FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Guided Data Cleansing of Large Connectivity Matrices

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Medizinische Informatik

eingereicht von

Florence Gutekunst
Matrikelnummer 01637640

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Mitwirkung: Dipl.-Math. Dr. Katja Bühler
Professor Guillaume Beslon

Wien, 29. Jänner 2019

Florence Gutekunst

Eduard Gröller

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Guided Data Cleansing of Large Connectivity Matrices

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Medical Informatics

by

Florence Gutekunst
Registration Number 01637640

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Assistance: Dipl.-Math. Dr. Katja Bühler
Professor Guillaume Beslon

Vienna, 29th January, 2019

Florence Gutekunst	Eduard Gröller

Erklärung zur Verfassung der Arbeit

Florence Gutekunst
Schäffergasse 2, 1040 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 29. Jänner 2019

Florence Gutekunst

Acknowledgements

First of all, I would like to sincerely thank Katja Bühler for giving me the opportunity to work on this fascinating topic, for her important scientific support and for guiding me during the whole thesis, especially in the design and method choices. I also express my gratitude to Eduard Gröller who has taught me scientific rigor, for his support and for orienting me from the beginning to the end. I would also like to thank Florian Ganglberger for his availability and for his very precious help during the whole thesis, in all important implementation choices especially.

My thanks are also addressed to Guillaume Beslon, who supervised me from France, for his availability and for everything he taught me at my French university (INSA Lyon). In addition, I want to thank my colleagues from VRVIS for the interesting discussions and their precious help. I would like to thank in particular Florian Schulze, Nicolas Swoboda and Markus Töpfer, team members of the Biomedical Visualization Group at VRVis, for their encouragement, for their support, feedback and suggestions, as well as for teaching me ReactJS.

To conclude, I would like to dedicate this thesis to my parents, to my sisters, to my great roomate Jelena and to Sébastien. I will never thank them enough for the support they provided me.

Kurzfassung

Die Untersuchung des Gehirns ist ein wichtiges Ziel in den Neurowissenschaften und der Psychiatrie. Die Biomedical Image Analysis Group am VRVis hat mit der Wulf Haubensak Group am Institute of Molecular Medicine ein Framework entwickelt, um Gehirn-Daten zu untersuchen. Gehirn-Daten können als Konnektivitätsmatrizen gespeichern werden. Diese sind aber sehr groß und enthalten Rauschen. Das Ziel dieser Diplomarbeit ist die Säuberung von großen Konnektivitätsmatrizen. Zu diesem Zweck wird die Reduzierung von Rauschen sowohl das Zusammenführen ähnlichen Zeilen und Spalten auf eine kleine Matrix mit Hilfe einer Visualisierungsfunktion vorgeführt. Diese beinhaltet ein visuelles und evaluirendes Feedback der Operationen, so dass die wichtigsten Informationen nicht während des Prozesses gelöscht werden. Die resultierende Matrix besieht sich auf eine zufällige Stichprobenentnahme aus den anatomischer Gehirn-Hierarchien. Dieses Werkzeug is ein Schritt in der Kette zur Verarbeitung von Konnektivitätmatrizen.

Abstract

Understanding the organization principle of the brain and its function is a continuing quest in neuroscience and psychiatry. Thus, understanding how the brain works, how it is functionally, structurally correlated as well as how the genes are expressed within the brain is one of the most important aims in neuroscience. The Biomedical Image Analysis Group at VRVis developed with the Wulf Haubensak Group at the Institute of Molecular Medicine an interactive framework that allows the real time exploration of large brain connectivity networks on multiple scales. The networks, represented as connectivity matrices, can be up to hundreds of gigabytes, and are too large to hold in current machines' memory. Moreover, these connectivity matrices are redundant and noisy. A cleansing step to threshold noisy connections and group together similar rows and columns can decrease the required size and thus ease the computations in order to mine the matrices. However, the choice of a good threshold and similarity value is not a trivial task. This document presents a visual guided cleansing tool. The sampling is based on random sampling within the anatomical brain hierarchies on a user-defined global hierarchical level and sampling size ratio. This tool will be a step in the connectivity matrices preprocessing pipeline.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Understanding the organization principle of the brain and its function is a continuing quest in neuroscience and psychiatry. In the past decades, progress in medicine and medical imaging techniques enabled procedures with ever increasing resolution to image the brain anatomical circuitry. This progress also enabled the identification of brain regions correlated to physiologically-based signals, such as fear. This encouraged the definition of transcriptional similarities networks for genes that may have impacts on the brain anatomy and the behaviour. Indeed, one aim in neuroscience and psychiatry is to understand how the genes and the brain structure can impact brain anatomy and behaviour [GKP+17]. It has also been shown that the circuitry of the nervous system is linked to genes [LHA+07], thus the fusion of these different connectivities is of high interest in order to better understand how the brain works.

Therefore, exploring how the brain is anatomically, structurally and functionally interconnected is a crucial task. In the last years, brain initiatives such as the Human Brain Project [hum] or the Allen Institute [OHN+14] have released multimodal neurobiological data with ever increasing resolution that can be used to create very large network graphs.

1.2 Problem Statement

The Biomedical Image Analysis Group at VRVis developed together with the Wulf Haubensak Group from the Institute of Molecular Medicine at the Research Institute of Molecular Pathology (IMP) an interactive framework that can perform a real time exploration of large brain connectivity networks on multiple scales. The networks, represented as connectivity matrices, can be up to hundreds of gigabytes. They are too large to hold in current machines' memory. This is especially the case if one wants to

fuse different connectivity matrices to better understand the underlying processes, i.e., on anatomical, structural, and functional scales. The underlying technology harnesses spatial organization, sparsity, and correlation of the data to overcome these obstacles with a highly optimized connectivity index. However, the matrices are still too large to be easily mined.

These networks, that are up to several gigabytes, contain noisy and redundant data. A cleansing of these networks is thus required in order to be able to fit them in current machines' memory. Operations could then be performed on them in order to infer new knowledge in neurosciences. The goal of this thesis is thus to realise a software prototype for visually guided cleansing connectivity matrices in order to reduce the size of the initial matrix while keeping the most important information.

1.3 Aim of the Work

The goal of this masters thesis is to develop a tool in order to cleanse large connectivity matrices. However, the user does not want to operate blindly. As a result, a visual tool is also required. Different cleansing operations can be applied to the initial large matrix. However, the parameters of these operations are data-dependant. In order to find good cleansing operation parameters, these matrices should first be sampled to relevant data. The reduced data (the sample), should remain representative of the initial data, so that the patterns and features found in the sample also exist and are important in the original data. As reading connectivity matrices can be long, the sampling step should be performed with a one-pass algorithm approach and the sampled version of the matrix should fit in a standard computer's RAM.

This sampled matrix should then be visualised in a web application as well as relevant information such as network measures about the sampled and the initial matrix. These measures can help the user to check that the most important information of the initial data are not lost. The user can then employ this web application tool in order to estimate cleansing parameters to apply to the initial matrix later on. According to Sporns [Spo16], the presence or absence of connections and the estimation of their strength are subject to noise, statistical biases, and observational errors. It is thus relevant to cleanse these matrices, both for removing noisy connections, but also to gain storage place and increase the computation efficiency of algorithms on the connectivity matrices. The visual feedback can help the user to check that the cleansing filters applied did not change the data "too much".

Thus, the main contribution of this thesis is the development of a visualisation prototype tool. This tool will first sample a connectivity matrix and apply cleansing filters on it (namely thresholding and row and column merging by similarity). Through the visual tool, scientists can visualise how the data is modified in order to find the right parameters for the later cleansing of the whole matrix. Finally, the users can apply the selected filters on the initial matrix and get the final cleansed matrix.

1.4 Methodological Approach

First of all, a literature review had to be performed in order to become familiar with the topic. This review also aimed at suggesting an appropriate way to sample the connectivity matrices as well as visualise the sampled matrices.

Then, one-pass sampling algorithms had to be engineered, requiring a literature review about sampling in big data, as well as their mathematical foundations. The appropriate implementation tools for the back-end had to be chosen and the back-end developed and tested. An evaluation of the sampling methods was then necessary in order to decide on the best sampling approach.

A literature review about visualisation for connectivity matrices as well as large matrices was then performed before creating a mockup for the visualisation application. Then, this mockup was developed as a web application.

Finally, the evaluation of all steps had to be realised. The sampled matrix, the sampled matrix after cleansing, and the initial matrix after cleaning had to be evaluated in order to define the appropriate sampling rates without loosing too much "information", and to evaluate how the tool performed.

1.5 Structure of the Work

This thesis is structured as follows.

Chapter 2 gathers the state of the art in terms of neuroanatomical concepts (Section 2.1), connections between brain connectomics and graph theory (Section 2.2), sampling strategies for large datasets (Section 2.3)), as well as visualisation of large connectomic matrices and more generally of large matrices (Section 2.4). See Section 2.1 to become familiar with the important knowledge about the neuroscience fields relevant for this thesis. The Section 2.2 will provide the reader with knowledge about the graph theory and how it is related to brain connectomics. It is especially relevant for the later evaluation. The Section 2.3 describes several sampling strategies for large datasets and is thus relevant for the development of a sampling strategy. Finally, Section 2.4 presents the visualisation of large matrices, and is thus relevant for the development of the visualisation tool.

Chapter 3 describes methodological aspects of the current work. The specifications of the project are first gathered, see Section 3.1. Then, the developed concepts (sampling algorithms, visualization, see Section 3.2), the programming language choices (see Section 3.3) and the data (see Section 3.4) are detailed.

Chapter 4 details the implementation choices and strategies for the concepts that were developed in the previous chapter. The implementation of the sampling algorithm is detailed in Section 4.1, as well as the development of all concepts useful for the visualization in Section 4.2. Finally the implementation of the cleaning operation is characterized in Section 4.3.

Chapter 5 reports the different evaluation steps and analyses them. It suggests interpretations of the results to answer the research question.

Finally, Chapter 6 summarises the present work. A discussion of how this work can be extended concludes the thesis. Supplementary figures can be found in Appendix A.

CHAPTER 2

State of the Art

Over the last decades, large brain datasets have become available, such as the ones provided by the Allen Institute [OHN+14] or the Human Brain Project [hum]. These datasets represent underlying neuroanatomical connections and can thus help to explain how the different mechanisms of the brain work. They are also very large. Indeed, humans have around 100 billions neurons, and there are dozens of different regions within the brain that are connected together.

While these brain connections were more and more studied by the scientific community, another field, neuroinformatics, also emerged. Neuroinformatics provides tools for large brain data storage, management, mining, analysis, and visualization. Thus, scientists can get insights into fundamental properties of neuronal ensembles [XYJ+15, KS05]. Because of the large size of the underlying data, neuroinformatics is one typical application of big data mining tools and algorithms. It is also related to graph theory, since the brain can be described as a complex network of nodes and edges. For example, on the microscale, nodes can be neurons, and edges can be the synaptic connections between these neurons.

Thus, this chapter provides an overview of the state of the art in neuroinformatics, regarding the current research topic. The underlying neuroanatomical processes will first be described in order to get general insights on the meaning of the data used during the thesis. Then, graph theory will be related to the brain connections. Indeed, connectivity matrices represent graphs. The review of graph theory related to brain connectivity aims at understanding the actual data the developed tool is provided with. These first two important topics were mainly studied in order to provide strategies in the sampling step. Indeed, as can be read in the following section, where a review of some big data processes is presented, sampling the data, especially in a one pass approach, is a hard task. Once the data is sampled, it should be visualized. Therefore, a review of the connectivity analysis visualisation is also presented. Finally, limitations and challenges are described and a comparison and summary of the existing approaches are presented.

2.1 Underlying Neuroanatomical Concepts

This section defines the different scales for analysis of neuroanatomy, as well as the different kinds of connectivity that are of use in neuroscience and psychiatry. The main purpose of this section is to give the reader more insights about the meaning of the data that is being used during the thesis.

An important concept in neuroscience is the connectome. A connectome is a comprehensive map of neural connections in the brain, and may be thought of as its "wiring diagram". Characterising and sketching the connectome is thus of high interest in neuroscience.

Here, two main aspects of the underlying neuroanatomical concepts will be presented. First of all, neuroscience can be performed on different scales. The different scales as well as their main characteristics will first be described. There also exist different connectivity types. These types will then be characterized.

2.1.1 The Different Scales in Neuroscience

Neuroscience can be performed on different scales. All of these scales can be perceived as points of view regarding neuroscience. The topological scale ranges from single nodes to the whole network. The spatial scale can gather individual neurons to brain regions. Finally, the time scale can range from milliseconds to the evolutionary process of species [BB17].

The spatial scale itself can be divided into three main scales, namely microscale, mesoscale, and macroscale. These are respectively also known as cellular, tissue, and whole brain scales. The nanoscale (molecules and synapses scale, also known as subcellular scale) can also be of use in neuroscience. Moreover, it is of interest to map different spatial scales to understand how the properties of one spatial scale are related to the properties of another one[BB17, Spo16].

The nanoscale The nanoscale describes the subcellular level, and typically gathers gene expression data, also known as transcriptomics or RNA-sequencing data. This data quantifies messenger RNA in a biological sample, and can help to characterize a brain cell. In order to elaborate a molecular profile of a cell, its messenger RNA needs to be measured, to determine which genes have discriminative power for this particular cell. Two main methods coexist for this purpose, namely *in situ hybridization*, which allows scientists to determine which cells within the brain express one particular gene. This allows researchers to get localization properties of the cells expressing the gene of interest, but then the co-expression with another gene is unknown. On the other hand, *single cell analysis* provides information about co-expression of genes, but does not provide "spatial information". Indeed, in *single cell analysis*, we are interested in a single cell, i.e., which other cells express the same genes.

The microscale The microscale describes the cell level, i.e., the level of individual neurons, and can be imaged using automated histology methods (such as electron microscopic reconstruction or light microscopy) or through cytoarchitectonics, myeloarchitectonics, chemoarchitectonics, etc [OHN+14, Spo16]. While MRI is often used to image the macroscale and the mesoscale, its resolution is not good enough yet to reach the cell or synapse level [BB17]. It thus aims at describing connections between individual neurons. Before describing further this scale, some basics of neuroanatomy will follow.

The mammalian central nervous system (CNS) is composed of two large categories of cells, namely the neurons and the glia cells. These two categories are further subdivided into a large variety of cell types, with unique morphology, connectivity, physiology, and function [LHA+07]. We will only make a broad overview of the main types of cells.

