Faculty of Science

Criticality and Brain Entropy as Predictors of PTSD Psychotherapy Treatment Response

Master's Thesis

Myrthe Sterk

Mathematical Sciences

Supervisors:

Dr. Cristian Spitoni (UU)
Dr. Remko van Lutterveld (EC MGGZ)

Second reader:

Dr. Martin Bootsma (UU)

January 24, 2024

Abstract

Post-traumatic stress disorder (PTSD) is a psychiatric disorder that may develop when an individual experiences or witnesses a traumatic stressful event. Although trauma-focused psychotherapy is a common approach for treatment, a significant proportion of patients continue to experience symptoms even after therapy. To enhance treatment response rates, it is important to increase the understanding of the neurobiological factors that may predict treatment response.

Recently, there has been a growing interest in characterizing neural activity in terms of order and disorder. It is theorized that the brain operating close to the border between order and disorder presents optimized information processing. The concepts of 'criticality' and 'entropy' provide analytical frameworks for studying these phenomena.

This thesis, conducted on behalf of the Expertisecentrum MGGZ of the Dutch Military, aims to investigate whether the criticality and entropy concepts are useful for explaining differences in treatment response. For this study, a set of resting-state functional magnetic resonance imaging (fMRI) data is used, acquired from Dutch soldiers diagnosed with PTSD before undergoing psychotherapy.

The first part of the thesis focuses on criticality, relying on the Pairwise Maximum Entropy Model. Due to constraints in data sampling, an archetype model is derived from the concatenated timeseries of the participants. Two approaches are presented, involving the inference of a 'personal' system temperature, analogous to the Ising model, and drawing phase diagrams inspired by the Sherrington-Kirkpatrick model. These methodologies aim to 'personalize' the archetype model and serve as metrics to compare the distance from criticality.

In the second part of the thesis, the concept of entropy is explored using the Permutation Fuzzy Entropy algorithm. This algorithm is chosen for its reliability in estimating Shannon entropy of neural signals in fMRI data.

Contents

Introduction

Post-traumatic stress disorder (PTSD) is a psychiatric disorder that may arise when a person experiences or observes a traumatic stressful event. The main characteristics of PTSD include re-experiencing the trauma, avoidance of traumatic reminders, negative thoughts and emotions, and increased arousal and reactivity [1]. Typically, PTSD is treated with trauma-focused psychotherapy [2]. However, the response rates are not optimal, as a subsubstantial number of patients, ranging between 30% and 50%, continue to experience symptoms after intervention. In order to enhance treatment response, we aim to investigate the neurobiological factors that may underlie the differences in respondence prior to intervention.

In the recent years, there is a growing interest in characterizing the order and disorder of brain activity. It is hypothesized that the brain organizes itself at the border between order and disorder, to optimize information processing. This notion can be analyzed using *criticality* and *entropy*. In this thesis, we are going to investigate whether these concepts can be used for explaining the difference in treatment response. For this, we will use a dataset consisting of resting-state functional magnetic resonance imaging (fMRI) scans of a group of veterans with PTSD, acquired around the start of their treatment with trauma-focused psychotherapy [3].

With criticality, we refer to a state of a system that behaves on the border between order and disorder. It has been shown, both theoretically and through experimental studies, that this state is associated with optimal computational abilities [4, 5]. The*critical brain hypothesis* proposes that the brain is such a complex system that organizes itself in a state near criticality [4, 6–10]. Due to the critical brain hypothesis, one would intuitively expect that the brains of individuals with higher cognitive abilities operates relatively closer to criticality. This relation is not empirically proven yet, but there are several studies that suggest that cognitive performance is associated with criticality [11–16]. Additionally, several researchers used criticality-based tools for studying the brain under various psychiatric conditions, which revealed differences between the healthy and affected brain [17]. As one would expect that better cognitive performance may contribute to improved treatment response, we aim to investigate whether the criticality hypothesis could be used to explain the differences in treatment response to trauma-focused psychotherapy for post-traumatic stress disorder (PTSD).

We will introduce two general methods for examining the distance to criticality, relying on the *pairwise maximum entropy model* (PMEM) [18], inspired by the work of Ezaki et al. [16] and Ruffini et al. [19]. The PMEM, rooted in statistical mechanics, serves as an abstract framework for understanding emerging phenomena, including phase transitions in large-scale networks such as neural networks [20].

This feature makes the PMEM particularly valuable for measuring the extent to which the brain operates near criticality.

In addition to investigating the concept of criticality, we will also explore the entropy of neural signals. Entropy serves as a measure of the complexity or amount of randomness (or uncertainty) within a system [21, 22]. Various studies suggest that the intricate complexity of the brain is a fundamental property underlying the manifestation of phenomena as consciousness and adaptability. Consequently, brain-related deficiencies and disorders are attributed to the decline of this neural complexity [23]. Entropy measures are therefore widely used for quantitatively characterizing abnormal (regional) brain activity in neural activity (including fMRI data), in individuals, including aged persons, patients with psychopathic and neurological disorders [24]. This encourages us to explore treatment response using the concept of entropy.

A higher entropy value indicates greater randomness and complexity, suggesting a more complicated and dynamic brain state associated with increased information processing capacity and functional brain development. Conversely, lower entropy implies less randomness and complexity, suggesting a more predictable, less flexible and rigid brain state, which means that the central nervous system is less flexible and efficient information processing [25]. In this sense, measuring the entropy of brain signals can be used for quantification of the brain function. As for the criticality, this leads to the question whether the concept of entropy could be used for explaining differences in treatment response using the resting state fMRI data.

In this thesis, conducted on behalf of the Expertise Center of the Military Mental Health outpatient clinics, we will analyze resting-state blood-oxygen-level-dependent (BOLD) fMRI data using the concepts of criticality and entropy. The dataset consists of $n = 46$ fMRI scans of veterans with PTSD, acquired around the start of treatment. After trauma-focused psychotherapy (consisting of cognitive therapy (tf-CBT), eye movement desensitization and reprocessing (EMDR) or a combination thereof), approximately half of the participants responded positively, while the other half did not. We hypothesize that the brains of participants who responded to the therapy are closer to a critical phase transition (i.e., criticality) than those of non-responders. Additionally, we aim to explore the entropy of the neural data.

The thesis is structured as follows. In Chapter 1, we introduce the dataset and provide a comprehensive overview of the findings from a previous study of this dataset. After that, the thesis is divided into two parts. In Part I, we study the concept of criticality. This part consists of four chapters, including:

- In Chapter 2, we give an introduction to criticality, the critical brain hypothesis and the methods we apply in this thesis;
- In Chapter 3, the essential preliminaries are discussed, including Spin Models (Section 3.1), the Pairwise Maximum Entropy Model (Section 3.2), the Inverse Ising Problem (Section 3.3), Biases in Inverse Ising Estimates (Section 3.4), and the Metropolis-Hastings Algorithm (Section 3.5);
- Chapter 4, the concept of criticality is studied using 'personal' system temperatures. In this chapter, we introduce the method (Section 4.1) and analyze the obtained results (Section 4.2); We will see that the 'personal' system temperature serves as a measure for the distance to criticality and is associated with treatment response.

- In Chapter 5, we explore the concept of criticality using phase diagrams. In this chapter, we, again, introduce the method (Section 5.1) and analyze the obtained results (Section 5.2). We will show that this analysis will not reveal a difference between the two groups, implying that we can not use this method to predict treatment response.

In Part II, we explore the entropy of the neural signals. This part consists of three chapters, including:

- Chapter 6 introduces three entropy algorithms, including Permutation Entropy (Section 6.1 and Fuzzy Entropy (Section 6.2). We will analyze a composition of these two algorithms, called Permutation Fuzzy Entropy, introduced in Section 6.3;
- In Chapter 7, we discuss some implementation details (Section 7.1) and we analyze the Permutation Fuzzy Entropy of the neural signals (Section 7.2). We will see that this algorithm is not sensitive enough for picking up differences in the functioning of the brain of the two groups.

Chapter 1

The Dataset

As already introduced in the introduction, the aim of this project is to investigate whether the concepts of criticality and entropy could be used for explaining differences in prospective treatment response for patients with post-traumatic stress disorder (PTSD). We aim to investigate this by analyzing a dataset consisting of resting-state fMRI scans of war veterans with a PTSD diagnosis, acquired at the beginning of their treatment. After trauma-focused psychotherapy about half of these participants were classified as treatment responders, the other half did not respond.

This dataset was studied before by van Lutterveld et al. [3], who tried to explore potential underlying (neuro)biological factors that could explain the difference in treatment response. In this study, they tried to investigate whether spontaneous brain activity, brain network characteristics and head motion during resting state could be associated with prospective treatment response using trauma-focused therapy.

The spontaneous resting-state activity of the brain was characterized using the amplitude of low-frequency fluctuations (ALFF), which is a measure used in neuroimaging to quantify the strength or intensity of spontaneous fluctuations that occur at low frequencies and reflects changes in regional brain activity. They calculated and analysed the ALFF for each voxel (volume element that is scanned) and region of interest (ROI; defined by the parcellation of the voxels according to the Brainnetome atlas [26]) in the brain and expected to find changes in spontaneous brain activity in responders compared to non-responders in areas most strongly discriminating prospective treatment responders and non-responders. However, no significant differences between the responder and non-responder group were observed for the ROI and exploratory whole-brain analysis.

To explore the functioning of the brain as a network, they performed a global network analysis using the minimum spanning tree (MST) and several network measures. The MST is a binary subgraph of the functional connectivity that connects all nodes, such that the strongest connections in the original network are included while avoiding loops. They expected to find differences in regional MST network characteristics and an increased MST global network integration. The latter, as there is suggestive evidence that network integration is related to cognition and better cognitive capabilities may lead to more effective treatment.

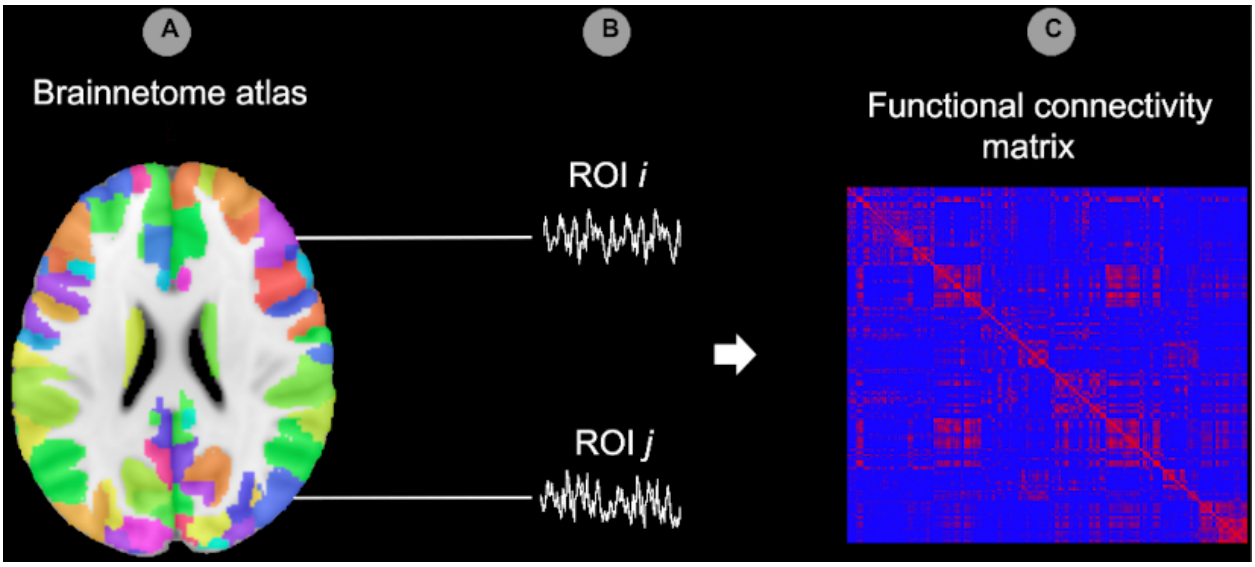


Figure 1.1: Illustration of the first steps of the graph analysis, adapted from van Lutterveld et al. [3]. (A) The Brainnetome atlas [26] is applied to the functional data. (B) Extracting the time-series for each Region Of Interest (ROI). (C) The functional connectivity between all possible pairs of ROIs i and j is calculated, using Pearson's correlation coefficient, for each participant.

The connectivity matrix was created by calculating Pearson's correlation coefficients between the regions of interest (ROIs) of the Brainnetome atlas [26]. A schematic representation of the pipeline of these steps is shown in Figure 1.1. After that, the MST was created for each functional connectivity matrix using Kruskal's algorithm and characterized using the average connectivity strength and four measures of network integration (maximum betweenness centrality, which reflects the strength of the most important note in the MST; the leaf fraction, which reflects to what extent the MST has a central organization; the diameter, which gives the efficiency of global network organization; and average eccentricity, the tendency of nodes in the network to be isolated and poorly integrated). The regional differences in network organization were studied with the centrality (i.e., relative importance) of each node, using degree and betweenness centrality. The global MST graph analysis did not show significant differences between the the group of responders compared to the non-responders. However, some significant differences between the two groups were observed in the (regional) betweenness centrality. It turned out that brain areas involved in executive function, attentional processes, learning, action and visual-object processing exhibit different functioning.

Lastly, the head motion, measured by the framewise displacement (FD), during acquisition was studied. Commonly, head motion metrics are calculated for the time series and subsequently regressed out, because it is a serious confounding factor by fMRI network analyses. However, it can also be considered as a behavioral measure in itself. Van Lutterveld et al. [3] explored whether head motion could be associated to treatment response. For this comparison, some factors that has been reported to be associated with head motion (including education level, cigarette use, dexterity, and age) were regressed out. Responders exhibited significantly less head motion than non-responders. The exact mechanism underlying the association between head motion and prospective treatment response are not clear and subject for further research.

For this project, we are going to study this dataset using the concepts of criticality and entropy. In the upcoming sections, we will discuss further relevant details about the participants, the image acquisition and preprocessing of the dataset.

1.1 Participants

We will analyze the resting-state fMRI scans of $n = 46$ war veterans diagnosed with PTSD. The PTSD diagnosis was determined by a psychologist or psychiatrist of the Military Mental Health outpatient clinics in the Netherlands. Participants were around the beginning of trauma-focus psychotherapy at the start of the study, which included trauma-focused psychotherapy included trauma-focused cognitive behavioral therapy (tf-CBT) and/or eye-movement desensitization and reprocessing (EMDR) therapy. In order to measure the severity of the PTSD symptoms, the researchers applied the Clinician-Administered PTSD Scale (CAPS) at the start of treatment and after $6 - 8$ months of treatment. Additionally, a clinical interview for DSM-IV Axis I disorders was conducted to identify any comorbid psychiatric disorders. Baseline fMRI scans and clinical interviews were performed as close as possible to the start of treatment, with clinical interviews were repeated $6 - 8$ months later to assess the changes in symptomatology.

Participants included in the study had a history of deployment to a warzone and were between 18 and 60 years old. Exclusion criteria involved a history of neurological disorders were excluded, while comorbid psychiatric disorders such as mood disorder, psychotic disorders, substance-related disorder, or any other psychiatric disorder were not considered exclusion criteria.

Participants were categorized as responder or non-responder based on their reduction in CAPS scores at the second timepoint. A participant is responsive if the total CAPS score at the second interview was reduced by at least 30 percent. The fMRI scans were acquired as close as possible to the start of treatment.

1.2 Image acquisition

The imaging was performed on a Philips Achieva 3 Tesla Clinical MRI scanner (Philips Medical System, Best, Netherlands). To improve localization of the functional data, a high-resolution T1 weighted anatomical scan was acquired with the following settings: repetition time (TR): 10 ms, echo time (TE): 4.6 ms, flip angle (FA): 8°, 200 sagittal slices, field of view (FOV) $240 \times 240 \times 160$, matrix 340×299. Hereafter, 320 blood-oxygenation level-dependent (BOLD) resting-state fMRI images were acquired per subject with the following settings: TR: 1600 ms, TE: 23 ms, FA: 72.5°, FOV $256 \times 208 \times 120$, 30 transverse slices, matrix 64×51, voxel size $4 \times 4 \times 3.60$ mm, 0.4 mm gap, total scan time 8 min and 44.8 s. During resting-state scanning, participants were asked to focus on on a fixation cross, let their min wander and relax. Furthermore, they were provided with thorough instructions to prevent head motion.

1.3 Preprocessing

As described by van Lutterveld et al. [3], the functional MRI data were preprocessed using the Data Processing Assistant for Resting-State fMRI (DPRASF) advanced edition (version 4.5) as part of the Data Processing and Analysis for Brain Imaging (DPABI) toolbox, running in MATLAB. The first 10 volumes were discarded for steady-state magnetization, leaving 310 volumes for further analysis. The remaining volumes underwent realignment, skull stripping using the Brain Extraction Tool (BET), co-registration to the individual structural scan (CSF) and linear trends removal.

After that, nuisance covariate regression was performed using a 36-parameter model and included temporal censoring and global signal regression. Temporal censoring was performed through spike regression based on framewise displacement (FD) limits of 0.2 mm calculated according to Jenkinson's algorithm. Hereafter, images were spatially normalized to Montreal Neurological Institute (MNI) space using Diffeomorphic Anatomical Registration Through Exponentiated Lie Algebra.

Part I

Criticality

Chapter 2

The criticality hypothesis

The brain is a textbook example of a complex system: complex cognitive functions are the result of neural activity across different scales in the brain; from large-scale brain networks to small neuronal circuits. In the recent years, there is a growing interest in discovering the fundamental principles underlying these global dynamics of the brain caused by the macroscopic interactions between neurons using neural data. One of these principles is the notion that neural networks, and thereby many aspects of brain activity, organizes itself into a *critical state*. This hypothesis is called the critical brain hypothesis and has been largely studied [4, 6–10]. With a critical state, or system that behaves at *criticality*, we refer to a state of a system that behaves on the border between *order* and *disorder*. It has been shown, both theoretically as with experimental studies, that this critical state is associated with a range of advantageous properties, including optimal sensitivity to inputs [27], it facilitates coordination among individual components [28, 29], it enables a large range of dynamic responses [30, 31], and maximizes the computational ability trough so-called edge-of-chaos computation [4, 32, 33]. The hypothesis is that this also applies to the brain and therefore will result in optimal memory and information processing capabilities [5].

In this section we will review the ideas behind the critical brain hypothesis, both from a neuroscientific as mathematical/physical point of view and explain how to connect these fields of science for studying the notion of criticality.

Within the framework of statistical physics, criticality refers to the behavior of a system that undergoes a *phase transition*. A phase transition is defined by the global order of the system, measured by the *order parameter*, that changes as a function of a model parameter, called the *control parameter*. The point where a small change in the control parameter leads to an abrupt change in the order parameter, is called a phase transition. This abrupt change could be a discontinuity (a jump), which is called a *first-order* or *discontinuous* phase transition, or a sharp corner (a non-differentiable point), a *second-order* or continuous phase transition. In case of a second-order phase transition, a system can be exactly at the transition point between the two phases. Then, the system is said to be 'at criticality' or in the critical state. Furthermore, when the control parameter of the system is below the critical value, the system is in the *subcritical* phase, while the system is in *supercritical* phase when the control parameter is above the critical value. [17, 34]

A good illustration of the concept of criticality, is the *two-dimensional Ising model*, which we will treat later more in detail (see Section 3.1). The classical Ising model is a system of spins, defined on a two-dimensional lattice, that can point in two directions (up- and downwards). We assume that each spin can interact with its nearest neighbors and favors to align in the same direction. In case of the Ising model, we measure the order of the model by the alignment of the spins and control the order by varying the temperature (the control parameter). At low temperature, below the critical temperature, these nearest neighbor interactions will dominate. We observe large clusters of aligned spins which will interfere with its surrounding and get larger and larger until all spins are aligned and the whole lattice is magnetized. Increasing the temperature, will temper the mutual interactions and spins will jostle the spins until the critical temperature is reached. Past this *critical temperature*, the thermal fluctuations overwhelm the interactions, leading to a loss of magnetization and randomly orientated spins. At the phase transition, the order and disorder in the system are balanced, leading to maximal correlation length. Furthermore, we observe that the order parameters (the magnetization, but also magnetization domain size and magnetic susceptibility) become power-law distributed (i.e., the random variable $x > x_0$ has probability distribution $p(x) = Cx^{-\alpha}$, where α the power-law exponent). The power-law distributed observable and increased correlation length, are characteristic for systems at criticality. [17]

From the notion of the brain operating near criticality and the additional advantageous properties, one would intuitively expect that the distance to criticality is associated with cognitive abilities. While this relation is not yet empirically proven, there is evidence supporting this hypothesis. Several studies have demonstrated an relation between cognitive performance and relatively discrete brain states during rest [12–14], as well as during working memory tasks [15], visual perception tasks [11] and intellectual ability [16]. Moreover, these studies and others [35, 36] provide evidence for the idea that the dynamics of state-transitions in the brain involve large-scale brain networks. This is in line with the notion that cognitive functions depend on the network connectivity among various regions scattered over the entire brain [37]. Additionally, several researchers used criticality-based tools to improve the understanding of common psychiatric conditions like depression [38–42], schizophrenia [43–46], anxiety [47], and PTSD [48]. Most of these studies were able to identify differences between the healthy and affected brain. This motivates us to explore whether criticality can be used for explaining the differences in treatment response. Moreover, assuming that there is a link between criticality, cognitive abilities, and treatment response, we hypothesize that the intricate dynamics of the brain near to criticality are associated with treatment response.

There are a few major conventional methods developed for studying criticality and edge-of-chaos computations in neural data. In the neuroscience, it is posed that the brain operates near criticality when the activation patterns of ensembles of neurons display typical signatures of critical behavior. Therefore, most of the methods examine the criticality hypothesis by considering neuronal avalanches and bursts of cascading activity. They aim to investigate whether these phenomena exhibit power-law properties, as these are related to criticality [49, 50]. However, this power-law behavior could be a result of a critical branching process (and therefore imply criticality), but does not necessarily imply that a system is operating near a critical state and could potentially be explained by alternative mechanisms [5, 16, 51]. To overcome this problem, we introduce a more general method that relies on statistical mechanics, a field of the mathematics and physics that can be used for understanding the collective behavior biological systems consisting of interacting particles [7].

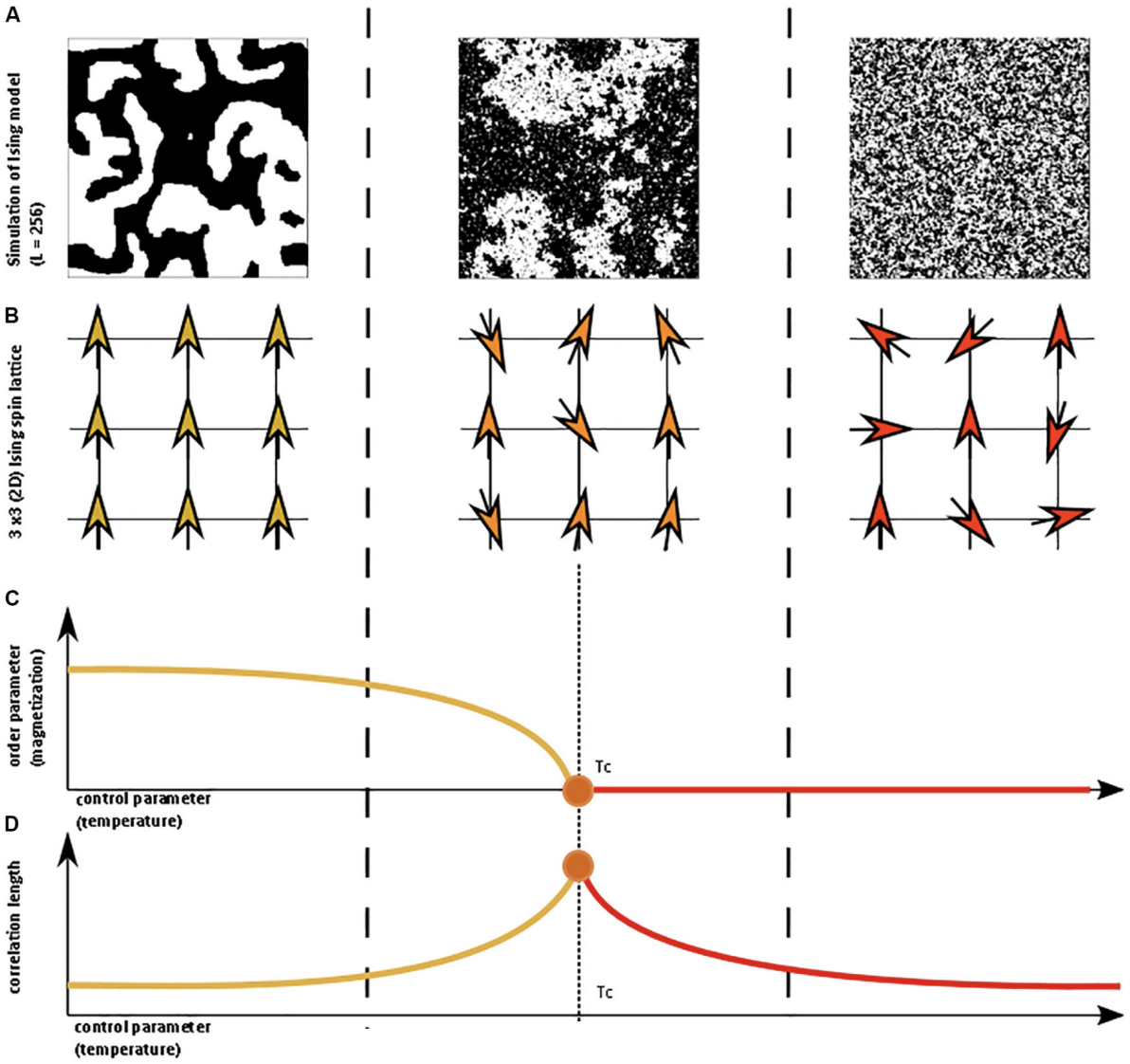


Figure 2.1: Illustration of a phase transition in the two-dimensional Ising model, adapted from Zimmern [17]. (A) Simulation of the Ising model defined on a lattice of size 256×256 in subcritical, critical and supercritical state. The orientation of the spins are represented by the black and white areas. (B) Illustration of the arrangement of spins on the lattice. (C) The order parameter (the magnetization) as a function of the control parameter (the temperature). (D) The correlation length as a function of the control parameter.

Assuming that a system can be described using a probability distribution on the different states (i.e. the system is in equilibrium), one can use the *Principle of Maximum Entropy* [18] to derive a model that provides a good estimate of the probability distribution on the observed data of an system in equilibrium. It has been shown that this model, the so-called *Pairwise Maximum Entropy Model* (PMEM), is very capable of modeling biological systems, including protein interactions, antibody diversity and the flocking behavior of birds [7, 20]. In the neuroscience, the PMEM models the pairwise interactions between nodes (or, at microscopic level, neurons) on a neuronal network. Several studies have demonstrated that this model accurately characterizes the resting-state human brain network [52, 53], including the network derived from resting-state fMRI scans [54, 55]. Additionally, the analogy of the PMEM with known statistical mechanical models, enables us to use this theory for studying their critical properties.

To obtain the PMEM that is able to describe neuronal data, the parameters of the model (including the pairwise interaction strength between the particles) has to be inferred. This problem is called the *Inverse Ising problem* and can be solved using several methods. Several studies suggest that one of the most accurate and efficient ways to do this, is using the method of *Pseudo-Likelihood Maximization (PLM)* [20,56]. However, it is well-established that this, as well as the other methods, are not accurate when the number of samples is small [56]. Given our dataset, consisting of only 310 fMRI scans per participant, it follows that it is challenging to accurately infer a PMEM for each of the participants based on their own neural data. To overcome this problem, we present two methods that relies on an *archetype* PMEM that is approximated using the concatenated data of all participants. This archetype PMEM can be adjusted in various ways to match the neural signals of each individual. We present two of these of approaches and will show that these can be utilized for studying the distance to criticality.

The first method, inspired by the work of Ruffini et al. [19], consists of inferring the *system temperature* for each participant using the archetype PMEM. The temperature of the system determines the qualitative behavior of the system and serves, as we will show, as a control parameter of the system. This behavior is determined by studying observables of the system that capture the emerging statistical properties of the inferred system. At a critical temperature T_c the system undergoes a phase transition from a ferro- to a paramagnetic state. We will show that this temperature can be inferred for each participant using the archetype PMEM and, again, pseudo-likelihood maximization. Additionally, as the temperature controls the characteristics of the system and undergoes a phase transition, we can study differences in distance to criticality individually by comparing these temperatures.

The second approach, inspired by the work of Ezaki et al. [16], involves drawing *phase diagrams* depicting observables of the archetype PMEM. For this, we use a parametrization inspired by the Sherrington-Kirkpatrick spin-glass model [57], resulting in a range of models, characterized by, again, the observables. This approach reveals, analogous to the SK-model, three qualitatively distinct types of behavior, reflected by three phases (a spin-glass, ferro- and paramagnetic phase) in the phase diagram, separated by a phase transition. Using the phase diagrams and the corresponding observables, which can be directly computed from each participants' fMRI data, allows us to approximate a parameter combination that aligns the archetype model with the neural data of each individual. This specific parameter combination corresponds to a position in the phase diagram, indicating a certain distance from the two curves where the phase transition occurs. Therefore, we study the distance to criticality using the distance to these curves.

Consequently, this method focuses on studying the distance to criticality by considering the distance from these curves.

A concern of inferring the PMEM from the whole brain imaging data (i.e., for all 238 ROIs), is that the resulting network is not necessarily informative in terms of the underlying neurology [16, 29, 58]. One could expect that certain regions and networks are more involved than others in certain processes that contribute to treatment response. Therefore, to get more insight into the underlying local interactions, we will also study seven functional networks, which are subsets of the ROIs and proposed by Yeo et al. [59], includes the Visual network (26 ROIs), Somatomotor network (33 ROIs), Dorsal Attention network (30 ROIs), Ventral Attention (22 ROIs) network, Limbic network (26 ROIs), Frontoparietal network (26 ROIs) and Default network (36 ROIs). Moreover, studying the smaller networks has the advantage that the PMEM can be inferred more accurately due to the (relatively) bigger sample size [51, 56] trying to infer is less sparsely sampled.

As mentioned in the introduction of this thesis, we hypothesize that the brains of participants who responded to psychotherapy operate closer to a phase transition than those of the non-responders.

Part I is structured as follows. In the first chapter, Chapter 3, we treat the preliminaries. Here, the point of departure will be the concept of criticality by introducing the framework of statistical mechanics by introducing three models that describe the dynamics of a spin system and contain a phase transition: the Ising model, the Generalized Ising model, the Curie Weiss model and the Sherrington-Kirkpatrick model. After that, we derive the Pairwise Maximum Entropy Model (PMEM), a model that can be used to describe the dynamics of the brain on a neural network. We will see that this model is analogous to a spin system, and we therefore can use the theory of statistical mechanics for analyzing it. These topics will be discussed in Section 3.1 respectively 3.2.

In order to obtain a PMEM that is able to describe a set of neural data that is available, we have to infer the parameters of the model. This problem, the Inverse Ising problem, can be solved using pseudolikelihood maximization. The derivation of this technique in Section 3.3.

For analyzing the obtained PMEM, must be able to generate samples and to calculate the corresponding observables. We will introduce in Section 3.5 the most basic and general methods in statistical mechanics for doing this: the Metropolis-Hastings algorithm. To conclude, it is shown that, although it is the best and most efficient algorithm, pseudolikelihood maximization could possibly cause a sample bias in the inferred model. To reduce this bias, we introduce use a bias correction (see Section 3.4), the last section of this chapter.

In the second and third chapter of this part, Chapter 4, resp. 5, we will study the concept of criticality using the system temperature and phase diagrams. Here, we will see that treatment response is associated with the system temperature when considering the functional networks. For both, we will discuss the general procedure, some implementation details and analyze the obtained results statistically.

Chapter 3

Preliminaries

3.1 Spin Models and Phase Transitions

The theory of equilibrium statistical mechanics is a field of the mathematics that was originally used for studying equilibrium thermodynamics. Today, its concepts and methods are applied across various scientific disciplines to examine the macroscopic behavior of systems composed of many interacting particles. In this thesis, we will apply this theoretical framework to the neuroscience to model neural networks and explore how these give rise to the observed neural activity.

In general, we are going to analyze *spin systems* consisting of i *spins* ($i = 1, \dots, N$) with orientation σ_i defined on a finite, undirected graph $G = (V, E)$. Typically, the set of nodes V is a subset of $\mathbb{Z}^d$ and the edges E are the *nearest neighbors* denoted by $i \sim j$, that is, pairs of nodes i, j with $\|j - i\|_1 = 1$, where $\|v\|_1 = \sum_k |v_k|$. The *configuration* or *microstate* of the system $\boldsymbol{\sigma} = \{\sigma_i\}_{i \in V}$ is an assignment of spin orientations to each node i.

These spin systems, also referred to as *magnets*, can behave in many ways influenced by external factors. In this section, we introduce some essential general concepts and methods for modeling these spin systems by presenting three spin models that describes the interactions between spins.

3.1.1 The Ising model

The Ising model [34, 60] is defined on a lattice $\Lambda_d \subset \mathbb{Z}^d$ and assume that each spin σ_i ($i \in \Lambda_d$) can point either 'up' or 'down', denoted by $+1$ or -1 respectively. This implies that $\sigma_i \in \{-1, 1\}$ and the configuration $\boldsymbol{\sigma} = \{\sigma_i\}_{i \in \Lambda_d} \in \Omega_{\Lambda_d}$, where $\Omega_{\Lambda_d} = \{-1, 1\}^{\Lambda_d}$.

The overall *energy* of a spin configuration is obtained by adding the interactions between all pairs of nodes with strength $J \in \mathbb{R}$ and the interaction with the constant external magnetic field of strength $h \in \mathbb{R}$. The energy as a function of the interaction strength J and strength of the external magnetic field h is described

by the *Hamiltonian* $\mathcal{H}^{\text{Ising}}_{\Lambda_d}(\boldsymbol{\sigma}|h,J)$, given by

$$\mathcal{H}^{\text{Ising}}_{\Lambda_d}(\boldsymbol{\sigma}|J,h) = -h\sum_{j\in\Lambda_d}\sigma_j - J\sum_{\substack{i,j\in V\\ i\sim j}}\sigma_i\sigma_j,$$

where spins at two distinct nodes $i,j\in V$ interact with interaction strength $J\in\mathbb{R}$ if and only if the pair $\{i,j\}\in E$ is an edge of the graph, implying that only neighboring nodes, denoted by $i\sim j$, interacts.

The Ising model is defined such that the system inherently tends towards configurations of minimal energy and, consequently, it satisfies two fundamental assumption as follows:

1. The interaction between nodes $i\sim j$ favors alignment of the spins, i.e. $\sigma_i=\sigma_j$ by *decreasing* the overall energy of the system by $-J\sigma_i\sigma_j$ if the spins at the nodes i,j agree. Consequently, configurations where most pairs of spins are aligned exhibit lower energy.

2. Spins align with the external magnetic field $h\in\mathbb{R}$ that acts on the system *decreasing* the energy by $-h\sigma_i$, implying that when $h>0$ spins favor to point upwards $(+1)$.

The probability of observing configuration $\boldsymbol{\sigma}$ is given by the Boltzmann distribution (also known as the *canonical Gibbs measure*) at *inverse temperature* $\beta = 1/k_BT \geq 0$:

$$P^{\text{Ising}}_{\Lambda_d}(\boldsymbol{\sigma}|J,h,\beta) = \frac{1}{Z^{\text{Ising}}_{\Lambda_d}(J,h,\beta)}\exp\left[-\beta\mathcal{H}^{\text{Ising}}_{\Lambda_d}(\boldsymbol{\sigma}|J,h)\right],$$

and *partition function*

$$Z(J,h,\beta)^{\text{Ising}}_{\Lambda_d} = \sum_{\{\boldsymbol{\sigma}\}}\exp\left[-\beta\mathcal{H}^{\text{Ising}}_{\Lambda_d}(\boldsymbol{\sigma}|J,h)\right].$$

Here, $\{\boldsymbol{\sigma}\}=\{\boldsymbol{\sigma}\in\{-1,1\}^N\}$ denotes the set of all possible configurations and $k_B = 1.380649\cdot10^{-23}m^2kgs^{-2}K^{-1}$ the *Boltzmann constant*, which relates the energy of the system to its temperature. [61]

The main goal of studying these spin systems is to investigate under which circumstances the local tendency to align induces order at global scale. For this purpose, we introduce the *magnetization* of configuration $\boldsymbol{\sigma}$, which measures the alignment quantitatively by averaging over the spin values σ_i $(i\in V)$ as follows

$$m_{\Lambda_d}(\boldsymbol{\sigma}) = \sum_{i\in\Lambda_d}\sigma_i.$$

Then, the *magnetization density*

$$\frac{m_{\Lambda_d}(\boldsymbol{\sigma})}{|\Lambda_d|}\in[-1,1]$$

gives the difference between the fractions of spins that are orientated upwards $(+1)$ and downwards (-1) respectively.

We will first study how the system depends on the temperature $T=1/k_B\beta$ by considering the fluctuations of the magnetization in the absence of a magnetic field, i.e. when $h=0$. In the limiting situation of *infinite temperature* $\beta = 1/k_BT\downarrow 0$, the Boltzmann distribution converges to the uniform distribution on Ω_{Λ_d}. We find for each $\boldsymbol{\sigma}\in\Omega_{\Lambda_d}$

$$\lim_{\beta\downarrow0}P^{\text{Ising}}_{\Lambda_d}(\boldsymbol{\sigma}|0,J,\beta) = P(\boldsymbol{\sigma}|0,J,0) = \frac{1}{|\Omega_{\Lambda_d}|},$$

implying that for $\beta \downarrow 0$ the magnetization $M_{\Lambda_d;h=0}$ is a sum of independent and identically distributed random variables. For this reason, we will find that $M_{\Lambda_d;h=0} \to 0$ as $|\Lambda_d| \to \infty$ using classical limit theorems of Probability Theory.

At *zero temperature* $\beta \uparrow \infty$, the configurations that minimize the Hamiltonian are favored. These states are called *ground states*, and are given by $\boldsymbol{\sigma}^+ = \{\sigma_i = 1\}_{i\in\Lambda_d}$ and $\boldsymbol{\sigma}^- = \{\sigma_i = -1\}_{i\in\Lambda_d}$. It can be shown easily that $P^{\text{Ising}}_{\Lambda_d}(\boldsymbol{\sigma}^+|0,J,\beta) = P^{\text{Ising}}_{\Lambda_d}(\boldsymbol{\sigma}^-|0,J,\beta)$, such that

$$\lim_{\beta\uparrow\infty} P^{\text{Ising}}_{\Lambda_d}(\boldsymbol{\sigma}|0,J,\beta) = \begin{cases} \frac{1}{2} & \text{if } \boldsymbol{\sigma} \in \{\boldsymbol{\sigma}^+, \boldsymbol{\sigma}^-\} \\ 0 & \text{otherwise} \end{cases}$$

which implies that, for low temperature, the Boltzmann distribution 'freezes' the system in either one of the ground states.

These two qualitative different types of behavior suggest two possible scenarios for high- and low-temperature behavior of the Ising model. First, when the inverse temperature β is low (and the temperature T is large), we expect that the global magnetization density $m_{\Lambda_d;h=0}/|\Lambda_d|$ is close to zero with high probability, implying that the proportion of spins that point upwards $(+1)$ is approximately equal to the part that point downwards (-1).
On the other hand, when the inverse temperature is large (and the temperature T is low), the probability distribution concentrates on the configurations that are close to the ground states $\boldsymbol{\sigma}^+, \boldsymbol{\sigma}^-$ implying that $m_{\Lambda_d;h=0}/|\Lambda_d| \simeq \pm 1$ with a high probability. In this case, we observe global order since (almost) all spins are aligned. The question that arises now, is how we can actually describe the system for values of $0 < \beta < \infty$, and where does the change in the global order of the model occur.

In case of the Ising model on a d-dimensional lattice (i.e. $\Lambda_d \subset \mathbb{Z}^d$, $d \geq 2$), it can be shown that the model can exhibit two distinct types types of behavior: *paramagnetism* and *ferromagnetism*. These states of the system, also referred to as *magnetic states*, are commonly observed in spin systems and arise from the arrangement of the spins and the interactions between them in the system.

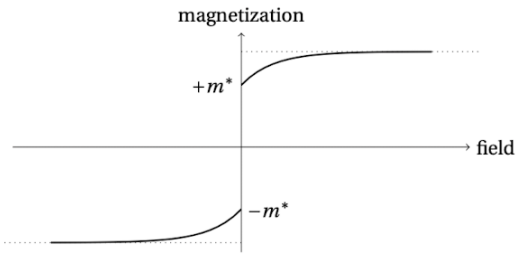


Figure 3.1: Illustration of a ferromagnetic system, adapted from the book of Friedli and Velenik [34]. We observe a discontinuity in the magnetization m when the field ($h \in \mathbb{R}$) goes through zero. The values $\pm m^*$ corresponds to the *spontaneous magnetization* of the system, that is, the magnetization of the system in the absence of an external field. The sign of the spontaneous magnetization depends on whether the external field approached zero from the positive or negative side of its domain.

A system is in a ferromagnetic state when the spins are aligned in the same direction due to strong local interactions, resulting in *global magnetization*, which persists when the external field is removed. Consequently, in case of the Ising model, looking at the graph of the magnetization m as a function of the external field strength h, we will observe discontinuity in the magnetization for $\boldsymbol{h} = 0$. An example of the graph of a ferromagnetic system is given in Figure 3.1.

In the paramagnetic state, the spins are randomly orientated, and there is no long-range magnetic order. This results in a weak attraction to external magnetic fields. In the presence of a (strong) magnetic field that forces spins to align uniformly, the system exhibits global ordering. However, when the external magnetic field is removed, the system loses its magnetization. Therefore, in case of the Ising model, looking at the graph of the magnetization m as a function of the external field strength h, we will observe a decrease of the magnetization with the strength of the external magnetic field. An example of the graph of a paramagnetic system is shown in Figure 3.2.

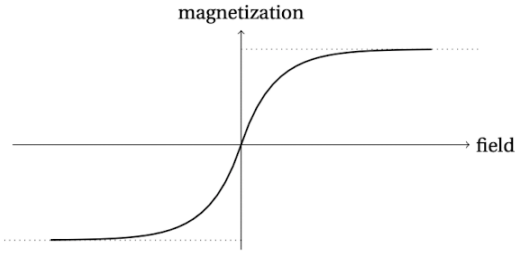


Figure 3.2: Illustration of a paramagnetic system, adapted from the book of Friedli and Velenik [34]. The magnetization m decreases with the field ($h \in \mathbb{R}$) to zero from both sides of the domain (i.e. $h > 0$ and $h < 0$.)

It can be shown that the Ising model can switch between these two states, also called *phases*, by adjusting the temperature of the system. When the temperature $T = 1/k_B\beta$ crosses a critical value T_c, the model the system will undergo a *phase transition*, leading to a qualitatively different behaving system. This transition can involve a shift from ferromagnetic to a paramagnetic system, or vice versa. We call the range range where $T > T_c$ (i.e. $\beta < \beta_c$) *supercritical regime* and where $T < T_c$ (i.e. $\beta > \beta_c$) the *subcritical regime*. The point $T = T_c$ is called the *critical regime*. For $h = 0$, the state of the system can be studied by considering the magnetization m as a function of the temperature T. An example of this graph is shown in Figure 2.1.

3.1.2 The Generalized Ising Model

More general, we define the *generalized Ising Model* as a spin system that behaves on an undirected graph $G = (V, E)$ where, as the classical Ising model, each spin σ_i ($i \in V$) can point either 'up' or 'down'. This implies that $\sigma_i \in \{-1, 1\}$ and, assuming $|V| = N$ the configuration $\boldsymbol{\sigma} = \{\sigma_i\}_{i \in V} = \{-1, 1\}^N$. Additionally, we assume that neighboring spins $i \sim j \in E$ ($i, j \in G$) are connected with interaction strength $J_{ij} \in \mathbb{R}$ and an external field works on node $i \in G$ with strength $h_i \in \mathbb{R}$.

Then, the Hamiltonian is given by

$$\mathcal{H}_G^{\text{Gen. Ising}}(\boldsymbol{\sigma}|\boldsymbol{J},\boldsymbol{h}) = -\sum_{i\in V}\sigma_i h_i - \sum_{\substack{i,j\in V\\ i\sim j}}\sigma_i\sigma_j J_{ij} = -\sum_{i=1}^{N}\sigma_i h_i - \sum_{i=1}^{N}\sum j=1, j\neq i^N \sigma_i\sigma_j J_{ij},$$

the corresponding Boltzmann distribution

$$P_V^{\text{Gen. Ising}}(\boldsymbol{\sigma}|J,h,\beta) = \frac{1}{Z(J,h,\beta)}\exp\left[-\beta\mathcal{H}_V^{\text{Gen. Ising}}(\boldsymbol{\sigma}|J,h)\right],$$

and *partition function*

$$Z(J,h,\beta)_V^{\text{Gen. Ising}} = \sum_{\{\boldsymbol{\sigma}\}}\exp\left[-\beta\mathcal{H}_{\Lambda_d}^{\text{Ising}}(\boldsymbol{\sigma}|J,h)\right].$$

3.1.3 The Curie-Weiss Model

The Curie-Weiss model [34] is such a generalized Ising model, defined on a complete graph, where it is assumed that $J_{ij} = 1/\sqrt{N}$ $(i,j=1,\dots,n)$ and $h_i = h$ $(i=1,\dots n)$. So, assuming $|V| = N$ and all nodes are connected by an unique edge, the Hamiltonian of the Curie-Weiss model for a configuration $\boldsymbol{\sigma}\in\{-1,1\}^N$ is given by

$$\mathcal{H}_V^{\text{CW}}(\boldsymbol{\sigma}|h) = -h\sum_{i\in V}\sigma_i - \frac{1}{\sqrt{N}}\sum_{i,j}\sigma_i\sigma_j,$$

with partition function

$$Z_V^{\text{CW}}(h,\beta) = \sum_{\{\boldsymbol{\sigma}\}}\exp\left[-\beta\mathcal{H}_V^{\text{CW}}(\boldsymbol{\sigma}|h)\right].$$

It can be shown that the Curie-Weiss model, like the Ising model, exhibits paramagnetic behaviour at high temperature and ferromagnetic behaviour at low temperature.

3.1.4 The Sherrington-Kirkpatrick model

The last, and main, model we consider, is the Sherrington-Kirkpatrick model [57, 62], which is a so-called spin-glass model. Spin glass models are Ising spin models with the assumption that the interaction strength between nodes is Gaussian distributed. This brings negative interactions into the system and hereby introduces a new type of behaviour: the *spin-glass* phase. In here, the quenched disorder in the interaction strengths results in both positive ferromagnetic and negative antiferromagnetic interactions between spins. The ferromagnetic behaviour forces spins to align at low temperatures, whereas antiferromagnetic couplings prefer to anti-align. Since this it is not possible to satisfy both tendencies, we say that the system is *frustrated* and we observe local, but not global magnetization.

We consider, as for the Curie-Weiss model, an generalized Ising spin system consisting of N spins defined on a complete graph where the interaction strength J_{ij} $(i,j=1,\dots,n)$ is Gaussian distributed with mean J_0 and standard deviation J (i.e. $J_{ij}\sim\mathcal{N}(J_0/\sqrt{N}, J^2/N)$). The energy of a configuration $\boldsymbol{\sigma}$ is given by the Hamiltonian

$$\mathcal{H}_V^{\text{SK}}(\boldsymbol{\sigma}|\boldsymbol{h},\boldsymbol{J}) = -\sum_i h_i\sigma_i - \frac{1}{2\sqrt{N}}\sum_{i,j=1}^{N}J_{ij}\sigma_i\sigma_j,$$

where σ_i denotes the orientation of the spin at node i, h_i describes the interaction with an external magnetic field and J_{ij} gives the interaction strength between the spins i and j. The corresponding probability distribution for observing the spin configuration $\boldsymbol{\sigma}$ is given by the Boltzmann distribution

$$P_V^{\mathrm{SK}}(\boldsymbol{\sigma}|\boldsymbol{h},\boldsymbol{J},\beta) = \frac{1}{Z_V^{\mathrm{SK}}(\boldsymbol{h},\boldsymbol{J},\beta)} \exp\left[-\beta \mathcal{H}_V^{\mathrm{SK}}(\boldsymbol{\sigma}|\boldsymbol{h},\boldsymbol{J})\right]$$

where

$$Z_V^{\mathrm{SK}}(\boldsymbol{h},\boldsymbol{J},\beta) = \sum_{\{\boldsymbol{\sigma}\}} \exp\left[-\beta \mathcal{H}_V^{\mathrm{SK}}(\boldsymbol{\sigma}|\boldsymbol{h},\boldsymbol{J})\right]$$

the partition function, $\beta = 1/k_B T$ the inverse temperature and $\{\boldsymbol{\sigma}\} = \{\boldsymbol{\sigma} \in \{-1,1\}^N\}$ the set of all possible configurations $\boldsymbol{\sigma}$.

As previously discussed, the magnetization m serves as an order parameter for the transition between the ferro- and paramagnetic phase. To analyses the behavior of a spin-glass system and the disordered arrangement and interactions of spins, a second parameter that quantifies the amount of *local* magnetization, and thereby whether the system is in a glassy state, is needed. This observable, called the *spin-glass order parameter* q, and follows as a side-effect by deriving an analytic expression for the partition function $Z(\boldsymbol{\sigma}|\boldsymbol{h},\boldsymbol{J},\beta)$ corresponding to the SK-model. [57, 62, 63]

For most of the thermodynamic quantities, we average the system over the nodes. The average $\overline{Z}$ is relatively easy to determine, but $\overline{\ln Z}$ is more difficult to calculate. To overcome this problem, the so called *replica-trick* is introduced [57, 64], which proposes to use the fact that

$$\overline{\ln Z} = \lim_{n\to 0} \frac{\overline{Z^n} - 1}{n}.$$

Here, Z^n is the *replicated partition function* of n identical replicas of the original system and is given by

$$Z^n(\boldsymbol{h},\boldsymbol{J},\beta) = \sum_{\{\boldsymbol{\sigma}\}} \exp\left[-\beta \sum_{\alpha=1}^{n} \mathcal{H}^\alpha(\boldsymbol{\sigma}|\boldsymbol{h},\boldsymbol{J})\right].$$

Here, $\alpha = 1,\ldots,n$ are the so called replica indices and $\mathcal{H}^\alpha$ is the Hamiltonian with dummy variables $\boldsymbol{\sigma}^\alpha$.

It requires some involved derivations, but one can show that for $N \to \infty$, the solution space is described by the magnetization and inter-replica spin correlation order parameters

$$m_\alpha = \frac{1}{N}\sum_i \sigma_{i\alpha},$$
$$q_{\alpha\beta} = \frac{1}{N}\sum_i \sigma_{i\alpha}\sigma_{i\beta},$$

where $\alpha,\beta = 1,\ldots,n$. Using that the replicas are mathematical artifices and are apparently indistinguishable, the simplifying *replica-symmetric* Ansatz proposes to substitute $m_\alpha = m$ for all α and $q_{\alpha\beta} = q$ for all $\alpha \neq \beta$. It follows that m and q can be identified as the average magnetization and *overlap*, given by

$$m = \frac{1}{N}\sum_{i=1}^{N} \langle\sigma_i\rangle$$
$$q = \frac{1}{N}\sum_{i=1}^{N} \langle\sigma_i\rangle^2$$

(where $\langle \sigma_i \rangle$, again, the ensemble average), and can be used to characterize three qualitatively different states of the system:

- *Paramagnetic phase* if $m = 0, q = 0$, characterized by randomly oriented magnetic moments that tend to align with the external magnetic field, exhibiting no long-range magnetic order;
- *Ferromagnetic phase* if $m \neq 0, q > 0$, characterized by a (tendency to) collective alignment of neighboring spins, exhibiting long-range magnetic order;
- *Spin-glass phase* if $m = 0, q > 0$, characterized by a disordered magnetic state with competing interactions, frustration and absence of long-range magnetic order.

On the borders between these phases, the system is said to be critical. Note that the model parameters $\boldsymbol{h}$ and $\boldsymbol{J}$ determine in which phase (and therefore how) a system will behave.

For the case where there is not external magnetic field (i.e. $\boldsymbol{h} = 0$), the corresponding phase diagram is given by figure 3.3.

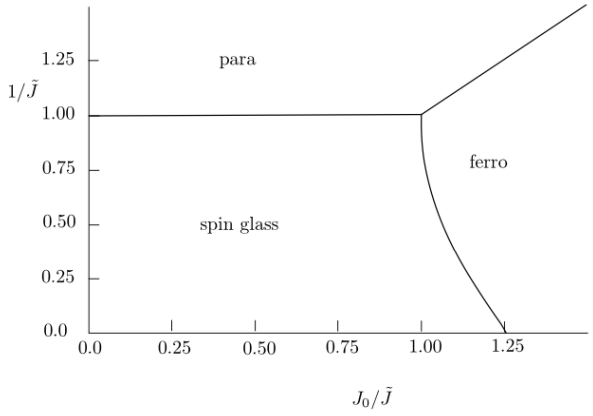


Figure 3.3: Schematic diagram of the observed phases that arise in the Sherrington-Kirkpatrick model, adapted from Sessak [65]. Here, it is assumed without loss of generality that $Tk_B = 1$, and $\boldsymbol{h} = 0$.

