Representational Similarity Analysis

Contents

Written by Luke Chang

Representational Similarity Analysis (RSA) is a multivariate technique that allows one to link disparate types of data based on shared structure in their similarity (or distance) matrices. This technique was initially proposed by Nikolaus Kriegskorte in 2008. Unlike multivariate prediction/classification, RSA does not attempt to directly map brain activity onto a measure, but instead compares the similarities between brain activity and the measure using second-order isomorphisms. This sounds complicated, but is actually quite conceptually simple.

In this tutorial we will cover:

1. how to embed a stimuli in a feature space
2. how to compute similarity or distance within this feature space
3. how to map variations in position within this embedding space onto brain processes.

For a more in depth tutorial, we recommend reading this excellent tutorial written by Mark Thornton for the 2018 MIND Summer School, or watching his video walkthrough.

Let's work through a quick example to outline the specific steps involved in RSA to make this more concrete.

RSA Overview

Imagine that you view 96 different images in the scanner and we are interested in learning more about how the brain processes information about these stimuli. Do we categorize these

Skip to main content

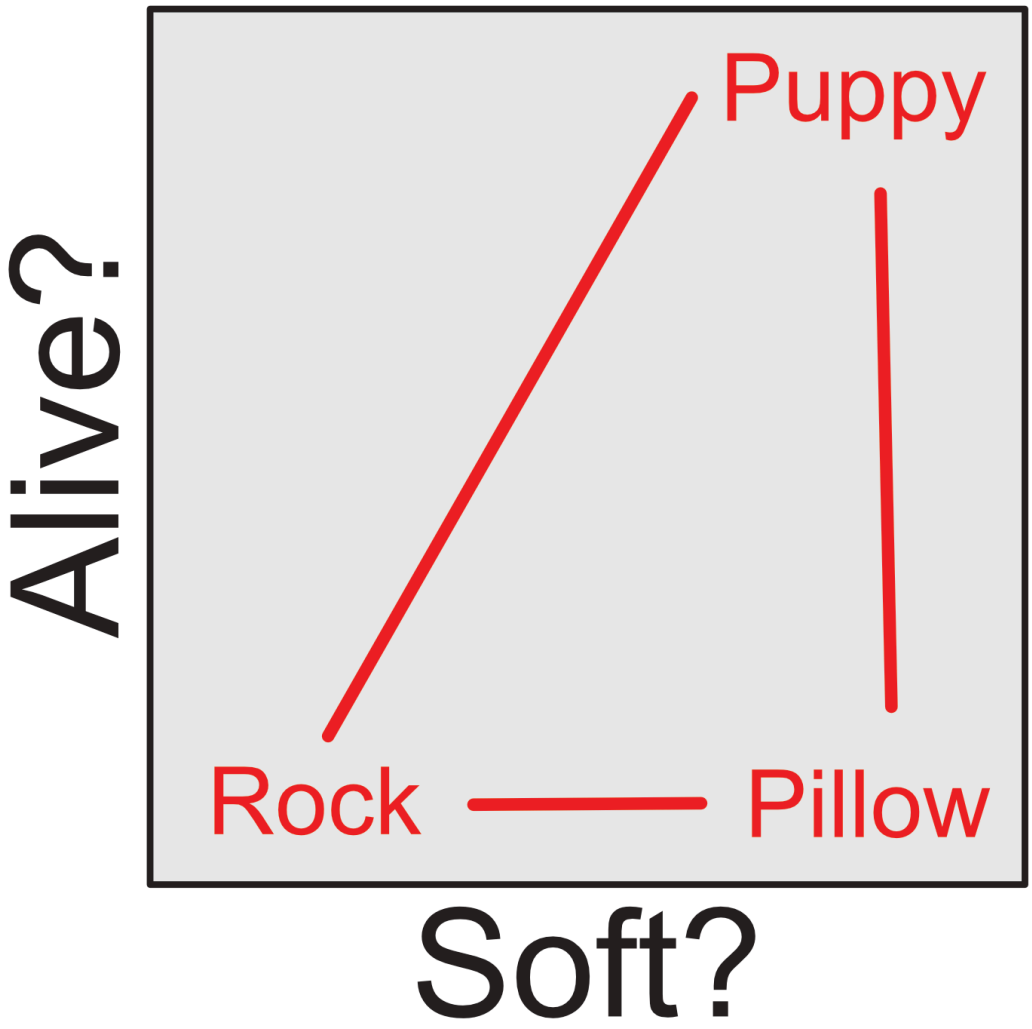
images into different groups? If so, how do we do this? It might depend on what features, or aspects, of the stimuli that we consider.

Feature embedding space

Each image can be described by a number of different variables. These variables could be more abstract, such as is it animate or inanimate? or they can be more low level, such as what color is each object?

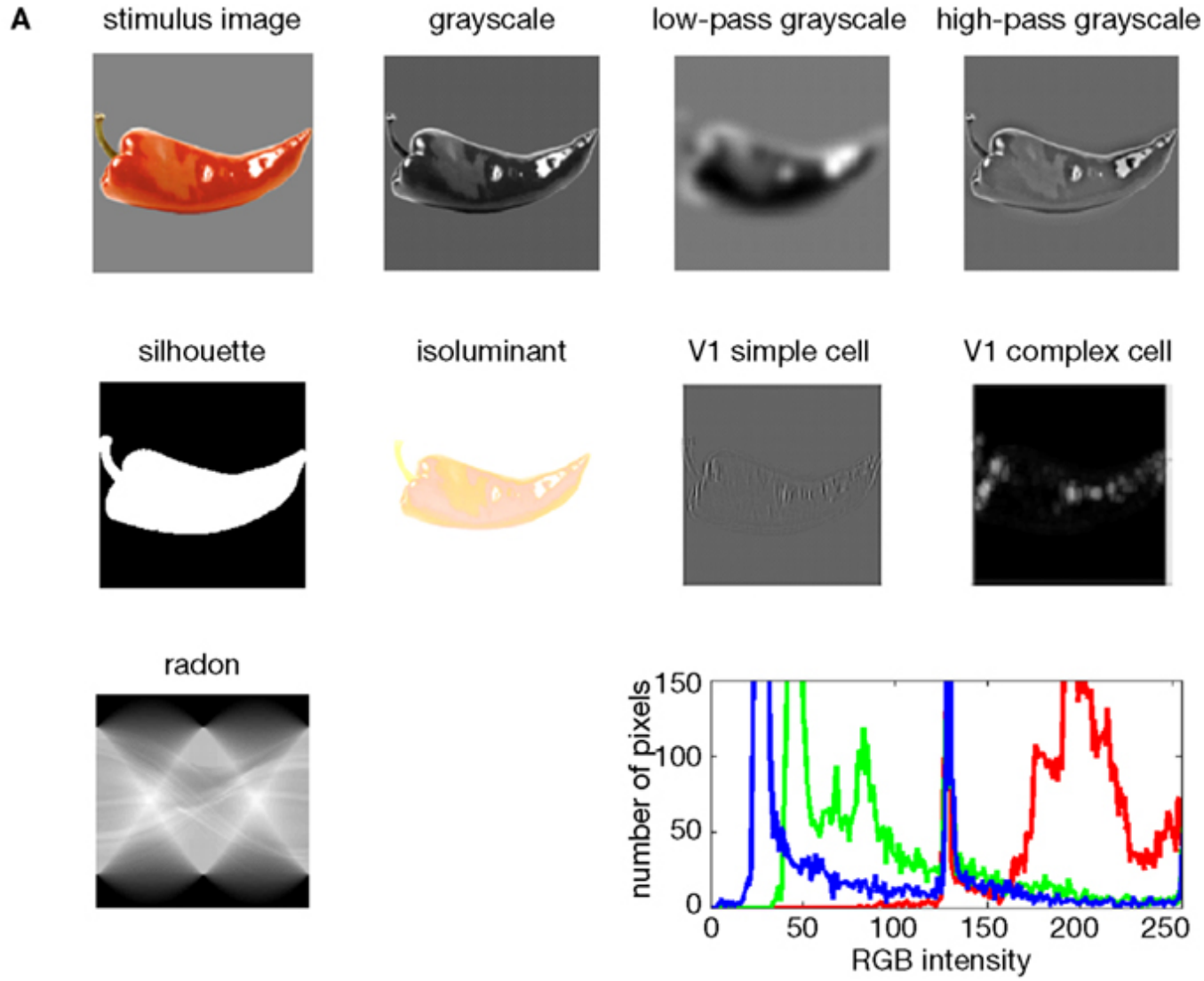
We can group together different variables into a feature space and then describe each image using this space. Think of each feature as an axis in a multidimensional space. For example, consider the two different dimensions of *animacy* and *softness*. Where would a puppy, rock, and pillow be positioned within this two dimensional embedding space?