Neurons are the basic cells in the nervous system and are surrounded by glia cells, who provide the structure and some insulation. There exists different types of glia cells with specific aims: the ependymal cells separate the brain from other organs, astroglia provide nutrients and energy to the neurons, oligodentrocytes provide the myelination (sheets around the axon that helps faster signal transmission), and microglia are immune cells to protect the neurons.

Neurons are composed of several parts. First of all, their cell body contains several parts, namely the nucleus, the ribosom, the Golgi apparatus and endoplasmic reticulum and mitochondria. The nucleus contains genetic material and where messenger RNA is synthesized. Ribosomes synthesize proteins based on messenger RNA. The Golgi apparatus and endoplasmic reticulum transport and modify proteins. Finally, mitochondria synthesize and generate energy for the cell.

In order to send or collect information, extensions are attached to this cell body. The dendrites are the receptor canals, whereas the axon is the canal through which the information is sent. Two neurons can thus be connected if the axon from one neuron is linked to the dendrites of the other one. This process is made through synaptic terminals (synapses). The function of any neuron is thus also linked to its input and output neurons, as well as how it connects to these other neurons [Spo16]. Most of the connections from a given neuron are with neighbouring neurons. As such, neurons often form groups of highly interconnected neurons. The connections from one group of neurons to another group of neurons is called circuits or pathways [KS05]. A projection is another term to describe the connection of a neuron to another one.

Different types of neurons exist, who differ according to the number and size of dendrites and axons. Figure 2.1 describes the organisation of a neuron. Neurons can be characterized using different criteria, namely morphology, connectivity, physiology, molecular profile, functional response properties, electrophysiological properties, position within the brain, etc.

Thus, microconnectomics focuses on the single cell network level [Spo16]. However, this level has several characteristics that make it rather difficult to sketch the connectome. First of all, the microscale is characterized by an important synaptic and extrasynaptic plasticity.

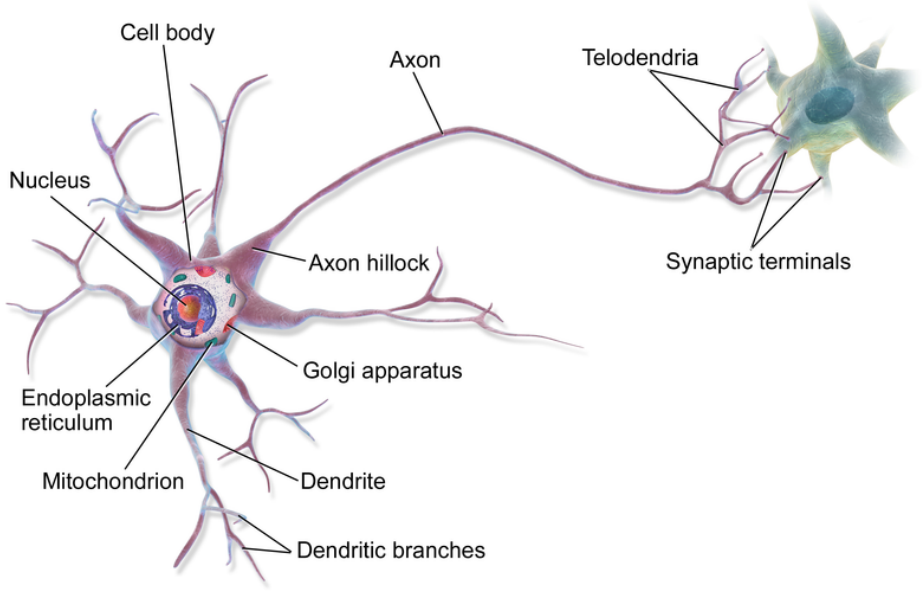


Figure 2.1: Schema of a multipolar neuron, from Bruce Blaus [Bla]

Indeed, neuron connections are constantly changing over time, as new connections are created when we learn and others are deleted when the neurons are not enough stimulated. The high variability of cells and their large number (around 20 billions neurons in the brain) are also intrinsic characteristics of the microscale. Moreover, getting a correct image of all the neurons and their connections would be a very time-consuming process. As such, the datasets on the microscale are only composed of a fraction of the neurons. Cunningham and Yu also add that some phenomena cannot be fully explained only on the single neuron basis [CY14].

The mesoscale On the mesoscale, both long-range and local connections are studied, i.e., the focus is on neuronal populations. These can be imaged using neuroanatomical tracers [OHN+14]. On the mesoscale, the connectome will not be characterized in terms of local and global properties, but rather in terms of differently sized clusters of neurons, their configurations and the association with functional or behavioural characteristics [BB17, Spo16]. This scale thus focus on neuronal populations [RRG+09] and on areas and inter-areal projections (i.e., inter-areal connections) [Spo16].

The macroscale The macroscale describes long-range region to region connections and pathways [OHN+14, YAC+15]. The focus is thus on inter-areal or large-scale projections on anatomically distinct brain regions. Analyses of the human cortex enabled the identification of 52 areas [Spo16]. The macroscale has been widely studied in the literature, and enabled the description of central lobes, surface landmarks, and white matter tracts [Spo16]. Neuronal popupaltions are defined as a composition of highly

similar and connected elements. However, brain areas and neuronal populations are difficult to delineate, and there is no consensus on the parcellation scheme for human brain regions. This causes problems when creating the connectome [STK05]. Moreover, the brain areas will be represented differently if we consider the functional or the structural areas. Figure 2.2 depicts the functional areas of the human brain, which can then be further divided into subregions.

The delineation could be strictly structural (e.g., anatomical), or also functional. Indeed, while there is most of the time a functional connection between structurally connected regions, it may not always be the case. Sporns et al. [STK05] add that the macroscale may provide several hundred brain regions and thousands of pathways. However the macroscale does not incorporate information on subdivisions and subcircuits within each of the aforementioned regions.

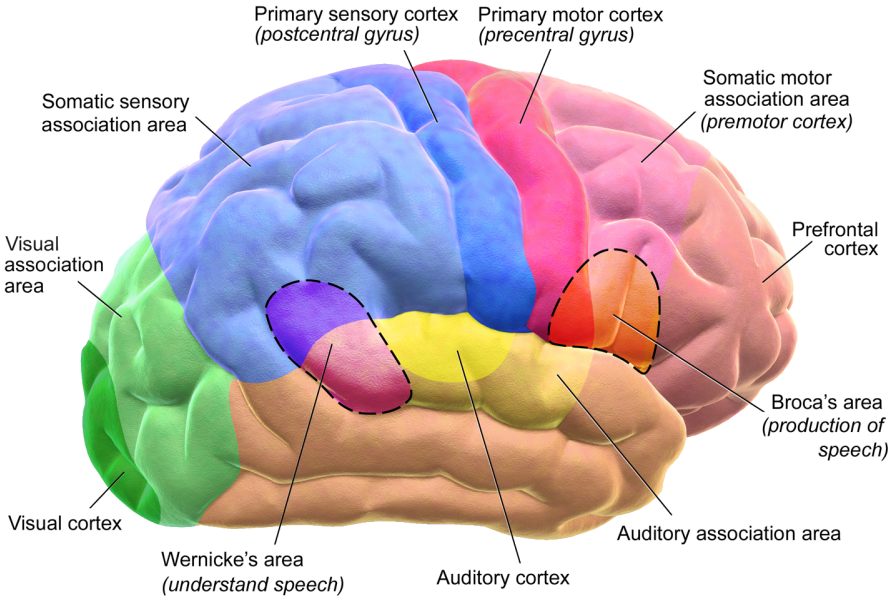


Figure 2.2: The functional areas of the human brain, from Wikipedia [bra]

There are different processes in order to acquire macroscale brain images. Diffusion-tensor imaging (DTI), functional Magnetic Resonance Imaging (fMRI, with Blood Oxygen Level Dependent (BOLD) signal), Diffusion-weighted Magnetic Resonance Imaging (dMRI), proton density MRI, and blood flow imaging are valuable non invasive imaging techniques and are mainly used in macroscale brain acquisition. As these are non-invasive techniques, the axons and fiber bundles are indirectly estimated and as such, they may be difficult to interpret quantitatively and are error-prone [BS12, HMB$^+$, KS05]. Indeed, MRI images return observations on a voxel scale, which may be noisy, according to the resolution of the imaging device.

It is also possible to use post-mortem dissection, as well as tract tracing. In recent years, other invasive approaches emerged, such as histological staining after injection of tracers

(such as fluorescent dye or a virus) into specific brain regions [HMB+]. These approaches also involve the sacrifice of the animal to see how the tracer material spreads through the tissue, along the axons [HMB+]. Contrary to non invasive techniques, tracer techniques are very precise and accurate, as they can determine whether two regions are effectively connected (which may not be ensured by non invasive techniques) as well as the direction of the connection. They require the sacrifice of the subject, which can not be performed for human brain imaging, and only a few connections can be tested on a single animal [BS12, HMB+].

Diffusion weighted imaging

Figure 2.3 describes the different scales of interest for neurology. Figure 2.4 from Betzel and Bassett [BB17] describes the different scales within the brain. The microscale we described corresponds to the synapses and neuron scales. The mesoscale gathers the network and maps scales. The macroscale is composed of the maps and systems scales. CNS (central nervous system) is the highest scale of the figure and it comprises the brain and spinal chord. In mammalian species, the mesoscale and macroscale maps are preferred to describe brain connectivity, as they allow the expert to establish relations between connectivity, functions, and behaviour [Spo16].

2.1.2 The Different Kinds of Connectivities

As previously mentioned, there are different scales. As a result, different kinds of connectivities are of interest in the neurosciences. The two most important kinds of connectivities are the structural (or anatomical) and the functional connectivity. It has been shown in numerous studies, such as in the work of Hermundstad et al. [HBB+13], that these two types of connectivities are significantly related. Structural connectivity may shape functional connectivity, however, a pure anatomical description is not enough to fully explain perception or behaviour [Spo16]. Indeed, there are regions with strong functional connectivity but with weak structural connectivity. This can be either explained by imaging limitations (e.g, inaccurate tractography of the fibers due to noise in MRI) or by pathways that have strong indirect (polysynaptic) connections [Spo16].

It is of high interest in the neurosciences and psychiatry to discover relations between genes, brain circuitry, and behaviour. This information can then help to better understand some psychiatric pathologies such as Alzheimer, or simply to better understand the brain and how the different physiological features of connections impact neuronal interactions and neuronal circuit computation [Bar12, Spo16]. Thus, the fusion of these different connectivities is of special interest in the neurosciences. Ganglberger et al. [GKP+17] proposed an algorithm for fusing sets of functionally related genes with connectomes and gene expression maps, in order to predict neuroanatomical maps of multigenic functions. As a result, researchers get a better understanding of the underlying brain function or behaviour and this can also help to identify the underlying function-specific brain circuitry. They used the Allen Mouse Brain Atlas Gene Expression and Connectivity Data framework [all]. They tested their method on several gene sets, for which the

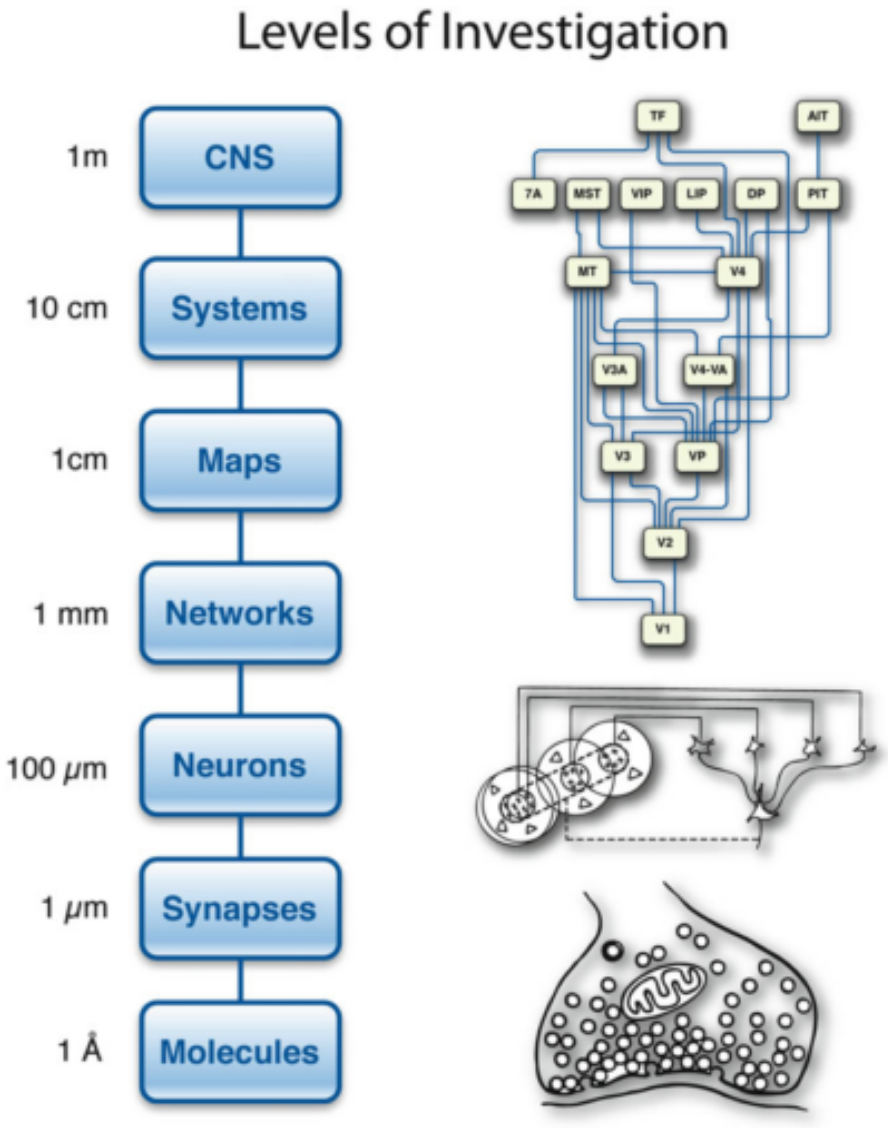


Figure 2.3: The different levels of investigation from Kennedy et al. [Spo16]

functional associations and neuroanatomy were already established. However, the fusion of the different connectivities is challenging. First of all, it requires the registration from one connectivity type to another one, and as such it requires annotations, especially if one wants to register data from different scales with each other [Spo16].

