Graph-based Feature Extraction for Functional Connectivity Analysis in rs-fMRI of Individuals with ASD and Controls

BSc Thesis



J.F. Groenenberg
G.G. Myburg

CONTENTS

Abstract iv

Preface v

1 Introduction 1
- 1.1 State of the Art . . . 2
- 1.2 Synopsis . . . 3

2 Program of Requirements 5

3 Methodology 7
- 3.1 Data extraction . . . 7
  - 3.1.1 Filtering . . . 7
  - 3.1.2 Pipeline . . . 8
  - 3.1.3 Brain atlas . . . 8
- 3.2 Node detection . . . 8
  - 3.2.1 Ward clustering . . . 8
  - 3.2.2 Spectral clustering . . . 9
  - 3.2.3 Community detection . . . 9
  - 3.2.4 Independent Component Analysis . . . 9
- 3.3 Edge inference . . . 11
- 3.4 Sample Covariance Estimation . . . 11
  - 3.4.1 Direct Empirical Covariance . . . 11
  - 3.4.2 Sliding-Window Averaged Covariance . . . 12
  - 3.4.3 Ledoit-Wolf Shrinkage Estimator . . . 12
  - 3.4.4 Graphical Lasso (Sparse Inverse Covariance Estimation) . . . 12
  - 3.4.5 Time-Varying Graphical Lasso . . . 12
  - 3.4.6 VAR(k) - Covariance of Residuals . . . 13
  - 3.4.7 NVAR - Residuals + Graphical Lasso . . . 13
- 3.5 Statistical Graph Inference Methods . . . 13
  - 3.5.1 Sample Covariance . . . 13
  - 3.5.2 Pearson's Correlation . . . 13
  - 3.5.3 Partial Correlation . . . 14
  - 3.5.4 Mutual Information . . . 14
  - 3.5.5 Granger Causality . . . 14
- 3.6 Thresholding . . . 15
- 3.7 GSP Graph Inference Methods . . . 15
  - 3.7.1 Spectral templates or rSpecT . . . 15
  - 3.7.2 rLogSpecT . . . 17
  - 3.7.3 Causal Graph Process . . . 17

3.8 Graph features . . . 18
3.9 Testing process . . . 19
3.9.1 Autocorrelation . . . 19
3.9.2 Testing for Ergodicity . . . 19
3.9.3 Graph validation . . . 20
3.10 NASDA GUI design plan . . . 20

4 Results **22**
4.1 Comparison of pre-processing settings . . . 22
4.1.1 Effect of Filtering . . . 22
4.1.2 Effect of Independent Component Analysis . . . 23
4.2 Tuning/testing of Spectral Template methods . . . 23
4.2.1 rSpecT . . . 23
4.2.2 rLogSpecT . . . 25
4.3 Comparison of Graph Inference methods . . . 25
4.3.1 Covariance estimate . . . 26
4.3.2 Inference method . . . 26
4.3.3 Impact of features . . . 27
4.4 Statistical test results . . . 27
4.4.1 Validating statistical inference methods . . . 28
4.5 Final GUI . . . 28

5 Conclusion **30**

6 Discussion **31**

Appendices **33**

A Regions of Interest from the AAL brain atlas **34**

B Code & Data **36**
B.1 Python code . . . 36

C Some extra properties **37**
C.1 GSP Properties . . . 37

D AAL graph extraction visualization **39**
D.1 Pearson's Correlation . . . 39
D.2 Mutual Information . . . 41
D.3 MRF process example - Direct neighbor graph estimate . . . 42

E Mathematical models **45**
E.1 Laplacian Operation & Context . . . 45
E.2 Linear model . . . 46
E.3 Diffusion Process Causal Graph Process . . . 46
E.4 Markov Random Field . . . 46
E.5 Stationary Graph Process . . . 46
E.6 Autoregressive (AR) process . . . 46
E.7 Vector Autoregressive (VAR) process . . . 46
E.8 Nonlinear Vector Autoregressive (NVAR) process . . . 47

E.9 White noise . . . 47

F Designed Features **48**
F.1 Statistical Features . . . 48
F.2 Graph Features . . . 48
F.2.1 Node specific features . . . 48
F.2.2 Node to node . . . 50
F.2.3 Global features . . . 50
F.2.4 Feature selection . . . 53

G recovery theorems and results spectral templates **59**
G.1 recovery guarantees of rSpecT . . . 59
G.2 recovery and feasibility guarantees of rLogSpecT . . . 59
G.3 results . . . 60

H Derivation for Causal Graph Process **64**

I Statistical testing **66**
I.1 Autocorrelation plots . . . 66
I.2 Ergodicity plots . . . 66

J Brain feature plots **70**
J.1 Visualisation of the features . . . 70
J.1.1 Brain Connectome . . . 70
J.1.2 Glass Brain . . . 70

K CPAC pipeline **75**

L Classification results **76**
L.1 Classification control metrics . . . 76
L.1.1 Raw metrics . . . 76
L.1.2 Accuracy . . . 76
L.1.3 AUC . . . 76
L.2 Class 0 or 1 Recall . . . 76
L.3 MSE . . . 76
L.4 Classification tables . . . 77
L.4.1 ICA . . . 77
L.4.2 Graph Inference methods . . . 79
L.4.3 Effect of thresholding . . . 79

M ICA components **82**

N Graphical User Interface **88**
N.1 Complexity . . . 89

ABSTRACT

This research is part of a broader study on Autism Spectrum Disorder (ASD) detection, which consists of three main components: feature design, classification, and feature selection. The primary objective of this study is to investigate whether different graph inference methods and graph-based features can contribute to finding new biomarkers in the brains of individuals with ASD. To easily interpret and visualize the results, a graphical user interface (GUI) was developed.

The feature design phase includes both classical statistical features and innovative graph-based features, derived through a variety of statistical and Graph Signal Processing (GSP) graph inference methods. The top three performing methods were: sample covariance with graphical lasso (59.3% accuracy, 64.2% AUC), rSpecT-adjacency (58.0% accuracy, 62.5% AUC), and rSpecT-laplacian (58.8% accuracy, 61.5% AUC). the connectivity patterns visualized using the developed tools revealed distinct differences between methods, offering promising insights for neuroscientific interpretation and potential biomarker discovery.

PREFACE

Autism Spectrum Disorder (ASD) and other forms of neurodivergence are prevalent in many communities, including TU Delft. Individuals with such conditions may benefit from tailored approaches in both education and healthcare. Currently, diagnoses of ASD are made by psychiatrists based on a range of behavioral criteria. However, the underlying biological causes of ASD remain largely unknown. Investigating the brain activity of individuals with ASD may help identify the neurobiological underpinnings and key regions or biomarkers associated with the condition. Such findings could ultimately contribute to improved diagnostic methods and more effective, personalized interventions.

We would like to express our sincere gratitude to Prof. Dr. ir. Geert Leus and ir. Ruben Wijnands for their guidance and supervision throughout this research project.

Delft, June 2025

1

INTRODUCTION

ASD is a complex neurodevelopmental condition characterized by difficulties in social interaction and communication, along with restricted or repetitive patterns of behavior and interests [1]. Although its behavioral manifestations are well documented, the underlying neurological mechanisms of ASD remain poorly understood [2]. A growing body of research suggests that atypical patterns of functional connectivity in the brain may contribute to ASD [3], [4], making the identification of reliable neural biomarkers a crucial area of study [5]. Non-invasive neuroimaging techniques, such as functional Magnetic Resonance Imaging (fMRI), hold significant promise for advancing our understanding of ASD.

This study utilizes data obtained from the Autism Brain Imaging Data Exchange (ABIDE) initiative [6], which is a dataset consisting of multi-site rs-fMRI data. The rs part stands for resting state, this comes from the fact that the individual whose brain is recorded is in a resting state, which is related to low frequency-type modes. To analyze brain activity, researchers often use a brain atlas, which divides the brain into a predefined set of regions based on anatomical or functional criteria. These regions are commonly referred to as Regions of Interest (ROIs). Once an atlas is defined, functional connectivity can be computed by measuring statistical dependencies—such as correlations—between the time series of different ROIs. This reveals patterns of synchronized activity across different brain areas, providing insight into the brain's functional organization. Functional fMRI is a type of MRI imaging technique that looks at relatively long time spans ($\geq$ 5 min) in order to more accurately model temporal behaviour. The fMRI data models the Blood-Oxygen-Level-Dependent (BOLD) signal, which serves as an indirect measure of underlying neuronal activity [7]. Because the BOLD signal reflects slow hemodynamic changes, the time between each sample (TR) is typically chosen to balance temporal resolution with adequate sampling of these slow fluctuations.

The dataset consists of 1112 subjects measured at different sites, of which 539 from people with ASD and 573 without. Only 884/871 subjects were used due to the fact that some subjects did not have the necessary phenotypic data available. The number of time samples differed between sites and was in the range of 150-300. The (TR) was approximately

2 seconds. This gives a total measurement time range of 5-10 minutes.
Graph Signal Processing (GSP) is an emerging branch of signal processing that exploits the underlying network structure of complex systems, such as the human brain. Measured signals are associated with nodes in a graph connected by weighted edges. Under the assumption that the signals are related to the topology of the graph where they are supported, the goal of graph signal processing (GSP) is to develop algorithms that fruitfully leverage this relational structure, and can make inferences about these relationships when they are only partially observed [8], [9]. Most GSP efforts to date assume that the underlying network is known, and then analyze how the graph's algebraic and spectral characteristics impact the properties of the graph signals of interest. This is however not the case with the human brain. Thus we focus instead on an even newer branch of GSP, methods that infer a graph structure based on the measured data.

In this thesis, we present different data-driven approaches to inferring the brain's underlying graph structure using only resting-state fMRI (rs-fMRI) signals from specific regions of interest (ROIs), rather than relying on a predefined network model. Several graph inference methods are applied to the data, and the resulting graph based features are input into a simple machine learning algorithm to detect ASD. This process allows us to evaluate the validity and effectiveness of different network models in capturing ASD-related neural patterns, and to determine different biomarkers for people with ASD. Additionally a GUI is developed that enables meaningful visualization of the network models in the brain.

1.1. STATE OF THE ART

Graph signal processing and the detection of ASD based on rs-fMRI is an active field of research. GSP is a relatively new branch of research, e.g. of the last 10 years. Yet the application of GSP based network inference methods on rs-fMRI data of people with ASD has not been extensively explored yet. In this section we list previous research using statistical and GSP based graph inference methods that were applied on the ABIDE dataset.

Several studies have applied traditional statistical connectivity methods to classify Autism Spectrum Disorder (ASD) using resting-state fMRI (rs-fMRI). A common approach is to compute correlation-based functional connectivity matrices. For example, [10] and [2] employed Pearson, partial, and tangent correlation metrics, with the latter incorporating these features into a neural network classifier. In a multi-site analysis [11], Fisher-transformed Pearson correlation matrices were used as input features. The reported results on the ABIDE dataset for these studies are as follows: – [10] achieved an accuracy of 69.43% and a specificity of 73.61% using partial correlation, the BASC444 atlas, and a radial basis function (RBF) kernel SVM; – [2] reported a maximum accuracy of 60.0%, with sensitivity and specificity of 62.0% and 58.0%, respectively; – [11] achieved accuracies around 75%, with corresponding sensitivities near 70%.
[12] uses pearson correlation with thresholding from a 264 region parcellation atlas to design graph-based features. They use the eigenvectors of the laplacian matrix derived from the obtained correlation matrix for classification. Other graph-based features such as centralities, clustering and assortativity are computed from the correlation matrix. Using LDA to classify ASD on the ABIDE dataset and the graph-based features, an accu-

racy of 77.7% is obtained. The paper also clearly illustrates the importance and effect of thresholding when using graph-based features.

In [13], a comparison analysis in ASD classification on the ABIDE-II dataset between traditional correlation and an introduced large-scale Augmented Granger Causality (lsAGC) algorithm was described. Granger-Causality was used together with PCA to obtain the results. They used the Granger-Causality matrix directly as classification features. The lsAGC method had an AUC of 0.940 ± 0.074 compared to the correlation (0.890 ± 0.100, -5.32%), local models (0.905 ± 0.108, -3.72%), and antisymmetric correlation (0.882 ± 0.102, -6.17%).

One of the earliest approaches to ASD classification using GSP was [14]. In this paper they represent populations as a sparse graph where its vertices are associated with fMRI based feature vectors and the edges encode phenotypic information. This structure was used to train a Graph Convolutional Network (GCN) model. This has a clear impact on the quality of the predictions, leading to 69.5% accuracy for ABIDE. Another paper building on this concept is [15]. However, these studies do not rely solely on fMRI data, rather, they incorporate phenotypic information as well.

In [16] the frequency behavior of the signals was analysed by computing their Graph Fourier Transform (GFT). The edges of the graph are defined by connecting close nodes in terms of the inverse of the euclidian distance in the brain. The time-varying GFT coefficients are then merged into a connectivity matrix, from which they compute discriminative graph frequency patterns, by using an extension of the Fukunaga-Koontz transform. They then exploit these features to separate ASD from neurotypical subjects, using an interpretable classification scheme such as the decision tree. This resulted in an accuracy of 73.5%.
Another approach in [17] leverages a structural graph built using tactography of the white matter of the brain. The GFT of these signals is then computed using the structural graph as a support and are used as features in a support vector classification algorithm. They reached accuracies of 60% on the ABIDE dataset.
Both studies rely on a pre-defined structural graph rather than inferring the graph directly from the fMRI data, highlighting a significant gap in current research. To our knowledge, very little research has been done on applying fMRI data-driven GSP-based graph inference methods on the ABIDE dataset.

1.2. SYNOPSIS

In this report we focus on a few statistical based and GSP based graph inference methods. More specifically, we implement the following methods:

1. A Pearson correlation based graph
2. A partial correlation based graph using different sample covariance techniques.
3. A Mutual Information (MI) based graph
4. A Granger Causality (GC) based graph

5. A Graphical Lasso based graph
6. A diffusion based model based on spectral templates
7. A Causal Graphical Process (CGP) based model for graph inference.

2

Program of Requirements

This section will state all the set requirements for the feature design based on the requirements from the BAP proposal and the self-set requirements for the group.

Main goals:

1. To identify and implement graph inference methods that capture distinct yet complementary characteristics of functional connectivity, enabling both cross-validation between methods and improved classification accuracy of ASD versus control groups.
2. To develop a graphical user interface that enables meaningful visualization, exploration, and interpretation of extracted graph-based connectivity features and inference methods, ensuring their interpretability and relevance for discovering potential neural biomarkers and supporting both scientific discovery and clinical application (see Appendix N).

Functional requirements:

- Extract a minimum of 5 unique statistical features for reference.
- Extract a minimum of 5 unique graph-based node specific features.
- Extract a minimum of 5 unique global graph features.
- Use at least 2 different methods for node detection.
- Design and implement a minimum of 2 different covariance estimation techniques.
- Design and implement a minimum of 5 different graph inference methods for edge detection.
- Compare the influence and significance of the graph-based features with the statistical features.

- Compare the influence and significance of the designed inference methods.
- Test the reliability of the designed features.

Non-functional requirements:

- The feature extraction algorithms shall be designed to generalize across different datasets and pre-processing pipelines.
- Provide outputs suitable for integration into a graphical user interface and the other subgroups (e.g. classification and feature selection).
- The feature name, relative ROIs used for computation, and subject data should be stored along with the output.
- Aim for computational efficiency to ensure reasonable processing time on standard hardware.

The group as a whole has to deliver a product, which in this case is the NASDA (Neuro-ASD Analysis) Graphical User Interface (GUI).

Final product specifications:

- The GUI should take a dataframe with graph-based features as input for visualization.
- The GUI should plot the found brain connectivity onto an existing brainmap (AAL)
- The GUI should be able to plot both edge features (e.g. connectivity) and node features (e.g. region intensity)
- The GUI should display information about the feature being plotted, the brainmap used and the subject data (e.g. the plot function parameters).
- In the GUI, the user should have the possibility to load their own dataframes and brainmaps.
- In the GUI, the user should be able to dynamically swap between inference models used to obtain the connectivity used for the plot
- In the GUI, the user should be able to choose what feature to base the plot on.

3

METHODOLOGY

In this section, the methodology for the design of the aforementioned graph inference methods are discussed.

3.1. DATA EXTRACTION

First off, the data will be extracted using a pre-processing pipeline (see section 3.1.2). This data extraction has many parameters and also includes site and phenotypical data per subject. As the site has an impact on the measurements and phenotypical data on the subject, this has to be taken into account and stored along with the data.
The ABIDE dataset can be fetched using the python function from the nilearn library, *fetch_abide_pcp*. Here, the pre-processing steps can be selected for fetching the data. The most important parameters for pre-processing are the pipeline strategy (cpac,etc.), the derivatives of the data (rois-aal, etc.), and the filtering options (global signal regression, band-pass filtering). From other works, including this paper [18], the pre-processing seems to have a significant impact on classification results obtained using fMRI data. This is why the impact of these parameters should be tested for the relative dataset before obtaining trustworthy results.

3.1.1. FILTERING

Common sources of noise in fMRI data are head motion, scanner-related fluctuations, and physiological processes such as respiration and cardiac activity. Although pre-processing steps, such as applying a band-pass filter (typically 0.01–0.1 Hz), can attenuate some of these noise components, they do not fully eliminate them. Furthermore, certain pre-processing strategies, including band-pass filtering and global signal regression, may inadvertently remove meaningful neuronal signals along with noise [19], [20], thereby introducing potential biases into the data.
Therefore, the data will initially be extracted using band-pass filtering without global signal regression. However, the impact of these pre-processing choices will be evaluated in 4, and the final settings for graph analysis will be determined based on those findings.

3.1.2. PIPELINE

The choice of pre-processing pipeline depends on the overall analysis strategy. For this study, the CPAC pipeline [21], an open-source framework, was employed. The pipeline can be found in appendix K. The reason for choosing the CPAC pipeline, is that it was already implemented for the ABIDE dataset, as the default, and is widely used across many different studies [22].

3.1.3. BRAIN ATLAS

The brain atlas chosen for this project is the Automated Anatomical Labeling atlas (AAL), which a widely used brain atlas from [23] consisting of 116 ROIs. The choice of the AAL atlas in this study was motivated by its relatively modest number of ROIs compared to other atlases, which improves computational efficiency.

3.2. NODE DETECTION

To estimate node locations based on functional MRI data, one can perform node detection using an existing brain atlas or by simply assuming that each pixel or voxel represents a node.
Other possible different node detection schemes include:

- **Independent Component Analysis** (ICA) Obtain spatially independent components [24].
- **Ward clustering** A parcellation algorithm that clusters based on similarity of nodes & minimizes variance [25].
- **Spectral clustering** Use GFT to seperate the data in terms of frequency components. Then cluster the data per frequency or set of frequencies [26].
- **Community detection** A community in a graph is a combination of nodes. These communities can then be found based on the properties of the graph, for example edge density [27]. There are many possible ways of going about community detection as there are lots of properties to base the detection on. A community can be seen as a kind of supernode or subgraph.

For rs-fMRI K-means, Ward, and ICA are mostly used. The Spectral and Community methods are especially useful when modeling directional and hierarchical networks [28]. If the complexity must be reduced, any of the above methods can be used more strictly to reduce the dimensionality as it depends on the number of nodes in the graph. For the actual node reduction, only the community detection and ICA methods are used. The other two methods are used to compute node specific features, which could then be used in the GUI N to find new networks by thresholding based on these features.

3.2.1. WARD CLUSTERING

We perform Ward's hierarchical clustering on the adjacency matrix, grouping nodes based on direct connectivity. The resulting cluster labels are used as node features. In addition, we compute cluster quality metrics, including the silhouette score (which measures the

cohesion and separation of clusters) and the standard deviation of cluster mean connectivity as a proxy for cluster separation. This approach emphasizes connectivity-driven grouping in the original node space [29].

3.2.2. SPECTRAL CLUSTERING

We apply k-means clustering to the top-k Laplacian eigenvectors (spectral embeddings) of the graph, where k is dynamically chosen based on the graph size (up to a maximum value). The cluster assignments are used as node-level features, while global properties such as the Calinski-Harabasz (CH) score are computed to quantify clustering compactness and separation in the spectral domain. This implementation reflects standard spectral clustering practices where k-means groups nodes based on frequency-like properties of their Laplacian embeddings.

3.2.3. COMMUNITY DETECTION

We apply the Louvain method [30], which detects communities (e.g. clusters) based on the edge density of graph. It will create a new supergraph, where each node indicates a community. The communities are further used to derive node-level features, such as participation coefficient (quantifying how evenly a node's connections are distributed across communities), within-community strength, and between-community strength. These features capture modular integration and segregation properties relevant to functional connectivity analyses.

3.2.4. INDEPENDENT COMPONENT ANALYSIS

ICA is a computational method used to separate a multivariate signal into additive, statistically independent components. In the context of fMRI data analysis, ICA is commonly applied to decompose brain activity signals into spatially independent patterns (components) and their associated time courses. These components can represent distinct neural networks, physiological noise (e.g., cardiac or respiratory artifacts), or scanner-related artifacts. This can improve the reliability of the graph features [31]. Unlike methods that rely on predefined models or regions of interest, ICA is a data-driven approach that does not require prior knowledge of the sources. This makes it particularly useful for identifying (functional) networks and isolating noise components for subsequent removal or further analysis.
Two separate ICA methods were used, namely canonical ICA (canICA), from Smith's brain maps [24], and group ICA [32].
For the predetermined canICA the strategy was as follows:

1. The Smith atlas provides precomputed ICA spatial maps $\mathbf{S}$ with different dimensionalities (10, 20, or 70 components).

2. Each ICA spatial component is mapped onto the AAL regions by using a mask to extract the spatial overlap or weight of each ICA component s_j in each AAL ROI x_i. This produces a transformation matrix $\mathbf{A}$ from ICA space to AAL space. This operation can be written as,

$$\mathbf{A}_{ij} = \frac{1}{V_i} \sum_{v} s_j(v) \tag{3.1}$$

where V_i is the number of voxels (3D pixels) in the i-th ROI and $s_j(v)$ the value of the j-th ICA component at voxel v.

3. To transform the original AAL time series into the Smith ICA component time series, the pseudo-inverse of this transformation matrix is computed (effectively mapping from AAL ROI signals back to ICA components). Mathematically this means,
$$\mathbf{A}^{\dagger} = \mathbf{A}^{T}(\mathbf{A}\mathbf{A}^{T})^{-1} \tag{3.2}$$

4. Each subject's standardized AAL time series (time × 116) is then projected by matrix multiplication with the pseudo-inverse, yielding time series in the Smith ICA component space.
$$\mathbf{X}_{Smith} = \mathbf{X}_{AAL}\mathbf{A}^{\dagger} \tag{3.3}$$

The strategy for the group ICA implementation was:

1. Each subject's AAL time series is standardized across time (z-scored) to normalize scales.

2. The standardized time series are pooled or concatenated together.
$$\mathbf{X}_{pooled} = \begin{bmatrix} \mathbf{X}^{(1)} \\ \vdots \\ \mathbf{X}^{(k)} \end{bmatrix} \tag{3.4}$$
Here k represents the subject index.