To quantify how a responds to an applied external field, the so-called *susceptibility* χ is used. This measure can be used to determine where a phase transition occurs.

As the ferromagnetic phase is distinguished from the paramagnetic phase by how the magnetization of the system responds to the applied field $\boldsymbol{h}$, the *local susceptibility* χ_{ij} to this phase transition can be measured for each spin σ_i $(i = 1, \ldots, N)$ by

$$\chi_{ij} = \frac{\partial}{\partial h_j} m_i = \beta \left(\langle \sigma_i \sigma_j \rangle - \langle \sigma_i \rangle \langle \sigma_j \rangle \right).$$

Note that when χ_{ij} diverges, the spin σ_i responds strongly on the applied field h_j $(i, j = 1, \ldots, N)$ implying an increase of the correlation between the spins σ_i and σ_j.

Averaging over all spins i and applied fields j, shows us how the system responds to the whole field $\boldsymbol{h}$ and

is called the *uniform susceptibility* [62]

$$\chi_{\text{Uni}} = \frac{1}{N}\sum_{i,j=1}^{N}\chi_{ij} = \beta\frac{1}{N}\sum_{i,j=1}^{N}\langle\sigma_i\sigma_j\rangle - \langle\sigma_i\rangle\langle\sigma_j\rangle.$$

In the phase transition between the paramagnetic and spin-glass phase, the uniform susceptibility will also diverge, as follows easily from the definition. However, here it is not the spin correlation $\langle\sigma_i\sigma_j\rangle$ that acquires long-range behavior, but its square (i.e. q, the spin-glass order parameter). Therefore, the so-called *spin-glass susceptibility* χ_{SG} is introduced, which measures the q-response to a random field $\boldsymbol{h}$ by

$$\chi_{\text{SG}} = \frac{1}{N}\sum_{i,j=1}^{N}\chi_{ij}^2 = \beta^2\frac{1}{N}\sum_{i,j=1}^{n}c_{ij}^2.$$

This quantity can be derived in a similar way as the uniform susceptibility and diverges at the boundary between the paramagnetic and spin-glass phase. [63]

3.2 The Pairwise Maximum Entropy Model (PMEM)

In this section, we will introduce and derive the *Pairwise Maximum Entropy Model*. This statistical model captures the pairwise interactions between particles in a system and relies on the *principle of maximum entropy*. This principle which states that the probability distribution that best represents the available information, maximizing uncertainty and is agreement with prior information, is distribution with the largest Shannon-entropy. As described, this principle can be applied for modeling a variety of complex systems with nonlinear interactions and has been successfully applied to complex systems such as the flocks of birds, social networks and gene expression [7]. We will use this model to describe the brain structure and function [55,66] by using the provided resting state fMRI data.

In this section, we will show how the PMEM can be derived using the maximum entropy principle and how it can be applied to model the functional dynamics of the brain. We will see that the PMEM is analogous to the statistical mechanical models that describe the dynamics of spin systems (as introduced in Section 3.1) and therefore undergo a phase transition, which enables us to use the PMEM for studying criticality.

3.2.1 The Maximum Entropy Principle

The maximum entropy principle serves as a basis for the PMEM. Therefore, for the sake of completeness, we will provide additional information on this principle.

The notion of maximizing the Shannon-entropy was proposed by Jaynes [18], stating that the probability distribution with the highest Shannon-entropy is the distribution that that represents best the available information about a system. In other words, if we have no information about a system except that it satisfies certain constraints, then we should assign the probability distribution that has the greatest degree of uncertainty or randomness. Here, we consider Shannon-entropy as a measure of uncertainty or randomness of a system, where a higher Shannon-entropy means that there is more uncertainty.

This implies that, in case of limited information, the most unbiased assumption is to choose the probability distribution that spreads the probability as evenly as possible by maximizing the uncertainty. A general definition of the Shannon-entropy can be found in the introduction of Part II.

It is important to emphasize that the maximum entropy principle relies on the on the assumption that we have no information about a system except that it satisfies certain constraints. If additional information is available, then the principle becomes inapplicable, and an alternative method of statistical inference that considers this information should be employed.

3.2.2 Derivation of the Pairwise Maximum Entropy Model (PMEM)

Consider the set of binary signals $\mathcal{S} = \{\boldsymbol{S}(t)\}_{t=1}^{B} \in \mathbb{R}^{N\times B}$, where $\boldsymbol{S} = \{S_i = \pm 1\}_{i=1}^{N}$ a vector of length N that describes activation of each brain region. Then, the average activation rate is given by,

$$\langle S_i \rangle_{\text{Empirical}} = \frac{1}{B}\sum_{t=1}^{B} S_i(t), \tag{3.1}$$

and the correlations

$$\langle S_i S_j \rangle_{\text{Empirical}} = \frac{1}{B}\sum_{t=1}^{B} S_i(t) S_j(t). \tag{3.2}$$

where $(i, j = 1, \ldots, N)$. The objective is to find a model that characterizes the relation between these binary signals by inferring a probability distribution $P(\boldsymbol{S})$ on all possible configurations [7]. We will do this by, applying the principle of maximum entropy as proposed by Jaynes [18], maximizing the Shannon-entropy

$$S(P) = -\sum_{\{\boldsymbol{S}\}} P(\boldsymbol{S}) \log P(\boldsymbol{\sigma}),$$

and taking into account the available information, including Equation (3.1), (3.2) and the normalization condition

$$\sum_{\{\boldsymbol{S}\}} P(\boldsymbol{S}) = 1.$$

Here $\{\boldsymbol{S}\} \equiv \{\boldsymbol{S} \in [-1, 1]^N\}$ is the set of all possible configurations the system can attain. Consequently, in the end it should hold that $\langle S_i \rangle_{\text{Empirical}} = \langle S_i \rangle_P$ and $\langle S_i S_j \rangle_{\text{Empirical}} = \langle S_i S_j \rangle_P$ where

$$\langle S_i \rangle_P = \sum_{\{\boldsymbol{S}\}} P(\boldsymbol{S}) S_i$$
$$\langle S_i S_j \rangle_P = \sum_{\{\boldsymbol{S}\}} P(\boldsymbol{S}) S_i S_j,$$

the expectation value of S_i resp. $S_i S_j$ $(i, j = 1, \ldots, N)$ under the distribution P.

Using the method of Lagrange multipliers [67], the Lagrangian corresponding to this problem is given by

$$\mathcal{L} = S + \lambda_0 g_0 + \lambda_1 g_1 + \lambda_2 g_2, \tag{3.3}$$

where

$$
\begin{aligned}
g_0 &= \sum_{\{\boldsymbol{S}\}} P(\boldsymbol{S}) - 1, \\
g_1 &= \sum_{i=1}^{N} \langle S_i \rangle_P - \langle S_i \rangle_{\text{Empirical}} \\
g_2 &= \sum_{i,j=1}^{N} \langle S_i S_j \rangle_P - \langle S_i S_j \rangle_{\text{Empirical}}
\end{aligned}
$$

and satisfies the constraints when it holds that $g_0 = g_1 = g_2 = 0$. In order to find the maximum, we have to solve

$$
\begin{aligned}
\frac{\partial}{\partial P(\boldsymbol{S})} &= \frac{\partial}{\partial P(\boldsymbol{S})} \left[S + \lambda_0 g_0 + \lambda_1 g_1 + \lambda_2 g_2 \right] \\
&= (-\ln P(\boldsymbol{S}) - 1) + \lambda_0 + \sum_{i=1}^{N} \lambda_{1_i} S_i + \sum_{i,j=1}^{N} \lambda_{2_{ij}} S_i S_j = 0,
\end{aligned}
$$

where λ_0, $\{\lambda_{1_i}\} \in \mathbb{R}^N$ and $\{\lambda_{2_{ij}}\} \in \mathbb{R}^{N \times N}$ the Lagrange multipliers.

It can be shown that this problem is maximized by the **pairwise maximum entropy probability distribution**, which is given by

$$
P(\boldsymbol{S}) = \frac{1}{Z} \exp \left[-\sum_{i=1}^{N} \lambda_{1_i} S_i - \frac{1}{2} \sum_{i,j=1}^{N} \lambda_{2_{ij}} S_i S_j \right], \tag{3.4}
$$

with the partition function

$$
Z = \sum_{\{\boldsymbol{S}\}} \exp \left[-\sum_{i=1}^{N} \lambda_{1_i} S_i - \frac{1}{2} \sum_{i,j=1}^{N} \lambda_{2_{ij}} S_i S_j \right].
$$

Note that this probability distribution is analogous to the Boltzmann distribution by interpreting the Lagrange multipliers as fields and couplings, i.e., $\lambda_{1_i} = \beta h_i$ and $\lambda_{2_{ij}} = \beta J_{ij}$.

3.3 The Inverse Ising problem

To infer the parameters for spin models as defined in Section 3.1 from a dataset, we have to solve the so called *Inverse Ising* problem. Solving the Inverse Ising problem involves computing (or actually, approximating) the interactions J_{ij} and external fields h_i $(i, j = 1, \ldots, N)$ from observed spin configurations sampled independently from the Boltzmann distribution [20].

Based on several studies [20, 56], we decided to use pseudo-likelihood inference, founded by Besag [68]. This is a method, based on logistic regression, is one of the most efficient and accurate ways of solving an inverse Ising problem [20, 56], even when the amount of data is severely limited.

In this section we treat the derivation of the pseudo-likelihood inference, as treated by Aurell and Ekeberg [56] and Ngyen et al. [20].

3.3.1 The Inverse Ising problem

We consider a system of N interacting spins $\sigma_i \in \pm 1$, $i = 1, \ldots, N$, as defined in Section 3.1.2, where a configuration is given by $\boldsymbol{\sigma} \in \{-1, 1\}^N$ and the energy of a spin configuration $\boldsymbol{\sigma}$ is given by the Hamiltonian

$$\mathcal{H}(\boldsymbol{\sigma}|\boldsymbol{h}, \boldsymbol{J}) = -\sum_{i=1}^{N} h_i \sigma_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} J_{ij} \sigma_i \sigma_j.$$

Assume that the probability of observing a given configuration $\boldsymbol{\sigma}$ is given by the Boltzmann distribution

$$P(\boldsymbol{\sigma}) = \frac{1}{Z(\boldsymbol{h}, \boldsymbol{J}, \beta)} \exp\left[-\beta \mathcal{H}\left(\boldsymbol{\sigma}|\boldsymbol{h}, \boldsymbol{J}, \beta\right)\right], \tag{3.5}$$

where $Z(\boldsymbol{h}, \boldsymbol{J}, \beta)$ the partition function, given by

$$Z(\boldsymbol{h}, \boldsymbol{J}, \beta) = \sum_{\{\boldsymbol{\sigma}\}} \exp\left[-\beta \mathcal{H}\left(\boldsymbol{\sigma}|\boldsymbol{h}, \boldsymbol{J}\right)\right],$$

where $\beta = 1/k_B T$ the inverse temperature, J_{ij} the pairwise interactions, h_i the external magnetic fields and $\{\boldsymbol{\sigma}\} = \{\boldsymbol{\sigma}|\sigma_i \in [-1, 1]^N, i = 1, \ldots, N\}$, the set of all possible spin configurations. Throughout this analysis we consider β without loss of generality to be constant, e.g. $\beta = C \in \mathbb{R}$ [20].

The expectation values of configurations of spin variables and functions $Q(\boldsymbol{\sigma})$ of them, are defined by the ensemble average

$$\langle Q(\boldsymbol{\sigma}) \rangle = \sum_{\{\boldsymbol{\sigma}\}} p(\boldsymbol{\sigma}) Q(\boldsymbol{\sigma}).$$

Subsequently, as introduction in Section 3.1, the magnetization of a spin σ_i $(i = 1, \ldots, N)$ is defined as as $m_i = \langle \sigma_i \rangle = \sum_{\{\boldsymbol{\sigma}\}} p(\boldsymbol{\sigma}) \sigma_i$ and the correlation between two spins i, j $(i, j = 1, \ldots, N, i \neq j)$ as $\chi_{ij} = \langle \sigma_i \sigma_j \rangle = \sum_{\{\boldsymbol{\sigma}\}} p(\boldsymbol{\sigma}) \left(\sigma_i \sigma_j\right)$. Furthermore, denote the pairwise correlation between two spins i, j by $c_{ij} = \chi_{ij} - m_i m_j$.

Then, inverse Ising inference is the procedure of inferring a set of parameters ($\boldsymbol{h}$, $\boldsymbol{J}$, and β) from the observations to the Boltzmann distribution, given by Equation (3.5). This can be verified by measuring the spin magnetization and correlations.

3.3.2 Solving the Inverse Ising problem using Pseudo-Likelihood Inference

Consider the set of B independent observations, $\mathrm{D} = \{\boldsymbol{\sigma}(t)\}_{t=1}^{B} \in \mathbb{R}^{N \times B}$, and assume that they are drawn from the Boltzmann distribution, given in Equation (3.5). The state of the entire set of spins at time $t = t'$, $\boldsymbol{\sigma}(t = t')$, is called a configuration, and the full data set of $B \times N$ observations the dataset. It is important to note that time is used as an index for the observations that are made in time and we are not studying how the signals evolve over time, i.e., the dynamics of the system as a function of time.

3.3.3 The Maximum Likelihood Estimator

The point of departure for the derivation of the method of pseudo-likelihood maximization, is the maximum likelihood framework [20].

Assume, given is the set of observations $x^1, x^2, \ldots, x^B$ which are all drawn from a statistical model $p(x^1, x^2, \ldots, x^B|\theta)$, where θ the unknown parameter that defines the model and has to be inferred from the observed data. One way to approximate this parameter, is using the so-called maximum likelihood estimator

$$\theta^{\mathrm{ML}} = \mathrm{argmax}_\theta p(x^1, x^2, \ldots, x^B|\theta). \tag{3.6}$$

Here, $p(x^1, x^2, \ldots, x^B|\theta)$ is the likelihood function, which is a function of the parameter θ with the observations $x^1, x^2, \ldots, x^B$ as constants. This estimator θ^{ML} converges in probability to the value θ in the limit of the number of observations.

As we are working with a lot of observations and the likelihood scales exponentially with the number of samples, it is convenient to use the log-likelihood.

The principle of maximum likelihood can also be applied to the Inverse Ising problem, where we want to infer the couplings $\boldsymbol{J}$ and magnetic fields $\boldsymbol{h}$. Assuming that the observations in a dataset $\mathrm{D} = \{\boldsymbol{\sigma}(t)\}_{t=1}^B \in \mathbb{R}^{N\times B}$ are sampled independent from the Boltzmann distribution, then the log-likelihood of corresponding to the observed configurations D is given by

$$\begin{aligned}
\mathcal{L}(\mathrm{D}|\boldsymbol{h}, \boldsymbol{J}) &= \frac{1}{B}\sum_{t=1}^{B} \ln P(\boldsymbol{\sigma}(t)|\boldsymbol{h}, \boldsymbol{J}) \\
&= \frac{1}{B}\sum_{t=1}^{B} \ln \frac{1}{Z(\boldsymbol{h}, \boldsymbol{J}, \beta)} \exp\left[-\beta \mathcal{H}\left(\sigma|\boldsymbol{h}, \boldsymbol{J}\right)\right] \\
&= \frac{1}{B}\sum_{t=1}^{B} \ln \frac{1}{Z(\boldsymbol{h}, \boldsymbol{J}, \beta)} \exp\left[\beta \sum_{i=1}^{N} h_i \sigma_i(t) + \beta \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1, j\neq i}^{N} J_{ij}\sigma_i(t)\sigma_j(t)\right] \\
&= \beta \sum_{i=1}^{N} h_i \frac{1}{B}\sum_{t=1}^{B}\sigma_i(t) + \frac{\beta}{2}\sum_{i=1}^{N}\sum_{j=1, j\neq i}^{N} J_{ij}\frac{1}{B}\sum_{t=1}^{B}\sigma_i(t)\sigma_j(t) - \frac{1}{B}\sum_{t=1}^{B}\ln Z(\boldsymbol{h}, \boldsymbol{J}, \beta) \\
&= \beta \sum_{i=1}^{N} h_i \langle\sigma_i\rangle_{\mathrm{Empirical}} + \frac{\beta}{2}\sum_{i=1}^{N}\sum_{j=1, j\neq i}^{N} J_{ij}\langle\sigma_i\sigma_j\rangle_{\mathrm{Empirical}} - \ln Z(\boldsymbol{h}, \boldsymbol{J}, \beta).
\end{aligned} \tag{3.7}$$

Here, the sample average or magnetization $\langle\sigma_i\rangle_{\mathrm{Empirical}}$ and correlation $\langle\sigma_i\sigma_j\rangle_{\mathrm{Empirical}}$ of the spins are defined by

$$\begin{aligned}
\langle\sigma_i\rangle_{\mathrm{Empirical}} &= \frac{1}{B}\sum_{t=1}^{B}\sigma_i(t) \\
\langle\sigma_i\sigma_j\rangle_{\mathrm{Empirical}} &= \frac{1}{B}\sum_{t=1}^{B}\sigma_i(t)\sigma_j(t).
\end{aligned}$$

3.3.4 Exact maximization of the likelihood

The most straightforward of inferring solving the Inverse Ising problem, is simply maximizing the log-likelihood, i.e.

$$\{\boldsymbol{h}^{\mathrm{ML}}, \boldsymbol{J}^{\mathrm{ML}}\} = \mathrm{argmax}\mathcal{L}(\mathrm{D}|\boldsymbol{h}, \boldsymbol{J}).$$

Since the log-likelihood is concave in the model parameters (see Appendix A.2), we can use, for example, gradient ascent [67] to maximize the objective. It can be derived (see Appendix A.1) that

$$\frac{\partial}{\partial h_i}\mathcal{L}(\mathrm{D}|\boldsymbol{h},\boldsymbol{J}) = \beta\left(\langle\sigma_i\rangle_{\text{Empirical}} - \langle\sigma_i\rangle_P\right)$$
$$\frac{\partial}{\partial J_{ij}}\mathcal{L}(\mathrm{D}|\boldsymbol{h},\boldsymbol{J}) = \beta\left(\langle\sigma_i\sigma_j\rangle_{\text{Empirical}} - \langle\sigma_i\sigma_j\rangle_P\right),$$

where $\langle\sigma_i\rangle_P$ and $\langle\sigma_i\sigma_j\rangle_P$ expectation values under the Boltzmann distribution P (as defined in Equation (3.5). Therefore the log-likelihood attains its maximum when, for both the magnetization and correlation $\forall i,j = 1,\ldots,N$, the empirical averages equals the the expectation under the Boltzmann distribution P, i.e.

$$\langle\sigma_i\rangle_{\text{Empirical}} = \langle\sigma_i\rangle_P$$
$$\langle\sigma_i\sigma_j\rangle_{\text{Empirical}} = \langle\sigma_i\sigma_j\rangle_P.$$

As a result, the gradient ascent updating scheme that can be used to solve this problem [67], is given by

$$h_i^{\text{new}} = h_i^{\text{old}} + \eta\frac{\partial}{\partial h_i}\mathcal{L}(\mathrm{D}|\boldsymbol{h}^{\text{old}},\boldsymbol{J}^{\text{old}})$$
$$J_{ij}^{\text{new}} = J_{ij}^{\text{old}} + \eta\frac{\partial}{\partial J_{ij}}\mathcal{L}(\mathrm{D}|\boldsymbol{h}^{\text{old}},\boldsymbol{J}^{\text{old}}),$$

where η the learning rate of the algorithm.

Now we arrive at the biggest disadvantage of this algorithm: in order to calculate the expectation values, we have to calculate a several time averages over all 2^N possible configurations. This is feasible for small N, but becomes a bottleneck when N increases. A good method that overcomes this problem, is pseudo-likelihood maximization.

3.3.5 Pseudo-likelihood maximization

With the concept of likelihood maximization in mind, we are able to derive the method of pseudo-likelihood maximization. This method, introduced by Besag [68], relies on the idea of likelihood maximization and infers the model parameters exact in the limit of an infinite number of samples.

We will start the derivation of this method by considering how the (log)likelihood (defined in Equation (3.7)) depends on the model parameters $\boldsymbol{h}$ and $\boldsymbol{J}$ and how the orientation of one spin influences the system by dividing the Hamiltonian into two parts:

$$\mathcal{H}(\boldsymbol{\sigma}|\boldsymbol{h},\boldsymbol{J}) = \mathcal{H}_i(\boldsymbol{\sigma}|\boldsymbol{h},\boldsymbol{J}) + \mathcal{H}_{/i}(\boldsymbol{\sigma}_{/i}|\boldsymbol{h},\boldsymbol{J}) = \sigma_i\left(-h_i - \frac{1}{2}\sum_{j=1,j\neq i}^{N} J_{ij}\sigma_j\right) + \mathcal{H}_{/i}(\boldsymbol{\sigma}_{/i}|\boldsymbol{h},\boldsymbol{J}), \tag{3.8}$$

where $\boldsymbol{\sigma}_{/i}$ denotes all spin variables except spin σ_i, and $\mathcal{H}_{/i}$ the Hamiltonian for all spins except spin σ_i ($i = 1,\ldots,N$). The first term of Equation 3.8 describes the contribution of spin σ_i to the total energy, which only depends on the magnetic field h_i and interactions J_{ij} of spin σ_i to the other spins, whereas the the second part gives the contribution of spin σ_i.

Using the contribution of one spin to the total energy, it follows that the conditional probability for observing a single spin $\sigma_i \in \{-1, 1\}$ is given by

$$\begin{aligned}\tilde{P}\left(\sigma_i|\boldsymbol{h}, \boldsymbol{J}, \beta, \boldsymbol{\sigma}_{/i}\right) &= \frac{\exp\left[-\beta\mathcal{H}_i(\boldsymbol{\sigma}|\boldsymbol{h}, \boldsymbol{J})\right]}{\sum_{\{\sigma_i\}}\exp\left[-\beta\mathcal{H}_i(\boldsymbol{\sigma}|\boldsymbol{h}, \boldsymbol{J})\right]} \\ &= \frac{\exp\left[\sigma_i\beta\left(h_i + \sum_{j=1, j\neq i}^{N} J_{ij}\sigma_j\right)\right]}{\exp\left[\beta\left(h_i + \sum_{j=1, j\neq i}^{N} J_{ij}\sigma_j\right)\right] + \exp\left[-\beta\left(h_i + \sum_{j=1, j\neq i}^{N} J_{ij}\sigma_j\right)\right]} \\ &= \frac{1}{2}\left(1 + \sigma_i \tanh\left[\beta\left(h_i + \sum_{j=1, j\neq i}^{N} J_{ij}\sigma_j\right)\right]\right).\end{aligned}$$

Notice that this probability only depends on the ith row of the model parameters h_i and J_{i*}, i.e. $\tilde{P}(\sigma_i|\boldsymbol{h}, \boldsymbol{J}, \beta, \boldsymbol{\sigma}_{/i}) = \tilde{P}(\sigma_i|h_i, J_{i*}, \boldsymbol{\sigma}_{/i})$. It follows that the log-likelihood of the ith row of the model parameters (i.e. h_i and J_{i*}) given the observed configurations $D = \{\sigma_i(t)\}_{t=1}^{B}$ is given by

$$\begin{aligned}\mathcal{L}_i(D|\boldsymbol{h}_i, \boldsymbol{J}_{i*}) &= \frac{1}{B}\sum_{t=1}^{B} \ln \tilde{P}(\sigma_i(t)|\boldsymbol{h}_i, \boldsymbol{J}_{i*}, \beta, \boldsymbol{\sigma}(t)_{/i}) \\ &= \frac{1}{B}\sum_{t=1}^{B} \ln \frac{1}{2}\left(1 + \sigma_i(t)\tanh\left[\beta\left(h_i + \sum_{j=1, j\neq i}^{N} J_{ij}\sigma_j(t)\right)\right]\right) \qquad (3.9)\end{aligned}$$

Now, considering all couplings and fields together by summing the log pseudo-likelihood $\mathcal{L}_i(D|\boldsymbol{h}_i, \boldsymbol{J}_{i*})$ over all $i = 1, \ldots, N$, gives us the so-called (log) pseudo-likelihood

$$\mathcal{L}(D|\boldsymbol{h}, \boldsymbol{J}) = \sum_{i=1}^{N} \mathcal{L}_i(D|\boldsymbol{h}_i, \boldsymbol{J}_{i*}) = \frac{1}{B}\sum_{i=1}^{N}\sum_{t=1}^{B} \ln \tilde{P}(\sigma_i(t)|\boldsymbol{h}_i, \boldsymbol{J}_{i*}, \beta, \boldsymbol{\sigma}(t)_{/i}). \qquad (3.10)$$

Maximizing the pseudo-likelihood with respect to all rows of $\boldsymbol{h}$ and $\boldsymbol{J}$, gives us the desired estimators.

It can be shown (again) that the log pseudo-likelihood is concave (see Appendix A.4, which implies that we can solve this system of equations using gradient ascent [67]. The updating scheme corresponding to this problem follows by the derivative of the log pseudo-likelihood with respect to h_i and J_{ij}, given by

$$\begin{aligned}\frac{\partial}{\partial h_i}\mathcal{L}(D|\boldsymbol{h}, \boldsymbol{J}) &= \beta\left(\langle\sigma_i\rangle_{\text{Empirical}} - \langle\overline{\sigma_i}\rangle_{\tilde{P}}\right) \\ \frac{\partial}{\partial J_{ij}}\mathcal{L}(D|\boldsymbol{h}, \boldsymbol{J}) &= \beta\left(\langle\sigma_i\sigma_j\rangle_{\text{Empirical}} - \langle\overline{\sigma_i\sigma_j}\rangle_{\tilde{P}}\right),\end{aligned}$$

where

$$\begin{aligned}
\langle \sigma_i \rangle_{\tilde{P}} &= \sum_{\{\sigma_i\}} \sigma_i \tilde{P}(\sigma_i | \boldsymbol{h}, \boldsymbol{J}, \beta, \boldsymbol{\sigma}_{/i}) \\
&= \sum_{\{\sigma_i\}} \sigma_i \cdot \frac{1}{2} \left(1 + \sigma_i \tanh \left[\beta \left(h_i + \sum_{j=1, j\neq i}^{N} J_{ij} \sigma_j \right) \right] \right) \\
&= \frac{1}{2} \left(1 + \tanh \left[\beta \left(h_i + \sum_{j=1, j\neq i}^{N} J_{ij} \sigma_j \right) \right] \right) - \frac{1}{2} \left(1 - \tanh \left[\beta \left(h_i + \sum_{j=1, j\neq i}^{N} J_{ij} \sigma_j \right) \right] \right) \\
&= \tanh \left[\beta \left(h_i + \sum_{j=1, j\neq i}^{N} J_{ij} \sigma_j \right) \right],
\end{aligned}$$

such that

$$\langle \overline{\sigma_i} \rangle_{\tilde{P}} = \frac{1}{B} \sum_{t=1}^{B} \tanh \left[\beta \left(h_i + \sum_{j=1, j\neq i}^{N} J_{ij} \sigma_j(t) \right) \right],$$

and

$$\begin{aligned}
\langle \sigma_i \sigma_j \rangle_{\tilde{P}} &= \sum_{\{\sigma_i\}} \sigma_i \sigma_j \tilde{P}(\sigma_i | \boldsymbol{h}, \boldsymbol{J}, \beta, \boldsymbol{\sigma}_{/i}) \\
&= \sum_{\{\sigma_i\}} \sigma_i \sigma_j \cdot \frac{1}{2} \left(1 + \sigma_i \tanh \left[\beta \left(h_i + \sum_{j'=1, j'\neq i}^{N} J_{ij'} \sigma_{j'} \right) \right] \right) \\
&= \frac{\sigma_j}{2} \left(1 + \tanh \left[\beta \left(h_i + \sum_{j'=1, j'\neq i}^{N} J_{ij'} \sigma_{j'} \right) \right] \right) - \frac{\sigma_j}{2} \left(1 - \tanh \left[\beta \left(h_i + \sum_{j'=1, j'\neq i}^{N} J_{ij'} \sigma_{j'} \right) \right] \right) \\
&= \sigma_j \tanh \left[\beta \left(h_i + \sum_{j'=1, j'\neq i}^{N} J_{ij'} \sigma_{j'} \right) \right],
\end{aligned}$$

which gives

$$\langle \overline{\sigma_i \sigma_j} \rangle_{\tilde{P}} = \frac{1}{B} \sum_{t=1}^{B} \sigma_j(t) \tanh \left[\beta \left(h_i + \sum_{j=1, j\neq i}^{N} J_{ij} \sigma_j(t) \right) \right].$$

We obtain the gradient ascent updating scheme:

$$\begin{aligned}
h_i^{\text{new}} &= h_i^{\text{old}} + \eta \left(\langle \sigma_i \rangle_{\text{Empirical}} - \langle \overline{\sigma_i} \rangle_{\tilde{P}} \right) \\
J_{ij}^{\text{new}} &= J_{ij}^{\text{old}} + \eta \left(\langle \sigma_i \sigma_j \rangle_{\text{Empirical}} - \langle \overline{\sigma_i \sigma_j} \rangle_{\tilde{P}} \right).
\end{aligned} \tag{3.11}$$

3.3.6 Inferring the temperature $T = 1/\beta$

Besides using pseudo-likelihood maximization for inferring the model parameters $\boldsymbol{h}$ and $\boldsymbol{J}$, we will also apply the framework of pseudo-likelihood optimization for inferring the temperature $\beta = 1/k_B T$. For this method, we consider $\boldsymbol{h}$ and $\boldsymbol{J}$ as known (e.g. already approximated) and fit the model to the dataset $D = \{\boldsymbol{\sigma}(t)\}_{t=1}^{M}$.

The log-likelihood for spin σ_i is in this case given by

$$\mathcal{L}_i(D|\beta) = \frac{1}{M}\sum_{t=1}^{M} \ln \tilde{P}(\sigma_i|\boldsymbol{h}_i, \boldsymbol{J}_{i*}, \beta, \boldsymbol{\sigma}_{/i}(t)),$$

where the conditional Boltzmann distribution is given by

$$\tilde{P}(\sigma_i|\boldsymbol{h}_i, \boldsymbol{J}_{i*}, \beta, \boldsymbol{\sigma}_{/i}(t)) = \frac{1}{2}\left(1 + \sigma_i \tanh\left[\beta\left(h_i + \sum_{j=1, j\neq i}^{N} J_{ij}\sigma_j\right)\right]\right).$$

The log pseudo-likelihood becomes

$$\begin{aligned}
\mathcal{L}(D|\beta) &= \sum_i \mathcal{L}_i(D|\beta) \\
&= \frac{1}{B}\sum_{i=1}^{N}\sum_{t=1}^{B} \ln \tilde{P}(\sigma_i(t)|\boldsymbol{h}_i, \boldsymbol{J}_{i*}, \beta, \boldsymbol{\sigma}(t)_{/i}) \\
&= \frac{1}{B}\sum_{i=1}^{N}\sum_{t=1}^{B} \ln \frac{1}{2}\left(1 + \sigma_i(t) \tanh\left[\beta\left(h_i + \sum_{j=1, j\neq i}^{N} J_{ij}\sigma_j(t)\right)\right]\right).
\end{aligned}$$

Using that the log pseudo-likelihood is concave (see Appendix A.6), it can be derived that the optimum can be found using gradient ascent [67]. The derivative of the log pseudo-likelihood with respect to β is given by

$$\frac{\partial}{\partial\beta}\mathcal{L}(D|\beta) = \langle\mathcal{H}\rangle_{\text{Empirical}} - \langle\overline{\mathcal{H}}\rangle_{\tilde{P}},$$

which implies that the gradient ascent algorithm becomes

$$\beta^{\text{new}} = \beta^{\text{old}} - \eta\left(\langle\mathcal{H}\rangle_{\text{Empirical}} - \langle\overline{\mathcal{H}}\rangle_{\tilde{P}}\right).$$

The derivation can be found in the Appendix A.5.

3.4 Biases in Inverse Ising Estimates: C^2 correction

Recent research has shown that estimators for inference of Inverse Ising models, such as pseudolikelihood maximization (PLM), are biased [51]. In particular, small-sample models inferred through PLM are quite large in critical regimes close to phase boundaries. This may alter the qualitative analysis of the inferred model. In order to correct for this bias, we will use a data-driven method proposed by Kloucek et al. [51].

In the limit of large sample size, PLM is exact. However, it is unlikely to have infinite sample size in real world datasets and the small sample size biases can dominate the inference. The parameter estimates obtained through PLM depend on both the number of samples B as a prefactor set by the true parameter and is not known a priori, resulting in the estimate

$$J^*_{ij} = J^0_{ij} + \frac{b_{1,ij}(\boldsymbol{h}^0, \boldsymbol{J}^0}{B} + \mathcal{O}(B^{-2}).$$

Here, the superscripts $*$ and 0 denote the inferred and true values respectively, $b_{1,ij}(\boldsymbol{h}^0, \boldsymbol{J}^0)$ is the state-dependent first order prefactor to the leading $1/B$ bias and $\mathcal{O}(B^{-2})$ consists of all lower order terms (i.e.

order $< B^{-2}$). The prefactors $b_{1,ij}(\boldsymbol{h}^0, \boldsymbol{J}^0)$ for $i, j = 1, \ldots, N$ together determine the difficulty of learning a given model by increasing or decreasing the required amount of data to dissipate the bias.
The averaged quantities of the inferred parameters, such as the standard deviation σ of the couplings $\boldsymbol{J}$ in terms of the bias is given by

$$\sigma^* = \sigma^0 + \frac{b_1(\boldsymbol{h}^0, \boldsymbol{J}^0}{B} + \mathcal{O}(B^{-1}),$$

where b_1 the combination of the bias terms $b_{1,ij}$ on the individual J_{ij} couplings and sets the bias contribution to the inferred standard deviation.
Besides this sample size bias, there is an additional small sample size issue called separation. Separation occurs when a subset of covariance (e.g. $\boldsymbol{\sigma}_{sep} \subset \boldsymbol{\sigma}_{\setminus r}$) in the logistic regression is able to perfectly predict the outcome variable (σ_r) and leads to large estimates for the corresponding parameters. The methods that correct for the first order bias term have been shown to also control separation.
It can be shown that for non-separated data, the first order bias behaves as

$$b_{1,ij}(\boldsymbol{J}^0, \boldsymbol{h}) \approx b_{1,ij}(\sigma_0),$$

implying that the bias is a function of the variance of the parameters (i.e. the inverse temperature $T = 1/(\sigma\sqrt{N})$).

The procedure that is introduced to correct for this bias, uses the result that the bias is captured by the temperature of the system and the PLM models over-estimate the covariance C^2, defined by

$$C^2 = \frac{1}{N} \sum_{i,j=1}^{N} C_{ij}^2, \tag{3.12}$$

It proposes to require that the inferred model has a covariance C^2 as close as possible to the one estimated from the input data. This is reached by performing a second optimization *after* estimating the PLM parameters by minimizing the objective

$$\mathcal{L}(T_f) = \left(C^2_{\text{empirical}} - C^2_{\text{MC}}(T_f)\right)^2,$$

where $C^2_{\text{empirical}}$ is determined as C^2 in equation 3.12 from the dataset and C^2_{MC} is calculated from MC simulations where the re-scaled parameters $\boldsymbol{J}/T_f$ and $\boldsymbol{h}/T_f$ are used. Here, the re-scaling parameter $T_f > 0$ acts as a fictitious temperature.

This procedure (significantly) improves the reconstructed temperature when a PLM solution can be found (i.e. separation does not occur) and provides the best improvement at higher temperatures.

3.5 Sampling from the Boltzmann distribution: the Metropolis-Hastings Algorithm

There are several ways to generate configurations of spins that follow the Boltzmann distribution. The most basic known algorithms, is *Metropolis-Hastings algorithm* [69–72], which is a *Monte Carlo method* that is used to generate random samples (in this case configurations $\boldsymbol{\sigma}$) according to the (conditional) Boltzmann distribution $P(\boldsymbol{\sigma}|\boldsymbol{h}, \boldsymbol{J}, \beta)$.

The idea of Monte Carlo sampling is to pick states in such a manner that the probability that a particular state $\boldsymbol{\sigma}$ is chosen, equals $P(\boldsymbol{\sigma}|\boldsymbol{h}, \boldsymbol{J}, \beta)$. We can do this by generating a Markov chain of successive states $\boldsymbol{\sigma}^1 \to \boldsymbol{\sigma}^2 \to \ldots$. Here, each state is generated from the previous one using the transition probability $T(\boldsymbol{\sigma} \to \boldsymbol{\sigma}')$, which gives the probability to transform from state $\boldsymbol{\sigma}$ to $\boldsymbol{\sigma}'$. This transition probability should be chosen such that the transition occurs with probability $P(\boldsymbol{\sigma}|\boldsymbol{h}, \boldsymbol{J}, \beta)$.
Suppose that the probability of observing state $\boldsymbol{\sigma}$ in the Markov chain at time t is denoted by $P_t(\boldsymbol{\sigma})$, then, in terms of transition probabilities, the probability observing the state $\boldsymbol{\sigma}$ at $t+1$ is given by

$$P_{t+1}(\boldsymbol{\sigma}) = P_t(\boldsymbol{\sigma}) + \sum_{\{\boldsymbol{\sigma}\}} \left[T(\boldsymbol{\sigma}' \to \boldsymbol{\sigma}) P_t(\boldsymbol{\sigma}') - T(\boldsymbol{\sigma} \to \boldsymbol{\sigma}') P_t(\boldsymbol{\sigma}) \right]. \tag{3.13}$$

When you generate enough samples (i.e. $t \to \infty$), the probability $P_{t+1}(\boldsymbol{\sigma})$ will converge to the stationary distribution, which we want to be the Boltzmann distribution. This can be obtained, by choosing the transition probabilities T in such a way that $P_t(\boldsymbol{\sigma}) = P(\boldsymbol{\sigma}|\boldsymbol{h}, \boldsymbol{J}, \beta)$ and all terms in the summation vanish. So, it should hold for all $\boldsymbol{\sigma}$ and $\boldsymbol{\sigma}'$ that

$$T(\boldsymbol{\sigma} \to \boldsymbol{\sigma}') P(\boldsymbol{\sigma}'|\boldsymbol{h}, \boldsymbol{J}, \beta) = T(\boldsymbol{\sigma}' \to \boldsymbol{\sigma}) P(\boldsymbol{\sigma}|\boldsymbol{h}, \boldsymbol{J}, \beta).$$

This is called the detailed balance equation and implies that the process has to be reversible. From this, it follows that it should hold that

$$\frac{T(\boldsymbol{\sigma} \to \boldsymbol{\sigma}')}{T(\boldsymbol{\sigma}' \to \boldsymbol{\sigma})} = \frac{P(\boldsymbol{\sigma}'|\boldsymbol{h}, \boldsymbol{J}, \beta)}{P(\boldsymbol{\sigma}|\boldsymbol{h}, \boldsymbol{J}, \beta)} = \exp\left[-\beta(\mathcal{H}(\boldsymbol{\sigma}') - \mathcal{H}(\boldsymbol{\sigma}))\right] = \exp\left[-\beta \Delta\mathcal{H}(\boldsymbol{\sigma}', \boldsymbol{\sigma})\right], \tag{3.14}$$

where $\Delta\mathcal{H}(\boldsymbol{\sigma}, \boldsymbol{\sigma}') = \mathcal{H}(\boldsymbol{\sigma}') - \mathcal{H}(\boldsymbol{\sigma})$ the change in energy.

There are different transition probabilities T that satisfies this condition (3.14). One of these possible transition probabilities was proposed by Metropolis [69] and is given by

$$T(\boldsymbol{\sigma} \to \boldsymbol{\sigma}') = \begin{cases} 1, & \text{if } \Delta\mathcal{H} \leq 0 \\ \exp\left[-\beta \Delta\mathcal{H}(\boldsymbol{\sigma}, \boldsymbol{\sigma}')\right] & \text{if } \Delta\mathcal{H} \geq 0. \end{cases} \tag{3.15}$$

The procedure where we generate a Markov chain using this transition probability, is called the *Metropolis-Hastings algorithm* [69, 70].

In order to determine change in energy $\Delta\mathcal{H}$ when flipping a spin k, first notice that we can rewrite the

Hamiltonian as

$$\begin{aligned}\mathcal{H}(\boldsymbol{\sigma}|\boldsymbol{h},\boldsymbol{J}) &= -\sum_{i=1}^{N} h_i\sigma_i - \sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N} J_{ij}\sigma_i\sigma_j \\ &= \left(-h_k\sigma_k - \sum_{i=1}^{N} J_{ik}\sigma_i\sigma_k\right) + \left(-\sum_{i=1}^{N} h_i\sigma_i - \sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N} J_{ij}\sigma_i\sigma_j\right),\end{aligned}$$

implying that the contribution of a spin k to the total energy of the system is given by

$$\mathcal{H}_k(\boldsymbol{\sigma}|\boldsymbol{h},\boldsymbol{J}) = -\sigma_k\left(\sum_{i=1}^{N} J_{ik}\sigma_i + h_k\right). \tag{3.16}$$

After flipping this spin k, the energy associated with this spin becomes

$$\left(\mathcal{H}_k(\boldsymbol{\sigma}|\boldsymbol{h},\boldsymbol{J})\right)_{k \text{ flipped}} = \sigma_k\left(\sum_{i=1}^{N} J_{ik}\sigma_i + h_k\right),$$

which implies that the change in energy after flipping a spin in our case, is given by

$$\Delta\mathcal{H}(\boldsymbol{\sigma}|\boldsymbol{h},\boldsymbol{J}) = \Big(\left(\mathcal{H}_k(\boldsymbol{\sigma}|\boldsymbol{h},\boldsymbol{J})\right)_{k \text{ flipped}} - \mathcal{H}_k(\boldsymbol{\sigma}|\boldsymbol{h},\boldsymbol{J})\Big) = 2\sigma_k\left(\sum_{i=1}^{N} J_{ik}\sigma_i + h_k\right).$$

The general outline of Metropolis Hastings algorithm is simple. We initialize a configuration $\boldsymbol{\sigma}$. This can be done in a random fashion (this is called a hot start because $T = \infty$) or in a structured way, all spins are aligned, way by choosing $\boldsymbol{\sigma} = \pm\mathbf{1}$ (a cold start, $T = 0$.) After initialization, we are going to generate states using the Markov chain. Each iteration, a new state $\boldsymbol{\sigma}'$ is generated and proposed by choosing a spin and flipping it. This new state is attained (i.e. the proposal is accepted) with the transition probability $T(\boldsymbol{\sigma} \to \boldsymbol{\sigma}')$ we defined in Equation (3.15). After a large number of iterations, the system will be converged to an equilibrium. When this equilibrium is reached (the process of reaching this equilibrium is called thermalization), we can draw samples form the obtained system. These samples can be used to calculate observables and, subsequently, analyse the model. [72]

In order to obtain a correct approximation of the observables, it is important to consider the time it takes for the Markov Chain to attain its equilibrium: the *equilibration time*. This time depends on the size of the system (i.e. the number of nodes N) and the model and is usually measured in units of sweeps, which equals N (the number of spin updates).

Moreover, after the system has reached its equilibrium, we have to sample from the Markov chain carefully. Due to the nature of the process, each successive state only one flipped spin from the previous, implying that the subsequent samples are not independent but strongly correlated. Therefore, it is important to average over a large number of observables and to sample states that are sufficiently far apart from each other so that they are no longer correlated. This distance can be determined by the *autocorrelation time*, measured in units of sweeps. [73]

Chapter 4

Temperature analysis

As previously discussed, we investigate the concept of criticality by examining how the system's behavior, measured by an order parameter, changes in response to a control variable. Inspired by the work of Ruffini et al. [19], we consider the *system temperature* $T = 1/k_B\beta$ as control parameter for this analysis.

As outlined in the introduction, concatenating the observed neural signals enables us to infer an archetype PMEM for of the neural network. In this analysis, we refine this model for each participant by inferring the system temperature using pseudo-likelihood optimization. This temperature serves as an individual measure for the distance to criticality.

We will perform this procedure for the whole-brain network, consisting of $N = 238$ nodes (ROIs parcellated according to the Brainnetome atlas [26], and for each of the functional networks [59], including the Visual (nr. 1), Somatomotor (nr. 2), Dorsal Attention (nr. 3), Ventral Attention (nr. 4), Limbic (nr. 5), Frontoparietal (nr. 6) and Default network (nr. 7).

This chapter is organized in two sections. In the first section (Section 4.1), we outline the general procedure of this analysis, consisting of inference the model parameters of the PMEM and the system temperature for each of the participants. We apply this method to both the functional networks, as the whole-brain network. In the second section (Section 4.2), we analyze the obtained results.

4.1 Methods

4.1.1 Inferring the archetype PMEM

Prior to inferring the PMEM, we have to prepare the data. Recall that, as described in section 3.2) the PMEM derived on the assumption that the random variables are discrete, or, in terms of neural signals, that a neuron or region of interest (ROI) is either active ($+1$) or inactive (-1).

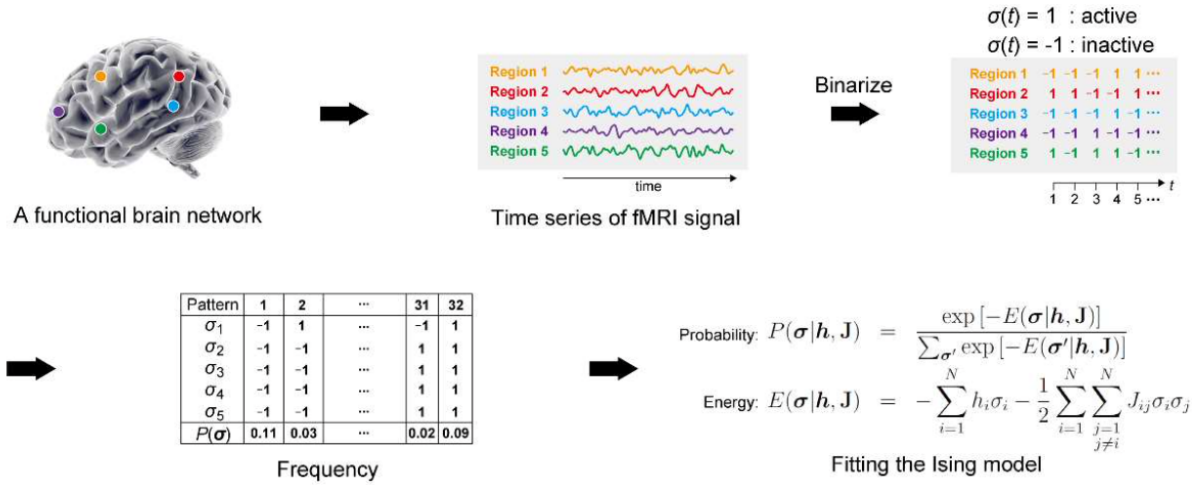


Figure 4.1: Schematic illustration of the steps for inferring the PMEM from the fMRI data, adapted from the Energy Landscape Analysis Toolbox User's Guide of Ezaki and Masuda [74].

Hence, the first step involves binarizing the fMRI data $z_j(t)$ for each ROI $j = 1, \ldots, N$ by using

$$S_j(t) = \begin{cases} +1 & \text{if } \; z_j(t) \geq \tilde{z}_j \\ -1 & \text{if } \; z_j(t) < \tilde{z}_j, \end{cases}$$

for $t = 1, \ldots, B$ the number of samples, where $\tilde{z}_j$ the median of $z_i(1), \ldots, z_j(B)$. This choice for binarization is inspired by a similar study of Ruffini et al. [19]. The binarized activity pattern at time t ($t = 1, \ldots, B$) is denoted by the vector $\boldsymbol{S}(t) = \{S_1(t), \ldots, S_N(t)\} \in \{\pm 1\}^N$. Note that this process needs to be repeated for each individual, resulting in n sets of timeseries $\boldsymbol{S}^i(t)$ ($i = 1, \ldots, n$).

The objective is to infer the archetype PMEM by concatenating the binarized signals $\{\boldsymbol{S}^i(t)\}_{t=1}^{B}$ from all participants ($i = 1, \ldots, n$) into one dataset $\boldsymbol{\mathcal{S}} = \{S^1(1), \ldots, S^1(B), S^2(1), \ldots, S^n(B)\} \in \{-1, 1\}^{N \times (nB)}$.

Using PLM (as introduced in Section 3.3) and assuming $\beta = 1$ without loss of generality, we can infer the PMEM by maximizing the pseudo-likelihood function (given by Equation (3.10)) using the gradient descent updating steps (see Equation (3.11)). The inferred model parameters, denoted by $\hat{\boldsymbol{h}}$ and $\hat{\boldsymbol{J}}$, satisfy the constraints of the PMEM such that the probability of observing a signal $\boldsymbol{S}$ is given by:

$$P(\boldsymbol{S}|\boldsymbol{h},\boldsymbol{J}) = \frac{1}{Z(\boldsymbol{h},\boldsymbol{J})}\exp\left[-\mathcal{H}(\boldsymbol{S}|\boldsymbol{h},\boldsymbol{J})\right],$$

where Z the partition function with

$$Z(\boldsymbol{h},\boldsymbol{J}) = \sum_{\{\boldsymbol{S}\}}\exp\left[-\mathcal{H}(\boldsymbol{S}|\boldsymbol{h},\boldsymbol{J})\right]$$

and the Hamiltonian

$$\mathcal{H}(\boldsymbol{S}|\boldsymbol{h},\boldsymbol{J}) = -\sum_{i=1}^{N} h_i S_i - \frac{1}{2}\sum_{j=1,j\neq i}^{N} J_{ij}S_iS_j.$$

4.1.2 Inferring the 'personal' system temperature T

Given the model parameters $\hat{\boldsymbol{h}}$ and $\hat{\boldsymbol{J}}$, inferred using the concatenated dataset, we adjust the PMEM for each participant by reintroducing the system temperature $\beta = 1/k_B T$ and defining the Boltzmann probability distribution as

$$P(\boldsymbol{\sigma}|\boldsymbol{h}, \boldsymbol{J}, \beta) = \frac{1}{Z(\boldsymbol{h}, \boldsymbol{J}, \beta)} \exp\left[-\beta \mathcal{H}(\boldsymbol{\sigma}|\boldsymbol{h}, \boldsymbol{J})\right], \tag{4.1}$$

where $Z(\boldsymbol{h}, \boldsymbol{J}, \beta)$ the partition function given by

$$Z(\boldsymbol{h}, \boldsymbol{J}, \beta) = \sum_{\{\boldsymbol{\sigma}\}} \exp\left[-\beta \mathcal{H}(\boldsymbol{\sigma}|\boldsymbol{h}, \boldsymbol{J})\right].$$

Using the framework of pseudo-likelihood maximization for inferring β we derived before in Section 3.3.6, we can approximate this system temperature $T = 1/k_B\beta$ with the gradient descent algorithm that is defined by

$$\beta^{\text{new}} = \beta^{\text{old}} - \eta \left(\langle \mathcal{H}(\boldsymbol{\sigma}|\hat{\boldsymbol{h}}, \hat{\boldsymbol{J}}) \rangle_{\text{empirical}} - \langle \overline{\mathcal{H}(\boldsymbol{\sigma}|\hat{\boldsymbol{h}}, \hat{\boldsymbol{J}})} \rangle_{\tilde{P}} \right), \tag{4.2}$$

where

$$\langle \mathcal{H}(\boldsymbol{\sigma}|\boldsymbol{h}, \boldsymbol{J}) \rangle_{\text{empirical}} = \frac{1}{B} \sum_{t=1}^{B} \sum_{i=1}^{N} \mathcal{H}_i(\boldsymbol{\sigma}(t)|\boldsymbol{h}, \boldsymbol{J}) = \frac{1}{B} \sum_{t=1}^{B} \sum_{i=1}^{N} \sigma_i(t) \left(h_i + \sum_{j=1, j\neq i}^{N} J_{ij} \sigma_j(t) \right)$$

the empirical average of the Hamiltonian, and

$$\langle \overline{\mathcal{H}(\boldsymbol{\sigma}|\boldsymbol{h}, \boldsymbol{J})} \rangle_{\tilde{P}} = \sum_{i=1}^{N} \left(h_i \langle \overline{\sigma_i} \rangle_{\tilde{P}} + \sum_{j=1, j\neq i}^{N} J_{ij} \langle \overline{\sigma_i \sigma_j} \rangle_{\tilde{P}} \right).$$

the (data dependent) model mean.

The procedure of adapting the PMEM for each participant, is quite straightforward. Fixing the model parameters the model parameters $\hat{\boldsymbol{h}}$ and $\hat{\boldsymbol{J}}$, we personalize the model by inferring the temperature $\hat{T}$ for each participant using the learning rate.

When varying the system temperature T induces a phase transition, then this quantity can be used as a measure for the distance to criticality. The point $T = T_c$ where the phase transition occurs, can be investigated by sampling from the archetype PMEM for different values of T and calculating the observables for this model. The distance to criticality of an individual is then defined as $\hat{T} - T_c$.

Note that for this temperature analysis, the bias correction (as introduced in Section 3.4) is not necessary. The correction factor, i.e., fictitious temperature T_f, scales the model parameters by a factor $1/T_f$ ($\boldsymbol{h}/T_f$ and $\boldsymbol{J}/T_f$) and will therefore only shift the obtained system temperature from or to the critical point, but not not change the mutual differences between the participants (the measure of interest). Therefore, we ignore this bias.