Skip to main content

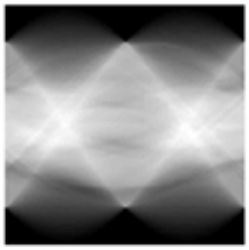


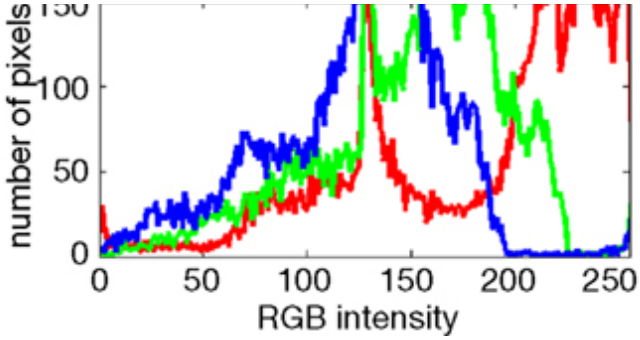
Of course, this is a just a 2-dimensional example, this idea can be extended to n-dimensions. What if were were interested in more low-level visual features?

Skip to main content



Skip to main content



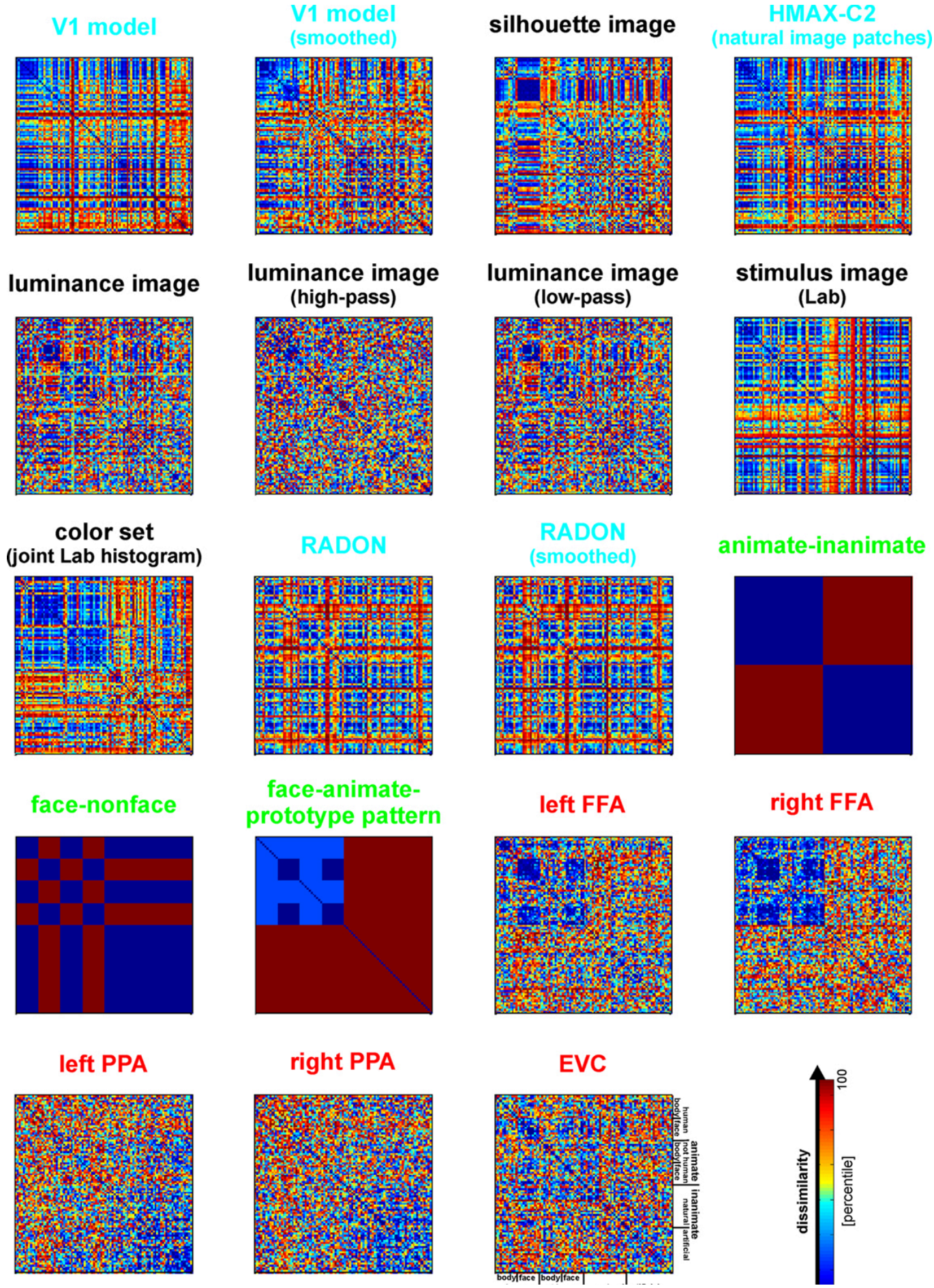


from Kriegeskorte et al., 2008

We can compute the *representational space* of these stimuli by calculating the pairwise distance between each image in this feature space (e.g., euclidean, correlation, cosine, etc). This will yield an image x image matrix. There are many different metrics to calculate distance. For example, the absolute distance is often computed using euclidean distance, but if we don't care about the absolute magnitude in each dimension, the relative distance can be computed using correlation distance. Finally, distance is inversely proportional to similarity and these can be used relatively interchangeability (note that we will probably switch back and forth between describing similarity and distance between stimuli).

For example, if we were interested in regions that process visual information, we might extract different low-level visual features of an image (e.g., edges, luminance, salience, color, etc).

