 1:length(roi_names), 'XTickLabel', roi_name
s);
set(gca, 'YTick', 1:length(roi_names), 'YTickLabel', roi_name
s);
xtickangle(45);

% MDS for visualization
subplot(2,2,2);
hold on;
colors = {'r', 'b', 'g'};
legend_entries = {};

for roi_idx = 1:length(rois)
    roi_name = roi_names{roi_idx};
    rdm = rdm_results.(roi_name).rdm_odd;

    rdm = (rdm + rdm')/2;  % RDM
    rdm(isnan(rdm)) = 0;   % NaN

    % MDS
    try
        mds_coords = mdscale(rdm, 2, 'Criterion', 'metricstre
ss');

        % MDS results
        scatter(mds_coords(:,1), mds_coords(:,2), 100, colors
{roi_idx}, 'filled');
        text(mds_coords(:,1), mds_coords(:,2), conditions_no_
rest, 'FontSize', 8);

        legend_entries{end+1} = roi_name;
    catch e
        warning('MDS failed for %s ROI: %s', roi_name, e.mess
age);
    end
end

hold off;
title('MDS Visualization of Conditions');
legend(legend_entries);
xlabel('Dimension 1');
ylabel('Dimension 2');

% Reliability
subplot(2,2,3);
reliability = zeros(1, length(rois));
for roi_idx = 1:length(rois)
    roi_name = roi_names{roi_idx};
    rdm_odd = rdm_results.(roi_name).rdm_odd(:);
    rdm_even = rdm_results.(roi_name).rdm_even(:);
    reliability(roi_idx) = corr(rdm_odd, rdm_even);
end

bar(reliability);
set(gca, 'XTick', 1:length(roi_names), 'XTickLabel', roi_name
s);
title('RDM Reliability (odd-even correlation)');
ylabel('Correlation');
xtickangle(45);

% Exemplar Discriminability Index
subplot(2,2,4);
```

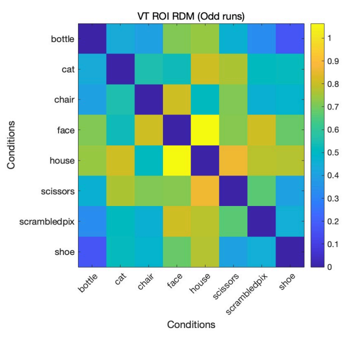
```
edi = zeros(1, length(rois));
for roi_idx = 1:length(rois)
    roi_name = roi_names{roi_idx};
    rdm_odd = rdm_results.(roi_name).rdm_odd;
    rdm_even = rdm_results.(roi_name).rdm_even;

    % diagonal & non-diagonal
    diagonal = diag(rdm_odd);
    off_diagonal = rdm_odd(~eye(size(rdm_odd)));