3. *FastICA*, from the scikit-learn python library, is applied on this concatenated data to extract group-level independent components as $\mathbf{S}_{ICA}$ by decomposing as,
$$\mathbf{X}_{pooled} = \mathbf{A}\mathbf{S}_{ICA} \tag{3.5}$$
where $\mathbf{A}$ is the mixing matrix.

4. To reconstruct each subject's ICA time courses, their original standardized time series is multiplied by the transpose of the unmixing matrix $\mathbf{U} = \mathbf{A}^{\dagger}$, resulting in individual component time series. The unmixing matrix satisfies $\mathbf{S} = \mathbf{U}\mathbf{X}$.
$$\mathbf{X}_{ICA} = \mathbf{X}_{AAL}\mathbf{U}^{T} \tag{3.6}$$

This approach follows the concept of group ICA with back-reconstruction, where a single ICA decomposition is computed on the pooled data, and individual subject time series are extracted by applying the mixing/unmixing matrices.

So while the first method converts the data into a set of predetermined independent components and is based on single subject ICA (canonical), the second approach learns these components through the input data which is slower but more flexible (e.g. $\mathbf{S}_{ICA}$ is not fixed, rather learned through data, while $\mathbf{S}_{Smith}$ is).

3.3. EDGE INFERENCE

The edges between the detected nodes from the previous section, can be inferred using various methods. Every edge can be seen as a measure of connectivity between the nodes. These edges can be statically determined from the data as with the statistical inference methods in section 3.5. Another possibility is to apply GSP based methods to infer a graph (see section 3.7).
The first step in edge inference consists of finding a suitable mathematical model to describe the system (see appendix E). This model can then be used to compute the adjacency matrix from the input data. Many methods use the empirical covariance matrix (see section 3.4) of the data as input to the graph inference method. In some cases, especially for state-space dynamic models (dynamic in the sense that $\mathbf{S}$ governs the next states of $\mathbf{x}$), residuals from a fitted model can be used to estimate a noise covariance matrix that further informs the graph structure. The GSP methods use the eigenvalues of the covariance matrix so these also have to be determined.
The last step in edge inference is to threshold the edges to remove noise and promote sparse graphs as they could potentially provide more distinction in the features.

3.4. SAMPLE COVARIANCE ESTIMATION

Many graph inference methods rely fundamentally on an accurate estimation of the sample covariance matrix. The quality of this estimation impacts the ability to recover meaningful and robust graph structures from data. Important properties of a suitable covariance estimator include:

- **Accuracy**, with respect to the true (unknown) covariance matrix;
- **Sparsity**, which is desirable when inferring sparse graphs;
- **Stability**, meaning low variance across different realizations of the data.

Typically, sample covariance estimation assumes independent and identically distributed (i.i.d.) samples, though time-varying and structured models can relax this assumption. Below, several common and advanced methods for estimating the sample covariance matrix are described.

3.4.1. DIRECT EMPIRICAL COVARIANCE

The most basic estimator is the empirical sample covariance matrix, computed as,

$$\hat{\Sigma} = \frac{1}{T-1} \sum_{t=1}^{T} (x_t - \bar{x})(x_t - \bar{x})^\top \tag{3.7}$$

where T is the number of time points, x_t is the sample at time t, and $\bar{x}$ is the sample mean. This estimator is unbiased under the i.i.d. assumption but can suffer from high variance, especially in high-dimensional settings.

3.4.2. SLIDING-WINDOW AVERAGED COVARIANCE

When dealing with potentially non-stationary signals (e.g., dynamic functional connectivity), it is common to estimate covariance over temporally localized windows [33],

$$\hat{\Sigma} = \frac{1}{W} \sum_{w=1}^{W} \hat{\Sigma}^{(w)} \tag{3.8}$$

where $\hat{\Sigma}^{(w)}$ is the empirical covariance computed within window w, and W is the number of windows. This method captures time-varying behavior, but introduces a trade-off between temporal resolution and estimation stability.

3.4.3. LEDOIT-WOLF SHRINKAGE ESTIMATOR

The Ledoit-Wolf shrinkage estimator [34] improves the empirical covariance matrix by shrinking it toward a structured target matrix F, reducing estimation variance at the expense of introducing a small bias,

$$\hat{\Sigma}_{\text{LW}} = (1-\lambda)\hat{\Sigma} + \lambda F \tag{3.9}$$

where $\lambda \in [0,1]$ is an automatically computed shrinkage coefficient. This estimator is particularly useful in high-dimensional regimes where the number of samples is small relative to the number of variables.

3.4.4. GRAPHICAL LASSO (SPARSE INVERSE COVARIANCE ESTIMATION)

The Graphical Lasso (Glasso) [35] estimates a sparse precision matrix $\Theta = \Sigma^{-1}$ from the empirical covariance 3.4.1, which directly encodes conditional independence relationships between variables,

$$\hat{\Theta} = \arg\min_{\Theta \succ 0} \left(\text{tr}(S\Theta) - \log\det\Theta + \lambda \|\Theta\|_1 \right) \tag{3.10}$$

where S is the empirical covariance matrix, and λ controls the degree of sparsity. Glasso is a popular method for inferring sparse graph structures from data and is robust to high dimensionality [36]. The methods 3.4.5 and 3.4.7 are both based on the Glasso. The magnitudes of the nonzero precision matrix entries, obtained through this problem, are a good contender to be used to define weighted edges in the inferred graph for fMRI based classification [37].

3.4.5. TIME-VARYING GRAPHICAL LASSO

The Time-Varying Graphical Lasso [38] combines the sliding-window approach with Graphical Lasso, applying sparsity-promoting estimation to each local covariance window,

$$\hat{\Theta}^{(w)} = \arg\min_{\Theta \succ 0} \left(\text{tr}(S^{(w)}\Theta) - \log\det\Theta + \lambda \|\Theta\|_1 \right) \tag{3.11}$$

This approach allows for the dynamic estimation of changing network structures over time.

3.4.6. VAR(k) - COVARIANCE OF RESIDUALS

When the data is assumed to follow an autoregressive process (see appendix E) of order k, i.e., VAR(k), the residuals of the model u_t provide an estimate of the innovation covariance [39],

$$\hat{\Sigma}_u = \mathrm{Cov}(u_t) \tag{3.12}$$

This estimator captures only the covariance of the new or unexplained variance after accounting for temporal dependencies, and can improve graph inference in dynamical settings.

3.4.7. NVAR - RESIDUALS + GRAPHICAL LASSO

The NVAR E method further enhances residual-based estimation by applying Graphical Lasso to the covariance of residuals S_u,

$$\hat{\Theta} = \arg\min_{\Theta \succ 0} \left(\mathrm{tr}(S_u \Theta) - \log\det\Theta + \alpha \|\Theta\|_1 \right) \tag{3.13}$$

where α controls sparsity. This approach combines the benefits of modeling temporal dependencies (via NVAR) with sparse structure learning (via Glasso).

3.5. STATISTICAL GRAPH INFERENCE METHODS

This section describes several widely used statistical methods for inferring brain connectivity graphs from time series data. These methods estimate a matrix of pairwise relationships between regions of interest (ROIs), which is then used to construct the graph. Each method makes different assumptions and captures different types of dependencies.

Let $X \in \mathbb{R}^{T \times N}$ denote the input time series data, where T is the number of time points and N is the number of ROIs. Each row of X corresponds to a time point, and each column to an ROI time series $x_i(t)$.

The aim is to estimate an adjacency matrix $W \in \mathbb{R}^{N \times N}$, capturing connectivity between ROIs based on X.

3.5.1. SAMPLE COVARIANCE

The sample covariance obtained in section 3.4, can also be used directly as adjacency matrix for inferring a graph by setting

$$W_{ij} = \Sigma_{ij} \tag{3.14}$$

This is a simple and computationally cheap way to obtain a graph that can be sensitive to noise or biases depending on the covariance estimation method.

3.5.2. PEARSON'S CORRELATION

The simplest and most widely used method is Pearson's Correlation Coefficient [40], which captures linear associations between pairs of time series. The correlation matrix R_x is computed as,

$$R_{ij} = \frac{\Sigma_{ij}}{\sigma_i \sigma_j} \tag{3.15}$$

where Σ_{ij} is the covariance, which has to be estimated, between regions i and j, and σ_i, σ_j are the standard deviations of their time series.
An edge between nodes i and j is defined based on a threshold $\sigma \geq \sigma_T$. Thresholding removes weak correlations that may be dominated by noise.
Assumptions: Linear dependence, stationary time series, symmetric relationships ($R = R^T$).
Advantages: Simple, fast, widely used in fMRI studies.
Limitations: Cannot distinguish direct from indirect relationships; sensitive to global confounds.
The implementation will be done using the *corrcoef* function from the numpy python library.

3.5.3. PARTIAL CORRELATION

Partial correlation [41] refines Pearson correlation by removing indirect relationships. It measures the linear dependence between two variables while conditioning on all others. To compute partial correlation, the precision matrix Θ is first estimated. The partial correlation is then computed as,

$$\rho_{ij|\text{others}} = -\frac{\Theta_{ij}}{\sqrt{\Theta_{ii}\Theta_{jj}}} \tag{3.16}$$

Assumptions: Linear dependence, stationary time series, multivariate Gaussianity.
Advantages: Captures direct relationships, removes confounding indirect effects.
Limitations: Requires accurate estimation of the precision matrix (see section 3.4); sensitive to estimation errors for small sample sizes.

3.5.4. MUTUAL INFORMATION

Mutual Information (MI) [42] captures both linear and non-linear dependencies between time series, without assuming a specific distribution. It is defined as,

$$I(x_i; x_j) = \sum_{x_i, x_j} p(x_i, x_j) \log \frac{p(x_i, x_j)}{p(x_i)p(x_j)} \tag{3.17}$$

Where $p(x_i, x_j)$ is the joint probability and $p(x_i), p(x_j)$ the marginal probabilities. MI is symmetric and non-negative, with higher values indicating stronger dependence.
Assumptions: Stationary data, sufficient sample size (~300+ samples recommended for reliable estimation using the Kraskov-Stögbauer-Grassberger (KSG) estimator).
Advantages: Captures complex non-linear relationships.
Limitations: Computationally intensive; estimation can be noisy for short time series; does not distinguish direct vs. indirect effects.
The implementation will be done using a python function from the scikit-learn library *mutual_info_regression*, which uses the KSG estimator that is based on k-nearest neighbors.

3.5.5. GRANGER CAUSALITY

Granger Causality (GC) [43] is a temporal inference method that assesses whether one time series can predict another. It is based on the improvement in prediction of $x_j(t)$

when including past values of $x_i(t)$ based on a VAR model (see appendix E). The GC statistic is computed as,

$$F_{i\to j} = \ln \frac{\mathrm{Var}(\epsilon_j)}{\mathrm{Var}(\epsilon_{j|i})} \tag{3.18}$$

where ϵ_j is the prediction error when modeling x_j without x_i, and $\epsilon_{j|i}$ is the error when including x_i.
Assumptions: Linearity, stationarity, sufficient temporal resolution (TR), proper lag order selection.
Advantages: Provides directed, time-resolved connectivity estimates.
Limitations: Sensitive to temporal resolution (fMRI TR may be too slow); assumes linear interactions; model selection can be challenging.
The implementation will be done using *grangercausalitytests* from the stattools python library. The standard implementation of GC relies on a linear VAR model to capture temporal dependencies between variables. However, this framework can be generalized by adopting alternative models of time dependency, such as those used in Causal Graph Processes (CGP), which is further explored in section 3.7.3. This process enables the modeling of both dynamic and structured dependencies beyond the linear VAR assumption.

3.6. THRESHOLDING

All of the above mentioned methods will result in a symmetric matrix, some which can contain negative values. These matrices will in general be very dense, and possibly noise, and thus should be thresholded. This can be done in a binary manner, to obtain a binary adjacency matrix B, or by considering the actual weights to obtain a weighted adjacency matrix W. It is often more useful to consider the actual weights to grasp an idea of the importance of the edges, however the traditional graph features (e.g. centralities) will not consider these weights.
The thresholding is done by removing the bottom % of edges based on a value $threshold$, this method of can be called percentile based thresholding.

3.7. GSP GRAPH INFERENCE METHODS

3.7.1. SPECTRAL TEMPLATES OR RSPECT

The method implemented in this section is derived from [44]. For this method we estimate the so-termed graph-shift operator (GSO) [45].

Fundamentally, the unknown GSO S encodes direct relationships between signal elements, which we aim to recover from observable indirect relationships generated by a diffusion process on the graph. This diffusion process is described by the following equation,

$$\mathbf{x} = \alpha_0 \prod_{l=1}^{\infty} (\mathbf{I} - \alpha_l \mathbf{S})\,\mathbf{w} = \sum_{l=0}^{\infty} \beta_l \mathbf{S}^l \mathbf{w} \tag{3.19}$$

Where $\mathbf{w}$ is a zero-mean white signal with covariance matrix $\mathbb{E}\left[\mathbf{w}\mathbf{w}^T\right] = \mathbf{I}$. Our goal is to recover the fundamental relationships described by $\mathbf{S}$ from a set $\{\mathbf{x}_p\}_{p=1}^{P}$ of P indepen-

dent samples of the random signal $\mathbf{x}$.
It turns out that, as long as the matrices $\mathbf{S}$ and $\mathbf{C}_x$ have the same eigenvectors, filter coefficients that generate x through a diffusion process on $\mathbf{S}$ exist. This makes the problem undetermined, there exist multiple solutions for $\mathbf{S}$. We select the $\mathbf{S}$ with the minimum l_0 norm.
For the case that we want $\mathbf{S}$ to be an adjacency matrix the problem is stated as,

$$\begin{aligned} \hat{\mathbf{S}}^* &:= \underset{\{\mathbf{S},\boldsymbol{\lambda},\mathbf{S}'\}}{\operatorname{argmin}} \|\mathbf{S}\|_1 \\ \text{s. to} \quad & \mathbf{S}' = \sum_{k=1}^{N} \lambda_k \hat{\mathbf{v}}_k \hat{\mathbf{v}}_k^T, \quad \mathbf{S} \in \mathscr{S}, \quad \left(d\left(\mathbf{S},\mathbf{S}'\right)^2 + \left(\sum_j S_{j1} - 1 \right)^2 \right)^{1/2} \leq \epsilon \end{aligned} \tag{3.20}$$

Where $\hat{\mathbf{v}}_k$ are the eigenvectors of the sample covariance matrix $\hat{\boldsymbol{\Sigma}}$ and $\mathscr{S}$ is chosen as,

$$\mathscr{S} = \mathscr{S}_{\mathrm{A}} := \left\{ \mathbf{S} \mid \mathbf{S} \in \mathscr{M}^N, S_{ii} = 0 \right\}$$

For the case that we want $\mathbf{S}$ to be a normalized Laplacian matrix (see Appendix E.1 for more context on the Laplacian) the problem is stated as,

$$\begin{aligned} \hat{\mathbf{S}}^* &:= \underset{\{\mathbf{S},\boldsymbol{\lambda},\mathbf{S}'\}}{\operatorname{argmin}} \|\mathbf{S}\|_1 \\ \text{s. to} \quad & \mathbf{S}' = \sum_{k=1}^{N} \lambda_k \hat{\mathbf{v}}_k \hat{\mathbf{v}}_k^T, \quad \lambda_1 = 0, \quad \mathbf{S} \in \mathscr{S}, \quad d\left(\mathbf{S},\mathbf{S}'\right)^2 \leq \epsilon \end{aligned} \tag{3.21}$$

Where $\hat{\mathbf{v}}_k$ are the eigenvectors of the sample covariance matrix $\hat{\boldsymbol{\Sigma}}$ and $\mathscr{S}$ is chosen as,

$$\mathscr{S} = \mathscr{S}_{\mathrm{L}} := \left\{ \mathbf{S} \mid S_{ij} \geq -1 \text{ for } i \neq j, \quad \mathbf{S} \in \mathscr{M}_+^N, \quad S_{ii} = 1 \text{ for all } i \right\}$$

We implement 3.20 and 3.21 in python using the CVXPY package and a Splitting Conic Solver [46]. Note that in the python implementation of this algorithm the variable ϵ is called α. These methods suffer from a high computational complexity, namely $\mathscr{O}(n^3)$. Consequently the models have a long runtime, thus only a single covariance estimate is employed — specifically, the direct empirical covariance. This is common practice for this method. After solving 3.20 or 3.21 the graph is normalized and thresholding is applied based on the chosen $\mathscr{S}$. In the case of the normalized Laplacian $\mathscr{S}_L$ we use the following,

$$\left(\hat{\mathbf{S}}_T^*\right)_{ij} = \begin{cases} \hat{\mathbf{S}}_{ij}^* / \left|\max\left\{\hat{\mathbf{S}}^*\right\}\right|, & \text{if} \quad \hat{\mathbf{S}}_{ij}^* / \left|\max\left\{\hat{\mathbf{S}}^*\right\}\right| \geq t, \\ 0, & \text{if} \quad \hat{\mathbf{S}}_{ij}^* / \left|\max\left\{\hat{\mathbf{S}}^*\right\}\right| < t. \end{cases}$$

In the case of the adjacency matrix $\mathscr{S}_A$ we use the following,

$$\left(\hat{\mathbf{S}}_T^*\right)_{ij} = \begin{cases} \dfrac{\hat{\mathbf{S}}_{ij}^* - \min\{\hat{\mathbf{S}}^*\}}{\max\{\hat{\mathbf{S}}^*\} - \min\{\hat{\mathbf{S}}^*\}}, & \text{if} \quad \dfrac{\hat{\mathbf{S}}_{ij}^* - \min\{\hat{\mathbf{S}}^*\}}{\max\{\hat{\mathbf{S}}^*\} - \min\{\hat{\mathbf{S}}^*\}} \geq t, \\ 0, & \text{if} \quad \dfrac{\hat{\mathbf{S}}_{ij}^* - \min\{\hat{\mathbf{S}}^*\}}{\max\{\hat{\mathbf{S}}^*\} - \min\{\hat{\mathbf{S}}^*\}} < t. \end{cases}$$

The hyperparameters ϵ and t are chosen by computing the graph and inputting the weights to a basic SVM. Then K-Fold cross-validation is used to determine which values are most suited for determining ASD using the ABIDE dataset.

3.7.2. rLOGSPECT

The method implemented in this section is derived from [47]. Unfortunately rSpecT is sensitive to the hyperparameter ϵ, and improper tuning of it may jeopardize the model performance or lead to model infeasibility.

The failure of rSpecT lies in the existence of the constraint $(\boldsymbol{S}\mathbf{1})_1 = 1$, which is used to preclude the trivial solution $\boldsymbol{S}^* = \mathbf{0}$. For this reason, we resort to an alternative approach to exclude the trivial solution. We implement a log barrier to preclude the zero solution. The problem now becomes,

$$\begin{aligned} \hat{\mathbf{S}}^* := \underset{\{\mathbf{S},\boldsymbol{\lambda},\mathbf{S}'\}}{\operatorname{argmin}} & \|\boldsymbol{S}\|_{1,1} - \alpha \mathbf{1}^\top \log(\boldsymbol{S}\mathbf{1}) \\ \text{s.t.} \quad & \boldsymbol{S} \in \mathscr{S}, \left\| \boldsymbol{S}\hat{\Sigma}_n - \hat{\Sigma}_n \boldsymbol{S} \right\|_F^2 \leq \delta_n^2 \end{aligned} \tag{3.22}$$

Where $\hat{\Sigma}$ is the sample covariance matrix based on n samples. From [47] it follows that the choice of the parameter α only affects the scale of $\mathbf{S}$, thus we set $\alpha = 1$.

If the input $\mathbf{w}$ follows a sub-Gaussian distribution and $\delta_n = \mathcal{O}(\sqrt{\log n / n})$, then the solution of rLogSpecT converges to the solution of the similar problem when the covariance matrix is known, called LogspecT, almost surely (corollary 4.8 in appendix section G.2).

We implement 3.22 in python using a linearized ADMM algorithm designed by the authors of the method rLogSpecT, [47]. For similar reasons as in the method rSpecT, only the direct empirical covariance is used as a covariance estimate. $\mathscr{S}$ was chosen as,

$$\mathscr{S} := \left\{ \boldsymbol{S} \in \mathbb{R}^{m \times m} \mid \boldsymbol{S} = \boldsymbol{S}^\top, \operatorname{diag}(\boldsymbol{S}) = \mathbf{0}, \boldsymbol{S} \geq \mathbf{0} \right\}$$

After solving 3.22 the graph is normalized and thresholding is applied in the following way,

$$\left(\hat{\mathbf{S}}_T^*\right)_{ij} = \begin{cases} \frac{\hat{\mathbf{S}}_{ij}^* - \min\{\hat{\mathbf{S}}^*\}}{\max\{\hat{\mathbf{S}}^*\} - \min\{\hat{\mathbf{S}}^*\}}, & \text{if } \frac{\hat{\mathbf{S}}_{ij}^* - \min\{\hat{\mathbf{S}}^*\}}{\max\{\hat{\mathbf{S}}^*\} - \min\{\hat{\mathbf{S}}^*\}} \geq t, \\ 0, & \text{if } \frac{\hat{\mathbf{S}}_{ij}^* - \min\{\hat{\mathbf{S}}^*\}}{\max\{\hat{\mathbf{S}}^*\} - \min\{\hat{\mathbf{S}}^*\}} < t. \end{cases}$$

The hyperparameter δ_n is chosen as $A\sqrt{\log n / n}$. Note that in the python implementation of this algorithm the variable A is called α. The hyperparameters A and t are chosen by computing the graph and inputting the weights to a basic SVM. Then K-Fold cross-validation is used to determine which values are most suited for determining ASD using the ABIDE dataset.

3.7.3. CAUSAL GRAPH PROCESS

The Causal Graph Process (CGP) is briefly mentioned in [48], but is originally from [49]. The definition of the CGP can be written as,

$$\mathbf{x}_t = \sum_{l=1}^{L} h_l \mathbf{S}^l \mathbf{x}_{t-l} + \mathbf{w}_t \tag{3.23}$$

Where **S** is the GSO (can be asymmetric), $\mathbf{x}_{t-l}$ indicate the lagged components, h_l represent the graph filter coefficients and $\mathbf{w}_t$ the white noise process. The process is based on the VAR process used for GC (see section 3.5.5). Where in GC each node can influence every other node, in CGP this influence is constrained by the GSO. The filter components h_l define how the signals diffuse over the graph at different lags.

Then the full problem can be written as a nonconvex optimization problem (see appendix H).

$$\min_{\mathbf{S},\mathbf{h}} \left\{ \sum_{t=L+1}^{T} \left\| \mathbf{x}_t - \sum_{l=1}^{L} \mathbf{H}_l(\mathbf{S},\mathbf{h})\mathbf{x}_{t-l} \right\|_2^2 + \alpha\|\mathbf{S}\|_1 + \beta\|\mathbf{h}\|_1 \right\} \tag{3.24}$$

Where T represents the total amount of time samples and L the amount of time shifts to consider. α and β are the tuning parameters for sparsity of **S** and **h** respectively.
It will learn a shift operator **S** and temporal filters coefficients h_l based on an initial guess for both the shift operator and the filter polynomial, given some input timeseries data $\mathbf{x}_t$. Due to the ℓ_1 norms on **S** and **h**, the resulting graph shift and filter matrices will be sparse depending on the tuning parameters.