4.1.3 Statistical analysis

When it can be shown that the system exhibits a phase transition when varying the system temperature (i.e., there is a critical temperature $T = T_c$), then we analyze the differences in distance to criticality by comparing the inferred 'personal' temperatures.

Functional networks

First, we identify outliers based on exceeding $1.5 \cdot \text{IQR}$ (InterQuartile Range) limits, and we remove these extreme outliers from the analysis. Subsequently, we conduct a mixed analysis of variance (ANOVA), using IBM SPSS Statistics 29.0, to examine whether the distance to criticality (in terms of temperature) is the result of the interaction between the functional network and treatment response with age and education (measured by ISCED) as covariate, as similar studies [16, 58, 75, 76] did. We verify the normality assumption using one-sample Kolmogorov-Smirnov tests, and the sphericity assumption with Mauchly's test of sphericity.

When a covariate turns out to not be significant, we repeat the analysis without that covariate. If there is a significant interaction, we perform for each network a post-hoc analysis of variance (ANOVA) to examine the effect further.

Whole-brain network

After identifying outliers based on exceeding $1.5 \cdot \text{IQR}$ (InterQuartile Range) limits and removing them from analysis, we conduct an analysis of variance (ANOVA) using IBM SPSS Statistics 29.0 to examine the effect of treatment response on distance to criticality (in terms of temperature) with age and education (measured by ISCED) as covariates. We verify the normality assumption using one-sample Kolmogorov-Smirnov tests and will only consider covariates that are significant.

4.2 Results

4.2.1 Analysis of the functional networks

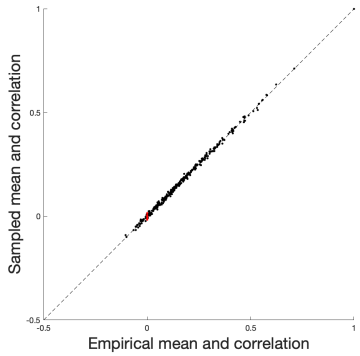
Inferring the PMEM and 'personal' system temperatures

The PMEM is fitted to the concatenated timeseries of all participants for each functional network with learning rate $\gamma = 0.05$.

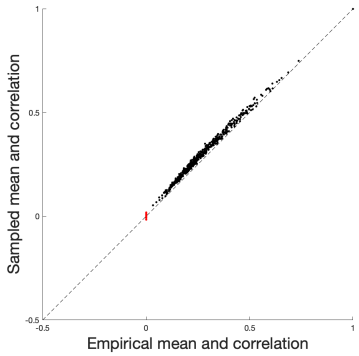
We validate the accuracy of this method by comparing, for each ROI i ($i = 1, \ldots, N$), the mean $\langle S_i \rangle_{\text{Empirical}}$ and correlation $\langle S_i S_j \rangle_{\text{Empirical}}$ between the empirical data and a simulated dataset using the inferred parameters $\hat{\boldsymbol{h}}$ and $\hat{\boldsymbol{J}}$, as proposed by Ezaki et al. [16]. The method is accurate (i.e., we estimated the parameters $\boldsymbol{h}$ and $\boldsymbol{J}$ well) when the estimated mean and correlation closely align with the empirical mean and correlation. These results are summarized in Figure 4.2.

Simulating the archetype model as a function of the system temperature T, i.e. using the temperature dependent Boltzmann distribution (4.1), shows us that tuning T forces the system to shift from paramagnetic to ferromagnetic phase, as both the uniform as the spin-glass susceptibility, χ_{Uni} resp. χ_{SG} diverge.

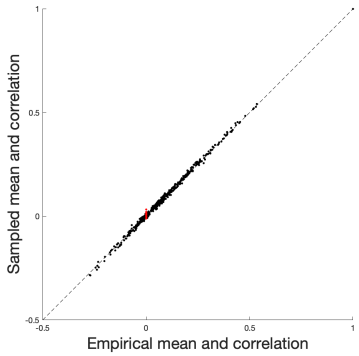
We performed this experiment both for the whole-brain network, as well for the functional networks using the Metropolis-Hastings algorithm (see Section 3.5) with 10^7 thermalization sweeps and collecting $B = 46 \times 310$ samples (i.e., the size of the concatenated dataset) every 1000 sweeps, as proposed by several similar studies [16, 51, 56] to ensure proper samples. To correct for variability, we averaged over 10 independent simulations. The results are shown in Figure 4.3.



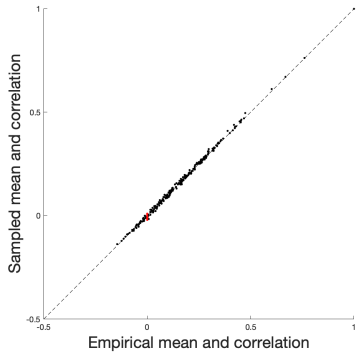
Functional network 1.



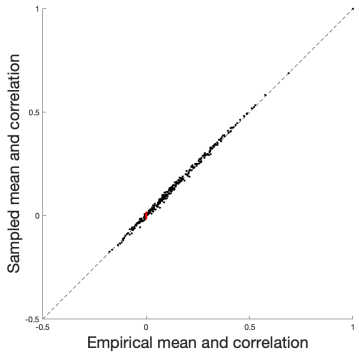
Functional network 2



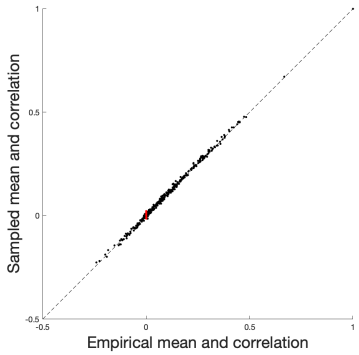
Functional network 3.



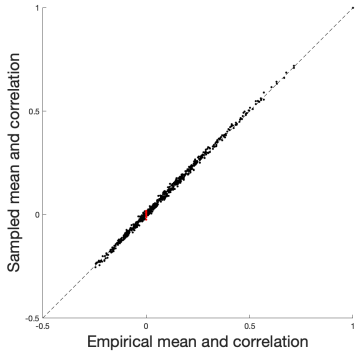
Functional network 4.



Functional network 5.

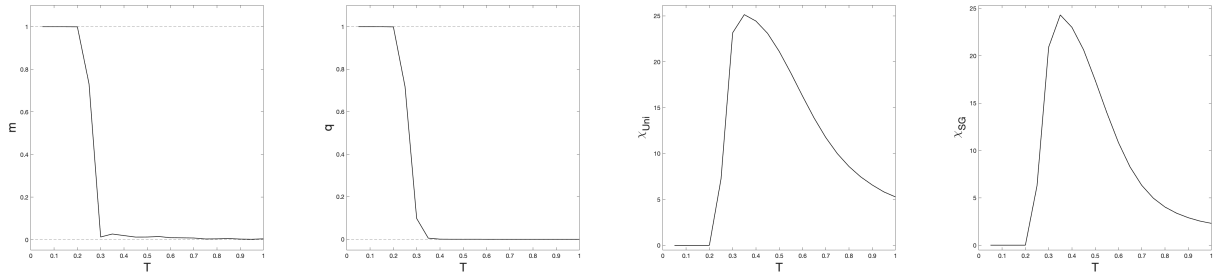


Functional network 6.

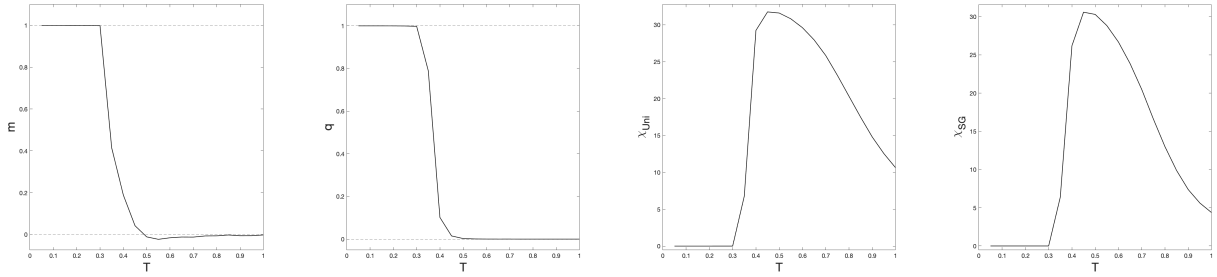


Functional network 7.

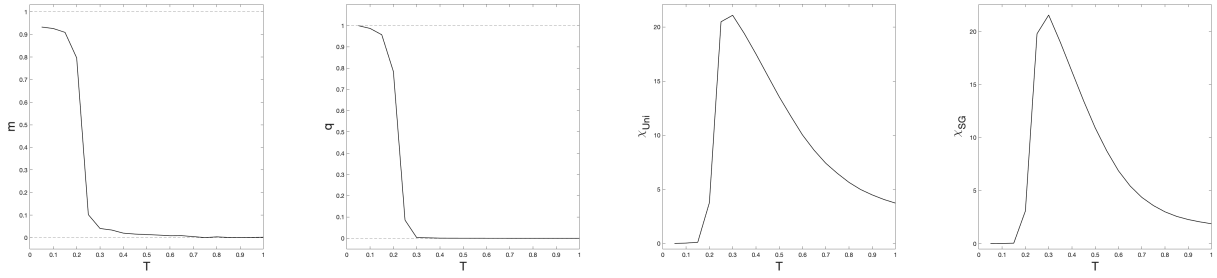
Figure 4.2: Accuracy of PLM, considered by comparing the empirical and sampled mean $\langle S_i \rangle$ and correlation $\langle S_i S_j \rangle$ for each ROI i $(i = 1, \ldots, N)$.



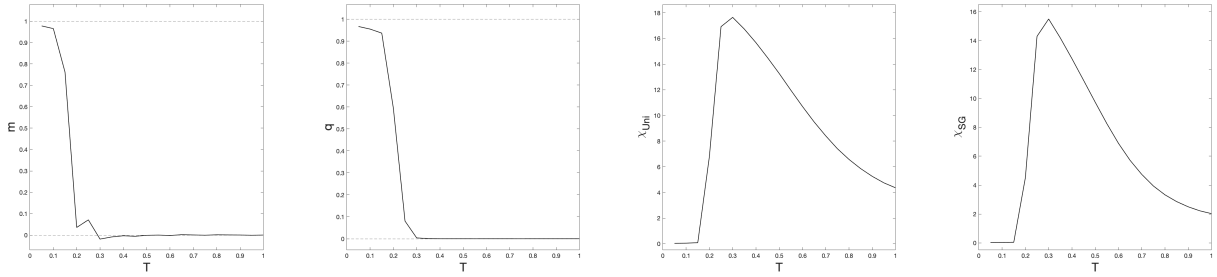
Functional network 1.



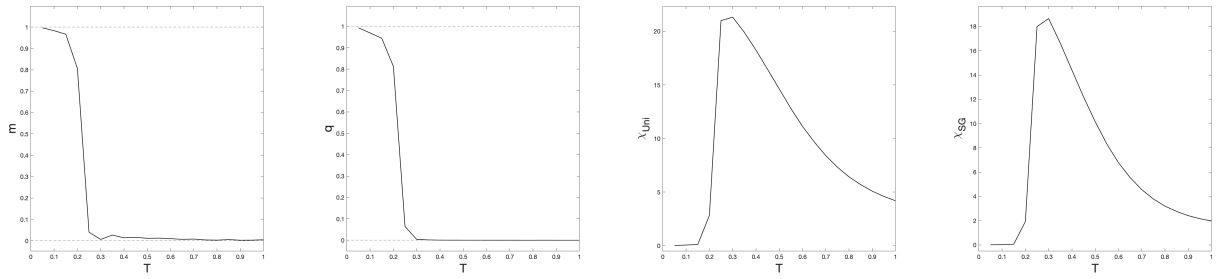
Functional network 2.



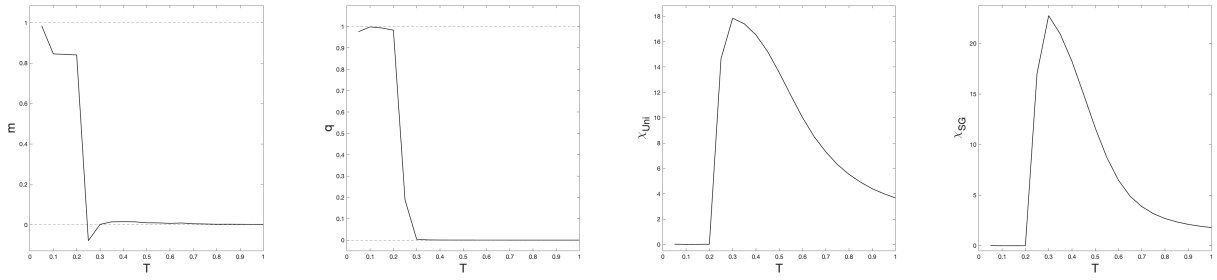
Functional network 3.



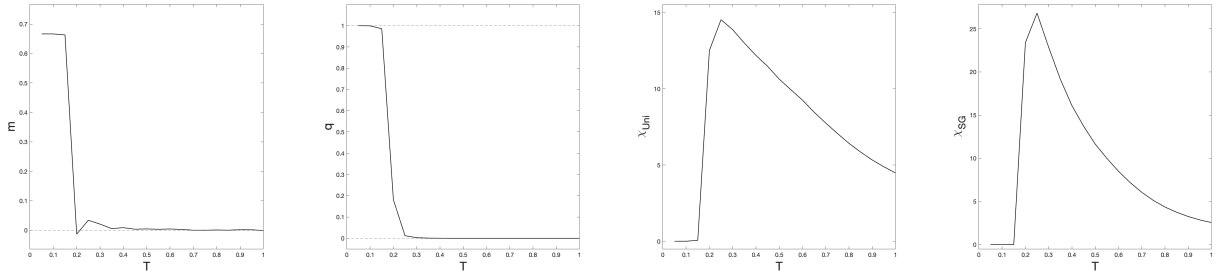
Functional network 4.



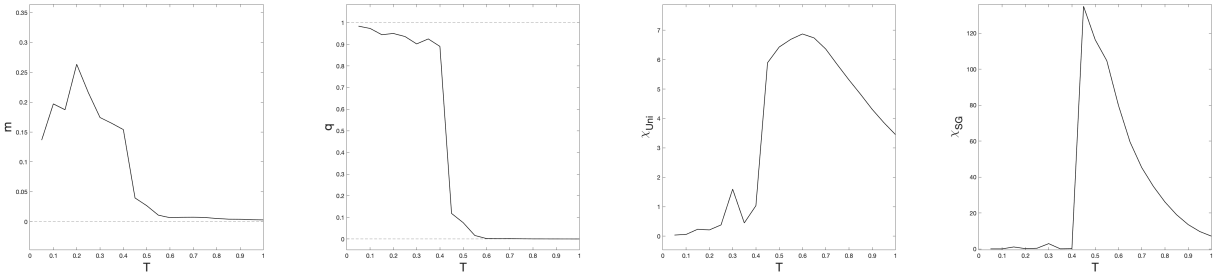
Functional network 5.



Functional network 6.



Functional network 7.



Whole-brain network

Figure 4.3: Plot of the observables m, q, χ_{Uni} and χ_{SG} respectively, as a function of the system temperature T for the functional networks and the whole-brain network.

Considering the uniform susceptibility χ_{Uni} and spin-glass susceptibility χ_{SG} (as defined in Section 3.1), we observe that these, both for the whole-brain network and functional networks, diverges at a certain point $T = T_c$. This implies that when temperature increases, the system undergoes a phase transition form ferro- to paramagnetic phase. Moreover, for $T < T_c$, we observe that the magnetization $m > 0$ and spin-glass order parameter $q > 0$ implying that the system is in ferromagnetic phase and when $T > T_c$ both, magnetization $m = 0$ and spin-glass order parameter $q = 0$, i.e. the system is in paramagnetic phase.

To ensure that the observed phase transitions are not resulting from the tendency to align with the external field $\boldsymbol{h}$, we also consider the absence of the external field, i.e., $\boldsymbol{h} = 0$. This analysis can be found in the Appendix B.1.1. Here, we observe qualitatively the same graphs for the observables as a function of the system temperature T.

We can conclude that an increasing temperature the system shifts from ferro- to paramagnetic phase. For the whole-brain network, this phase transition occurs for $T_c \sim 0.5$ and for the functional networks for $T_c \sim 0.3 - 0.4$.

Inferring the 'personal' system temperature

After that, we 'personalized' this archetype PMEM for each of the participants using their 'own' neural data. We applied the gradient ascent updating steps (see Equation (4.2)) with learning parameter $\eta = 0.01$. The obtained 'personal' system temperatures $\hat{T}$ are shown in Figure 4.4.

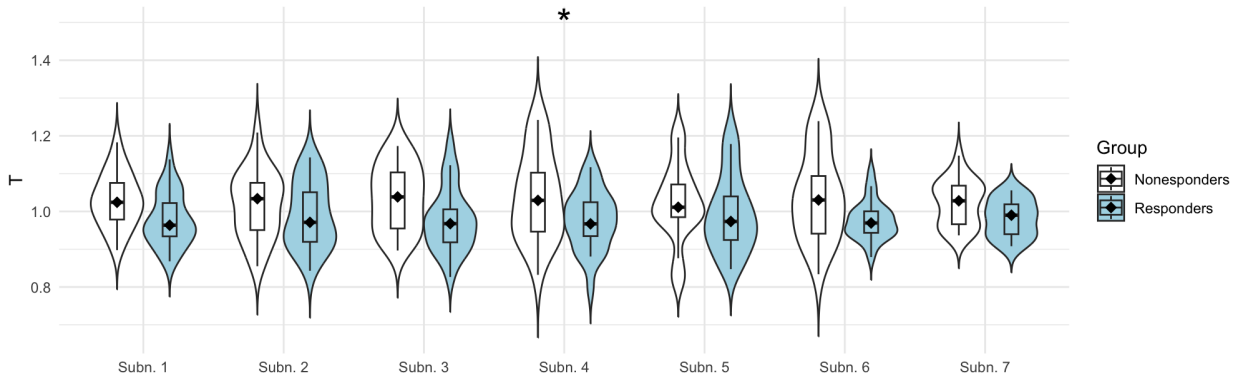


Figure 4.4: Inferred 'personal' system temperatures T, depicted for each of the functional networks.
$*$ A main effect was observed ($P = 0.014$).

Statistical analysis

First, 5 outliers (3 from the non-responder, and 2 of the responder group) were identified and removed from analysis. The normality assumption was verified for both groups using one-sample Kolmogorov-Smirnov tests, which were not significant for any of the functional networks. Education (ISCED) was not a significant covariate and therefore removed from analysis.

However, there was a significant main effect of the covariate, age, on the system temperature ($F(1, 38) = 15.405$, $p < 0.001$), implying that age is associated with the system temperature. The sphericity assumption was tested using Mauchly's test, which was significant ($\chi^2(20) = 39.338$, $P = 0.006$). Therefore, the Greenhouse-Geisser correction was applied ($\varepsilon = 0.749$).

There was no significant interaction effect between functional network and the covariate, age, on the temperature (($F(4.492, 170.692) = 2.065$, $P = 0.079$). This implies that the effect of age on the temperature was similar for the seven functional networks. There was no significant interaction effect between functional network and treatment response on the system temperature ($F(4.492, 170.692) = 0.701$, $P = 0.608$), implying that there are no significant differences between networks between the two groups. There was no significant main effect of functional network on the temperature ($F(4.492, 170.692) = 2.067$, $P = 0.079$). This means that the temperature does not vary for the functional networks.

However, there was a significant main effect between treatment response and the system temperature T ($F(1, 38) = 0.098$, $P = 0.014$). This implies that treatment response does depend on the system temperature.

A statistical comparison of the demographics and clinical data revealed no significant differences between the two groups at the start of the study. This means that the two groups can be considered as equivalent at the start of treatment. By the end of the study, the non-responder group had significantly higher clinical scores (measured by the Clinician Administered PTSD Scales (CAPS)), defining the difference in treatment response. Furthermore, the non-responder group exhibited a significantly higher frequency of (comorbid) anxiety disorders and usage of SSRIs (commonly prescribed for depression) post-treatment. This can be expected as depressive, substance use, and other anxiety disorders are comorbid with PTSD [77]. These demographics and clinical data and statistical comparisons are shown in Table 4.1.

We can conclude that there is an association between treatment response and the 'personal' system temperature. Specifically, responders to psychotherapy exhibit a significantly lower system temperature compared to non-responders, indicating that they are closer to criticality. This is in line with what we hypothesized. Additionally, there are no significant differences between the two groups in demographics and clinical data at the start of treatment. Therefore, we may conclude that treatment response could be explained by the distance to criticality calculated (using an archetype PMEM and the 'personal' system temperature). However, because only a main effect is measured, we are not able to identify functional networks that are more involved than others.

4.2.2 Analysis of the whole-brain network

Inferring the PMEM and 'personal' system temperatures

As for the functional networks, we fitted the PMEM to the concatenated timeseries of all participants for each functional network with learning rate $\gamma = 0.05$. The accuracy is summarized in Figure 4.5a. Subsequently the 'personal' system temperatures were inferred using, again, learning rate $\eta = 0.01$. The obtained temperatures are shown in Figure 4.5b.

	Responders ($n = 22$)	**Non-responders ($n = 19$)**	**Test-value (df)**	**P-value**
Age (median, IQR [years])	35 (12)	35 (18)	$U = 151$	$P = 0.129^{a}$
Gender (m/f)	22/0	19/0		
Handedness (left/right/ambidextrous)	1/19/2	2/15/2	0.792	$P = 0.838^{b}$
Education (0/1/2/3/5 *[ISCED])*				
Own	0/0/2/15/4	0/0/7/11/1	$U = 146$	$P = 0.054^{a}$
Mother	1/0/13/4/3	1/2/8/3/3	$U = 169$	$P = 0.758^{a}$
Father	0/1/9/3/7	1/1/4/6/6	$U = 169.5$	$P = 0.747^{a}$
Time since last deployment (median, IQR [months])	38 (167)	60 (126)	$U = 181$	$P = 0.644^{a}$
Number of times deployed (median, IQR)	2 (4)	2 (2)	$U = 177.5$	$P = 0.558^{a}$
Early traumatic experiences (median, IQR [ETI total])	3(3.5)	3.5 (7.5)	$U = 175$	$P = 0.883^{a}$
Therapy received				
EMDR/tfCBT/EMDR and tfCBT	16/4/2	10/4/5	2.427	$P = 0.288^{b}$
Total number of therapy sessions (median, IQR)	6.0 (9.0)	8.0 (5.0)	$U = 126.5$	$P = 0.265^{a}$
Therapy received prior to scanning and CAPS				
No therapy/EMDR/tfCBT/EMDR and tfCBT	16/4/2/0	12/3/3/1	1.756	$P = 0.782^{b}$
Total number of therapy sessions (median IQR)	0.0 (1.0)	0.0 (4)	$U = 175$	$P = 0.281^{a}$
Baseline clinical scores				
Clinical scores at baseline (mean, SD [CAPS])				
Total	70.68 (15.126)	69.47 (11.467)	$U = 200.5$	$P = 0.824^{a}$
Re-experiencing	22.909 (4.849)	22.316 (6.237)	$U = 194$	$P = 0.694^{a}$
Avoiding	23.636 (11.290)	23.105(6.839)	$U = 203.5$	$P = 0.886^{a}$
Hyperarousal	24.136 (5.213)	24.053 (3.865)	$U = 202.5$	$P = 0.864^{a}$
Comorbid disorder baseline (no. [SCID])				
Mood disorders	12	11	$\chi^2(1) = .046$	$P = 0.829^{c}$
Schizophrenia and other psychotic disorders	1	0	·	$P = 1.000^{b}$
Substance-related disorders	1	1	·	$P = 1.000^{b}$
Anxiety disorders	7	9	$\chi^2(1) = 1.036$	$P = 0.309^{c}$
Somatoform disorders	1	1	·	$P = 1.000^{b}$
Baseline medication (no.)				
SSRI	3	6	·	$P = 0.260^{b}$
Benzodiazepines	7	3	·	$P = 0.292^{b}$
SARI	1	0	·	$P = 1.000^{b}$
Antipsychotics	2	0	·	$P = 0.490^{b}$
β-blockers	0	2	·	$P = 0.209^{b}$
Nicotine agonists	1	0	·	$P = 1.000^{b}$
Ritalin	0	0		

	Responders ($n = 22$)	**Non-responders ($n = 19$)**	**Test-value (df)**	**P-value**
Clinical scores post-treatment (median, IQR [CAPS])				
Total	32 (25)	63 (16)	$U = 12.5$	$P < 0.001$[a*]
Re-experiencing	7.5 (13.5)	23 (7)	$U = 39.0$	$P < 0.001$[a*]
Avoiding	6 (10.25)	17 (16)	$U = 36.0$	$P < 0.001$[a*]
Hyperarousal	13 (10.75)	21 (11)	$U = 48.5$	$P < 0.001$[a*]
Comorbid disorder post-treatment (no. [SCID])				
Mood disorders	4	8	$\chi^2(1) = 3.252$	$P = 0.071$[c]
Schizophrenia and other psychotic disorders	0	1	·	$P = 0.450$[b]
Substance-related disorders	0	2	·	$P = 0.196$[b]
Anxiety disorders	3	8	·	$P = 0.040$[c*]
Somatoform disorders	0	1	·	$P = 0.450$[b]
Post-treatment medication (no.)				
SSRI	4	9	$\chi^2(1) = 4.011$	$P = 0.045$[c*]
Benzodiazepines	6	2	·	$P = 0.249$[b]
SARI	1	0	·	$P = 1.000$[b]
Antipsychotics	2	1	·	$P = 1.000$[b]
β-blockers	0	0		
Nicotine agonists	0	0		
Ritalin	0	0		

SD, standard deviation; IQR, interquartile range; ISCED, international scale for education; CAPS, clinician administered PTSD scale; SCID, structured clinical interview for DSM IV Axis II disorders; SSRI, serotonin reputake inhibitor; SARI, serotonin antagonist and reuptake inhibitors; EMDR, eye movement desensitization and reprocessing; tf-CBT, trauma-focused cognitive behavioral therapy.

· No test-value is provided by SPSS for 2×2 Fisher's exact tests.

* $P < 0.05$.

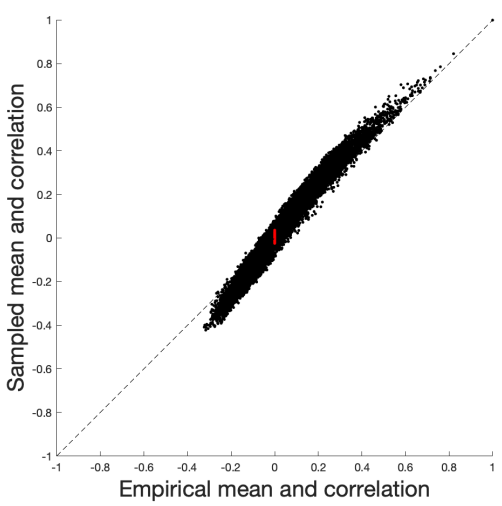
Number of cases with missing data: education mother responders $n = 1$; education mother non-responders $n = 2$; education father responders $n = 1$; education father non-responders $n = 1$; time since last deployment non-responders $n = 1$; number of times deployed non-responders $n = 1$; ETI responders $n = 2$; ETI non-responders $n = 1$; total number of therapy sessions responders $n = 5$; SCID post-treatment non-responders $n = 1$.

[a] Mann-Whitney U test.

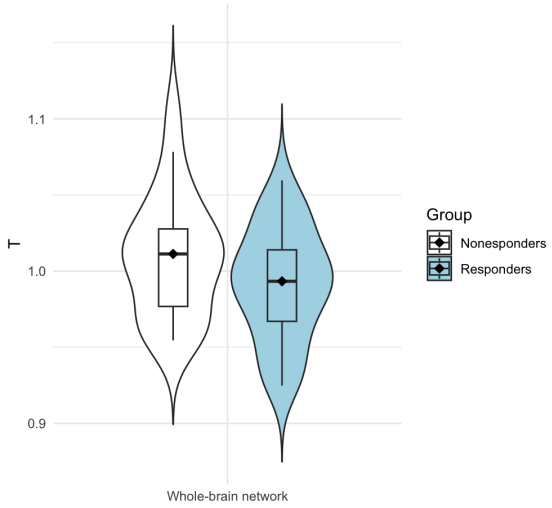
[b] Fisher's exact test.

[e] χ^2-test

Table 4.1: The demographics at the start of the study and after treatment.



(a) Accuracy PLE whole-brain network.



(b) Inferred 'personal' system temperatures T for the whole-brain network.

Figure 4.5

Statistical analysis

No outliers were identified and the normality assumption was met for each group (tested using a one-sample Kolmogorov-Smirnov test, which was not significant for both groups). Education (ISCED) was not a significant covariate and therefore removed from analysis. However, there was a significant main effect of the covariate, age, on the temperature ($F(1, 43) = 15.837$, $p < 0.001$). This implies that the effect of age on the temperature was similar for the seven functional networks.

There was no significant main effect of treatment response on the distance to criticality ($F(1, 43) = 0.672$, $P = 0.417$). This implies that treatment response does not depend on the system temperature for the whole-brain network.

A statistical comparison of the demographics and clinical data at the start of the study revealed a significant difference between the two groups in education level (measured by ISCED) ($U = 186, P = 0.042$.) There were no significant differences between the two groups on the other test-values observed ($0.079 \leq P \leq 1.000$).

By the end of the study, the non-responder group did have significantly higher clinical scores (measured by the Clinician Administered PTSD Scales (CAPS)) ($P < 0.001$), defining the difference in treatment response between the two groups. Furthermore, the non-responder group exhibited a significantly higher frequency of comorbid mood disorders ($\chi^2(1) = 5.008, P = 0.025$), anxiety disorders ($\chi^2(1) = 6.724, P = 0.010$), and SSRI usage ($\chi^2(1) = 3.919, P = 0.048$). As mood, depressive, substance use, and other anxiety disorders are comorbid with PTSD [77, 78], this can be expected.

We conclude, based on the statistical tests, that the distance to criticality, measured by the 'personal' system temperature, is not significantly different between the two groups of participants.

Moreover, an analysis of the demographics and clinical data and clinical data of the two groups revealed a significant difference in education level between the groups meaning that the groups cannot be completely considered as equivalent. Consequently, treatment response cannot be explained by the distance to the ferromagnetic phase transition calculated using an archetype PMEM for the whole-brain network and the corresponding phase diagrams

Chapter 5

Phase diagram analysis

For the second approach, inspired by Ezaki et al. [16], we parameterize the archetype PMEM inspired by the Sherrington-Kirkpatrick model [57] by linearly transforming the interaction matrix $\boldsymbol{J}$. We will see that the used parameters, equivalent to the mean and standard deviation of the interaction matrix when this would be Gaussian distributed, induces three qualitatively distinct types of behavior within the model, corresponding to regions in the phase diagram: a ferromagnetic, paramagnetic and spin-glass phase. Hence, these parameters function as control parameters of the model. The objective is to personalize the archetype PMEM for each participant by approximating these model parameters.

As for the temperature analysis, we infer the PMEM for the concatenated timeseries of all participants by assuming without loss of generality $\beta = 1$. This gives us the archetype PMEM that can be used to draw phase diagrams based on the observables for the SK-model, including the magnetization m, spin-glass order parameter q, spin-glass susceptibility χ_{SG} and uniform susceptibility χ_{Uni}. Furthermore, it is possible to calculate the spin-glass and uniform susceptibility (χ_{SG} resp. χ_{Uni}, as defined in section 3.1) for each participant directly from the data. The combination of these two observations provides us a procedure to locate the position of a participant in the phase diagram, and thereby which parameter combination reflects best the individuals neural data. Additionally, this position in the phase diagrams serves as a measure for the distance from a critical phase transition.

Note that the 'personal' temperature is not relevant for this analysis. Personalizing the archetype PMEM would generate unique models for each participant, and thereby distinct phase diagrams. Since we utilized the archetype phase diagrams as the common factor to quantify the distance to criticality, direct comparison between participants in this manner is no longer feasible.

We will perform the procedure, as we also did for the system temperature analysis, for the whole-brain network, consisting of $N = 238$ nodes (ROIs), and for each of the functional networks, including the Visual (nr. 1), Somatomotor (nr. 2), Dorsal Attention (nr. 3), Ventral Attention (nr. 4), Limbic (nr. 5), Frontoparietal (nr. 6) and Default network (nr. 7).

This chapter is organized in two sections. In the first section (Section 5.1), we introduce the general procedure, inspired by Ezaki et al. [16], of this analysis, consisting of inference the PMEM, the drawing of phase diagrams and the calculation of the distance to criticality. The second section (Section 5.2) is dedicated to discussing the obtained results and a statistical analysis of them.

5.1 Methods

5.1.1 Inferring the PMEM

For this method, we infer the archetype PMEM using the same concatenated timeseries as for the temperature analysis. Therefore, the inference is exactly the same and the details can be found in Section 4.1.1.

However, for this analysis the bias correction as explained in section 3.4, is relevant: although the PLM is considered one of the most accurate methods for inferring the PMEM, a residual bias may still exist and could influence the results. As we are going to parametrize the model in another way than adjusting the system by a constant factor (i.e. the system temperature), we have to correct for this bias. Therefore, as an additional step, we correct the obtained PMEM for these biases by applying the C^2-correction, as proposed by Kloucek et al. [51] and described in Section 3.4. Recall that this C^2 correction consists of optimizing the objective function

$$\mathcal{L}(T_f) = \left(C^2_{\text{empirical}} - C^2_{\text{MC}}(T_f)\right)^2,$$

where $C^2 = \frac{1}{N}\sum_{i,j} C^2_{ij}$. We will utilize the Fibonacci search algorithm for this, where we evaluate the objective by sampling from the Boltzmann distribution with temperature T_f using the Metropolis-Hastings algorithm (see Section 3.5). For this, we used, again, 10^7 thermalization sweeps and collected $B = 46 \times 310$ samples (i.e., the size of the concatenated dataset) every 1000 sweeps, as proposed by several similar studies [16, 51, 56] to ensure proper samples. To correct for variability, we averaged over 30 samples per evaluation.

The resulting fictitious temperature T_f that optimizes the objective, should improve the model and the corrected model parameters are given by $\hat{\boldsymbol{h}}/T_f$ and $\hat{\boldsymbol{J}}/T_f$.

5.1.2 Drawing phase diagrams

To characterize the behavior of the brain and to measure the distance to a phase transition for each participant, we need to parameterize the model. As previously discussed (in the introduction and section 3.1 about the SK-model), in case of SK-model, the mean and standard deviation of the interaction matrix $\boldsymbol{J}$ determine the order of the model. Inspired by this observation, we parameterize the archetype PMEM by linearly transforming the inferred interaction matrix $\hat{\boldsymbol{J}}$ by the parameters μ and σ (corresponding to the mean and standard deviation in case of the SK-model) using the affine linear transformation

$$J_{ij} = (\hat{J}_{ij} - \hat{\mu})\frac{\sigma}{\hat{\sigma}} + \mu,$$

where $\hat{\mu}$ the mean and $\hat{\sigma}$ the standard deviation of the off-diagonal elements of $\hat{\boldsymbol{J}}$. (Note that when it holds that $\hat{J}_{ij} \sim \mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$, then, the linear transformation with parameters μ, σ would imply that $J_{ij} \sim \mathcal{N}(\mu, \sigma^2)$.) Additionally, we keep the external field $\boldsymbol{h} = \hat{\boldsymbol{h}}$ fixed.

The latter is permitted as it can be demonstrated that the external field $\boldsymbol{h}$ does not qualitatively change the observables/phase diagrams. A justification is given in the supplementary material (see Appendix B.1.2).

By varying the parameters μ and σ across a range of values, we obtain a set of functional brain networks that we can analyze by generating samples from the model using the Metropolis-Hastings algorithm (as introduced in section 3.5). For this, we used, as proposed by several similar studies [16,51,56], 10^7 thermalization sweeps and collected $B = 46 \times 310$ samples (i.e., the size of the concatenated dataset) every 1000 sweeps to ensure proper samples. Additionally, to account for variability, the results were averaged over 10 independent experiments.

We analyze the obtained samples by calculating the observables of the SK-model, including the magnetization m, spin-glass parameter q, uniform susceptibility χ_{Uni} and spin-glass susceptibility χ_{SG} (as introduced in section 3.1). These observables are summarized in phase diagrams, which provide insights into how the system behaves for each parameter combination (μ, σ).

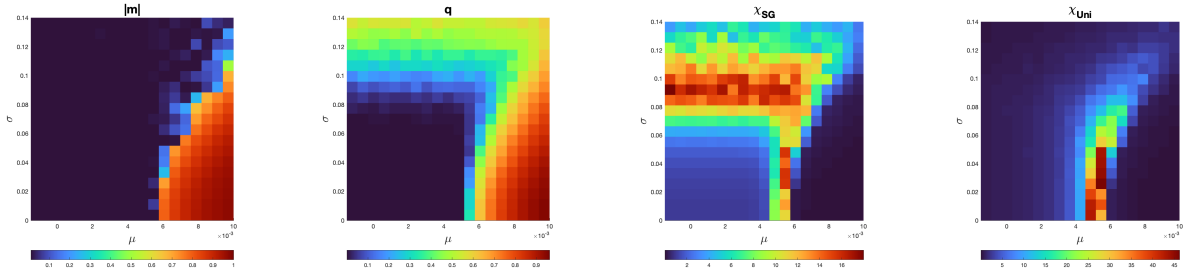


Figure 5.1: Phase-diagrams for the Sherrington-Kirkpatrick model, where the couplings $J_{ij} \sim \mathcal{N}(\mu, \sigma^2)$. Here, for each parameter combination (μ, σ) the corresponding value of the magnetization m, spin-glass susceptibility q, spin-glass susceptibility χ_{SG} and uniform susceptibility χ_{Uni} is represented. The diagrams were obtained using the software provided by Ezaki et al. [16, 79] for a system of size $N = 238$.

An illustrative example of the phase diagrams for the SK-model is given in Figure 5.1. Here, we parameterized the SK-model by the mean μ and standard deviation σ of the interaction model $\boldsymbol{J}$ and with zero external field (i.e. $\boldsymbol{h} = 0$. We will see later that the phase diagrams computed from the archetype PMEM are analogous to diagrams corresponding to the SK-model. In these phase diagrams, three phases can be identified by looking at the magnetization m and spin-glass order parameter q, each corresponding to a qualitatively distinct state of the system. Moreover, note that, as described before, the uniform respectively spin-glass susceptibility diverges at the boundaries of the phases. At these curves, the system is said to be at criticality.

5.1.3 Locating position participant in phase diagrams

We adapt the archetype PMEM by locating the position of each participant in the phase diagram. This involves identifying the model (or, actually, functional network) with the parameter combination (μ, σ) that best describes the brain activity patterns. We denote the estimated position by $(\tilde{\mu}, \tilde{\sigma})$.

The procedure relies on the phase diagrams corresponding to the archetype PMEM (inferred by the concatenated timeseries) and the values of the uniform and spin-glass susceptibility $\chi_{\text{SG}} = N^{-1} \sum_{i,j=1}^{N} c_{ij}^2$ resp.

$\chi_{\text{Uni}} = N^{-1} \sum_{i,j=1}^{N} c_{ij}$, with $c_{ij} = \langle S_i S_j \rangle - \langle S_i \rangle \langle S_j \rangle$, that can be directly calculated from the individual data (denoted by $\tilde{\chi}_{\text{SG}}$ and $\tilde{\chi}_{\text{Uni}}$) without inferring the PMEM.

We analyze the generated phase diagrams in terms of χ_{SG} and χ_{Uni}, obtained by sampling $\chi_{\text{SG}}(\mu, \sigma)$ resp. $\chi_{\text{Uni}}(\mu, \sigma)$ using the inferred model parameters $\hat{\boldsymbol{h}}$ and $\hat{\boldsymbol{J}}$ at the points $(\mu, \sigma) = (\mu_k, \sigma_\ell)$ $(k = 1, \ldots, m$ and $\ell = 1, \ldots, n$ $)$. This gives us two surfaces as illustrated in Figure 5.2. The aim of the procedure is to determine the point $(\tilde{\mu}, \tilde{\sigma})$ where it holds that both $\chi_{\text{SG}}(\tilde{\mu}, \tilde{\sigma}) = \tilde{\chi}_{\text{SG}}$ and $\chi_{\text{Uni}}(\tilde{\mu}, \tilde{\sigma}) = \tilde{\chi}_{\text{Uni}}$. For this parameter combination, the archetype model with interactions

$$\hat{J}_{ij} = (\hat{J}_{ij} - \hat{\mu}) \frac{\tilde{\sigma}}{\hat{\sigma}} + \tilde{\mu}$$

would reflect best the observed neural signals of the participant. Note that this combination $(\tilde{\mu}, \tilde{\sigma})$ can be found by considering the intersection of the level-sets $L_{\tilde{\chi}_{\text{SG}}} = \{(\mu, \sigma) | \chi_{\text{SG}}(\mu, \sigma) = \tilde{\chi}_{\text{SG}}\}$ and $L_{\tilde{\chi}_{\text{Uni}}} = \{(\mu, \sigma) | \chi_{\text{Uni}}(\mu, \sigma) = \tilde{\chi}_{\text{Uni}}\}$. These level-sets are depicted in Figure 5.2 by the blue resp. red curve. Assuming that there is only one point where these intersects, it should hold that $(\tilde{\mu}, \tilde{\sigma}) = L_{\tilde{\chi}_{\text{SG}}} \cap L_{\tilde{\chi}_{\text{Uni}}}$.

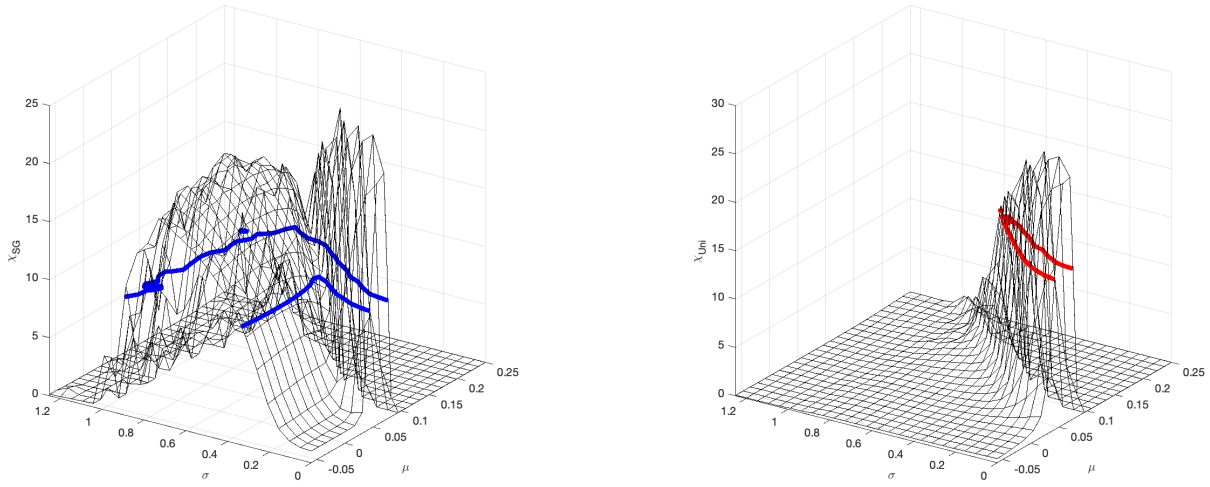


Figure 5.2: The surfaces of $\chi_{\text{SG}}(\mu, \sigma)$ and $\chi_{\text{Uni}}(\mu, \sigma)$ obtained by simulating the model for different parameter combinations (μ, σ). The piecewise linear curves at which it holds that $\chi_{\text{SG}} \approx \tilde{\chi}_{\text{SG}}$ and $\chi_{\text{Uni}} \approx \tilde{\chi}_{\text{Uni}}$ are depicted by a blue resp. red curve.

To determine this intersection from the generated phase diagrams, we use a procedure as follows. For each μ_k $(k = 1, \ldots, m)$ we determine the value of $\overline{\sigma}_k$ such that $\chi_{\text{SG}}(\mu_k, \overline{\sigma}_k) = \tilde{\chi}_{\text{SG}}$ using linear interpolation: $\sigma'_k = \alpha \sigma_{\ell'} + (1 - \alpha) \sigma_{\ell'}$, where ℓ' $(1 \leq \ell' < n)$ satisfies $\chi_{\text{SG}}(\mu_k, \sigma_{\ell'}) \leq \tilde{\chi}_{\text{SG}} < \chi_{\text{SG}}(\mu_k, \sigma_{\ell'+1})$ and $\alpha = [\chi_{\text{SG}}(\mu_k, \sigma_{\ell'+1}) - \tilde{\chi_{\text{SG}}}] / [\chi_{\text{SG}}(\mu_k, \sigma_{\ell'+1}) - \chi_{\text{SG}}(\mu_k, \sigma_{\ell'})]$. This gives us the piecewise linear curve $(\mu_k, \overline{\sigma}_k)$ $(k = 1, \ldots, m)$ (See Figure 5.3).

Subsequently, we calculate the curve for which it holds that $\chi_{\text{Uni}}(\overline{\mu}_\ell, \sigma_\ell) = \tilde{\chi}_{\text{Uni}}$ $(\ell = 1, \ldots, n)$ using the same algorithm as for χ_{SG}. Note that these curves corresponds to the level-sets $L_{\tilde{\chi}_{\text{SG}}}$ and $L_{\tilde{\chi}_{\text{Uni}}}$.

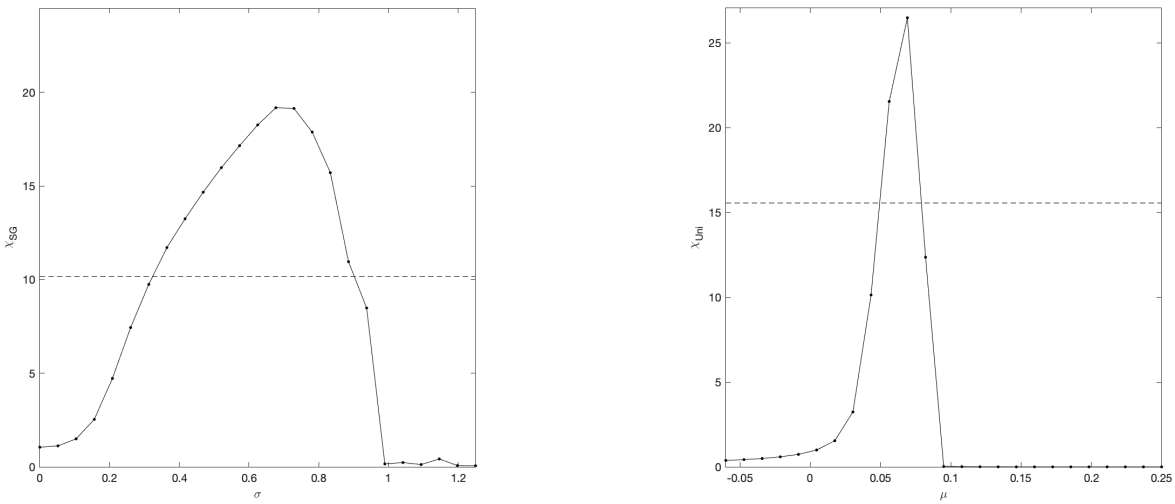


Figure 5.3: The piecewise linear curves $\chi_{\mathrm{SG}}(\mu, \sigma_\ell)$ resp. $\chi_{\mathrm{Uni}}(\mu_k, \sigma)$ $(k = 1, \ldots, m$ and $\ell = 1, \ldots, n)$.

Now, we estimate $(\tilde{\mu}, \tilde{\sigma})$ by examining the intersections of these two curves. An example is shown in Figure 5.4.

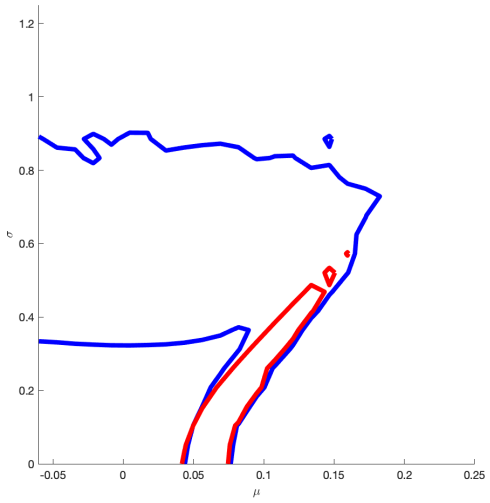


Figure 5.4: The piecewise linear curves in the (μ, σ)-plane.

Note that there is a possibility that the curves intersect more than two times. In that case, we consider the point $(\tilde{\mu}, \tilde{\sigma})$ for which it holds that $m(\tilde{\mu}, \tilde{\sigma}) \approx \tilde{m}$ and $q(\tilde{\mu}, \tilde{\sigma}) \approx \tilde{q}$ using, again, linear interpolation. Here, $\hat{m}$ and $\hat{q}$ are directly calculated from the timeseries.

5.1.4 Computing the distance from criticality

After we have determined the participant's position in the phase diagram, and thereby for which parameter combination the archetype PMEM reflects best the observed signals, we can use this to calculate the distance to criticality for each of them. In contrast to Ezaki et al. [16], who determined this distance in terms of σ and μ by fixing $\mu = \hat{\mu}$ and looking at χ_{SG} resp. $\sigma = \hat{\sigma}$ and χ_{Uni}, we introduce use a more general method.

By definition, the values of χ_{Uni} and χ_{SG} diverges at a phase transition. For the phase transition between the paramagnetic and spin-glass phase χ_{Uni} will increase, whereas between the ferro- and paramagnetic phase both χ_{Uni} and χ_{SG} increases (as explained in Section 3.1). These 'extreme' curves define the boundary between the different phases and, therefore, where a phase-transition occurs. Using the phase diagrams of χ_{Uni} and χ_{SG}, we can approximate these curves. We will denote the boundary between the paramagnetic and spin-glass phase by the curve $\sigma_c^{\text{SG}}(\mu)$ (here, both χ_{Uni} and χ_{SG} are maximized) and the boundary between the ferro- and paramagnetic phase by $\sigma_c^{\text{Uni}}(\mu)$ (at this curve, both χ_{Uni} is maximized). Subsequently, we simply define the distance to a phase transition as the shortest Euclidean distance to one of these curves.

Practically speaking, we approximate the curves $\chi^{\uparrow}_{\text{Uni}}(\mu, \sigma)$ and $\chi^{\uparrow}_{\text{SG}}(\mu, \sigma)$ by looking where the values of χ_{Uni} and χ_{SG} reach their maximal value and fitting a continuous curve to it. To account for variability arising from the stochastic nature of the simulations, we utilize the curve that spans 1.5 the grid size. This provides a smoother representation of the critical boundary in the phase diagrams.

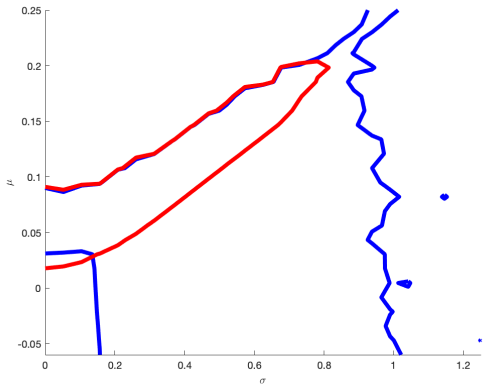
5.1.5 Refining the position using refined phase diagrams

While the Metropolis-Hastings algorithm is effective in generating samples from the Boltzmann distribution, achieving accurate samples requires a considerable amount of time for the system to equilibrate. Furthermore, to obtain uncorrelated samples, a substantial number of intermediate sweeps is necessary. Consequently, the computational cost is high, leading to long computation times.

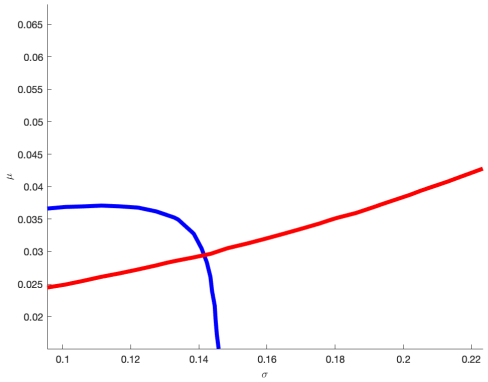
To give an idea, for a network with $N \sim 20 - 30$ nodes, it takes approximately 6 hours to compute a phase diagram of size 25×25. This computation involves performing 10 independent experiments per combination of (μ, σ) to correct for variability, consisting of $1e7$ thermalization steps and sampling B times every 1000 sweeps. As a consequence, we are constrained to a phase diagram with a relatively low resolution, potentially leading to coarse approximations of the participants' positions. To enhance these approximations, we generate a second *refined* phase diagram, wherein the range of μ and σ is determined based on the position of the participants' position in the initial *rough* phase diagrams by taking

$$\mu \in \left[\min_{i=1,\dots,n} \hat{\mu}^i - \alpha, \max_{i=1,\dots,n} \hat{\mu}^i + \alpha \right], \quad \alpha = \frac{1}{5} \left(\min_{i=1,\dots,n} \hat{\mu}^i - \max_{i=1,\dots,n} \hat{\mu}^i \right)$$
$$\sigma \in \left[\min_{i=1,\dots,n} \hat{\sigma}^i - \beta, \max_{i=1,\dots,n} \hat{\sigma}^i + \beta \right], \quad \beta = \left(\min_{i=1,\dots,n} \hat{\sigma}^i - \max_{i=1,\dots,n} \hat{\sigma}^i \right).$$

This will result in a a phase diagram that is smoother and exhibits less variation between the different combinations (μ, σ). In Figure 5.5 the improvement for locating the position of participants is illustrated. Here, we observe that the piecewise linear curve, computed using the refined phase diagrams, is smoother.