Skip to main content

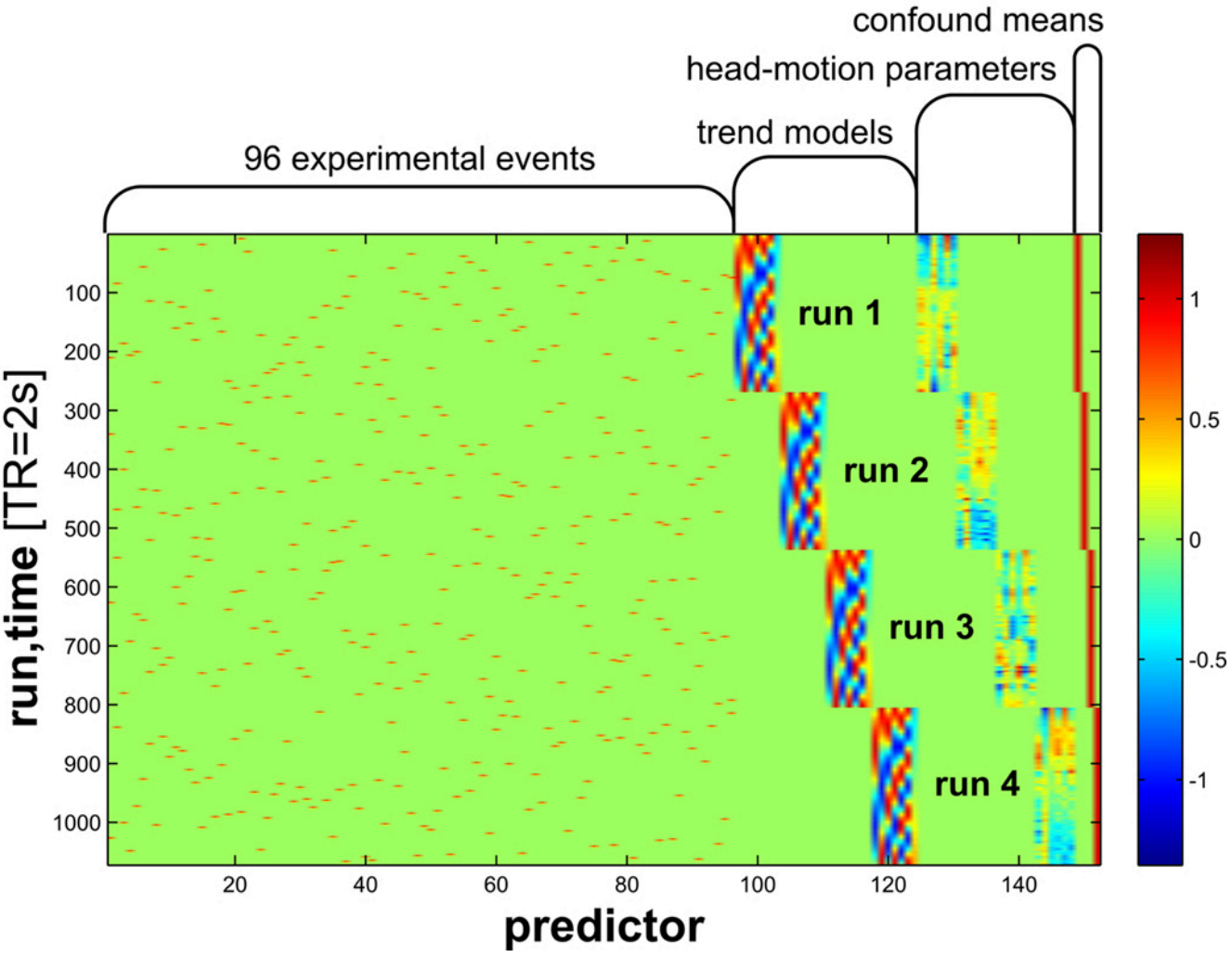


Skip to main content

Brain embedding space

We can also embed each image in brain space. Suppose we were interested in identifying the relationship between images within the visual cortex. Here the embedding space is each voxel's activation within the visual cortex. Voxels are now the axes of the representational space.

To calculate this we need to create a brain map for every single image using a single trial model. We can use a standard first level GLM to estimate a separate beta map for each image while simultaneously removing noise from the signal by including nuisance covariates (e.g., head motion, spikes, discrete cosine transform filters, etc.)

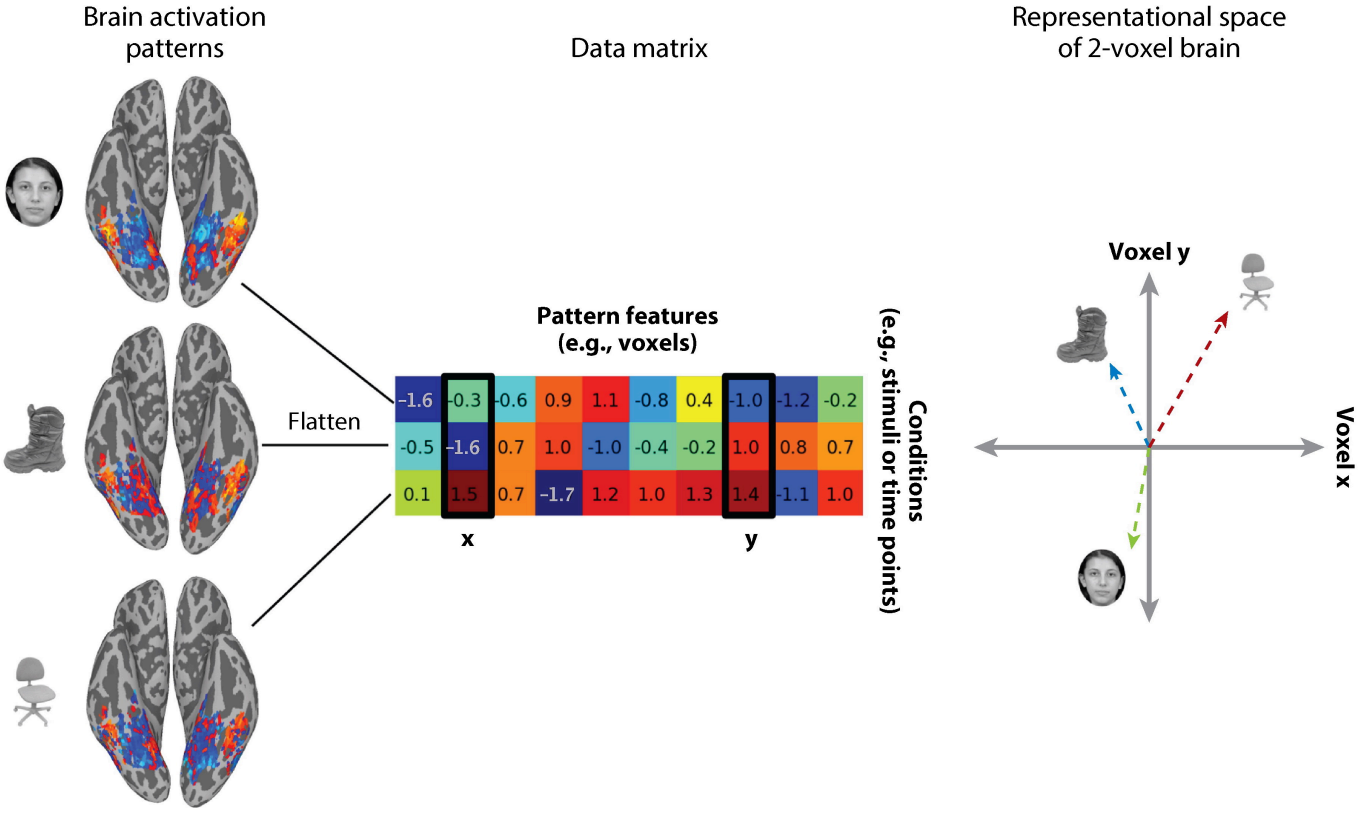


from Kriegeskorte et al., 2008

After we have a single beta map for each image, we can extract patterns of activity for a single ROI to yield an image by ROI voxel matrix. This allows us to calculate the representational space of how this region responds to each image by computing the pairwise similarity of the