To solve the problem from 3.24, any gradient-based optimization algorithm can be used. As there are three different tuning parameters, an adjustable learning rate per parameter for the algorithm is suited. The algorithm used is Adam [50], because it has adaptive learning rates, and is widely used.
Another important aspect for the problem lies in the initial guess of the GSO and filter. This is especially hard in fMRI analysis as there is no known ground truth matrix. An option would be to start with a random connectivity matrix, some anatomical atlas or a connectivity matrix from another inference method in section 3.5.
The value of L (=TR) should be taken with respect to the autocorrelation function (ACF) of the timeseries I. For fMRI that would be ≈2-3 lags for TR=2s, as the BOLD-signal should peak at 4-6 seconds [51], [52]. However, too large lags will consider in faulty results.
Cross-Validation can be used to choose the parameters such that the Mean Squared Error (MSE) of the validation will be minimized. This can also be used to test for stability of the GSO estimation. On top of that, the consistency of **S** should be tested between subsets of the data and for different initial guesses for **S** and **h**. The initial guess for **S** is taken as an $N \times N$-matrix with elements drawn from a normal distribution $\mathcal{N}(0,1)$ and the initial guess for **h** as a list of length L with elements drawn from a normal distribution $\mathcal{N}(0,0.1^2)$.

The causal nature of the process gives rise to more temporal and lag based features. The graph will also be directional which can also be modeled as features. An example of lag based features would be to use the filter coefficients directly.

3.8. GRAPH FEATURES

From an estimated graph, features can be extracted. These features can be directly taken from the adjacency matrix (e.g. the edge weights) or they can be computed from the timeseries per node to compute statistical features or graph specific features. The de-

signed features can be used as a connectivity metric for visualization or used for classification. Examples are to check for relations/patterns like: clustering, subnetworks, degrees per node/cluster, variability (related to frequency), spatial/temporal dependence in weights (see appendix F).
These features can be subdivided into three types: node specific, node to node, or global. Doing so will make it easier to store, label, and plot them. Also, the global features (or sometimes called summary features) can be the average of a set of node specific features, and are thus less sensitive to noise. They also grasp more information per feature. Although this is the case, node specific and node to node features are still very helpful, as they have a higher spatial resolution and thus can be used to visualize connections or regions of interest better. This makes them especially helpful in connectivity analysis and thus in a visual application tool for researchers.

3.9. TESTING PROCESS

To ensure the validity and relevance of the derived graphs and features, it is essential to perform appropriate validation tests. The features will be tested and validated based on the classification results and separate statistical tests.

3.9.1. AUTOCORRELATION

Autocorrelation measures the temporal dependency within fMRI signals, reflecting both physiological processes and measurement characteristics. As discussed in 3.7.3, the autocorrelation is expected to be significantly positive in the range 1-3 TRs. The presence of moderate autocorrelation justifies the use of temporal models such as Granger causality, and its influence was considered in interpreting correlation-based functional connectivity estimates. The presence and degree of autocorrelation can be assessed by plotting the autocorrelation function (ACF) of the time series (see appendix I). For practical purposes, autocorrelation can be considered negligible if the average ACF magnitude across lags is below 30% of the peak ACF value, relative to the mean. However, this threshold is heuristic and should be interpreted cautiously depending on the analysis context.

3.9.2. TESTING FOR ERGODICITY

Ergodicity states that the average of some signal in time gives a good representation of the average of the ensemble, which is the sum of all possible states in the system. The process is then ergodic if it follows this principle.
Many of the statistical inference methods require i.i.d. samples, as they require an estimate for the covariance. As this is hard to prove, ergodicity can be used to neglect this property.
The sample covariance converges for ergodic processes as,

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} (x_t - \bar{x})(y_t - \bar{y}) = \mathbb{E}[(X - \mu_X)(Y - \mu_Y)] \tag{3.25}$$

To check for ergodicity on the fMRI data, two aspects must be tested. The first is that the process should be Wide Sense Stationary (WSS), and the other is the mixing property.

WIDE SENSE STATIONARY

Wide Sense Stationarity (WSS) is a milder version of Strict Stationarity (SS), meaning no time dependence. As the methods used in this report require only WSS, which is easier to prove than SS, this can be tested instead. WSS just indicates a process where the mean μ and variance σ^2 do not change over time, e.g. they are constant. This means that the autocorrelation $R_{x,x}(\tau)$ depends only on the time difference τ (see appendix I). To prove these facts, it is possible to split-up the data into blocks and test the mean and variances of each block to see how stable they are.
WSS of a timeseries can be tested using the Augmented Dickey-Fuller (ADF) [53] test. Here, the null hypothesis H_0 states that the timeseries has a unit root, indicating non-stationarity. The null hypotheses should be rejected for p-value0.05, indicating stationarity.

MIXING

The mixing property says that the autocorrelation decays sufficiently fast such that samples that are far enough apart from each other can be assumed time-independent. Mathematically,

$$\lim_{\tau \to \infty} R_{x(t),x(t+\tau)} = 0 \tag{3.26}$$

Compute the autocorrelation via 3.26, and test for which value of τ it is below some threshold. In our data τ can take steps according to the time sample difference $\Delta t = TR \approx 2000ms$.
For the limited data this might be hard to prove and is unnecessary. In general it is assumed that within the fMRI measurement, the time averages to approximate ensemble properties [54] and is thus ergodic to a certain degree. [55] States that the violation of ergodicity is mitigated during the pre-processing.

3.9.3. GRAPH VALIDATION

By checking the stationarity, the reliability of the methods requiring this, including the ones that use covariance, will be enforced.

To be able to compare the different methods against each other and test their relevance, classification metrics can be used. The metrics include accuracy, recall (sensitivity), MSE (variance) and AUC (decision overlap), which are further explained in L.

3.10. NASDA GUI DESIGN PLAN

The actual product to be made, and thus the final goal, is a Graphical User Interface (GUI) that plots the brain functional connectivity based on the designed features. Therefore, it should take the feature data directly and use it to plot it on a brain map.
As the output of the features is a dataframe consisting of all personal information per subject and the corresponding features, the GUI should take this as input. To speed up the generation process in the GUI itself, the dataframes should be stored within the GUI itself. The Nilearn plotting functions *connectome* and *glass*, will be used to visualize the edge and node features respectively.

To make this work, all the different types of dataframes have to be stored for the different inference and ICA combinations. It is also possible to select a fixed range of thresholds per method, however this would require $(N_{methods} \times N_{ica}) \times N_{thresholds}$ more dataframes. The user will essentially just switch between the pre-computed dataframes. This also gives the possibility for the user to run their own data, however this will require the actual computation code to be available in the GUI executable. This will also require the user's system to be compatible with the code and it will also have to compute on their system, which can take a long time.
The visualization based on the dataframe can be done without the GUI. The plotting function can be ran directly through the code to obtain some intermediate results (see appendix J).

4

RESULTS

This chapter will present all the results obtained from experimentations with the ABIDE dataset and test datasets during this project. In appendix L additional classification results are shown and a brief explanation of the classification metrics. All classification is done using 5-fold cross validation using the classifiers Support Vector Machine (SVM) and Multi-Layer Perceptron(MLP), with k-nearest neighbor feature selection for $k = 30$.

4.1. COMPARISON OF PRE-PROCESSING SETTINGS

The testing is done using the *mutual_info* graph inference method, considering only the graph features, and considering only subjects from the *NYU* database to reduce the influence of site specific noise and artifacts.

4.1.1. EFFECT OF FILTERING

The best results (see 4.1) are obtained when using both the *global signal regression* and the *band-pass filtering* options. The differences are significant enough to state this, even with the relatively small amount of data (172 subjects) used for this test. From table 4.1,

Filter option	Accuracy raw	MSE	Class 1 recall	AUC
BPF+GSR	0.57	0.43	0.13	0.79
BPF	0.51	0.49	0.07	0.57

Table 4.1: Table showing the classification results for different filtering options using a support vector machine classifier (SVM). For the accuracy, Class 1 recall and AUC higher is better, for the MSE lower is better.

it can be assumed that using both band-pass filtering and global signal regression will yield best classification results using this database. Therefore, to obtain the rest of the results, these pre-processing steps will be used to fetch the data.

4.1.2. EFFECT OF INDEPENDENT COMPONENT ANALYSIS

The test is done using the glasso precision matrix directly as the adjacency matrix with a percentile threshold of 0.74. The test is ran over data loaded using: AAL brain atlas, (10,20,70)-component canICA and (5,10,15,20,25,30,35,40,45,50,70)-component groupICA. The results are shown in table 4.2.

Table 4.2: Top 5 best ICA maps in terms of overall classification metrics

Rank	Classifier	Components	Accuracy	F1 Score	AUC	MSE
1	SVM	15 groupICA	0.603	0.482	0.613	0.397
2	SVM	40 groupICA	0.599	0.515	0.590	0.401
3	MLP	40 groupICA	0.582	0.497	0.608	0.418
4	MLP	30 groupICA	0.571	0.466	0.590	0.429
5	MLP	15 groupICA	0.578	0.442	0.541	0.422

The results from 4.2 clearly show that the groupICA outperformed both the canICA from Smith (RSN10,RSN20,RSN70) and the raw AAL brain atlas using the glasso method. The results also show that the 15 component groupICA is the best contender for classifying ASD, these components can be found in M.

4.2. TUNING/TESTING OF SPECTRAL TEMPLATE METHODS

4.2.1. RSPECT

First rSpecT was tested on generated data from an Erdős–Rényi (ER) graph. The amount of nodes N=20 and the probability of including an edge p is 0.2. $\mathscr{S}$ is chosen as the set of normalized Laplacian matrices $\mathscr{S}_L$. The data is generated through a diffusion process in the normalized Laplacian matrix of the ER graph,

$$x = (I + S + S^2)w$$

The hyperparameter ϵ was chosen as the smallest value s.t. the problem is still feasible and the threshold was set at 0.05. The relative error of the Laplacian matrix was computed for a different number of samples as can be seen in figure 4.1. It can be seen that a relative error of < 0.1 is only reached at 10^4 samples. In the ABIDE dataset only 150-300 samples are available per participant, this example illustrates that tight recovery (a relative error of < 0.1) is not a realistic expectation for the ABIDE dataset. For this particular filter model a heatmap of the true GSO S and the learned S can be seen in Appendix G.3 figure G.1.

rSpecT was also tested on ER graphs with a varying number of nodes from $N = 10$ to $N = 50$ at 10^5 samples. The test was performed in the same way as the previous test and the results can be seen in Appendix G.3 figure G.2. It can be seen that, for ER graphs in particular, rSpecT has trouble recovering a graph with more than 30 nodes, even at 10^4 samples. This is another motivation to implement ICA.

To tune the hyper-parameters for rSpecT 20 canICA is applied to reduce the dimensionality. A range of ϵ between $[0, 0.25]$ was chosen and thresholding was applied with t be-

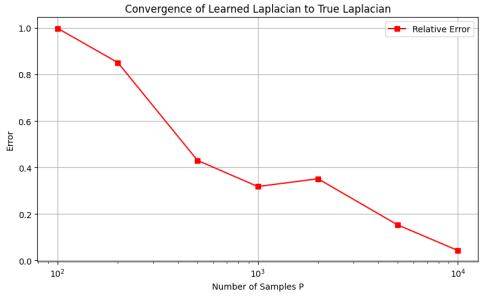
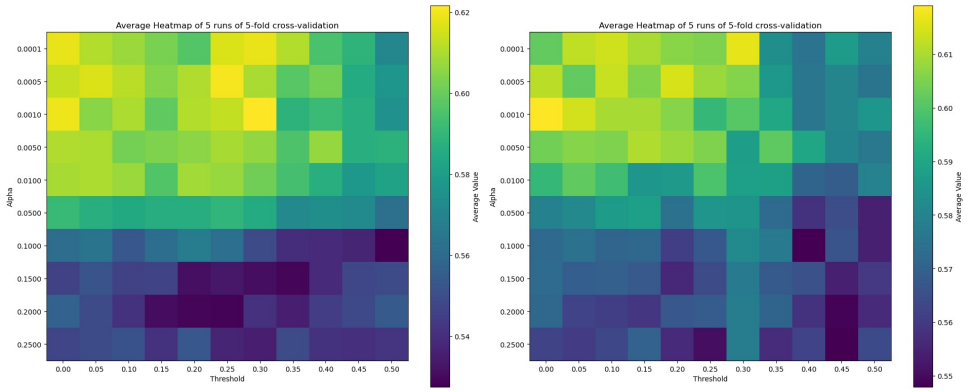


Figure 4.1: The relative error of the Laplacian matrix of rSpecT on an ER graph with $\mathscr{S} = \mathscr{S}_L$

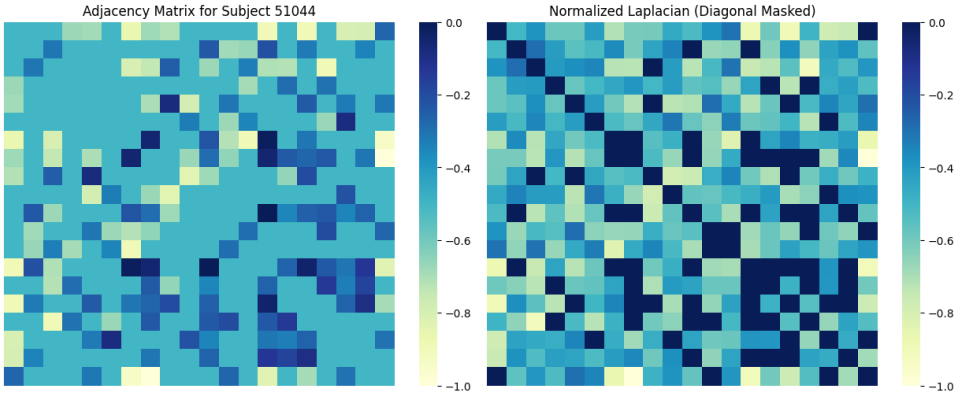


(a) $\mathscr{S} = \mathscr{S}_L$ (b) $\mathscr{S} = \mathscr{S}_A$

Figure 4.2: Heatmap of cross validation accuracy for different combinations of parameter values for rSpecT

tween $[0, 0.5]$. The results of which can be seen in the heatmap in figure 4.2. Based on these results and the fact that we want to promote sparsity we choose for rSpecT on $\mathscr{S}_L$, $[\epsilon, t] = [0.001, 0.3]$ and on $\mathscr{S}_A$, $[\epsilon, t] = [0.0001, 0.3]$. For these values the distribution of the subjectwise matrix distance between consecutive values of ϵ was plotted, see figure G.6 in Appendix section G.3.

From these figures it can be concluded that the GSO S converges for the chosen values of alpha. An example of a computed adjacency matrix and Laplacian matrix can be seen in figure G.4 and G.3 in Appendix section G.3. The normalized Laplacian matrix of the computed adjacency matrix can also be computed. This Laplacian matrix is shown next to the Laplacian matrix computed by rSpecT (on which some rescaling is done s.t. it's values are between [-1,0]) in figure 4.3. There are clear similarities between both matrices, which indicates that the Laplacian matrix and adjacency matrix solutions of rSpecT represent a very similar graph structure, this is reflected by their similar accuracy in the machine learning model.



(a) rSpecT on subject 51044 with $\mathscr{S} = \mathscr{S}_L$ normalized to the interval [0,1]

(b) The Laplacian computed from rSpecT on subject 51044 with $\mathscr{S} = \mathscr{S}_A$

Figure 4.3: A comparison between the computed Laplacian and the Laplacian derived from the computed Adjacency matrix for rSpecT

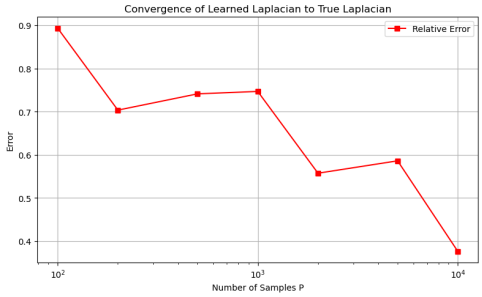


Figure 4.4: The relative error of the adjacency matrix of rLogSpecT on an ER graph

4.2.2. rLOGSPECT

This method was similarly tested on the adjacency matrix of an ER graph as with rSpecT, the results of which can be seen in 4.4. For parameter tuning a range of A between $[0,10]$ was chosen and thresholding was applied with t between $[0,0.5]$. The results of which can be seen in the heatmap in figure 4.5. Based on these results and the fact that we want to promote sparsity we choose $[A,t] = [0.05,0.45]$. For these values the distribution of the subjectwise matrix distance between consecutive values of ϵ was plotted, see figure G.7 in Appendix section G.1.
From these figures it can be concluded that the GSO S converges for the chosen values. An example of a computed adjacency matrix can be seen in figure G.5.

4.3. COMPARISON OF GRAPH INFERENCE METHODS

In this section the results of testing different sample covariance estimates, graph inference methods and features on the classifier are listed. All results from subsections 4.3.1,

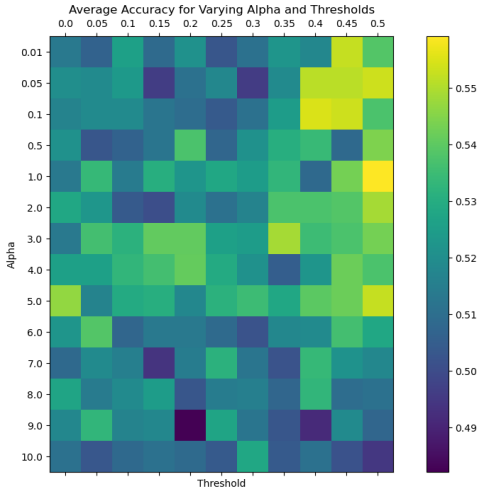


Figure 4.5: Heatmap of cross validation accuracy for different combinations of parameter values for rLogSpecT

4.3.3, and 4.3.2 are obtained using 20 component canICA from Smith (e.g. RSN20) over the dataset after removing subjects with zero variance timeseries (816 subjects total of which 374 ASD and 442 Allistic).

4.3.1. COVARIANCE ESTIMATE

The test is done by using the weights of the covariance matrix as features. The best covariance methods ranked by AUC are listed in table 4.3.

Table 4.3: Top covariance methods ranked by AUC, all best results are obtained using the SVM classifier and the sample covariance graph inference.

Rank	Cov method	Threshold	Accuracy	AUC	Recall	MSE
1	glasso	0.5	0.593	0.642	0.484	0.407
2	window	0.9	0.547	0.585	0.364	0.453
3	direct	0.9	0.532	0.557	0.278	0.468
4	ledoit	0.9	0.537	0.556	0.321	0.463
5	nvar	0.5	0.547	0.535	0.310	0.453
6	var	0.9	0.531	0.518	0.202	0.469
7	tv-glasso	0.5	0.518	0.524	0.393	0.482

4.3.2. INFERENCE METHOD

The test is done by applying all the discussed graph inference methods on the 20 component canICA data. The features used are the graph based features. If any covariance estimate other than the direct estimate is used, it is noted after the inference method. The inference methods, including the glasso sample variance, are ranked based on AUC

in table 4.4.

Table 4.4: Top inference methods ranked by AUC, all best results are obtained using the SVM classifier.

Rank	Inf method & cov method	Threshold	Accuracy	AUC	Recall	MSE
1	sample-cov glasso	0.5	0.593	0.642	0.484	0.407
2	rSpecT-adjacency	0.3	0.580	0.625	0.539	0.420
3	rSpecT-laplacian	0.3	0.588	0.615	0.573	0.412
4	partial-corr glasso	0.9	0.571	0.573	0.457	0.429
5	pearson-corr	0.5	0.551	0.547	0.412	0.449
6	mutual-info	0.9	0.539	0.558	0.409	0.461
7	rLogSpecT	0.5	0.521	0.524	0.257	0.479
8	gr-causality	0.5	0.528	0.525	0.348	0.472
9	cgp	0.5	0.542	0.511	0.493	0.458

4.3.3. IMPACT OF FEATURES

Using the obtained adjacency matrix from the graphical lasso, the features from appendix F are computed. The type of features are stored into categories as node features, node to node features, and global features. These can be further categorized as statistical, graph-based, and for the node level whether they are cluster-based. All possible combinations of these subsets are tested, the results are shown in table 4.5.

Table 4.5: Top feature subsets ranked by AUC, all best results are obtained using the glasso for graph inference. The graph feature subset means all the graph-based features (node, node-to-node, global). Node means all node-level features.

Rank	Feature subset	Classifier	Accuracy	AUC
1	node-to-node	SVM	0.593	0.642
2	node	MLP	0.554	0.539
3	graph	SVM	0.539	0.561
4	node cluster	SVM	0.518	0.534
5	stat	SVM	0.516	0.509
6	global graph	SVM	0.545	0.482

The results from 4.5 show that the edge weights of the adjacency matrix (e.g. node-to-node) best grasp the ASD properties from the ABIDE dataset. The rest of the features score quite a bit lower compared to the edge weights. The graph methods seem to be working better when compared to the used statistical method. The selected features per category can be found in appendix F.

4.4. STATISTICAL TEST RESULTS

For these tests, the statistical significance was set to be $\alpha = 0.05$, corresponding to a confidence interval of 95%. The tests consisted of Dickey-Fuller (ADF) (to prove station-

arity), Bartlett (to prove variance ergodicity), Levene (to prove variance ergodicity) and comparing the time averages with the ensemble averages for mean ergodicity. The results from 4.6 indicate weak mean ergodicity, due to high variability, with distributional differences. However, the Levene p-value shows a significant variance ergodicity. Also, the ROIs seem relatively stationary with some outliers.

% Stationary ROIs (ADF p<0.05)	92.9%
Average ADF p-value	0.0161
Average Lag-1 Autocorrelation	0.8853
Mean Ergodicity Ratio	1.000 ± 2.329
Levene p-value	0.0213

Table 4.6: Table showing the ergodicity test results

The effect of autocorrelation can be neglected after 5 lags (=5 TRs=10s), as the ACF shows relatively low values (all below 30% of the peak) after the 4th lag (see appendix I for the plots). This gives an indication of what lags to consider for VAR based methods and indicates the process to be ergodic, as the autocorrelation 'converges' after 5 lags.

4.4.1. VALIDATING STATISTICAL INFERENCE METHODS

From section 3.9.2, the fMRI process used for computations can be considered ergodic with stationary variance.
As Pearson's Correlation computes the sample covariance and divides this by the sample variance, the correlation matrix obtained using this method should be reliable given an ergodic process with stationary variance. The same thing can be said for the other methods using the covariance.
Without ergodicity, functional relationships inferred from fMRI time series, such as those estimated via Granger causality or mutual information, may be overly influenced by transient fluctuations that are not representative of the underlying neural dynamics. Ergodicity ensures that time-averaged statistics reflect ensemble properties, supporting the stability and generalizability of these measures [41]. This also applies to the causal graph process.