Structural Connectivity

The structural connectivity describes how regions are anatomically (physically) connected [BB17, Spo16]. They represent synaptic connections between neuronal elements and

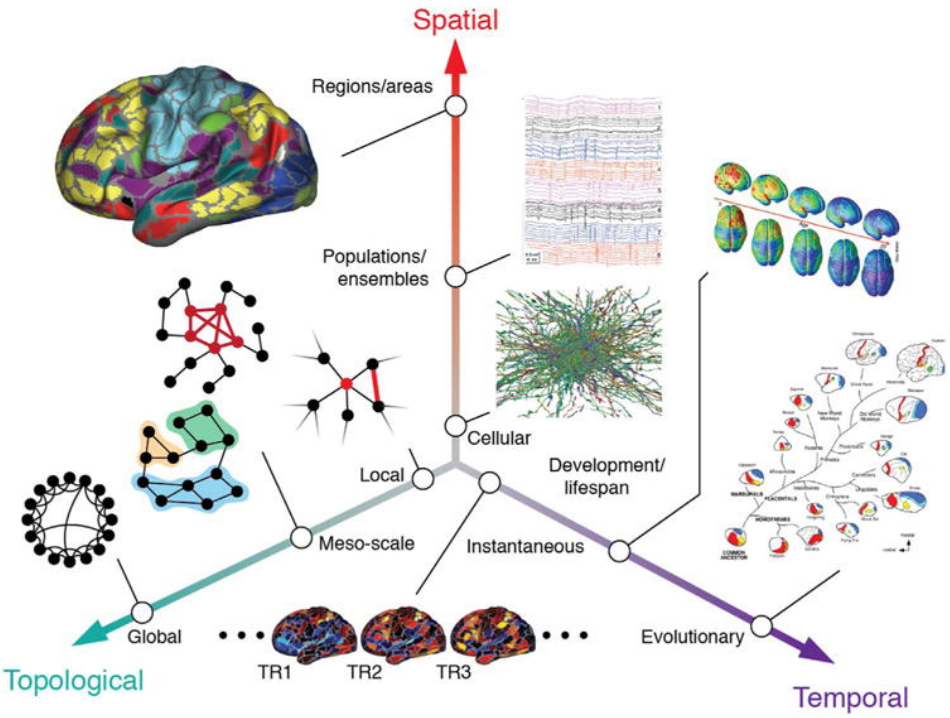


Figure 2.4: The different scales of the brain, from Betzel and Bassett [BB17]

allow researchers to infer knowledge on how these regions are physically connected. The strength, the size, density, number of synapses, or coherence of the connection are typical examples of interesting connection characteristics [Spo16, RS10]. On the macroscale and mesoscale, connections are often imaged with MRI and then they typically reflect the different characteristics of the reconstructed white-matter fiber tracts [BB17, RS10].

Structural connectivity can also be imaged through axonal tracing. A tracer dye is injected in a neuronal brain area. It is then taken up by the local membrane receptors of the local neurons. Finally, it is automatically transported by an axon protein machinery to various projection targets of that neuronal area. Then, the subject has to be sacrified and the neural tissue can be sliced up. This allows researchers to describe how the neuronal area projects within the brain, based on the spread of dyes. The approach leads to high resolution information, but is a very invasive technique that requires the sacrifice of the subject [Kai11].

Diffusion-weighted MRI (dMRI) is an important imaging modality for determining structural connectivity in the white matter. It uses tractography algorithms and cannot give any information about specific axons. Moreover, it can be difficult to disantangle fibers coming from very close regions.

A characteristic of structural connectivity networks on the macroscale is that they are often sparse. Most potential connections do not exist, and the regions tend to be highly

connected to neighbouring regions. These networks are also relatively stable across time [Spo16].

Structural connectivity is often expressed as a square matrix. There, the weights in the cells can indicate the number of fibers between brain regions (or neurons, depending on the scale of interest), the degree of myelination, the probability that a region (respectively neuron) can be reached by another region, and so on [Kai11].

Effective Connectivity

Effective connectivity is a connectivity measure that is related to the underlying mechanisms (direct or indirect causes) for correlated activity and the influence of one region on another one [Spo16, RS10]. Indeed, the real importance of a connection does not only depend on anatomical or functional connectivity weights. It can also be influenced by a variety of other factors, such as the neurotransmitters, the excitation/inhibition balance, whether the connection is feedforward or feedback, and so on [Spo16]. As a result, the weights in the effective connectivity matrix usually represent the causal relationships between the different nodes [Kai11].

Functional Connectivity

Functional connectivity models how different regions of the brain work together [Spo16]. For a given task, the activation patterns of different brain regions are analysed and compared in order to discover corresponding firing patterns across different regions. Functional connectivity requires a functional connectivity mapping, which can be performed either using Resting State or Meta-Analytic Connectivity Modeling. Resting State networks focus on the coupling of brain regions in the absence of a task, whereas Meta-Analytic Connectivity Modeling is task dependent. The combination of the two can lead to inferences about which regions are actually taking part in the studied mechanism or task (for instance moving a finger). Thus, it relies on the strength of the statistical relationship between regions' or neurons' activities, quantified through the covariance, the cross-correlation, the Fisher-transformed correlation coefficient [ZFB12], or other coherence measures [Spo16, BB17, Kai11]. While there is most of the time a good correlation between functional connectivity regions and strong close connections [MERG+14], functional connections may also occur between anatomically unconnected regions. Functional connectivity networks are therefore likely to be denser than structural ones [RS10, DG09].

Functional connectivity is divided into functional specialisation and functional integration. The former considers how large regions are individually engaged in a specific functional context, for instance how is one region individually responsible for moving a finger. On the other hand, functional integration is concerned with how the different regions relate to each other in order to produce coherent behaviours [BS12].

Genetic Connectivity

Genetic studies have demonstrated that genes play an important role in our brains. First of all, it has been shown that several characteristics of the brain structure are heritable, such as the total amount of grey or white matter, the volume of the ventricles, the size of the hippocampus, and so on [Spo16]. Several projects, such as the ENIGMA project (Enhancing Neuroimaging Genetics through Meta-Analysis), focus on how variants in the genetic code may influence the brain [Spo16]. The Allen Brain Atlas project focuses on how the genes influence the structural and cellular architecture of the mouse brain. This project uses an atlas containing expression patterns of around 20.000 genes in the adult mouse brain [LHA+07]. Gene expressions have been found to be linked to several personality traits, to cognition processes and also to risks for neurological or psychiatric diseases [Spo16].

Even if it seems rather difficult due to the differences in terms of scale, Stein et al. [SHL+10] proposed a method to link voxels from the brain to genomics. This requires to perform more than one billion statistical tests, thus one has to find sampling methods to ease this computation. Moreover, this also requires comprehensive gene correlation atlases [LHA+07]. It has also been shown that functionally related genes are not distributed at random within the brain and have a rather specific structure [GKP+17].

2.1.3 Summary of the Underlying Neuroscience Concepts

The concept of connectome can be defined as a "comprehensive structural description of the network of elements and connections" of a given nervous system [Spo16, OHN+14, STK05]. It aims at providing spatial and strength characteristics of the connections between areas [Spo16]. The connectome faces several challenges. First, relating brain connectivity networks from different scales (scale challenges) is not a trivial task. Then, connections evolve, both on a small time window on the synaptic plasticity level, and on the evaluation level (time challenges). Last but not least, inter-individual differences is another tough task, since it requires the mapping to atlases, and defining a correct atlas on the first place is not a trivial task [Spo16].

The brain connectivity data have some specific characteristics. First of all, they are subject to noise, statistical biases, and observational error [Spo16]. For macroscale or mesoscale data, errors can be due to inaccurate mappings, to the resolution and noise level of the imaging device, or also to inaccurate intersubject registrations to an atlas. Inaccurate mapping often occurs because of bleeding or signals from draining veins during the imaging procedure [Spo16].

Another important characteristic is that the data is large, since humans have around 100 billion neurons, and each neuron contacts more than 1.000 others on average. The data is also rather sparse, especially when it is evaluated at fine-grain levels. There is also a rather high intra-region connectivity: around 80 % of the inputs for one node come from local circuits [MMF+11]. Indeed, the strongest connections are always the ones with the nearest neighbours. It has been observed that the neuronal projection strength

is exponentially related to distance [Spo16]. The different pathways within the brain cover five orders of magnitude in connection strength [MMF+11, Spo16, KERKT16]. As a result, high clustering and short path length are important characteristics of such data [Spo16, BS09].

The brain connectivity data also appears to be characterized at multiple physiological and anatomical levels by a log normal distribution [Spo16, MERG+14]. A log normal law (or log normal distribution) is a continuous probability distribution of a random variable whose logarithm is normally distributed. The variability of the brain connectivity log normal distribution can be described by a negative binomial distribution [MERG+14].

These concepts are of interest for the current topic since they describe on a high level the meaning of the data that is being used during the thesis. Indeed, the datasets that were used are on different connectivity levels and cover different scales.

2.2 Brain and Graph Theory

Studies of the brain connectivity and graph and network science were developed in parallel. Since the mid-1990 a lot of transdisciplinary approaches combine these two scientific domains.

First of all, a graph is a mathematical representation of interactions between a set of objects (nodes), whose interactions are materialized with connections (edges). Visual representations of graphs are common, where nodes are drawn as points and edges as lines connecting the nodes [Spo16]. There is an evident link with brain connectivity data. The nodes are the neurons, or regions, or other node structures depending on the scale. The connectivities may be symbolized through the edges. The pairwise relationships are then summarized in a connectivity matrix and represent the network's topology [Spo16].

Moreover, graph and network sciences provide tools and methods in order to characterize, model, and analyse complex networks. It has been shown that connectivities from different types and scales had features of complex networks, such as small-world topology, highly connected hubs and modularity, rich club organizations, and so on [BB17, Spo16, BS09]. Graph models applied to the neurosciences allow researchers to test structure and function hypotheses, and also to explore the brain topology, organization, and complex dynamics [BS09, ZFB10, HSC+09, SCKH04].

Connectivity matrices are being used for this thesis, which are adjacency matrices that represent the underlying brain network on a given scale. The main goal of this section is to understand the actual data that is being used, as well as to help to define sampling strategies. Another goal is to define graph measures of interest to display in the visual tool. These measures can then provide the user a quantitative evaluation of the cleansing tools parameters that are tested. Some measures will also be used in the evaluation of the prototype.

Network sciences provide network measurements that help to get insights into the network's function and highlight important nodes and other features that characterize

a network such as its robustness or vulnerability [BB17]. These quantities range from measurements at the individual node scale to measurements at the whole network scale. Measurements of individual network elements reflect how these elements are embedded in the network, whereas measurements at the network scale provide a global description of the network [RS10].

Figure 2.5 describes the most important measurements of network topology.

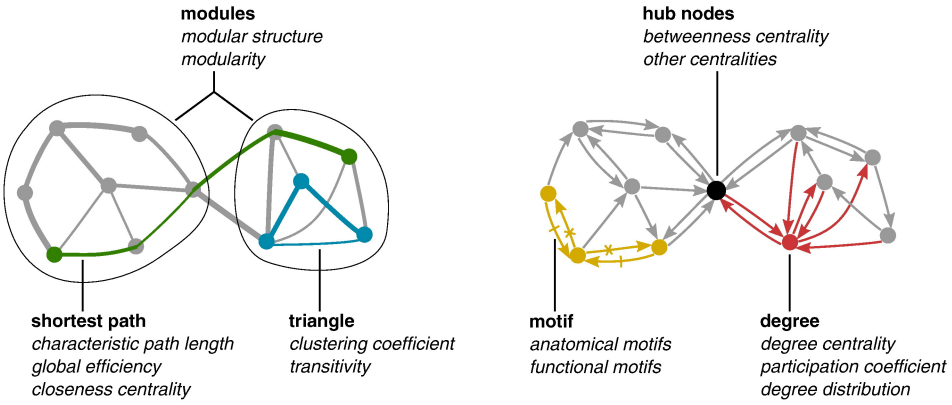


Figure 2.5: Measures of network topologies, from Rubinov and Sporns [RS10]

There are different types of networks. The most important ones are: random networks, scale-free networks, small-world networks and modular networks. In random networks, all connections are equally probable, i.e., the degree distribution is Gaussian and symmetrically centred.

In a Scale Free Graph, nodes follow a longtail or heavy tail distribution, which is well approximated by a power law. The human brain connectivity networks do not usually follow a pure powerlaw, which has only been reported by analyses on the voxel level. They usually follow an exponentially truncated law, which is associated with a lower probability of very high degree nodes [BS09, Spo16, SWM05]. An example of a small-world network can be the links between users of a social network. Most nodes are not direct neighbours, but the neighbours of a node are likely to be neighbours of each other. Moreover, most nodes can be reached from any node by a small number of steps [sma]. Modular networks are characterized by a high modularity, i.e., they have dense connections between the nodes within modules but sparse connections between nodes in different modules.

Node Degree and Degree Distribution

The node degree is the number of edges for a given node [BS09, BB17, TB13, RS10]. If the graph is weighted (i.e, if the edges have a value symbolizing the strength of the connection between two nodes), it is also referred as to node strength and aggregates the weights of its edges [RS10, BS09]. If the graph is directed, there is a distinction between the incoming (in-degree) and out-going (out-degree) links to the node [RS10, LF06].

The degree distribution is defined by the degrees of all the nodes in the network [RS10, BS09]. Brain connectivity networks, and more generally most graphs describing real-world networks are complex networks. They usually deviate from a Gaussian degree distribution. Indeed, in real-world networks, most nodes have a rather small degree, while a few of them have a high degree (hub), and thus are closer to a power-law distribution. In biological systems, the degree exponent often ranges between 2 and 3 [BS09].

The node degree can be be seen as a measure of the influence of a given node and it also represents the node's number of neighbours. High degree nodes will have the greatest influence on the network and are called hubs [BB17]. Most important nodes are either integrator or distributor nodes. Integrator nodes are nodes whose connections are mostly incoming ones, while distributor nodes are nodes whose connections are mostly out-going ones. Moreover, the ratio between the in- and out-degree of a node is also of interest and can give information about its function, about integrator nodes or about distributor nodes [Kai11].