Skip to main content

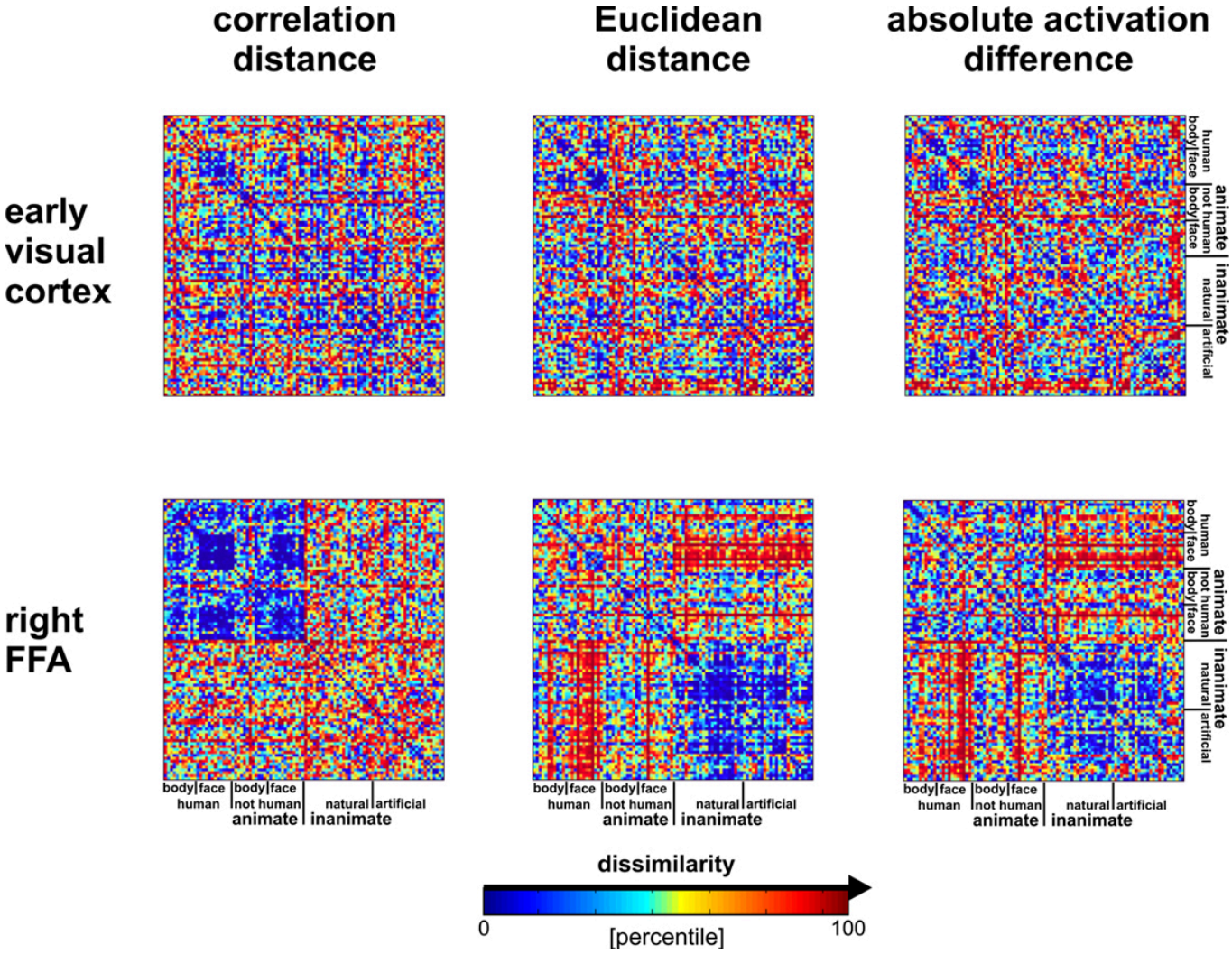
a region becomes an axis in a high dimensional space, and how the region responds to a single picture will be a point in that space. Images that have a similar response in this region will be closer in this high dimensional voxel space.



from Haxby et al., 2014

For fMRI, we are typically not concerned with the magnitude of activation, so we often use a relative pairwise distance metric such as correlation or cosine distance.

Skip to main content



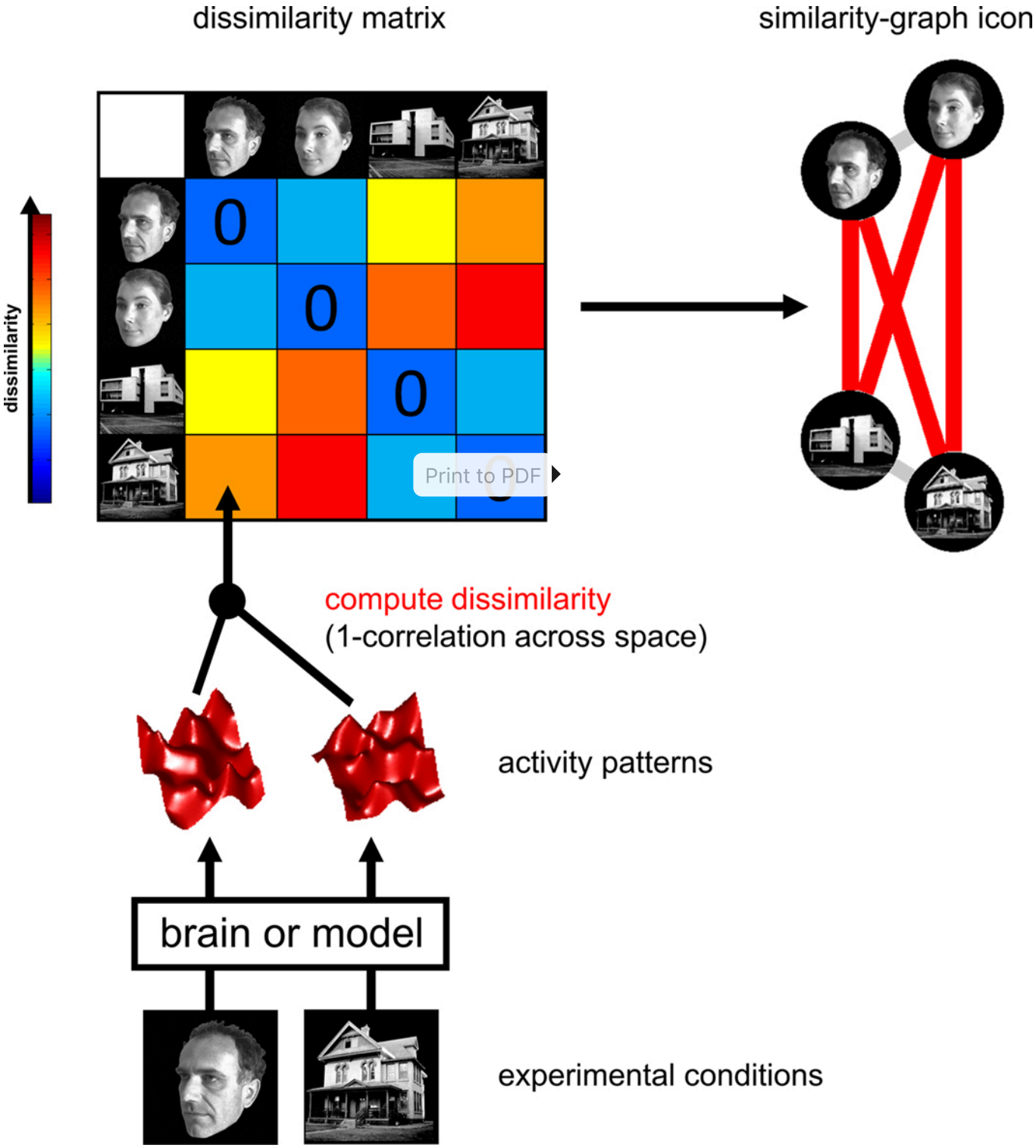
from Kriegeskorte et al., 2008

Mapping stimuli relationships onto brain

Now we can test different hypotheses about how the brain might be processing information about each image by mapping the representational structure of the brain space to different features spaces.

To make an inference about what each region of the brain is computing, we can evaluate if there are any regions in the brain that exhibit a similar structure as the feature representational space. Remember, these matrices are typically symmetrical (unless there is a directed relationship), so you can extract either the upper or lower triangle of these matrices. Then these triangles are flattened or vectorized, which makes it easy to evaluate the similarity between the triangle of the brain and feature matrices. Because these relationships might not necessarily be linear, it is common to use a spearman rank correlation to examine monotonic relationships between different similarity matrices.