4.5. FINAL GUI

The designed GUI can be seen in figure N.1. The GUI has options to load custom data, atlasses, and ICA components. As the input data takes csv, a sample is included (no meaningful data). We have stored a numerous amount of dataframes and ICA components which can be used with the GUI. The GUI also shows information on the data, it has the possibility of dynamically changing the threshold for the features used to plot, and can plot all the features that are extracted in this project. It has a *glass* plot to display node features and a *connectome* plot to display connectivity features (e.g. node to node) J. The GUI uses the AAL brain atlas as a default.

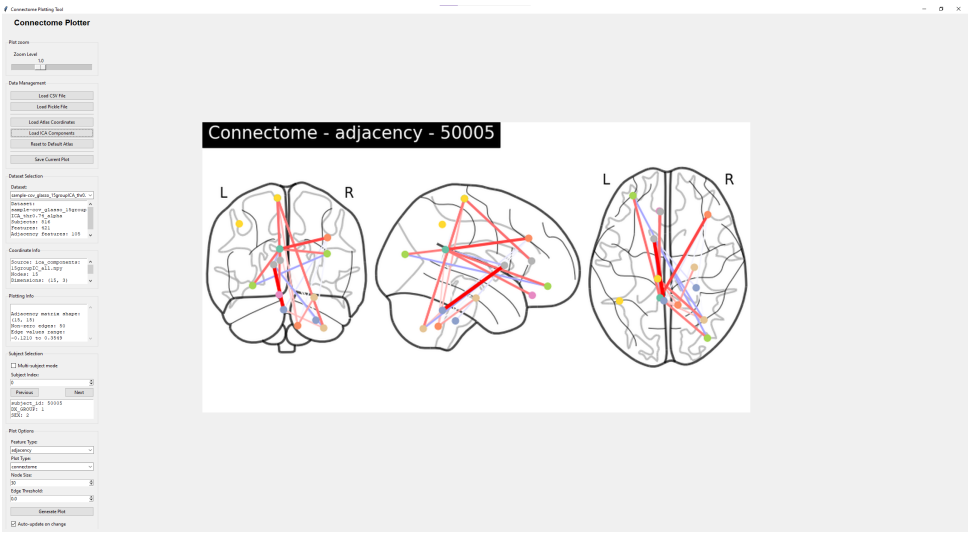


Figure 4.6: Screenshot from the designed GUI, plotting a connectome for the glasso method using 15 component groupICA.

5

CONCLUSION

This thesis explores a wide range of inference schemes, providing a clear opportunity for cross-validation and model comparison. A crucial first step in fMRI analysis is selecting the appropriate preprocessing strategy.
Based on the experiments conducted, pipelines that included band-pass filtering and global signal regression consistently yielded the best classification performance on the ABIDE dataset.
Independent Component Analysis (ICA) proved to be an effective method for dimensionality reduction. Specifically, group ICA with 15 components achieved the highest classification performance, reaching 60.3% accuracy and 61.3% AUC when paired with an SVM classifier.
The choice of covariance estimation method also significantly influenced classification results. Among the methods tested, graphical lasso achieved the best performance, with 59.3% accuracy and 64.2% AUC.
Various graph inference techniques were evaluated. The top three performing methods were: sample covariance with graphical lasso (59.3% accuracy, 64.2% AUC), rSpecT-adjacency (58.0% accuracy, 62.5% AUC), and rSpecT-laplacian (58.8% accuracy, 61.5% AUC). Among these, the GSP-based inference method rSpecT outperformed both the statistical methods and other GSP methods, but none of the methods outperformed the graphical lasso covariance estimate.
Among all feature categories, the edge weight features performed best. When computed from the graphical lasso covariance matrix, these features achieved an accuracy of 59.3% and an AUC of 64.2%.
Finally, the connectivity patterns visualized using the developed tools revealed distinct differences between methods, offering promising insights for neuroscientific interpretation and potential biomarker discovery.

6

DISCUSSION

Overall, there are a lot of different methods implemented and working. There is also a wide range of different features used, also statistical for reference. This can provide a lot of reliability when used to cross-validate based on the same setup instead of using only results from other papers.
The final version of the features was completed relatively late in the project timeline, which limited the ability of the classification and feature selection teams to fully evaluate them. As a result, more comprehensive testing could not be performed, potentially limiting the accuracy and depth of the final results.
It would be a good idea to start testing intermediately, every time a new model has been implemented. A better literature research at the start might provide a clearer path to take for this project. The pre-processing and data should be looked at as fast as possible as this is single-handedly the most important aspect for good results. Also, a lot of code for extracting the data is available online. We failed to notice this at the start which cost us some time.
The causal graph process was implemented relatively late during the project, which can be further improved on.
The tuning of the LADMM solver used in the rLogSpecT model could still be improved for better performance. Alternatively, the SCS solver used in rSpecT could be applied, but this would result in a significantly longer runtime.
The best results for the method rSpecT were achieved for the smallest value of ϵ. ϵ is an upper bound for the distance between the solution of the algorithm called S, and the GSO that has the same eigenvectors as the covariance matrix called S'. This means that decreasing the value of ϵ decreases a measure of distance between the solution and the covariance matrix. This fact is further reflected by the covariance matrix itself reaching similar accuracies to rSpecT. In fact it is usually not the best option to choose ϵ as small as possible if no infeasibility issues are encountered. This is reflected by the convergence result listed in Appendix G.1 proposition 2. This states that convergence is only implied if ϵ is chosen big enough w.r.t. the distance between the minimum $l0$ norm solution of the problem and S'. Yet, evidently, we do not know the minimum $l0$ norm solution

of the problem, thus there is no certainty in choosing the value of ϵ. If we can assume that the data is generated by such a diffusion process, it is thus possible that choosing ϵ smaller brings us further away from the actual solution. But this is not reflected in the ML accuracies. This fact, combined with the fact that rSpecT reaches similar results to the covariance matrix, could imply that the model behind the data is inaccurate.
Additional tests were conducted, including the testing of different clustering and community detection methods. However, since these were implemented relatively late in the project, significant debugging and validation were still required, and the resulting data was ultimately not reliable. Nevertheless, the impact of different node detection methods can still be explored through the Graphical User Interface (GUI), making the inclusion of these tests less critical to the overall outcome.
The designed GUI is more technical than it is smooth, in terms of visual design or performance, though it has all the necessary features to be useful and fulfills all the requirements.

Appendices

A

Regions of Interest from the AAL brain atlas

Index	Region Name	Index	Region Name
1	Precentral_L	59	Calcarine_R
2	Precentral_R	60	Cuneus_R
3	Frontal_Sup_L	61	Lingual_R
4	Frontal_Sup_R	62	Occipital_Sup_R
5	Frontal_Sup_Orb_L	63	Occipital_Mid_R
6	Frontal_Sup_Orb_R	64	Occipital_Inf_R
7	Frontal_Mid_L	65	Fusiform_R
8	Frontal_Mid_R	66	Postcentral_R
9	Frontal_Mid_Orb_L	67	Parietal_Sup_R
10	Frontal_Mid_Orb_R	68	Parietal_Inf_R
11	Frontal_Inf_Oper_L	69	SupraMarginal_R
12	Frontal_Inf_Oper_R	70	Angular_R
13	Frontal_Inf_Tri_L	71	Precuneus_R
14	Frontal_Inf_Tri_R	72	Paracentral_Lobule_R
15	Frontal_Inf_Orb_L	73	Caudate_R
16	Frontal_Inf_Orb_R	74	Putamen_R
17	Rolandic_Oper_L	75	Pallidum_R
18	Rolandic_Oper_R	76	Thalamus_R
19	Supp_Motor_Area_L	77	Heschl_R
20	Supp_Motor_Area_R	78	Temporal_Sup_R
21	Olfactory_L	79	Temporal_Pole_Sup_R
22	Olfactory_R	80	Temporal_Mid_R

Index	Region Name	Index	Region Name
23	Frontal_Sup_Medial_L	81	Temporal_Pole_Mid_R
24	Frontal_Sup_Medial_R	82	Temporal_Inf_R
25	Frontal_Mid_Orb_Medial_L	83	Cerebellum_Crus1_R
26	Frontal_Mid_Orb_Medial_R	84	Cerebellum_Crus2_R
27	Rectus_L	85	Cerebellum_3_R
28	Rectus_R	86	Cerebellum_4_5_R
29	Insula_L	87	Cerebellum_6_R
30	Insula_R	88	Cerebellum_7b_R
31	Cingulum_Ant_L	89	Cerebellum_8_R
32	Cingulum_Ant_R	90	Cerebellum_9_R
33	Cingulum_Mid_L	91	Cerebellum_10_R
34	Cingulum_Mid_R	92	Vermis_1_2
35	Cingulum_Post_L	93	Vermis_3
36	Cingulum_Post_R	94	Vermis_4_5
37	Hippocampus_L	95	Vermis_6
38	Hippocampus_R	96	Vermis_7
39	ParaHippocampal_L	97	Vermis_8
40	ParaHippocampal_R	98	Vermis_9
41	Amygdala_L	99	Vermis_10
42	Amygdala_R	100	Cerebellum_Crus1_L
43	Calcarine_L	101	Cerebellum_Crus2_L
44	Cuneus_L	102	Cerebellum_3_L
45	Lingual_L	103	Cerebellum_4_5_L
46	Occipital_Sup_L	104	Cerebellum_6_L
47	Occipital_Mid_L	105	Cerebellum_7b_L
48	Occipital_Inf_L	106	Cerebellum_8_L
49	Fusiform_L	107	Cerebellum_9_L
50	Postcentral_L	108	Cerebellum_10_L
51	Parietal_Sup_L	109	Temporal_Sup_L
52	Parietal_Inf_L	110	Temporal_Pole_Sup_L
53	SupraMarginal_L	111	Temporal_Mid_L
54	Angular_L	112	Temporal_Pole_Mid_L
55	Precuneus_L	113	Temporal_Inf_L
56	Paracentral_Lobule_L	114	Caudate_L
57	Caudate_L	115	Putamen_L
58	Putamen_L	116	Pallidum_L

Table A.1: Table containing all the 116 regions of interest corresponding to the nodes 0-115 in the graphs from the AAL brain atlas.

B

Code & Data

B.1. Python code

All code used in this project is available on github, https://github.com/cxwchen/AutismDetection.

C

SOME EXTRA PROPERTIES

C.1. GSP PROPERTIES

Shift-invariant over discretized sampling domain (n-axis) implies that something is time-invariant. Where shift invariance if defined as

$$\mathbf{SH} = \mathbf{HS} \tag{C.1}$$

Convert some filter $H(z)$ in z-domain ($\mathbb{C}$) to a discrete set of values $h[n]$ in time using Discrete Fourier Transform (DFT), which is defined as

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi}{N}kn}, \quad k = 0, 1, \ldots, N-1 \tag{C.2}$$

The Z-transform is

$$X[k] = X(z)\Big|_{z=e^{j\frac{2\pi}{N}k}} = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn} \tag{C.3}$$

The z^{-1} term represents a shift of 1 in the time index (Δt). Using this notation, every discrete set of time samples can be represented as a polynomial (linear combination of shifts).
Inverse Discrete Fourier Transform (IDFT), which is defined as

$$x[n] = \frac{1}{N} \sum_{n=0}^{N-1} X[n] \cdot e^{j\frac{2\pi}{N}kn}, \quad k = 0, 1, \ldots, N-1 \tag{C.4}$$

A system that interacts on a signal x_i is described by some transfer function/filter H. Using the z-transform, this filter H can be written as a linear combination of shifts. A finite impulse response filter FIR can be written as,

$$h(z) = \sum_{n=0}^{N-1} h_n z^{-n} \tag{C.5}$$

As the system evolves over time, x_i is expressed as a vector indexed over time. This means that H will be a matrix. As this matrix is a combination of shifts, it should have some corresponding graph.

The spectral modes (or graph Fourier modes) are the eigenvectors $\{\phi_k\}_{k=1}^{|\mathcal{V}|}$ of the graph Laplacian L, obtained by solving

$$L\phi_k = \lambda_k \phi_k, \quad \text{with } \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{|\mathcal{V}|}$$

where

- λ_k are the eigenvalues (graph frequencies),
- ϕ_k are the orthonormal eigenvectors (spectral modes).

Extra properties:

1. High edge weight between 2 vertices -> high correlation between the vertices -> similar valued node signals x_i
2. Frequency is proportional to variation of the graph
3. if filter can be diagonalized by graph fourier transform (GFT) -> filter = shift invariant
4. Neighborhood in graph scales for n-hop as: xS^n
5. DFT assumes signals are periodic -> force periodicity by looping real time axis
6. positive semi-definite -> real positive eigenvalues -> can 'naturally' order systems basis vectors (=eigenvectors =node vector x_i in GFT) in terms of frequency

D

AAL GRAPH EXTRACTION VISUALIZATION

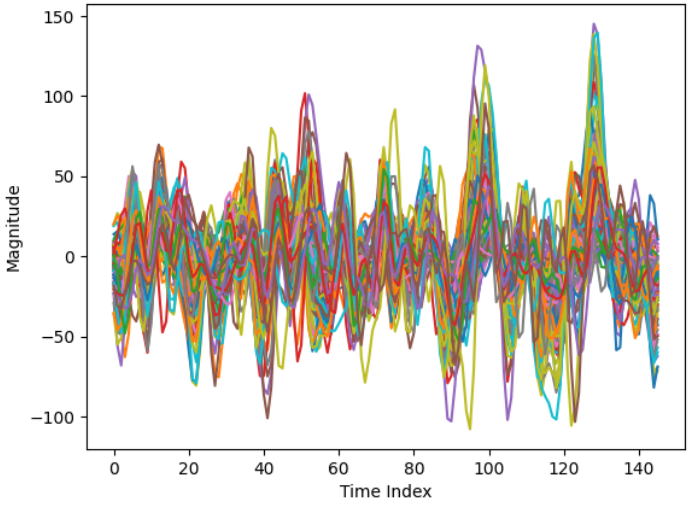


Figure D.1: The timeseries plotted for 116 different ROIs following the AAL brain atlas for individual 0051461. Every color corresponds to a different ROI.

D.1. PEARSON'S CORRELATION

From the definition of Pearson's Correlation in equation [3.15], the following adjacency matrix can be computed.

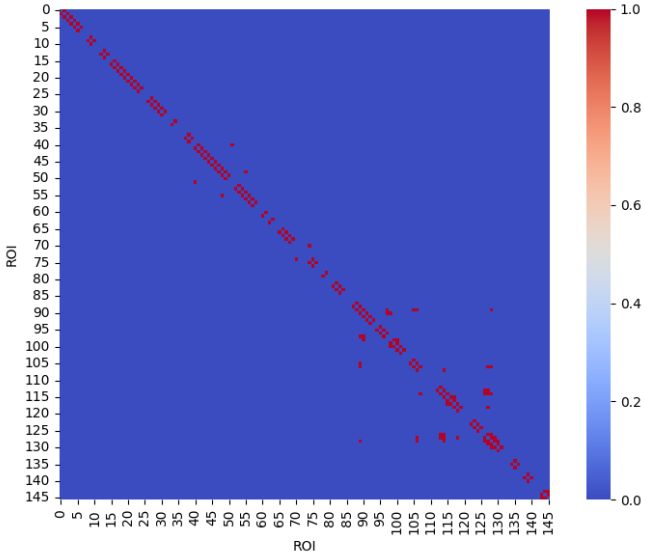


Figure D.2: Pearson's Correlation adjacency matrix

Now using this and Pearson's correlation, a graph can be estimated. Then use thresh-

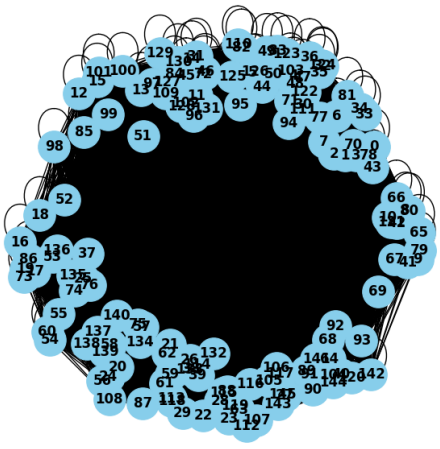


Figure D.3: Graph estimated from the time series in Figure [D.1] using Pearson's correlation coefficient without thresholding.

olding and remove self-loops to create a sparser adjacency matrix.

The found correlations are between 4-1, which corresponds to $Community5 : [0,1,2,3,4,5,6]$ and 8-0-2-9, which corresponds to $Community13 : [8,9,10]$ and $Community5 : [0,1,2,3,4,5,6]$.

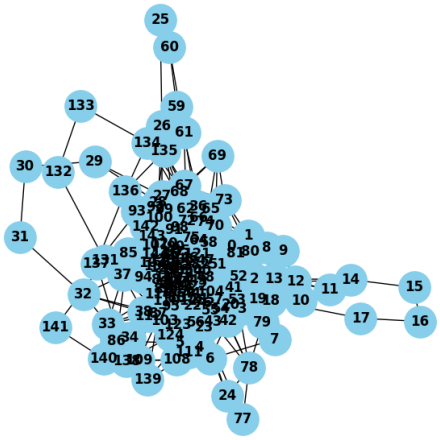


Figure D.4: Graph estimated from the time series in Figure [D.1] using Pearson's correlation coefficient with binary thresholding with a threshold value of 0.5

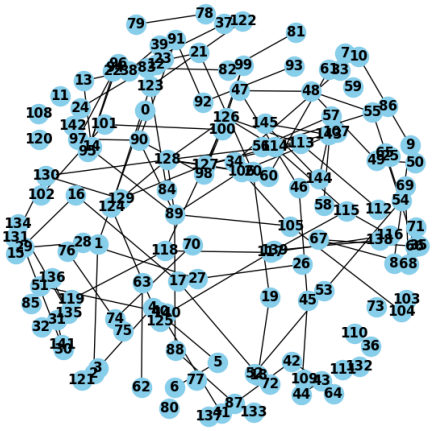


Figure D.5: Graph estimated from the time series in Figure [D.1] using Pearson's correlation coefficient with binary thresholding with a threshold value of 0.8

D.2. MUTUAL INFORMATION

Using the definition of mutual information in [3.17] and the sklearn function

'mutual_information_regression', the following connectivity matrix is obtained.

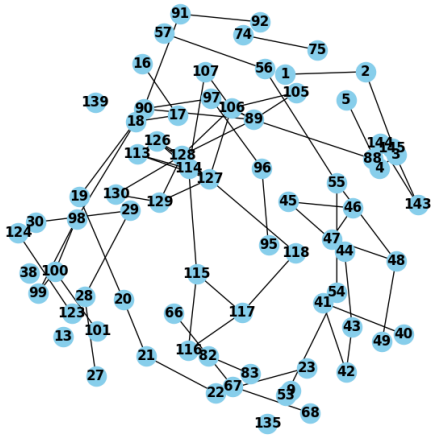


Figure D.6: Graph estimated from the time series in Figure [D.1] using Pearson's correlation coefficient with binary thresholding with a threshold value of 0.8 and removing all nodes with a degree less than 2.

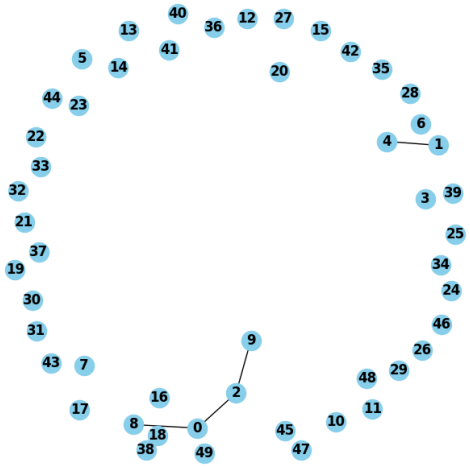


Figure D.7: This graph uses the graph in [D.6] and clusters communities of nodes into 'supernodes' to show the connections between these communities.

D.3. MRF PROCESS EXAMPLE - DIRECT NEIGHBOR GRAPH ESTIMATE

It is possible to estimate the Adjacency matrix using a method called k-Nearest Neighbors (k-NN)[ref]. For simplicity, we assume k=1 to consider the 1-hop neighborhood along set amount of axes, for example an along the x- and y-axis of a 2D image.

The resulting graph structure can be interpreted as a Markov Random Field (MRF) when

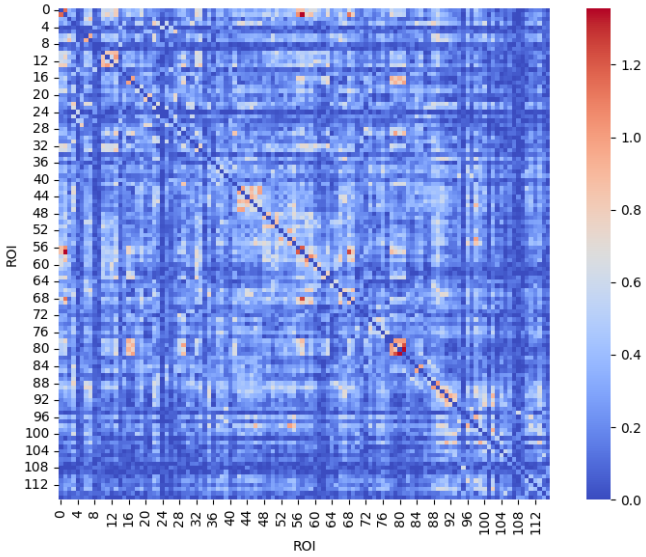


Figure D.8: Mutual information adjacency matrix

modeling spatial dependencies, or as part of an Autoregressive (AR) model when modeling temporal dependencies.

$$x_{i,j} = \sum_{(m,n)\in\mathcal{N}(i,j)} A_{i,j}^{m,n} x_{m,n} + w_{i,j} \tag{D.1}$$

where:

- $x_{i,j}$: Pixel value of some 2D image at location x=i (row), y=j (column)
- $\mathcal{N}(i,j)$: Neighbors of (i,j)
- $A_{i,j}^{m,n}$: Weights modeling influence of neighbor (m,n) on (i,j)
- $w_{i,j}$: Noise term, assume white Gaussian noise

For a 4-connected neighborhood, the neighbors of some pixel at index i,j can be written as:

$$\mathcal{N}(i,j) = \{(i-1,j),(i+1,j),(i,j-1),(i,j+1)\}$$

The weight matrix is random walk normalized[ref]

$$W = D^{-1}A \tag{D.2}$$

The solution to some MRF as the one in [E.5]

$$x = (I-W)^{-1}n \tag{D.3}$$

By generating some random 10x10 binary image, the following graph [fig] is obtained. The binary image, and the graph that was estimated from it.