2.2.1 Influence Measures - Rich Club

Measures of influence aim at quantifying the importance of network elements (nodes or edges) for the global functionality of the network [Spo16].

Centrality, Hubs and Robustness

Hubs are nodes with high degrees [BS09] and usually high centrality. Centrality measures how important a node is in a given network. Freeman [Fre78] defined three different measures for network centrality: degree, closeness and betweenness. As previously described, the degree is the number of nodes that are connected to the given node. Closeness is defined as the inverse average shortest path length from one node to the others. It thus describes how quickly the node can access the others. Finally, betweenness describes how often a node is part of the path from one node to another one, i.e., it measures the proportion of shortest paths passing through the node with respect to all shortest paths. As shortest paths are preferred, the given node will have an important role [BS09, TB13, XYJ+15, Spo16].

Hubs are nodes that are often of special interest in network studies, because they often have high centrality and are thus important for information integration, while being also vulnerable to attacks [Spo16]. The importance of a given node can then be assessed by deleting it and estimating how the information spreads in the residual network [BS09]. There are different kinds of hubs. Provincial hubs are hubs connecting nodes within a given community, whereas connector hubs interconnect different communities [Spo16].

Density - Sparsity

Network density describes the number of potential connections in a network that are actual connections. It is thus the ratio between the number of edges in the network

and the maximum number of edges it could have (where all nodes would be linked) [Spo16, BS09]. A network is then sparse if its density is very low [Spo16]. The mean network degree can also be used as a measure of density [RS10].

Rich-Club Organizations

Rich clubs are groups of hubs (high degree nodes) that are densely interconnected to one another. They thus often have an important role in a network. They can be detected by calculating a so-called rich-club coefficient $\phi(k)$, which measures the density among nodes with degree higher or equal to k, and compares the coefficient with the one from a random network. If the network indeed contains rich-clubs, then its rich-club coefficient will significantly deviate from the one of a random network [BB17].

Core-periphery structures are related to rich-club organizations and consist of a dense cohesive core and a sparsely connected periphery, whose nodes sparsely interact with one another. The cores thus play an important role within the network to link different regions and exchange information [BB17].

A scale free graph architecture is characterized by a small number of hubs, while other nodes are less densely connected, but these networks do not necessarily contain rich-clubs if the hubs are separated by lower degree nodes [Spo16].

Van den Heuvel and Sporns [vdHS11] stated that several brain regions have the rich-club property. The bilateral precuneus, the superior frontal cortex, the superior parietal cortex, the hippocampus, the putamen, and the thalamus are the main brain regions that have this property. These regions are thus very important in the brain functioning, as a very high percentage of the shortest paths from the brain passes through the rich club [Spo16]. Indeed, removal of these regions could cause severe neuropathologies [vdHS11, YAC$^+$15, TB13]. Figure 2.6, from the publication of van den Heuvel and Sporns [vdHS11] summarizes these findings.

There are also other graph measures linked to linear algebra and spectral properties. Indeed, graphs can also be represented by their adjacency matrix. It has also been shown that the spectral properties of graphs often follow a heavy-tail distribution [LF06].

2.2.2 Segregation Measures - Community Detection

In graphs, segregation measures how the nodes aggregate into distinct communities, and can be expressed through a clustering coefficient or modularity [Spo16].

Clustering Coefficient, Modularity and Motifs

In graph theory, a cluster or module describes a group of nodes that are directly connected to each other. Thus, the clustering coefficient measures the fraction of the number of edges of a node with his neighbours over the maximum number of edges that could theoretically exist between these nodes [BS09, XYJ$^+$15, LF06, WS98]. The modularity

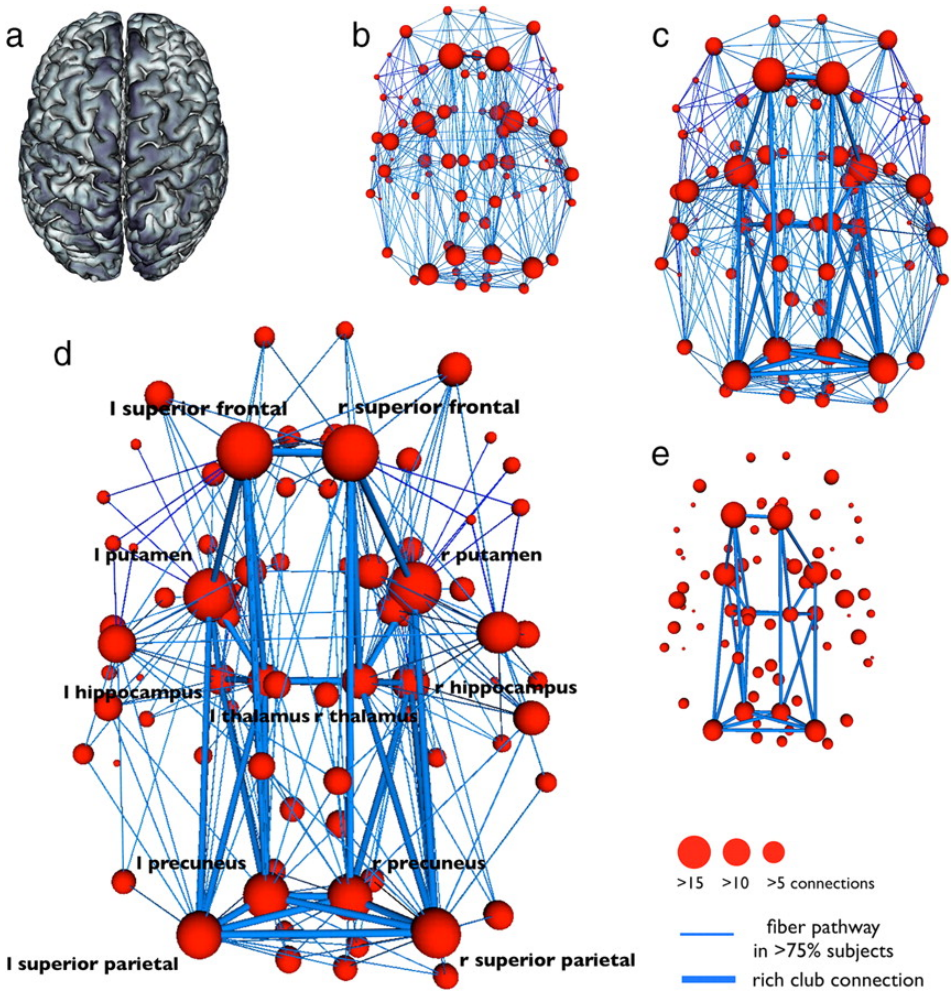


Figure 2.6: Rich-Club Organisation within the human brain from van den Heuvel and Sporns [vdHS11], (a): anatomical brain representation; (b): group averaged connectome; (c): group averaged with Rich-Club connections; (d): group averaged with Rich-Club connections connected to other regions; (e): Rich-Club

also measures the strength of the division of the network into clusters (or modules). Networks with high modularity have dense connections between nodes in clusters and sparse connections with nodes from different clusters [RS10].

Random networks usually have a low average clustering coefficient, whereas complex networks are often characterized by a high clustering coefficient [BS09]. Various algorithms allow scientists to measure the modularity of a network, most of the time based on hierarchical clustering [BS09, GN02].

The fraction of triangle structures around a node also has a role in the clustering coefficient. It corresponds to the number of neighbours that are neighbours of each other. The more

triangles, the higher the clustering coefficient will be [RS10].

Community Detection

Community detection within graphs is of high interest. Indeed, first of all, it provides some characteristics of the network. The network can also be split into subgraphs, where the similarity within the subgraph is high (communities). This is particularly relevant in neurosciences in order to detect brain areas or brain neurons (depending on the scale) that are similar in some ways, for instance to detect sets of neurons that fire for the same task. It can also help in dimensionality reduction by grouping nodes together. The most often used method for community detection consists in finding the best set of subgraphs that maximize the modularity of the network, thus groups of nodes with similar connectivity patterns [BB17, HMB$^+$, OW14, RS10].

A well-known hypothesis about the brain is that it is organized into hierarchical communities, where each community found at a given scale can be refined into smaller communities on a lower scale [BB17]. Indeed, community detection can occur at different scales. Most community-detection studies of the brain have focused on communities at the same scale [PCN$^+$11, GSM$^+$15], or were based on heuristics if dealing with different scales [PCN$^+$11], which can lead to suboptimal solutions [BB17]. Some studies, however, use a multi-scale modularity maximization by tuning the resolution parameter, in order to obtain estimates of the community structure over multiple scales [GSM$^+$15, BMP$^+$17, BB17]. An optimization function that takes into account the resolution can thus be $Q(\gamma) = \sum_{ij}(A_{ij} - \gamma P_{ij}).\delta(\sigma_i, \sigma_j)$, where δ is the Kronecker delta function (equal to 1 if both arguments are the same and 0 otherwise), A_{ij} is the observed connectivity between nodes i and j, P_{ij} is the expected connectivity under a null model, σ_i indicates to which community the node i is assigned and γ is a parameter for resolution tuning [BB17]. By varying the resolution parameter, one can perform multi-scale community detection. It can be difficult to choose a good resolution parameter, because some communities will be found at every scale, but they may not be of interest and may also not be representative. The question remains how to define whether a found community is of interest.

Some approaches consider calculating a community contribution measure for all found communities, and for instance use the OSLOM algorithm [LRRF11]. This algorithm will first identify the worst node in a community, and then the community will be assigned a score, which is the probability to have a node within the community that is more connected within the community than it would be in a random network [BB17].

In order to detect whether a graph contains significant communities, it can also be fitted to a stochastic block model. A stochastic block model is a generative model for random graphs that tends to have communities. This model is characterized by a number of communities, but knowing the number of communities in the graph we want to test is not straightforward. Thus, a solution according to Olhede and Wolfe [OW14] can be to

fit the graph where we want to discover communities to a stochastic block model with a varying number of blocks.

The distribution of sizes of strongly connected components can be of interest in community detection. A component consists of a set of nodes where for each pair of nodes, there exists a directed path between them. On the other end of the spectrum of segregation measures lies the distribution of sizes of weakly connected components. These are sets of nodes where for every pair of nodes there exists an undirected path between them [LF06].

Another method explained in the book by Leskovec et al. [LRU14] to find communities is to use betweenness. The higher the betweenness value of a node (or of an edge) is, the most probable it lies between different communities. Thus communities could be extracted by removing the edges or points of highest betweenness.

2.2.3 Integration Measures

The integration describes how quickly or how easily information can travel along network paths, and thus describes how nodes can exchange information [Spo16, RS10]. Integration is mostly measured through path length and network efficiency.

Integration can also be applied to brain networks and then measures how fast brain regions can exchange information for a given task. As previously mentioned, neurons tend to be connected to their neighbours rather than to distant neurons, and this can be explained by the need to make the information travel faster.

Path Length and Efficiency

The path length is the minimum number of edges that must be traversed to go from one node to another one, in other words the number of hops from one node to another one [BS09, TB13, XYJ+15, Spo16]. On the network scale, the characteristic path length is defined by the average path length through the whole network [BB17, RS10, WS98].

Thus, in networks with short characteristic path length, nodes can be quickly reached, and thus the information can be quickly shared. Efficiency is inversely related to path length, so a network with a short characteristic path length will thus be efficient. The global efficiency is described as the average inverse of the shortest path length, while the local efficiency is the measure on the node level of how quickly the current node can reach its neighbours.

Another network measure linked to path length is the hop-plot. It is the number of reachable nodes under a given distance. In the case of unweighted graphs, the distance is the number of edges, whereas in weighted graphs it is the sum of weights to get from a node to another one [LF06].

Finally, the diameter of a graph is also linked to path length and defines the largest distance between any pair of vertices. Reachability is also linked to these notions and defines the ability to go from a node to another one in the graph.

2.2.4 Small-World Property

The small-world property defines a small-world network and combines high clustering among nodes of a network and network efficiency. It is defined as the ratio between the clustering coefficient and the path length [BS09, RS10, GZFA10, Spo16, HG08]. All nodes from the network are linked through relatively few intermediate steps. This has been described mainly in social networks [TM67]. The small-world property combines measures of segregation (high clustering) and integration (small average distance).

It has been shown that small-worldness is a characteristic of brain functional networks, but occurs also in genetics or many other natural networks [BS09, Spo16]. It rather characterises how the brain has high modularity for specialized computations, since the neighbouring neurons often have a similar function and are often connected. It also keeps an efficient communication across the brain to combine different subtasks [Spo16].

2.2.5 Graphs and Brains

In summary, it has been found that the human brain is a complex network that has specific topological properties such as centrality, a modular community structure (high clustering), which is hierarchically built, contains hubs and sparse connections, a short average path length (associated with a high efficiency, especially on the structural connectivity graph), high robustness, a small-world property, etc [BS09, ZFB10, BB17, MERG$^+$14]. The connection patterns are rather complex, because of the above properties and the large size of the network, thus a graph approach can help to explore the different brain properties [MERG$^+$14].

Moreover, Betzel et al. [BAKG$^+$16] evaluated different generative models for the human connectome. Their evaluation was based on the Kolgomorov-Smirnov statistics of the network's degree, clustering, betweenness centrality, and edge length distributions. They define edge length as the Euclidean length between the nodes, even if the fiber length would be more appropriate. They took the maximum value of the Kolgomorov-Smirnov statistics as final value. They found that the best result is a model based on homophilic attraction combined with geometry constraints. This model can reproduce degree, betweenness coefficient, edge length distribution, characteristic path length, mean clustering coefficient, global efficiency, modularity, propensity for high degree nodes to be connected via long distance edges as well as local node statistics (degree and clustering coefficient).

Table 2.2 summarizes all important network measures in the work from Rubinov and Sporns [RS10]. These measures are of interest mostly for the evaluation of the sampled matrix compared to the original one, as well as for the evaluation of the cleansing.