(a) "Roughly" computation position.



(b) "Refined" computation position.

Figure 5.5: The piecewise linear curves at which it holds that $\chi_{\text{SG}} \approx \tilde{\chi}_{\text{SG}}$ and $\chi_{\text{Uni}} \approx \tilde{\chi}_{\text{Uni}}$ using the rough resp. refined phase diagram.

5.1.6 Statistical analysis

Functional networks

First, we identify outliers based on exceeding $1.5 \cdot \text{IQR}$ (InterQuartile Range) limits, and we remove these extreme outliers from the analysis. Subsequently, we conduct a mixed analysis of variance (ANOVA), using IBM SPSS Statistics 29.0, to examine whether the distance to criticality is the result of the interaction between the functional network and treatment response with age and education (measured by ISCED) as covariate, as similar studies [16, 58, 75, 76] did. We verify the normality assumption using one-sample Kolmogorov-Smirnov tests, and the sphericity assumption with Mauchly's test of sphericity.

When a covariate turns out to not be significant, we repeat the analysis without that particular covariate. If there is a significant interaction, we perform for each network a post-hoc analysis of variance (ANOVA) to examine the effect further.

Whole-brain network

After identifying outliers based on exceeding $1.5 \cdot \text{IQR}$ (InterQuartile Range) limits and removing them from analysis, we conduct an analysis of variance (ANOVA) using IBM SPSS Statistics 29.0 to examine the effect of treatment response on distance to a phase transition with age and education (measured by ISCED) as covariate. We verify the normality assumption using one-sample Kolmogorov-Smirnov tests and will only consider the covariates that have a significant main effect.

Functional network nr.	T_f
1	1.00092
2	1.03234
3	1.00598
4	1.00598
5	1.01159
6	1.00902
7	1.01738

Table 5.1: Correction factors (fictitious temperatures).

5.2 Results

5.2.1 Analysis of the functional networks

Fitting the PMEM

The PMEM is fitted to the concatenated timeseries of all participants for each functional network with learning rate $\gamma = 0.05$. Recall that the accuracy is summarized in Figure 4.2.

After performing the bias correction, we observe an improved accuracy. These results are given in Figure 5.6.The fictitious temperatures, i.e. corrective factors, are given in the Table 5.1.

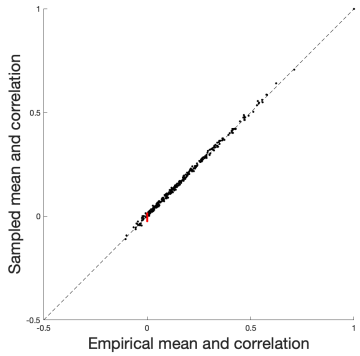
It's worth noting that for the functional networks, these factors are relatively small and their impact on the model is minimal. This suggests that there was only a slight bias in the model.

Drawing phase diagrams

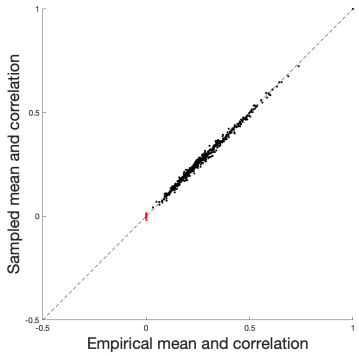
After the archetype PMEM is obtained, we draw phase diagrams for each of the functional networks. The domain over which μ and σ are varied were determined experimentally.

Figure 5.7 shows the obtained (rough) phase diagrams. It can be observed that the diagrams have qualitatively the same structure as the diagrams corresponding to the SK-model, as shown in Figure 5.1. Here, the model was simulated for $N = 238$ nodes. However, based on the definition of the SK-model discussed in Section 3.1, it follows that the phase diagrams scale with the system size N. Therefore, the phase diagrams for smaller systems exhibit the same characteristics. This suggests that the inferred archetype PMEM models of the functional networks are analogous to the SK-model, and motivates to analyse the system using the observables of a spin-glass system.

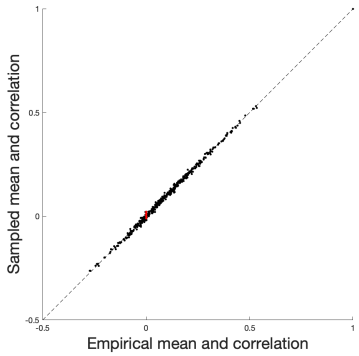
In the phase diagrams, the solid white line represents the boundary between the different phases based on the curve at which $\chi_{\mathbf{SG}}$ and χ_{Uni} "diverges" (i.e. are maximized). The dashed line is the curve where the potential variability of the simulations is considered. We will use the latter for computing the distance to criticality. Additionally, the positions of the participants are indicated by dots, with red dots representing responders and white dots nonresponders.



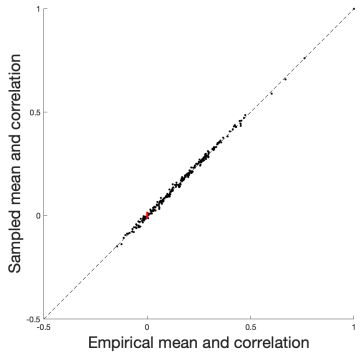
Functional network 1



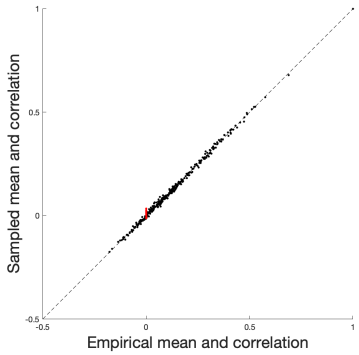
Functional network 2



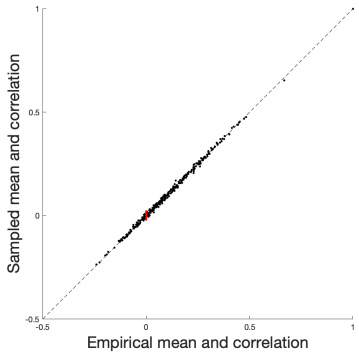
Functional network 3.



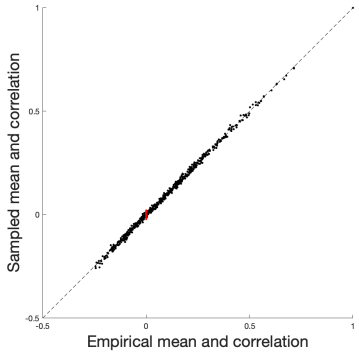
Functional network 4



Functional network 5

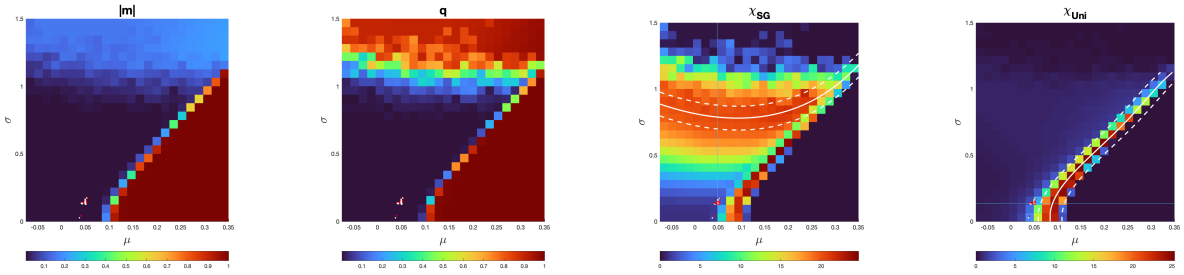


Functional network 6

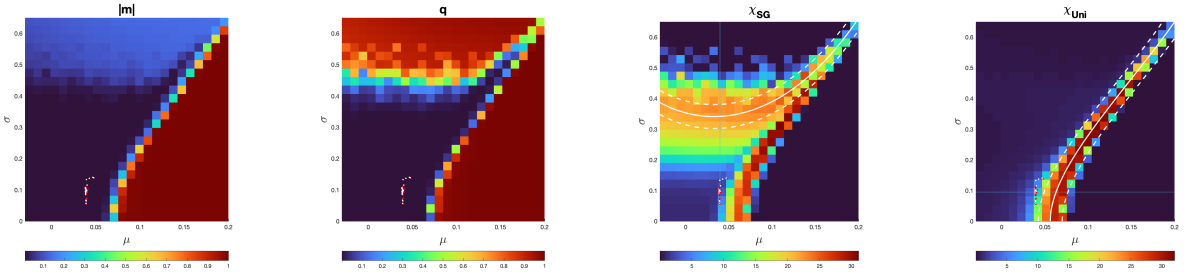


Functional network 7

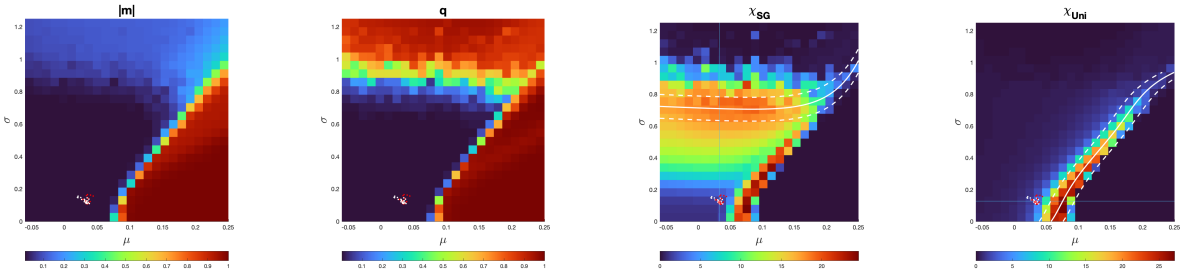
Figure 5.6: Accuracy of PLM combined with the C^2-bias correction, considered by comparing the empirical and sampled mean $\langle S_i \rangle$ and correlation $\langle S_i S_j \rangle$ for each ROI i ($i = 1, \ldots, N$).



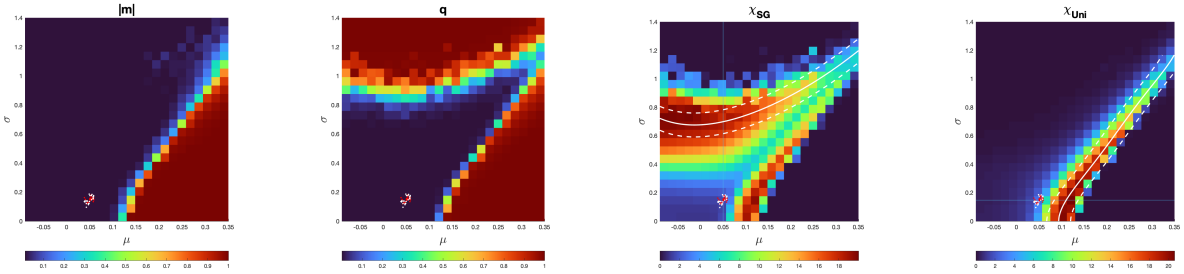
Functional network 1.



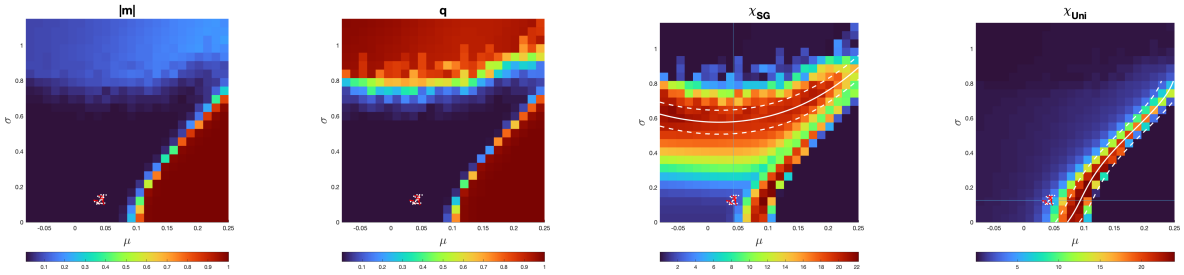
Functional network 2.



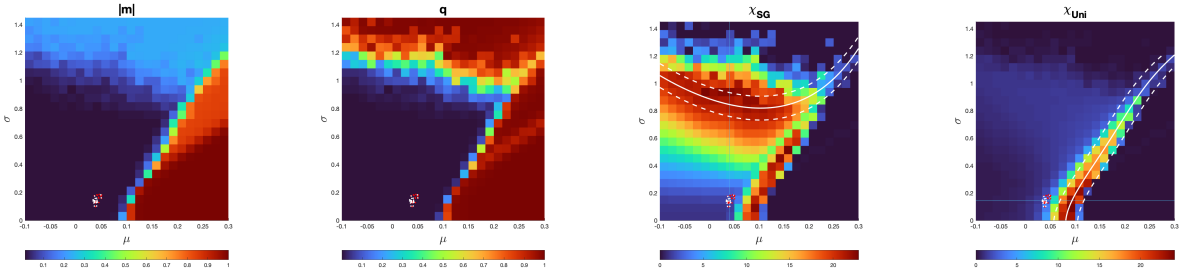
Functional network 3.



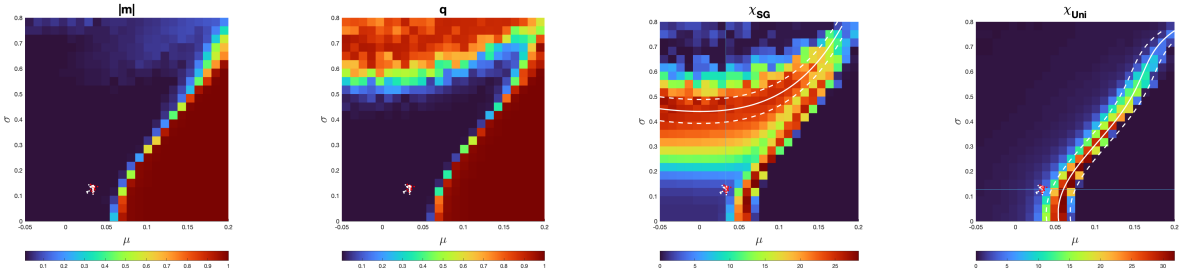
Functional network 4.



Functional network 5.

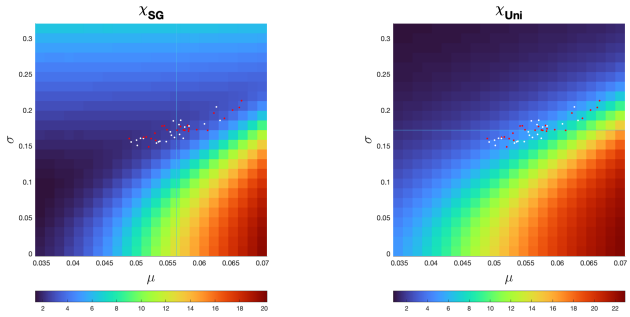


Functional network 6.

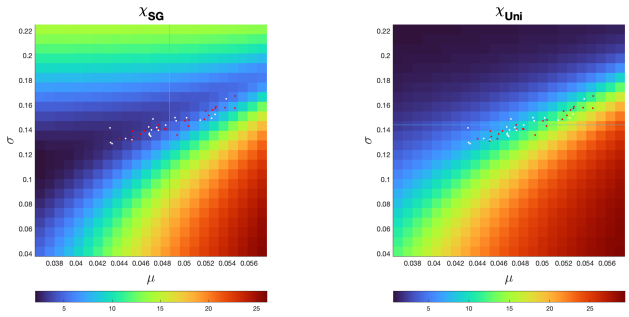


Functional network 7.

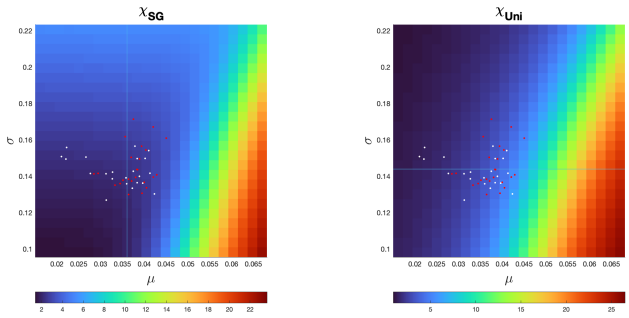
Figure 5.7: Phase diagrams of the functional networks for the observables m, q, χ_{Uni} and χ_{SG}. To take into account the bi-stability of the system (that arises in the finite case), we selected the one of the stable branches by flipping the whole system so that the magnetization is positive (i.e., we consider $|m|$). This bi-stability does only affect the magnetization. As we do not use the value of the magnetization for the whole analysis, this is only for representing the obtained observable correctly. [16, 19]



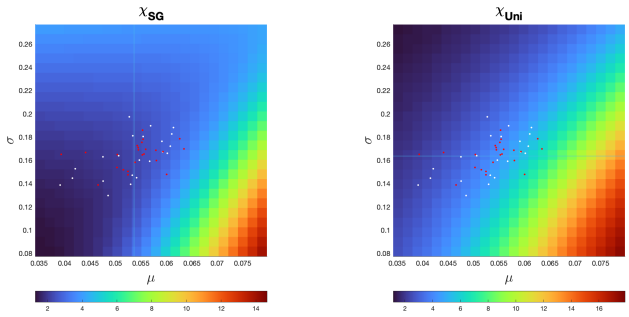
Functional network 1.



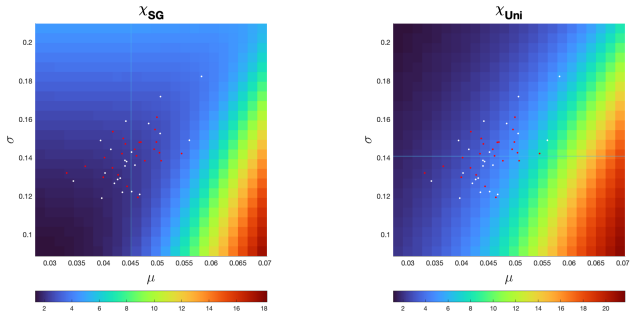
Functional network 2.



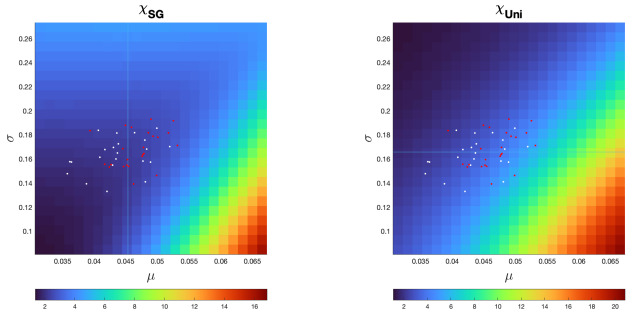
Functional network 3.



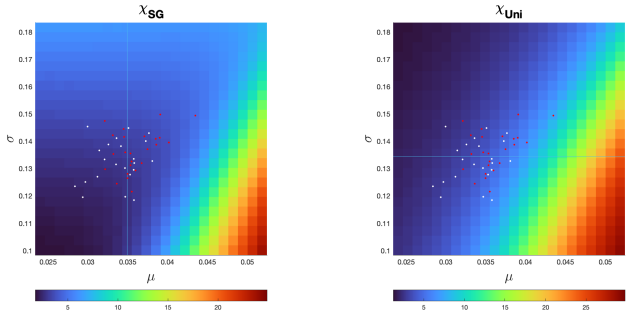
Functional network 4.



Functional network 5.



Functional network 6.



Functional network 7.

Figure 5.8: Refined phase diagrams of the functional networks for the observables χ_{Uni} and χ_{SG}. Due to the smaller range of (μ, σ), the diagram is restricted to mainly the paramagnetic phase, implying that the phase diagrams for the magnetization m and spin-glass order parameter q are not interesting anymore (as these remain zero in the paramagnetic phase).

Functional network nr.	**Age**		**Treatment response**		
	Test-value (df)	*P-value*	*Test-value (df)*	*P-value*	*FDR corrected*
1	$F(1,32) = 4.191$	$P = 0.049^{*}$	$F(1,32) = 3.107$	$P = 0.88$	$P = 0.880$
2	$F(1,32) = 13.415$	$P < 0.001^{*}$	$F(1,32) = 2.142$	$P = 0.153$	$P = 0.373$
3	$F(1,32) = 28.963$	$P < 0.001^{*}$	$F(1,32) = 1.599$	$P = 0.215$	$P = 0.376$
4	$F(1,32) = 1.002$	$P = 0.324$	$F(1,33) = 2.062$	$P = 0.160^{a}$	$P = 0.373$
5	$F(1,32) = 0.237$	$P = 0.629$	$F(1,33) = 0.196$	$P = 0.661^{a}$	$P = 0.771$
6	$F(1,32) = 8.584$	$P = 0.006^{*}$	$F(1,32) = 0.418$	$P = 0.552$	$P = 0.771$
7	$F(1,32) = 2.108$	$P = 0.156$	$F(1,33) = 6.051$	$P = 0.019^{a}$	$P = 0.133$

* $P < 0.05$.

^{a} Age was not significant and therefore removed from analysis.

Table 5.2: An ANOVA was performed for each of the functional networks, with age as covariate. When age had revealed no main effect, the covariate was removed from analysis. The False Discovery Rate (FDR) was used to correct the obtained P-values for multiple comparisons [80].

Using the positions of the participants, we drew the refined phase diagrams to enhance the accuracy of their positions. The refined phase diagrams are shown in Figure 5.8.

From the phase diagrams, we observe that the brain of the participants are in paramagnetic phase, more in the vicinity of the uniform phase transition than of the spin glass phase transition. This is confirmed by looking at the cross-section of the phase diagram for χ_{SG} and χ_{Uni} through $\mu = \tilde{\mu}$ resp. $\sigma = \tilde{\sigma}$ (where, $\hat{\mu} = \sum_{i=1}^{n} \hat{\mu}^i$ and $\hat{\sigma} = \sum_{i=1}^{n} \hat{\sigma}^i$, i.e. averaged over the position of the participants in terms of $\hat{\mu}$ and $\hat{\sigma}$). These figures can be found in the supplementary material (see Appendix B.2). Therefore, for the statistical analysis, we only consider the shortest distance to the boundary between the para- and ferromagnetic phase.

Statistical analysis

First, 11 outliers (5 from the nonresponders, 6 from the responders) were identified and removed from analysis. The normality assumption was verified using one-sample Kolmogorov-Smirnov tests, which were not significant for any of the functional networks. Education (ISCED) was not a significant covariate and therefore removed from analysis. However, there was a significant main effect of the covariate, age, on the distance to criticality ($F(1,32) = 17.089$, $p < 0.001$), implying that the effect of age on the temperature was similar for the seven functional networks. The sphericity assumption was tested using Mauchly's test, which was significant ($\chi^2(20) = 33.859$, $P = 0.028$). Therefore, the Greenhouse-Geisser correction was applied ($\varepsilon = 0.753$).

There was no significant interaction effect between functional network and treatment response on distance to criticality ($F(4.519, 144.615) = 0.322$, $P = 0.883$). However, there was a significant interaction effect between functional network and the covariate, age, on the distance to criticality ($F(4.519, 144.615) = 4.535$, $p < 0.001$). This implies that the age did significantly differ across the networks across the groups. Therefore, we consider each functional network independently using an analysis of variance (ANOVA). The results are summarized in Table 5.2.

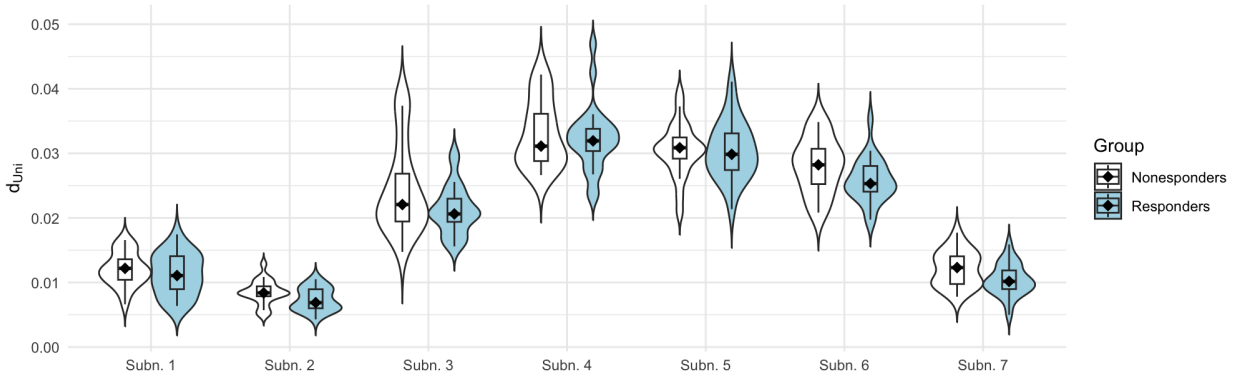


Figure 5.9: Approximated distance to the ferromagnetic phase transition for each of the functional networks, based on the refined phase diagrams.

Applying an correction for multiple comparisons (using the False Discovery Rate) to the obtained P-values for the main effect of treatment response on the distance to criticality, using R version 4.3.0, revealed that there are no significant main effects. This means that there is no functional network where the distance to criticality is associated with treatment response.

A statistical comparison of the demographics and clinical data at the start of the study where the outliers were removed revealed no significant difference between the two groups ($0.073 \leq P \leq 1.000$). By the end of the study, the non-responder group did have significantly higher clinical scores (measured by the Clinician Administered PTSD Scales (CAPS)) ($P < 0.001$), defining the difference in treatment response between the two groups. Furthermore, the non-responder group exhibited a significantly higher frequency of comorbid mood disorders ($\chi^2(1) = 4.300, P = 0.038$). As depressive, substance use and other anxiety disorders are comorbid with PTSD [77], this can be expected.

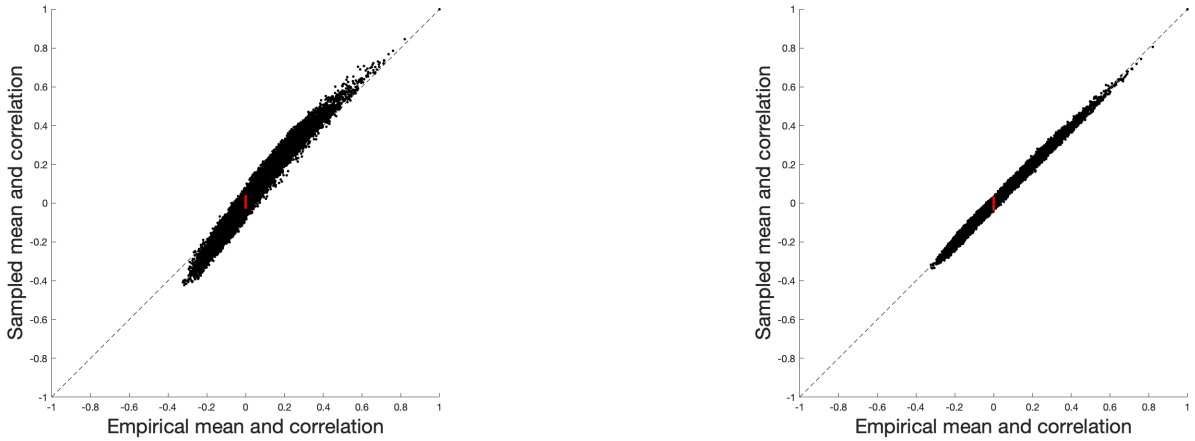
See Figure 5.9 for a graphical representation of the results.

We conclude that, based on the statistical tests, the distance to criticality is not significantly different between the two groups of participants, considering the seven functional networks as defined by Yeo et al. [59]. Additionally, there are no significant differences between the two groups in demographics and clinical data at the start of treatment, implying that the groups may be considered equivalent. Consequently, treatment response cannot be explained by the distance to criticality calculated using an archetype PMEM and phase diagrams

5.2.2 Analysis of the whole-brain network

Fitting the PMEM

As for the functional networks, we fitted the PMEM to the concatenated timeseries of all participants for each functional network with learning rate $\gamma = 0.05$. Subsequently, we corrected the obtained model using the C^2-correction and obtained the fictitious temperature $T_f = 1.06729$. This seemingly small corrective factor demonstrates a positive impact on the model's accuracy. The accuracy for both the initial and corrected model is given in Figure 5.10.



Accuracy PLM.

Accuracy PLM after C^2-correction.

Figure 5.10: Accuracy of PLM for the whole-brain network, considered by comparing the empirical and sampled mean $\langle S_i \rangle$ and correlation $\langle S_i S_j \rangle$ for each ROI i $(i = 1, \ldots, N)$.

Drawing phase diagrams

We draw phase diagrams using the obtained PMEM. The domain for μ and σ were determined experimentally. Because of the computation time of these phase diagrams, the resolution of the phase diagram was reduced to 20×20. The resulting rough phase diagrams are shown in Figure 5.11 and the refined phase diagrams (where the range of μ and σ are adapted to the participants' position in the rough phase diagrams) in Figure 5.12.

Also in this case, the phase diagram of the archetype PMEM for the whole-brain network exhibits the same characteristics as the diagrams of the SK-model. This suggests that the inferred archetype PMEM is analogous to the SK-model and can be analyzed as a spin-glass model.

Note that the participants are still in the paramagnetic phase, but it is less clear whether they are closer to the spin-glass or ferromagnetic phase transition. The cross-section of the phase diagram for χ_{SG} and χ_{Uni} through $\mu = \tilde{\mu}$ resp. $\sigma = \tilde{\sigma}$ (the corresponding figures can be found in the supplementary material in Appendix B.2)) does not reveal much more information. Therefore, for the statistical analysis, consider both the the shortest distance to the boundary between the para- and ferromagnetic phase as between the paramagnetic and spin-glass phase.

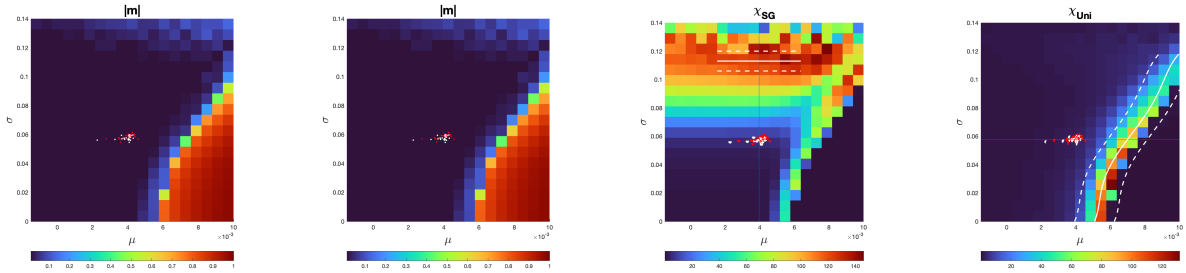


Figure 5.11: Phase diagrams corresponding to the whole-brain network for the observables m, q, χ_{Uni} and χ_{SG}.

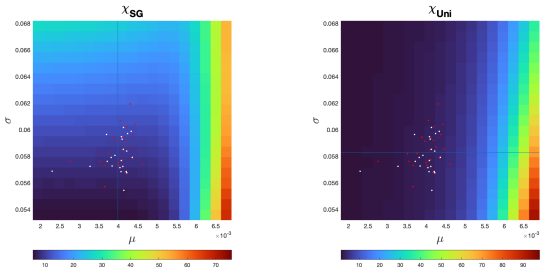


Figure 5.12: Refined phase diagrams corresponding to the whole-brain network for the observables χ_{Uni} and χ_{SG}. As the participants are in the paramagnetic phase, we observe agian that the phase diagrams for m and $q$are not relevant.

Statistical analysis

Distance to ferromagnetic phase transition

After 3 outliers (2 from the nonresponders, 1 from the responders) were identified and removed from analysis, we verified the normality assumption for each group using a one-sample Kolmogorov-Smirnov test. These were not significant. Both age and education were not significant covariates and therefore removed from analysis.

There was not a significant main effect of treatment response on the distance to criticality ($F(1, 41) = 0.133$, $P = 0.717$). This implies that distance to the spin-glass phase transition is not different across groups for the whole-brain network.

A statistical comparison of the demographics and clinical data at the start of the study where the outliers were removed revealed a significant difference in education level (ISCED) between the two groups ($U = 157$, $P = 0.038$). However, no significant difference between the other test-were observed ($0.098 \leq P \leq 1.000$). By the end of the study, the non-responder group did have significantly higher clinical scores (measured by the Clinician Administered PTSD Scales (CAPS)) ($P < 0.001$), defining the difference in treatment response between the two groups.

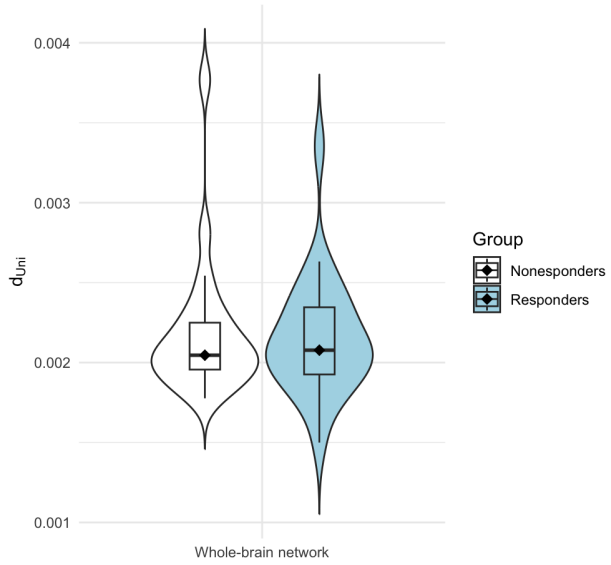


Figure 5.13: Approximated distance to the spin-glass phase transition for the whole-brain network, based on the refined phase diagram.

Furthermore, the non-responder group exhibited a significantly higher frequency of comorbid mood disorders ($\chi^2(1) = 5.815, P = 0.016$), anxiety disorders ($\chi^2(1) = 6.007, P = 0.014$) and usage of SSRIs ($\chi^2(1) = 4.773, P = 0.029$). As depressive, mood, substance use and other anxiety disorders are comorbid with PTSD [77, 78], this can be expected.

See Figure 5.13 for a graphical representation of the results.

We conclude that, based on the statistical tests, the distance to the ferromagnetic phase transition is not significantly different between the two groups of participants. Moreover, an analysis of the demographics and clinical data of the two groups revealed a significant difference in education level between the groups. Therefore, the groups cannot be completely considered as equivalent. Consequently, treatment response cannot be explained by the distance to the ferromagnetic phase transition calculated using an archetype PMEM for the whole-brain network and the corresponding phase diagrams

Distance to the spin-glass phase transition

No outliers were identified and the normality assumption was met for each group (tested using a one-sample Kolmogorov-Smirnov test, which were both not significant). Education (ISCED) was not a significant covariate and therefore removed from analysis.

There was a significant main effect of the covariate, age, on the distance to criticality ($F(1, 43) = 6.369$, $P = 0.015$), meaning that distance to the spin-glass phase transition is associated with the covariate, age, for the whole-brain network. There was not a significant main effect of treatment response on the distance to criticality ($F(1, 43) = 0.707$, $P = 0.405$). This implies that treatment response does not depend on the distance to the spin-glass phase transition for the whole-brain network.

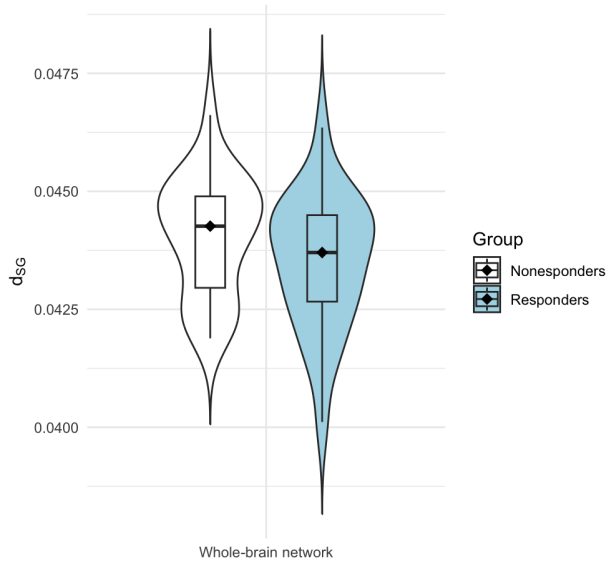


Figure 5.14: Approximated distance to the ferromagnetic phase transition for the whole-brain network, based on the refined phase diagram.

A statistical comparison of the demographics and clinical data at the start of the study revealed a significant difference between the two groups in education level (measured by ISCED) $U = 186, P = 0.042$. There were no significant differences between the two groups on the other test-values observed ($0.079 \leq P \leq 1.000$).

By the end of the study, the non-responder group did have significantly higher clinical scores (measured by the Clinician Administered PTSD Scales (CAPS)) ($P < 0.001$), defining the difference in treatment response between the two groups. Furthermore, the non-responder group exhibited a significantly higher frequency of comorbid mood disorders ($\chi^2(1) = 5.007, P = 0.025$), anxiety disorders ($\chi^2(1) = 6.724, P = 0.010$) and usage of SSRIs ($\chi^2(1) = 3.919, P = 0.048$). As depressive, mood, substance use and other anxiety disorders are comorbid with PTSD [77, 78], this can be expected.

See Figure 5.14 for a graphical representation of the results.

We conclude that, based on the statistical tests, the distance to the spin-glass phase transition is not significantly different between the two groups of participants. Additionally, an analysis of the demographics and clinical data of the two groups revealed a significant difference in education level between the groups. This means that the groups cannot be completely considered as equivalent. Consequently, treatment response cannot be explained by the distance to the spin-glass phase transition calculated using an archetype PMEM for the whole-brain network and the corresponding phase diagrams.

Part II

Entropy

There are numerous ways to study the complexity of (fMRI) brain signals. The complexity of is measured by the unpredictability of these signals: irregular signals are more complex than regular signals. One of the most commonly used measures, is the *Shannon-entropy* due to its simple algorithm, the small amount of data required and robustness to noise [24, 25]. The entropy of a signal measures the *randomness* or *predictability* of a stochastic process and increases with the degree of irregularity [21, 22]. In the last decades, several studies provided both empirical and theoretical evidence for the crucial role of brain signal variability for the functioning of the brain [23].

Additionally, there is evidence that entropy could be a robust feature for classification of mental states of the brain. Studies on cognitive performance [81, 82], Schizophrenia [83], attention-deficit/hyperactivity disorder [84], and depression [85] showed that entropy can be used to analyze the temporal changes in resting state fMRI signals and locate the relevant brain regions. Furthermore, it provided evidence that the change in entropy could be related to disease states. These different studies showed that entropy can be used to analyze the temporal changes in resting state fMRI signals and locate the relevant brain regions. Furthermore, the change in entropy could be related to disease states. This motivates to explore the entropy of our dataset. As for the criticality, this leads to the question whether the concept of entropy could be used for explaining differences in treatment response using the resting state fMRI data.

Mathematically, the *Shannon-entropy* is defined as the rate of new "information" generation or "uncertainty" and measures the probability of generating a new pattern in the signal: the greater the probability of generating a new pattern that there is, the greater the signal complexity [24]. The main formula for the (Shannon-)entropy S is defined as

$$S = -\sum_i p_i \log p_i,$$

where p_i the probability of observing the ith state a system can attain ($i = 1, \ldots, n$).

Consequently, in case of the neuroscience, the aim of an entropy algorithm is to approximate this probability distribution from the given neural data. There are several ways to do this. The most common used methods include approximate entropy (AE), sample entropy (SE), permutation entropy (PE) and fuzzy entropy (FE). Niu et al. [24] discovered the test-retest reliability of these four different methods and proposed a new entropy algorithm: permutation fuzzy entropy (PFE). The results showed that the highest reliability was achieved with PFE. Therefore, for this thesis we are going to study the PFE to explore whether entropy could play a role in treatment response. For each measured signal per ROI [26], we will calculate the PE and PFE and compare the obtained values at whole-brain level, at subnetwork level and per ROI.

Part II is structured as follows. In Chapter 6 we will discuss Permutation Entropy (PE) and Fuzzy Entropy (FE), which form together the foundation of Permutation Fuzzy Entropy (PFE). In Chapter 7, some implementation choices are introduced after which we analyze the PFE of the neural signals at whole-brain and subnetwork level. The analysis of the PFE per ROI is discussed in the Appendix C.

Chapter 6

Preliminaries

6.1 Permutation Entropy

The first entropy algorithm we consider is *permutation entropy* (PE). PE, proposed by Bandt and Pompe [86], was introduced for measuring the *irregularity* of dynamic time series [86] and uses the shape of neighboring time points to evaluate complexity based on permutation patterns [24]. The method has several advantages over other entropy measures: it is simpler, has lower computation complexity without the need of making further modeling assumptions, and is robust in the presence of observational and dynamical noise [25]. Furthermore, as described before, it has a good test-retest reliability [24].

The process of calculating PE is as follows. Assume, given is the time series X of length B, i.e.

$$X = \{x(1), x(2), \ldots, x(B)\}.$$

We are going to construct a phase space consisting of vectors composed of the $m-$th subsequent values by reconstructing a phase space as follows:

$$X(i) = (x(i), x(i+\tau), \ldots, x(i+(m-1)\tau)),$$

where $i = 1, 2, \ldots, N-(m-1)\tau$, m the embedding parameter and τ the delay time.

Thereafter, we rearrange each component in ascending numerical order as follows:

$$x(i+(j_1-1)\tau) \leq x(i+(j_2-1)\tau) \leq \cdots \leq x(i+(j_m-1)) \tag{6.1}$$

where $j_1, j_2, \ldots, j_m$ the index of the column in the reconstructed component. Notice that in the case that two values are equal, i.e. $x(i+(j_n-1)\tau) = x(i+(j_{n+1}-1)\tau)$, then they are ordered according to the size of the value of j_n and j_{n+1}, such that when $j_n < j_{n+1}$ we have $x(i+(j_n-1)\tau) < x(i+(j_{n+1}-1)\tau)$. We obtained a set of permutation vectors, that maps each vector $X(i)$ to a sequence $(1, 2, \ldots, m)$, or $(2, 1, \ldots, m), \ldots,$ or $(m, m-1, \ldots, 1)$ (in total $m!$ possible sequences).

Now, denote the probability of observing one of these sequences by $P_g(g = (1, \ldots, k))$. Then, we define the PE as the Shannon entropy for the k different sequences

$$\mathrm{PE}(N, \mathrm{m}, \tau) = -\sum_{j \leq k} P_j \ln P_j.$$

To conclude this definition, notice that PE reaches its maximum $\ln m!$ when $P_g = 1/m!$. The standardized PE (by $\ln m!$) is given by [21]

$$\mathrm{PE}_s = \mathrm{PE}/\ln m!.$$

In order to calculate the PE, we have to consider three parameter values. One value, the length of the time series B is set. The other two parameters, the embedding dimension m and time delay τ should be chosen.

6.2 Fuzzy Entropy

The second entropy algorithm we will discuss, is *fuzzy entropy* (FE). FE is proposed by Chen [87] and based on the concept of "fuzzy sets" introduced by Zadeh [88]. The main advantage of FE, it that it is applicable to relatively short signals and is very robust to noise.

A *fuzzy set*, is a set whose elements have varying degrees of membership. This is different from the "classical" definition of a set, where elements have full membership or not. The degree of membership of an elements of a fuzzy set is described by the membership function $\mu_C(x)$ that maps elements x of the set C into a real value on the interval $[0, 1]$. This gives us a mechanism for measuring the degree to which an element belongs to a set, or equivalently, a pattern belongs to a certain class. The latter is what we are using for calculating the entropy of a signal: we are actually counting how many signals belongs to which class by looking for a similarity in two vectors.

For the PFE algorithm, we use the exponential membership function $\mu(d_{ij}) = \exp(-(d_{ij}^m)^n/r)$ (the fuzzy function) to get a fuzzy measurement of two vectors' similarity based on their shapes. The closer the measurements X_i and X_j are, the greater the similarity. Note that the exponential function attains its maximum *i.e.*$\mu(d_{ij}) = 1$ when the two vectors are similar. Now, the PE is defined as the conditional probability that patterns observed in signals of length m are the same for $m + 1$ points and depends on the average of the similarity $D_{ij}^m = \mu(d_{ij})$ of the different sequences of length m and $m + 1$.

The algorithm for calculating FE, is as follows. Assume, given is the time series X of length B :

$$X = \{x(1), x(2), \ldots, x(B)\}.$$

We reconstruct the phase space by as follows:

$$X_i^m = \{x(i), x(i + 1), \ldots, x(i + m - 1)\} - x_0(i), \quad 1 \leq i \leq N - m + 1.$$

Here, we remove a baseline, given by

$$x_0(i) = \frac{1}{m} \sum_{j \leq m-1} x(i + j).$$

We define the distance d_{ij}^m between the vectors X_i^m and X_j^m as the largest absolute difference between its elements (maximum norm), i.e.

$$d_{ij}^m = \|X_i^m - X_j^m\|_\infty = \max_{k=0,1,\ldots,m-1} |x(i+k) - X(j+k)| \quad i,j = 1,2,\ldots,N-m+1, j \neq i.$$

Using the fuzzy membership function $\mu(d_{ij}^m, n, r)$, we calculate the similarity degree D_{ij}^m of the vector X_j^m to X_i^m as

$$D_{ij}^m = \mu(d_{ij}^m, n, r) = \exp\left(\frac{-(d_{ij})^n}{r}\right).$$

Here, n and r are the width and gradient of the exponential function.

Defining the function $\phi^m(r)$ as

$$\phi^m(r) = \frac{1}{N-m+1} \sum_{i=1}^{N-m+1} \left(\frac{1}{N-m} \sum_{j \neq i \leq N-m+1} D_{ij}^m \right),$$

gives us the following expression for the estimated value of the corresponding PE:

$$\mathrm{FE}(N,m,r) = \ln \phi^m(r) - \ln \phi^{m+1}(r).$$

Here, m is the phase space dimension, r the similar tolerance. Both has to be chosen beforehand.

6.3 Permutation fuzzy entropy

Permutation fuzzy entropy (PFE), proposed by Niu et al. [24] combines the advantages of PE and PE and performed in their study the best test-retest reliability. The algorithm is a composition of the procedure for calculating the PE and FE as introduced above.

We first sort and symbolize the original time series X of length B and then calculate the PE of the newly obtained sequence

$$Y(i), 1 \leq i \leq N - (pm-1)\tau,$$

here, pm is the permutation embedding parameter and τ the time delay. The PFE follows by calculating the PE of this obtained sequence Y and is given by

$$\mathrm{PFE}(N, pm, \tau, m, r) = \ln \phi^m(r) - \ln \phi^{m+1}(r),$$

where, again, m is the phase space dimension and r the similar tolerance.

Chapter 7

Entropy analysis

As the Permutation Fuzzy Entropy (PFE) performed the best test-retest reliability, we will analyze our neural data using this entropy algorithm. In this exploration, we will study the entropy across the two different levels as utilized in the criticality analysis, including the whole-brain network, consisting of $N = 238$ nodes (ROIs), and the functional networks, including the Visual (1), Somatomotor (2), Dorsal Attention (3), Ventral Attention (4), Limbic (5), Frontoparietal (6) and Default network (7).

This chapter is organized in two sections. In the first section (Section 7.1), we outline the algorithm for computing the PFE per ROI, functional network and the whole-brain network. In the second section (Section 7.2), we analyze the obtained results for the functional networks and the whole-brain network. Additionally, the statistical comparison of the entropy per ROI is explored. These results can be found in the supplementary material (see Appendix C).

7.1 Methods

7.1.1 Computing the PFE

For the PFE algorithm, we have to set the parameters pm, τ and m. The optimal values for these parameters does not depend on the data(structure), so we set these as defended in other studies [22, 24, 89] as follows: the permutation embedding $pm = 4$, the time delay $\tau = 1$, the phase space dimension $m = 2$ and the similar tolerance $r = 0.25 \cdot \sigma_i$, where σ_i the standard deviation of the signal i ($i = 1, \ldots, N$).

Note that this algorithm computes the PFE per individual ROI. We calculate the entropy for each functional network by averaging the PFE across the relevant ROIs. The entropy for the whole-brain network is obtained by averaging over all ROIs.

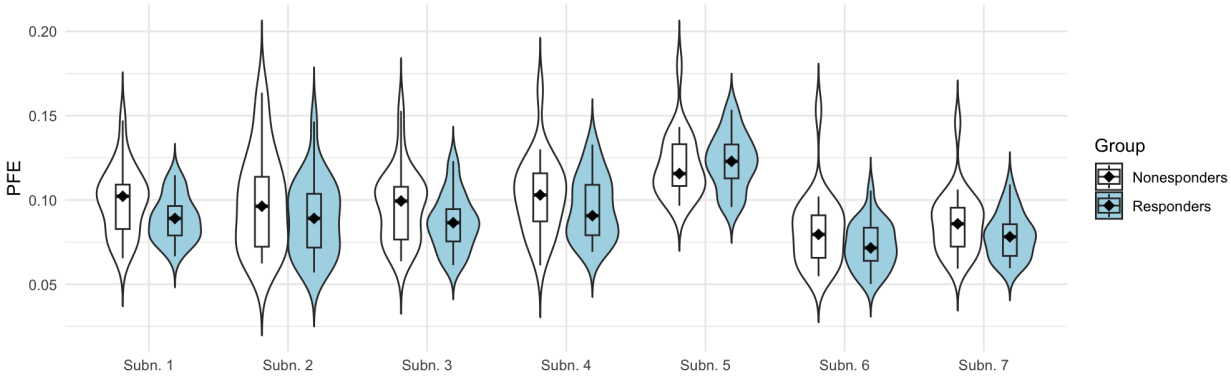


Figure 7.1: PFE per functional network for each participant.

7.1.2 Statistical analysis

Functional networks

First, we identify outliers based on exceeding $1.5 \cdot \text{IQR}$ (InterQuartile Range) limits, and we remove these extreme outliers from the analysis. Subsequently, we conduct a mixed analysis of variance (ANOVA), using IBM SPSS Statistics 29.0, to examine whether the PFE is the result of the interaction between the functional network and treatment response with age and education (measured by ISCED) as covariate, as similar studies [90–92] did. We verify the normality assumption using one-sample Kolmogorov-Smirnov tests, and the sphericity assumption with Mauchly's test of sphericity.

When a covariate turns out to not be significant, we repeat the analysis without that particular covariate. If there is a significant interaction, we perform for each network a post-hoc analysis of variance (ANOVA) to examine the effect further.

Whole-brain network

After identifying outliers based on exceeding $1.5 \cdot \text{IQR}$ (InterQuartile Range) limits and removing them from analysis, we conduct an analysis of variance (ANOVA) using IBM SPSS Statistics 29.0 to examine the effect of treatment response on PFE with age and education (measured by ISCED) as covariates. We verify the normality assumption using one-sample Kolmogorov-Smirnov tests and will only consider covariates that are significant.

7.2 Results

The obtained values of PFE per functional network are summarized in Figure 7.1. For the whole-brain network, the obtained values (per participant) are represented in Figure 7.2.

Functional network nr.	**Age**		**Treatment response**		
	Test-value (df)	*P-value*	*Test-value (df)*	*P-value*	*FDR corrected*
1	$F(1, 41) = 22.104$	$P < 0.001^*$	$F(1, 41) = 0.842$	$P = 0.364$	$P = 0.868$
2	$F(1, 41) = 22.613$	$P < 0.001^*$	$F(1, 41) = 0.051$	$P = 0.823$	$P = 0.868$
3	$F(1, 41) = 25.566$	$P < 0.001^*$	$F(1, 41) = 0.103$	$P = 0.749$	$P = 0.868$
4	$F(1, 41) = 29.395$	$P < 0.001^*$	$F(1, 41) = 0.127$	$P = 0.723$	$P = 0.868$
5	$F(1, 41) = 2.755$	$P = 0.105$	$F(1, 42) = 1.396$	$P = 0.244$[a]	$P = 0.868$
6	$F(1, 41) = 28.412$	$P < 0.001^*$	$F(1, 41) = 0.035$	$P = 0.852$	$P = 0.868$
7	$F(1, 41) = 21.497$	$P < 0.001^*$	$F(1, 41) = 0.028$	$P = 0.868$	$P = 0.868$

* $P < 0.05$.

[a] Age was not significant and therefore removed from analysis.

Table 7.1: An ANOVA was performed for each of the functional networks, with age as covariate. When age had revealed no main effect, the covariate was removed from analysis. The False Discovery Rate (FDR) was used to correct the obtained P-values for multiple comparisons [80].