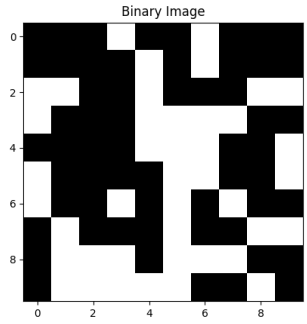


Figure D.9: 10 by 10 pixel randomly generated binary image

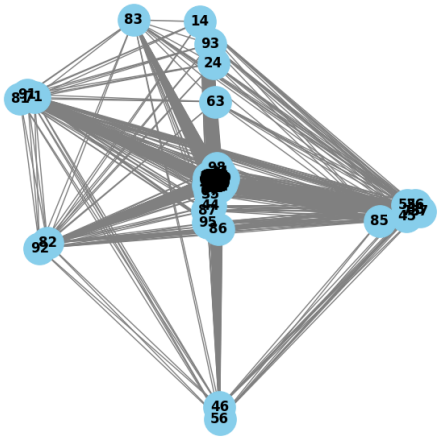


Figure D.10: Graph estimated from random binary image in Fig[D.9] using 4-hop neighborhood along x and y axes

E

MATHEMATICAL MODELS

E.1. LAPLACIAN OPERATION & CONTEXT

rs-fMRI represents brain activity using the flow of blood inside the brain. Divergence is a representation of the flow contributions relative to some point/region.
The Laplacian operator is defined as

$$\nabla^2 = \nabla \cdot \nabla = \sum_i \frac{\partial^2}{\partial x_i^2} \tag{E.1}$$

It represents the divergence of the gradient, which can be applied to a function.
The edge of some region(surface in $\mathbb{R}^2$) can be defined as the boundary of that region where descent/derivative is maximal (locally), which is modeled by the gradient ∇. Therefore, the Laplacian gives information about the smoothness of some region (in space) and can detect regional outliers or clusters in some domain (data). A graph can be defined as a discrete set of nodes x_i, each with some set of parameters. Thus, every node can be seen as an individual function. The graph then corresponds to a combination of these functions where the nodes could have some covariance σ_{ij}.
The discrete Laplace Operator **L**, also Graph Laplacian, is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{W} \tag{E.2}$$

Where $\mathbf{D} = trace(\mathbf{W})$ is the degree matrix and **W** the adjacency matrix of some discrete graph.
This Laplacian represents a matrix form of the negative discrete Laplace operator on a graph approximating the negative continuous Laplacian mentioned above. The Laplacian can be viewed as a kind of filter or kernel. It can be used to smooth out data points and to find edges of some cluster.
Spectral clustering is done by computing the laplacian matrix, then computing the eigenvectors and -values of **L** using *linalg.eigh* from the numpy python library.
A node with a large degree can dominate properties of the Laplacian matrix. To avoid

this we can compute the normalized Laplacian matrix,

$$L_n := D^{-1/2} L D^{-1/2} = I D^{-1/2} W D^{-1/2}$$

E.2. LINEAR MODEL

$$y = Ax + n \tag{E.3}$$

Where x is input and y is output.

E.3. DIFFUSION PROCESS CAUSAL GRAPH PROCESS

This process is characterized by a diffusion of information over time. This diffusion is the netto diffusion relative to some spacial domain x_t

$$x_{t+1} = Ax_t + w \tag{E.4}$$

Here A could also be replaced by $f(A)$, and w indicates some noise component.

E.4. MARKOV RANDOM FIELD

A random field is a generalized stochastic (random) process that can take any dimensional input and resides in domain $\mathbb{R}^n$. A MRF is a random field characterized by the Markov Property, which basically says any node in the graph is independent of any previous states in time (=memoryless).

$$x_t = Ax_t + w \tag{E.5}$$

E.5. STATIONARY GRAPH PROCESS

$$x = \sum_{l=0} h_l S^l w = Hw \tag{E.6}$$

$$C_x = \mathbb{E}[xx^H] \tag{E.7}$$

E.6. AUTOREGRESSIVE (AR) PROCESS

The autoregression is of order k.

$$x_t = \rho x_{t-k} + w \tag{E.8}$$

Here the ρ represents the magnitude of the time-dependence and thus the autocorrelation of x_t. This is a univariate and linear representation, the x consist of a single element. The model could have any amount of lagged components x_{t-a}.

E.7. VECTOR AUTOREGRESSIVE (VAR) PROCESS

The VAR process follows the same structure as the AR process [E.9], however the signal x represents a vector instead of a single element.

E.8. NONLINEAR VECTOR AUTOREGRESSIVE (NVAR) PROCESS

The NVAR process is any VAR process that is nonlinear. An example could be,

$$x_t = Ax_{t-a} \otimes Bx_{t-b} \tag{E.9}$$

Here the process in nonlinear as it includes any superposition on the lag components of x.

E.9. WHITE NOISE

$$w \sim \mathcal{N}(0, \sigma^2) \tag{E.10}$$

$$C_w = \mathbb{E}[ww^H] = \sigma^2 I \tag{E.11}$$

F

Designed Features

All statistical features are computed per timeseries.

F.1. Statistical Features

- Mean Signal-to-Noise Ratio (SNR), measures the quality of the time series signal for each ROI.
- Skewness, measures the asymmetry of the time series distribution for each ROI.
- Kurtosis, measures the tailedness of the time series distribution for each ROI.
- Mean, the average value of the time series for each ROI.
- Standard deviation (std), measures the variability of the time series for each ROI.

F.2. Graph Features

F.2.1. Node specific features

- **In-degree**: Number of incoming edges to a node in directed graphs:

$$k_i^{\text{in}} = \sum_{j=1}^{N} A_{ji}$$

- **Out-degree**: Number of outgoing edges from a node in directed graphs:

$$k_i^{\text{out}} = \sum_{j=1}^{N} A_{ij}$$

- **Degree Centrality**: Number of edges connected to a node. For a node i:

$$k_i = \sum_{j=1}^{N} A_{ij}$$

where A_{ij} is the adjacency matrix (binary or weighted).

- **Closeness Centrality**: Inverse of the average shortest path length from node i to all others:
$$C_i = \frac{N-1}{\sum_{j \neq i} d_{ij}}$$
where d_{ij} is the shortest path length between nodes i and j.

- **Eigenvector Centrality**: Measures influence based on neighbors' centrality. Computed from the dominant eigenvector $\mathbf{v}$ of the adjacency matrix:
$$\mathbf{v} = \lambda^{-1} A\mathbf{v}$$
where λ is the largest eigenvalue.

- **Clustering Coefficient**: Local connectivity density. For node i with degree k_i:
$$c_i = \frac{2T_i}{k_i(k_i-1)}$$
where T_i is the number of triangles around i.

- **Participation Coefficient**: Measures distribution of connections across modules. For node i in modules m_s:
$$PC_i = 1 - \sum_{s=1}^{N_m} \left(\frac{k_{i,s}}{k_i}\right)^2$$
where $k_{i,s}$ is the strength of connections between i and module s.

- **Between-module Strength**: Sum of edge weights from node i to other modules:
$$S_i^{\text{between}} = \sum_{j \notin m_i} A_{ij}$$
where m_i is i's module.

- **Within-module Strength**: Sum of edge weights from node i to its own module:
$$S_i^{\text{within}} = \sum_{j \in m_i} A_{ij}$$

- **Spectral Cluster**: Community label for node i from spectral clustering. Computed via:

  1. Compute the graph Laplacian $L = D - A$ where D is the degree matrix and A is the adjacency matrix
  2. Compute the first k eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ of L corresponding to the smallest eigenvalues
  3. Form the spectral embedding matrix $V \in \mathbb{R}^{n \times k}$ with eigenvectors as columns

4. Apply k-means clustering to the rows of V:

$$\text{Cluster}_i = \text{k-means}(V_{i:}, k)$$

where $V_{i:}$ is the i-th row of V (spectral coordinates of node i)

- **Ward Cluster**: Community label for node i from Ward's hierarchical clustering. Computed via:

1. Initialize each node as its own cluster
2. At each step, merge the pair of clusters (C_a, C_b) that minimizes the Ward criterion:

$$\Delta(C_a, C_b) = \frac{|C_a||C_b|}{|C_a| + |C_b|} \|\mu_a - \mu_b\|^2$$

where μ_a, μ_b are cluster centroids and $|C|$ is cluster size

3. Continue until k clusters remain
4. The final assignment Cluster_i is the leaf cluster containing node i

- **Betweenness Centrality**: Fraction of shortest paths passing through node i:

$$B_i = \sum_{j \neq k \neq i} \frac{\sigma_{jk}(i)}{\sigma_{jk}}$$

where σ_{jk} is the total number of shortest paths between j and k, and $\sigma_{jk}(i)$ is the number passing through i.

F.2.2. NODE TO NODE

Node to node features are all the features that consist of more than one node. So for example, the weights of the edges can be used as a feature. This feature then corresponds to the two nodes/vertices that are connected by this edge.

- **Edge weights** (Connectivity matrix elements), represented as $A_{i,j}$ where i and j are node indices. These capture the strength of functional connectivity between pairs of brain regions.
- **Pearson's/full** correlation between two regions (is not really a graph specific feature).

F.2.3. GLOBAL FEATURES

Global features indicate that the relative feature is the same over the whole graph and thus not node specific. These could be for instance obtained by taking the average or maximum value of the list of node specific features of a graph.

- **Average Clustering**, this is the average of the clustering coefficient over the complete graph.

- **Spectral Entropy**: Measures the complexity of the graph's spectral properties. Computed using the Shannon entropy of the normalized eigenvalue distribution:

$$H = -\sum_{i=1}^{n} \frac{\lambda_i}{\sum_j \lambda_j} \log\left(\frac{\lambda_i}{\sum_j \lambda_j}\right)$$

where n is the number of nodes and λ_i are the eigenvalues of the adjacency matrix.

- **Mean Laplacian Eigenvalue**: Average eigenvalue of the graph Laplacian matrix L:

$$\bar{\lambda}_L = \frac{1}{n}\sum_{i=1}^{n} \lambda_i(L)$$

where $\lambda_i(L)$ are the eigenvalues of L.

- **Max Laplacian Eigenvalue**: Maximum eigenvalue of the graph Laplacian matrix:

$$\lambda_{\max}(L) = \max_i \lambda_i(L)$$

- **Frobenius Norm (Laplacian Spectrum)**: Magnitude of the Laplacian eigenvalue spectrum:

$$\|L\|_F = \sqrt{\sum_{i=1}^{n} \lambda_i^2(L)}$$

- **Algebraic Connectivity** (λ_2): Second smallest eigenvalue of the Laplacian:

$$\lambda_2(L)$$

Measures how well-connected the graph is.

- **Graph Energy**: Sum of absolute values of adjacency matrix eigenvalues:

$$E(G) = \sum_{i=1}^{n} |\lambda_i(A)|$$

where A is the adjacency matrix and G is the graph.

- **Asymmetry Index**: Measures directional asymmetry in the graph structure:

$$AI = \frac{\|A - A^T\|_F}{\|A\|_F}$$

where A^T is the transpose of the adjacency matrix.

- **Global Efficiency**: Efficiency of information transfer across the network:

$$\eta = \frac{1}{n(n-1)} \sum_{i \neq j} \frac{1}{d_{ij}}$$

where d_{ij} is the shortest path length between nodes i and j.

- **Characteristic Path Length**: Average shortest path length:

$$L = \frac{1}{n(n-1)} \sum_{i \neq j} d_{ij}$$

- **Small-worldness**: Balance between clustering and path length:

$$\sigma = \frac{C/C_{\text{rand}}}{L/L_{\text{rand}}}$$

  where C is clustering coefficient and L is path length, compared to random networks.

- **Modularity**: Strength of network division into modules:

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

  where m is total edges, k_i is degree of node i, and δ is Kronecker delta for community membership.

- **Ward Silhouette**: Average silhouette score for Ward clustering:

$$S = \frac{1}{n} \sum_{i=1}^{n} \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

  where $a(i)$ is within-cluster distance, $b(i)$ is nearest-cluster distance.

- **Spectral CH Score**: Calinski-Harabasz index for spectral clustering:

$$CH = \frac{\text{Between-cluster SS}/(k-1)}{\text{Within-cluster SS}/(n-k)}$$

  where SS is sum of squares, k is number of clusters.

- **Cluster Separation**: Measures inter-cluster distinctness:

$$\text{Sep} = \min_{i \neq j} \|\mu_i - \mu_j\|^2$$

  where μ_i is centroid of cluster i.

- **Mean Degree Centrality**, average degree centrality across all nodes.
- **Variance Degree Centrality**, variance of degree centrality across all nodes.
- **Mean Closeness Centrality**, average closeness centrality across all nodes.
- **Variance Closeness Centrality**, variance of closeness centrality across all nodes.
- **Mean Eigenvector Centrality**, average eigenvector centrality across all nodes.

- **Variance Eigenvector Centrality**, variance of eigenvector centrality across all nodes.
- **Mean Participation Coefficient**, average participation coefficient across all nodes.
- **Variance Participation Coefficient**, variance of participation coefficient across all nodes.
- **Mean In-Degree**, average in-degree across all nodes.
- **Variance In-Degree**, variance of in-degree across all nodes.
- **Mean Out-Degree**, average out-degree across all nodes.
- **Variance Out-Degree**, variance of out-degree across all nodes.

F.2.4. FEATURE SELECTION

The following tables show the selected features for classification from table 4.5.

Table F.1: Selected Statistical Features

Index	Feature Name
4	SNR_ROI_4
13	SNR_ROI_13
16	SNR_ROI_16
19	SNR_ROI_19
22	Skewness_ROI_22
23	Skewness_ROI_23
28	Kurtosis_ROI_28
29	Kurtosis_ROI_29
33	mean_ROI_33
39	mean_ROI_39
57	std_ROI_57
64	std_ROI_64
67	std_ROI_67
73	std_ROI_73
76	std_ROI_76
79	std_ROI_79
86	std_ROI_86
87	std_ROI_87
88	std_ROI_88
98	std_ROI_98

Table F.2: Selected Node-to-Node Features

Index	Feature Name
18	A_18_19
20	A_20_21
21	A_21_22
25	A_25_26
37	A_37_38
47	A_47_48
53	A_53_54
57	A_57_58
59	A_59_60
60	A_60_61
67	A_67_68
100	A_100_101
105	A_105_106
115	A_115_116
119	A_1_119
141	A_25_141
142	A_26_142
167	A_50_167
170	A_53_170
179	A_61_179

Table F.3: Selected Global Graph Features

Index	Feature Name
0	Average Clustering
2	Spectral Entropy
3	Mean Laplacian Eigenvalue
4	Max Laplacian Eigenvalue
5	Frobenius Norm (Laplacian Spectrum)
6	Connectivity ()
7	Graph Energy
8	Asymmetry Index
9	Global Efficiency
10	Characteristic Path Length
12	Smallworldness
14	Mean Degree Centrality
15	Var Degree Centrality
16	Mean Closeness Centrality
17	Var Closeness Centrality
18	Mean Eigenvector Centrality
19	Var Eigenvector Centrality
20	Mean Participation Coefficient
22	Mean In-Degree
24	Mean Out-Degree

Table F.4: Selected Node Graph Features

Index	Feature Name
27	Between-module Strength_ROI_27
28	Between-module Strength_ROI_28
31	Closeness Centrality_ROI_31
41	Closeness Centrality_ROI_41
51	Clustering Coefficient_ROI_51
52	Clustering Coefficient_ROI_52
66	Degree Centrality_ROI_66
67	Degree Centrality_ROI_67
68	Eigenvector Centrality_ROI_68
71	Eigenvector Centrality_ROI_71
86	In-degree_ROI_86
87	In-degree_ROI_87
88	Out-degree_ROI_88
91	Out-degree_ROI_91
118	Participation Coefficient_ROI_118
138	Participation Coefficient_ROI_138
185	Within-module Strength_ROI_185
193	Within-module Strength_ROI_193
195	Spectral Cluster_ROI_195
206	Ward Cluster_ROI_206

Table F.5: Selected Node Cluster Features

Index	Feature Name
1	Spectral Cluster_ROI_1
3	Spectral Cluster_ROI_3
5	Spectral Cluster_ROI_5
7	Spectral Cluster_ROI_7
11	Spectral Cluster_ROI_11
13	Spectral Cluster_ROI_13
14	Spectral Cluster_ROI_14
15	Spectral Cluster_ROI_15
16	Spectral Cluster_ROI_16
17	Spectral Cluster_ROI_17
20	Ward Cluster_ROI_20
21	Ward Cluster_ROI_21
25	Ward Cluster_ROI_25
26	Ward Cluster_ROI_26
27	Ward Cluster_ROI_27
29	Ward Cluster_ROI_29
31	Ward Cluster_ROI_31
32	Ward Cluster_ROI_32
34	Ward Cluster_ROI_34
39	Ward Cluster_ROI_39

Table F.6: Selected Graph-Based Features

Index	Feature Name
52	Clustering Coefficient_ROI_52
53	Clustering Coefficient_ROI_53
66	Degree Centrality_ROI_66
76	Eigenvector Centrality_ROI_76
77	Eigenvector Centrality_ROI_77
92	In-degree_ROI_92
93	In-degree_ROI_93
111	Participation Coefficient_ROI_111
112	Participation Coefficient_ROI_112
113	Participation Coefficient_ROI_113
116	Within-module Strength_ROI_116
143	Within-module Strength_ROI_143
163	Spectral Cluster_ROI_163
210	Ward Cluster_ROI_210
218	Average Clustering
231	Graph Energy
250	Global Efficiency
258	Smallworldness
260	Modularity
271	Ward Silhouette

G

RECOVERY THEOREMS AND RESULTS SPECTRAL TEMPLATES

G.1. RECOVERY GUARANTEES OF RSPECT

Proposition 2 in [44].
When $d(\mathbf{S},\mathbf{S}') = \|\mathbf{S}-\mathbf{S}'\|_F$, and assuming that there exists at least one $\mathbf{S}'$ such that $\tilde{d}(\mathbf{S}_0^*,\mathbf{S}') \leq \epsilon$, the solution $\hat{\mathbf{s}}_1^* := \text{vec}(\hat{\mathbf{S}}_1^*)$ to 3.20 for $\mathscr{S} = \tilde{\mathscr{S}}_\text{A}$ with scale ambiguity satisfies

$$\|\hat{\mathbf{s}}_1^* - \mathbf{s}_0^*\|_1 \leq C\epsilon, \quad \text{with} \quad C = 2C_1 + 2C_2C_3$$

if the same conditions stated in Theorem 1 hold but for $\hat{\mathbf{R}}$ instead of $\mathbf{R}$. Constants C_1, C_2, and C_3 are given by

$$C_1 = \frac{\sqrt{|\mathscr{K}|}}{\sigma_{\min}(\hat{\mathbf{R}}_{\mathscr{K}}^T)}, C_2 = \frac{1+\|\hat{\mathbf{R}}^T\|_{M(2)} C_1}{1-\psi_{\hat{\mathbf{R}}}}, C_3 = \|\hat{\mathbf{R}}^\dagger\|_{M(2)} N$$

where $\sigma_{\min}(\cdot)$ denotes the minimum singular value of the argument matrix. An analogous result can be derived for the case $\mathscr{S} = \tilde{\mathscr{S}}_\text{L}$ (where the non-positivity constraint is dropped) whenever $\hat{\mathbf{Q}}$ satisfies the conditions in Theorem 2.

G.2. RECOVERY AND FEASIBILITY GUARANTEES OF RLOGSPECT

Theorem 3.2 in [47]
Consider the linear system

$$\boldsymbol{A}_n\boldsymbol{B}\boldsymbol{y} = \mathbf{0}, \quad \boldsymbol{y} \leq \mathbf{0}, \quad \sum_{i=1}^{m-1} y_i \neq \mathbf{0} \tag{G.1}$$

where $\boldsymbol{y} \in \mathbb{R}^{\frac{m(m-1)}{2}}$. If G.1 has no feasible solution, then there exists a $\bar{\delta}_n > 0$ such that r SpecT is infeasible for all $\delta_n \in [0, \bar{\delta}_n)$.

Remark 3.3 From Theorem 3.2 we can infer that for any fixed n, the linear system G.1 has no solution when $\boldsymbol{A}_n\boldsymbol{B}$ has full column rank (e.g., Example 3.1). This leads to the infeasibility of rSpecT with $\delta_n \in [0, \delta_n)$. As remarked in [44], one should tackle the feasibility issue of rSpecT with caution.

Proposition 3.4 in [47]
Let $\mathrm{Opt}(\boldsymbol{C}, \alpha)$ be the optimal solution set of LogSpecT with input covariance matrix $\boldsymbol{C}$ and parameter $\alpha > 0$. Then, for any $\gamma > 0$, it follows that

$$\mathrm{Opt}(\boldsymbol{C}, \alpha) = \gamma\, \mathrm{Opt}(\boldsymbol{C}, \alpha/\gamma) = \alpha\, \mathrm{Opt}(\boldsymbol{C}, 1)$$

Proposition 3.6 in [47]
For $\delta_n > 0$ and $\boldsymbol{C}_n$ with any fixed n, rLogSpecT always has a nontrivial feasible solution.

Corollary 4.4 in [47]
Under the assumptions in Theorem 4.1, it follows that if $\delta_n \to 0$, then

$$\lim_{n\to\infty} \mathrm{dist}\left(\mathscr{S}^{n,*}, \mathscr{S}_0^*\right) = 0 \tag{G.2}$$

Remark 4.5 in [47]
From the strong law of large numbers, if we choose $\delta_n = \mathscr{O}(\|\boldsymbol{C}_n - \boldsymbol{C}_\infty\|)$, then almost surely one has $\lim_{n\to\infty} \|\boldsymbol{C}_n - \boldsymbol{C}_\infty\| = 0$ and consequently $\delta_n \to 0$. Hence, Corollary 4.4 shows that G.2 holds almost surely.

Definition 4.6 (Sub-Gaussian Distributions) in [47]
The probability distribution of a random vector $\boldsymbol{w}$ is called sub-Gaussian if there are $C, v > 0$ such that for every $t > 0, \mathbb{P}(\|\boldsymbol{w}\|_2 > t) \le Ce^{-vt^2}$.

Lemma 4.7 in [47]
Suppose that $\boldsymbol{w}$ in the generative model 3.19 follows a sub-Gaussian distribution. Then, $\boldsymbol{x}$ follows a sub-Gaussian distribution, and with probability larger than $1 - \mathscr{O}\left(\frac{1}{n}\right)$, one has $\|C_n - C_\infty\| \le \mathscr{O}(\sqrt{\log n/n})$.

Corollary 4.8 in [47]
If the input $\boldsymbol{w}$ in 3.19 follows a sub-Gaussian distribution and $\delta_n = \mathscr{O}(\sqrt{\log n/n})$, then the assumptions in Theorem 4.1 hold with probability larger than $1 - \mathscr{O}\left(\frac{1}{n}\right)$.