    % EDI calculaton
    edi(roi_idx) = mean(diagonal) - mean(off_diagonal);
end

bar(edi);
set(gca, 'XTick', 1:length(roi_names), 'XTickLabel', roi_name
s);
title('Exemplar Discriminability Index');
ylabel('EDI');
xtickangle(45);
```

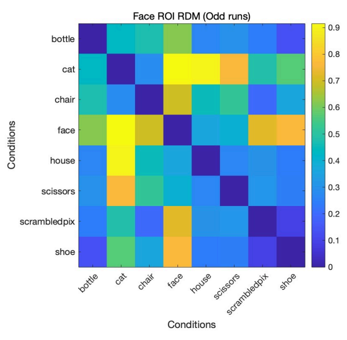


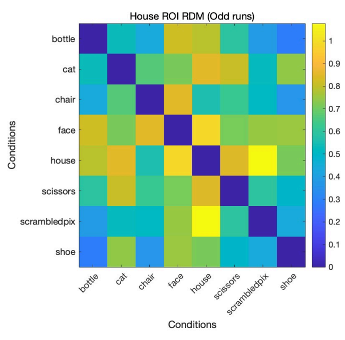












## Homework Questions:

### Homework 1:

Please read the book chapter on "Analysis of Functional MRI Data" (see Materials), and answer shortly to the following questions:

(1) What are the approximate spatial and temporal resolutions of fMRI data?

(2) What is the difference between a block and an event-related design?

(3) What is the aim of general linear modeling (GLM) of fMRI data?

## Answer to Homework 1:

1. Spatial Resolution: usually approximately 1mm*1mm*1mm for visualization of data. Temporal Resolution: usually limited by hemodynamic response time, typically the BOLD response has a width of 2-3s
2. Block Design: blocks are composed of stimulus of the same nature(~15s is the best for research) Event-Related Design: individual trials in different conditions are presented in a randomly generated sequence, there will be sufficient time for subjects to response to every trial. Also event-related design can avoid cognitive adaption and expectational strategies
3. GLM is one of the statistical approach used to analyze fMRI data. The goal of GLM is to explain the observed BOLD signal(one dependent variable) via a combination of various predictors in the experiment design(linear combination of several functions) because of its suitability for multiple level analysis.

## Homework 2:

Please read the review "Imaging retinotopic maps in the human brain" by Wandell and Winawer, and answer the following questions:

(1) What is a cortical visual field map?

(2) How do you measure a visual field map in the human brain?

(3) What is functional specialization?

## Answer to Homework 2:

1. A cortical visual field map is a representation of the visual field on the cortical surface of the brain. It is a spatial arrangement where specific areas of the brain correspond to specific regions of the visual field. Maps are arranged in the way that preserves the spatial relationship from the retina. They create a

retinotopic organization in the brain. Different regions of the visual cortex represent different parts of the visual field, with the foveal (central) vision often owning a larger cortical area due to cortical magnification.

2. Visual field maps in the human brain can be measured by fMRI. One of the primary methods is phase-encoded retinotopy, in there stimuli are presented to the subject. Then we record the brain’s response to these stimuli, allowing researchers to identify and map the visual field representation in specific cortical areas.
3. Functional specialization: different regions of the brain are specialized for different types of visual processing. In visual field maps, it means some areas of the visual cortex are more attuned to processing specific aspects of visual stimuli, such as color, motion, etc. Functional specialization shows that each area of the visual cortex processes a particular type of information to contribute to the overall visual perception.

## Homework 3:

Please read the book chapter "Tutorial on Pattern Classification in Functional Imaging" by Mur & Kriegeskorte  (see Materials), and answer the following questions:

(1) What is pattern-information analysis (also called multi-voxel pattern analysis, MVPA)?

(2) What is novel about pattern-information analysis compared to standard (univariate) fMRI analysis?

(3) In practice, how would you perform pattern-information analysis on fMRI data?

## Answer to Homework 3:

1. Pattern-information analysis, or MVPA, is a technique in functional magnetic resonance imaging  that examines the fine-grained patterns of activity across multiple voxels (the 3D pixels in brain imaging). Instead of just looking at overall activation levels in a brain region, MVPA focuses on the specific spatial patterns of activation. By analyzing these patterns, researchers can infer the representational content of neural populations, essentially decoding what information is being processed in particular brain areas.

2. Traditional univariate fMRI analyses assess each voxel independently, identifying regions where the average activation differs between conditions. This approach may miss subtle, yet informative, patterns distributed across voxels. MVPA, on the other hand, considers the multivariate pattern of activity across multiple voxels simultaneously. This allows for the detection of distributed information that univariate methods might overlook, providing a more nuanced understanding of how information is represented in the brain.
3. Data Collection - Data Splitting and Preprocessing -  Estimating Activity Patterns - Voxels Selection - Construction of Classifier - Testing the Classifier

## Homework 4:

Please read the review  "Representational geometry: integrating cognition, computation, and the brain" by Kriegeskorte and Kievit (see Materials), and answer the following questions:

(1) What is representational similarity analysis (RSA)?

(2) What is a representational dissimilarity matrix (RDM)?

(3) Give an example of a research question that could be addressed using RSA. This can be some of the examples given in the paper or you could think of a research question of your own.

## Answer to Homework 4:

1. Representational Similarity Analysis (RSA) is a computational framework used to investigate the representational structure of information within neural activity patterns. It allows us to quantify and compare these structures across different experimental conditions, brain regions, or species. By examining the pairwise similarity of activity patterns elicited by various stimuli or tasks, RSA facilitates an understanding of how information is encoded and processed in the brain.
2. A Representational Dissimilarity Matrix (RDM) is a core component of RSA that encapsulates the relationships between neural or representational patterns. Each entry in the RDM represents the dissimilarity between two conditions, typically derived from a measure such as Euclidean distance or correlation

distance between activity patterns. The RDM provides a compact, interpretable representation of how distinct different stimuli or conditions are from each other in a given neural or computational space.

3. Are impairments of perception and cognition in mental illness associated with atypical representational geometries? Can representational geometries be tracked over time to characterize the course of an illness and the effect of therapeutic interventions? RSA could be used to compare representational dissimilarity matrices (RDMs) between clinical and healthy populations. It can also be used to track individual's RDM over time.

## Homework 5:

Please read the review "Encoding and decoding in fMRI" by Naselaris et al (see Materials), and answer the following questions:

(1) What are the differences between decoding and encoding models?

(2) What are the main advantages of using an encoding model?

(3) What might be the reasons why decoding models are much more common in fMRI than encoding models?

## Answer to Homework 5:

1. The primary difference between encoding and decoding models lies in the direction of their predictions.Encoding models predict brain activity (fMRI responses) based on known experimental stimuli or tasks. They describe how specific features of a stimulus or task are represented in individual voxels of the brain.Decoding models use brain activity to predict information about the stimuli or tasks. They determine what can be inferred about external stimuli or cognitive states from observed voxel activity.Also, encoding models start with stimulus features and map them to brain responses, while decoding models start with brain responses and map them back to stimulus features.

2. A perfect encoding model can fully describe the function of a brain region (region of interest, ROI), it also allows researchers to systematically test and identify the specific features represented in a given brain region. Multiple encoding models can be directly compared to assess how different feature spaces contribute to activity in an ROI, something that is much harder to do

with decoding models, and a decoding model can be easily derived from an encoding model using Bayes’ theorem, but the reverse process is much more difficult.

3. Decoding models are more common in fMRI because decoding models can be directly compared to behavioral performance, allowing researchers to assess the functional relevance of specific patterns of brain activity. It's easy to operate and apply to Brain-Machine Interface as well. By pooling voxel activity patterns without averaging them, decoding models can also detect fine-grained information.