7.2.1 Statistical analysis

Functional networks

First, 2 outliers (2 from the non-responder group) were identified and removed from analysis. The normality assumption was verified for both groups using one-sample Kolmogorov-Smirnov tests, which were not significant for any of the subnetworks. Education (ISCED) was not a significant covariate and therefore removed from analysis. However, there was a significant main effect of age on PFE ($F(1, 41) = 26.798$, $p < 0.001$). This implies that age is associated with PFE. The sphericity assumption was tested using Mauchly's test, which was significant ($\chi^2(20) = 98.871$, $p < 0.001$). Therefore, the Greenhouse-Geisser correction was applied ($\varepsilon = 0.604$).

There was no significant interaction effect between functional network and treatment response on PFE ($F(3.627, 148.699) = 1.819$, $p = 0.135$). This means that PFE was not significantly different for the networks and groups. However, was a significant interaction effect between functional network and age on PFE ($F(3.627, 148.699) = 6.212$, $p < 0.001$). This implies that the effect of age on the PFE was different across networks. Therefore, we consider each functional network independently using an analysis of variance (ANOVA). The results are summarized in Table 7.1.

Applying an correction for multiple comparisons (using the False Discovery Rate) to the obtained P-values for the main effect of treatment response on the distance to criticality, using R version 4.3.0, revealed that there are no significant main effects. This means that there is no functional network where the PFE is associated with treatment response.

A statistical comparison of the demographics and clinical data (conducted using the same methods as summarized in Table 4.1) at the start of the study of the groups where the outliers were removed revealed a significant difference in education level (measured by ISCED) $U = 153, P = 0.015$. There were no significant differences between the two groups on the other test-values observed ($0.065 \leq P \leq 1.000$).

By the end of the study, the non-responder group did have significantly higher clinical scores (measured by the Clinician Administered PTSD Scales (CAPS)) ($P < 0.001$), defining the difference in treatment response between the two groups. Furthermore, the non-responder group did exhibit a significantly higher frequency of comorbid mood disorder ($\chi^2(1) = 6.247, P = 0.012$), anxiety disorder ($\chi^2(1) = 8.097, P = 0.004$), and usage of SSRIs ($\chi^2(1) = 5.065, P = 0.024$). As depressive, mood, substance use and other anxiety disorders are comorbid with PTSD [77, 78], this can be expected.

We conclude that, based on the statistical tests, the PFE of fMRI signals for each of the functional networks was not significantly different between the two groups of participants. Moreover, an analysis of the demographics and clinical data at the start of the study revealed a significant difference in education level between the groups. Therefore, the groups cannot be completely considered as equivalent. Consequently, treatment response cannot be explained by the PFE of the neural signals of the functional networks.

Whole-brain network

No outliers were identified and the normality assumption was met for each group (tested using a one-sample Kolmogorov-Smirnov test, which was not significant for both groups). Education (ISCED) was not a significant covariate and therefore removed from analysis. However, there was a significant main effect of age on PFE ($F(1, 43) = 28.172$, $p < 0.001$), meaning that PFE is associated with age for the whole-brain network.

There was no significant main effect of treatment response on the PFE ($F(1, 43) = 0.050$, $p = 0.825$). This implies that treatment response does not depend on the PFE for the whole-brain network.

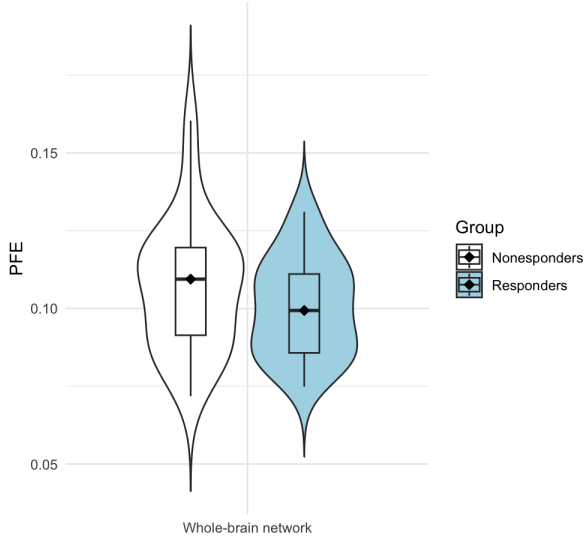


Figure 7.2: PFE of the whole brain network for each participant.

A statistical comparison of the demographics and clinical data at the start of the study revealed a significant difference between the two groups in education level (measured by ISCED) $U = 186, P = 0.042$. There were no significant differences between the two groups on the other test-values observed ($0.079 \leq P \leq 1.000$). By the end of the study, the non-responder group did have significantly higher clinical scores (measured by the Clinician Administered PTSD Scales (CAPS)) ($P < 0.001$), defining the difference in treatment response between the two groups. Furthermore, the non-responder group exhibited a significantly higher frequency of comorbid mood disorders ($\chi^2(1) = 5.007, P = 0.025$), anxiety disorders ($\chi^2(1) = 6.724, P = 0.010$) and usage of SSRIs ($\chi^2(1) = 3.919, P = 0.048$). As depressive, mood, substance use and other anxiety disorders are comorbid with PTSD [77, 78], this can be expected.

Based on these statistical tests, we conclude that the PFE is not significantly different between the two groups of participants. Moreover, an analysis of the demographics and clinical data of the two groups revealed a significant difference in education level between the groups. Therefore, the groups cannot be completely considered as equivalent. Consequently, treatment response cannot be explained by the PFE of the neural signals of the whole-brain network.

Discussion

In conclusion, in this thesis we studied the difference of treatment response of patients with PTSD to psychotherapy using the concepts of criticality and entropy.

Criticality

We studied the notion of criticality, based on the criticality hypothesis, using two methods. Both relied on an archetype PMEM, inferred from the concatenated neural signals of all participants, and aimed to investigate whether there exists an association between the distance to a phase transition and treatment response.

We hypothesized that the brains of participants who responded to psychotherapy operate closer to criticality than those of the non-responders. To explore our hypothesis, we applied the two different approaches to both the whole-brain network based on the Brainnetome atlas [26], consisting of 238 Regions Of Interest (ROIs), similar as previous studies [16,19,29,58], and to seven functional networks (the Visual, Somatomotor, Dorsal Attention, Ventral Attention, Limbic, Frontoparietal and Default network) inspired by the study of Xin et al. [58] and proposed by Ezaki et al. [16].

System temperature analysis

The first method, inspired by Ruffini et al. [19], involved inferring the system temperature for each participant using the archetype Pairwise Maximum Entropy Model (PMEM). We showed that this 'personal' system temperature that was used to personalize the archetype PMEM also served as a measure for the distance to criticality.

Analysis of simulations revealed that participants were consistently in the paramagnetic phase, with a system temperature above the critical temperature. While no significant difference in temperature was observed for the whole-brain network, a significant main effect in temperature was found for the functional networks. Specifically, responders to psychotherapy exhibited a significantly lower system temperature compared to non-responders, indicating that they were closer to criticality. This is in line with our hypothesis and supports the notion that the brain is functioning more optimal in the vicinity of a phase transition, as proposed by the criticality hypothesis. Additionally, there were no significant differences observed in the demographics and clinical data at the start of treatment. Therefore, we can conclude that the distance to criticality could be explained by the distance to criticality, measured by the 'personal' system temperature.

Moreover, this result suggests that the inferred whole-brain network does not operate equivalently to the seven

functional networks functioning collectively. This emphasizes the importance of considering the functional networks individually.

A practical limitation that should be taken into account, for both analyses, is that the posterior part of the occipital cortex (i.e., a part of the back of the brain) was not included for all participants. As a result, eight occipital ROIs (out of a total of 246, 3.3%), all part of one of the seven functional networks, were excluded from the analysis. This may lead to an structural difference in the inferred whole-brain network and the relevant functional network, but should not have influenced the other inferred functional networks.

Since several studies suggest that cognitive performance is associated with criticality [11–16], our findings could imply that improved cognitive abilities contributes to treatment response. To ensure that variations in cognitive abilities cannot be attributed to intelligence, we corrected for intelligence (indirectly) by using education as covariate (unfortunately, IQ was not measured on itself). The results of the statistical tests did not reveal an association between education level (measured by ISCED) and the 'personal' system temperature.

A similar study, performed by Ruffini et al. [19], employed this method to a dataset of fMRI scans to study whether the different functioning of a LSD-induced brain could be explained by the system temperature. He concluded, considering the whole-brain network, that the brain operates in a paramagnetic state and LSD shifts the brain further away from criticality by an increased system temperature. These results align partially with our findings, as we observed that the (more) healthy brain operates closer to a phase transition than the affected brain.

In contrast to our findings, they found that the temperature served as a control parameter to shift the system from the spin-glass to the paramagnetic phase. However, there is no clear evidence that this phase transition state is associated with the criticality hypothesis [16]. Moreover, the observed phase transition between the ferro- and paramagnetic phases is associated with advantageous increased computational ability [4, 32, 33] and is supported by several other computational studies on the appearance of criticality in the functioning of the brain [93–95].

One might consider that the difference in the observed phase transition is influenced by network size; however, we demonstrated using simulations that this is not the case. The ferromagnetic phase transition appears for both the whole-brain network and the functional networks. Thus, another factor must contribute to this difference. One potential factor could be that our study involves a group of patients with an impaired brain, as opposed to the healthy group of participants used by Ruffini et al. [19].

Phase diagram analysis

The second method, inspired by the work of Ezaki et al. [16], involved drawing phase diagrams using the archetype Pairwise Maximum Entropy Model (PMEM) with inverse temperature $\beta = 1.$. For this approach, we adapted the archetype PMEM for each participant by parameterizing the model. To determine which parameter combination reflected best the individual neural signals, using the found analogy between the inferred archetype PMEM and SK-model, we proposed a procedure based on the observed data and phase diagrams. As a result, each participant was assigned to a position in the phase diagram, corresponding to their optimal parameter combination.

We demonstrated that this position can be used to calculate the distance to criticality.

Consistent with the temperature analysis, we observed that the participants' brains operated in the paramagnetic phase, close to the phase transition between para- and ferromagnetic phase. The distance to criticality was defined as the shortest distance from participants in the phase diagram to the curve marking the boundary between the different phases. To address the variability resulting from the low resolution of the drawn phase diagrams, we introduced (adapted) refined phase diagrams for calculating a more accurate distance.

Statistical analysis of the obtained distances to criticality did not reveal any significant differences in the distance to a phase transition between the two groups of participants, for both the whole-brain and functional networks. This means that, following this analysis, the difference in treatment response cannot be explained by the distance to criticality.

As previously discussed, the two different approaches we explored both rely on the archetype PMEM, introduced to overcome the challenge of the sparsely sampled neural signals. The way in which we defined the distance to criticality (i.e., by the system temperature and the position in the phase diagram), followed by the procedure that was used to adapt the archetype model to the participants.

However, it is questionable whether the individual subjects behave the same as the group-level archetype model and thereby whether the used method is valid. A simple robustness analysis, provided in Appendix B.3, by grouping the participants randomly into two groups, shows us that it is possible to infer a PMEM and these give rise to equivalent phase diagrams. This suggests that, for both approaches we used, the findings are not caused by the way in which the participants were grouped.

To enhance the accuracy of the results, one might consider to improve the methods that are used. There are various paths you could take to achieve this. The most thorough solutions involve finding methods that enable the inference of the PMEM individually. For this, it is important that the neural signals are less sparsely sampled and/or a bigger dataset is provided. Otherwise, for example, a Bayesian framework maybe be useful for the estimation of the PMEM [96, 97]. The method of Bayesian inference can reliably estimate individual PMEMs, even with small samples, by incorporating the prior group information. However, for these personalized PMEMs, it is less obvious how to study the notion of criticality in the system. Where a definition for the distance to criticality arises naturally in the way we analyzed the archetype PMEM, this will more difficult in case you have to study multiple distinct models. Therefore, a more obvious path to take is to improve the methods we used to analyze criticality, i.e., by using an archetype model and further considering the different approaches for 'personalizing' the PMEM.

The main advantage of the temperature analysis is that we do not need to sample from the inferred model. Consequently, this method is more accurate than the phase diagram analysis where you are dealing with variability. However, this means also that there is not much more space for improvement.

For the phase diagram analysis, one can enhance the method's accuracy by improving the sampling technique. The Metropolis-Hastings algorithm we used, motivated by similar studies [16, 51, 56], was computationally costly. Exploring more efficient methods such as parallel and simulated tempering [73] could significantly enhance computational efficiency.

This would enable to generate more accurate results by drawing finer phase diagrams with higher resolution. Additionally, it facilitates better correction for variability by leveraging a larger number of experiments in the averaging process.

Besides improving the method, there are other interesting aspects of this analysis that can be improved. One of the most interesting and important findings is the relevance of the functional networks. Therefore, key recommendation is to continue studying functional networks individually rather than considering the entire whole-brain network. Building on this, further improvements could involve investigating the interactions between these functional networks. The so-called Block spin Ising model, as described by Contucci et al. [98], could be a useful model for doing this. This model explores interactions among 'blocks' of (Ising) spin systems.

Another fundamental aspect that is ignored in the analysis, is the spatial correlation among ROIs. To address for this, one may consider incorporating a two- or three-dimensional spin-glass model [62].

A last suggestion for further research, to conclude, is to consider the biological interpretation of the (archetype) PMEM and the way in which we defined the defined distance to criticality. The quantities used to control and analyze the statistical mechanical models, provide insight into the system's structure and whether they differ, but are biologically not meaningful. Therefore, based on our findings, we cannot draw clear conclusions about *how* the distance to criticality affects the functioning of the brains. Additionally, the theory of statistical mechanics, developed for studying thermodynamics, relies on the assumption that a system is in equilibrium. In our analysis, we assumed that brain is operates in equilibrium, implying that observed states can be sampled independently from a distribution. However, this does not take into account the observed temporal dynamics. Consequently, unless the assumptions seems to align well with the functioning of the brain, the neurological relevance of our findings is difficult to determine.

Entropy

We explored the measure of entropy using Permutation Fuzzy Entropy, motivated by the good test-retest reliability [24]. For each participant, we computed the entropy of the neural signals per ROI and compared the results at the same levels we employed for the examination of criticality, i.e., for functional networks and the whole-brain network, by averaging across the relevant ROIs. The statistical comparison revealed no significant differences in entropy, for across functional networks and for the whole-brain network. Additionally, we explored the entropy per ROI. This analysis did reveal significant differences for several ROIs, however, these did not survive correction for multiple comparisons.

To conclude, these results suggest that the PFE is not sensitive enough for picking up differences in treatment response from the acquiered fMRI data.

In summary, we studied whether treatment response in PTSD participants could be explained through criticality or differences in entropy. Promising results in criticality analysis indicated a closer distance to criticality in responders, while entropy analysis did not yield significant differences.

Bibliography

[1] V. del Barrio. *Diagnostic and Statistical Manual of Mental Disorders*. Elsevier, 2017.

[2] American Psychological Association. Clinical practice guideline for the treatment of ptsd, 2017.

[3] Remko van Lutterveld, Tim Varkevisser, Karlijn Kouwer, Sanne J. H. van Rooij, Mitzy Kennis, Martine Hueting, Simone van Montfort, Edwin van Dellen, and Elbert Geuze. Spontaneous brain activity, graph metrics, and head motion related to prospective post-traumatic stress disorder trauma-focused therapy response. *Frontiers in Human Neuroscience*, 16, August 2022.

[4] Robert Legenstein and Wolfgang Maass. Edge of chaos and prediction of computational performance for neural circuit models. *Neural Networks*, 20(3):323–334, April 2007.

[5] Janina Hesse and Thilo Gross. Self-organized criticality as a fundamental property of neural systems. *Frontiers in systems neuroscience*, 8:166, 2014.

[6] Dante R Chialvo. Emergent complex neural dynamics. *Nature physics*, 6(10):744–750, 2010.

[7] Thierry Mora and William Bialek. Are biological systems poised at criticality? *Journal of Statistical Physics*, 144:268–302, 2011.

[8] D Plenz. Neuronal avalanches and coherence potentials. *The European Physical Journal Special Topics*, 205(1):259–301, 2012.

[9] Shan Yu, Hongdian Yang, Oren Shriki, and Dietmar Plenz. Universal organization of resting brain activity at the thermodynamic critical point. *Frontiers in systems neuroscience*, 7:42, 2013.

[10] Gašper Tkačik, Thierry Mora, Olivier Marre, Dario Amodei, Stephanie E Palmer, Michael J Berry, and William Bialek. Thermodynamics and signatures of criticality in a network of neurons. *Proceedings of the National Academy of Sciences*, 112(37):11508–11513, 2015.

[11] Takamitsu Watanabe, Naoki Masuda, Fukuda Megumi, Ryota Kanai, and Geraint Rees. Energy landscape and dynamics of brain activity during human bistable perception. *Nature Communications*, 5(1), August 2014.

[12] Takamitsu Watanabe and Geraint Rees. Brain network dynamics in high-functioning individuals with autism. *Nature Communications*, 8(1), July 2017.

[13] Takahiro Ezaki, Michiko Sakaki, Takamitsu Watanabe, and Naoki Masuda. Age-related changes in the ease of dynamical transitions in human brain activity. *Human Brain Mapping*, 39(6):2673–2688, March 2018.

[14] Diego Vidaurre, Laurence T. Hunt, Andrew J. Quinn, Benjamin A. E. Hunt, Matthew J. Brookes, Anna C. Nobre, and Mark W. Woolrich. Spontaneous cortical activity transiently organises into frequency specific phase-coupling networks. *Nature Communications*, 9(1), July 2018.

[15] Jalil Taghia, Weidong Cai, Srikanth Ryali, John Kochalka, Jonathan Nicholas, Tianwen Chen, and Vinod Menon. Uncovering hidden brain state dynamics that regulate performance and decision-making during cognition. *Nature Communications*, 9(1), June 2018.

[16] Takahiro Ezaki, Elohim Fonseca dos Reis, Takamitsu Watanabe, Michiko Sakaki, and Naoki Masuda. Closer to critical resting-state neural dynamics in individuals with higher fluid intelligence. *Communications Biology*, 3(1), February 2020.

[17] Vincent Zimmern. Why brain criticality is clinically relevant: A scoping review. *Frontiers in Neural Circuits*, 14, 2020.

[18] E. T. Jaynes. Information theory and statistical mechanics. *Phys. Rev.*, 106:620–630, May 1957.

[19] Giulio Ruffini, Giada Damiani, Diego Lozano-Soldevilla, Nikolas Deco, Fernando E. Rosas, Narsis A. Kiani, Adrián Ponce-Alvarez, Morten L. Kringelbach, Robin Carhart-Harris, and Gustavo Deco. Lsd-induced increase of ising temperature and algorithmic complexity of brain dynamics. *bioRxiv*, 2022.

[20] H. Chau Nguyen, Riccardo Zecchina, and Johannes Berg. Inverse statistical problems: from the inverse ising problem to data science. *Advances in Physics*, 66(3):197–261, June 2017.

[21] Bin Wang, Yan Niu, Liwen Miao, Rui Cao, Pengfei Yan, Hao Guo, Dandan Li, Yuxiang Guo, Tianyi Yan, Jinglong Wu, Jie Xiang, and Hui Zhang. Decreased complexity in alzheimer's disease: Resting-state fmri evidence of brain entropy mapping. *Frontiers in Aging Neuroscience*, 9, 2017.

[22] Jie Xiang, Yuan Tan, Yan Niu, Jie Sun, Nan Zhang, Dandan Li, and Bin Wang. Analysis of functional mri signal complexity based on permutation fuzzy entropy in bipolar disorder. *Neuroreport*, 32(6):465—471, April 2021.

[23] Soheil Keshmiri. Entropy and the brain: An overview. *Entropy*, 22(9):917, August 2020.

[24] Yan Niu, Jie Sun, Bin Wang, Waqar Hussain, Chanjuan Fan, Rui Cao, Mengni Zhou, and Jie Xiang. Comparing test-retest reliability of entropy methods: Complexity analysis of resting-state fmri. *IEEE Access*, 8:124437–124450, 2020.

[25] Yan Niu, Jie Sun, Bin Wang, Yanli Yang, Xin Wen, and Jie Xiang. Trajectories of brain entropy across lifetime estimated by resting state functional magnetic resonance imaging. *Human Brain Mapping*, 43(14):4359–4369, 2022.

[26] Lingzhong Fan, Hai Li, Junjie Zhuo, Yu Zhang, Jiaojian Wang, Liangfu Chen, Zhengyi Yang, Congying Chu, Sangma Xie, Angela R. Laird, Peter T. Fox, Simon B. Eickhoff, Chunshui Yu, and Tianzi Jiang. The human brainnetome atlas: A new brain atlas based on connectional architecture. *Cerebral Cortex*, 26(8):3508–3526, May 2016.

[27] Osame Kinouchi and Mauro Copelli. Optimal dynamical range of excitable networks at criticality. 2006.

[28] Andrea Cavagna, Alessio Cimarelli, Irene Giardina, Giorgio Parisi, Raffaele Santagati, Fabio Stefanini, and Massimiliano Viale. Scale-free correlations in starling flocks. *Proceedings of the National Academy of Sciences*, 107(26):11865–11870, June 2010.

[29] Enzo Tagliazucchi, Pablo Balenzuela, Daniel Fraiman, and Dante R. Chialvo. Criticality in large-scale brain fmri dynamics unveiled by a novel point process analysis. *Frontiers in Physiology*, 3, 2012.

[30] P. Rämö, J. Kesseli, and O. Yli-Harja. Perturbation avalanches and criticality in gene regulatory networks. *Journal of Theoretical Biology*, 242(1):164–170, September 2006.

[31] Gustavo Deco and Viktor K. Jirsa. Ongoing cortical activity at rest: Criticality, multistability, and ghost attractors. *The Journal of Neuroscience*, 32(10):3366–3375, March 2012.

[32] Chris G. Langton. Computation at the edge of chaos: Phase transitions and emergent computation. *Physica D: Nonlinear Phenomena*, 42(1–3):12–37, June 1990.

[33] Nils Bertschinger and Thomas Natschläger. Real-time computation at the edge of chaos in recurrent neural networks. *Neural Computation*, 16(7):1413–1436, July 2004.

[34] Sacha Friedli and Yvan Velenik. *Statistical Mechanics of Lattice Systems: A Concrete Mathematical Introduction.* Cambridge University Press, November 2017.

[35] Vince D. Calhoun, Robyn Miller, Godfrey Pearlson, and Tulay Adalı. The chronnectome: Time-varying connectivity networks as the next frontier in fmri data discovery. *Neuron*, 84(2):262–274, October 2014.

[36] Mikhail I. Rabinovich, Alan N. Simmons, and Pablo Varona. Dynamical bridge between brain and mind. *Trends in Cognitive Sciences*, 19(8):453–461, August 2015.

[37] Aron K. Barbey. Network neuroscience theory of human intelligence. *Trends in Cognitive Sciences*, 22(1):8–20, January 2018.

[38] Klaus Linkenkaer-Hansen, Simo Monto, Heikki Rytsälä, Kirsi Suominen, Erkki Isometsä, and Seppo Kähkönen. Breakdown of long-range temporal correlations in theta oscillations in patients with major depressive disorder. *Journal of Neuroscience*, 25(44):10131–10137, 2005.

[39] Jun-Seok Lee, Byung-Hwan Yang, Jang-Han Lee, Jun-Ho Choi, Ihn-Geun Choi, and Sae-Byul Kim. Detrended fluctuation analysis of resting eeg in depressed outpatients and healthy controls. *clinical Neurophysiology*, 118(11):2489–2496, 2007.

[40] Samuel Leistedt, Martine Dumont, J-P Lanquart, Fabrice Jurysta, and Paul Linkowski. Characterization of the sleep eeg in acutely depressed men using detrended fluctuation analysis. *Clinical neurophysiology*, 118(4):940–950, 2007.

[41] Xavier Bornas, Miquel Noguera, Maria Balle, Alfonso Morillas-Romero, Blanca Aguayo-Siquier, Miquel Tortella-Feliu, and Jordi Llabrés. Long-range temporal correlations in resting eeg. *Journal of Psychophysiology*, 2013.

[42] Matti Gärtner, Mona Irrmischer, Emilia Winnebeck, Maria Fissler, Julia M Huntenburg, Titus A Schroeter, Malek Bajbouj, Klaus Linkenkaer-Hansen, Vadim V Nikulin, and Thorsten Barnhofer. Aberrant long-range temporal correlations in depression are attenuated after psychological treatment. *Frontiers in human neuroscience*, 11:340, 2017.

[43] VB Slezin, EA Korsakova, MA Dytjatkovsky, EA Schultz, TA Arystova, and JR Siivola. Multifractal analysis as an aid in the diagnostics of mental disorders. *Nordic Journal of Psychiatry*, 61(5):339–342, 2007.

[44] Anca R Radulescu, Denis Rubin, Helmut H Strey, and Lilianne R Mujica-Parodi. Power spectrum scale invariance identifies prefrontal dysregulation in paranoid schizophrenia. *Human brain mapping*, 33(7):1582–1593, 2012.

[45] Vadim V Nikulin, Erik G Jönsson, and Tom Brismar. Attenuation of long-range temporal correlations in the amplitude dynamics of alpha and beta neuronal oscillations in patients with schizophrenia. *Neuroimage*, 61(1):162–169, 2012.

[46] James K Moran, Georgios Michail, Andreas Heinz, Julian Keil, and Daniel Senkowski. Long-range temporal correlations in resting state beta oscillations are reduced in schizophrenia. *Frontiers in psychiatry*, 10:517, 2019.

[47] Denis Tolkunov, Denis Rubin, and Lilianne R Mujica-Parodi. Power spectrum scale invariance quantifies limbic dysregulation in trait anxious adults using fmri: adapting methods optimized for characterizing autonomic dysregulation to neural dynamic time series. *Neuroimage*, 50(1):72–80, 2010.

[48] Tomas Ros, Bernard J. Baars, Ruth A Lanius, and Patrik Vuilleumier. Tuning pathological brain oscillations with neurofeedback: a systems neuroscience framework. *Frontiers in human neuroscience*, 8:1008, 2014.

[49] John M. Beggs and Dietmar Plenz. Neuronal avalanches in neocortical circuits. *The Journal of Neuroscience*, 23(35):11167–11177, December 2003.

[50] John M Beggs. The criticality hypothesis: how local cortical networks might optimize information processing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1864):329–343, August 2007.

[51] Maximilian B. Kloucek, Thomas Machon, Shogo Kajimura, C. Patrick Royall, Naoki Masuda, and Francesco Turci. Biases in inverse ising estimates of near-critical behavior. *Physical Review E*, 108(1), jul 2023.

[52] Gasper Tkacik, Elad Schneidman, Michael J Berry, and William Bialek. Ising models for networks of real neurons, 2006.

[53] Yasser Roudi. Statistical physics of pairwise probability models. *Frontiers in Computational Neuroscience*, 3, 2009.

[54] Takamitsu Watanabe, Satoshi Hirose, Hiroyuki Wada, Yoshio Imai, Toru Machida, Ichiro Shirouzu, Seiki Konishi, Yasushi Miyashita, and Naoki Masuda. A pairwise maximum entropy model accurately describes resting-state human brain networks. *Nature Communications*, 4(1), January 2013.

[55] Igor Fortel, Mitchell Butler, Laura E. Korthauer, Liang Zhan, Olusola Ajilore, Anastasios Sidiropoulos, Yichao Wu, Ira Driscoll, Dan Schonfeld, and Alex Leow. Inferring excitation-inhibition dynamics using a maximum entropy model unifying brain structure and function. *Network Neuroscience*, 6(2):420–444, 2022.

[56] Erik Aurell and Magnus Ekeberg. Inverse ising inference using all the data. *Phys. Rev. Lett.*, 108:090201, March 2012.

[57] David Sherrington and Scott Kirkpatrick. Solvable model of a spin-glass. *Physical Review Letters*, 35(26):1792–1796, December 1975.

[58] Yumeng Xin, Tongjian Bai, Ting Zhang, Yang Chen, Kai Wang, Shan Yu, Ning Liu, and Yanghua Tian. Electroconvulsive therapy modulates critical brain dynamics in major depressive disorder patients. *Brain Stimulation*, 15(1):214–225, January 2022.

[59] B. T. Thomas Yeo, Fenna M. Krienen, Jorge Sepulcre, Mert R. Sabuncu, Danial Lashkari, Marisa Hollinshead, Joshua L. Roffman, Jordan W. Smoller, Lilla Zöllei, Jonathan R. Polimeni, Bruce Fischl, Hesheng Liu, and Randy L. Buckner. The organization of the human cerebral cortex estimated by intrinsic functional connectivity. *Journal of Neurophysiology*, 106(3):1125–1165, September 2011.

[60] Ernst Ising. Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik*, 31(1):253–258, February 1925.

[61] Boltzmann constant.

[62] K. H. Fischer and J. A. Hertz. *Spin Glasses*. Cambridge Studies in Magnetism. Cambridge University Press, 1991.

[63] David Sherrington. Spin glasses, 1998.

[64] S F Edwards and P W Anderson. Theory of spin glasses. *Journal of Physics F: Metal Physics*, 5(5):965–974, May 1975.

[65] Vitor Sessak. Inverse problems in spin models. October 2010.

[66] Elad Schneidman, Michael J. Berry, Ronen Segev, and William Bialek. Weak pairwise correlations imply strongly correlated network states in a neural population. *Nature*, 440(7087):1007–1012, April 2006.

[67] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[68] Julian Besag. Statistical analysis of non-lattice data. *The Statistician*, 24:179–195, 1975.

[69] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, June 1953.

[70] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, April 1970.

[71] Siddhartha Chib and Edward Greenberg. Understanding the metropolis-hastings algorithm. *The American Statistician*, 49(4):327, November 1995.

[72] Helmut G. Katzgraber. Introduction to monte carlo methods, 2009.

[73] M. E. J. Newman and G. T. Barkema. *Monte Carlo methods in statistical physics*. Clarendon Press, Oxford, 1999.

[74] Takahiro Ezaki and Naoki Masuda. *Energy Landscape Analysis Toolbox (ELAT) User's Guide (ver. 3.2)*, January 2022. Available at `https://github.com/tkEzaki/energy-landscape-analysis/blob/master/users_guide.pdf`.

[75] Lili Jiang, Kaini Qiao, and Chunlin Li. Distance-based functional criticality in the human brain: intelligence and emotional intelligence. *BMC Bioinformatics*, 22(1), January 2021.

[76] Longzhou Xu, Jianfeng Feng, and Lianchun Yu. Avalanche criticality in individuals, fluid intelligence, and working memory. *Human Brain Mapping*, 43(8):2534–2553, February 2022.

[77] Kathleen T Brady, Therese K Killeen, Tim Brewerton, and Sylvia Lucerini. Comorbidity of psychiatric disorders and posttraumatic stress disorder. *Journal of clinical psychiatry*, 61:22–32, 2000.

[78] Kelly A. Knowles, Rebecca K. Sripada, Mahrie Defever, and Sheila A. M. Rauch. Comorbid mood and anxiety disorders and severity of posttraumatic stress disorder symptoms in treatment-seeking veterans. *Psychological Trauma: Theory, Research, Practice, and Policy*, 11(4):451–458, May 2019.

[79] Phase diagram of the sherrinton-kirkpatrick (sk) model. `https://github.com/elohimfr/sk_model`, 2020.

[80] Sigrid Rouam. *False Discovery Rate (FDR)*, page 731–732. Springer New York, 2013.

[81] Moses Sokunbi, Roger Staff, Gordon Waiter, George Cameron, Trevor Ahearn, and Alison Murray. Functional mri entropy measurements of age-related brain changes. 06 2011.

[82] Mianxin Liu, Xinyang Liu, Andrea Hildebrandt, and Changsong Zhou. Individual cortical entropy profile: Test–retest reliability, predictive power for cognitive ability, and neuroanatomical foundation. *Cerebral Cortex Communications*, 1(1):tgaa015, 2020.

[83] Moses O. Sokunbi, Victoria B. Gradin, Gordon D. Waiter, George G. Cameron, Trevor S. Ahearn, Alison D. Murray, Douglas J. Steele, and Roger T. Staff. Nonlinear complexity analysis of brain fmri signals in schizophrenia. *PLoS ONE*, 9(5):e95146, May 2014.

[84] Moses O. Sokunbi, Wilson Fung, Vijay Sawlani, Sabine Choppin, David E.J. Linden, and Johannes Thome. Resting state fmri entropy probes complexity of brain activity in adults with adhd. *Psychiatry Research: Neuroimaging*, 214(3):341–348, December 2013.

[85] Chemin Lin, Shwu-Hua Lee, Chih-Mao Huang, Guan-Yen Chen, Pei-Shan Ho, Ho-Ling Liu, Yao-Liang Chen, Tatia Mei-Chun Lee, and Shun-Chi Wu. Increased brain entropy of resting-state fmri mediates the relationship between depression severity and mental health-related quality of life in late-life depressed elderly. *Journal of Affective Disorders*, 250:270–277, May 2019.

[86] Christoph Bandt and Bernd Pompe. Permutation entropy: A natural complexity measure for time series. *Physical Review Letters*, 88(17), April 2002.

[87] Weiting Chen, Zhizhong Wang, Hongbo Xie, and Wangxin Yu. Characterization of surface emg signal based on fuzzy entropy. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 15(2):266–272, 2007.

[88] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, June 1965.

[89] Waqar Hussain, Muhammad Shahid Iqbal, Jie Xiang, Bin Wang, Yan Niu, Yuan Gao, Xin Wang, Jie Sun, Qionghui Zhan, Rui Cao, and Zhou Mengni. Epileptic seizure detection with permutation fuzzy entropy using robust machine learning techniques. *IEEE Access*, 7:182238–182258, 2019.

[90] Y. Yao, W. L. Lu, B. Xu, C. B. Li, C. P. Lin, D. Waxman, and J. F. Feng. The increase of the functional entropy of the human brain with age. *Scientific Reports*, 3(1), October 2013.

[91] Liangfeng Kuang, Weijia Gao, Luoyu Wang, Yongxin Guo, Weifang Cao, Dong Cui, Qing Jiao, Jianfeng Qiu, Linyan Su, and Guangming Lu. Increased resting-state brain entropy of parahippocampal gyrus and dorsolateral prefrontal cortex in manic and euthymic adolescent bipolar disorder. *Journal of Psychiatric Research*, 143:106–112, November 2021.

[92] Shishun Fu, Sipei Liang, Chulan Lin, Yunfan Wu, Shuangcong Xie, Meng Li, Qiang Lei, Jianneng Li, Kanghui Yu, Yi Yin, Kelei Hua, Wuming Li, Caojun Wu, Xiaofen Ma, and Guihua Jiang. Aberrant brain entropy in posttraumatic stress disorder comorbid with major depressive disorder during the coronavirus disease 2019 pandemic. *Frontiers in Psychiatry*, 14, June 2023.

[93] Daniel Fraiman, Pablo Balenzuela, Jennifer Foss, and Dante R. Chialvo. Ising-like dynamics in large-scale functional brain networks. *Physical Review E*, 79(6), June 2009.

[94] Manfred G. Kitzbichler, Marie L. Smith, Søren R. Christensen, and Ed Bullmore. Broadband criticality of human brain network synchronization. *PLoS Computational Biology*, 5(3):e1000314, March 2009.

[95] Daniele Marinazzo, Mario Pellicoro, Guorong Wu, Leonardo Angelini, Jesús M. Cortés, and Sebastiano Stramaglia. Information transfer and criticality in the ising model on the human connectome. *PLoS ONE*, 9(4):e93616, April 2014.

[96] Jiyoung Kang, Seok-Oh Jeong, Chongwon Pae, and Hae-Jeong Park. Bayesian estimation of maximum entropy model for individualized energy landscape analysis of brain state dynamics. *Human Brain Mapping*, 42(11):3411–3428, May 2021.

[97] Seok-Oh Jeong, Jiyoung Kang, Chongwon Pae, Jinseok Eo, Sung Min Park, Junho Son, and Hae-Jeong Park. Empirical bayes estimation of pairwise maximum entropy model for nonlinear brain state dynamics. *NeuroImage*, 244:118618, 2021.

[98] Pierluigi Contucci, Ignacio Gallo, and Giulia Menconi. Phase transitions in social sciences: Two-population mean field theory. *International Journal of Modern Physics B*, 22(14):2199–2212, June 2008.

Appendix A

Derivations

A.1 Derivation MLE Updating Steps for $\boldsymbol{h}$ and $\boldsymbol{J}$

In order to obtain the updating steps for (pseudo)likelihood maximization, we first have to derive the derivatives of the log-likelihood function $\mathcal{L}(\mathrm{D}|\boldsymbol{h},\boldsymbol{J})$ w.r.t. h_i and J_{ij} for $i,j=1,\ldots,N$.

The log-likelihood $\mathcal{L}(\boldsymbol{h},\boldsymbol{J})$, (defined in Equation (3.7)) is given by

$$\mathcal{L}(D|\boldsymbol{h},\boldsymbol{J}) = \beta\sum_{i=1}^{N} h_i\langle\sigma_i\rangle_{\text{Empirical}} + \beta\sum_{j=1,j\neq i}^{N} J_{ij}\langle\sigma_i\sigma_j\rangle_{\text{Empirical}} - \ln Z(\boldsymbol{h},\boldsymbol{J},\beta).$$

It follows directly that

$$\frac{\partial}{\partial h_i}\mathcal{L}(D|\boldsymbol{h},\boldsymbol{J}) = \beta\langle\sigma_i\rangle_{\text{Empirical}} - \frac{\partial}{\partial h_i}\ln Z(\boldsymbol{h},\boldsymbol{J},\beta) \tag{A.1}$$

$$\frac{\partial}{\partial J_{ij}}\mathcal{L}(D|\boldsymbol{h},\boldsymbol{J}) = \beta\langle\sigma_i\sigma_j\rangle_{\text{Empirical}} - \frac{\partial}{\partial J_{ij}}\ln Z(\boldsymbol{h},\boldsymbol{J},\beta). \tag{A.2}$$

The derivative of the partition function $Z(\boldsymbol{h},\boldsymbol{J},\beta)$ w.r.t. h_i $(i=1,\ldots,N)$ is given by

$$\begin{aligned}
\frac{\partial}{\partial h_i}\ln Z(\boldsymbol{h},\boldsymbol{J},\beta) &= \frac{1}{Z(\boldsymbol{h},\boldsymbol{J},\beta)}\frac{\partial}{\partial h_i}\sum_{\{\boldsymbol{\sigma}\}}\exp\left[-\beta\mathcal{H}(\boldsymbol{\sigma}|\boldsymbol{h},\boldsymbol{J})\right] \\
&= \frac{1}{Z(\boldsymbol{h},\boldsymbol{J},\beta)}\frac{\partial}{\partial h_i}\sum_{\{\boldsymbol{\sigma}\}}\exp\left[\beta\sum_{i=1}^{N}h_i\sigma_i + \beta\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N}J_{ij}\sigma_i\sigma_j\right] \\
&= \beta\frac{1}{Z(\boldsymbol{h},\boldsymbol{J},\beta)}\sum_{\{\boldsymbol{\sigma}\}}\sigma_i\exp\left[-\beta\mathcal{H}(\boldsymbol{\sigma}|\boldsymbol{h},\boldsymbol{J})\right] \\
&= \beta\langle\sigma_i\rangle_P.
\end{aligned} \tag{A.3}$$

and w.r.t. J_{ij} gives

$$\begin{aligned}
\frac{\partial}{\partial J_{ij}} \ln Z(\boldsymbol{h}, \boldsymbol{J}, \beta) &= \frac{1}{Z(\boldsymbol{h}, \boldsymbol{J}, \beta)} \frac{\partial}{\partial J_{ij}} \sum_{\{\boldsymbol{\sigma}\}} \exp\left[-\beta \mathcal{H}(\boldsymbol{\sigma}|\boldsymbol{h}, \boldsymbol{J})\right] \\
&= \frac{1}{Z(\boldsymbol{h}, \boldsymbol{J}, \beta)} \frac{\partial}{\partial J_{ij}} \sum_{\{\boldsymbol{\sigma}\}} \exp\left[\beta \sum_{i=1}^{N} h_i \sigma_i + \beta \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} J_{ij} \sigma_i \sigma_j\right] \\
&= \beta \frac{1}{Z(\boldsymbol{h}, \boldsymbol{J}, \beta)} \sum_{\{\boldsymbol{\sigma}\}} \sigma_i \sigma_j \exp\left[-\beta \mathcal{H}(\boldsymbol{\sigma}|\boldsymbol{h}, \boldsymbol{J})\right] \\
&= \beta \langle \sigma_i \sigma_j \rangle_P. \qquad \text{(A.4)}
\end{aligned}$$

Substituting the resulting derivatives given by Equations (A.3) and (A.4) into the Equations (A.1) resp. (A.2), we conclude that

$$\begin{aligned}
\frac{\partial}{\partial h_i} \mathcal{L}(D|\boldsymbol{h}, \boldsymbol{J}) &= \beta \left(\langle \sigma_i \rangle_{\text{Empirical}} - \langle \sigma_i \rangle_P\right) \\
\frac{\partial}{\partial J_{ij}} \mathcal{L}(D|\boldsymbol{h}, \boldsymbol{J}) &= \beta \left(\langle \sigma_i \sigma_j \rangle_{\text{Empirical}} - \langle \sigma_i \sigma_j \rangle_P\right).
\end{aligned}$$

A.2 Concavity of the Log-Likelihood Function in h and J

In order to use convex (or concave) optimization techniques [67], we have to show that the log-likelihood function $\mathcal{L}(D|\boldsymbol{h}, \boldsymbol{J})$ is strictly concave in the parameters $\boldsymbol{h}$ and $\boldsymbol{J}$. As a result, we can conclude that the maximum of the log-likelihood is uniquely determined. We will follow the intuition of the proof by Nguyen et al. [20].

Recall that a twice differentiable function f is strictly concave if and only if its Hessian matrix $D^2 f(\boldsymbol{x})$ is negative definite [67]. The aim is to use this theorem for proving concavity of the log-likelihood function.

For simplicity, let $\boldsymbol{\theta} = (\boldsymbol{h}, \boldsymbol{J})$, i.e. $\theta_i = (h_i, J_{i1}, \ldots, J_{iN})$, and $\boldsymbol{Q}(\boldsymbol{\sigma}) = \left(\boldsymbol{\sigma}, \boldsymbol{\sigma}\boldsymbol{\sigma}^T\right)$ implying that $Q_k(\boldsymbol{\sigma}) = (\sigma_k, \sigma_k\sigma_1, \ldots \sigma_k\sigma_N)$. Without loss of generality, assume $\beta = 1$, such that we the Boltzmann distribution is given by

$$P(\boldsymbol{\sigma}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left[\sum_k \theta_k \left(Q_k(\boldsymbol{\sigma})\right)^T\right],$$

with partition function

$$Z(\boldsymbol{\theta}, \beta) = \sum_{\boldsymbol{\sigma}} \exp\left[\sum_k \theta_k \left(Q_k(\boldsymbol{\sigma})\right)^T\right].$$

and, subsequently, the log-likelihood function

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}) &= \ln P(\boldsymbol{\sigma}) \\
&= \ln \frac{1}{Z(\boldsymbol{\theta}, \beta)} \exp\left[\sum_k \theta_k \left(Q_k(\boldsymbol{\sigma})\right)^T\right].
\end{aligned}$$

It follows that the first and second derivative of the log-likelihood $\mathcal{L}$ are defined by

$$\begin{aligned}
\frac{\partial}{\partial \theta_i} \mathcal{L}(\boldsymbol{\theta}) &= \frac{\partial}{\partial \theta_i} \ln \frac{1}{Z(\boldsymbol{\theta})} \exp\left[\sum_k \theta_k \left(Q_k(\boldsymbol{\sigma})\right)^T\right] \\
&= \frac{\partial}{\partial \theta_i} \sum_k \theta_k \left(Q_k(\boldsymbol{\sigma})\right)^T - \frac{1}{Z(\boldsymbol{\theta})} \frac{\partial}{\partial \theta_i} Z(\boldsymbol{\theta}) \\
&= Q_i(\boldsymbol{\sigma}) - \frac{1}{Z(\boldsymbol{\theta})} \frac{\partial}{\partial \theta_i} Z(\boldsymbol{\theta})
\end{aligned}$$

respectively

$$\frac{\partial^2}{\partial\theta_i\partial\theta_j}\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{Z(\boldsymbol{\sigma})}\left(\frac{\partial^2}{\partial\theta_i\partial\theta_j}Z(\boldsymbol{\theta}) - \left(\frac{\partial}{\partial\theta_i}Z(\boldsymbol{\theta})\right)^2\right).$$

Using

$$\frac{\partial}{\partial\theta_i}Z(\boldsymbol{\sigma}) = \sum_{\boldsymbol{\sigma}} Q_i(\boldsymbol{\sigma})\exp\left[\sum_k \theta_k \left(Q_k(\boldsymbol{\sigma})\right)^T\right]$$

and

$$\frac{\partial^2}{\partial\theta_i\partial\theta_j}Z(\boldsymbol{\sigma}) = \sum_{\boldsymbol{\sigma}} Q_i(\boldsymbol{\sigma})Q_j(\boldsymbol{\sigma})\exp\left[\sum_k \theta_k \left(Q_k(\boldsymbol{\sigma})\right)^T\right],$$

we may conclude that

$$\frac{\partial^2}{\partial\theta_i\partial\theta_j}\mathcal{L}(\boldsymbol{\theta}) = -\left(\langle Q_i(\boldsymbol{\sigma})Q_j(\boldsymbol{\sigma})\rangle - \langle Q_i(\boldsymbol{\sigma})\rangle\langle Q_j(\boldsymbol{\sigma})\rangle\right).$$

The only step that remains is showing that the Hessian matrix $H(\mathcal{L})$ is negative definite on its domain. Recall that a matrix A is negative definite if and only if $x^TAx \leq 0$ for all $x \in \mathbb{R}^n$. It can be verified easily that

$$\begin{aligned}
x^TH(\mathcal{L})x &= \sum_{i,j=1}^{N}\frac{\partial^2}{\partial\theta_i\partial\theta_j}\mathcal{L}(\boldsymbol{\theta})x_ix_j \\
&= -\sum_{i,j=1}^{N}\left(\langle Q_i(\boldsymbol{\sigma})Q_j(\boldsymbol{\sigma})\rangle - \langle Q_i(\boldsymbol{\sigma})\rangle\langle Q_j(\boldsymbol{\sigma})\rangle\right)x_ix_j \\
&= \left\langle\left(\sum_{k=1}^{N}(x_kQ_k(\boldsymbol{\sigma}) - \langle x_kQ_k(\boldsymbol{\sigma})\rangle\right)^2\right\rangle \\
&\leq 0.
\end{aligned}$$

Consequently, we may conclude that the Hessian matrix is negative-definite and therefore the log-likelihood is concave. This implies that it has a unique maximum and allows us to utilize convex (or, equivalently, concave) optimization techniques.

A.3 Derivation of the PLM Updating Steps for h and J

Given the log pseudolikelihood

$$\begin{aligned}
\mathcal{L}(D|\boldsymbol{h},\boldsymbol{J}) &= \frac{1}{B}\sum_{i=1}^{N}\sum_{t=1}^{B}\tilde{P}(\sigma_i(t)|\boldsymbol{h}_i,\boldsymbol{J}_{i*},\boldsymbol{\sigma}(t)_{/i}) \\
&= \frac{1}{B}\sum_{i=1}^{N}\sum_{t=1}^{B}\ln\frac{1}{2}\left(1+\sigma_i(t)\tanh\left[\beta\left(h_i+\frac{1}{2}\sum_{j=1,j\neq i}^{N}J_{ij}\sigma_j(t)\right)\right]\right),
\end{aligned}$$

it follows that

$$\begin{aligned}
\frac{\partial}{\partial h_k}\log\mathcal{L}(D|\boldsymbol{h},\boldsymbol{J}) &= \frac{\partial}{\partial h_k}\frac{1}{B}\sum_{i=1}^{N}\sum_{t=1}^{B}\ln\frac{1}{2}\left(1+\sigma_i(t)\tanh\left[\beta\left(h_i+\frac{1}{2}\sum_{j=1,j\neq i}^{N}J_{ij}\sigma_j(t)\right)\right]\right) \\
&= \frac{1}{B}\sum_{t=1}^{B}\beta\left(\sigma_k(t)-\tanh\left[\beta\left(h_k+\sum_{j=1,j\neq k}^{N}J_{kj}\sigma_j(t)\right)\right]\right) \\
&= \underbrace{\frac{1}{B}\sum_{t=1}^{B}\beta\sigma_k(t)}_{=\beta\langle\sigma_k\rangle_{\text{Empirical}}} - \underbrace{\frac{1}{B}\sum_{t=1}^{B}\beta\tanh\left[\beta\left(h_k+\sum_{j=1,j\neq k}^{N}J_{kj}\sigma_j(t)\right)\right]}_{=\beta\langle\overline{\sigma_k}\rangle_{\tilde{P}}},
\end{aligned}$$

where we used the fact that

$$\frac{d}{dx}\ln\frac{1}{2}\left[1\pm\tanh f(x)\right]=\frac{d}{dx}\left[\pm f(x)-\ln\left[e^{f(x)}+e^{-f(x)}\right]\right]=f'(x)\left(\pm 1-\tanh f(x)\right). \tag{A.5}$$

Furthermore, it holds that

$$\begin{aligned}\frac{\partial}{\partial J_{kl}}\log\mathcal{L}(D|\boldsymbol{h},\boldsymbol{J}) &= \frac{\partial}{\partial J_{kl}}\frac{1}{B}\sum_{i=1}^{N}\sum_{t=1}^{B}\ln\frac{1}{2}\left(1+\sigma_i(t)\tanh\left[\beta\left(h_i+\frac{1}{2}\sum_{j=1,j\neq i}^{N}J_{ij}\sigma_j(t)\right)\right]\right)\\ &= \frac{1}{B}\sum_{t=1}^{B}\beta\sigma_l(t)\left(\sigma_k(t)-\tanh\left[\beta\left(h_k+\sum_{j=1,j\neq k}^{N}J_{kj}\sigma_k(t)\right)\right]\right)\\ &= \underbrace{\frac{1}{B}\sum_{t=1}^{B}\beta\sigma_k(t)\sigma_l(t)}_{=\beta\langle\sigma_k\sigma_l\rangle_{\text{Empirical}}}-\underbrace{\frac{1}{B}\sum_{t=1}^{B}\beta\sigma_k(t)\tanh\left[\beta\left(h_k+\sum_{j=1,j\neq k}^{N}J_{kj}\sigma_j(t)\right)\right]}_{=\beta\langle\overline{\sigma_k\sigma_l}\rangle_{\tilde{P}}}.\end{aligned}$$

A.4 Concavity of the Log-Pseudolikelihood Function in h and J

Similar as for the log-likelihood function in Section A.2, we have to show that the log-pseudolikelihood function $\mathcal{L}(D|\boldsymbol{h},\boldsymbol{J})$ is strictly concave in the parameters $\boldsymbol{h}$ and $\boldsymbol{J}$ such that may conclude that the maximum of the log-likelihood is uniquely determined. This allows us to use convex (or concave) optimization techniques [67].