G.3. RESULTS

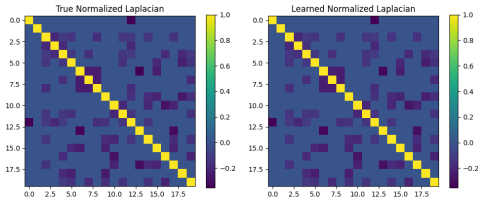


Figure G.1: rSpecT on an ER graph with $\mathscr{S} = \mathscr{S}_L$ with 10000 samples compared to the original graph

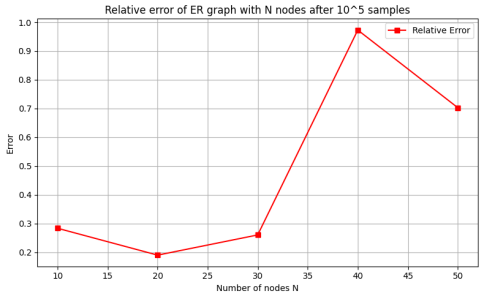


Figure G.2: rSpecT on an ER graph with a varying number of nodes, with $\mathscr{S} = \mathscr{S}_L$ and 10000 samples, compared to the original graph

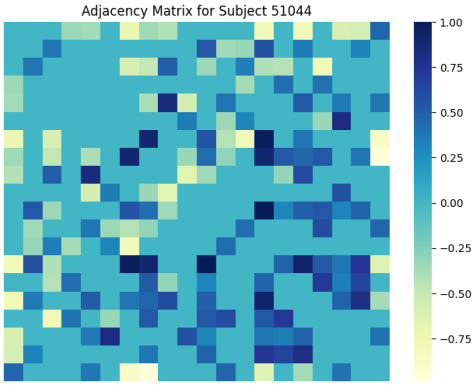


Figure G.3: rSpecT with $\mathscr{S} = \mathscr{S}_L$ on subject 51044

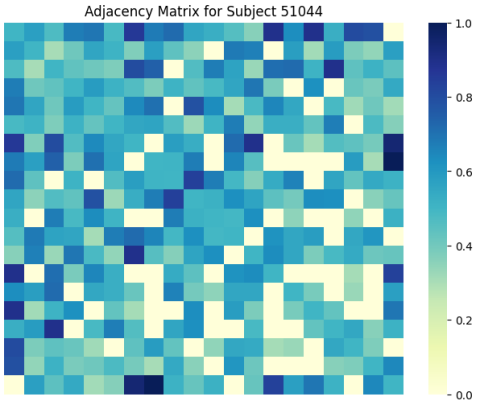


Figure G.4: rSpecT with $\mathscr{S} = \mathscr{S}_A$ on subject 51044

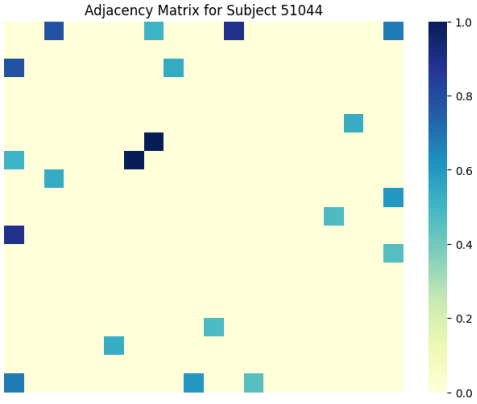
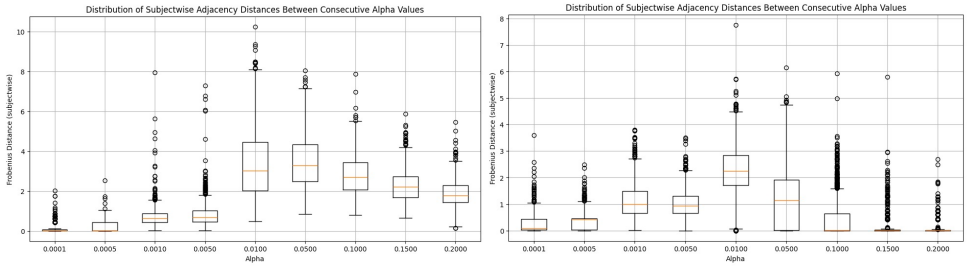


Figure G.5: rLogSpecT on subject 51044



(a) $\mathscr{S} = \mathscr{S}_L$ (b) $\mathscr{S} = \mathscr{S}_A$

Figure G.6: Distribution of the subjectwise matrix distance for consecutive alpha values on rSpecT

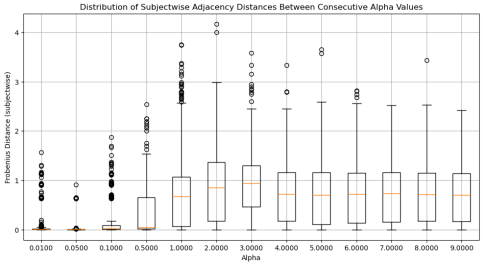


Figure G.7: Distribution of the subjectwise matrix distance for consecutive alpha values on rLogSpecT

H

Derivation for Causal Graph Process

The Causal Graph Process (CGP) models the temporal evolution of a multivariate signal $\mathbf{x}[k] \in \mathbb{R}^N$ as:

$$\mathbf{x}[k] = \mathbf{w}[k] + \sum_{i=1}^{M} P_i(\mathbf{A};\mathbf{c})\,\mathbf{x}[k-i] \tag{H.1}$$

where $\mathbf{w}[k]$ is a zero-mean innovation noise vector, and $P_i(\mathbf{A};\mathbf{c})$ is a matrix polynomial defined as:

$$P_i(\mathbf{A};\mathbf{c}) = \sum_{j=0}^{i} c_{i,j}\,\mathbf{A}^j \tag{H.2}$$

Model Residual: For each time step k, the model residual (prediction error) is defined as:

$$\mathbf{r}[k] = \mathbf{x}[k] - \sum_{i=1}^{M} P_i(\mathbf{A};\mathbf{c})\,\mathbf{x}[k-i] \tag{H.3}$$

Data Fitting Term: The overall fitting cost is the sum of squared residuals over all time steps $k = M,\ldots,K-1$:

$$J_{\text{fit}}(\mathbf{A},\mathbf{c}) = \frac{1}{2}\sum_{k=M}^{K-1}\left\|\mathbf{x}[k] - \sum_{i=1}^{M} P_i(\mathbf{A};\mathbf{c})\,\mathbf{x}[k-i]\right\|_2^2 \tag{H.4}$$

Regularization Terms: To promote sparsity in the learned graph and polynomial coefficients, ℓ_1 penalties are added:

- Sparsity on $\mathbf{A}$:

$$J_{\mathbf{A}}(\mathbf{A}) = \lambda_1 \|\text{vec}(\mathbf{A})\|_1 \tag{H.5}$$

- Sparsity on $\mathbf{c}$:

$$J_{\mathbf{c}}(\mathbf{c}) = \lambda_2 \|\mathbf{c}\|_1 \tag{H.6}$$

Full Objective Function: Combining the data fitting term and the regularization terms, the final optimization problem is:

$$\min_{\mathbf{A},\mathbf{c}} \frac{1}{2} \sum_{k=M}^{K-1} \left\| \mathbf{x}[k] - \sum_{i=1}^{M} \left(\sum_{j=0}^{i} c_{i,j} \mathbf{A}^j \right) \mathbf{x}[k-i] \right\|_2^2 + \lambda_1 \|\text{vec}(\mathbf{A})\|_1 + \lambda_2 \|\mathbf{c}\|_1 \tag{H.7}$$

Summary: This objective seeks to find an adjacency matrix $\mathbf{A}$ (graph topology) and polynomial coefficients $\mathbf{c}$ (temporal model) that:

- Accurately model the temporal dynamics of the data $\mathbf{x}[k]$
- Promote sparse solutions for both $\mathbf{A}$ and $\mathbf{c}$ for interpretability and robustness

I

STATISTICAL TESTING

I.1. AUTOCORRELATION PLOTS

For a time series x_t at lag τ, the autocorrelation function (ACF) is defined as,

$$R_{x(t),x(t+\tau)} = \rho(k) = \frac{\mathbb{E}[(x_t - \mu)(x_{t+\tau} - \mu)]}{\sigma^2} \tag{I.1}$$

where μ and σ^2 are the mean and variance of the process. The autocorrelation is the normalized autocovariance:

$$R_{xx} = \frac{C_{xx}}{\sigma_x^2} \tag{I.2}$$

This means that to prove ergodicity, proving that R depends only on the lag τ is equivalent to proving that C depends only on the lag.
If there is significant autocorrelation in the time series, the samples can no longer be considered independent and identically distributed (i.i.d.), which may affect statistical assumptions in downstream analyses such as ICA. Autocorrelation inflates variance and can bias covariance estimates, particularly in high-dimensional or model-based analyses. Although shuffling time points can artificially enforce the i.i.d. assumption, shuffling time points can remove temporal dependencies and may not be appropriate for fMRI data, where temporal dynamics carry meaningful information.

Here are several ACF plots across different ROIs from the AAL atlas.

Here are several ACF plots for different ICA components.

I.2. ERGODICITY PLOTS

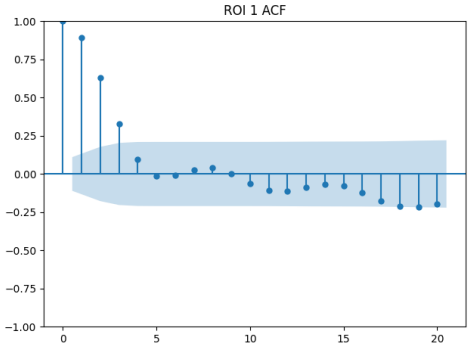


Figure I.1: Autocorrelation function for ROI-1 over the number of lags, where the horizontal axis represents the time index (number of lags) and the vertical axis the amplitude of the timeseries of ROI-1

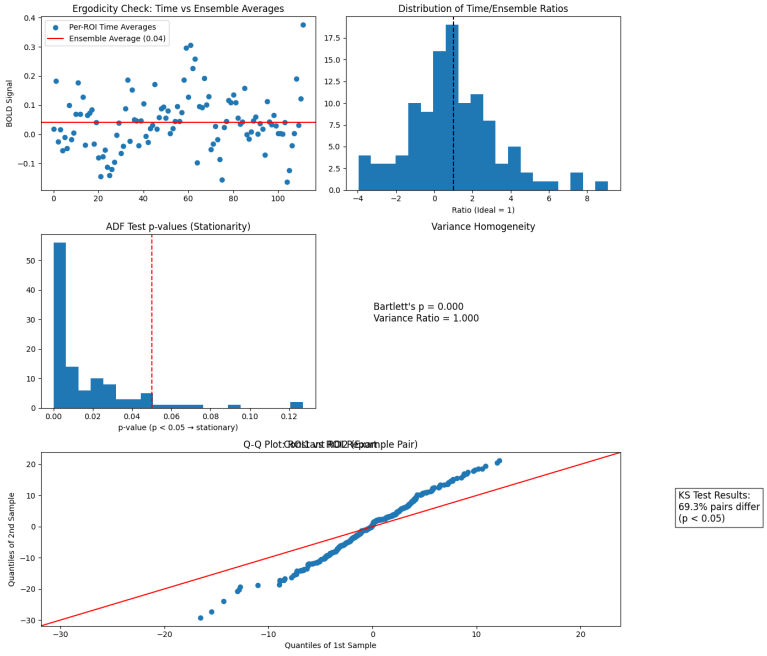


Figure I.2: Datasheet displaying the ergodicity conditions

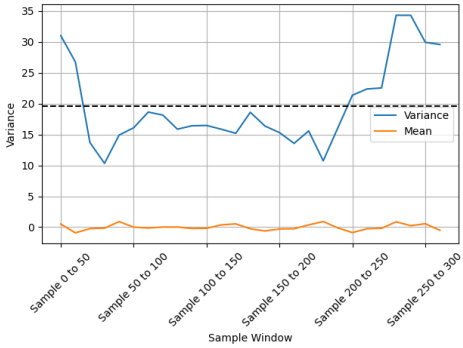


Figure I.3: Graph showing the mean and variance using a sliding window to check for WSS conditions. The sliding window is applied per ROI and then averaged over the ROIs to obtain these graphs

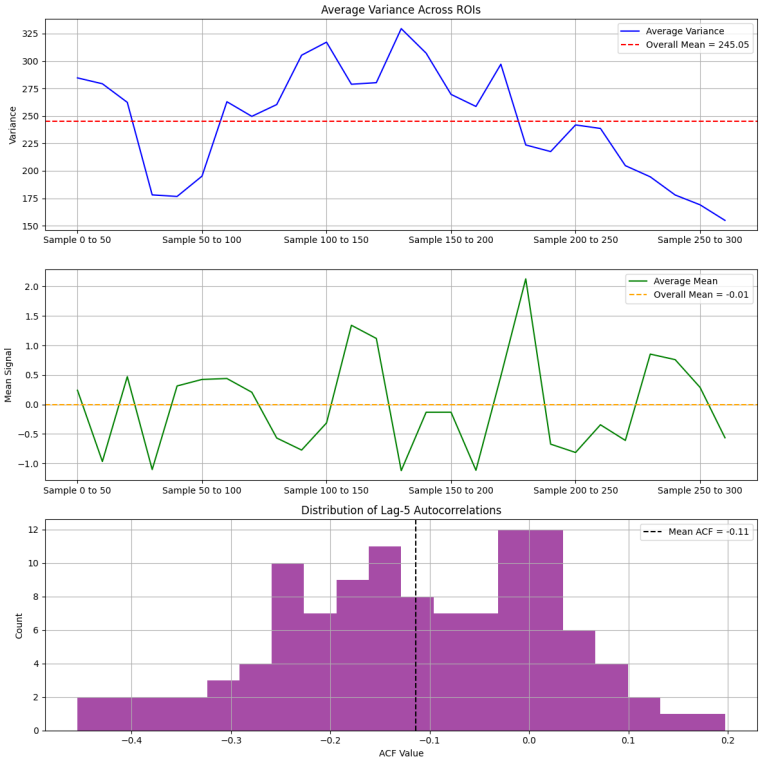


Figure I.4: Average mean and variance plots and a histogram showing the acf distribution for 5 lags using a sliding window technique

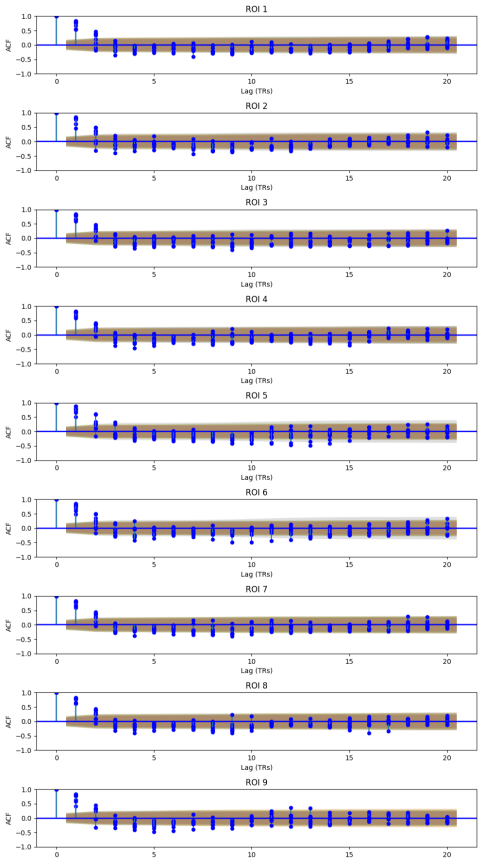


Figure I.5: ACF plotted for the first 9 ROIs (AAL) of a random subject from the ABIDE database

J

BRAIN FEATURE PLOTS

J.1. VISUALISATION OF THE FEATURES

Since there are too many different types of features and number of subjects, only one edge feature (node-to-node) and one node feature will be plotted, each for a randomly selected subject from the ASD group and one from the control (allistic) group. This will be done for all inference methods detailed in chapter 3. The used brain map will be the group ICA map with 30 components obtained through the data. The spatial coordinates for the most dominant (e.g. the highest valued) region of interest from the AAL atlas per component is set as coordinates for that component.

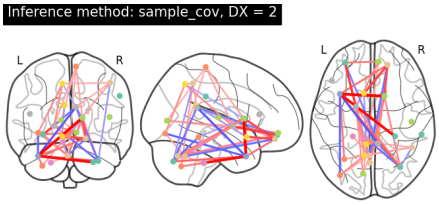
J.1.1. BRAIN CONNECTOME

The connectome plots will show the most important connectivity of the brain based on an adjacency matrix and a brain map this matrix corresponds to. This importance is determined by thresholding the adjacency matrix. A higher edge weight will correspond to a higher opacity. Red will indicate positive edges and blue negative edges.

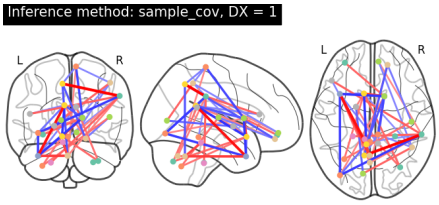
As can be seen from figures J.1 and J.2, there are some clear differences between connectivities per method and diagnosis. For some methods the top connectivities are closely related in terms of magnitude, while for some there are only a few extremes. The thresholding of the adjacency matrix has a high impact on the visibility of the connections.

J.1.2. GLASS BRAIN

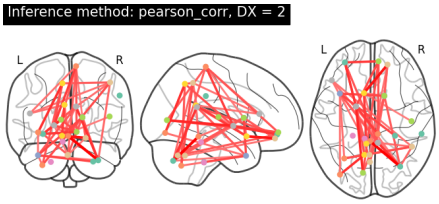
The glass brain plot will show the importance of the specified regions (nodes) as dots, based on the strength of a feature corresponding to this region. The higher the value of the feature for a region, the warmer the color of the dot (depending on the colormap used). The dots in the plot correspond to the center of a region based on the brain map.



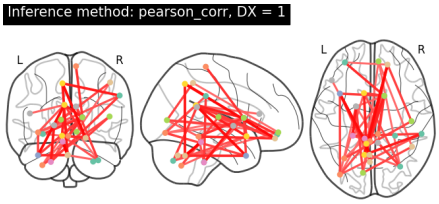
(a)



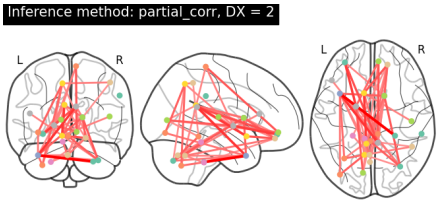
(b)



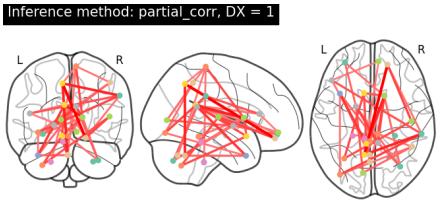
(c)



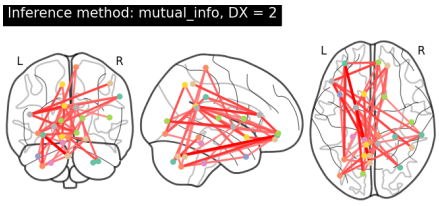
(d)



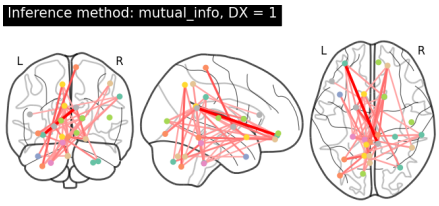
(e)



(f)

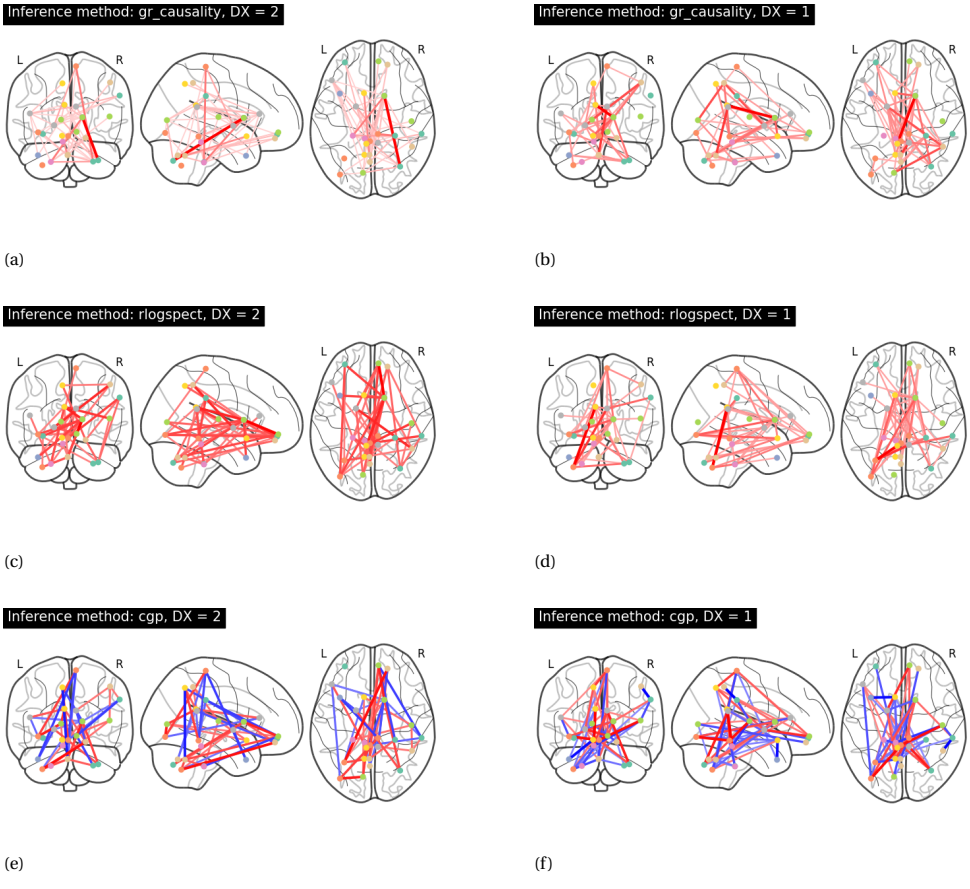


(g)



(h)

Figure J.1: Brain Connectome plot for 4 different inference methods: Sample Covariance, Pearson's Correlation, Partial Correlation and Mutual Info. Where the covariance estimate is done using the Ledoit-Wolf estimator and the data is thresholded such that only the top 10% of edges are kept. DX indicates the diagnosis, where DX=1 indicates ASD.



(a) (b)

(c) (d)

(e) (f)

Figure J.2: Brain Connectome plot for 3 different inference methods: Granger Causality, rLogSpecT and Causal Graph Process. Where the covariance estimate is done using the Ledoit-Wolf estimator and the data is thresholded such that only the top 10% of edges are kept. DX indicates the diagnosis, where DX=1 indicates ASD.