Skip to main content

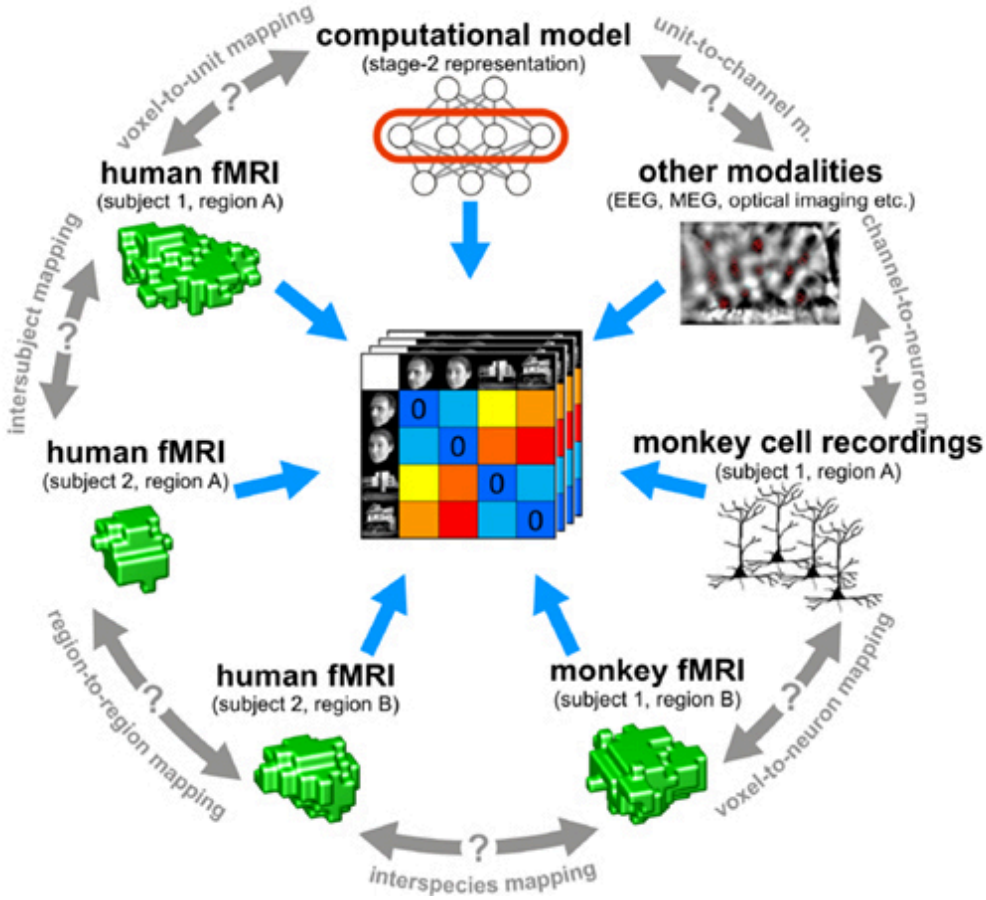


from Kriegeskorte et al., 2008

We can make inferences across subjects, by transforming each correlation value for a given region to a continuous metric using a fisher r-to-z transformation and then computing a one-sample t-test over participants to test if the observed distribution is significantly different from zero. This is typically computed using resampling methods such as a permutation test. If inferences are over participants, then a sign test similar to a one-sample t-test is appropriate. This is a fairly standard practice when all participants viewed the same stimuli and you are

Skip to main content

There are extensions to apply this method to other types of data. For example, in Intersubject-RSA (IS-RSA), one might be interested in whether participants exhibit a particular representational structure over some feature space that maps on to inter-subject differences in brain patterns van Baar et al., 2019. In this particular, extension we cannot use the same type of permutation test to make our inferences (unless we have many different groups of participants). Instead, we might randomly shuffle one of the matrices and repeatedly re-calculate the similarity to produce an empirical distribution of similarity values. It is important to note that this type of permutation violates the exchangeability hypothesis and might yield overly optimistic p-values see Chen et al., 2017. Instead, a more conservative hypothesis testing approach is to use the mantel test, in which we only permute the rows and columns with respect to one another. Personally, I think this approach is too conservative for small sample sizes and it is still an open statistical question about how to best make inferences in this particular type of use.



from Kriegeskorte et al., 2008

One of the reasons why this technique is so powerful is that it allows the possibility of testing different computational hypotheses. Different feature representations might be associated with specific regions involved in various computations. Alternatively, different types of data

Skip to main content

have demonstrated that it is possible to map the function of IT cortex across humans and monkeys using this technique.

Now that we understand the basic steps of RSA, let's apply it to some test data. First, let's load the modules we will use for this tutorial.

```
%matplotlib inline

import os
import glob
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from nltools.data import Brain_Data, Adjacency
from nltools.mask import expand_mask, roi_to_brain
from nltools.stats import fdr, threshold, fisher_r_to_z, one_sample_permutation
from sklearn.metrics import pairwise_distances
from nilearn.plotting import plot_glass_brain, plot_stat_map, view_img_on_surf,
from bids import BIDSLayout, BIDSValidator

data_dir = '../data/localizer'
layout = BIDSLayout(data_dir, derivatives=True)
```

Single Subject Pattern Similarity

Recall that in the Single Subject Model Lab that we ran single subject models for 10 different regressors for the Pinel Localizer task. In this tutorial, we will use our results to learn how to conduct RSA style analyses.

First, let's get a list of all of the subject IDs and load the beta values from each condition for a single subject into a `Brain_Data` object. We will be using the output of running a 1st-level model for each participant where we saved a separate file for each task condition. See this code for reference from the group analysis tutorial.

```
sub = 'S01'

file_list = glob.glob(os.path.join(data_dir, 'derivatives', 'betas', f'{sub}*')
file_list = [x for x in file_list if f'{sub}_betas.nii.gz' not in x]
file_list.sort()
conditions = [os.path.basename(x).split(f'{sub}_beta_')[1].split('.nii.gz')[0]
beta = Brain_Data(file_list)
```

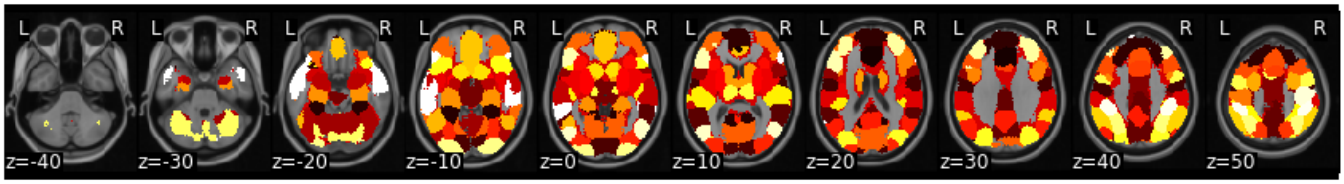
Skip to main content

Next we will compute the pattern similarity across each beta image. We could do this for the whole brain, but it probably makes more sense to look within a single region. Many papers use a searchlight approach in which they examine the pattern similarity within a small sphere centered on each voxel. The advantage of this approach is that it uses the same number of voxels across searchlights and allows one to investigate the spatial topography at a relatively fine-scale. However, this procedure is fairly computationally expensive as it needs to be computed over each voxel and just like univariate analyses, will require stringent correction for multiple tests as we learned about in tutorial 12: Thresholding Group Analyses. Personally, I prefer to use whole-brain parcellations as they provide a nice balance between spatial specificity and computational efficiency. In this tutorial, we will continue to use functional regions of interest from our 50 ROI Neurosynth parcellation. This allows us to cover the entire brain with a relatively course spatial granularity, but requires several orders of magnitude of less computations than using a voxelwise searchlight approach. This means it will run much faster and will require us to use a considerably less stringent statistical threshold to correct for all independent tests. For example, for 50 tests bonferroni correction is $p < 0.001$ (i.e., .05/50). If we ever wanted better spatial granularity we could use increasingly larger parcellations (e.g., 100 or 200).

Let's load our parcellation mask so that we can examine the pattern similarity across these conditions for each ROI.

```
mask = Brain_Data('https://neurovault.org/media/images/8423/k50_2mm.nii.gz')
mask_x = expand_mask(mask)

mask.plot()
```



Ok, now we will want to calculate the pattern similar within each ROI across the 10 conditions.

We will loop over each ROI and extract the pattern data across all conditions and then compute the correlation distance between each condition. This data will now be an `Adjacency` object that we discussed in the Lab 13: Connectivity. We will temporarily store this in a list.

Skip to main content

Notice that for each iteration of the loop we apply the ROI mask to our beta images and then calculate the correlation distance.