2.3 Sampling Methods

As previously stated, the brain connectivity matrices are very large, but tend to be sparse and noisy. It is important to define sampling methods in order to be able to infer

Metric name	Metric explanation	Biological significance	Measure type
Assortativity	Correlation coefficient between the degrees of all nodes on two opposite ends of a link (for a binary connectivity matrix)	Assortativity measures the tendency for one neuron to be connected to similar neurons in terms of connection strength. As a result, a positive assortativity coefficient will indicate that neurons tend to be connected with other neurons with similar connection strength, whereas a negative assortativity coefficient may indicate that there are some widely distributed neurons with high strength (hubs), which are then vulnerable.	influence
Betweenness centrality	Fraction of all shortest paths in the network that contain a given node	Betweenness centrality measures how important a node is in terms of integration. In natural settings, information tends to use the shortest path. As a result, if many shortest paths go through a neuron, it means this neuron has a key role in information exchange and is thus very vulnerable.	influence
Degree distribution (in and out)	Number of links connected to the node	Number of neurons one neuron projects to (out-going connectivity) and number of neurons one neuron gets projections from (in-coming connectivity). It basically measures the importance of a neuron in terms of number of connections with other neurons.	influence
Density	Fraction of present connections to possible connections (counts non-zero connections)	Neurons are connected in a particular way, and not all neurons are connected. Density thus evaluates the total number of connections compared to the number of possible connections. The denser the brain, the more connections between neurons there are.	influence

Edge betweenness centrality	Fraction of all shortest paths in the network that contain a given edge	Edge betweenness centrality measures how important a connectivity path between two neurons is in terms of integration. In natural settings, information tends to use the shortest path. As a result, if many shortest paths go through an edge (connectivity path between two neighbouring neurons), it means this edge has a key role in information exchange and is thus very vulnerable.	influence
Strength distribution (in and out)	Sum of weights of links connected to the node	The weight between neurons can represent different properties (the number of fibers between neurons, the degree of myelination, the probability that a neuron can be reached by another one...). It thus somehow describes the connection importance between neurons. The strength distribution thus measures the strength of the total out-going connection from one neuron to the others, and the strength of the in-coming connection to this neuron. It basically measures the importance of a neuron in terms of connection strength with other neurons.	influence
Strength ratio distribution	Ratio between in- and out- strength distributions	Ratio between in and out strength distributions. Some neurons are mostly "receivers", while others are "emitters". The strength ratio distribution evaluates this behaviour in order to define whether one neuron has more a receiver or emitter behaviour.	influence
Rich-club coefficient	Fraction of edges that connect nodes of degree k or higher out of the maximum number of edges that such nodes might share	Rich clubs are neurons of high connection strength (strength higher than a value of interest k) that are intensely connected to another. Several regions within the brain have a rich-club property (among which the bilateral precuneus, the superior frontal cortex, the superior parietal cortex, the hippocampus, the putamen and the thalamus). Rich-clubs can facilitate the transmission of information.	influence

		The rich-club coefficient measures how the neurons with highest strength are connected to each other. It will be higher if there are such rich-clubs, and lower in the other case.	
Clustering coefficient	Fraction of triangles around a node (in the case of a binary connectivity matrix) Average weight of triangles around a node (in the case of a weighted connectivity matrix)	The clustering coefficient computes the fraction of neurons among a neuron's neighbours that are also neighbours of each other. It thus measures the interconnection of the neighbourhood of a neuron. The higher the clustering coefficient, the more interconnected the neighbourhood of a neuron is. Moreover, high clustering is associated with robustness of a network. It indicates redundancy in the paths between neighbouring neurons, and information can go through different paths of about the same length to reach its destination, even if some paths are damaged.	segregation
Core structure	Partition of the network into two non-overlapping groups of nodes, a core group and a periphery group	Core-periphery structures assume that the brain consists of a dense cohesive core and a rarely connected periphery, whose neurons sparsely interact with one another. The cores play an important role in the network to link different regions and exchange information. This core-periphery structure can measure to which extent one neuron behaves more like a core (important information exchange role) or like a periphery. Anatomically central nodes (i.e., neurons or regions) often facilitate integration and enable links between anatomically unconnected regions.	segregation
Modularity (community structure)	Statistic that quantifies the degree to which the network may be subdivided into clearly delineated non-overlapping groups	The community structure aims at finding non-overlapping groups of neurons to maximize the within connectivity between neurons of the same community, and minimize the connectivity with neurons from other communities.	segregation

		The modularity measures the strength of the division of the brain network into communities (or modules, or clusters). Networks with high modularity have dense connections between neurons and sparse connections with neurons from different communities. It measures how "modular" the brain is, i.e., to which extent we can subdivise it into modules. It is also a well-known fact that the brain has a high modularity.	
Characteristic Path Length	Average shortest path length in the network	The characteristic path length measures the average of the shortest path between neurons. It describes how fast the information can travel across the brain from one neuron to another one.	integration
Diameter	Maximum eccentricity	The diameter describes the longest path length between two neurons. The smaller the diameter, the more efficient the brain network, i.e., the information will proceed quickly between far away neurons.	integration
Global efficiency	Average inverse shortest path length in the network	Global efficiency is the inverse of the characteristic path length. Indeed, the shorter the characteristic path length, the faster the information can travel across the brain and the more efficient is the brain network.	integration
Node eccentricity	Maximal shortest path length between a node and any other node	The node eccentricity describes the maximum path length between one neuron and all other neurons. The larger the node eccentricity, the longer it will take for the information to travel from the neuron of interest to another one in the worst case (i.e., if these neurons are far apart). It often means the node is at the periphery and far from the center.	integration

Radius	Minimum eccentricity	The radius describes the shortest of the maximum of all paths between two neurons. The larger the radius, the less efficient is the brain network. This also measures how long it will take for information to travel from a rather central neuron (i.e., a neuron that is close to all others) to other neurons in the periphery.	integration
Local efficiency	Global efficiency computed on the neighbourhood of the node	The local efficiency computes the inverse path length between a neuron and its neighbours. As a result, the closer the neighbouring neurons are (in terms of weights and distances), the more efficient the neuron is because the information can travel faster to the other neurons.	integration

Table 2.2: Most important network measures

knowledge from the data. There are different possible ways to sample the data. However, it is also of interest to keep the most important connections. We also need to keep a dataset that is still representative of the underlying data, i.e., preserve the structure of the dataset.

Different approaches coexist in the literature in order to sample large connectivity matrices. In this section, we will present approaches based on underlying neuroanatomical concepts, and also more general sampling approaches for dimensionality reduction and graph sampling. As the data is large, it is also linked with big data analytics, which offers different sampling methods, like the so-called streaming methods that aim at analysing the data in real-time on the fly. Sampling has to be performed carefully, since the result can become inaccurate especially when testing for power-law degree distributions, according to Stumpf et al. [SWM05].

As a result, this section aims at presenting different sampling approaches. One of these approaches will then be chosen for the sampling preprocessing step, and will be specialised and tuned to the current topic using the knowledge acquired in the review of the precedent sections.

2.3.1 Sampling Based on Underlying Neuroanatomical Concepts

Sampling large data can be a tedious task, especially if there is no a priori knowledge about the data. The data used for this thesis is rather specific as it represents brain connectivity. As such, it has some characteristics, as described in Section 2.2. These characteristics can be used in order to sample the graph.

Parcellation Methods

A characteristic of the brain is its hierarchical structure. A possibility to sample the brain is to group together nodes that are in the same hierarchical cluster [BJG+13, Spo16]. These are so-called parcellation methods. These methods aim at grouping nodes (or voxels when considering images) together into regions [BB17].

Parcellation methods need some criteria in order to group nodes together, and there are a plethora of different possible criteria, such as for instance spatial variation in functional connectivity, myelination, etc. The desired number of parcels, as well as the scale of study also play a role in the method and will have impacts on the resulting network's topology, and are not trivial to find [WWZ+09, BB17]. Multi-scale community detection is another possibility in order to parcellate the brain at different resolutions [BB17, BRNL+10]. Parcellation methods can also be based on dimensionality reduction, as presented by Roca et al. [RRG+09].

Definition of Sampling Rates

Brain connectivity matrices often contain noisy and missing data. A possible sampling method is to delete noisy connections. For instance, Oh et al. [OHN+14] applied several

sampling rates on their mouse connectome in order to delete noisy connectivities and thus keeping mostly true positives. They found that true positive values predominantly fell within the range of $[10^{-4}; 10^{5}]$.

Oh et al. estimated in their mesoscale mouse brain connectome an upper bound for sparsity of 36 % for the entire brain and 52 % for cortico-cortical connections, and a lower bound of 13 % and 32 % respectively. They found out that the distribution of the strength in their connectome fits a lognormal distribution. In fact, the log distribution of the whole brain strength is even better approximated by a two-component Gaussian Mixture Model. They also performed network analyses, such as clustering coefficient measurements and degree distribution determination and found out that neither small-world nor scale-free topologies well fitted the data..

Markov et al. [MERG+14] also stated that Fraction of Labeled Neurons (FLNe) from the strength distribution of their mesoscale monkey brain connectome was approximated by a lognormal distribution, and that the cortical matrix structure was rather dense (66 %). They defined a connection to be moderate if its FLNe value ranges between 10^{-4} and 10^{-2}, the ones with values under 10^{-4} being described as sparse, while the ones whose value exceeds 10^{-2} define strong projections. They showed that the standard deviation and mean of the FLNe was approximated by a negative binomial distribution with a dispersion parameter of 5.0. The sampling rate clearly depends on the data and is not trivial to define ad hoc.

2.3.2 Dimensionality Reduction

The data used in the thesis has a high dimensionality, since it consists of connectivity matrices. These are indeed adjacency matrices with many nodes. As a result, the data is high dimensional. This subsection presents dimensionality reduction methods. This type of methods can produce low-dimensional representations of high-dimensional data, where the prominent features are kept.

Roca et al. [RRG+09] proposed a tractography-based parcellation method for the cortex, relying on a dimensionality reduction of the connectivity matrix based on spatial information, without relying on apriori knowledge about the cortex structure. First of all, they randomly assigned all neuronal nodes to a Voronoi patch, and computed the associated connectivity matrix. They then performed dimensionality reduction to this connectivity matrix and applied a k-medoid clustering and iterated on all these steps until there were no parcels added to the final parcellation (there were at most 5 iterations). The dimensionality reduction consists here in computing the density of connections to the given patch and to compute the watershed to split the cortex into catchment basins (areas that are connected to the patch) leading to the patch. The basins are then pruned to keep only the most significant basins, and the connectivity matrix is updated accordingly.

Most dimensionality reduction methods (Principal Component Analysis, Singular Value Decomposition, Multi-Dimensional Scaling, Factor Analysis or Eigenvalue/Eigenvectors

Decomposition) are costly and slow to perform on big data. Multi-Dimensional Scaling aims at finding a set of low-dimensional coordinates that best preserves pairwise distances in the original high-dimensional space. This provides a visual representation of similarities within a dataset for any pairwise dissimilarity, and reconstructs a map that preserves distances. Principal Component Analysis is a statistical method to emphasize variation and detect strong patterns along so-called principal directions. It thus outputs an ordered set of orthogonal directions that captures the greatest variance in the data. Moreover, classical linear techniques may not be very effective for large connectivity matrices. Indeed, neural networks represent rather nonlinear data, thus linear methods may fail to realize a good dimensionality reduction [YAC+15, MORG91].

Ye et al. [YAC+15] proposed methods to construct the geometry of the brain using dimensionality reduction techniques. They evaluated some dimensionality reduction techniques and applied them to the human brain structural connectome. Among them, they evaluated Multi-Dimensional Scaling, which is a classic linear embedding technique, Isomap, and t-distributed stochastic neighbourhood embedding. They found that the structural connectome could not be well explained by a linear embedding technique, and that the t-distributed stochastic neighbourhood embedding also led to poor results. They found that Isomap was the most appropriate dimensionality reduction technique in this case. In order to test their node deletion method and study if this dimensionality technique still captures meaningful information, they studied the impact of node removal by rich-club removal and also by random deletion of 21.5 % of the nodes according to various connectome measures: descending nodal strength, ascending clustering, ascending path length, descending betweenness centrality, and embeddedness. Apparently, removing nodes with the lowest 21.5 % of nodal clustering coefficients minimally impacts the structural connectome's intrinsic geometry. They also found out that highly embedded regions had an important role in the structural connectome.

Cunningham and Yu [CY14] declared that dimensionality reduction is well-suited for analysing mesoscale neuronal activity. In order to be able to generate hypotheses about the activity of single neurons or a population, an exploratory data analysis step is required, and often involves the visualization of a large amount of data. The aim of dimensionality reduction is to be able to explain the D measured variables by K variables that capture the underlying distribution, with $K << D$. These K variables are also called latent variables (not directly observed variables). They reviewed the dimensionality reduction methods that were mostly used in neuroinformatics and provided guidelines on how to select one. Among them are basic covariance methods such as Principal Component Analysis, or Factor Analysis. Time series methods were also presented like Hidden Markov Models, Gaussian Process Factor Analysis, or Latent Dynamical Systems. Finally, they reviewed methods with dependent variables such as Linear Discriminant Analysis, or mixed dimensionality reduction methods such as a variant of linear regression, as well as non-linear methods like Isomap and Locally Linear Embedding.

2.3.3 Sampling and Summarization of Large Data

Nowadays, companies and scientists have to deal with large amounts of data. In data mining and data-analysis, they try to find interesting patterns and features in the data. However, most algorithms that can find patterns and features cannot scale well. The data needs to be reduced in the first place. The reduced data (the sample), should remain representative of the initial data, so that the patterns and features found in the sample also exist and are important in the original data. Summarization techniques (also known as sketching techniques) aim at providing a summary or synopsis of the data and are particularly relevant.

Hesabi et al. [HTG+15] provided a complete review about the different data summarization techniques. They divide the techniques into six different categories, namely clustering, sampling, compression, histograms, wavelets, and micro-clustering, and review how and when these techniques are applicable. Other categories such as hash-based or spectral-sparsification based methods are also present in the literature. These categories will be described and their applicability for large connectivity matrices will be analysed in the next subsections. We will also introduce the notion of streaming and of streaming algorithms first, and elaborate on how and why these streaming techniques are relevant in the context of sampling large connectivity matrices.

Streaming

In the big-data era, massive amounts of data are continually arriving at a fast pace and need to be processed and analysed. Due to computer memory limitations among others, only a small fraction of the data can be stored at the same time in the RAM [BMKK14, LRU14]. A model emerged, considering datasets as streams, where the information is only accessible once, and is then lost if it is not stored or processed. Streaming algorithms aim at analysing data in real-time (in one-pass if possible), relying on the fact that it is more useful to have a fast and approximate solution than an exact solution [LRU14]. Among them, streaming sampling algorithms are of particular interest for this thesis. We first want to sample very large connectivity matrices. A streaming model can be applied to represent the data, as we are reading it sequentially, and a sampling streaming algorithm is of great interest.