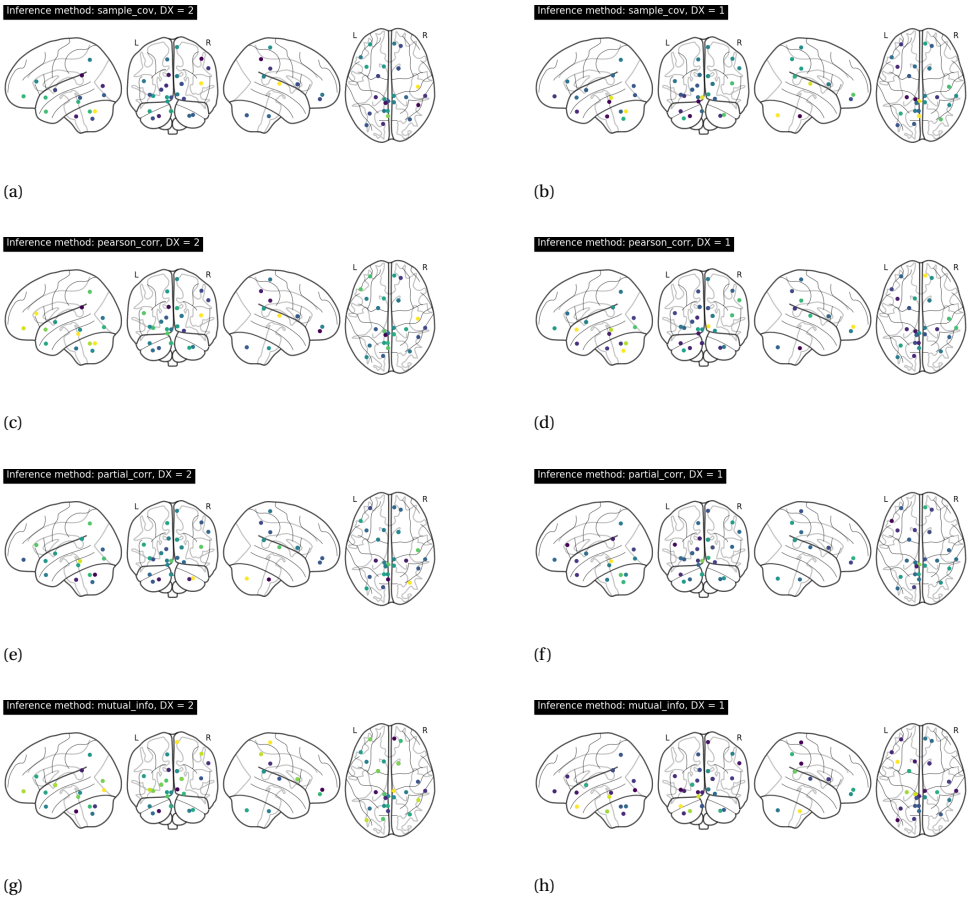
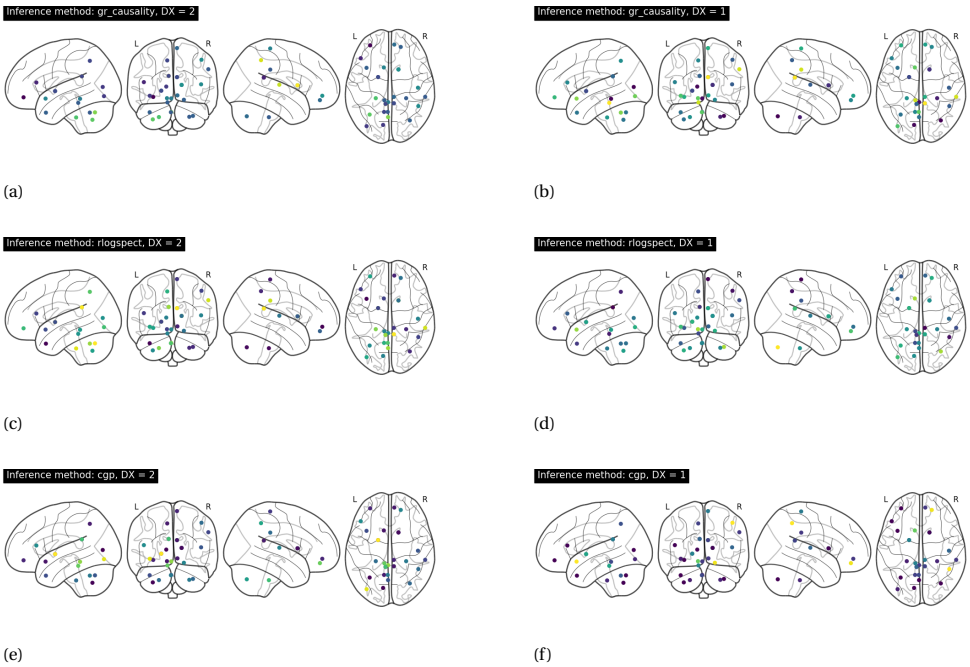


Figure J.3: Glass Brain plot based on the clustering coefficient for 4 different inference methods: Sample Covariance, Pearson's Correlation, Partial Correlation and Mutual Info. Where the covariance estimate is done using the Ledoit-Wolf estimator and the data is thresholded such that only the top 70% of edges are kept. DX indicates the diagnosis, where DX=1 indicates ASD. A warmer color corresponds to a higher clustering coefficient



(a) (b)

(c) (d)

(e) (f)

Figure J.4: Glass Brain plot based on the clustering coefficient for 3 different inference methods: Granger Causality, rLogSpecT and Causal Graph Process. Where the covariance estimate is done using the Ledoit-Wolf estimator and the data is thresholded such that only the top 70% of edges are kept. DX indicates the diagnosis, where DX=1 indicates ASD. A warmer color corresponds to a higher clustering coefficient.

K

CPAC PIPELINE

The pipeline includes the following steps:

- **Anatomical Pre-processing**
  - Skull stripping (BET or ANTs)
  - Bias field correction
  - Non-linear registration to MNI152 space (ANTs)
  - Tissue segmentation (gray matter, white matter, CSF)
- **Functional Pre-processing**
  - Slice timing correction
  - Motion correction (realignment to mean functional volume)
  - Co-registration to anatomical image
  - Spatial normalization to MNI152 space (2mm isotropic)
  - Spatial smoothing (Gaussian kernel, FWHM = 6mm)
  - Nuisance regression:
    - 6 motion parameters + derivatives
    - White matter signal
    - CSF signal
    - Optional: Global signal regression (GSR)
  - Temporal band-pass filtering (0.01–0.1 Hz)
- **Outputs**
  - Preprocessed 4D functional images (NIfTI)
  - Anatomical image in MNI space
  - Brain masks
  - ROI time series (AAL, EZ, HO, CC200, CC400 atlases)

L

CLASSIFICATION RESULTS

Here all the extra classification results will be displayed.

L.1. CLASSIFICATION CONTROL METRICS

L.1.1. RAW METRICS

Raw indicates that the metrics are from classification before feature selection.

L.1.2. ACCURACY

Flat/mean classification accuracy.

L.1.3. AUC

Area Under the Curve (AUC) can be seen as the overlap in the decision boundaries of the trained model. It serves as a balanced accuracy between the different classes.

L.2. CLASS 0 OR 1 RECALL

In binary classification there are 2 classes, 0 and 1. The recall per class represents how often the model chooses a specific class.
In most tests, the class 1 recall score is extremely low, indicating that the machine is heavily biased towards predicting class 0. This is common in classification based on neuroimaging [56], [57].

L.3. MSE

The mean squared error is also extremely high, indicating a high bias and variance in the machine's decisions. Due to this high MSE it is hard to make conclusions based on the classification results.

L.4. CLASSIFICATION TABLES

L.4.1. ICA

Table L.1: Machine Learning Classification Results

Classifier	Components	Accuracy	F1 Score	AUC	MSE
SVM	10	0.538	0.382	0.551	0.462
SVM	10	0.527	0.300	0.486	0.473
SVM	10	0.542	0.419	0.531	0.458
SVM	10	0.515	0.389	0.529	0.485
MLP	10	0.550	0.399	0.552	0.450
MLP	10	0.527	0.226	0.518	0.473
MLP	10	0.531	0.431	0.530	0.469
MLP	10	0.512	0.406	0.525	0.488
SVM	116	0.512	0.378	0.483	0.488
SVM	116	0.520	0.237	0.479	0.480
SVM	116	0.487	0.322	0.448	0.513
SVM	116	0.466	0.290	0.431	0.534
MLP	116	0.528	0.271	0.517	0.472
MLP	116	0.531	0.295	0.490	0.469
MLP	116	0.511	0.376	0.474	0.489
MLP	116	0.483	0.264	0.449	0.517
SVM	15	0.603	0.482	0.613	0.397
SVM	15	0.509	0.063	0.529	0.491
SVM	15	0.531	0.425	0.533	0.469
SVM	15	0.518	0.401	0.520	0.482
MLP	15	0.578	0.442	0.541	0.422
MLP	15	0.532	0.341	0.520	0.468
MLP	15	0.505	0.393	0.497	0.495
MLP	15	0.529	0.416	0.496	0.471
SVM	20	0.564	0.457	0.585	0.436
SVM	20	0.529	0.329	0.536	0.471
SVM	20	0.511	0.415	0.526	0.489
SVM	20	0.523	0.425	0.533	0.477
MLP	20	0.547	0.422	0.548	0.453
MLP	20	0.505	0.339	0.480	0.495
MLP	20	0.530	0.417	0.541	0.470
MLP	20	0.532	0.385	0.526	0.468
SVM	25	0.553	0.452	0.561	0.447

Continued on next page

Table L.1 – continued from previous page

Classifier	Components	Accuracy	F1 Score	AUC	MSE
SVM	25	0.512	0.251	0.508	0.488
SVM	25	0.559	0.470	0.568	0.441
SVM	25	0.543	0.460	0.561	0.457
MLP	25	0.564	0.441	0.561	0.436
MLP	25	0.513	0.358	0.483	0.487
MLP	25	0.531	0.441	0.541	0.469
MLP	25	0.539	0.481	0.557	0.461
SVM	30	0.523	0.364	0.528	0.477
SVM	30	0.532	0.325	0.518	0.468
SVM	30	0.561	0.459	0.573	0.439
SVM	30	0.561	0.467	0.575	0.439
MLP	30	0.527	0.335	0.511	0.473
MLP	30	0.536	0.372	0.525	0.464
MLP	30	0.566	0.453	0.574	0.434
MLP	30	0.571	0.466	0.590	0.429
SVM	35	0.545	0.373	0.534	0.455
SVM	35	0.562	0.399	0.568	0.438
SVM	35	0.536	0.451	0.553	0.464
SVM	35	0.548	0.452	0.557	0.452
MLP	35	0.532	0.305	0.543	0.468
MLP	35	0.561	0.395	0.577	0.439
MLP	35	0.534	0.348	0.538	0.466
MLP	35	0.564	0.452	0.556	0.436
SVM	40	0.514	0.354	0.480	0.486
SVM	40	0.529	0.264	0.481	0.471
SVM	40	0.599	0.515	0.590	0.401
SVM	40	0.598	0.510	0.593	0.402
MLP	40	0.509	0.288	0.476	0.491
MLP	40	0.520	0.292	0.512	0.480
MLP	40	0.559	0.445	0.585	0.441
MLP	40	0.582	0.497	0.608	0.418
SVM	45	0.501	0.353	0.487	0.499
SVM	45	0.502	0.248	0.498	0.498
SVM	45	0.549	0.446	0.549	0.451
SVM	45	0.542	0.419	0.542	0.458
MLP	45	0.510	0.355	0.460	0.490
MLP	45	0.515	0.270	0.494	0.485

Continued on next page

Table L.1 – continued from previous page

Classifier	Components	Accuracy	F1 Score	AUC	MSE
MLP	45	0.543	0.458	0.555	0.457
MLP	45	0.567	0.491	0.567	0.433
SVM	50	0.553	0.436	0.574	0.447
SVM	50	0.526	0.261	0.490	0.474
SVM	50	0.555	0.453	0.546	0.445
SVM	50	0.550	0.442	0.539	0.450
MLP	50	0.560	0.392	0.569	0.440
MLP	50	0.538	0.359	0.533	0.462
MLP	50	0.544	0.409	0.534	0.456
MLP	50	0.547	0.404	0.552	0.453
SVM	5	0.537	0.258	0.460	0.463
SVM	5	0.521	0.138	0.531	0.479
SVM	5	0.540	0.204	0.446	0.460
SVM	5	0.527	0.164	0.489	0.473
MLP	5	0.542	0.329	0.517	0.458
MLP	5	0.525	0.120	0.488	0.475
MLP	5	0.528	0.200	0.510	0.472
MLP	5	0.547	0.235	0.547	0.453
SVM	70	0.556	0.315	0.545	0.444
SVM	70	0.533	0.234	0.527	0.467
SVM	70	0.522	0.358	0.474	0.478
SVM	70	0.493	0.323	0.458	0.507
MLP	70	0.560	0.361	0.566	0.440
MLP	70	0.540	0.273	0.498	0.460
MLP	70	0.498	0.314	0.488	0.502
MLP	70	0.513	0.325	0.483	0.487

L.4.2. GRAPH INFERENCE METHODS

L.4.3. EFFECT OF THRESHOLDING

The test is done using the glasso sample covariance using 20 component groupICA, where the threshold was set for 20 values uniformly distributed over the range [0.000, 1.000]. The results are shown in L.3.

The results from table L.3 indicate that the optimal value of a percentile based threshold, lies somewhere between 0.6 and 0.8.

Table L.2: Machine Learning Classification Results

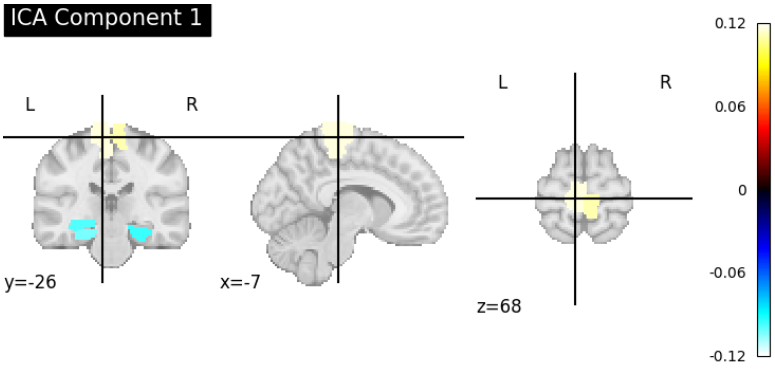
Feature Subset	Inf Method	Cov Method	Components	Threshold	Accuracy
connectivity_features	gr-causality	ledoit	20	0.9	0.525
graph_global_features	gr-causality	ledoit	20	0.9	0.528
roi_graph_features	gr-causality	ledoit	20	0.9	0.528
roi_cluster_features	gr-causality	ledoit	20	0.9	0.509
graph_based_features	gr-causality	ledoit	20	0.9	0.504
connectivity_features	gr-causality	ledoit	20	0.9	0.529
graph_global_features	gr-causality	ledoit	20	0.9	0.522
roi_graph_features	gr-causality	ledoit	20	0.9	0.536
roi_cluster_features	gr-causality	ledoit	20	0.9	0.531
graph_based_features	gr-causality	ledoit	20	0.9	0.500
connectivity_features	unknown	unknown	20	0.45	0.518
graph_global_features	unknown	unknown	20	0.45	0.514
roi_graph_features	unknown	unknown	20	0.45	0.514
roi_cluster_features	unknown	unknown	20	0.45	0.514
graph_based_features	unknown	unknown	20	0.45	0.514
connectivity_features	unknown	unknown	20	0.45	0.521
graph_global_features	unknown	unknown	20	0.45	0.507
roi_graph_features	unknown	unknown	20	0.45	0.507
roi_cluster_features	unknown	unknown	20	0.45	0.507
graph_based_features	unknown	unknown	20	0.45	0.507
connectivity_features	mutual-info	ledoit	20	0.9	0.538
graph_global_features	mutual-info	ledoit	20	0.9	0.533
roi_graph_features	mutual-info	ledoit	20	0.9	0.539
roi_cluster_features	mutual-info	ledoit	20	0.9	0.549
graph_based_features	mutual-info	ledoit	20	0.9	0.527
connectivity_features	mutual-info	ledoit	20	0.9	0.545
graph_global_features	mutual-info	ledoit	20	0.9	0.507
roi_graph_features	mutual-info	ledoit	20	0.9	0.536
roi_cluster_features	mutual-info	ledoit	20	0.9	0.520
graph_based_features	mutual-info	ledoit	20	0.9	0.533
connectivity_features	norm-laplacian	direct	20	0.30	0.586
graph_global_features	norm-laplacian	direct	20	0.30	0.588
roi_graph_features	norm-laplacian	direct	20	0.30	0.588
roi_cluster_features	norm-laplacian	direct	20	0.30	0.588
graph_based_features	norm-laplacian	direct	20	0.30	0.588
connectivity_features	norm-laplacian	direct	20	0.30	0.545
graph_global_features	norm-laplacian	direct	20	0.30	0.565
roi_graph_features	norm-laplacian	direct	20	0.30	0.565
roi_cluster_features	norm-laplacian	direct	20	0.30	0.565
graph_based_features	norm-laplacian	direct	20	0.30	0.565

Table L.3: Top 5 results ranked by composite score (accuracy, f1, auc, precision, recall, mse)

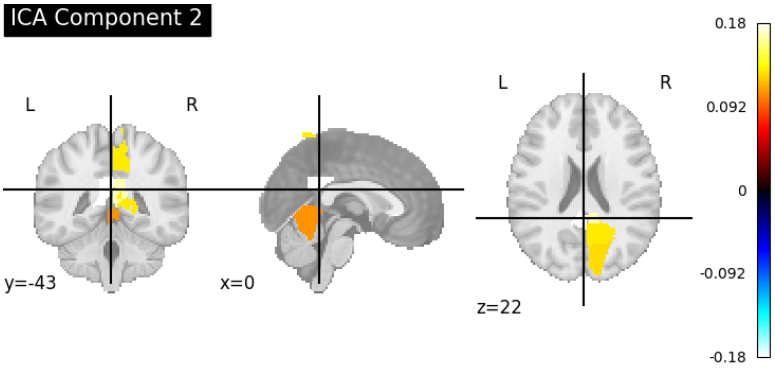
Rank	Classifier	Threshold	Accuracy	AUC	Recall	MSE
1	SVM	0.789	0.621	0.643	0.468	0.379
2	SVM	0.632	0.612	0.645	0.471	0.388
3	SVM	0.737	0.608	0.648	0.465	0.392
4	SVM	0.684	0.607	0.641	0.468	0.393
5	SVM	0.579	0.605	0.639	0.463	0.395

M

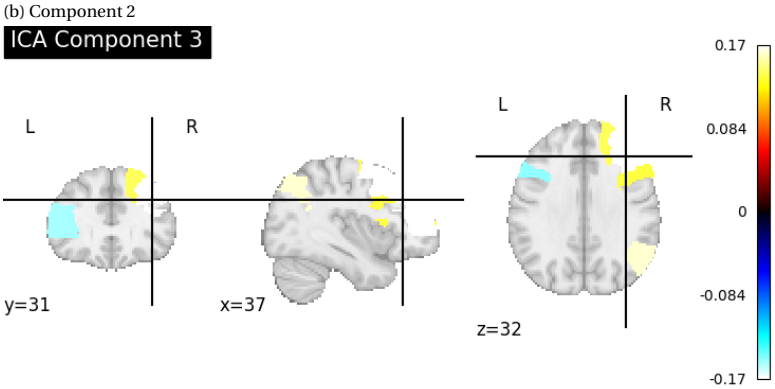
ICA COMPONENTS



(a) Component 1

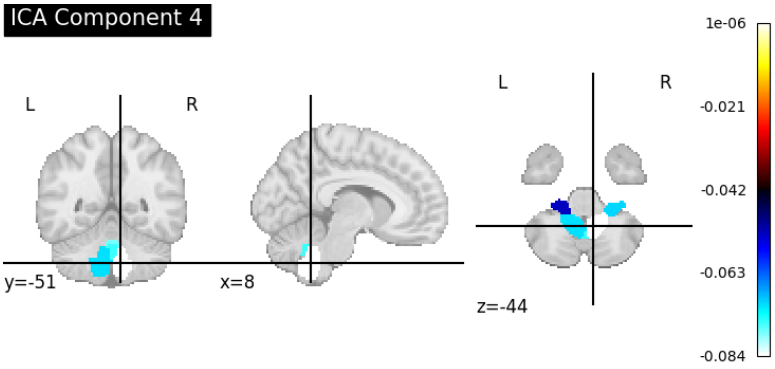


(b) Component 2

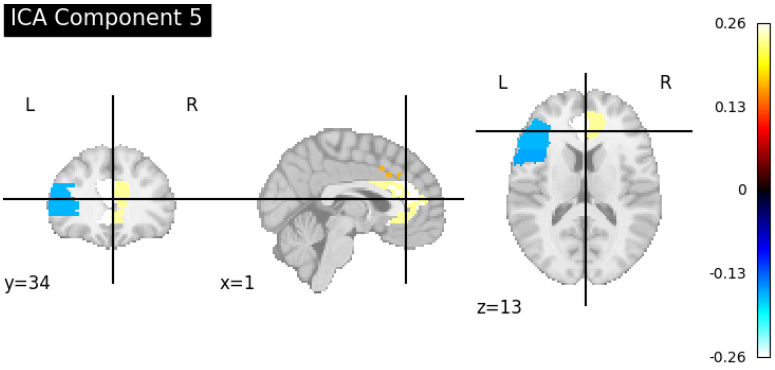


(c) Component 3

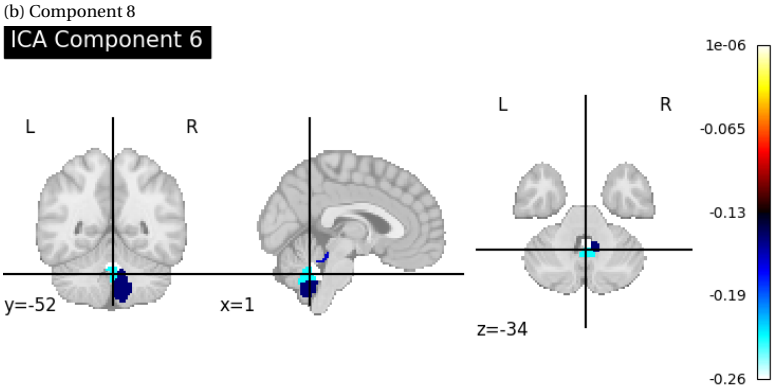
Figure M.1: Top 5% of the weights of ICA-components 1–3 from 15 group ICA plotted on the brain



(a) Component 4

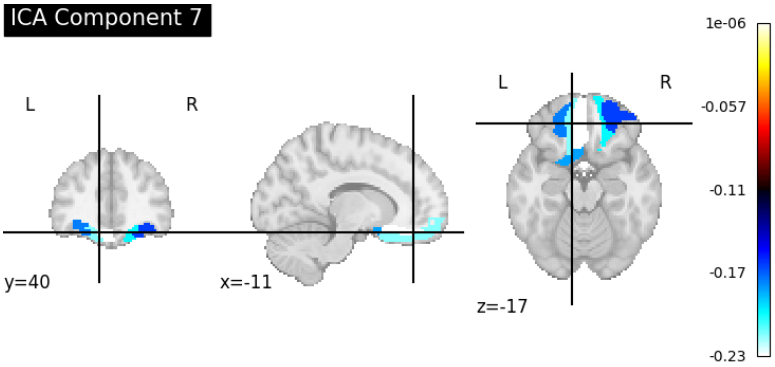


(b) Component 8

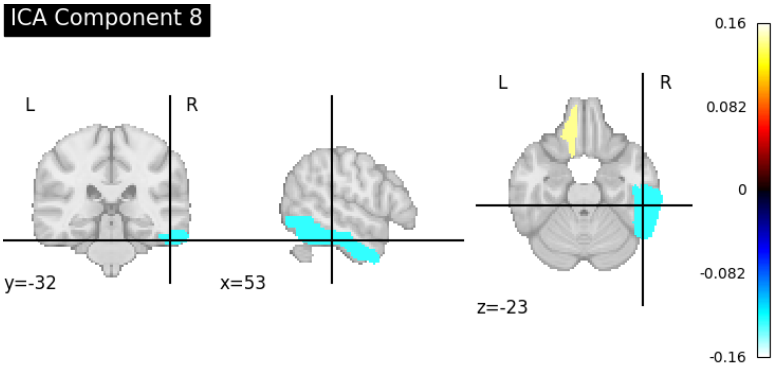


(c) Component 6

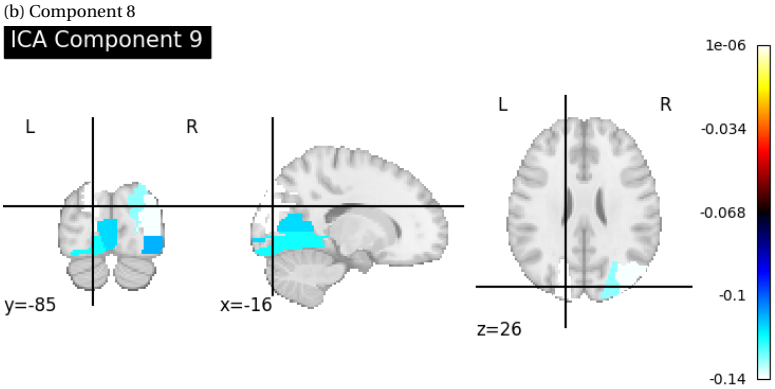
Figure M.2: Top 5% of the weights of ICA-components 4–6 from 15 group ICA plotted on the brain