Let $\boldsymbol{\theta}=(\boldsymbol{h},\boldsymbol{J})$, and $\boldsymbol{Q}(\boldsymbol{\sigma})=\left(\boldsymbol{\sigma},\boldsymbol{\sigma}\boldsymbol{\sigma}^T\right)$. Consequently, $\theta_i=(h_i,J_{i1},\ldots,J_{iN})$, and $Q_i(\boldsymbol{\sigma})=(\sigma_i,\sigma_i\sigma_1,\ldots\sigma_i\sigma_N)$. Without loss of generality assume $\beta=1$. Then, the conditional Boltzmann distribution can be written as

$$\tilde{P}(\sigma_i(t)|\boldsymbol{\theta},\boldsymbol{\sigma}_{/i}(t))=\frac{\exp\left[\theta_i\left(Q_i(\boldsymbol{\sigma}(t))\right)^T\right]}{\exp\left[\theta_i\left(Q_i(\boldsymbol{\sigma}(t))\right)^T\right]+\exp\left[-\theta_i\left(Q_i(\boldsymbol{\sigma}(t))\right)^T\right]},$$

and the log-pseudolikelihood

$$\begin{aligned}\mathcal{L}(D|\boldsymbol{\theta}) &= \frac{1}{B}\sum_{i=1}^{N}\sum_{t=1}^{B}\ln\tilde{P}(\sigma_i(t)|\boldsymbol{\theta},\boldsymbol{\sigma}_{/i}(t))\\ &= \frac{1}{B}\sum_{i=1}^{N}\sum_{t=1}^{B}\ln\frac{\exp\left[\theta_i\left(Q_i(\boldsymbol{\sigma}(t))\right)^T\right]}{\exp\left[\theta_i\left(Q_i(\boldsymbol{\sigma}(t))\right)^T\right]+\exp\left[-\theta_i\left(Q_i(\boldsymbol{\sigma}(t))\right)^T\right]}\\ &= \frac{1}{B}\sum_{i=1}^{N}\sum_{t=1}^{B}\theta_i\left(Q_i(\boldsymbol{\sigma}(t))\right)^T-\ln\left(\exp\left[\theta_i\left(Q_i(\boldsymbol{\sigma}(t))\right)^T\right]+\exp\left[-\theta_i\left(Q_i(\boldsymbol{\sigma}(t))\right)^T\right]\right).\end{aligned}$$

It follows that the first and second derivative of the log-pseudolikelihood $\mathcal{L}$ are given by

$$\begin{aligned}\frac{\partial}{\partial\theta_k}\mathcal{L}(D|\boldsymbol{\theta}) &= \frac{1}{B}\sum_{t=1}^{B}Q_k(\boldsymbol{\sigma}(t))^T-Q_k(\boldsymbol{\sigma}(t))^T\frac{\exp\left[\theta_iQ_i(\boldsymbol{\sigma}(t))^T\right]-\exp\left[-\theta_iQ_i(\boldsymbol{\sigma}(t))^T\right]}{\exp\left[\theta_iQ_i(\boldsymbol{\sigma}(t))^T\right]+\exp\left[-\theta_iQ_i(\boldsymbol{\sigma}(t))^T\right]}\\ &= \frac{1}{B}\sum_{t=1}^{B}Q_k(\boldsymbol{\sigma}(t))^T-Q_k(\boldsymbol{\sigma}(t))^T\tanh\left[-\theta_iQ_i(\boldsymbol{\sigma}(t))^T\right]\end{aligned}$$

and respectively

$$\frac{\partial^2}{\partial\theta_k^2}\mathcal{L}(D|\boldsymbol{\theta})=\frac{1}{B}\sum_{t=1}^{B}-Q_k(\boldsymbol{\sigma}(t))Q_k(\boldsymbol{\sigma}(t))^T\left(1-\tanh^2\left[-\theta_iQ_i(\boldsymbol{\sigma}(t))^T\right]\right).$$

Furthermore, it holds that

$$\frac{\partial^2}{\partial\theta_k\partial\theta_l}\mathcal{L}(D|\boldsymbol{\theta}) = 0.$$

which implies that

$$\begin{aligned}
x^T H(\mathcal{L})x &= \sum_{i,j=1}^{N} \frac{\partial^2}{\partial\theta_i\partial\theta_j}\mathcal{L}(\boldsymbol{D}|\boldsymbol{\theta})x_i x_j \\
&= \sum_{k=1}^{N} \frac{\partial^2}{\partial\theta_k^2}\mathcal{L}(\boldsymbol{D}|\boldsymbol{\theta})x_k^2 \\
&= \frac{1}{B}\sum_{t=1}^{B} -Q_k(\boldsymbol{\sigma}(t))Q_k(\boldsymbol{\sigma}(t))^T \left(1 - \tanh^2\left[-\theta_i Q_i(\boldsymbol{\sigma}(t))^T\right]\right). \\
&\leq 0,
\end{aligned}$$

because, clearly, it holds that $Q_k(\boldsymbol{\sigma}(t))Q_k(\boldsymbol{\sigma}(t))^T \geq 0$ and $\left(1 - \tanh^2\left[-\theta_i Q_i(\boldsymbol{\sigma}(t))^T\right]\right) \geq 0$. Consequently, the Hessian matrix $H(\mathcal{L})$ is negative-definite implying that the log-likelihood is concave.

A.5 Derivation PLM Updating Steps for β

Given the conditional Boltzmann distribution

$$\tilde{P}(\sigma_i|\boldsymbol{h}_i, \boldsymbol{J}_{i*}, \beta, \boldsymbol{\sigma}_{/i}(t)) = \frac{1}{2}\left(1 + \sigma_i \tanh\left[\beta\left(h_i + \frac{1}{2}\sum_{j=1,j\neq i}^{N} J_{ij}\sigma_j\right)\right]\right),$$

we have to determine the derivative of the log pseudolikelihood with respect to β

$$\mathcal{L}(D|\beta) = \frac{1}{B}\sum_{i=1}^{N}\sum_{t=1}^{B} \ln\frac{1}{2}\left(1 + \sigma_i(t)\tanh\left[\beta\left(h_i + \frac{1}{2}\sum_{j=1,j\neq i}^{N} J_{ij}\sigma_j(t)\right)\right]\right).$$

Using Equation (A.5), it follows that

$$\begin{aligned}
\frac{\partial}{\partial\beta}\mathcal{L}(D|\beta) &= \frac{\partial}{\partial\beta}\frac{1}{B}\sum_{i=1}^{N}\sum_{t=1}^{B} \ln\frac{1}{2}\left(1 + \sigma_i(t)\tanh\left[\beta\left(h_i + \frac{1}{2}\sum_{j=1,j\neq i}^{N} J_{ij}\sigma_j(t)\right)\right]\right) \\
&= \frac{1}{B}\sum_{i=1}^{N}\sum_{t=1}^{B}\left(h_i + \frac{1}{2}\sum_{j=1,j\neq i}^{N} J_{ij}\sigma_j(t)\right)\left(\sigma_i(t) - \tanh\left[\beta\left(h_i + \frac{1}{2}\sum_{j=1,j\neq i}^{N} J_{ij}\sigma_j(t)\right)\right]\right) \\
&= \frac{1}{B}\sum_{t=1}^{B}\sum_{i=1}^{N}\sigma_i(t)\left(h_i + \frac{1}{2}\sum_{j=1,j\neq i}^{N} J_{ij}\sigma_j(t)\right)\ldots \\
&\qquad - \frac{1}{B}\sum_{t=1}^{B}\sum_{i=1}^{N}\left(h_i + \frac{1}{2}\sum_{j=1,j\neq i}^{N} J_{ij}\sigma_j(t)\right)\left(\tanh\left[\beta\left(h_i + \frac{1}{2}\sum_{j=1,j\neq i}^{N} J_{ij}\sigma_j(t)\right)\right]\right) \\
&= \langle\mathcal{H}(\boldsymbol{\sigma}(t)|\boldsymbol{h},\boldsymbol{J})\rangle_{\text{Empirical}} - \langle\overline{\mathcal{H}(t)(\boldsymbol{\sigma}|\boldsymbol{h},\boldsymbol{J})}\rangle_{\tilde{P}},
\end{aligned}$$

where

$$\langle\mathcal{H}(\boldsymbol{\sigma}|\boldsymbol{h},\boldsymbol{J})\rangle_{\text{Empirical}} = \frac{1}{B}\sum_{t=1}^{B}\sum_{i=1}^{N}\mathcal{H}_i(\boldsymbol{\sigma}(t)|\boldsymbol{h},\boldsymbol{J}) = \frac{1}{B}\sum_{t=1}^{B}\sum_{i=1}^{N}\sigma_i(t)\left(h_i + \frac{1}{2}\sum_{j=1,j\neq i}^{N} J_{ij}\sigma_j(t)\right)$$

and,

$$\begin{aligned}
\langle\overline{\mathcal{H}(\boldsymbol{\sigma}|\boldsymbol{h},\boldsymbol{J})}\rangle_{\tilde{P}} &= \frac{1}{B}\sum_{i=1}^{N}\sum_{t=1}^{B}\left(h_i+\frac{1}{2}\sum_{j=1,j\neq i}^{N}J_{ij}\sigma_j\right)\left(\tanh\left[\beta\left(h_i+\frac{1}{2}\sum_{j=1,j\neq i}^{N}J_{ij}\sigma_j\right)\right]\right)\\
&= \sum_{i=1}^{N}\left(\frac{1}{B}\sum_{t=1}^{B}h_i\tanh\left[\beta\left(h_i+\frac{1}{2}\sum_{j=1,j\neq i}^{N}J_{ij}\sigma_j(t)\right)\right]\dots\right.\\
&\qquad\left.-\frac{1}{B}\sum_{t=1}^{B}\frac{1}{2}\sum_{j=1,j\neq i}^{N}J_{ij}\sigma_j(t)\tanh\left[\beta\left(h_i+\frac{1}{2}\sum_{j=1,j\neq i}^{N}J_{ij}\sigma_j(t)\right)\right]\right)\\
&= \sum_{i=1}^{N}\left(h_i\langle\overline{\sigma_i}\rangle_{\tilde{P}}+\frac{1}{2}\sum_{j=1,j\neq i}^{N}J_{ij}\langle\overline{\sigma_i\sigma_j}\rangle_{\tilde{P}}\right).
\end{aligned}$$

A.6 Concavity of the log-pseudolikelihood function in β

In order to use convex (or concave) optimization techniques [67], we have to show again that the log-pseudolikelihood function $\mathcal{L}(D|\beta)$ is strictly concave in the parameter β.

Given is the log-pseudolikelihood

$$\mathcal{L}(D|\beta)=\frac{1}{B}\sum_{i=1}^{N}\sum_{t=1}^{B}\ln\tilde{P}(\sigma_i(t)|\boldsymbol{h}_i,\boldsymbol{J}_{i*},\beta,\boldsymbol{\sigma}(t)_{/i})=\frac{1}{B}\sum_{i=1}^{N}\sum_{t=1}^{B}\ln\frac{1}{2}\left(1+\sigma_i(t)\tanh\left[\beta\left(h_i+\frac{1}{2}\sum_{j=1,j\neq i}^{N}J_{ij}\sigma_j(t)\right)\right]\right).$$

In this case, we can use that a twice-differentiable function in one variable is concave if and only if the second derivative is negative on its entire domain [67]. Notice that

$$\frac{\partial}{\partial\beta}\mathcal{L}(D|\beta)=\frac{1}{B}\sum_{i=1}^{N}\sum_{t=1}^{B}\left(h_i+\frac{1}{2}\sum_{j=1,j\neq i}^{N}J_{ij}\sigma_j(t)\right)\left(\sigma_i(t)-\tanh\left[\beta\left(h_i+\frac{1}{2}\sum_{j=1,j\neq i}^{N}J_{ij}\sigma_j(t)\right)\right]\right),$$

and, using Equation A.5,

$$\frac{\partial^2}{\partial\beta^2}\mathcal{L}(D|\beta)=\frac{1}{B}\sum_{i=1}^{N}\sum_{t=1}^{B}-\left(h_i+\frac{1}{2}\sum_{j=1,j\neq i}^{N}J_{ij}\sigma_j(t)\right)^2\left(1-\tanh^2\left[\beta\left(h_i+\frac{1}{2}\sum_{j=1,j\neq i}^{N}J_{ij}\sigma_j(t)\right)\right]\right)\leq 0$$

because $\left(h_i+\frac{1}{2}\sum_{j=1,j\neq i}^{N}J_{ij}\sigma_j(t)\right)^2\geq 0$ and $\left(1-\tanh^2\left[\beta\left(h_i+\frac{1}{2}\sum_{j=1,j\neq i}^{N}J_{ij}\sigma_j(t)\right)\right]\right)\geq 0$, for all $\beta\in\mathbb{R}$. We can conclude that the log-pseudolikelihood is concave.

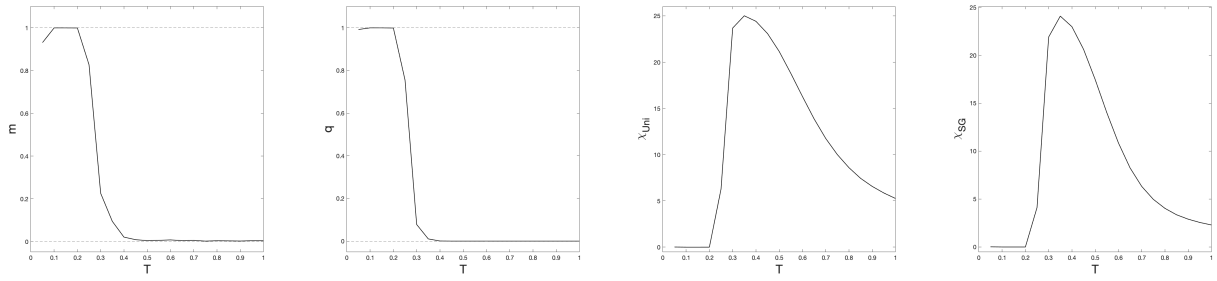
Appendix B

Supplementary figures

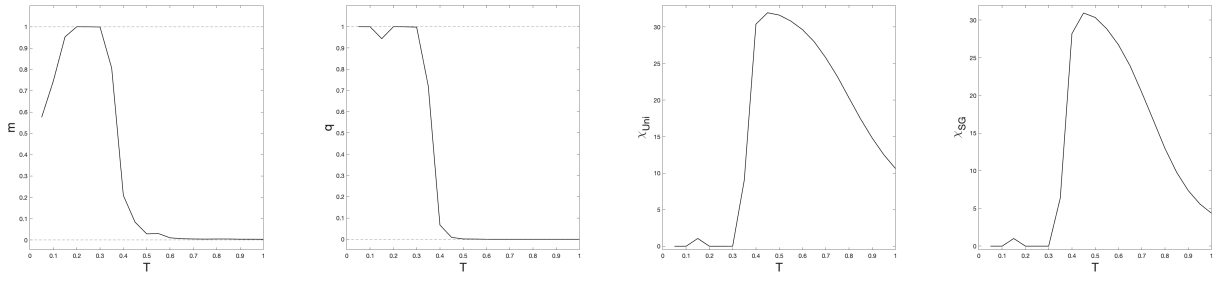
B.1 Robustness against variations in $\boldsymbol{h}$

B.1.1 Temperature Analysis

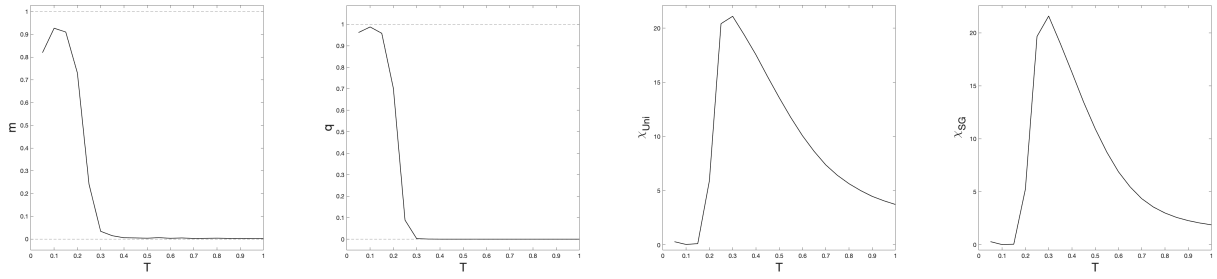
In order to ensure that the found phase transition (i.e., from ferro- to paramagnetic phase as the temperature increases) does not result from the fluctuations in the external field $\boldsymbol{h}$ and the tendency of the spins to align with it, we drew the plots for the observables m, q, χ_{SG} and χ_{Uni} as a function of the system temperature T also for $\boldsymbol{h} = 0$. The resulting graphs are shown in Figure B.3. We observe that the obtained curves are qualitatively similar to what we observed for the inferred external $\hat{\boldsymbol{h}}$ and does not change our conclusions about the observed phase transitions.



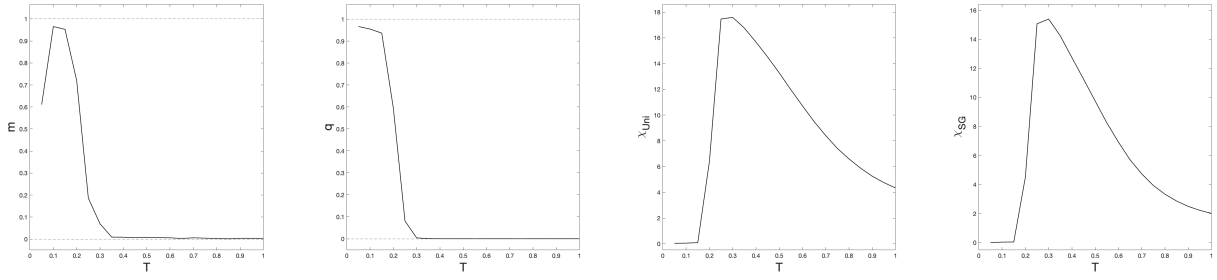
Functional network 1.



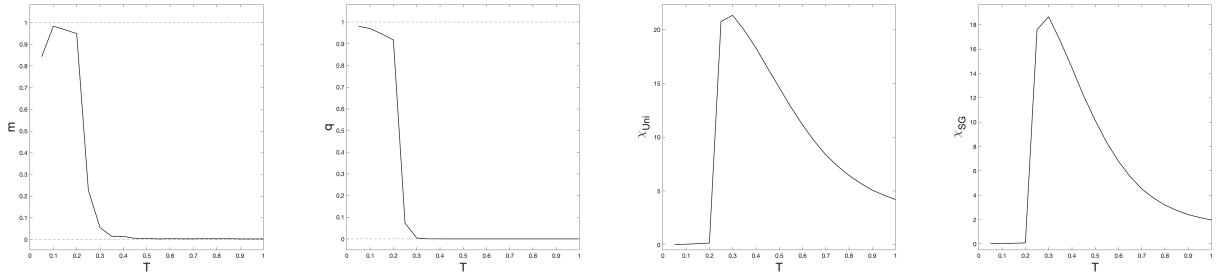
Functional network 2.



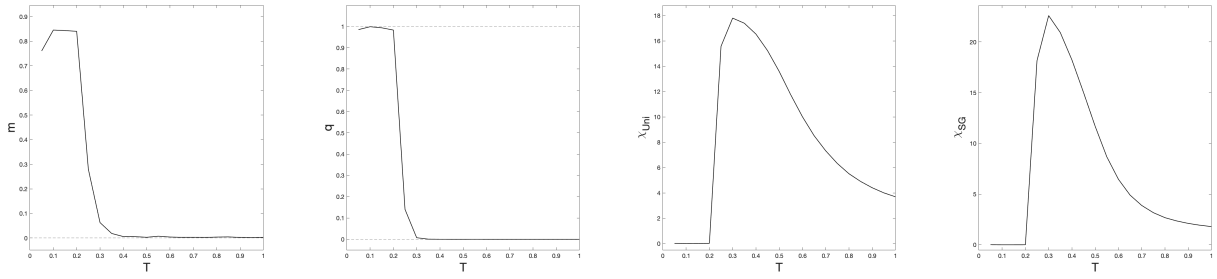
Functional network 3.



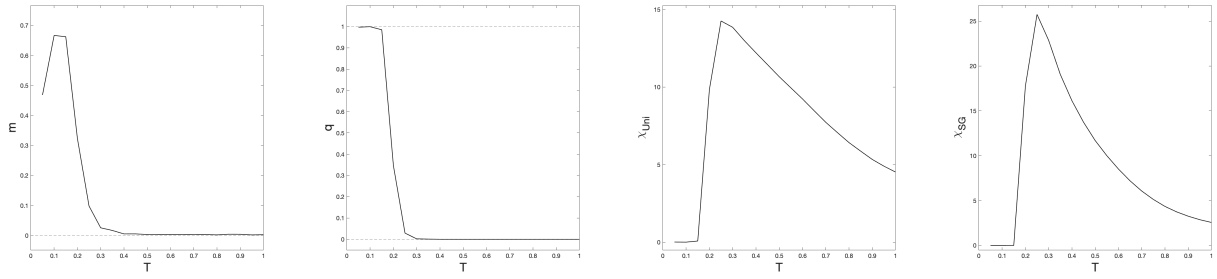
Functional network 4.



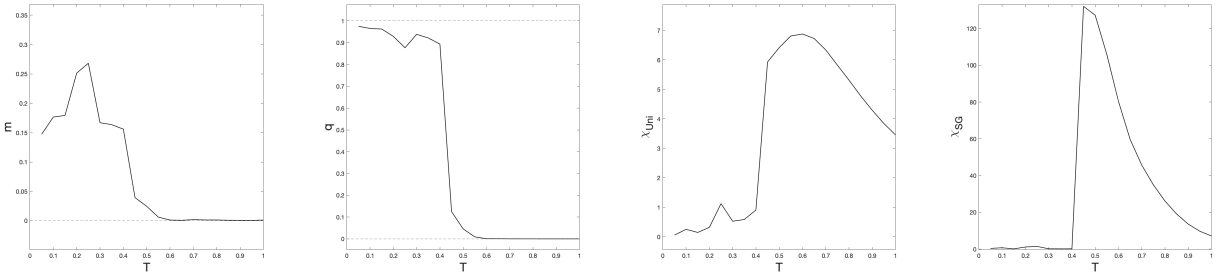
Functional network 5.



Functional network 6.



Functional network 7.



Whole-brain network

Figure B.3: Plot of the observables m, q, χ_{Uni} and χ_{SG} respectively, as a function of the system temperature T for the functional networks and the whole-brain network where $\boldsymbol{h} = 0$.

B.1.2 Phase Diagram Analysis

In order to test the robustness of the phase diagrams for the empirical data against variations in $\boldsymbol{h}$, we drew phase diagrams where $h_i = 0$ $(i = 1, \ldots, N)$ and where $h_i = -2 \times \max_{1 \leq i' \leq N} \hat{h}_i$ $(i = 1, \ldots, N)$. The resulting diagrams are shown in Figure B.4 until B.11. Here, the first row of phase diagrams corresponds to the case where $h_i = 0$ $(i = 1, \ldots, N)$, the second row where $h_i = -2 \times \max_{1 \leq i' \leq N} \hat{h}_i$ $(i = 1, \ldots, N)$. We observe that the phase diagrams are qualitatively similar to each other.

To save time, we reduced the accuracy of the phase diagrams by using less thermalization steps (10^6 instead of 10^7) and by generating samples ($B/500$ instead of $B = 46 \cdot 310$).

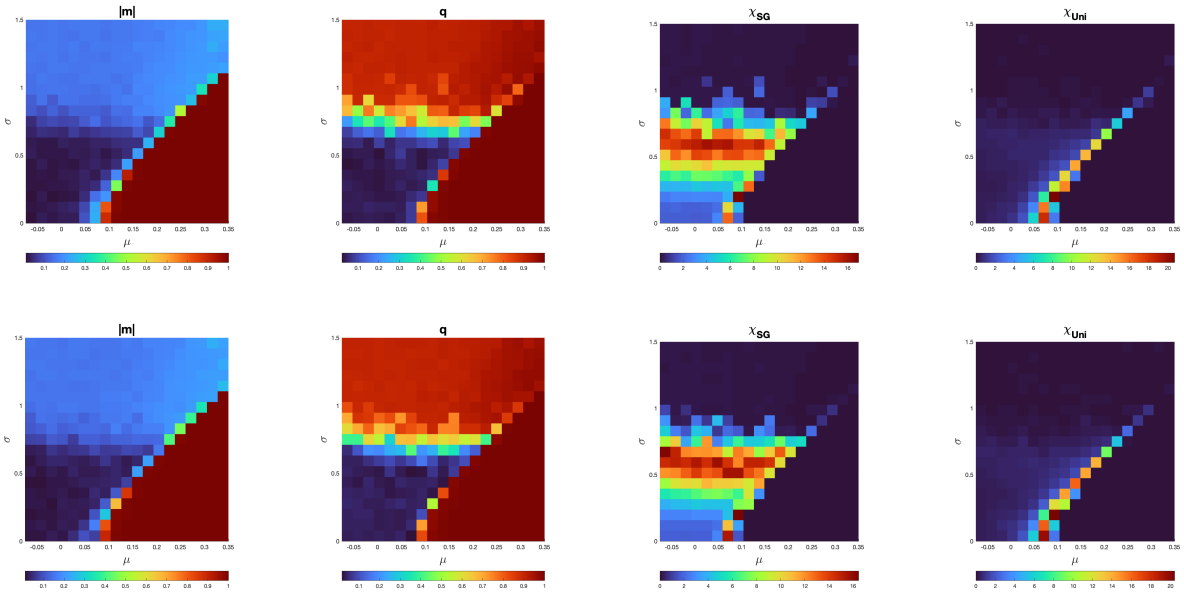


Figure B.4: Functional network 1.

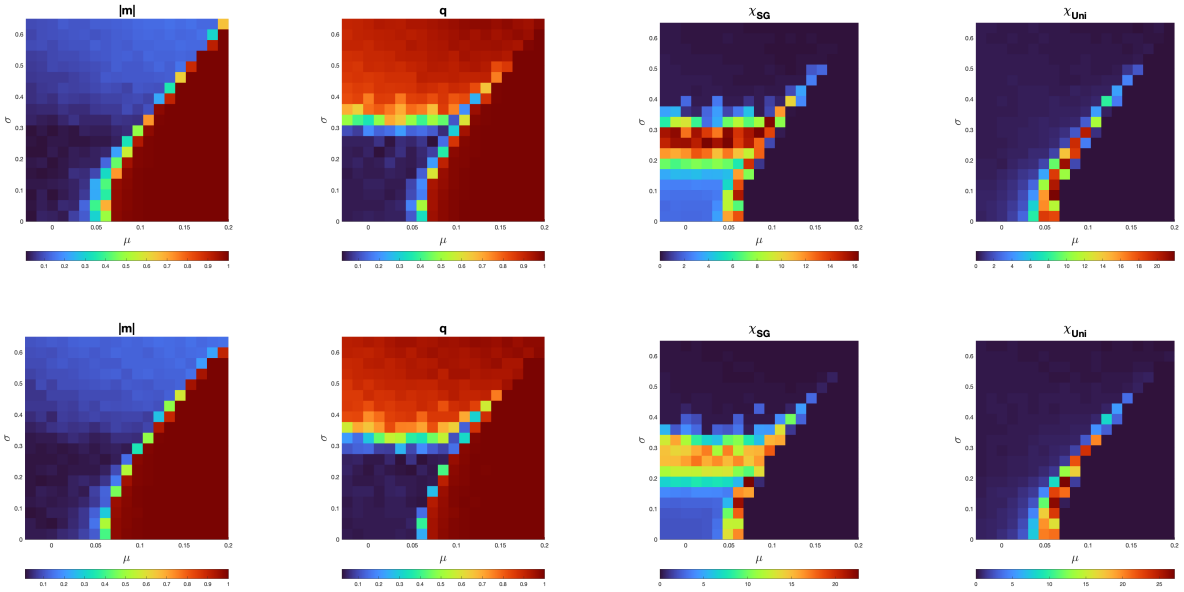


Figure B.5: Functional network 2.

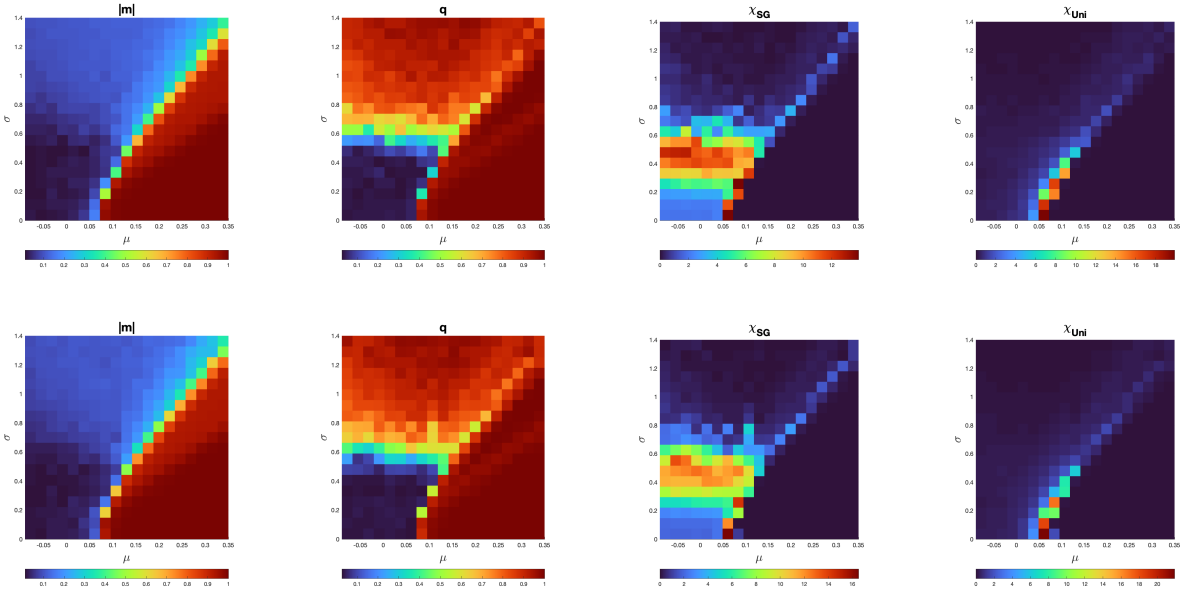


Figure B.6: Functional network 3.

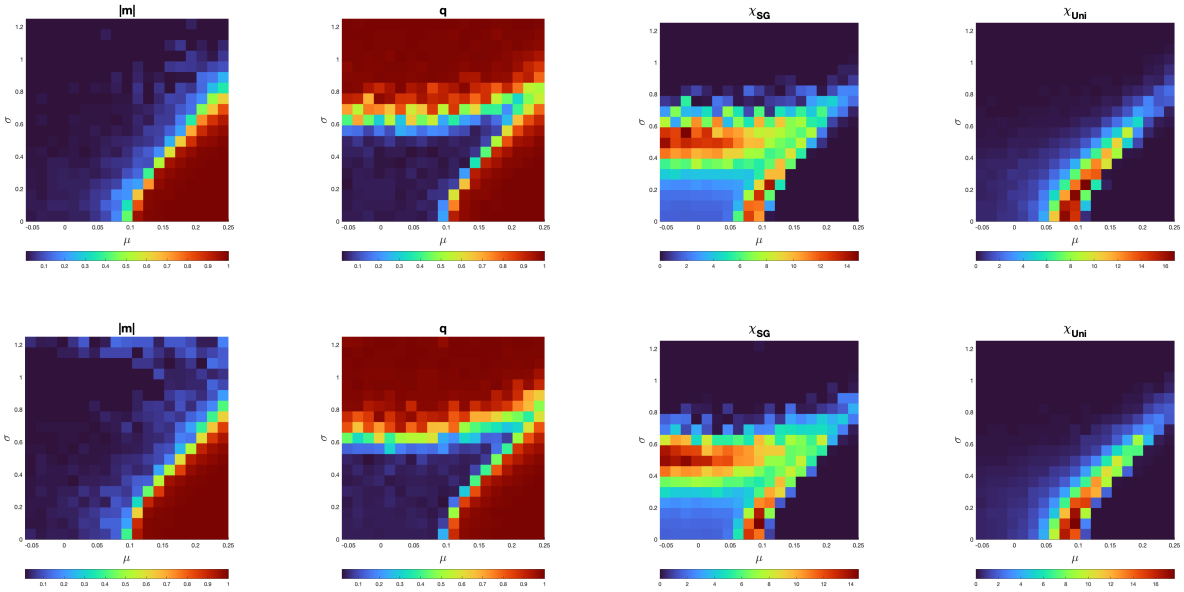


Figure B.7: Functional network 4.

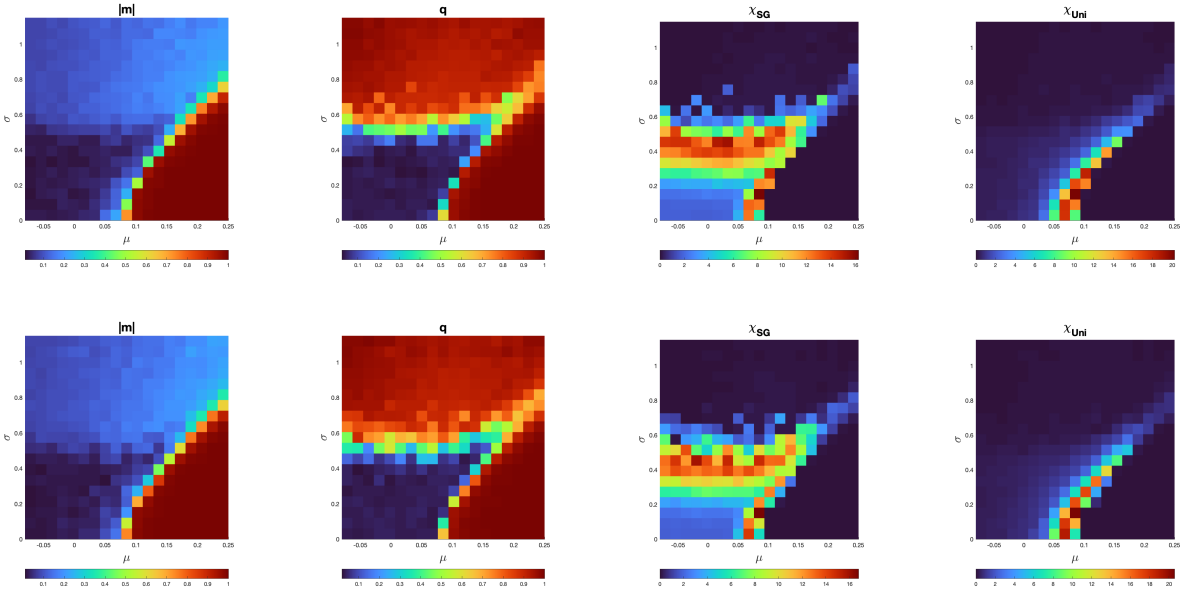


Figure B.8: Functional network 5.

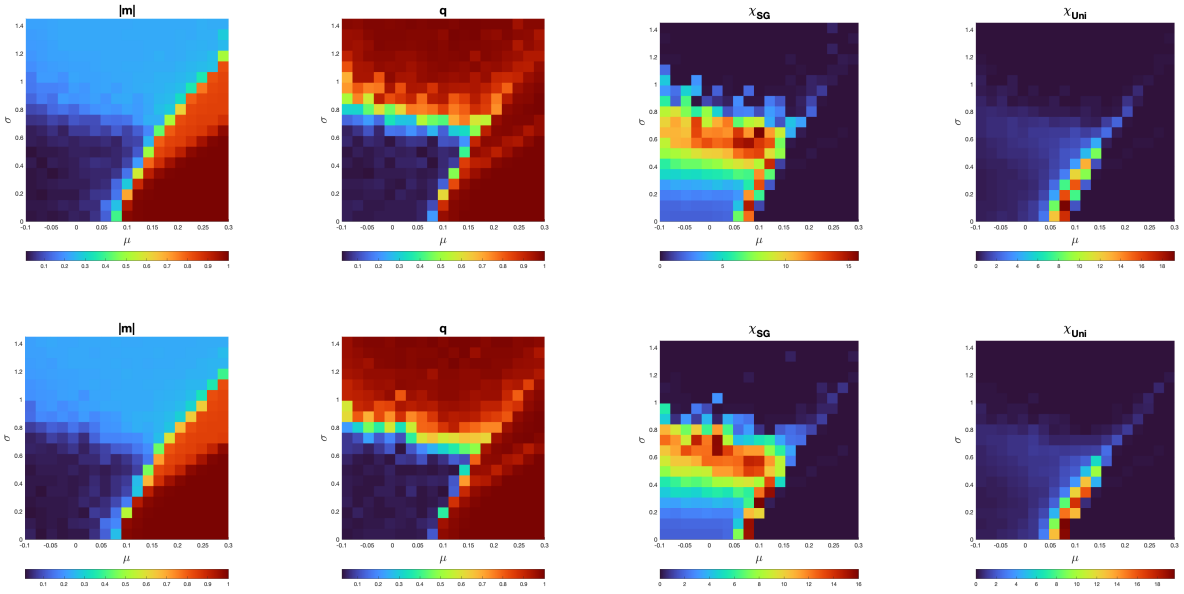


Figure B.9: Functional network 6.

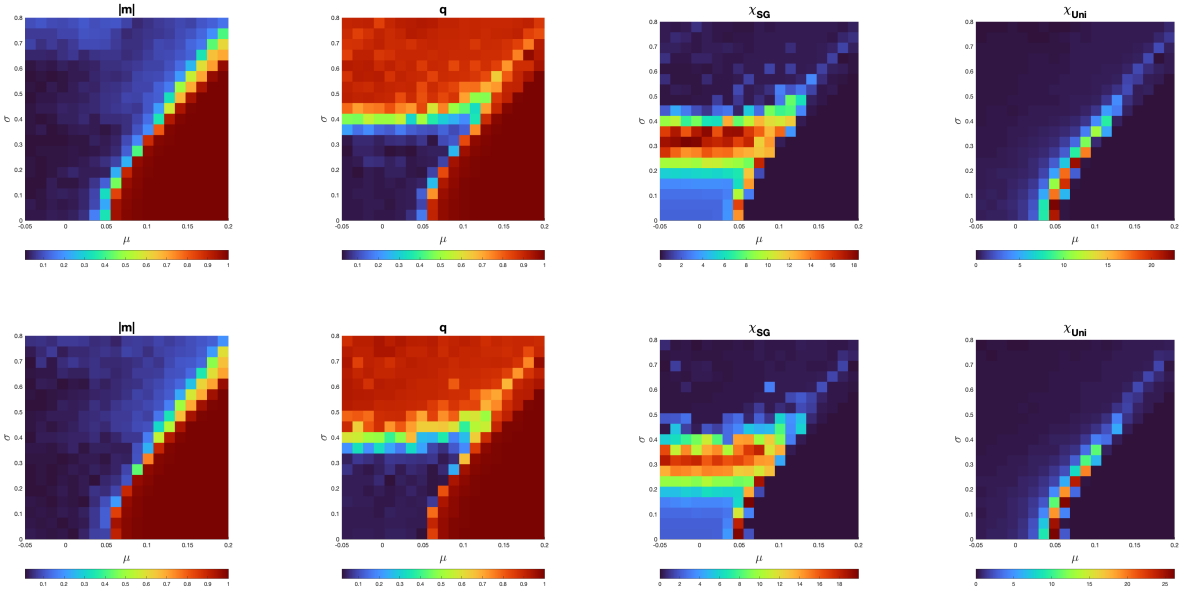


Figure B.10: Functional network 7.

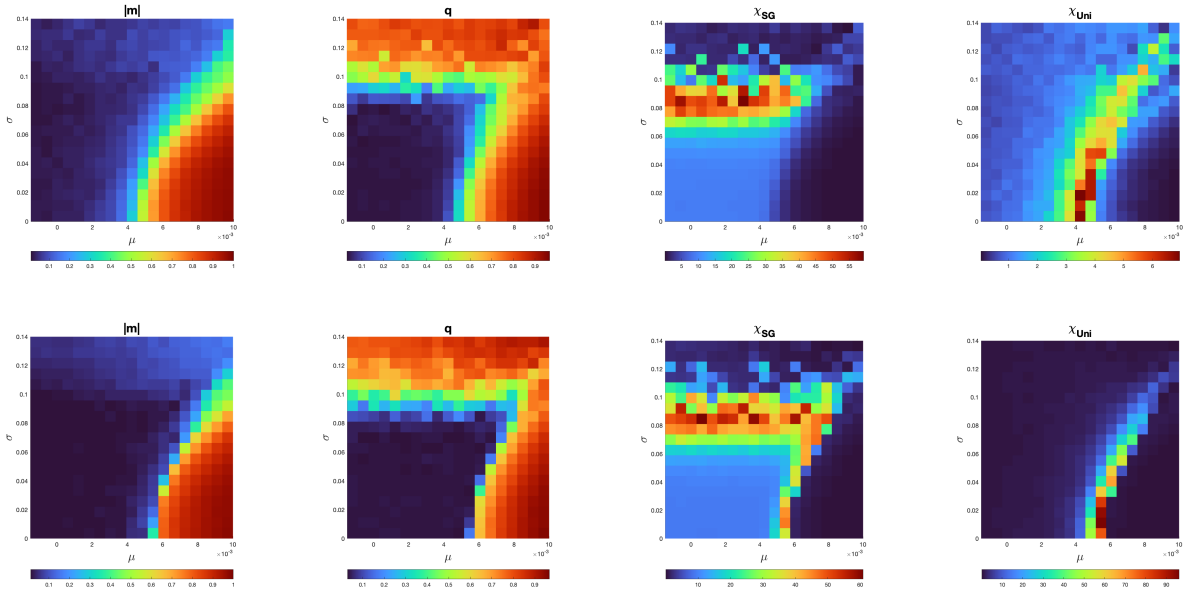
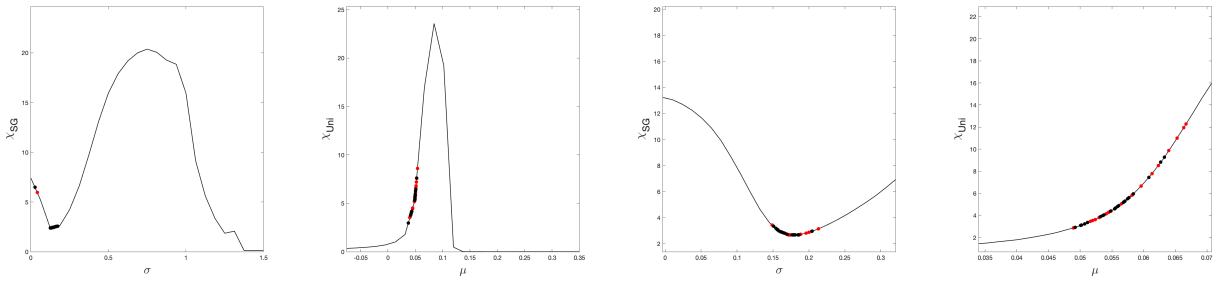
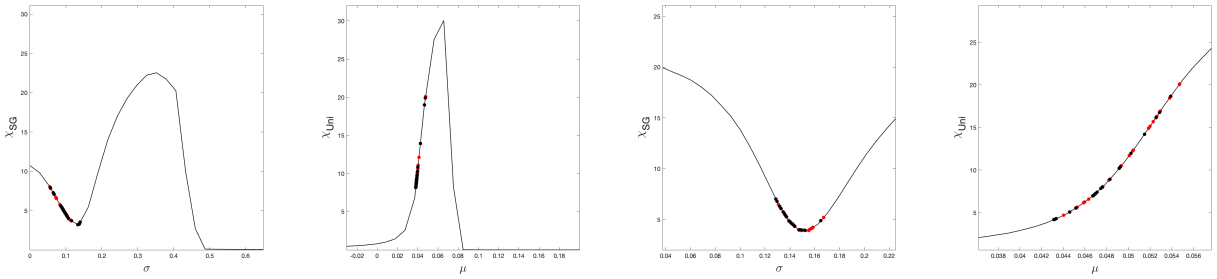


Figure B.11: Whole-brain network.

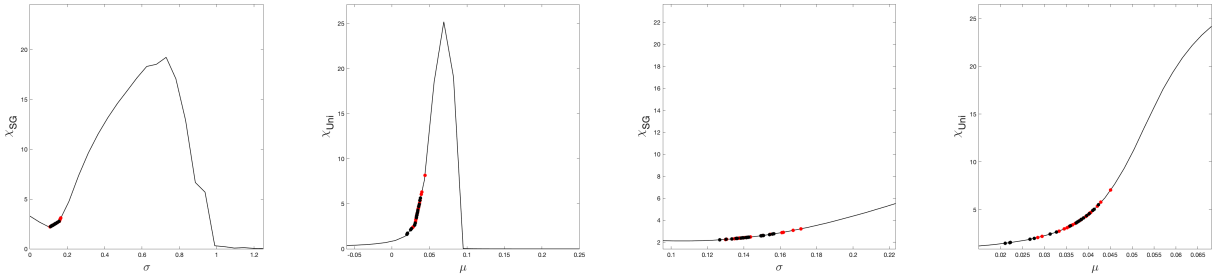
B.2 Cross-sections



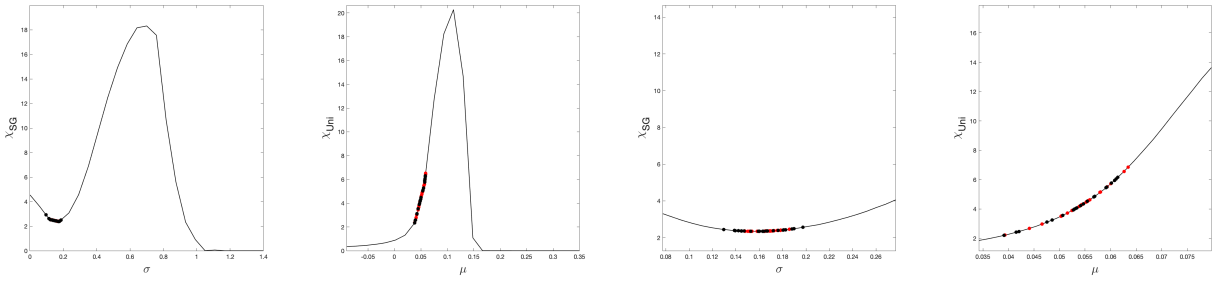
Functional network 1.



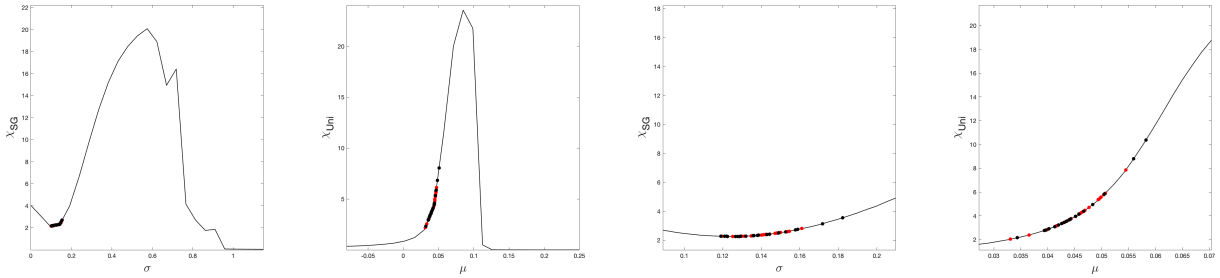
Functional network 2.



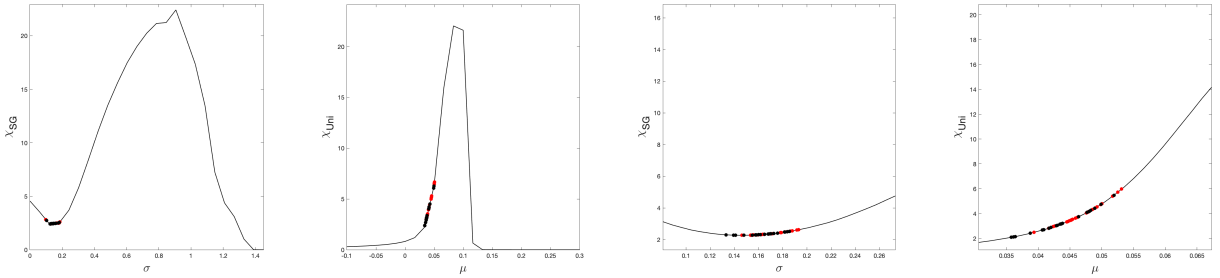
Functional network 3.



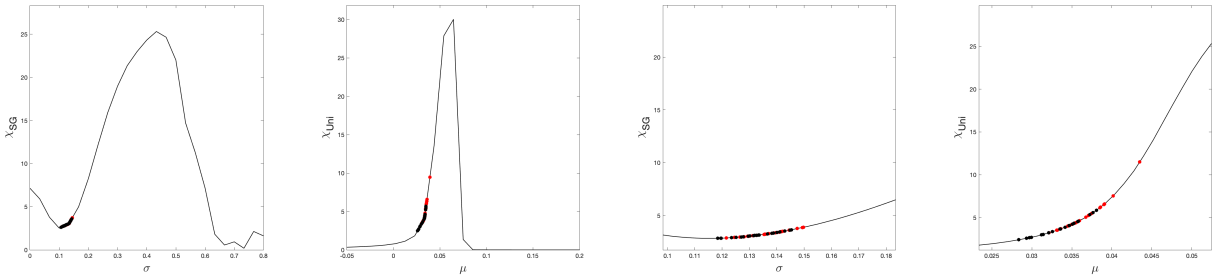
Functional network 4.



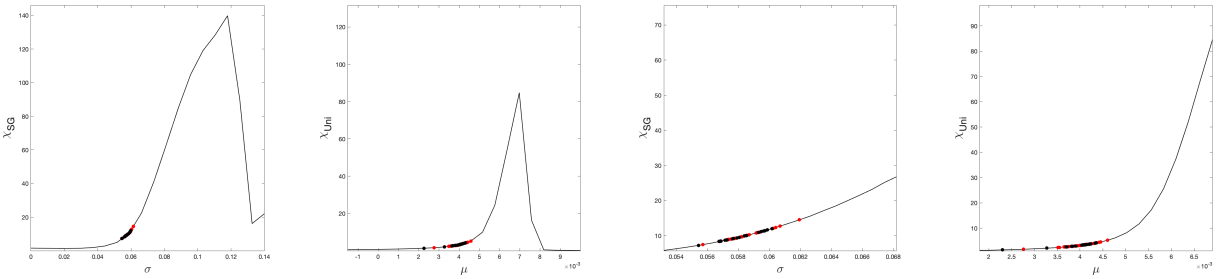
Functional network 5.



Functional network 6.



Functional network 7.



Whole-brain network.

Figure B.13: Cross-sections of the rough resp. refined phase diagrams of χ_{SG} resp. χ_{Uni} through $\mu = \hat{\mu}$ resp. $\sigma = \hat{\sigma}$ as described in Section 5.1.

B.3 Robustness of the method against variation in groups

To test the robustness of the method, i.e., that the estimation of the individual parameters μ, σ based on the phase diagrams corresponding to the archetype PMEM are robust enough for noise, we performed two additional experiments.

First, we divided all participants randomly into two groups. For both of the groups, we fitted the PMEM and compared the empirical and sampled spin correlations $\langle S_i S_j \rangle$ $(i, j = 1, \ldots, N, i \neq j)$ for each functional network in four ways:

a By comparing the spin correlations calculated for the empirical data from the first subgroup and that for the second subgroup;

b By comparing the spin correlations sampled from the archetype PMEM that we estimated for the first subgroup and that for the second subgroup;

c By comparing the spin correlations sampled from the archetype PMEM that we estimated for the first subgroup, shown on the horizontal axis, and the spin correlations directly calculated for the empirical data for the first subgroup (represented by the red dots) and the second subgroup (represented by the blue ones), shown on the vertical axis;

d By comparing the spin correlations sampled from the archetype PMEM that we estimated for the second subgroup, shown on the horizontal axis, and the spin correlation directly calculated for the empirical data for the first subgroup (represented by the blue dots) and the second subgroup (represented by the red ones), shown on the vertical axis.

These results are shown in the Figures B.14 till B.21. We observe that the covariance obtained from the two halves are strongly correlated, implying that the way of grouping the participants does not have a major affect on the inferred PMEM.

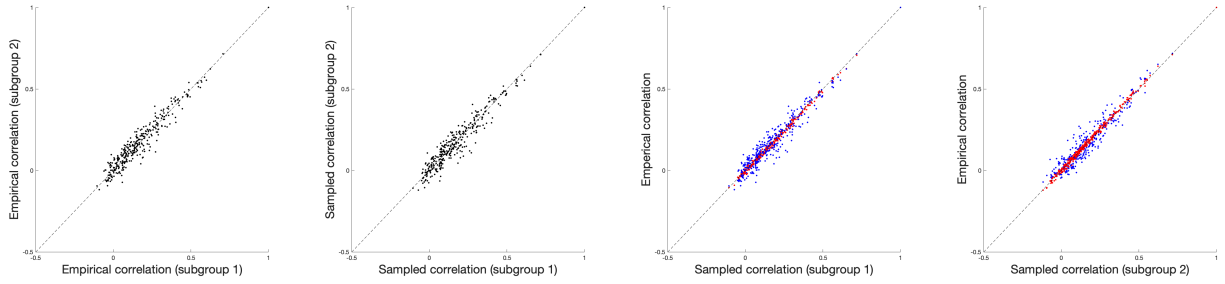


Figure B.14: Functional network 1.

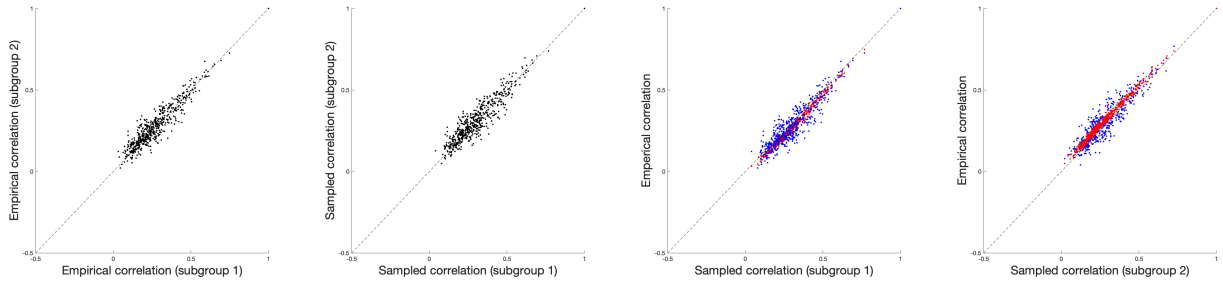


Figure B.15: Functional network 2.

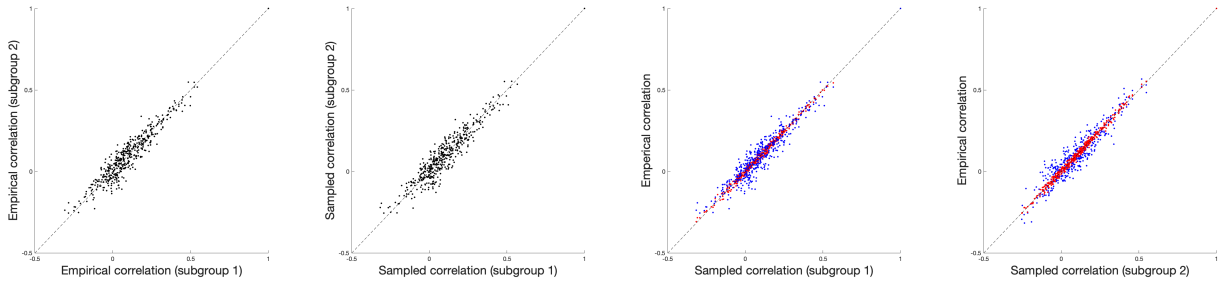


Figure B.16: Functional network 3.