The performance of a streaming sampling algorithm is one of its key features. It can be measured in terms of number of passes over the data stream, the memory required by the algorithm and its internal structures, the running time of the algorithm, as well as the approximation ratio (i.e., how well the algorithm approximates the data stream) [Mir17].

There are different streaming sampling possibilities described in the literature. The most obvious one is random sampling, which can lead to surprisingly good results. There are also specific so-called sketching streaming techniques. These algorithms aim at building a summary or synopsis of the data stream (to have a representative sample), while using only a small amount of memory [BMKK14, JP17, TCM16]. A challenge in sketching methods is to keep a high accuracy and a good approximation of the data. They should

optimize a representativeness function [BMKK14]. Two other measures are of importance in data streaming algorithms, namely the time required to add an element in the sketch (update time), as well as the reconstruction time (time to produce an approximate summary, also known as query time).

Sliding windows is another streaming possibility. It consists in performing a detailed analysis of the last items and of the summary of all other items. According to Joshi and Patel [JP17], these sliding windows techniques have several advantages: they are well-defined, easy to understand, as well as deterministic. As such, they have been broadly studied in the streaming literature. These techniques are particularly used when the data change over time. For instance, in the web case, if every IP address is a node and every request an edge, it can be of interest to determine how these requests evolve over time, in order to find frequent patterns. It can then happen, and actually rather often does, that the same request will be performed several times. With these sliding windows methods, only the last requests are in the sliding window while the others are in the summary. It is well adapted for dynamically changing structures. However, it requires to keep values in memory, as well as to scan the summary in order to know whether an incoming value should be stored or not. A possible solution is to keep statistics of the summary, and recompute these statistics only once a whole window has been summarized.

Clustering

As previously stated, a cluster gathers elements that have a high similarity, while having a low similarity with other elements from different clusters. These methods are unsupervised and there are different similarity metrics to gather points within the same cluster. Distance (Minkowski, Manhattan, Euclidean, Mahalanobis) is the most commonly used metric in clustering. It can be difficult to apply distance measures to non-numerical data, thus other metrics also exist. In our case the matrix contains numerical data, thus distance measures can be applied. According to Hesabi et al. [HTG+15], clustering algorithms can be subdivided into four main types, namely hierarchical, density-based, partitioning and grid-based clustering algorithms.

Hierarchical clustering, also known as connectivity clustering, creates clusters in the form of a tree, where each level of the tree can cluster the data into bigger groups. The BIRCH [ZRL96], CURE (Clustering Using REpresentatives) [GRS98], and ROCK (RObust Clustering using linKs) [GRS00] algorithms are the three main hierarchical clustering algorithms that are applicable to big data. The BIRCH algorithm cannot be applied to clustering of arbitrarily shaped groups, it is thus not applicable for high dimensional data such as our connectivity matrices. ROCK is similar to CURE but designed for categorical data. These two approaches are not that relevant for large connectivity matrices sampling.

CURE is based on random sampling and partitioning and comprises six steps. First, samples are randomly chosen. These samples are divided into a fixed number of partitions and these partitions are hierarchically clustered. The outliers are removed and the

clusters are refined. An important aspect here is that the clusters are not defined only by a centroid, but rather by a set of representatives, and as such CURE can find clusters of any shape. Finally, the samples are assigned to the closest cluster by computing the distance to the representatives of all clusters. CURE is robust against outliers and can deal with any cluster shape, which may be of interest. It runs with an $O(N^2_{sample} log(N_{sample}) + N_{sample} k)$ time complexity, k being the number of final clusters, which cannot be evaluated in advance, the last term is due to the labelling task. It has a space complexity of $O(N_{sample})$ [HTG+15].

Partitioning algorithms divide the data set into k partitions by maximizing an error function. The most well-known algorithm in this context is k-means [For65]. CLARA (Clustering LARge Applications) [KR90] and its improved version CLARANS (Clustering Large Applications based upon Randomized Search) [NH94] are other important partitioning algorithms that can deal with big data [HTG+15]. The time complexity of CLARANS is quadratic, and the time complexity for CLARA is in $O(k(40+k)^2+k(N-k))$. Although these algorithms require a predefined number of clusters, and we do not know how many representative clusters there are in our connectivity matrices, these approaches can still partition our matrices by keeping somehow representative items (columns or rows) as the clusters' centres. The algorithm should not be based on k-means, which computes the mean for the cluster's center, but rather on k-median, in order to keep a real column (respectively row) of the matrix.

Density-based algorithms rely on the density of the input space. Then, dense areas can be kept while sparse areas are discarded and classified as noise. Because of the network-topology characteristics that are assumed for connectivity matrices (such as small-worldness, high clustering, and presence of hubs), this could be a good strategy in order to sample such matrices. The main algorithms are DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [EKSX96], DBCLASD (Distribution Based Clustering of Large Spatial Databases) [XEKS98], GDBSCAN (Generalized Density Based Spatial Clustering of Applications with Noise) [SEKX98], DENCLUE (DENsity-based CLUstEring) [HK98] or OPTICS (Ordering Points To Identify the Clustering Structure) [ABKS99]. These algorithms run in at least $O(NlogN)$ time [HTG+15], which is not suitable in our case, for time complexity is an important factor.

Grid-based clustering methods first partition the dataset into cells, based on a grid-structure, and then sort the cells according to their densities in order to identify clusters. STING (STatistical INformation Grid-based clustering) [WYM97], CLIQUE (CLustering In QUEst)[AGGR98], GRIDCLUS [Sch96], Wave Cluster [SCZ00], FC (Fractal Clustering) [BC00] and OptiGrid[HK99] are common grid-based clustering methods in the big-data analysis. FC for example has a computation time of $O(N)$. A drawback of these clustering methods lies in the fact that the cluster shapes are limited to unions of cells [HTG+15], and the cells may not contain similar data, especially for connectivity matrices.

Micro-clustering techniques can be applied to streaming data and provide a real time summarization of these streams. An early work on micro-clustering is described by Aggarwal et al. [APHW03], who developed an algorithm called CluStream, which runs

in two phases. An online phase gathers summary statistics about the streaming data, and an offline phase performs clustering on these summary statistics. In the online phase, Aggarwal et al. use a pyramidal time frame approach in order to store the statistics as micro-clusters: they store the sum and sum of square of times stamps, the sum and sum of square of the data values as well as the number of values within the cluster. Aggarwal et al. use a k-means clustering on the micro-clusters in the offline phase. There are also variants, using the same micro-clustering approach at first, and then applying other clustering algorithms on the micro-clusters, such as Denstream (a density-based approach) [CEQZ06].

Compression

Compression allows scientists to represent the data in a compact way in order to save space. Compression methods can be either lossless or lossy. Lossless compression consists in removing statistical redundancies, thus the original data can be fully retrieved after decompression. Lossy compression throws away part of the data, which can never be fully reconstructed afterwards [HTG+15]. The main aim in compression is thus to identify similar patterns and then reduce redundancies, so that the size is reduced while keeping a representative version of the data.

Compression is based on the so-called MDL (Minimum Description Length), where all redundancies have been removed. This kind of method is particularly well adapted to datasets containing a lot of redundancies, such as web queries for example.

The purpose in our case is not to compress the data, because it then needs to be processed. As such, data compression does not seem to be the appropriate sampling solution.

Wavelets and other Signal processing-based Techniques

Signal processing methods have been broadly applied for compression and feature extraction, for instance through Fast Fourier Transforms or Discrete Wavelet Transforms [PBF16]. In this section, we focus on wavelets, because they gather both time and frequency information, and can decompose and store a signal more efficiently than with a Fourier Transform [HTG+15].

Wavelet methods decompose the signal into wavelet coefficients on a pyramidal basis, but most of these coefficients are close to zero. Only the largest ones are of interest in order to get an approximated reconstruction of the initial signal. The wavelet decomposition can be a very efficient lossy compression method, while keeping the most important information and features of the signal. Wavelet decomposition can be seen as the decomposition of the signal into a coarse approximation with details at different scales. The Discrete Wavelet Transform decomposes a signal into its low and high frequency parts by applying highpass and lowpass filters on different scales. Wavelets are produced from a "mother wavelet", which then defines the wavelet family. The most notable ones are Haar wavelets and Daubechies wavelets. Haar wavelets have been broadly studied in the literature, because of their simplicity, and have been applied to summary construction

[GH05]. The main purpose of Haar wavelets is to minimize the mean square error, while other families of wavelets can be applied to minimize other error types [HTG+15].

Wavelets are both time and frequency representations and have been proven to provide a good representation of discrete signals [GH05, HTG+15, PBF16]. Wavelet decomposition is often used in order to get a summary of the data. This technique has been widely applied in image and signal processing, leading to good compression results. As a result, the use of wavelets for data summarization of large datasets or streams was rather straightforward, and several streaming approaches are based on wavelets.

Most of the previous body of work focused on Euclidean errors ($L2$ distance), thus Guha and Harb [GH05] proposed a one pass streaming sampling algorithm based on wavelets for Non-Euclidean error measures in $O(N)$ and space in $O(B + log(N))$ with B being the maximum number of coefficients stored, using Haar wavelets. The computation time is actually in $O(|R|^2 NB)$ with R being a subset where the estimated error would be close to the optimum.

Cormode et al. [CGS06] proposed a streaming method for an approximated wavelet decomposition. It relies on a structure called Group Count Sketch, stored in polylogarithmic size, with a logarithmic update time, and a polylogarithmic query time to compute the top wavelet coefficients.

Papadimitriou et al. [PBF16] proposed the AWSOM method (Arbitrary Window Stream mOdeling Method) to automatically, effectively, and efficiently discover patterns and trends in streaming data in constant time using logarithmic space, without any prior knowledge. The method captures and identifies periodic components, identifies noise, performs forecasts, while being unsupervised and using limited space in one pass. It relies on Daubechies-6 wavelets. For each level, they performed auto-regression on the wavelet representation and expressed the wavelet coefficients at each level as a function of the previous coefficients of the same level. This auto-regression is based on applying the Fisher test using the Square Sum of Residuals error function. This determines whether the reduction in error achieved by adding a new parameter to the model is statistically significant. Only the really significant wavelet coefficients are kept.

The same as for compression schemes, wavelet-based data can be difficult to process. Moreover, the mathematical meaning of such an operation on connectivity matrices is not trivial and there is no guarantee that zero values lie next to each other.

Histograms

Histograms are structures that gather data into intervals (called buckets) and count the number of values that are mapped into these intervals. They represent data in a concise manner [JP17, HTG+15]. Different histogram techniques exists. Equi-sum are histograms whose buckets have the same length, but are not suitable for skewed data, as in our case. Equal-depth histograms are histograms where the frequency of the items for the different buckets are similar. The computation of the buckets' boundaries is computationally intensive.

V-optimal histograms aim at finding the optimal set of buckets so that the variance inside each bucket is minimized. Variants are the End-biased V-optimal histograms, where the value with the highest frequency is placed into its own bucket. Even if these histograms represent the data well, they are difficult to update and as such may not scale well to large datasets. MaxDiff histograms or Spline histograms are other types of histograms, and there exist also multi-dimensional histogram variants [HTG+15], but none of these algorithms is really applicable in our situation.

Hash-based Sampling

Hashing is a further possibility in streaming and offers various algorithms to sample data. The key idea behind the hashing methods is to rely on a hash-function, that will map the values into a smaller space. Cormode and Duffield [CD14] provided an overview of the hashing methods in 2014. For instance, the bottom-k algorithm hashes each key and keeps only the keys with the smallest hash values. Applied in our case the keys could be the ids of the row and of the column, or the value of the cell in the matrix.

Another possibility described by Leskovec et al. [LRU14] is to hash the key and check if the hashed value is already present in the summary. If it is present then the value is added, otherwise the value is added with some probability. It is also possible to use multiple hash functions and create several small summaries. This keeps more information while still having a limited storage space. This is the idea behind the sampling strategy presented by Tang et al. [TCM16], where different hash functions are applied to the data so that less information is lost.

Leskovec et al. [LRU14] proposed other hashing approaches, such as minhash or Locality-Sensitive Hashing. Locality-Sensitive Hashing hashes similar rows or columns together based on some distance measure. It is then easier to find similarities. The purpose here is to hash together (i.e., group together) similar rows or columns so that to decrease the size. The hash function should be chosen so that, if vectors $\vec{p}$ and $\vec{q}$ are close (i.e., $d(\vec{p}, \vec{q}) \leq D$ with D being a distance threshold), then, the probability that their hash values will be equal is below a probability threshold P_1 (i.e., $Pr[h(p) = h(q)] \leq P_1$), and if their distance is higher, the probability that they will be mapped together is below a probability threshold P_2 (i.e., $Pr[h(p) = h(q)] \leq P_2$), with $P_1 \geq P_2$.

A good choice of the hashing function is of particular importance in order to have a good final sampling, as well as the choice of an appropriate key, i.e., which attributes should be chosen to be part of the key. Indeed, as the hashing space is smaller, care has to be taken so that to avoid too many collisions, i.e., key values map to the same hash value. Hash functions can be based on random projections, using for instance cryptographic hash functions such as SHA-1, but these functions may be too complex, as we do not necessarily need the power of cryptographic security. They can also be based on heuristic hash functions such as rand, but it may not be random enough. Finally they can be based on mathematical hash functions such as universal hashing, which have precise mathematical properties and can be implemented in such a way that it is fast [CD14].

Probability Sampling

Probability sampling is a broadly used technique in the literature in order to get a concise representation of the data. According to Hesabi et al. [HTG+15], there are four main probability sampling categories, namely simple random techniques, systematic sampling methods, stratified sampling, and clustering sampling.

Random sampling randomly takes elements from the total population. It is very easy to perform and does not require any a priori knowledge. Random sampling with reservoir is a common random sampling streaming strategy. It consists of keeping a reservoir of size m and filling it with the first m elements. If the $m + 1^{\text{th}}$ element arrives, it will be added to the reservoir at a random position with a given probability [HTG+15].

Systematic sampling simply computes intervals of a given size and takes one value in every interval. This approach may not be representative, as not all points have equal chances of being selected since their selection depends only on their position in the stream [HTG+15].