(a) Component 7

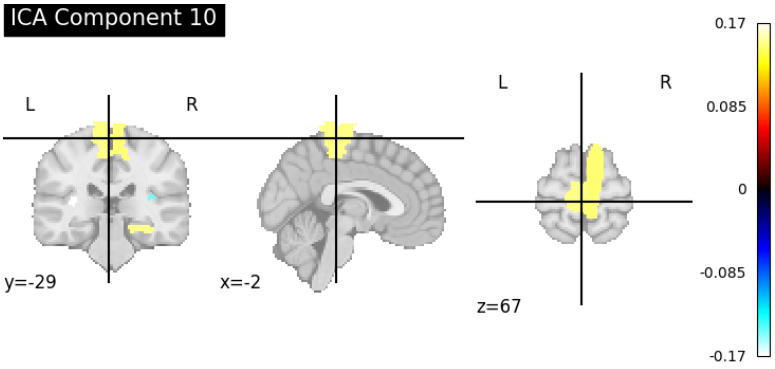


(b) Component 8

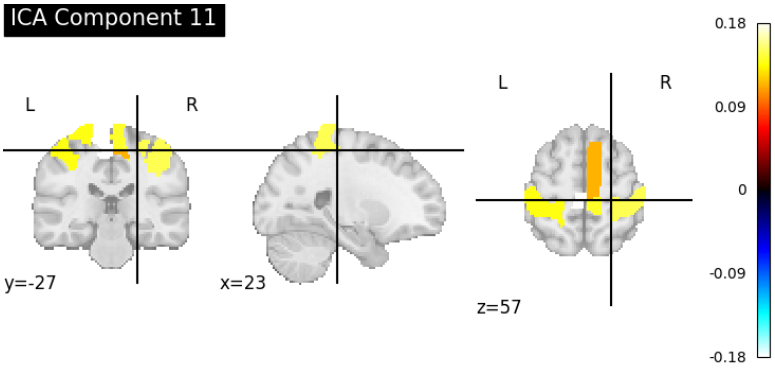


(c) Component 9

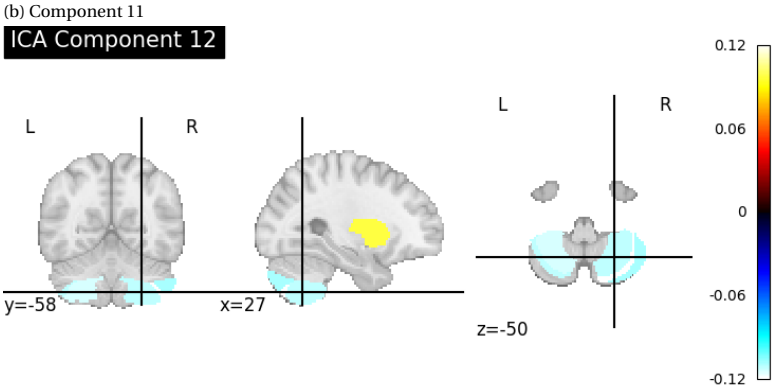
Figure M.3: Top 5% of the weights of ICA-components 7–9 from 15 group ICA plotted on the brain



(a) Component 10

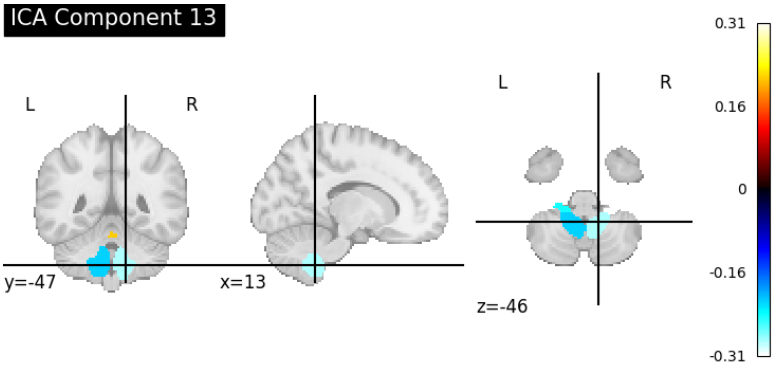


(b) Component 11

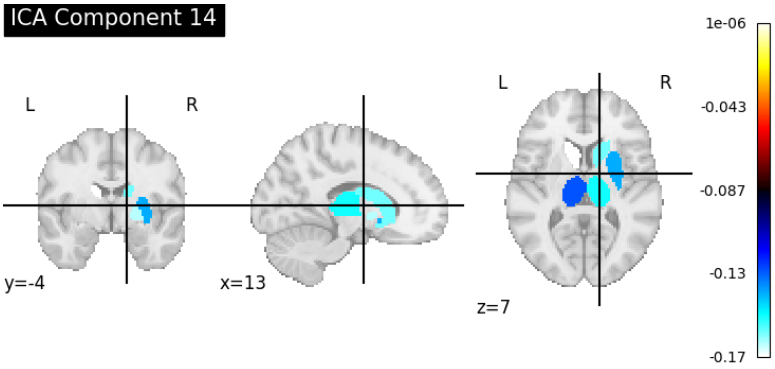


(c) Component 12

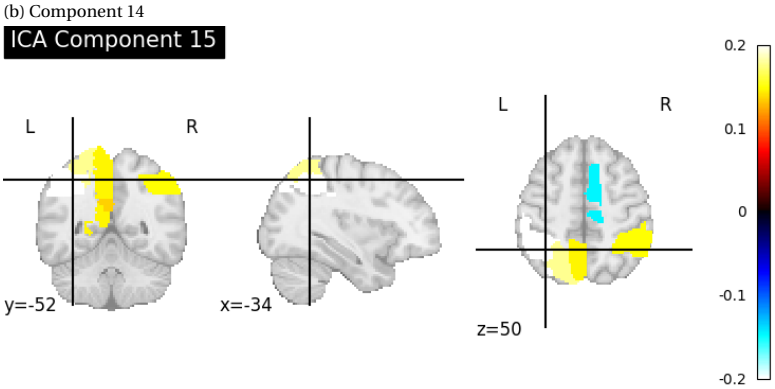
Figure M.4: Top 5% of the weights of ICA-components 10–12 from 15 group ICA plotted on the brain



(a) Component 13



(b) Component 14



(c) Component 15

Figure M.5: Top 5% of weights of ICA-components 13–15 from 15 group ICA plotted on the brain

N

Graphical User Interface

The designed GUI can be seen in figure N.1. The GUI has options to load custom data, atlasses, and ICA components. As the input data takes csv, a sample is included (no meaningful data). We have stored a numerous amount of dataframes and ICA components which can be used with the GUI. The GUI also shows information on the data, it has the possibility of changing the threshold for the connectome plot, and can plot all the features that are extracted in this project. It also includes a glass plot which is meant for node-level features. The GUI uses the AAL brain atlas as a default.

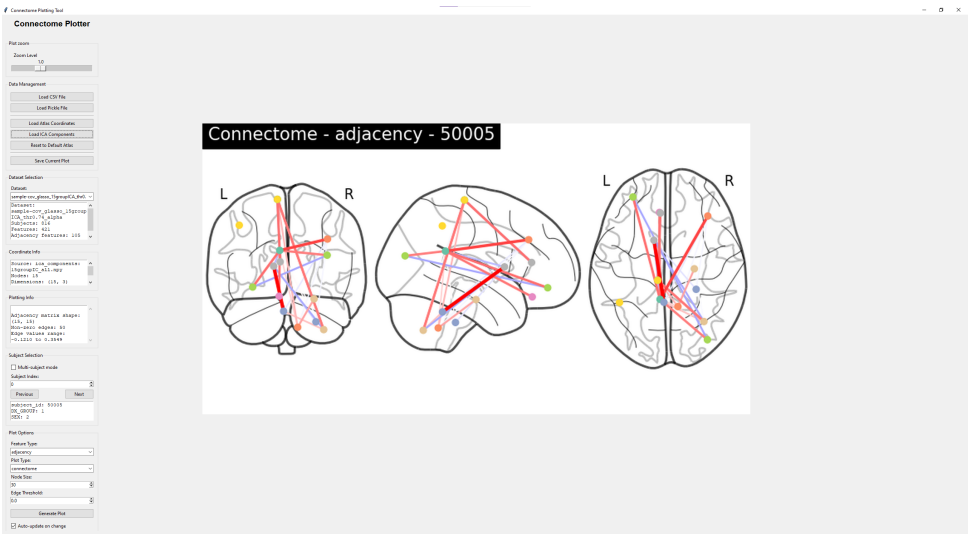


Figure N.1: Screenshot from the designed GUI, plotting a connectome for the glasso method using 15 component groupICA.

N.1. COMPLEXITY

As some models/methods used in this project are complex in nature, for example $O(N^3)$, the runtimes of the code can be very long when processing a relatively big dataset. As the goal of the product is not the diagnosis itself but more the visualization of the data, reducing the time-complexity is not necessary. However, for debugging and testing it is very beneficial to optimize it at least somewhat. This can be done in multiple ways, reducing spatial dimensionality of the data or using parallel processing to compute features. The tests are performed on the same system without interference for maximal reliability. The graph inference method used was *mutual_info*.

After reducing the number of regions from $N = 116$ to $N = 50$, the runtime decreased by a factor of approximately 4.7, corresponding to a speed-up of $4.7\times$. Reducing further to $N = 20$ resulted in a speed-up of $12.8\times$. Parallel computing, which involves utilizing multiple CPU cores simultaneously, yielded an additional speed-up of approximately $10\times$. Naturally, the actual gain depends on the hardware used and the baseline runtime of the program without parallelization. In general, parallelization becomes more effective as the runtime increases.

BIBLIOGRAPHY

[1] M. Guha, "Diagnostic and statistical manual of mental disorders: Dsm-5," *Reference Reviews*, vol. 28, no. 3, pp. 36–37, 2014.

[2] D. Wang, X. Yang, and W. Ding, "Autism spectrum disorder (asd) classification with three types of correlations based on abide i data," *Math. Found. Comput*, vol. 10, 2023.

[3] J. V. Hull, L. B. Dokovna, Z. J. Jacokes, C. M. Torgerson, A. Irimia, and J. D. Van Horn, "Resting-state functional connectivity in autism spectrum disorders: A review," *Frontiers in psychiatry*, vol. 7, p. 205, 2017.

[4] L. Q. Uddin, K. Supekar, and V. Menon, "Reconceptualizing functional brain connectivity in autism from a developmental perspective," *Frontiers in Human Neuroscience*, vol. 7, p. 458, 2013. DOI: 10.3389/fnhum.2013.00458. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnhum.2013.00458/full.

[5] M. A. Just, T. A. Keller, V. L. Malave, R. K. Kana, and S. Varma, "Autism as a neural systems disorder: A theory of frontal-posterior underconnectivity," *Neuroscience & Biobehavioral Reviews*, vol. 36, no. 4, pp. 1292–1313, 2012.

[6] Mar. 2017. [Online]. Available: https://fcon_1000.projects.nitrc.org/indi/abide/.

[7] N. K. Logothetis, J. Pauls, M. Augath, T. Trinath, and A. Oeltermann, "Neurophysiological investigation of the basis of the fmri signal," *nature*, vol. 412, no. 6843, pp. 150–157, 2001.

[8] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs," *IEEE transactions on signal processing*, vol. 61, no. 7, pp. 1644–1656, 2013.

[9] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE signal processing magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[10] X. Yang, N. Zhang, and P. Schrader, "A study of brain networks for autism spectrum disorder classification using resting-state functional connectivity," *Machine Learning with Applications*, vol. 8, p. 100 290, 2022.

[11] J. A. Nielsen, B. A. Zielinski, P. T. Fletcher, *et al.*, "Multisite functional connectivity mri classification of autism: Abide results," *Frontiers in human neuroscience*, vol. 7, p. 599, 2013.

[12] S. Mostafa, L. Tang, and F.-X. Wu, "Diagnosis of autism spectrum disorder based on eigenvalues of brain networks," *Ieee Access*, vol. 7, pp. 128 474–128 486, 2019.

[13] A. Wismüller, A. Vosoughi, and A. Kasturi, "Large-scale augmented granger causality (lsagc) for enhanced analysis of brain connectivity in autism spectrum disorder," in *Medical Imaging 2025: Clinical and Biomedical Imaging*, SPIE, vol. 13410, 2025, pp. 170–179.

[14] S. Parisot, S. I. Ktena, E. Ferrante, *et al.*, "Spectral graph convolutions for population-based disease prediction," in *Medical Image Computing and Computer Assisted Intervention- MICCAI 2017: 20th International Conference, Quebec City, QC, Canada, September 11-13, 2017, Proceedings, Part III 20*, Springer, 2017, pp. 177–185.

[15] S. Parisot, S. I. Ktena, E. Ferrante, *et al.*, "Disease prediction using graph convolutional networks: Application to autism spectrum disorder and alzheimer's disease," *Medical image analysis*, vol. 48, pp. 117–130, 2018.

[16] S. Itani and D. Thanou, "A graph signal processing framework for the classification of temporal brain data," in *2020 28th European Signal Processing Conference (EUSIPCO)*, 2021, pp. 1180–1184. DOI: 10.23919/Eusipco47968.2020.9287486.

[17] A. Brahim and N. Farrugia, "Graph fourier transform of fmri temporal signals based on an averaged structural connectome for the classification of neuroimaging," *Artificial Intelligence in Medicine*, vol. 106, p. 101 870, 2020.

[18] N. Bhagwat, A. Barry, E. W. Dickie, *et al.*, "Understanding the impact of preprocessing pipelines on neuroimaging cortical surface analyses," *GigaScience*, vol. 10, no. 1, giaa155, 2021.

[19] K. Murphy and M. D. Fox, "Towards a consensus regarding global signal regression for resting state functional connectivity mri," *NeuroImage*, vol. 154, pp. 169–173, 2017, Cleaning up the fMRI time series: Mitigating noise with advanced acquisition and correction strategies, ISSN: 1053-8119. DOI: https://doi.org/10.1016/j.neuroimage.2016.11.052. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1053811916306711.

[20] M. A. Lindquist, S. Geuter, T. D. Wager, and B. S. Caffo, "Modular preprocessing pipelines can reintroduce artifacts into fmri data," *Human brain mapping*, vol. 40, no. 8, pp. 2358–2376, 2019.

[21] C. Craddock, S. Sikka, B. Cheung, *et al.*, "Towards automated analysis of connectomes: The configurable pipeline for the analysis of connectomes (c-pac)," *Front Neuroinform*, vol. 42, no. 10.3389, 2013.

[22] A. Di Martino, C.-G. Yan, Q. Li, *et al.*, "The autism brain imaging data exchange: Towards a large-scale evaluation of the intrinsic brain architecture in autism," *Molecular psychiatry*, vol. 19, no. 6, pp. 659–667, 2014.

[23] N. Tzourio-Mazoyer, B. Landeau, D. Papathanassiou, *et al.*, "Automated anatomical labeling of activations in spm using a macroscopic anatomical parcellation of the mni mri single-subject brain," *Neuroimage*, vol. 15, no. 1, pp. 273–289, 2002.

[24] S. M. Smith, P. T. Fox, K. L. Miller, *et al.*, "Correspondence of the brain's functional architecture during activation and rest," *Proceedings of the national academy of sciences*, vol. 106, no. 31, pp. 13 040–13 045, 2009.

[25] F. Murtagh and P. Legendre, "Ward's hierarchical agglomerative clustering method: Which algorithms implement ward's criterion?" *Journal of classification*, vol. 31, pp. 274–295, 2014.

[26] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, pp. 395–416, 2007.

[27] S. Fortunato and D. Hric, "Community detection in networks: A user guide," *Physics Reports*, vol. 659, pp. 1–44, 2016.

[28] B. Thirion, G. Varoquaux, E. Dohmatob, and J.-B. Poline, "Which fmri clustering gives good brain parcellations?" *Frontiers in neuroscience*, vol. 8, p. 167, 2014.

[29] T. Blumensath, S. Jbabdi, M. F. Glasser, *et al.*, "Spatially constrained hierarchical parcellation of the brain with resting-state fmri," *Neuroimage*, vol. 76, pp. 313–324, 2013.

[30] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, P10008, 2008.

[31] R. H. Pruim, M. Mennes, D. van Rooij, A. Llera, J. K. Buitelaar, and C. F. Beckmann, "Ica-aroma: A robust ica-based strategy for removing motion artifacts from fmri data," *NeuroImage*, vol. 112, pp. 267–277, 2015.

[32] V. D. Calhoun, T. Adali, G. D. Pearlson, and J. J. Pekar, "A method for making group inferences from functional mri data using independent component analysis," *Human brain mapping*, vol. 14, no. 3, pp. 140–151, 2001.

[33] A. Zalesky, A. Fornito, L. Cocchi, L. L. Gollo, and M. Breakspear, "Time-resolved resting-state brain networks," *Proceedings of the National Academy of Sciences*, vol. 111, no. 28, pp. 10 341–10 346, 2014.

[34] O. Ledoit and M. Wolf, "A well-conditioned estimator for large-dimensional covariance matrices," *Journal of multivariate analysis*, vol. 88, no. 2, pp. 365–411, 2004.

[35] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical lasso," *Biostatistics*, vol. 9, no. 3, pp. 432–441, 2008.

[36] G. Varoquaux, A. Gramfort, J.-B. Poline, and B. Thirion, "Brain covariance selection: Better individual functional connectivity models using population prior," *Advances in neural information processing systems*, vol. 23, 2010.

[37] S. Ryali, K. Supekar, D. A. Abrams, and V. Menon, "Sparse logistic regression for whole-brain classification of fmri data," *NeuroImage*, vol. 51, no. 2, pp. 752–764, 2010, ISSN: 1053-8119. DOI: https://doi.org/10.1016/j.neuroimage.2010.02.040. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1053811910002089.

[38] V. D. Calhoun, R. Miller, G. Pearlson, and T. Adalı, "The chronnectome: Time-varying connectivity networks as the next frontier in fmri data discovery," *Neuron*, vol. 84, no. 2, pp. 262–274, 2014.

[39] P. A. Valdes-Sosa, A. Roebroeck, J. Daunizeau, and K. Friston, "Effective connectivity: Influence, causality and biophysical modeling," *Neuroimage*, vol. 58, no. 2, pp. 339–361, 2011.

[40] K. J. Friston, "Functional and effective connectivity: A review," *Brain Connectivity*, vol. 1, no. 1, pp. 13–36, 2011. DOI: 10.1089/brain.2011.0008.

[41] S. M. Smith, K. L. Miller, G. Salimi-Khorshidi, *et al.*, "Network modelling methods for fmri," *Neuroimage*, vol. 54, no. 2, pp. 875–891, 2011. DOI: 10.1016/j.neuroimage.2010.08.063.

[42] R. Salvador, J. Suckling, M. R. Coleman, J. D. Pickard, R. S. Menon, and E. T. Bullmore, "Neurophysiological architecture of functional magnetic resonance images of human brain," *Cerebral Cortex*, vol. 15, no. 9, pp. 1332–1342, 2005. DOI: 10.1093/cercor/bhh172.

[43] A. Roebroeck, E. Formisano, and R. Goebel, "Mapping directed influence over the brain using granger causality and fmri," *Neuroimage*, vol. 25, no. 1, pp. 230–242, 2005. DOI: 10.1016/j.neuroimage.2004.11.017.

[44] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro, "Network topology inference from spectral templates," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 3, pp. 467–483, 2017.

[45] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1644–1656, 2013.

[46] [Online]. Available: https://www.cvxgrp.org/scs/.

[47] S. Liu, L. Zhu, and A. M.-C. So, "Logspect: Feasible graph learning model from stationary signals with recovery guarantees," *Advances in Neural Information Processing Systems*, vol. 36, pp. 80 142–80 164, 2023.

[48] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, "Connecting the dots: Identifying network structure via graph signal processing," *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 16–43, 2019.

[49] J. Mei and J. M. Moura, "Signal processing on graphs: Causal modeling of unstructured data," *IEEE Transactions on Signal Processing*, vol. 65, no. 8, pp. 2077–2092, 2016.

[50] D. P. Kingma, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[51] K. J. Friston, "Functional and effective connectivity in neuroimaging: A synthesis," *Human brain mapping*, vol. 2, no. 1-2, pp. 56–78, 1994.

[52] S. L. Bressler and A. K. Seth, "Wiener–granger causality: A well established methodology," *NeuroImage*, vol. 58, no. 2, pp. 323–329, 2011, ISSN: 1053-8119. DOI: https://doi.org/10.1016/j.neuroimage.2010.02.059. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1053811910002272.

[53] D. A. Dickey and W. A. F. and, "Distribution of the estimators for autoregressive time series with a unit root," *Journal of the American Statistical Association*, vol. 74, no. 366a, pp. 427–431, 1979. DOI: 10.1080/01621459.1979.10482531.

[54] T. O. Laumann, A. Z. Snyder, A. Mitra, *et al.*, "On the stability of bold fmri correlations," *Cerebral cortex*, vol. 27, no. 10, pp. 4719–4732, 2017.

[55] D. J. Lurie, D. Kessler, D. S. Bassett, *et al.*, "Questions and controversies in the study of time-varying functional connectivity in resting fmri," *Network neuroscience*, vol. 4, no. 1, pp. 30–69, 2020.

[56] A. S. Heinsfeld, A. R. Franco, R. C. Craddock, A. Buchweitz, and F. Meneguzzi, "Identification of autism spectrum disorder using deep learning and the abide dataset," *NeuroImage: Clinical*, vol. 17, pp. 16–23, 2018.

[57] M. R. Arbabshirani, S. Plis, J. Sui, and V. D. Calhoun, "Single subject prediction of brain disorders in neuroimaging: Promises and pitfalls," *Neuroimage*, vol. 145, pp. 137–165, 2017.