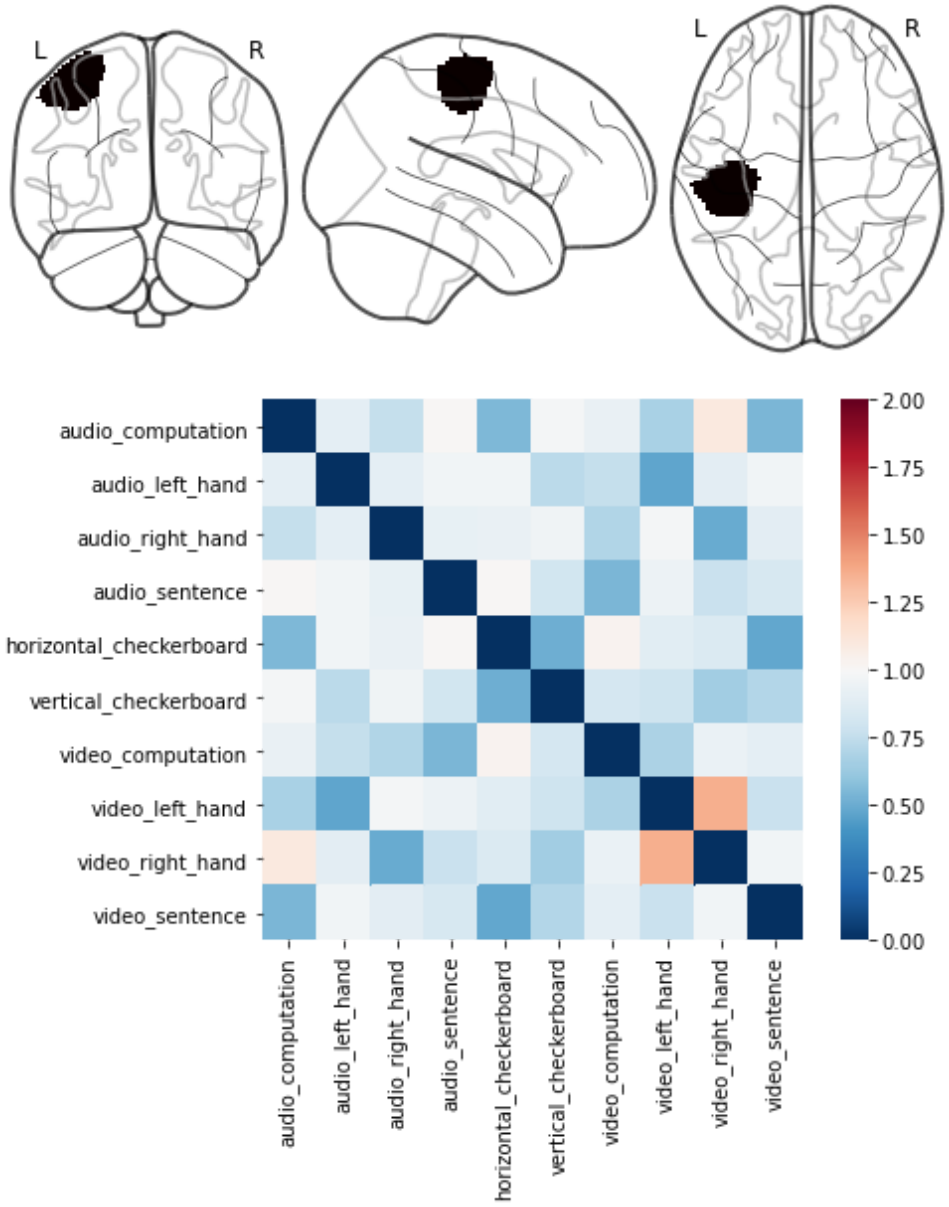
```
out = []
for m in mask_x:
    out.append(beta.apply_mask(m).distance(metric='correlation'))
```

Let's plot an example ROI and it's associated distance matrix.

Here is a left motor parcel. Notice how the distance is small between the motor left auditory and visual and motor right auditory and visual beta images?

```
roi = 26
plot_glass_brain(mask_x[roi].to_nifti())
out[roi].labels = conditions
f2 = out[roi].plot(vmin=0, vmax=2, cmap='RdBu_r')
```

Skip to main content

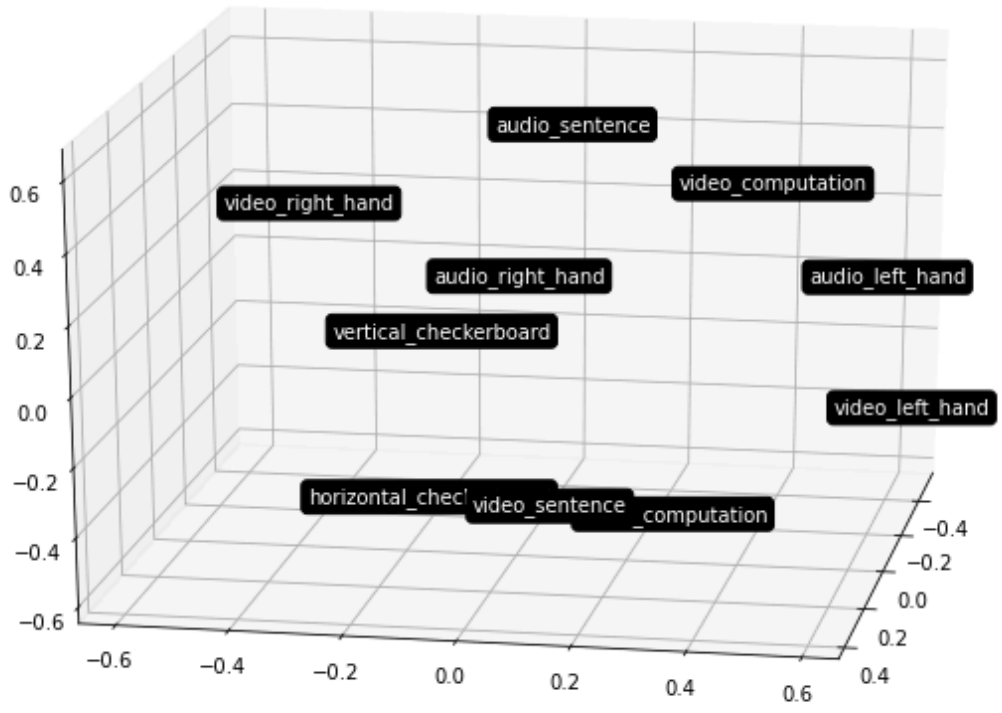


We can also visualize this distance matrix using multidimensional scaling with the `plot_mds()` method. This method projects the images into either a 2D or 3D plane for ease of visualization. There are many other distance based projection methods such as T-SNE or UMAP, we encourage readers to check out the excellent hypertools package that has a great implementation of all of these methods.

Notice how the motor right visual and auditory are near each other, while the motor left visual and auditory are grouped together further away?

```
f = out[roi].plot_mds(n_components=3, view=(20, 10))
```

Skip to main content



It's completely fine to continue to work with distance values, but just to make this slightly more intuitive to understand what is going on we will convert this to similarity. For correlation distance, this entails subtracting each value from 1. This will yield similarity scores in the form of pearson correlations. If you are using unbounded metrics (e.g., euclidean distance), then use the `distance_to_similarity()` Adjacency method.