Stratified sampling takes samples from identifiable subgroups (strata). Even small groups can be represented. This sampling can be performed randomly. In an optimal selection, the number of samples per strata depends on the strata size. It is more complicated than simple random techniques and requires a good definition of a strata [HTG+15].

Another method called clustering sampling is also based on groups in the population. This method is different from the clustering methods presented in Subsection 2.3.3. Contrary to stratified random sampling, it takes a sample of whole clusters. It may not be very representative of the data, depending on how the clusters are grouped together and how the clusters are sampled [HTG+15]. Multi-stage sampling is commonly used in clustering sampling, where, instead of keeping the whole clusters, another sampling scheme is performed to take only a sample from each cluster.

Graph Sampling

Graph measures can be very interesting in neuroinformatics in order to infer new knowledge about the roles and interlinks between the different structures. The graphs are very large and these graph methods often become too costly to perform. This also applies to graph clustering and partitioning. These methods usually do not scale well for large graphs or matrices. Therefore the graph should be sampled [LF06].

Leskovec et al. [LF06] addresses four steps before performing graph or matrix sampling. First, a sampling method needs to be chosen. Graphs are particular structures and as such, some of the previously presented sampling methods may not work or have to be adapted. The sample size should also be defined as well as how to estimate if the proposed sampling is satisfying.

Different possibilities exist to sample large matrices of graphs. Some are random-based, some are hash-based, and other are based on spectral sparsification. We are interested

in so-called scale-down sampling, which consists in getting a representative subgraph of the current graph, i.e., summarizing the graph or matrix. On the other hand, a scale-up sampling will estimate the initial graph from a sampled one. Another approach exists, the back-in-time sampling, which consists in finding the most similar graph on a time perspective. We focus here on static graphs, thus these sampling methods are out of the scope of this thesis.

Some streaming sampling methods have been applied to graphs. The purpose is to find a subgraph fitting in sublinear space, while being linear in construction time with a constant edge update cost. It should also approximate well the initial large graph. Then, the graph is modelled in a streaming way: an ordered sequence of edges continuously arrives. These graph streaming models can be divided into two categories. The adjacency streaming model assumes the edges arrive in arbitrary order. In the incidence streaming model, all edges incident to a node arrive successively. This later model is particularly suited to our case, where we will sequentially read the matrix row or column wise.

Tang et al. [TCM16] proposed an approach related to the Count Min algorithm based on hash functions in order to sample a graph (either directed or undirected). It is a one pass approach, in linear construction time, constant update time and it fits in sublinear space, while keeping the connections of the original graph. The idea is to use a set of d pairwise independent hash functions h_i and store an adjacency matrix M_i for each of these. As the hash functions are different, the initial values can then be reconstructed if the hash functions are chosen wisely. As a result, the storage space is smaller and the reconstruction is possible. For each edge e of weight w (1 in an unweighted graph) between nodes x ans y, increase $M_i[h_i(x), h_i(y)]$ by w, for each $i \in [0, d-1]$. They suggest to keep a $d \times n$ hash table to store the correspondence between hash values and node indices. With this extension, the time complexity is in $O(n)$, the space complexity is in $O(d * (k^2 + n))$ with k being the width of the hash functions mapping and n being the number of nodes.

According to Leskovec and Faloutsos [LF06], there are three main groups of random graph sampling, which are adapted from random sampling. They can either be based on random node selection, on random edge selection, or on exploration techniques.

First, the sampling can be performed by random node selection (node-induced subgraph). This consists in sampling a set of nodes and collecting the edges linked to these nodes. There are three main variants. The first one is called Random Node Sampling and uniformly selects a set of nodes and the corresponding edges. This method has no bias towards high degree nodes and thus cannot retain a power-law node distribution. The second one is called Random PageRank Node Sampling, which uses the PageRank weight of a node as its probability of being selected. The third approach is called Random Degree Node Sampling, which uses the degree of the node as its probability of being selected. A drawback of the random node selection methods is that we cannot forecast the number of edges that will be sampled, while there are usually more edges than nodes in graphs [LF06, ANK14].

Another random graph sampling group is based on random edge selection (edge-induced subgraph). The first algorithm in this category is Random Edge Sampling, which uniformly picks edges and collects the associated nodes. Another possibility is to have a node and edge approach, for instance with the Random Node Edge Sampling method. It consists in uniformly selecting a node, and then uniformly selecting some of its edges. Another method, called Hybrid Sampling, combines these two methods: Random Node Edge Sampling will be performed with probability p, and Random Edge Sampling with probability 1-p ($p = 0.8$ was found to perform best). These random edge selection methods tend to lead to rather sparse graphs and thus do not retain community structures [LF06, ANK14].

The third group of random-based sampling methods consists in sampling by exploration (topology sampling). By Random Node Neighbour Sampling, a node will be uniformly selected, and all its neighbours will be taken as well. The Random Walk Sampling consists in selecting uniformly a node and performing a random walk starting at this node. At every step, it flies back to the starting node with a given probability (0.15 is a value commonly used in the literature). It selects another node to start again if the algorithm gets stuck, i.e., if after a long number of steps not enough nodes were visited. Leskovec and Faloutsos set the number of steps to be $100n$ with n being the number of nodes. Random Jump Sampling is a variant of the Random Walk Sampling. At every iteration it can jump to a new node with a given probability (a probability of 0.15 was found to perform well). This variant will not get stuck. Snowball Sampling is another variant of Random Walk Sampling and picks many initial nodes at initialization time [CD14]. The four previous methods are biased towards high degree nodes and densely connected parts of the graph. The Forest Fire Sampling starts by choosing uniformly a node, and randomly generates a percentage of its neighbours that will be visited, if they have not been visited yet. The same will be applied recursively to all selected nodes. Once a node has been visited, it can not be visited again [LF06].

Leskovec and Faloutsos [LF06] reviewed large graph sampling. They conclude that edge-induced methods do not perform well, while Random Node Sampling appears to perform well. The best performing methods are the ones based on exploration (Random Walk or Random Jump, as well as the Forest Fire Sampling algorithm with a burning probability of around 0.7), where keeping only 15 % of the graph is then usually enough to have a rather representative subgraph. Indeed, these methods are biased towards high degree nodes and the sampled graphs are connected, whereas the Random Node, Random Edge, or Random Node Edge Sampling methods cannot retain a power-law distribution, for they are not biased towards high degree nodes.

However, Xu et al. [XLYE14] nuanced this and argued that the Random Walk Sampling produced high correlated samples and thus may lead to a poor estimation accuracy, while the Random Jump Sampling can be very costly in terms of iterations if the nodes' ids are sparsely populated (i.e., the id range is bigger than the number of nodes). They proposed a hybrid graph sampling method between these two approaches to overcome the disadvantage of the Random Jump Sampling. These approaches may not be practically

applicable on very large matrices that can be read only once [CD14].

Explorative random procedures become too costly for large graphs, because they involve a random exploration of neighbours. They thus require several passes over the graph. Ahmed et al. [ANK14] proposed a two-passes approach in order to effectively sample large graphs, with the result being more representative than from simple random-based approaches. They also proposed a streaming version, requiring only one pass over the graph. Their two-pass algorithm is a combination of node and edge selection. In the first pass, they select uniformly a subset of nodes. In the second pass, they take all edges whose nodes are in the subset. It can combine properties of nodes and edges sampling and avoids some of their drawbacks. Ahmed et al. also proposed a one-pass algorithm called PIES, which is based on the reservoir sampling idea. They showed that PIES preserved many network statistics and sampled particularly well if the data is sparse. Ahmed et al. also suggested a modification of their algorithm, that will replace the node with the minimum degree with the new incoming node. Thus, the algorithm becomes biased towards high degree nodes, and this also preserves eigenvalues.

In a later publication, Ahmed et al. [ADNK14] presented a generic stream sampling framework for big-graph analytics called graph Sample and Hold (gSH). For any incoming edge, if at least one of its nodes are kept, it will also be kept with probability q. If none of its nodes are kept, the incoming edge will also be kept with probability p. The values p and q have an impact on the number of edges that are sampled, and should be appropriately chosen, depending on the problem. The authors suggest to use small p and q values (i.e, below 0.008), leading to a number of edges in the sample between 0.5 % and 2.5 % of the initial number of edges.

Column Subset Selection and Graph Sparsification

Graph sparsification is another possibility, where the graph will be approximated by a sparse graph (where the number of edges can reasonably be seen as proportional to its number of nodes) with respect to some metrics [TCM16, BSST13]. The idea behind these approaches is to keep the same number of nodes, while changing the weights of the edges. Setting a weight of an edge to zero would thus mean that there is actually no edge. This can be particularly relevant for connectivity matrices.

Spielman and Srivastava [SS11] proposed a graph sparsification algorithm based on effective resistances: they see the whole graph as an electric circuit, whose edges carry resistances. They proved that any graph with n nodes and m edges can be approximated by a sparse graph. Their algorithm consists in sampling each edge of the initial graph with a probability $p_e \propto R_{eff}(e)$, with $R_{eff}(e)$ corresponding to the effective resistance of the edge.

There are various approaches in linear algebra to sample a matrix in order to get a representative smaller version of the matrix. For instance, one can compute the SVD of the matrix and store the SVD as sample, since it is somehow representative of the data. However, it is computationally expensive and we will not preserve structure nor

sparsity. As a result, the sampled matrix is not interpretative anymore, which has to be avoided in our case. Some authors in the literature focus on properties of linear algebra in order to sample a matrix by a meaningful subset of its columns. This is also known as the column subset selection problem, which is a low-rank matrix approximation, and is linked to graph sparsification. Such a subset can maintain some interesting properties of the initial matrix such as non-negativity, sparsity, and interpretativity [MNP17].

Uniform column sampling often fails to preserve some properties of the initial matrix, since the columns are not sampled according to their importance [CMM17]. McCurdy et al. [MNP17], relying on previous work from Papailiopoulos et al. [PKB14], computed leverage scores in order to select appropriate columns from the genome-wide expression profiles of 3.005 cells from the mouse brain. Leverage scores, particularly used in statistics and regression, measure how far variables are from the others, and as such have been widely used for outlier detection. Leverage scores can capture the importance of the individual columns and are sensitive to collinearity between columns. The DCSS (deterministic column subset selection) algorithm was described in these two papers [MNP17, PKB14]. First of all, the leverage score τ_i is computed for each column. Then, the τ_i values are sorted from largest to smallest and the corresponding columns are added to a solution. These methods require the computation of the SVD approximation, as well as the computation of the Moore-Penrose pseudo inverse, which can become costly for large matrices.

Cohen et al. [CMP16] proposed another method on the same idea for row sampling, but in fact it can also be used for column sampling. In their approach, each new row is either added to the final subset or discarded, depending on the ridge leverage score. They used the ridge leverage scores instead of the leverage score, because it provides a regularization term, so that the scores are less sensitive to noise. They proved that taking an approximation of the ridge leverage scores by computing the ridge leverage score of the previously sampled rows was indeed sufficient. The algorithm adds a new row based on a probability depending on the ridge leverage score value. However, this requires to compute an inverse of a matrix, which can become costly. In order to speed up their algorithm, they proposed a batch version, that stores a smaller set of columns and gives a constant factoral spectral approximation.

Ubaru and Saad [US17] presented another approach, which is a bit different to graph sparsification, and is called graph coarsening. It aims at finding a reduced representation of the original graph, with both less nodes and edges. They stated that graph coarsening methods are less expensive than sampling methods using leverage scores. They assumed that this technique could thus be used for the column subset selection problem, but also for graph sparsification purposes. Their algorithm basically computes the dot product between all columns, and adds the most similar one (where the dot product is the highest) if the dot product's value is above a given threshold. It requires a computation of $O(n^2)$, which is too costly for our purpose.

2.3.4 Sampling Evaluation

Besides the choice of the sampling method, choosing the right tradeof between sample size and representativeness of the sample is another challenge. We want to get a sample size as small as possible in order to be able to process it, but we also want to be able to infer knowledge from it. The sample should be representative of the underlying data. An evaluation is required for the sampled data in order to know if it is representative of the initial data. Statistical tests are well adapted for this purpose, and can help to choose the right tradeof.

The evaluation of a graph sample depends on what we are interested in, i.e., which features we want to keep to state that the subgraph is representative of the initial graph. Leskovec and Faloutsos [LF06] proposed to evaluate the sampled static graph based on its in-degree and out-degree distributions, i.e., histograms of the degree values. The hop-plot, also known as path length distribution, gives insights about how the number of hops between nodes evolves with distance, as well as the hop-plot of the largest weakly connected component can also be useful for this evaluation purpose. Finally, they also suggest to compute the distribution of the clustering coefficient as well as the distribution of singular values of the graph adjacency matrix versus the rank, or the distribution of sizes of strongly and weakly connected components. Ahmed et al. [ANK14] also added to this list the K-core distribution, i.e., the fraction of nodes in the graph that belongs to a hub with minimum degree K, as well as spectral graph analysis measures such as the adjacency matrix's eigenvalues and eigenvectors.

Each property for both sampled and initial graph should be compared using some distributional distance measure in order to evaluate the representativeness of the subgraph. A possibility is to use the Kolmogorov-Smirnov D-statistic, $KS(F_i, F_s) = max_x(|F_i(x) - F_s(x)|)$, where F_i and F_s represent the cumulative distributions (for instance in-degree distribution) of the initial and sampled graph respectively. Another possible evaluation metric is the Skew Divergence $SD(P_i, P_s, \alpha) = KL(\alpha P_i + (1 - \alpha P_s)||\alpha P_s + (1 - \alpha P_i))$. It relies on the Kullback-Leibler divergence KL of the probability density functions of properties between the initial P_i and sampled P_s graphs and is particularly of use if the probability density function of the graph is skewed. This is the case for the degree distributions in this work. Normalized distances (often $L1$ or $L2$ distances) can also be used to compare the similarity of the different properties of the graph [ANK14]. Leskovec and Faloutsos [LF06] also computed the visiting probability of a node in both initial and sampled graph and used the Frobenius norm to calculate the difference in terms of visiting probability. This probability depends on the number of times the node will get visited if the graph is explored randomly. Ubaru and Saad [US17] also used the Frobenius Norm in the evaluation.