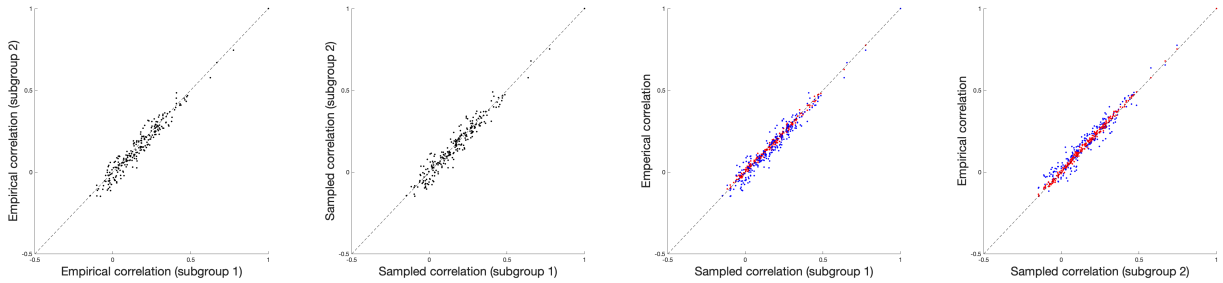


Figure B.17: Functional network 4.

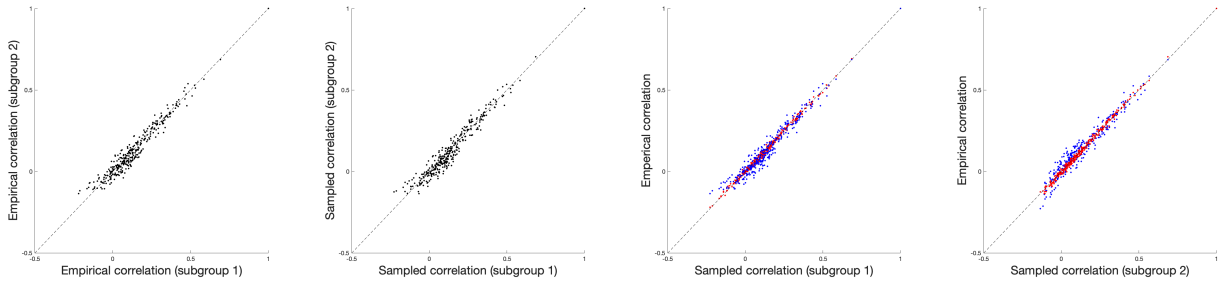


Figure B.18: Functional network 5.

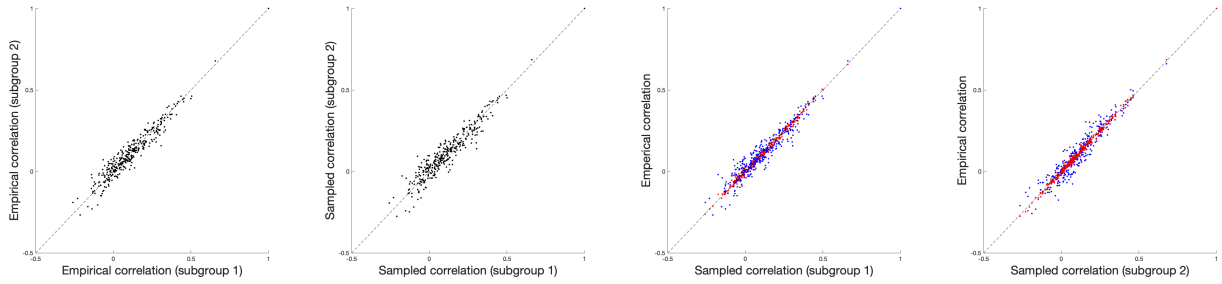


Figure B.19: Functional network 6.

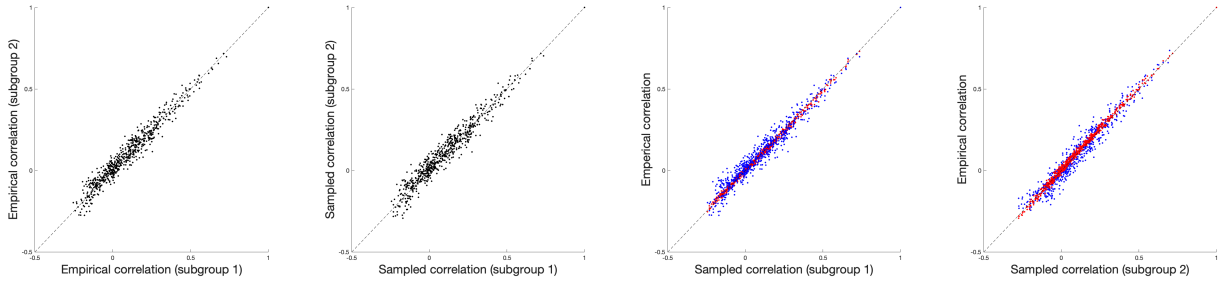


Figure B.20: Functional network 7.

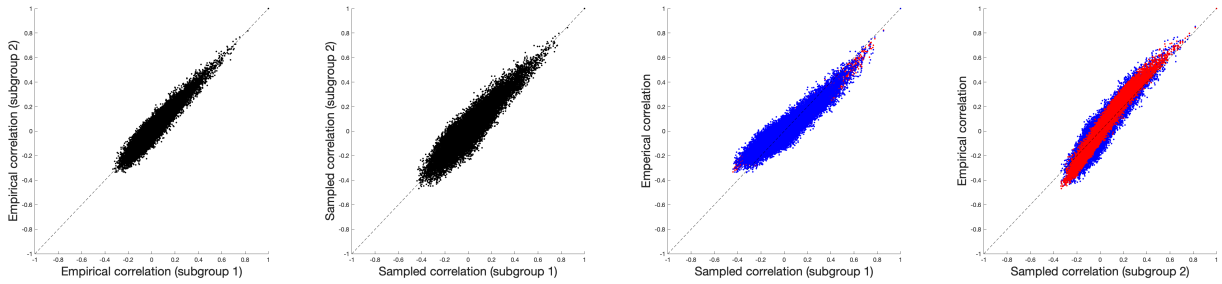


Figure B.21: Whole-brain network.

Additionally, we drew the phase diagrams corresponding to the inferred PMEMs for each of the subgroups. In the Figures B.22 till B.29, these resulting diagrams are shown for the functional networks and the whole-brain network. One can observe that these phase diagrams are very similar. In these phase diagrams we plotted for each group the mean and standard deviation of the inferred interaction matrix $\hat{\boldsymbol{J}}$ (denoted by a cross), and, in the same way, the mean and standard deviation of the interaction matrix inferred to the whole dataset (denoted by a dot). For most of the networks, these values corresponds very well.

These two experiments suggest that the inference of the PMEM based on the concatenated timeseries is robust against variations in the data, as the groups exhibited corresponding characteristics (in terms of observables calculated from the inferred archetype model).

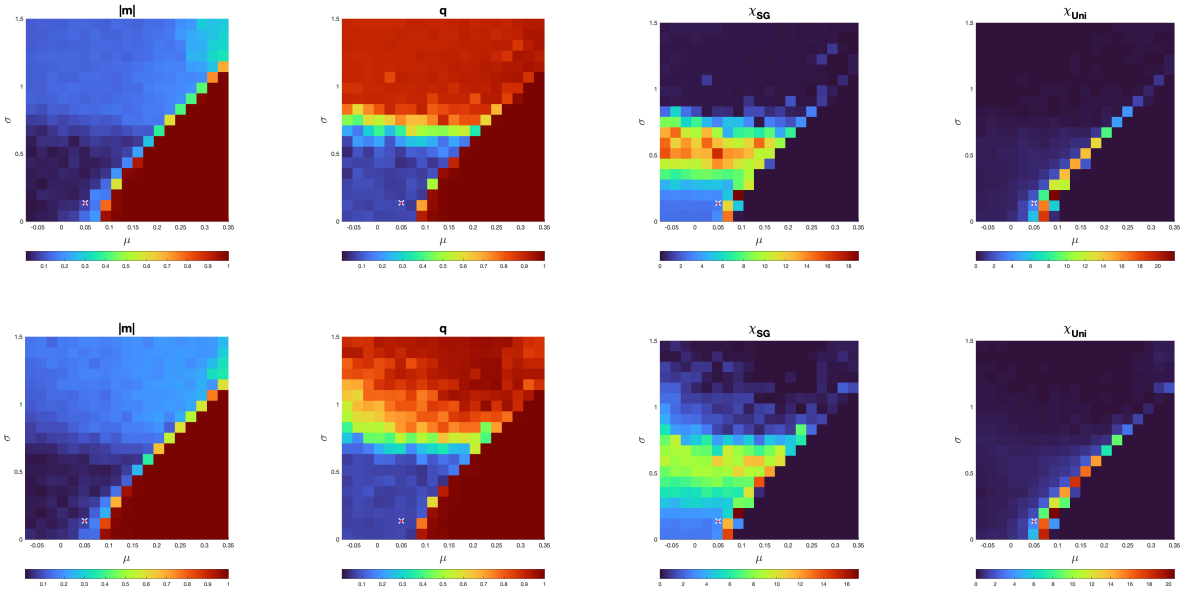


Figure B.22: Functional network 1.

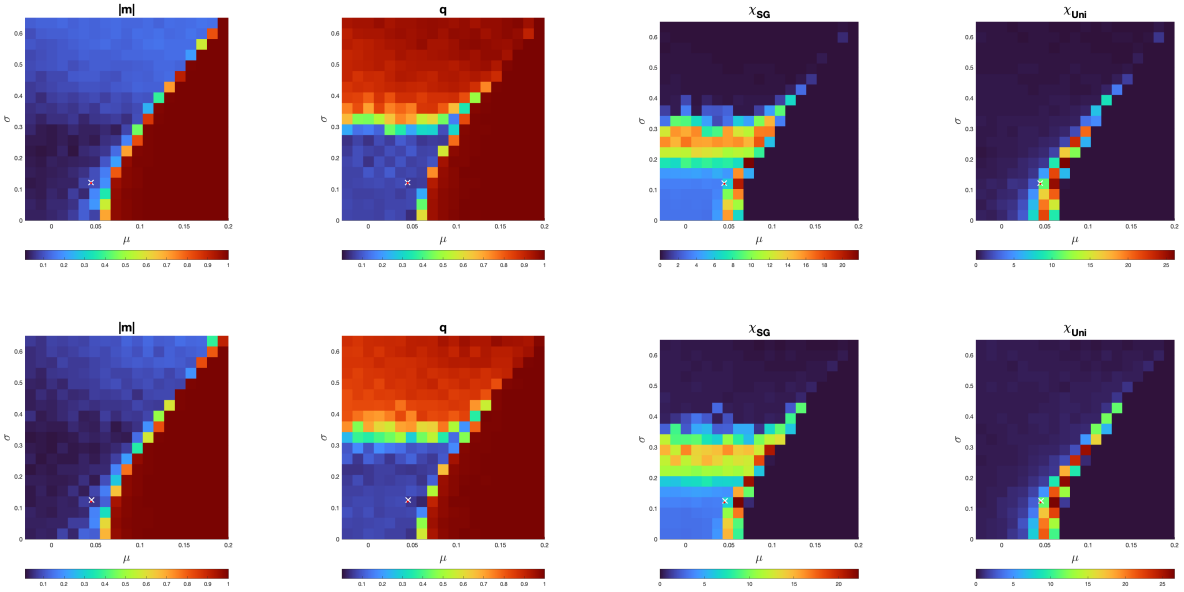


Figure B.23: Functional network 2.

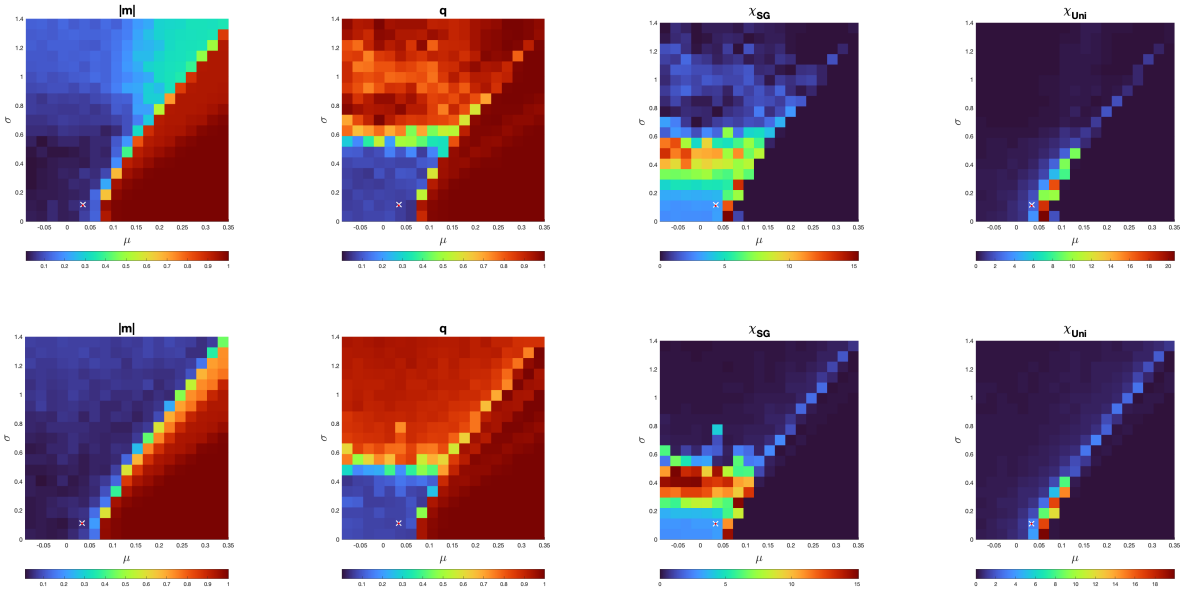


Figure B.24: Functional network 3.

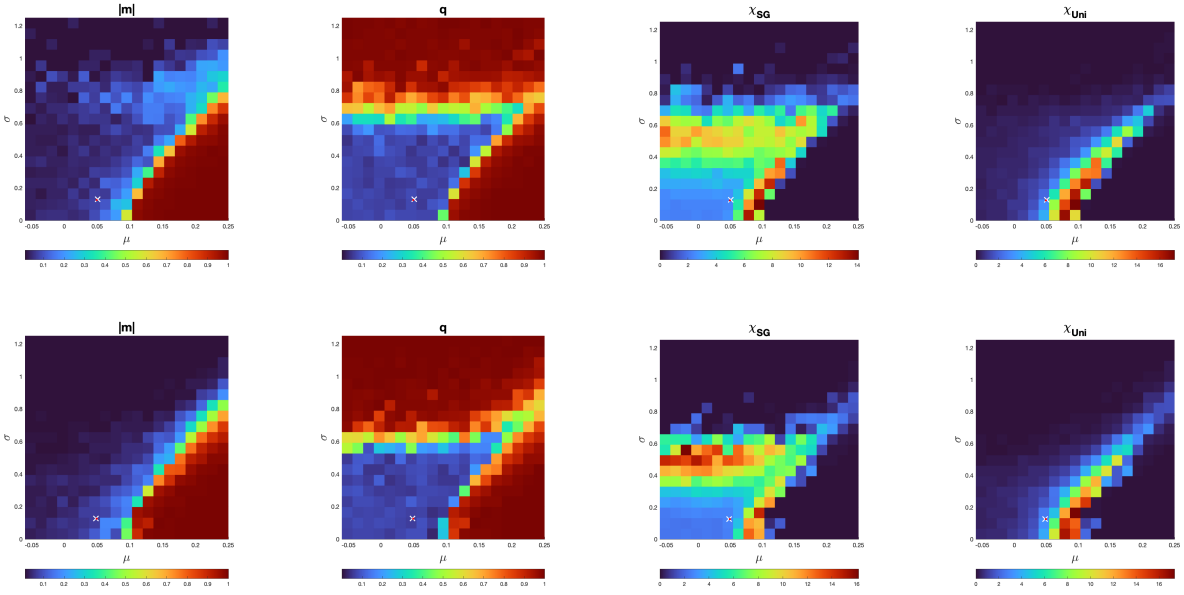


Figure B.25: Functional network 4.

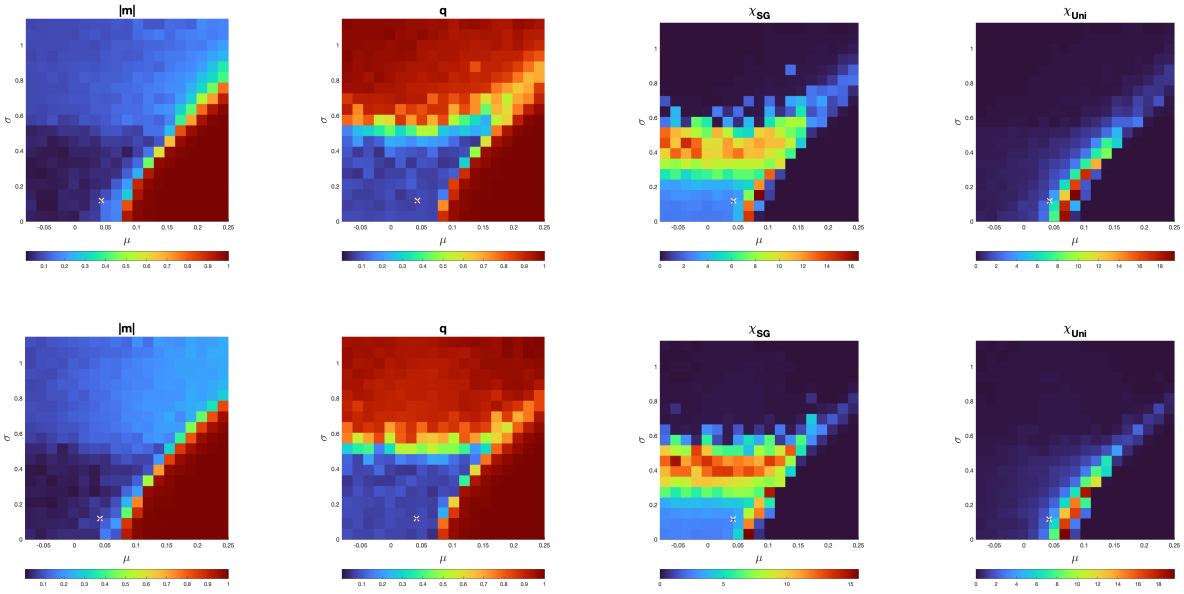


Figure B.26: Functional network 5.

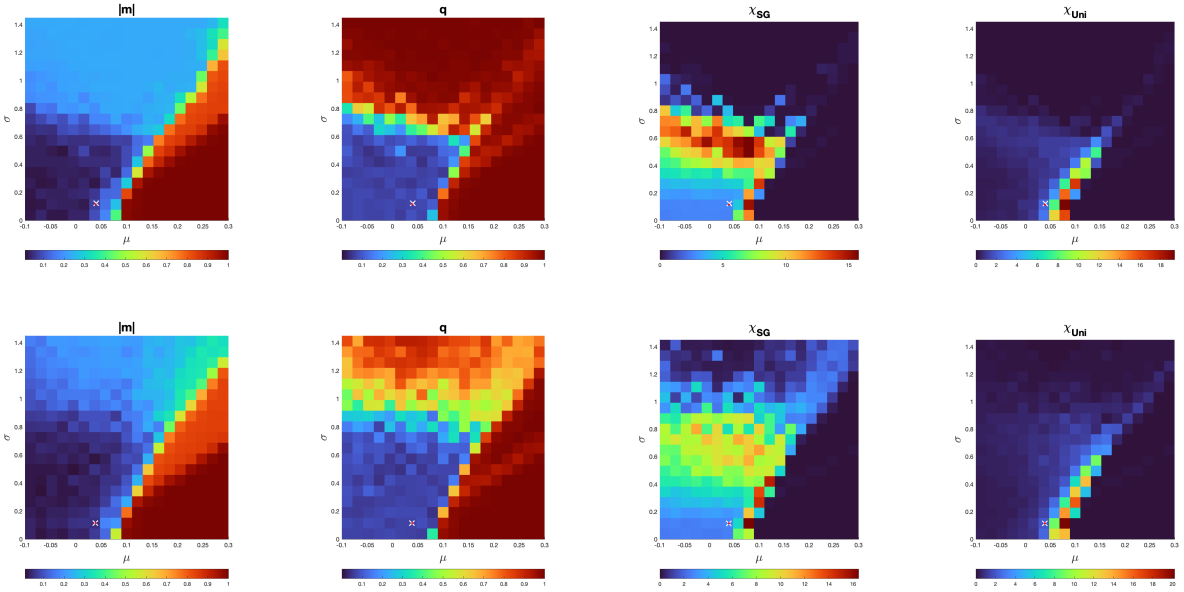


Figure B.27: Functional network 6.

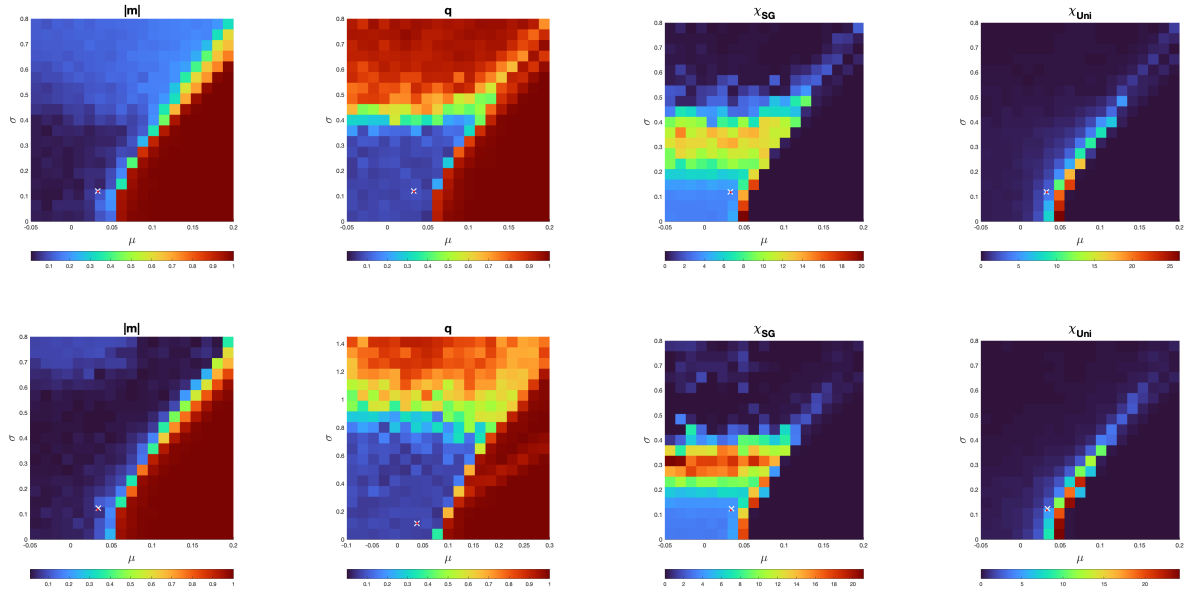


Figure B.28: Functional network 7.

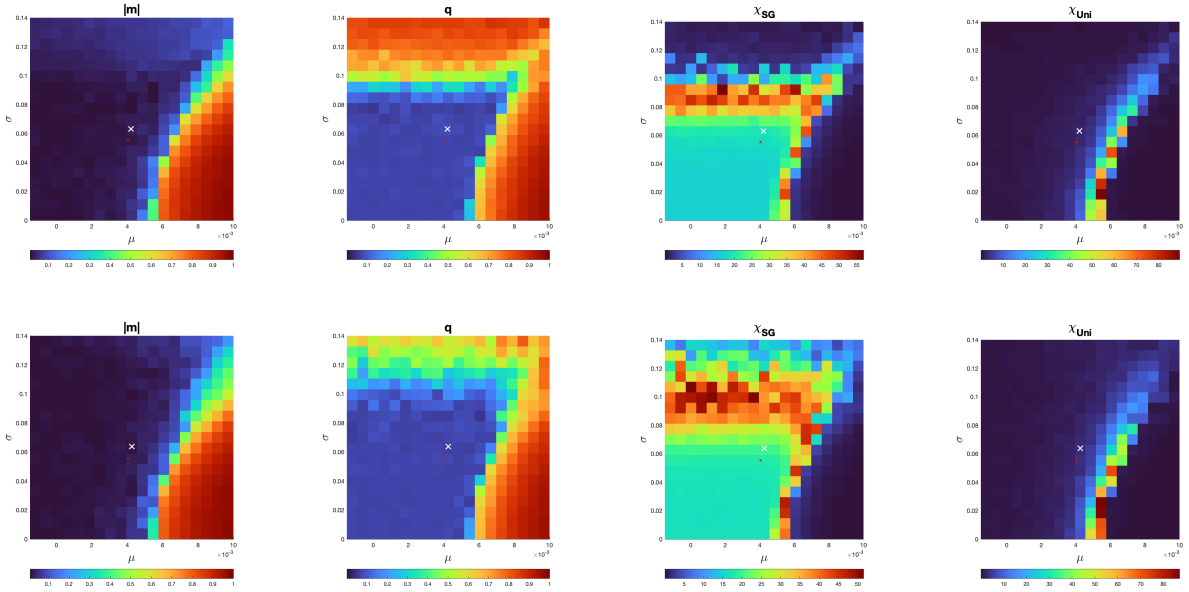


Figure B.29: Whole-brain network.

Appendix C

Statistical analysis PFE per ROI

As an exploratory analysis, we also consider the Permutation Fuzzy Entropy per ROI.

C.1 Methods

C.1.1 Calculating the PFE

The PFE per ROI was calculated as proposed for the other analyses (i.e, per functional network and for the whole-brain network). These details are discussed in Section 7.1.

C.1.2 Statistical analysis

We perform 238 ANOVA's (i.e., for each ROI) to compare the obtained results, using R version 4.3.0.

Beforehand, we identify outliers based on exceeding $1.5 \cdot \text{IQR}$ (InterQuartile Range) limits, and we remove these extreme outliers from the analyses. Subsequently, we conduct an analysis of variance (ANOVA), to examine whether the PFE is the result of the interaction between the ROI and treatment response with age and education (measured by ISCED) as covariate, motivated by similar studies [90–92]. When a covariate turns out to not be significant, we repeat the analysis without that particular covariate.

C.2 Results

C.2.1 Statistical analysis

Out of the $N = 238$ ROIs, for 39 ROIs the ANOVA's revealed a significant difference (i.e. $P < 0.05$). However, none did survive the correction for multiple comparisons using the False Discovery Rate ($0.0753 <= P <= 0.9694$).

Appendix D

Code

Most of the experiments were compiled and runned on a MacBook Pro with a 1,4 GHz Quad-Core Intel Core i5.

For the experiments, a combination of MATLAB and C++ was used. We will indicate for each section which programming language has been used. The code programmed in MATLAB was run using MATLAB R2022a, and the code programmed in C++ was compiled using Apple clang version 15.0.0.

All results can be reproduced by using the code provided in chapter.

Collect Data (MATLAB)

This section includes the functions that are used for extracting the data from the provided dataset in different ways and preparing it for further analysis by binarizing the neural signals.

The dataset consisted of 46 files, where file consists of the observed signals for the 238 Regions Of Interest (ROIs), acquired at 310 timepoints.

Additionally, it includes one experiment that collects and binarizes the data for each of the 7 functional networks and the whole-brain network. The other functions are used in other analyses.

Experiment: Collect Data Set

```
clear; close all;

%% Collect data for all networks
% Output: binarized time series for all networks.

participants = readmatrix('data/Participants/participants.dat');

%% Functional networks

for i = 1:7
    folder = sprintf('Data/Subnetworks/Subnetwork%d', i);
    labels = readmatrix(sprintf('Data/Subnetworks/Subnetwork%d/labels.dat', i));

    binarized_data = func_collectData(participants, labels);
```

```
    writematrix(binarized_data, sprintf('%s/input_data/data_set.dat', folder));
end

%% Whole-brain network
labels = 'Data/Whole-brain/labels.dat';

binarized_data = func_collectData(participants, labels);

writematrix(binarized_data, 'Whole-brain/input_data/data_set.dat');
```

Function: Collect Data of a Participant

```
%% Collect data for specified nodes of a participant
% Input: number of the participant (k), numbers of the signals (signal_nrs).
% Output: binarized signal, using the median of each signal as threshold.


function [data] = func_collectDataParticipant(participant_nr, signal_nrs)

    ROI_signals = func_readROISignal(participant_nr, signal_nrs, 1, 310);
    M = median(ROI_signals, 2);
    data = func_binarizeData(ROI_signals, M);

end
```

Function: Collect Data

```
%% Collect Data
% Collects data and binarizes data per patient, median as threshold
% Returns: matrix with all data; ROI's in rows, timepoints in columns.

function [data] =  func_collectData(participants, signal_nrs)
    data = [];

    for k = participants
        data_part = func_collectDataParticipant(k, signal_nrs);
        data = [data, data_part];
    end

end
```

Function: readROISignal

```
%% ReadROISignal
% Input: number patient k, numbers of the signals signal_nrs and time window
% t1 and t2 (i.e. timepoint 1 and 2).
% Return: data

function [ROISignal] = func_readROISignal(k, signal_nrs, t1, t2)

    if (k < 10)
        filename = sprintf('/.../ROISignals_00%d.txt', k);
    elseif (k < 100)
        filename = sprintf('/.../ROISignals_0%d.txt', k);
    else
        filename = sprintf('/.../ROISignals_%d.txt', k);
    end

    temp = readmatrix(filename)';
    ROISignal = temp(signal_nrs, t1:t2);
```

```
end
```

Function: Binarize Data

```
%% Binarize Data
% Input: dataset z_i(t), t \in {0,...,T}, i \in N (size: T \times N), threshold.
% Output: Binarized data S_i(t).
% Threshold: S_i(t) = +1 if z_i(t) \geq threshold, -1 otherwise.

function [binarized] = func_binarizeData(data, threshold)
    [numberROI, timePoints] = size(data);
    binarized = sign(data - (threshold+eps).*ones(numberROI, timePoints));
end
```

Infer Archetype PMEM (MATLAB)

Assuming that the datasets are collected and binarized, we can infer the archetype PMEM using the experiment provided in this section.

Experiment: Infer Pairwise Maximum Entropy Model

```
clear; close all;

addpath(genpath('/Users/myrthesterk/surfdrive/Shared/Criticality_and_entropy_BETER/Code/CollectData'));

%% Estimating PMEM for all networks
% input: binarized data
% output: model parameters h and J

%% Functional networks

for i = 1:7
    folder = sprintf('Data/Subnetworks/Subnetwork%d', i);

    binarized_data = readmatrix(sprintf('%s/input_data/data_set.dat', folder));

    [h, J] = func_inferParams(binarizedData, 1e-10);

    writematrix(h, sprintf('%s/PMEM/h.dat', folder));
    writematrix(J, sprintf('%s/PMEM/J.dat', folder));
end

%% Whole-brain network

folder = 'Data/Whole-brain';

binarized_data = readmatrix(sprintf('%s/input_data/data_set.dat', folder));

[h, J] = func_inferParams(binarizedData, 1e-10);

writematrix(h, sprintf('%s/PMEM/h.dat', folder));
writematrix(J, sprintf('%s/PMEM/J.dat', folder));
```

Function: Infer Model Parameters PMEM

```
%% Infer model params PMEM
% Using pseudo-likelihood maximization and gradient descent.
% Input: binarized data, permissible error
% Output: parameters h and J

function [h,J] = func_inferParams(binarized_data, permissible_error)
```

```
[nr_nodes, data_length] = size(binarized_data);
max_iterations = 5000000;
dt = 0.05; %learning rate

h = ones(nr_nodes,1);
J = zeros(nr_nodes);

data_mean = mean(binarized_data,2);
data_correlation = (binarized_data*binarized_data')/data_length;

for t=1:max_iterations
    % Calculate G_h
    G_h = - data_mean + mean(tanh(( J * binarized_data ...
        + h * ones(1, data_length))),2);

    % Calculate G_J; zorgt ervoor dat matrix symmetrisch wordt
    G_J = -data_correlation + ...
    0.5 * binarized_data * (tanh((J * binarized_data + ...
    h * ones(1, data_length))))'/data_length...
        + 0.5 * (binarized_data * (tanh((J * binarized_data + ...
        h * ones(1, data_length))))')'/data_length;
    G_J = G_J - diag(diag(G_J)); %eliminate diagonal entities (J_{ii}=0)

    % Update h and J
    h =  h - dt * G_h;
    J = J - dt * G_J;

    % Check stopping criterium
    diff = sqrt(norm(G_J)^2 + norm(G_h)^2)/nr_nodes/(nr_nodes+1);
    if (diff < permissible_error)
        break
    end

end
end
```

Temperature Analysis (MATLAB)

The temperature analysis was performed in MATLAB. In this section, the experiment that infers the personal temperature for each participant and each network (including the functional networks and whole-brain network) is provided, as well as the algorithm for inferring the personal temperature.

Experiment: Infer Personal Temperatures

```
%% Determine personal temperature
% Input: archetype PMEMs.
% Output: file with inferred personal temperatures.

% Make ordered list of responders and nonresponders
responders = readmatrix('/.../Data/Participants/responders.dat');
nonresponders = readmatrix('/.../Code/Data/Participants/nonresponders.dat');
participants = [responders nonresponders];

% Initialize results
results = zeros(length(participants), 8);

% Start with inferring beta per subnetwork
for n = 1:7
    % Read labels of ROI's belonging to the subnetwork, corresponding
    % parameters h and J.
```

```
    labels = readmatrix(sprintf('/.../Data/Subnetworks/Subnetwork%d/labels.dat', n));
    h = readmatrix(sprintf('/.../Data/Subnetworks/Subnetwork%d/PMEM/h.dat', n));
    J = readmatrix(sprintf('/.../Data/Subnetworks/Subnetwork%d/PMEM/J.dat', n));

    % For each participant, infer beta
    for i = 1:length(participants)
        i
        binarized_data = func_collectDataParticipant(participants(i), labels);

        beta = func_inferBeta(binarized_data, h, J, 1e-10);
        results(i, n) = 1/beta;
    end
end

% Same for the whole-brain network

% Read labels of ROI's belonging to the whole-brain network
labels = readmatrix('/.../Data/Whole-Brain/labels.dat');
h = readmatrix('/.../Data/Whole-Brain/Version_02/PMEM/h.dat');
J = readmatrix('/.../Data/Whole-Brain/Version_02/PMEM/J.dat');

% For each participant, infer beta
for j = 1:length(participants)
    binarized_data = func_collectDataParticipant(participants(j), labels);

    beta = func_inferBeta(binarized_data, h, J, 1e-10);
    results(j, 8) = 1/beta;
end

% Write results to output.csv, including stats participants
results = [participants', results];
table = array2table(results, 'VariableNames', ["participant_nr","T_sub1","T_sub2", ...
    "T_sub3","T_sub4","T_sub5","T_sub6","T_sub7","T_wb"]);
stats = readtable('/.../Code/Data/stats_participants.csv');
table = [table stats(:,2:end)];

writetable(table, 'output.csv');
```

Function: Infer β

```
%% Calculate Individual Tempertature
% Using the archetype PLM and gradient descent. Note: T = 1/beta.
% Input: individual binarized data, h and J obtained by PLM
% Output: for each subgroup the personal temperature.

function beta = func_inferBeta(binarized_data, h, J, permissible_error)
    [nr_nodes, data_length] = size(binarized_data);
    max_iterations = 5000000;
    dt = 0.01; %learning rate
    beta = 0;

    data_mean = mean(binarized_data,2);
    data_correlation = (binarized_data*binarized_data')/data_length;

    for t = 1:max_iterations

        mean_P = mean(tanh(beta*(J * binarized_data + ...
            h * ones(1, data_length))),2);
        corr_P = 0.5 * binarized_data *      (tanh(beta * (J * binarized_data + ...
            h * ones(1, data_length))))'/data_length...
                + 0.5 * (binarized_data * (tanh(beta*(J * binarized_data + ...
                    h * ones(1, data_length))))')'/data_length;
```

```
        %calculate gradient of beta
        H_emp = - h' * data_mean - trace(data_correlation' * J);
        H_P = -h' * mean_P - trace(corr_P' * J);
        G_beta = H_emp - H_P;

        %Update
        beta = beta - dt * G_beta;

        diff = abs(G_beta); %Andere stopping criteria bepalen?
        if (diff < permissible_error)
            break
        end
    end
end
```

Draw Phase Diagrams (C++)

The phase diagram analysis was performed in both C++ and MATLAB. Using the inferred archetype PMEM, we drew the phase diagrams using C++. These phase diagrams were analyzed (i.e., calculating the position of the participants) and visualized using MATLAB.

The implementation in C++ consists of three directories. In this section, we maintained this structure by using a subsection for each directory.

Src

InverseIsingProblem.cpp

```
#include "include/InverseIsingProblem.h"
#include "include/util.h"

#include <Eigen/Eigenvalues>

using namespace Eigen;
using namespace std;

InverseIsingProblem::InverseIsingProblem(const string& folder, bool corrected) :
    folder(folder), T_(1.)
{
    data_ = openData(folder + "/input_data/data_set.dat");

    // If fictitious temperature is given & corrected, adjust h and J.
    // Corrected == true => fictitious temperature is incorporated.

    if (file_exists(folder + "/PMEM/T_f.dat" && corrected)) {
        VectorXd temp = openData(folder + "/PMEM/T_f.dat");
        T_ *= temp(0);
    }

        // Read h, terminate if not available.
    if (file_exists(folder + "/PMEM/h.dat"))
        h_ = openData(folder + "/PMEM/h.dat")/T_;
    else {
        cout << "h not available!";
        exit(0);
        }

    // Read J, terminate if not available.
    if (file_exists(folder + "/PMEM/J.dat"))
        J_ = openData(folder + "/PMEM/J.dat")/T_;
```

```
    else {
        cout << "h not available!";
        exit(0);
    }
}


// Save h and J
void InverseIsingProblem::save() {
    if (nr_ == 0) {
        saveData(folder + "/PMEM/h.dat", h_);
        saveData(folder + "/PMEM/J.dat", J_);
    }
    else {
        saveData(folder + "/PMEM/h" + to_string(nr_) + ".dat", h_);
        saveData(folder + "/PMEM/J" + to_string(nr_) + ".dat", J_);
    }
}


// Update h and J
void InverseIsingProblem::update(VectorXd &h, MatrixXd &J) {
    h_ = h;
    J_ = J;
}


// Return h and J
pair<VectorXd, MatrixXd> InverseIsingProblem::return_params() {
    pair<VectorXd, MatrixXd> temp;
    temp.first = h_;
    temp.second = J_;
    return temp;
}


// Return T.
double InverseIsingProblem::return_T() {
    return T_;
}


// Return dataset.
MatrixXd InverseIsingProblem::return_data() {
    return data_;
}
```

Include

InverseIsingProblem.h

```
#pragma once

#include <Eigen/Eigenvalues>

using namespace Eigen;
using namespace std;

class InverseIsingProblem {
public:
    InverseIsingProblem(const string& folder, bool corrected = false);
    string folder;
```

```
    void update(VectorXd &h, MatrixXd &J);
    void save();
    pair<VectorXd, MatrixXd> return_params();
    double return_T();
    MatrixXd return_data();

protected:
    MatrixXd data_;
    VectorXd h_;
    MatrixXd J_;
    double T_;
};
```

util.cpp

```
#pragma once

#include <Eigen/Eigenvalues>
#include <Eigen/Dense>

#include <iostream>
#include <cstdio>
#include <vector>
#include <fstream>
#include <string>
#include <cmath>
#include <random>
#include <thread>
#include <chrono>

using namespace Eigen;
using namespace std;


// Read data from .dat file
// Adapted from:
// https://aleksandarhaber.com/eigen-matrix-library-c-tutorial-saving-and-loading-data-in-from-a-csv-file/
static MatrixXd openData(const string& datafile_to_open)
{
    vector<double> matrix_entries;
    ifstream matrix_data_file(datafile_to_open);
    string matrix_row_string;
    string matrix_entry;
    int matrix_row_number = 0;

    while (getline(matrix_data_file, matrix_row_string))
    {
        stringstream matrix_row_string_stream(matrix_row_string);

        while (getline(matrix_row_string_stream, matrix_entry, ','))
        {
            matrix_entries.push_back(stod(matrix_entry));
        }
        matrix_row_number++;

    Eigen::MatrixXd temp = Eigen::Map<Eigen::MatrixXd>(matrix_entries.data(),
                                (Eigen::Index) matrix_entries.size() / matrix_row_number, matrix_row_number);
    return temp.transpose();
}
```

```
// Write data to .dat file
// Adatpted from:
// https://aleksandarhaber.com/eigen-matrix-library-c-tutorial-saving-and-loading-data-in-from-a-csv-file/
static void saveData(const string& datafile_name, const MatrixXd&  matrix)
{
    const static IOFormat CSV_format(FullPrecision, DontAlignCols, ",", "\n");

    ofstream file(datafile_name);
    if (file.is_open())
    {
        file << matrix.format(CSV_format);
        file.close();
    }
}


// Calculate mean over off-diagonal elements J
static double matrixMean(MatrixXd &matrix) {
    double sum = 0;

    for (size_t i = 0; i < matrix.rows(); i++) {
        for (size_t j = 0; j < matrix.cols(); j++) {
            if (i != j)
                sum += matrix(i,j);
        }
    }

    return sum/((double) matrix.rows()*matrix.cols() - matrix.rows());
}


// Calculate standard deviation over off-diagonal elements J
static double matrixSTD(MatrixXd &matrix) {
    double mean = matrixMean(matrix);
    double sum = 0;

    for (size_t i = 0; i < matrix.rows(); i++) {
        for (size_t j = 0; j < matrix.cols(); j++) {
            if(i != j) {
                sum += (matrix(i,j) - mean)*(matrix(i,j) - mean);
            }
        }
    }

    return sqrt(sum/((double) matrix.rows()*matrix.cols() - matrix.rows() - 1));
}
```

PhaseDiagram.h

```
#pragma once

#include <Eigen/Eigenvalues>
#include <chrono>

#include "include/InverseIsingProblem.h"
#include "include/Simulate.h"
#include "include/util.h"

using namespace Eigen;
using namespace std;
```

```
/* Affine transfrom for phase diagram */
static MatrixXd affine_transform(MatrixXd &matrix, double mu, double sigma, double mean, double stdev) {
    MatrixXd matrix_tilde(matrix.rows(), matrix.cols());

    for (size_t i = 0; i < matrix.rows(); i++) {
        for (size_t j = 0; j < matrix.cols(); j++) {
            matrixTilde(i,j) = (matrix(i,j) - mean) * sigma/stdev + mu;
        }
    }

    return matrix_tilde;
}


/* Determine boundaries for refined phase diagram */
MatrixXd refinedInterval(string &folder) {
    MatrixXd stats_participants = openData(folder + "/stats_participants.dat");

    double sigma_min =  stats_participants.col(1).minCoeff() -
            (stats_participants.col(1).maxCoeff() - stats_participants.col(1).minCoeff())/5;
    double sigma_max = stats_participants.col(1).maxCoeff() +
            (stats_participants.col(1).maxCoeff() - stats_participants.col(1).minCoeff())/1;

    double mu_min = stats_participants.col(2).minCoeff() -
            (stats_participants.col(2).maxCoeff() - stats_participants.col(2).minCoeff())/5;
    double mu_max = stats_participants.col(2).maxCoeff() +
            (stats_participants.col(2).maxCoeff() - stats_participants.col(2).minCoeff())/1;

    Eigen::MatrixXd new_intervals(2,2);
    new_intervals << sigma_min, sigma_max,
            mu_min, mu_max;

    return new_intervals;
}


/* Calculate phase diagram */
// Input: Inverse Ising problem (consisting of data, inferred h, inferred J), folder to save the diagram
// t_max: thermalization time
// nr_samples: nr of samples that are generated

void computePD(InverseIsingProblem &problem, string &folder_pd, VectorXd range_sigma,
               VectorXd range_mu, long t_max, long nr_samples) {

    VectorXd h = problem.return_params().first;
    MatrixXd J = problem.return_params().second;

    // Nr experiments for averaging
    double nr_experiments = 10;

    long res_mu = range_mu.size();
    long res_sigma = range_sigma.size();

    /* Mean and standard deviation of J for affine transformation */
    double mean = matrixMean(J);
    double stdev = matrixSTD(J);

    /* Initialize results */
    MatrixXd mResults = MatrixXd::Constant(res_mu, res_sigma, 0);
    MatrixXd qResults = MatrixXd::Constant(res_mu, res_sigma, 0);
    MatrixXd chiSGResults = MatrixXd::Constant(res_mu, res_sigma, 0);
    MatrixXd chiUniResults = MatrixXd::Constant(res_mu, res_sigma, 0);
```

```
    MatrixXd CResults = MatrixXd::Constant(res_mu, res_sigma, 0);

    long count = res_mu * res_sigma;
    size_t i, j;


    /* Simulation of the model */
#pragma omp parallel for collapse(2) schedule(dynamic) private(i, j)
    for (i = 0; i < res_mu; i++) {

        for (j = 0; j < res_sigma; j++) {

            MatrixXd JTilde = affine_transform(J, range_mu(i), range_sigma(j), mean, stdev);
            VectorXd observables = VectorXd::Constant(5,0);

            for (int k = 0; k < nr_experiments; k++) {
                observables += sampleObservables(problem.return_params().first, JTilde, t_max, nr_samples);
            }

            observables /= nr_experiments;

            /* Store results */
            mResults(i, j) = observables(0);
            qResults(i, j) = observables(1);
            chiUniResults(i, j) = observables(2);
            chiSGResults(i, j) = observables(3);
            CResults(i, j) = observables(4);
        }
    }

    /* Save results */
    if (problem.return_nr() == 0) {
        saveData(folder_pd + "/pd_m.dat", mResults);
        saveData(folder_pd + "/pd_q.dat", qResults);
        saveData(folder_pd + "/pd_chiUni.dat", chiUniResults);
        saveData(folder_pd + "/pd_chiSG.dat", chiSGResults);
        saveData(folder_pd + "/pd_C.dat", CResults);

        saveData(folder_pd + "/mu_list.dat", range_mu);
        saveData(folder_pd + "/sigma_list.dat", range_sigma);
    } else {
        saveData(folder_pd + "/pd_m" + to_string(problem.return_nr()) + ".dat", mResults);
        saveData(folder_pd + "/pd_q" + to_string(problem.return_nr()) + ".dat", qResults);
        saveData(folder_pd + "/pd_chiUni" + to_string(problem.return_nr()) + ".dat", chiUniResults);
        saveData(folder_pd + "/pd_chiSG" + to_string(problem.return_nr()) + ".dat", chiSGResults);
        saveData(folder_pd + "/pd_C" + to_string(problem.return_nr()) + ".dat", CResults);

        saveData(folder_pd + "/mu_list" + to_string(problem.return_nr()) + ".dat", range_mu);
        saveData(folder_pd + "/sigma_list" + to_string(problem.return_nr()) + ".dat", range_sigma);
    }
}
```

Simulate.h

```
#pragma once

#include </usr/local/include/eigen3/Eigen/Eigenvalues>
#include <cstdio>
#include <utility>
#include <vector>
#include <iostream>
#include <random>
```

```
#include "include/InverseIsingProblem.h"
#include "include/util.h"

struct sample {
    double m = 0;
    double q = 0;
    double C = 0;
    VectorXd local_magnetization;
    MatrixXd correlation;
};

/* Generate samples from the inferred parameters */
// t_max: thermalization time
// nr_samples: nr of samples that are generated

static struct sample simulate(VectorXd h, MatrixXd J, long t_max, long nr_samples) {
    int nr_nodes = (int) h.size();
    long n;
    double E = 0;
    double E2 = 0;
    double dE, p;

    /* Initialize random number generator */
    // Initialized per experiment for properly generated random samples when running in parallel.

    std::random_device rd;
    std::default_random_engine engine(rd());
    std::uniform_real_distribution<double> randomReal(0, 1);

    VectorXd local_magnetization = VectorXd::Constant(nr_nodes, 0);
    MatrixXd correlation = MatrixXd::Constant(nr_nodes, nr_nodes, 0);

    //VectorXd sigma = VectorXd::Constant(nr_nodes, 1);
    VectorXd sigma(nr_nodes);
    std::generate(sigma.begin(), sigma.end(), [&]()
        {double p = randomReal(engine); int s = 1; if (p <= 0.5) {s *= 1;} return s;});

    // Thermalization process
    for (long t = 0; t < t_max * nr_nodes; t++) {
        n = (long) t % nr_nodes;

        dE = -2. * h(n) * sigma(n) - 2 * J.col(n).dot(sigma) * sigma(n);
        p = min<float>(1, (float) exp(dE));

        if(randomReal(engine) <= p)
            sigma(n) = sigma(n) * (-1);
    }

    // Sampling process (sample every 1000 sweeps)
    for (long t = 0; t < nr_samples * nr_nodes * 1000; t++) {
        n = (long) t % nr_nodes;
        dE = -2 * h(n) * sigma(n) - 2 * J.col(n).dot(sigma) * sigma(n);
        p = min<float>(1, (float) exp(dE));

        if(randomReal(engine) <= p)
            sigma(n) = sigma(n) * (-1);

        if (t % (nr_nodes * 1000) == 0) {
            local_magnetization = local_magnetization + sigma;
            correlation = correlation + sigma * sigma.transpose();
```

```
            double energy = -(0.5 * J * sigma).transpose() * sigma - h.dot(sigma);
            E = E - energy;
            E2 = E2 + energy * energy;
        }
    }

    // Store observables

    struct sample temp;
    local_magnetization = local_magnetization / (double) nr_samples;

    E2 = E2 / (double) nr_samples;
    E = E / (double) nr_samples;
    temp.C = (E2 - E * E) / nr_nodes;
    temp.q = local_magnetization.dot(local_magnetization) / nr_nodes;
    temp.m = local_magnetization.sum() / nr_nodes;

    temp.local_magnetization = local_magnetization;
    temp.correlation = correlation / nr_samples;

    return temp;
}

/* Calculate and return magnetization and correlation */
static pair<VectorXd, MatrixXd> sampleStats(VectorXd h, MatrixXd J, long t_max, long nr_samples) {
    struct sample temp = simulate(std::move(h), std::move(J), t_max, nr_samples);

    pair<VectorXd, MatrixXd> temp_pair;
    temp_pair.first = temp.local_magnetization;
    temp_pair.second = temp.correlation;

    return temp_pair;
}


/* Calculate and return magnetization, spin-glass order parameter, uniform and spin-glass susceptibility */
static VectorXd sampleObservables(VectorXd h, MatrixXd J, long t_max, long nr_samples) {
    struct sample temp = simulate(std::move(h), std::move(J), t_max, nr_samples);

    MatrixXd covariance = temp.correlation - temp.local_magnetization * temp.local_magnetization.transpose();

    double chi_Uni = covariance.sum() / (double) covariance.cols();
    double chi_SG = covariance.squaredNorm() / (double) covariance.cols();

    VectorXd vec = VectorXd::Constant(5,0);
    vec[0] = abs(temp.m);
    vec[1] = temp.q;
    vec[2] = chi_Uni;
    vec[3] = chi_SG;
    vec[4] = temp.C;

    return std::move(vec);
}

/* Calculate and return C^2 */
// Used for C2 correction
static double sampleC2(const VectorXd& h, MatrixXd J, long t_max, long nr_samples) {
    struct sample temp = simulate(h, std::move(J), t_max, nr_samples);
    MatrixXd covariance = temp.correlation - temp.local_magnetization * temp.local_magnetization.transpose();

    return covariance.squaredNorm()/(double) h.size();
}
```