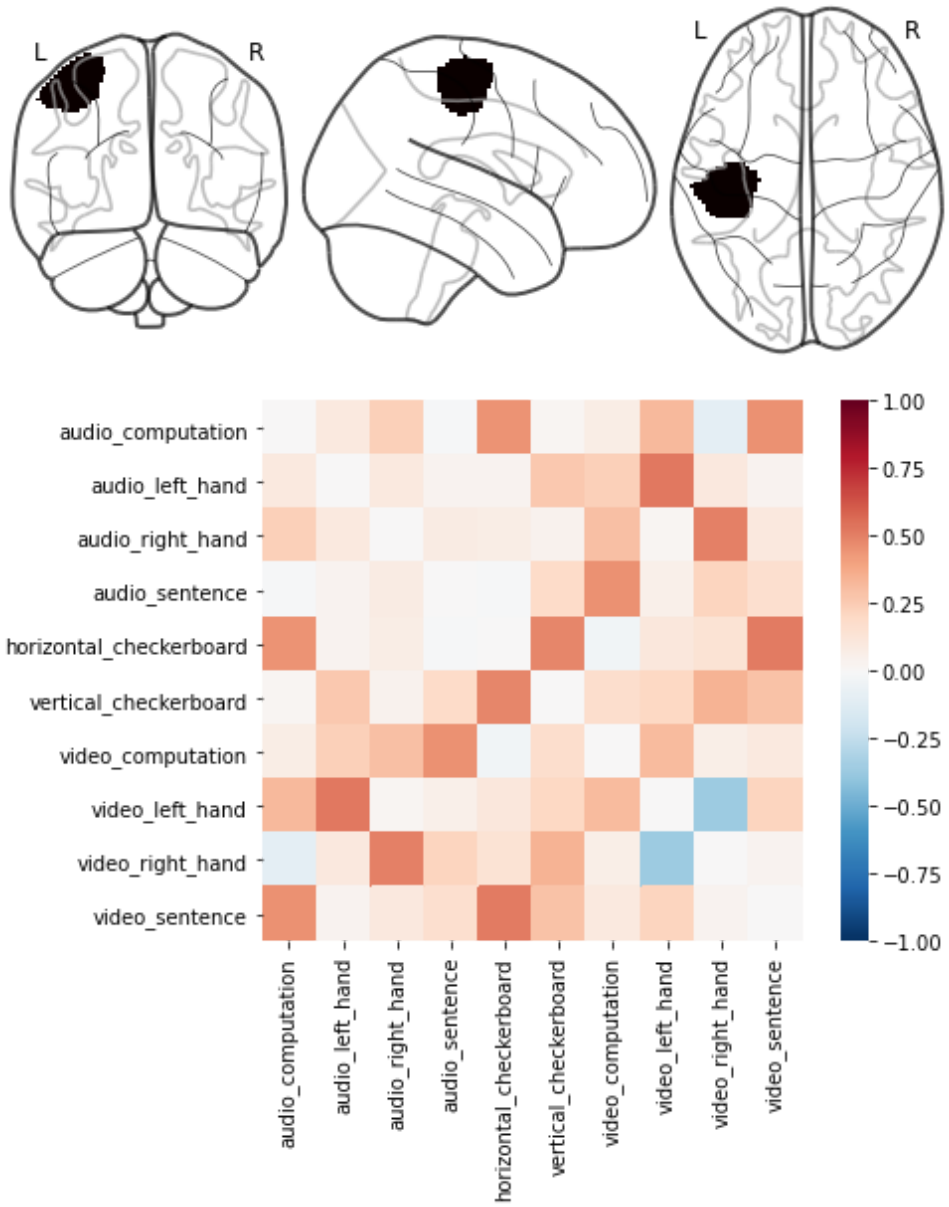
We are also adding conditions as labels to the object, which make the plots easier to read.

```
out_sim = []
for m in out:
    mask_tmp = m.copy()
    mask_tmp = 1 - mask_tmp
    mask_tmp.labels = conditions
    out_sim.append(mask_tmp)
```

Let's look at the heatmap of the similarity matrix to see how more red colors indicate greater similarity between patterns within the left motor cortex across conditions. The `Adjacency` class only stores the lower triangle for similarity and distance matrices. At the moment, it is incorrectly setting the diagonal to zero when plotting the similarity matrix. The diagonal should be a vector of ones.

Skip to main content

```
roi = 26
plot_glass_brain(mask_x[roi].to_nifti())
f = out_sim[roi].plot(vmin=-1, vmax=1, cmap='RdBu_r')
```



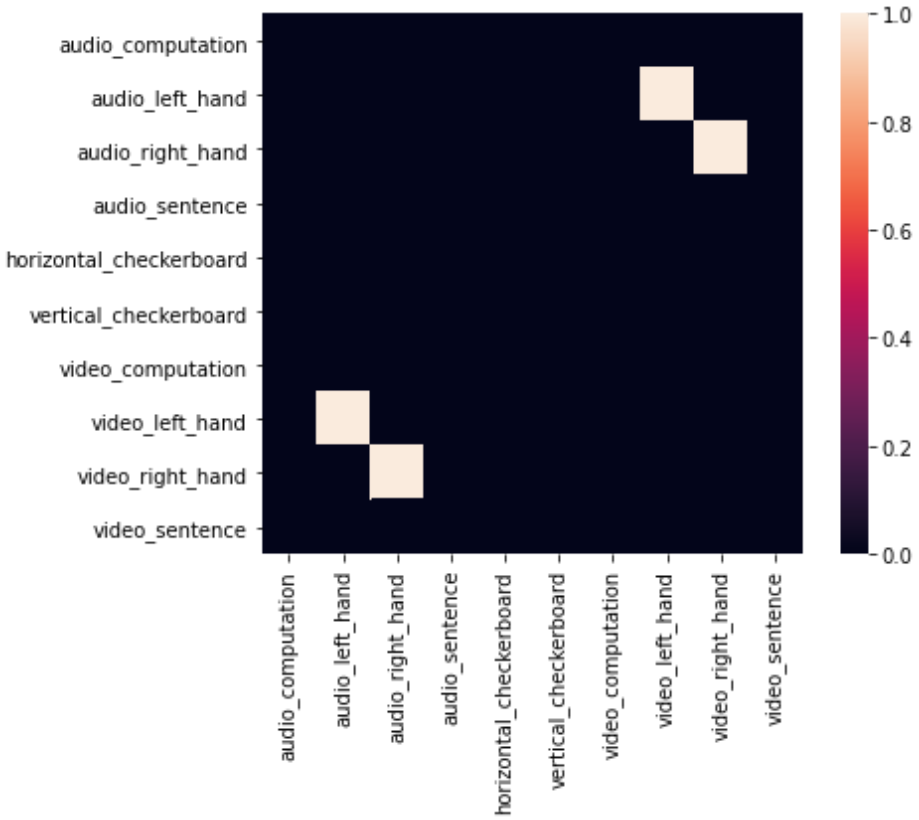
Testing a Representation Hypothesis

Ok, now that we have a sense of what the similarity of patterns look like in left motor cortex, let's create an adjacency matrix indicating a specific relationship between left hand finger tapping across the auditory and visual conditions. This type of adjacency matrix is one way in which we can test a specific hypotheses about the representational structure of the data across all images.

Skip to main content

As you can see this only includes edges for the motor-left auditory and motor-left visual conditions.

```
motor = np.zeros((len(conditions),len(conditions)))
motor[np.diag_indices(len(conditions))] = 1
motor[1,7] = 1
motor[7,1] = 1
motor[2,8] = 1
motor[8,2] = 1
motor = Adjacency(motor, matrix_type='distance', labels=conditions)
motor.plot()
```



Now let's search over all ROIs to see if any match this representational structure of left motor-cortex using the `similarity()` method from the `Adjacency` class. This function uses spearman rank correlations by default. This is probably a good idea as we are most interested in monotonic relationships between the two similarity matrices.

The `similarity()` method also computes permutations within columns and rows by default. To speed up the analysis, we will set the number of permutations to zero (i.e., `n_permute=0`).

```
motor_sim_r = []
for m in out_sim:
```

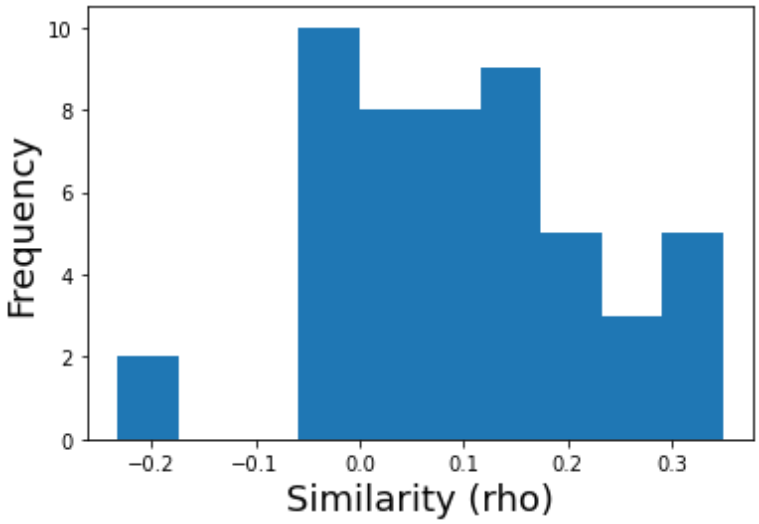
Skip to main content

```
    s = m.similarity(motor, metric='spearman', n_permute=0)
    motor_sim_r.append(s['correlation'])
```

This will return a vector of similarity scores for each ROI, we can plot the distribution of these 50 ρ values.

```
plt.hist(motor_sim_r)
plt.xlabel('Similarity (rho)', fontsize=18)
plt.ylabel('Frequency', fontsize=18)
```

```
Text(0, 0.5, 'Frequency')
```

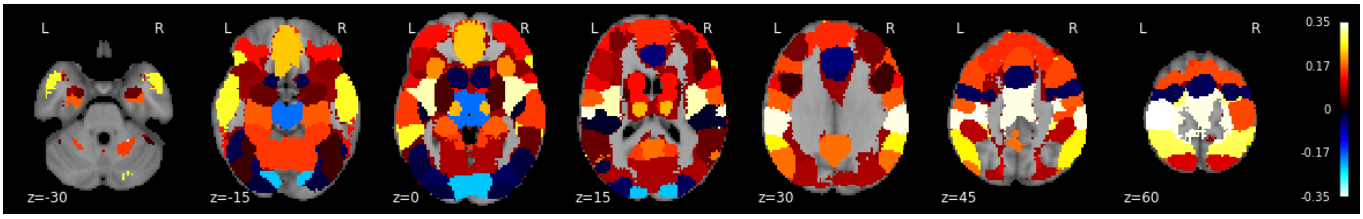


We can also plot these RSA values back onto the brain to see the spatial distribution.

```
rsa_motor = roi_to_brain(motor_sim_r, mask_x)

plot_stat_map(rsa_motor.to_nifti(), draw_cross=False, display_mode='z', black_b
```

```
<nilearn.plotting.displays.ZSlicer at 0x7fc04c15bd50>
```



Notice how left motor cortex is among the ROIs with the highest similarity value? Unfortunately, we can only plot the similarity values and can't threshold them yet because we

Skip to main content

We could calculate p-values using a permutation test, but this would require us to repeatedly recalculate the similarity between the two matrices and would take a long time (i.e., 5,000 correlations X 50 ROIS). Plus, the inference we want to make isn't really at the single-subject level, but across participants.

Let's now run this same analysis across all participants and run a one-sample t-test across each ROI.

RSA Group Inference

Here we calculate the RSA for each ROI for every participant. This will take a little bit of time to run (30 participants X 50 ROIs).

```
sub_list = layout.get_subjects(scope='derivatives')

all_sub_similarity = {}; all_sub_motor_rsa = {};
for sub in sub_list:
    file_list = glob.glob(os.path.join(data_dir, 'derivatives','fmriprep', f'su
    file_list = [x for x in file_list if 'betas' not in x]
    file_list.sort()
    conditions = [os.path.basename(x).split(f'sub-{sub}_')[1].split('_denoised'
    beta = Brain_Data(file_list)

    sub_pattern = []; motor_sim_r = [];
    for m in mask_x:
        sub_pattern_similarity = 1 - beta.apply_mask(m).distance(metric='correl
        sub_pattern_similarity.labels = conditions
        s = sub_pattern_similarity.similarity(motor, metric='spearman', n_permu
        sub_pattern.append(sub_pattern_similarity)
        motor_sim_r.append(s['correlation'])

    all_sub_similarity[sub] = sub_pattern
    all_sub_motor_rsa[sub] = motor_sim_r
all_sub_motor_rsa = pd.DataFrame(all_sub_motor_rsa).T
```

Now let's calculate a one sample t-test on each ROI, to see which ROI is consistently different from zero across our sample of participants. Because these are r-values, we will first perform a fisher r to z transformation. We will use a non-parametric permutation sign test to perform our null hypothesis test. This will take a minute to run as we will be calculating 5000 permutations for each of 50 ROIs (though these permutations are parallelized across cores).

```
rsa_stats = []
```

Skip to main content

```
    rsa_stats.append(one_sample_permutation(fisher_r_to_z(all_sub_motor_rsa[i])
```

We can plot a thresholded map using fdr correction as the threshold

```
fdr_p = fdr(np.array([x['p'] for x in rsa_stats]), q=0.05)
print(fdr_p)

rsa_motor_r = Brain_Data([x*y['mean'] for x,y in zip(mask_x, rsa_stats)]).sum()
rsa_motor_p = Brain_Data([x*y['p'] for x,y in zip(mask_x, rsa_stats)]).sum()

thresholded = threshold(rsa_motor_r, rsa_motor_p, thr=fdr_p)

plot_glass_brain(thresholded.to_nifti(), cmap='coolwarm')
```

```
-1
```

```
<nilearn.plotting.displays.OrthoProjector at 0x7fbf3929acd0>
```

Looks like nothing survives FDR. Let's try a more liberal uncorrected threshold.

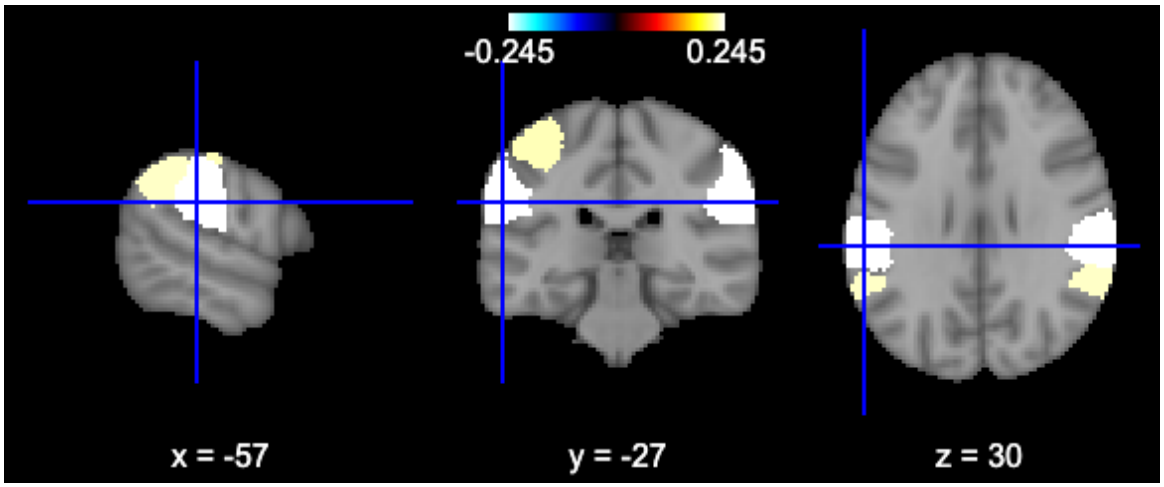
```
thresholded = threshold(rsa_motor_r, rsa_motor_p, thr=0.01)

plot_glass_brain(thresholded.to_nifti(), cmap='coolwarm')
```

```
<nilearn.plotting.displays.OrthoProjector at 0x7fbf2cd13c90>
```

Skip to main content


```
view_img(thresholded.to_nifti())
```



This looks a little better and makes it clear that the ROI that has the highest similarity with our model specifying the representational structure of motor cortex is precisely the left motor cortex. Though this only shows up using a liberal uncorrected threshold, remember that we are only using about 9 subjects for this demo.

Though this was a very simple model using a very simple dataset, hopefully this example has illustrated how straightforward it is to run RSA style analyses using any type of data. Most people use this method to examine representations of much more complicated feature spaces.

It is also possible to combine different hypotheses or models to decompose a brain representational similarity matrix. This method typically uses regression rather than correlations (see Parkinson et al., 2017)

We have recently used this technique in our own work to map similarity in brain space to

Skip to main content

seeing an example of intersubject RSA (IS-RSA), check out the paper, and accompanying python code.

Previous
Multivariate Prediction

Next
Introduction to Parcellations