CorrectC2.h

```
#include <Eigen/Eigenvalues>

#include "include/util.h"
#include "include/Simulate.h"

using namespace std;
using namespace Eigen;

/* Compute objective for C2 correction for fictitious temperature T */
double compute_objective(InverseIsingProblem &problem, double T,
                         double target, int nr_experiments, long t_max) {
    VectorXd h = problem.return_params().first / T;
    MatrixXd J = problem.return_params().second / T;

    VectorXd objectives = VectorXd::Constant(nr_experiments, 0);

    double mean_objective = 0, sampled_C2 = 0;
    int j;

#pragma omp parallel for private(j, sampled_C2) shared(h, J) reduction(+: mean_objective)
    for(j = 0; j < nr_experiments; j++) {
        sampled_C2 = sampleC2(h, J, t_max, 310*46);
        mean_objective += (target - sampled_C2) * (target - sampled_C2);
        cout << T << " & " << sampled_C2 << endl;
    }

    return mean_objective / nr_experiments;
}

/* Compute target C2 */
double compute_target(InverseIsingProblem &problem) {
    VectorXd data_mean = problem.return_data().rowwise().mean();
    MatrixXd data_correlation = (problem.return_data()*problem.return_data().transpose())
            /problem.return_data().cols();

    MatrixXd covariance(problem.return_params().first.size(), problem.return_params().first.size());
    covariance = data_correlation - data_mean * data_mean.transpose();

    return covariance.squaredNorm()/(double) problem.return_params().first.rows();
}

/* C2 Correction using Fibonacci search */
// Input: Inverse Ising problem
// Outline Fibonacci search algorithm: http://homepages.math.uic.edu/~jan/mcs471/goldensection.pdf

double correctC2(InverseIsingProblem &problem) {
    double target = compute_target(problem);

    double a = 0.9;
    double b = 1.1;
    double tol = 1e-5; //tolerated relative error

    double golden_ratio = (sqrtf(5) - 1)/2;

    double left = golden_ratio * a + (1 - golden_ratio) * b;
    double obj_left = compute_objective(problem, left, target, 10, 1e7);

    double right = (1 - golden_ratio) * a + golden_ratio * b;
    double obj_right = compute_objective(problem, right, target, 10, 1e7);
```

```
    while (abs(b - a) > tol) {
        if (obj_left < obj_right) {
            b = right;
            right = left;
            obj_right = obj_left;
            left = golden_ratio * a + (1 - golden_ratio) * b;
            obj_left = compute_objective(problem, left, target, 10, 1e7);
        }
        else {
            a = left;
            left = right;
            obj_left = obj_right;
            right = (1 - golden_ratio) * a + golden_ratio * b;
            obj_right = compute_objective(problem, right, target, 10, 1e7);
        }
    }

    return T = (b - a)/2;
}
```

App

Experiment: Draw Phase Diagrams

```
#include "include/InverseIsingProblem.h"
#include "include/PhaseDiagram.h"
#include "include/CorrectC2.h"
#include "include/util.h"

#include <Eigen/Eigenvalues>

using namespace Eigen;
using namespace std;

void performC2Correction(string &folder) {
    InverseIsingProblem problem = InverseIsingProblem(folder);
    T = correctC2(problem);
    saveData(T, folder + "/T_f.dat");
};

int main() {

    string folder = "Data/Subnetworks/Subnetwork1";
    performC2Correction(folder);

    string folder = "Data/Subnetworks/Subnetwork2";
    performC2Correction(folder);

    string folder = "Data/Subnetworks/Subnetwork3";
    performC2Correction(folder);

    string folder = "Data/Subnetworks/Subnetwork4";
    performC2Correction(folder);

    string folder = "Data/Subnetworks/Subnetwork5";
    performC2Correction(folder);

    string folder = "Data/Subnetworks/Subnetwork6";
    performC2Correction(folder);

    string folder = "Data/Subnetworks/Subnetwork7";
    performC2Correction(folder);
```

```
    string folder = "Data/Subnetworks/Whole-brain";
    performC2Correction(folder);

    return 0;
}
```

Experiment: Draw Rough Phase Diagrams

```
#include "include/InverseIsingProblem.h"
#include "include/PhaseDiagram.h"
#include "include/CorrectC2.h"
#include "include/util.h"

#include <Eigen/Eigenvalues>

using namespace Eigen;
using namespace std;

/* Calculate rough phase diagram */
// Input: folder where problem is stored, range_mu and range_sigma

void calculatePDRough(const string& folder, VectorXd range_mu, VectorXd range_sigma) {
    InverseIsingProblem problem = InverseIsingProblem(folder);
    int N = (int) problem.return_params().first.size(); // nr nodes
    int B = (int) problem.return_data().cols(); //nr samples

    string folder_pd_rough = folder + "/phase_diagrams_rough";

    computePD(problem, folder_pd_rough, range_sigma, range_mu, 1e7, B);
}

int main() {

    /* Subnetwork 1 */
    folder = "Data/Subnetworks/Subnetwork1";
    range_mu_rough = VectorXd::LinSpaced(25, -0.075, 0.35);
    range_sigma_rough = VectorXd::LinSpaced(25, 0, 1.5);

    calculatePDRough(folder, range_mu_rough, range_sigma_rough)

    /* Subnetwork 2 */
    folder = "Data/Subnetworks/Subnetwork2";
    range_mu_rough = VectorXd::LinSpaced(25, -0.03, 0.2);
    range_sigma_rough = VectorXd::LinSpaced(25, 0, 0.65);

    calculatePDRough(folder, range_mu_rough, range_sigma_rough)

    /* Subnetwork 3 */
    folder = "Subnetwork3/Version_02";
    range_mu_rough = VectorXd::LinSpaced(25, -0.09, 0.35);
    range_sigma_rough = VectorXd::LinSpaced(25, 0, 1.4);

    calculatePDRough(folder, range_mu_rough, range_sigma_rough)

    /* Subnetwork 4 */
    string folder = "Data/Subnetworks/Subnetwork4";
    range_mu_rough = VectorXd::LinSpaced(25, -0.06, 0.25);
    range_sigma_rough = VectorXd::LinSpaced(25, 0, 1.25);

    calculatePDRough(folder, range_mu_rough, range_sigma_rough)
```

```
    /* Subnetwork 5 */
    folder = "Data/Subnetwork5/Version_02";
    range_mu_rough = VectorXd::LinSpaced(25, -0.08, 0.25);
    range_sigma_rough = VectorXd::LinSpaced(25, 0, 1.15);

    calculatePDRough(folder, range_mu_rough, range_sigma_rough)

    /* Subnetwork 6 */
    folder = "Data/Subnetworks/Subnetwork6/Version_02";
    range_mu_rough = VectorXd::LinSpaced(25, -0.1, 0.3);
    range_sigma_rough = VectorXd::LinSpaced(25, 0, 1.45);

    calculatePDRough(folder, range_mu_rough, range_sigma_rough)

    /* Subnetwork 7 */
    folder = "Data/Subnetworks/Subnetwork7/Version_02";
    range_mu_rough = VectorXd::LinSpaced(25, -0.05, 0.2);
    range_sigma_rough = VectorXd::LinSpaced(25, 0, 0.8);

    calculatePDRough(folder, range_mu_rough, range_sigma_rough)

    /* Whole-brain network */
    string folder = "Data/Whole-brain";
    range_mu_rough = VectorXd::LinSpaced(20, -0.0015, 0.01);
    range_sigma_rough = VectorXd::LinSpaced(20, 0, 0.14);

    calculatePDRough(folder, range_mu_rough, range_sigma_rough);

    return 0;
}
```

Experiment: Draw Refined Phase Diagrams

```
#include "include/InverseIsingProblem.h"
#include "include/PhaseDiagram.h"
#include "include/CorrectC2.h"
#include "include/util.h"

#include <Eigen/Eigenvalues>

using namespace Eigen;
using namespace std;

/* Calculate refined phase diagram */
// Input: folder where problem is stored, resolution refined phase diagrams
// Remark: rough phase diagram should be calculated and positions
// for participants to be approximated (using MATLAB)

void calculatePDRefined(const string& folder, int resolution) {
    InverseIsingProblem problem = InverseIsingProblem(folder);
    int N = (int) problem.return_params().first.size();
    int B = (int) problem.return_data().cols();

    InverseIsingProblem problem = InverseIsingProblem(folder);

    MatrixXd refined_interval = refinedInterval(folder_pd_rough);
    VectorXd range_sigma_refined = VectorXd::LinSpaced(resolution,
                                                       refined_interval(0,0), refined_interval(0,1));
    VectorXd range_mu_refined = VectorXd::LinSpaced(resolution,
                                                    refined_interval(1,0), refined_interval(1,1));

    string folder_pd_refined = folder + "/phase_diagrams_corrected_refined";
```

```
    computePD(problem, folder_pd_refined, range_sigma_refined, range_mu_refined, 1e7, B);
}

int main() {

    /* Subnetwork 1 */
    folder = "Data/Subnetworks/Subnetwork1";
    calculatePDRefined(folder, 25);

    /* Subnetwork 2 */
    folder = "Data/Subnetworks/Subnetwork2";
    calculatePDRefined(folder, 25);

    /* Subnetwork 3 */
    folder = "Subnetwork3/Version_02";
    calculatePDRefined(folder, 25);

    /* Subnetwork 4 */
    string folder = "Data/Subnetworks/Subnetwork4";
    calculatePDRefined(folder, 25);

    /* Subnetwork 5 */
    folder = "Data/Subnetwork5/Version_02";
    calculatePDRefined(folder, 25);

    /* Subnetwork 6 */
    folder = "Data/Subnetworks/Subnetwork6/Version_02";
    calculatePDRefined(folder, 25);

    /* Subnetwork 7 */
    folder = "Data/Subnetworks/Subnetwork7/Version_02";
    calculatePDRefined(folder, 25);

    /* Whole-brain network */
    string folder = "Data/Whole-brain";
    calculatePDRefined(folder, 25);

    return 0;
}
```

Analyze Phase Diagrams (MATLAB)

Determine μ, σ for each participant

Given the phase diagrams corresponding to an inferred PMEM, we can determine the parameters μ, σ for each participant using the experiment and function provided below.

Experiment: Determine μ, σ for each Participant

```
clear; close all;

%% Estimating and sigma for each participant
% Using the phase diagrams
% Output: file with the stats
% (mu, sigma, m, q, chiSG, chiUni) for each participant

network = 'Subnetworks/Subnetwork3'; %Adjust per network
folder =  sprintf('Data/%s/phase_diagrams', network);
labels = readmatrix(sprintf('Data/%s/labels.dat', network));
```

```
responders = readmatrix('Data/Participants/responders.dat');
nonresponders = readmatrix('Data/Participants/nonresponders.dat');
list_participants = [responders nonresponders];
stats_participants = [];

% General phase diagram
m_results = readmatrix(sprintf('%s/pd_m.dat', folder));
q_results = readmatrix(sprintf('%s/pd_q.dat', folder));
pd_Chi_Uni = readmatrix(sprintf('%s/pd_chiUni.dat', folder));
pd_Chi_SG = readmatrix(sprintf('%s/pd_chiSG.dat', folder));

mu_list = readmatrix(sprintf('%s/mu_list.dat', folder));
sigma_list = readmatrix(sprintf('%s/sigma_list.dat', folder));

% Data per participant
for x = list_participants
    binarized_data = func_collectDataParticipant(x, labels);

    [m, q, chi_SG, chi_Uni] = func_alculateObservables(binarized_data);
    [mu, sigma] = func_computePositionParticipant(m, q, chi_SG, chi_Uni, ...
        q_results, m_results, pd_Chi_SG, pd_Chi_Uni, mu_list, sigma_list);

    stats_participants = [stats_participants; x, sigma, mu, m,
        q, chi_SG, chi_Uni];
end
close all;

writematrix(stats_participants, ...
    sprintf('%s/stats_participants.dat', folder));
```

Function: Compute Position Participant

```
%% Compute position for each participant
% Input: observables calculated from data participant (m, q, chiSG, chiUni)
% and the phase diagrams for all observables.
% Output: mu, sigma that corresponds best to data participant (

function [mu, sigma] = func_computePositionParticipant(m, q, chi_SG, chi_Uni, q_results, m_results, pd_Chi_SG, pd

figure();
ylabel("\mu")
xlabel("\sigma")
hold on
ylim([mu_list(1), mu_list(end)])
xlim([sigma_list(1), sigma_list(end)])

%% Estimate mu-sigma curve
% satisfying pdChiUni(mu, sigma) = chiUni & pdChiSg(mu, sigma) = chiSG

c_Uni = contourc(sigma_list, mu_list, pd_Chi_Uni, [chi_Uni,chi_Uni]);
c_SG = contourc(sigma_list, mu_list, pd_Chi_SG, [chi_SG,chi_SG]);

if (size(c_Uni, 2) > 0) && (size(c_SG, 2) > 0)

    % Split curves Chi_Uni
    cs_Uni = {}; %contour splitted

    k = c_Uni(2, 1) + 1;
    temp = [];

    for i = 2:length(c_Uni)
        if i <= k
```

```
        temp = [temp c_Uni(:, i)];
    else
        cs_Uni = [cs_Uni temp];

        k = c_Uni(2,i) + i;
        temp = [];
    end
end

cs_Uni = [cs_Uni temp];

% Split curves Chi_SG
cs_SG = {};

k = c_SG(2, 1) + 1;
temp = [];

for i = 2:length(c_SG)
    if i <= k
        temp = [temp c_SG(:, i)];
    else
        cs_SG = [cs_SG temp];

        k = c_SG(2,i) + i;
        temp = [];
    end
end

cs_SG = [cs_SG temp];

% Collect intersection between curves
inter = [];

for curve_Uni = cs_Uni
    for curve_SG = cs_SG
        [sigma, mu] = intersections(curve_Uni{1,1}(1,:), ...
            curve_Uni{1,1}(2,:), curve_SG{1,1}(1,:), ...
            curve_SG{1,1}(2,:));
        inter = [inter [sigma mu]'];
    end
end

if (size(inter, 2) == 0)
    best = [nan; nan];
else
    best = [inter(:,1)];
    best_q = interp2(sigma_list, mu_list, q_results, ...
        inter(1,1), inter(2,1));
    best_m = interp2(sigma_list, mu_list, m_results, ...
        inter(1,1), inter(2,1));

    for j = 1:size(inter,2)
        temp_q = interp2(sigma_list, mu_list, q_results, ...
            inter(1,j), inter(2,j));
        temp_m = interp2(sigma_list, mu_list, m_results, ...
            inter(1,j), inter(2,j));

        if abs(temp_q - q) < abs(best_q - q) && ...
            abs(temp_m - m) < abs(best_m - m)^2
            best = [inter(:,j)];
        end
    end
```

```
    end

    sigma = best(1,1);
    mu = best(2,1);

else
    sigma = nan;
    mu = nan;

end


hold off

end
```

Function: Compute Observables Participant

```
%% Calculate observables of the dataset
% Input: binarized_data
% Ouput: m, q, chiSG, chiUni

function [m, q, chi_SG, chi_Uni] = func_calculateObservables(binarized_data)

local_magnetization = sum(binarized_data, 2)/size(binarized_data,2);
nr_nodes = size(local_magnetization, 1);
m = sum(local_magnetization, 1) / nr_nodes;
q = dot(local_magnetization, local_magnetization) / nr_nodes;
covariance = (binarized_data*binarized_data')/size(binarized_data, 2) - ...
local_magnetization * local_magnetization';
chi_Uni = sum(covariance, "all") / nr_nodes;
chi_SG = sum(covariance .* covariance, "all") / nr_nodes;

end
```

Function: intersections

This function is used to calculate the intersection between the level-sets of χ_{Uni} and χ_{SG}.

```
function [x0,y0,iout,jout] = intersections(x1,y1,x2,y2,robust)
%INTERSECTIONS Intersections of curves.
%   Computes the (x,y) locations where two curves intersect.  The curves
%   can be broken with NaNs or have vertical segments.
%
% Example:
%   [X0,Y0] = intersections(X1,Y1,X2,Y2,ROBUST);
%
% where X1 and Y1 are equal-length vectors of at least two points and
% represent curve 1.  Similarly, X2 and Y2 represent curve 2.
% X0 and Y0 are column vectors containing the points at which the two
% curves intersect.
%
% ROBUST (optional) set to 1 or true means to use a slight variation of the
% algorithm that might return duplicates of some intersection points, and
% then remove those duplicates.  The default is true, but since the
% algorithm is slightly slower you can set it to false if you know that
% your curves don't intersect at any segment boundaries.  Also, the robust
% version properly handles parallel and overlapping segments.
%
% The algorithm can return two additional vectors that indicate which
% segment pairs contain intersections and where they are:
%
```

```
%   [X0,Y0,I,J] = intersections(X1,Y1,X2,Y2,ROBUST);
%
% For each element of the vector I, I(k) = (segment number of (X1,Y1)) +
% (how far along this segment the intersection is).  For example, if I(k) =
% 45.25 then the intersection lies a quarter of the way between the line
% segment connecting (X1(45),Y1(45)) and (X1(46),Y1(46)).  Similarly for
% the vector J and the segments in (X2,Y2).
%
% You can also get intersections of a curve with itself.  Simply pass in
% only one curve, i.e.,
%
%   [X0,Y0] = intersections(X1,Y1,ROBUST);
%
% where, as before, ROBUST is optional.

% Version: 2.0, 25 May 2017
% Author:  Douglas M. Schwarz
% Email:   dmschwarz=ieee*org, dmschwarz=urgrad*rochester*edu
% Real_email = regexprep(Email,{'=','*'},{'@','.'})


% Theory of operation:
%
% Given two line segments, L1 and L2,
%
%   L1 endpoints:  (x1(1),y1(1)) and (x1(2),y1(2))
%   L2 endpoints:  (x2(1),y2(1)) and (x2(2),y2(2))
%
% we can write four equations with four unknowns and then solve them.  The
% four unknowns are t1, t2, x0 and y0, where (x0,y0) is the intersection of
% L1 and L2, t1 is the distance from the starting point of L1 to the
% intersection relative to the length of L1 and t2 is the distance from the
% starting point of L2 to the intersection relative to the length of L2.
%
% So, the four equations are
%
%    (x1(2) - x1(1))*t1 = x0 - x1(1)
%    (x2(2) - x2(1))*t2 = x0 - x2(1)
%    (y1(2) - y1(1))*t1 = y0 - y1(1)
%    (y2(2) - y2(1))*t2 = y0 - y2(1)
%
% Rearranging and writing in matrix form,
%
%  [x1(2)-x1(1)        0        -1   0;        [t1;        [-x1(1);
%        0        x2(2)-x2(1)  -1   0;    *    t2;    =    -x2(1);
%   y1(2)-y1(1)        0         0  -1;         x0;         -y1(1);
%        0        y2(2)-y2(1)   0  -1]        y0]         -y2(1)]
%
% Let's call that A*T = B.  We can solve for T with T = A\B.
%
% Once we have our solution we just have to look at t1 and t2 to determine
% whether L1 and L2 intersect.  If 0 <= t1 < 1 and 0 <= t2 < 1 then the two
% line segments cross and we can include (x0,y0) in the output.
%
% In principle, we have to perform this computation on every pair of line
% segments in the input data.  This can be quite a large number of pairs so
% we will reduce it by doing a simple preliminary check to eliminate line
% segment pairs that could not possibly cross.  The check is to look at the
% smallest enclosing rectangles (with sides parallel to the axes) for each
% line segment pair and see if they overlap.  If they do then we have to
% compute t1 and t2 (via the A\B computation) to see if the line segments
% cross, but if they don't then the line segments cannot cross.  In a
```

```
% typical application, this technique will eliminate most of the potential
% line segment pairs.


% Input checks.
if verLessThan('matlab','7.13')
        error(nargchk(2,5,nargin)) %#ok<NCHKN>
else
        narginchk(2,5)
end

% Adjustments based on number of arguments.
switch nargin
        case 2
                robust = true;
                x2 = x1;
                y2 = y1;
                self_intersect = true;
        case 3
                robust = x2;
                x2 = x1;
                y2 = y1;
                self_intersect = true;
        case 4
                robust = true;
                self_intersect = false;
        case 5
                self_intersect = false;
end

% x1 and y1 must be vectors with same number of points (at least 2).
if sum(size(x1) > 1) ~= 1 || sum(size(y1) > 1) ~= 1 || ...
                length(x1) ~= length(y1)
        error('X1 and Y1 must be equal-length vectors of at least 2 points.')
end
% x2 and y2 must be vectors with same number of points (at least 2).
if sum(size(x2) > 1) ~= 1 || sum(size(y2) > 1) ~= 1 || ...
                length(x2) ~= length(y2)
        error('X2 and Y2 must be equal-length vectors of at least 2 points.')
end


% Force all inputs to be column vectors.
x1 = x1(:);
y1 = y1(:);
x2 = x2(:);
y2 = y2(:);

% Compute number of line segments in each curve and some differences we'll
% need later.
n1 = length(x1) - 1;
n2 = length(x2) - 1;
xy1 = [x1 y1];
xy2 = [x2 y2];
dxy1 = diff(xy1);
dxy2 = diff(xy2);


% Determine the combinations of i and j where the rectangle enclosing the
% i'th line segment of curve 1 overlaps with the rectangle enclosing the
% j'th line segment of curve 2.
```

```
% Original method that works in old MATLAB versions, but is slower than
% using binary singleton expansion (explicit or implicit).
% [i,j] = find( ...
%       repmat(mvmin(x1),1,n2) <= repmat(mvmax(x2).',n1,1) & ...
%       repmat(mvmax(x1),1,n2) >= repmat(mvmin(x2).',n1,1) & ...
%       repmat(mvmin(y1),1,n2) <= repmat(mvmax(y2).',n1,1) & ...
%       repmat(mvmax(y1),1,n2) >= repmat(mvmin(y2).',n1,1));

% Select an algorithm based on MATLAB version and number of line
% segments in each curve.  We want to avoid forming large matrices for
% large numbers of line segments.  If the matrices are not too large,
% choose the best method available for the MATLAB version.
if n1 > 1000 || n2 > 1000 || verLessThan('matlab','7.4')
        % Determine which curve has the most line segments.
        if n1 >= n2
                % Curve 1 has more segments, loop over segments of curve 2.
                ijc = cell(1,n2);
                min_x1 = mvmin(x1);
                max_x1 = mvmax(x1);
                min_y1 = mvmin(y1);
                max_y1 = mvmax(y1);
                for k = 1:n2
                        k1 = k + 1;
                        ijc{k} = find( ...
                                min_x1 <= max(x2(k),x2(k1)) & max_x1 >= min(x2(k),x2(k1)) & ...
                                min_y1 <= max(y2(k),y2(k1)) & max_y1 >= min(y2(k),y2(k1)));
                        ijc{k}(:,2) = k;
                end
                ij = vertcat(ijc{:});
                i = ij(:,1);
                j = ij(:,2);
        else
                % Curve 2 has more segments, loop over segments of curve 1.
                ijc = cell(1,n1);
                min_x2 = mvmin(x2);
                max_x2 = mvmax(x2);
                min_y2 = mvmin(y2);
                max_y2 = mvmax(y2);
                for k = 1:n1
                        k1 = k + 1;
                        ijc{k}(:,2) = find( ...
                                min_x2 <= max(x1(k),x1(k1)) & max_x2 >= min(x1(k),x1(k1)) & ...
                                min_y2 <= max(y1(k),y1(k1)) & max_y2 >= min(y1(k),y1(k1)));
                        ijc{k}(:,1) = k;
                end
                ij = vertcat(ijc{:});
                i = ij(:,1);
                j = ij(:,2);
        end

elseif verLessThan('matlab','9.1')
        % Use bsxfun.
        [i,j] = find( ...
                bsxfun(@le,mvmin(x1),mvmax(x2).') & ...
                bsxfun(@ge,mvmax(x1),mvmin(x2).') & ...
                bsxfun(@le,mvmin(y1),mvmax(y2).') & ...
                bsxfun(@ge,mvmax(y1),mvmin(y2).'));

else
        % Use implicit expansion.
        [i,j] = find( ...
                mvmin(x1) <= mvmax(x2).' & mvmax(x1) >= mvmin(x2).' & ...
```

```
                mvmin(y1) <= mvmax(y2).' & mvmax(y1) >= mvmin(y2).');

end


% Find segments pairs which have at least one vertex = NaN and remove them.
% This line is a fast way of finding such segment pairs.  We take
% advantage of the fact that NaNs propagate through calculations, in
% particular subtraction (in the calculation of dxy1 and dxy2, which we
% need anyway) and addition.
% At the same time we can remove redundant combinations of i and j in the
% case of finding intersections of a line with itself.
if self_intersect
        remove = isnan(sum(dxy1(i,:) + dxy2(j,:),2)) | j <= i + 1;
else
        remove = isnan(sum(dxy1(i,:) + dxy2(j,:),2));
end
i(remove) = [];
j(remove) = [];

% Initialize matrices.  We'll put the T's and B's in matrices and use them
% one column at a time.  AA is a 3-D extension of A where we'll use one
% plane at a time.
n = length(i);
T = zeros(4,n);
AA = zeros(4,4,n);
AA([1 2],3,:) = -1;
AA([3 4],4,:) = -1;
AA([1 3],1,:) = dxy1(i,:).';
AA([2 4],2,:) = dxy2(j,:).';
B = -[x1(i) x2(j) y1(i) y2(j)].';

% Loop through possibilities.  Trap singularity warning and then use
% lastwarn to see if that plane of AA is near singular.  Process any such
% segment pairs to determine if they are colinear (overlap) or merely
% parallel.  That test consists of checking to see if one of the endpoints
% of the curve 2 segment lies on the curve 1 segment.  This is done by
% checking the cross product
%
%   (x1(2),y1(2)) - (x1(1),y1(1)) x (x2(2),y2(2)) - (x1(1),y1(1)).
%
% If this is close to zero then the segments overlap.

% If the robust option is false then we assume no two segment pairs are
% parallel and just go ahead and do the computation.  If A is ever singular
% a warning will appear.  This is faster and obviously you should use it
% only when you know you will never have overlapping or parallel segment
% pairs.

if robust
        overlap = false(n,1);
        warning_state = warning('off','MATLAB:singularMatrix');
        % Use try-catch to guarantee original warning state is restored.
        try
                lastwarn('')
                for k = 1:n
                        T(:,k) = AA(:,:,k)\B(:,k);
                        [unused,last_warn] = lastwarn; %#ok<ASGLU>
                        lastwarn('')
                        if strcmp(last_warn,'MATLAB:singularMatrix')
                                % Force in_range(k) to be false.
                                T(1,k) = NaN;
```

```
                                % Determine if these segments overlap or are just parallel.
                                overlap(k) = rcond([dxy1(i(k),:);xy2(j(k),:) - ...
                    xy1(i(k),:)]) < eps;
                        end
                end
                warning(warning_state)
        catch err
                warning(warning_state)
                rethrow(err)
        end
        % Find where t1 and t2 are between 0 and 1 and return the corresponding
        % x0 and y0 values.
        in_range = (T(1,:) >= 0 & T(2,:) >= 0 & T(1,:) <= 1 & T(2,:) <= 1).';
        % For overlapping segment pairs the algorithm will return an
        % intersection point that is at the center of the overlapping region.
        if any(overlap)
                ia = i(overlap);
                ja = j(overlap);
                % set x0 and y0 to middle of overlapping region.
                T(3,overlap) = (max(min(x1(ia),x1(ia+1)),min(x2(ja),x2(ja+1))) + ...
                        min(max(x1(ia),x1(ia+1)),max(x2(ja),x2(ja+1)))).'/2;
                T(4,overlap) = (max(min(y1(ia),y1(ia+1)),min(y2(ja),y2(ja+1))) + ...
                        min(max(y1(ia),y1(ia+1)),max(y2(ja),y2(ja+1)))).'/2;
                selected = in_range | overlap;
        else
                selected = in_range;
        end
        xy0 = T(3:4,selected).';

        % Remove duplicate intersection points.
        [xy0,index] = unique(xy0,'rows');
        x0 = xy0(:,1);
        y0 = xy0(:,2);

        % Compute how far along each line segment the intersections are.
        if nargout > 2
                sel_index = find(selected);
                sel = sel_index(index);
                iout = i(sel) + T(1,sel).';
                jout = j(sel) + T(2,sel).';
        end
else % non-robust option
        for k = 1:n
                [L,U] = lu(AA(:,:,k));
                T(:,k) = U\(L\B(:,k));
        end

        % Find where t1 and t2 are between 0 and 1 and return the corresponding
        % x0 and y0 values.
        in_range = (T(1,:) >= 0 & T(2,:) >= 0 & T(1,:) < 1 & T(2,:) < 1).';
        x0 = T(3,in_range).';
        y0 = T(4,in_range).';

        % Compute how far along each line segment the intersections are.
        if nargout > 2
                iout = i(in_range) + T(1,in_range).';
                jout = j(in_range) + T(2,in_range).';
        end
end

% Plot the results (useful for debugging).
% plot(x1,y1,x2,y2,x0,y0,'ok');
```

```
function y = mvmin(x)
% Faster implementation of movmin(x,k) when k = 1.
y = min(x(1:end-1),x(2:end));

function y = mvmax(x)
% Faster implementation of movmax(x,k) when k = 1.
y = max(x(1:end-1),x(2:end));
```

Calculate Distance to Criticality

Using the phase diagrams and determined positions of the participants in it, we can calculate the distance to the phase transitions. This experiment calculates the distances to the phase transitions and write these to a file. Additionally, the phase diagrams with approximated boundaries between and the positions of the participants are generated.

Experiment: Calculate Distance

```
clear; close all;

%% Calculate distance to phase transition, save data and plot participants in phase diagrams
% Input: phase_diagrams_rough and phase_diagrams_refined
% Output: distances to phase transitions and plot of the diagrams

responders = readmatrix('Data/Participants/responders.dat');
nonresponders = readmatrix('Data/Participants/nonresponders.dat');
list_participants = [responders nonresponders];

all_stats_p = [];

% Subnetworks
for i = 1:7
    folder = sprintf('Data/Subnetworks/Subnetwork%d/phase_diagrams_rough', i);

    %% Collect phase diagrams, stats participants rough data
    pd_m = readmatrix(sprintf('%s/pd_m.dat', folder));
    pd_q = readmatrix(sprintf('%s/pd_q.dat', folder));
    pd_ChiUni = readmatrix(sprintf('%s/pd_chiUni.dat', folder));
    pd_ChiSG = readmatrix(sprintf('%s/pd_chiSG.dat', folder));

    mu_list = readmatrix(sprintf('%s/mu_list.dat', folder));
    sigma_list = readmatrix(sprintf('%s/sigma_list.dat', folder));


    %% Determine bound between phase transitions
    [curve_chiUni] = func_calculateBoundUni(pd_ChiUni, mu_list, sigma_list);
    [curve_chiSG] = func_calculateBoundSG(pd_ChiSG, pd_ChiUni, mu_list, sigma_list);

    curve_chiSG = [curve_chiSG(1,:); curve_chiSG(2,:) - 1.5*(sigma_list(end) ...
        - sigma_list(1))/length(sigma_list)*ones(1,size(curve_chiSG(2,:), 2))];
    curve_chiUni = [curve_chiUni(1,:) - 1.5*(mu_list(end) ...
        - mu_list(1))/length(mu_list)*ones(1,size(curve_chiUni(1,:), 2)); curve_chiUni(2,:)];


    %% Collect refined data
    folder = sprintf('Data/Subnetworks/Subnetwork%d/phase_diagrams_refined', i);
    pd_m = readmatrix(sprintf('%s/pd_m.dat', folder));
    pd_q = readmatrix(sprintf('%s/pd_q.dat', folder));
    pd_ChiUni = readmatrix(sprintf('%s/pd_chiUni.dat', folder));
    pd_ChiSG = readmatrix(sprintf('%s/pd_chiSG.dat', folder));

    mu_list = readmatrix(sprintf('%s/mu_list.dat', folder));
```

```
sigma_list = readmatrix(sprintf('%s/sigma_list.dat', folder));

stats_p = readmatrix(sprintf('%s/stats_p.dat', folder));
stats_p = [stats_p, zeros(size(stats_p, 1), 2)];

mean_stats_p = mean(stats_p, 1);
mean_sigma = mean_stats_p(2);
mean_mu = mean_stats_p(3);


%% Project participants on cross section
chiSG_cross = interp2(sigma_list, mu_list, pd_ChiSG, sigma_list, mean_mu);
chiUni_cross = interp2(sigma_list, mu_list, pd_ChiUni, mean_sigma, mu_list);

fig1 = figure();
plot(sigma_list, chiSG_cross,'LineWidth', 1, 'Color','black')
hold on;

for m = 1:length(list_participants)
    pos = interp1(sigma_list, chiSG_cross, [stats_p(m,2)]);

    if m <= length(responders)
        scatter(stats_p(m,2), pos, 'r', 'o', 'filled');
    else
        scatter(stats_p(m,2), pos, 'k', 'o', 'filled');
    end
end

xlabel("\sigma", "FontSize",20)
ylabel("\chi_{SG}", "FontSize",20)
xlim([sigma_list(1) sigma_list(end)]);
ylim([min(pd_ChiSG, [], "all"), max(pd_ChiSG, [], "all")])
hold off;

fig2 = figure();
plot(mu_list, chiUni_cross','LineWidth', 1, 'Color','black')
hold on;

for m = 1:length(list_participants)
    pos = interp1(mu_list, chiUni_cross', [stats_p(m,3)]);

    if m <= length(responders)
        scatter(stats_p(m,3), pos, 'r', 'o', 'filled');
    else
        scatter(stats_p(m,3), pos, 'k', 'o', 'filled');
    end
end

xlabel("\mu", "FontSize",20)
ylabel("\chi_{Uni}", "FontSize",20)
xlim([mu_list(1) mu_list(end)]);
ylim([min(pd_ChiUni, [], "all"), max(pd_ChiUni, [], "all")])
hold off;


%% Plot refined phase diagrams
fig5 = figure();
hold on
surf(mu_list, sigma_list',  pd_ChiSG', EdgeColor="none")

scatter3(stats_p(1:length(responders), 3), ...
    stats_p(1:length(responders), 2), 100+ stats_p(1:length(responders), 6),'.', 'r');
```

```
scatter3(stats_p(length(responders)+1:length(list_participants), 3), ...
    stats_p(length(responders)+1:length(list_participants), 2), ...
    stats_p(length(responders)+1:length(list_participants), 6) + 100, '.', 'w');

plot3(mean_mu*ones(length(mu_list)), sigma_list', chiSG_cross);

xlim([mu_list(1), mu_list(end)]);
ylim([sigma_list(1) sigma_list(end)]);
xlabel("\mu", "FontSize",20)
ylabel("\sigma", "FontSize",20)
view(2)
axis square;
colormap("turbo")
colorbar('southoutside')
title('\chi_{SG}',"FontSize",20)
hold off


fig6 = figure();
hold on
surf(mu_list, sigma_list', pd_ChiUni', EdgeColor="none")

scatter3(stats_p(1:length(responders), 3), ...
    stats_p(1:length(responders), 2), ...
    stats_p(1:length(responders), 7)+ 100, '.', 'r');
scatter3(stats_p(length(responders)+1:length(list_participants), 3), ...
    stats_p(length(responders)+1:length(list_participants), 2), ...
    stats_p(length(responders)+1:length(list_participants), 7) + 100, '.', 'w');

plot3(mu_list, mean_sigma*ones(length(sigma_list))', chiUni_cross);

xlim([mu_list(1), mu_list(end)]);
ylim([sigma_list(1) sigma_list(end)]);
xlabel("\mu", "FontSize",20)
ylabel("\sigma", "FontSize",20)
colormap("turbo")
view(2)
axis square;
colorbar('southoutside')
title('\chi_{Uni}', "FontSize",20)
hold off

%% Determine distance from participants to phase transition

for j = 1:length(stats_p)
   mu = stats_p(j, 3);
    sigma = stats_p(j, 2);

    distances_SG = sqrt((curve_chiSG(1, :) ...
        - ones(1, size(curve_chiSG,2)) * mu).^2 + (curve_chiSG(2, :) ...
        - ones(1, size(curve_chiSG, 2)) * sigma).^2);
    [mininmal_distance_SG, index_SG] = min(distances_SG);

    distances_Uni = sqrt((curve_chiUni(1, :) ...
        - ones(1, size(curve_chiUni,2)) * mu).^2 + (curve_chiUni(2, :) ...
        - ones(1, size(curve_chiUni, 2)) * sigma).^2);
    [mininmal_distance_Uni, index_Uni] = min(distances_Uni);

    stats_p(j, 8:9) = [mininmal_distance_SG, mininmal_distance_Uni];
end

all_stats_p = [all_stats_p, stats_p(:, 8:9)];
```

```
    fig3 = figure();
    hold on
    surf(mu_list, sigma_list', abs(pd_m)', EdgeColor="none")

    scatter3(stats_p(1:length(responders), 3), ...
        stats_p(1:length(responders), 2), ...
        abs(stats_p(1:length(responders), 4))+ 100, '.', 'r');
    scatter3(stats_p(length(responders)+1:length(list_participants), 3), ...
        stats_p(length(responders)+1:length(list_participants), 2)+ ...
        abs(stats_p(length(responders)+1:length(list_participants), 4)), 100, '.', 'w');

    colormap("turbo")
    clim([0.01,1]);
    xlabel("\mu", "FontSize",20)
    ylabel("\sigma", "FontSize",20)
    xlim([mu_list(1), mu_list(end)]);
    ylim([sigma_list(1) sigma_list(end)]);
    view(2)
    axis square;
    colorbar('southoutside')
    title('|m|',"FontSize",20)
    hold off


    fig4 = figure();
    hold on
    surf(mu_list,sigma_list',  pd_q', EdgeColor="none")

    scatter3(stats_p(1:length(responders), 3), ...
        stats_p(1:length(responders), 2), ...
        stats_p(1:length(responders), 5)+ 100, '.', 'r');
    scatter3(stats_p(length(responders)+1:length(list_participants), 3), ...
        stats_p(length(responders)+1:length(list_participants), 2), ...
        stats_p(length(responders)+1:length(list_participants), 5)+ 100, '.', 'w');

    colormap("turbo")
    clim([0.0001,1])
    xlim([mu_list(1), mu_list(end)]);
    ylim([sigma_list(1) sigma_list(end)]);
    xlabel("\mu","FontSize",20)
    ylabel("\sigma","FontSize",20)
    view(2)
    axis square;
    colorbar('southoutside')
    title('q',"FontSize",20);
    hold off
end

%% Whole-brain

%% Collect phase diagrams, stats participants rough data
folder = 'Data/Whole-brain/phase_diagrams_rough';

pd_m = readmatrix(sprintf('%s/pd_m.dat', folder));
pd_q = readmatrix(sprintf('%s/pd_q.dat', folder));
pd_ChiUni = readmatrix(sprintf('%s/pd_chiUni.dat', folder));
pd_ChiSG = readmatrix(sprintf('%s/pd_chiSG.dat', folder));

mu_list = readmatrix(sprintf('%s/mu_list.dat', folder));
sigma_list = readmatrix(sprintf('%s/sigma_list.dat', folder));
```

```
%% Determine bound between phase transitions

[curve_chiUni] = func_calculateBoundUni(pd_ChiUni, mu_list, sigma_list);
[curve_chiSG] = func_calculateBoundSG(pd_ChiSG, pd_ChiUni, mu_list, sigma_list);

curve_chiSG = [curve_chiSG(1,:); curve_chiSG(2,:) - ...
    1.5*(sigma_list(end) - sigma_list(1))/length(sigma_list)*ones(1,size(curve_chiSG(2,:), 2))];
curve_chiUni = [curve_chiUni(1,:) - 1.5*(mu_list(end) - ...
   mu_list(1))/length(mu_list)*ones(1,size(curve_chiUni(1,:), 2)); curve_chiUni(2,:)];


%% Collect refined data
folder = 'Data/Whole-brain/phase_diagrams_refined';

pd_m = readmatrix(sprintf('%s/pd_m.dat', folder));
pd_q = readmatrix(sprintf('%s/pd_q.dat', folder));
pd_ChiUni = readmatrix(sprintf('%s/pd_chiUni.dat', folder));
pd_ChiSG = readmatrix(sprintf('%s/pd_chiSG.dat', folder));

mu_list = readmatrix(sprintf('%s/mu_list.dat', folder));
sigma_list = readmatrix(sprintf('%s/sigma_list.dat', folder));

stats_p = readmatrix(sprintf('%s/stats_p.dat', folder));
stats_p = [stats_p, zeros(size(stats_p, 1), 2)];

mean_stats_p = mean(stats_p, 1);
mean_sigma = mean_stats_p(2);
mean_mu = mean_stats_p(3);


%% Project participants on cross section
chiSG_cross = interp2(sigma_list, mu_list, pd_ChiSG, sigma_list, mean_mu);
chiUni_cross = interp2(sigma_list, mu_list, pd_ChiUni, mean_sigma, mu_list);

fig1 = figure();
plot(sigma_list, chiSG_cross,'LineWidth', 1, 'Color','black')
hold on;

for m = 1:length(list_participants)
    pos = interp1(sigma_list, chiSG_cross, [stats_p(m,2)]);

    if m <= length(responders)
        scatter(stats_p(m,2), pos, 'r', 'o', 'filled');
    else
        scatter(stats_p(m,2), pos, 'k', 'o', 'filled');
    end
end

xlabel("\sigma","FontSize",20)
ylabel("\chi_{SG}","FontSize",20)
xlim([sigma_list(1) sigma_list(end)]);
ylim([min(pd_ChiSG, [], "all"), max(pd_ChiSG, [], "all")])
hold off;

fig2 = figure();
plot(mu_list, chiUni_cross','LineWidth', 1, 'Color','black')
hold on;

for m = 1:length(list_participants)
    pos = interp1(mu_list, chiUni_cross', [stats_p(m,3)]);
```

```
    if m <= length(responders)
        scatter(stats_p(m,3), pos, 'r', 'o', 'filled');
    else
        scatter(stats_p(m,3), pos, 'k', 'o', 'filled');
    end
end

xlabel("\mu","FontSize",20)
ylabel("\chi_{Uni}","FontSize",20)
xlim([mu_list(1) mu_list(end)]);
ylim([min(pd_ChiUni, [], "all"), max(pd_ChiUni, [], "all")])
hold off;


%% Plot refined phase diagrams
fig5 = figure();
hold on
surf(mu_list, sigma_list', pd_ChiSG', EdgeColor="none")

scatter3(stats_p(1:length(responders), 3), ...
    stats_p(1:length(responders), 2), ...
    stats_p(1:length(responders), 6)+ 100, '.', 'r');
scatter3(stats_p(length(responders)+1:length(list_participants), 3), ...
    stats_p(length(responders)+1:length(list_participants), 2), ...
    stats_p(length(responders)+1:length(list_participants), 6)+ 100, '.', 'w');

plot3(mean_mu*ones(length(mu_list)), sigma_list', chiSG_cross);

xlim([mu_list(1), mu_list(end)]);
ylim([sigma_list(1) sigma_list(end)]);
xlabel("\mu","FontSize",20)
ylabel("\sigma","FontSize",20)
view(2)
axis square;
colormap("turbo")
colorbar('southoutside')
title('\chi_{SG}', "FontSize",20)
hold off


fig6 = figure();
hold on
surf(mu_list, sigma_list', pd_ChiUni', EdgeColor="none")

scatter3(stats_p(1:length(responders), 3), ...
    stats_p(1:length(responders), 2), ...
    stats_p(1:length(responders), 7) + 100, '.', 'r');
scatter3(stats_p(length(responders)+1:length(list_participants), 3), ...
    stats_p(length(responders)+1:length(list_participants), 2), ...
    stats_p(length(responders)+1:length(list_participants), 7)+ 100, '.', 'w');

plot3(mu_list, mean_sigma*ones(length(sigma_list))', chiUni_cross);

xlim([mu_list(1), mu_list(end)]);
ylim([sigma_list(1) sigma_list(end)]);
xlabel("\mu","FontSize",20)
ylabel("\sigma","FontSize",20)
colormap("turbo")
view(2)
axis square;
colorbar('southoutside')
title('\chi_{Uni}',"FontSize",20)
```

```
hold off

%% Determine distance from participants to phase transition

for j = 1:length(stats_p)
   mu = stats_p(j, 3);
    sigma = stats_p(j, 2);

    distances_SG = sqrt((curve_chiSG(1, :) - ones(1, size(curve_chiSG,2)) * mu).^2 + ...
        (curve_chiSG(2, :) - ones(1, size(curve_chiSG, 2)) * sigma).^2);
    [mininmal_distance_SG, index_SG] = min(distances_SG);

    distances_Uni = sqrt((curve_chiUni(1, :) - ones(1, size(curve_chiUni,2)) * mu).^2 + ...
        (curve_chiUni(2, :) - ones(1, size(curve_chiUni, 2)) * sigma).^2);
    [mininmal_distance_Uni, index_Uni] = min(distances_Uni);

    stats_p(j, 8:9) = [mininmal_distance_SG, mininmal_distance_Uni];
end

all_stats_p = [all_stats_p, stats_p(:, 8:9)];


fig3 = figure();
hold on
surf(mu_list, sigma_list', abs(pd_m)', EdgeColor="none")

scatter3(stats_p(1:length(responders), 3), ...
    stats_p(1:length(responders), 2), ...
    abs(stats_p(1:length(responders), 4)) + 100, '.', 'r');
scatter3(stats_p(length(responders)+1:length(list_participants), 3), ...
    stats_p(length(responders)+1:length(list_participants), 2), ...
    abs(stats_p(length(responders)+1:length(list_participants), 4)) + 100, '.', 'w');

colormap("turbo")
clim([0.01,1]);
xlabel("\mu","FontSize",20)
ylabel("\sigma","FontSize",20)
xlim([mu_list(1), mu_list(end)]);
ylim([sigma_list(1) sigma_list(end)]);
view(2)
axis square;
colorbar('southoutside')
title('|m|',"FontSize",20)
hold off


fig4 = figure();
hold on
surf(mu_list,sigma_list',  pd_q', EdgeColor="none")

scatter3(stats_p(1:length(responders), 3), ...
    stats_p(1:length(responders), 2), ...
    stats_p(1:length(responders), 5) + 100, '.', 'r');
scatter3(stats_p(length(responders)+1:length(list_participants), 3), ...
    stats_p(length(responders)+1:length(list_participants), 2), ...
    stats_p(length(responders)+1:length(list_participants), 5) + 100, '.', 'w');

colormap("turbo")
clim([0.0001,1])
xlim([mu_list(1), mu_list(end)]);
ylim([sigma_list(1) sigma_list(end)]);
xlabel("\mu","FontSize",20)
```

```
ylabel("\sigma","FontSize",20)
view(2)
axis square;
colorbar('southoutside')
title('q', "FontSize",20);
hold off


%% Collect and save data

all_stats_p = [list_participants', all_stats_p];
list_observables = [];

for i = 1:7
    list_observables = [list_observables, [sprintf("dist2SG_s%d", i), sprintf("dist2Uni_s%d", i)]];
end
list_observables = [list_observables, ["dist2SG_wb", "dist2Uni_wb"]];

list_observables = ["participant_nr", list_observables];

table = array2table(all_stats_p, 'VariableNames', list_observables);
stats = readtable('Data/stats_p.csv');
table = [table stats(:,2:end)];

writetable(table, 'outputRPDA_corrected_subnetworks.csv');

close all;
```

Function: Calculate Boundary Between Paramagnetic and Spin-Glass Phase

```
%% Calculate the boundary between the paramagnetic and SG phase
% Input: phase diagrams
% Output: set of points that defines the boundary

function [curve_SG] = func_calculateBoundSG(pd_ChiSG, pd_ChiUni, pd_mu, pd_sigma)

res_mu = size(pd_mu, 1);
res_sigma = size(pd_sigma, 1);

bound_SG = [];  %sigma as a function of mu

% Remove overlap between chiUni and chiSG
pd_ChiSG_temp = pd_ChiSG - 0.5 .* pd_ChiUni;

% Determine where chiSG is maximal
j = 1;
while j <= res_mu
    [value, index] = max(pd_ChiSG_temp(j, :));
    bound_SG = [bound_SG, [pd_mu(j); pd_sigma(index)]];
    if index == res_sigma
        break
    end
    j = j + 1;
end

% Fit a curve to the found points
f = fittype('exp2');
spline_SG = fit(bound_SG(1,:)', bound_SG(2,:)', f);

% Return the curve
range_SG = bound_SG(1,1):((bound_SG(1,2)-bound_SG(1,1))/100000):bound_SG(1,end);
curve_SG = feval(spline_SG, range_SG);
```

```
curve_SG = [range_SG; curve_SG'];

curve_SG = [curve_SG(1,:); curve_SG(2,:)];

end
```

Function: Calculate Boundary Between Para- and Ferromagnetic Phase

```
%% Calculate the boundary between the para- and ferromagnetic phase
% Input: phase diagrams
% Output: set of points that defines the boundary

function [curve_Uni] = func_calculateBoundUni(pd_ChiUni, pd_mu, pd_sigma)

res_mu = size(pd_mu, 1);
res_sigma = size(pd_sigma, 1);

%% Determine the boundaries in the phase diagrams of m and q
% In pd_m: transition from m = 0 to m \neq 1;
% In pd_q: transition from q = 0 to q > 0;

bound_Uni = []; %mu as a function of sigma

% Determine where chiUni is maximal
i = 1;
while i <= res_sigma
    [value, index] = max(pd_ChiUni(:, i));
    bound_Uni = [bound_Uni, [pd_mu(index); pd_sigma(i)]];
    if index == res_mu
        break
    end
    i = i + 1;
end

% Fit a curve to the found points
f = fittype('poly5');
spline_Uni = fit(bound_Uni(2,:)', bound_Uni(1,:)', f) ;
range_Uni = bound_Uni(2,1):((bound_Uni(2,2)-bound_Uni(2,1))/100000):bound_Uni(2,end);
curve_Uni = feval(spline_Uni, range_Uni);
curve_Uni = [curve_Uni'; range_Uni];

curve_Uni = [curve_Uni(1,:); curve_Uni(2,:)];

end
```

Entropy Analysis (MATLAB)

The entropy analysis was performed in MATLAB. In this section, we provide the experiment that computes the Permutation Fuzzy Entropy (PFE) per ROI for each participant, including the averaging procedure. The algorithm for calculating the PFE of a signal is provided separately.

Experiment: Compute Permutation Fuzzy Entropy (PFE)

```
%% Compute Permutation Fuzzy Entropy of the fMRI signals for each participant.
% Generates three output files:
% the PFE for each ROI,
% functional network/ subnetwork
% and the whole-brain network.

responders = readmatrix('/.../Data/Participants/responders.dat');
```

```
nonresponders = readmatrix('/.../Data/Participants/nonresponders.dat');
list_participants = [responders nonresponders];

PFE_all = [];

for k = list_participants
    k
    signals = func_readROISignal(k, 1:238, 1, 310);
    temp = zeros(1, size(signals, 1));

    for i = 1:size(signals, 1)
        signal = signals(i, :);
        r = std(signal); %Similarity tolerance

        PFE = func_computePFE(signal, 4, 1, 2, 0.25*r);
        %%Parameter choices as in Niu et al (2020)
        temp(i) = PFE;
    end

    PFE_all = [PFE_all, temp'];
end


%% Write results to output.csv, including stats participants
results_all = [list_participants', PFE_all];
list_ROIs = [];

for i = 1:size(PFE_all, 2)
    list_ROIs = [list_ROIs, sprintf("ROI_%d", i)];
end
list_ROIs = ["participant_nr", list_ROIs];

table = array2table(results_all, 'VariableNames', list_ROIs);
stats = readtable('/.../Data/stats_participants.csv');
table = [table stats(:,2:end)];

writetable(table, 'outputPFE.csv');


%% Write results to output file averaged over wb
results_all = [list_participants', mean(PFE_all, 2)];

table = array2table(results_all, 'VariableNames', ["participant_nr", "wb"]);
table = [table stats(:,2:end)];

writetable(table, 'outputPFE_wb.csv')


%% Write results to output file per subnetwork, including stats participants
label_subn_ROI = readmatrix("/.../Brainnetome_ROIs_network_labels_Yeo.xlsx");
temp_results_subn = [label_subn_ROI'; PFE_all];

results_subn = [];
for i = 1:7
    temp = all(temp_results_subn(1, :) == i, 1);
    results_subn = [results_subn, mean(temp_results_subn(2:end, temp), 2)];
end

results_subn = [list_participants', results_subn];

list_subn = [];
```

```
for i = 1:size(results_subn, 2) - 1
    list_subn = [list_subn, sprintf("subnetwork_%d", i)];
end

list_subn = ["participant_nr", list_subn];

table = array2table(results_subn, 'VariableNames', list_subn);
stats = readtable('/.../Data/stats_participants.csv');
table = [table stats(:,2:end)];

writetable(table, 'outputPFE_subnetworks.csv');
```

Function: Compute Permutation Fuzzy Entropy (PFE)

```
%% Calculate permutation entropy
% Input: signal, order of permutation entropy/
embedding dimension pm, delay time of permutation
% entropy tau, phase space dimension m and similar tolerance r.
% Output: permutation fuzzy entropy.

function [PFE] = func_computePFE(signal, pm, tau, m, r)
    B = length(signal);
    list_permutations = perms(1:pm);
    U(B-(pm-1)*tau) = 0; %Initialize phase space.

    %Phase space reconstruction following as for PE
    for i = 1:B-tau*(pm-1)
        [a, indices] = sort(signal(i:tau:i+tau*(pm-1)));
        for j = 1:length(list_permutations)
            if (abs(list_permutations(j,:) - indices)) == 0
                U(i) = j;
                break
            end
        end
    end

    %For the newly obtained time series U(i) (i=1,...,B-(pm-1)*tau), the
    %fuzzy entropy is calculated.

    PFE = log(phi(U, m, r)) - log(phi(U, m+1, r));
end

%% CalculateFuzzy Entropy of signal U
function [res] = phi(U, m, r)
    N = size(U,2); %Length time series.
    X(m, N) = 0;
    X0(N) = 0;

    %Calculate X0
    for i = 1:N-m+1
        X0(i) = mean(U(i:i+m-1));
    end

    %Phase space reconstruction
    for i = 1:N-m+1
        X(:,i) = U(i:i+m-1) - X0(i);
    end

    %Calculate distance between vectors Xi^m and Xj^m
    d(N-m+1, N-m+1) = 0;
    for i = 1:size(d,1)
        for j = 1:size(d,2)
```

```
            if i ~= j
                d(i,j) = max(abs(X(:,i) - X(:,j)));
            end
        end
    end

    n = 2;
    D = exp(-(d).^n/r);

    res = sum(D, "all")/((N-m-1)*(N-m));
end
```