In a later work, Ahmed et al. [ADNK14] proposed ways to estimate the mean and variance of the number of edges, of the number of triangles, of the number of connected paths of length two and of the clustering coefficients. Another often-used method to evaluate a sampled graph, consists in comparing graph properties of the sampled graph

with the ones of a graph of the same number of nodes and the same total number of edges picked uniformly at random. Stumpf et al. [SWM05] added that a sampled network can only extrapolate the properties of the initial network if both of their degree distributions belong to the same family of probability distribution.

Zalesky et al. [ZFB10] also proposed network-based statistics to identify functional or structural differences in connectivity networks. This is based on test-statistics of interest computed on the connectivity matrix, and can sample the network into its most relevant subcomponents. The method aims at identifying all potentially connected structures formed by an appropriately chosen set of edges above a given threshold, using a breadth first search. The authors stated the complexity of the algorithm to be $O(M * (N + P))$ with N being the number of nodes, M being the number of random permutations for the test-statistics of interest and P being the number of edges above the given threshold. They added that the algorithm may scale well with larger datasets.

Different sampling strategies were presented, in order to first sample the initial matrix. This is the first step of the cleansing tool developed in this thesis. Then, the sample should be visualised and the visual tool should provide the user with information useful for the final cleansing of the initial matrix.

2.4 Connectivity Analysis Visualization

After sampling the matrix, a graphical user interface (GUI) is needed. The user can visualize the sampled matrix and perform network computations on it. This allows him to define a global threshold and a grouping scheme that can be applied to the initial matrix in order to reduce the matrix size while preserving its most important characteristics. It is thus important to define which informations to display and, more importantly, how to display them in a meaningful and intuitive way. This section describes important large graph visualization methods.

In the literature, two main types of methods exist in order to visualize graphs: node-link diagrams, and adjacency matrices. Node-link diagrams are well adapted to show the global structure of the network, whereas adjacency matrices can help to better visualize communities (how similar two nodes are) [HF07, HFM07].

Connectivity graphs, as stated in Section 2.2, have intrinsic characteristics and, as most "omics" data, are very large. As a result, specific designs have to be developed in order to visualise them properly and to allow researchers to mine them, i.e., detect similarity patterns, or outliers for example. Different techniques have been studied in the literature, and Margulies et al. [MBWG13] identified six main groups of visualisations for brain connectomics [Nyl17]. A group using

- connectivity matrices where the cells are color-coded so that they represent the connection between the neurons (or brain regions), with possibly a visual representation of the brain regions the neurons belong to.

- a 2D undirected weighted or unweighted node-link diagram, where the nodes represent neurons of interest, or brain regions, depending on the scale. Here, edges represent the connections between the neurons or brain regions.
- a 3D undirected weighted or unweighted node-link diagram superimposed with a brain volume, where the positions of the nodes reflect their real anatomical positions.
- a 2D undirected weighted or unweighted node-link diagram superimposed with a brain cross-section (coronal, frontal, or sagittal plane), where the positions of the nodes reflect their anatomical projections on the plane of interest, i.e., the chosen cross-section.
- connectograms, which are chord diagrams between different brain regions.
- using gray scale fMRI images, coloring areas according to activity for networks of interest.

2.4.1 Node-link Diagrams

It has been shown that node-link diagrams behave poorly for very large networks, because of edge cluttering, i.e., there is then a large number of edge crossings. They can be difficult to interpret and visualize. Force-based layouts can help to reduce the cluttering of edges and display a node-link diagram in a more intuitive way. General force-based layouts are not well suited to display small-world networks, as they result in high degree nodes having a central position, often forming a hairball-like layout [GV16].

Bundling techniques such as Hierarchical Edge Bundles [Hol06] or node clustering can help to reduce edge cluttering. Edge bundling is an approach to reduce edge cluttering in graphs, parallel coordinates, or flow maps. Another possibility is to use multiple scales and build a hierarchy in the graph to display higher level hierarchies and zoom within the hierarchy. These techniques should maintain the context of the full graph during its exploration. Some authors also remove weaker edges in order to help in the visualisation, i.e., they remove edges that have a low influence on the graph. However, a threshold has to be defined and one of the goals of this thesis is to get a feedback on how thresholding influences the connectome. Thus, removing weaker edges in the visualisation may not be a good approach. The challenge of such node-link diagrams is to avoid occlusion and cluttering, while emphasizing the important characteristics of the network.

Pajek [BM98] is an important tool in the network analysis and visualisation community, as well as being one of the oldest graph exploration software. It provides powerful visualisation tools and subquadratic algorithms to handle large network analysis. It can realise a recursive decomposition of networks in order to visualize them. Other software such as Cytoscape [SMO+03], Tulip[Aub04] or Gephi [BHJ09] exist in the graph visualisation community.

In the brain context, a big advantage of node-link diagrams is their possible 3D representation that can preserve the brain anatomy. It can display more information about the spatial anatomical context [All15]. According to Henry and Fekete [HF07], these diagrams are also very intuitive for the user and allow the user to realise complex tasks such as following a path between two non-adjacent nodes, while being very efficient and relatively compact for small graphs.

2.4.2 Adjacency Matrices

Adjacency matrices do not clearly display the topological information of the network and behave poorly for path-finding tasks. Most of the time, they are better suited for very large networks, because they can represent the information using less space and avoid cluttering. As a result, they are heavily used for visualizing large matrices.

Heatmaps are one of the most common visualisation of adjacency matrices, representing the data in a 2D grid where the cells are color-encoded according to the edge weights. This visualisation approach is very well suited to display adjacencies of nodes, i.e., how every node is connected to all other nodes.

Adjacency matrices benefit from the absence of occlusion or cluttering and are thus more readable. The are also known for their efficiency in the realisation of simple tasks such as finding the most connected node, links between nodes, or common neighbours. They are however not very suited to represent sparse graphs and require a rather large space to be fully displayed, where the screen space might be too limited to display the entire matrix. Moreover, most of the time, a random positioning of the nodes does not provide any information, and there is a need of node reordering in the matrix in order to visualise patterns.

There are different solutions provided by the literature in order to overcome the problem of the large space required to display a matrix, among which are ZAME [EFD+08], or MultiLayerMatrix [DCF16]. In these solutions, the matrix is aggregated by merging nodes and edges based on clustering algorithms. Aggregation lowers the size and complexity of a graph and allows the user to interact with the matrix based on different visualisation scales.

Interactions with the graph are important to fully explore it. There exist two different kinds of interaction techniques with either a graph displacements (panoramic displacements, zooms), or lenses. The later allow the user to focus on a specific part (or on several parts) of the graph while keeping an overview of the whole graph (context). The user can also interact with the graph using visual parameters, such as highlighting, brushing and linking, or semantic zooming.

ZAME (Zoomable Adjacency Matrix Explorer) [EFD+08] is a visualisation tool to explore large graphs using multiple scale aggregation on adjacency matrices. It aims at overcoming the problem of the large space required to display a matrix. The matrix can be explored by zooming and panning at many levels. Aggregation is performed by first ordering the

elements of the matrix. The resulting aggregates are stored in a pyramidal hierarchy, where every level of detail has half the number of nodes of the level below. It uses tiles of 2D textures to represent parts of the adjacency matrix at different hierarchy levels. A tile management component is responsible to cache and load the individual tiles, and uses an "imposter tile" of coarser level than the wanted one during loading of the tile in zooming and panning interactions. This imposter tile strategy gives the user a direct feedback and then computes the real values to display within the tile at the wanted level. The tool aggregates the elements at different levels and aggregation should be fast and meaningful for the user. The tool uses different representations of the aggregation through glyphs. The aggregation can thus be represented as an average value, or as histograms. It can also show the range of values within the aggregate.

HiPiler [LBK+18] also aims at overcoming the need of a large space to display the matrix. It is based on a divide and conquer approach, where the region of interest is divided into interactive snippets that can be manipulated independently of their neighbourhood. The snippets can then be ordered, arranged, filtered, or grouped. This tool provides both a matrix view, to display a context, and a snippet view.

PathwayMatrix [DMF15] aims at visualising relationships between proteins in biological pathways using an adjacency matrix. In order to explore and visualize these protein interactions, the visualization tool provides filtering, lensing, clustering, brushing, and linking tools, as well as coloured glyphs indicating the reaction types between two proteins. PathwayMatrix also displays a hierarchy view in order to display information about the hierarchy in the proteins.

Later, the same authors developed MultiLayerMatrix [DCF16]. This visualisation tool uses the leader algorithm to cluster similar nodes in order to get an aggregated global view of the matrix. This is based on a given number of final clusters. The clusters can be highlighted according to their within similarity, and every row and column has a thumbnail whose colour represents its within similarity. The tool also displays outliers as greyed out rows or columns. A lensing tool allows the user to navigate within the matrix in order to show more details. MultiLayerMatrix provides a within similarity tool. By hovering the clusters, a similarity matrix appears on the right, displaying the similarity between the members of the hovered cluster and highlighting the leader member of the cluster, i.e., the element that has the most influence on the cluster.

Net-Ray [KLKF14] is an open-source package to visualise and mine billion-scale graphs, with two algorithms Net-Ray-Spy and Net-Ray-Scatter. Net-Ray-Spy allows the user to visualize the adjacency matrix of very large graphs using a reordering of the nodes as well as a scaling of the values and the axes. The matrix is projected into a small matrix that fits on a typical screen, where an element of the small matrix is set to the number of non-zeros in the corresponding submatrix of the big matrix. However the small matrix will be mostly full, so the package uses SlashBurn clustering [KF11] to reorder the nodes and scale the x and y axes using different log scales.

Hagmann et al. [HCG+08] represent the brain connections using different tools. They

display the brain structural connectivity in an adjacency matrix, where entries of the matrix represent fiber densities between pairs of regions of interest. Finally, they also display a 3D representation of the graph with the anatomical positions of the different regions of interest. They use a colour and size code for the nodes and edges in order to display different measures, such as the node strength, the network cores, the network modularity, and hubs.

Another important aspect linked with adjacency matrices is the reordering of rows and columns. Without any particular ordering, the adjacency matrices can be very difficult to interpret. There exist different possibilities in the literature, such as Principal Component Analysis, Barycenter heuristic [MS05], Optimal Leaf Ordering [BJGJ01] or Spectral Ordering [Fek15].

2.4.3 Hybrid Approaches for Connectivity Analysis Visualization

Both node-link diagrams and adjacency matrices have opposite drawbacks. Combining the two approaches in hybrid visualisations can help to overcome the drawbacks of each approach.

Henry and Fekete [HF07] implemented Matlink, a hybrid representation using a matrix view while displaying paths between nodes. Henry et al. [HFM07] also implemented Nodetrix, another hybrid visualisation tool, using node-link diagrams and displaying arbitrary portions of the network as adjacency matrices, which is particularly suitable for networks with strong relations within communities and sparse relations between communities. Even if they increase readability and scalability, these approaches are not sufficient to visualise large-scale graphs [GCR15].

Alper et al. [ABHR+13] compare different techniques for the visualisation of weighted graphs to assess which representation best supports weighted graph comparison tasks in the brain connectivity analysis use case. They gathered seven common neuroscience tasks. For functional connectivity, the main representation techniques use a 3D spatial node-link diagram within the brain volume. 2D projections of node-link diagrams, or force-directed node-link diagrams with a spatial representation of the actual anatomical positions of the nodes are also common for functional connectivity. Anatomical connectivity is often represented using fibers. The authors focused on the comparison of two techniques: an augmented node-link diagram and an augmented adjacency matrix representation. They proposed different visual encodings to facilitate the comparison of two connectivity graphs. They conducted a controlled study to determine which of the two techniques is better. The results showed that the matrices outperformed the node-link representations for the chosen tasks, especially if the graph became dense or large. Alper et al. recommend that adjacency matrices are used unless the graphs are small and sparse.

2.5 Limitations and Challenges

The matrices representing connectivity between neurons or groups of neurons are large, high dimensional, and noisy. As a consequence, it is computationally very expensive and sometimes even impossible to compute measurements on them in order to infer new knowledge about how the brain works. A cleansing of the matrices could reduce their size in memory, which could then accelerate the computations. This cleansing involves thresholding and merging of similar rows and columns. A threshold must be defined, as well as a similarity criterion between rows or columns, and these should not change the important characteristics of the connectivity matrix.

As the user does not want to operate blindly, these cleansing operations should be visualised. As the matrices are usually large and may not fit into memory, a sampling is thus a required preliminary step. The sampled data should be representative of the initial data and fit in the RAM of the user's machine. The sampling is only a preliminary step. It should not take long, and it should be performed in one pass because the reading of some large connectivity matrices can already be time consuming.

The sampled version has to be visualised. Usually, the bigger the sampled data, the more representative it is of the underlying data. At the same time, the bigger the sample, the more difficult it is to compute measures on it and to visualise it, either by adjacency matrices, which may be too large to be entirely displayed, or by node-link diagrams, which are cluttered. The user should have an idea of how representative the sample is of the initial matrix, and how the cleansing operations he or she performs on the sampled version are likely to affect the initial connectivity matrix during the final cleansing on the initial data.

2.6 Comparison and Summary of Existing Approaches

Sampling and visualisation methods were presented in Section 2.3 and Section 2.4. Sampling aims at reducing the initial matrix size in order to get an overview of the content of the matrix. Then, the user can visualise it with a tool providing him feedback on some cleansing operations. The current section provides a comparison and a summary of the different proposed sampling approaches, and of the visualisation approaches. Constraints will be described for both approaches, as well as the applicability of the proposed methods. Moreover, the sample, as well as the cleansed matrices should be evaluated. Hence, evaluation approaches are also analysed.

2.6.1 Sampling

Different approaches exist in order to sample datasets. The most straightforward methods are the ones based on random selection, or on systematic sampling. These methods can be performed in one pass, with an O(1) update cost. Systematic sampling suffers mainly from the fact that not all values have the same probability of being sampled. As a result there is no guarantee that the sample actually represents the underlying data. Moreover,

if there is some periodicity in the data that is the same as the frequency of the sampling, the data will not be representative [Haa16].

Random selection improves the systematic sampling in the sense that all values have equal probability of being chosen in a non-weighted approach. We could use an approach linked to random node sampling, where we first sample uniformly the indices of columns and rows, and then collect the weights at the positions from the sampled indices. Random selection approaches include binomial sampling, or reservoir sampling [Haa16]. In binomial sampling, an item is chosen with probability p or discarded with probability $1 - p$, resulting in a non-fixed sample size. Reservoir sampling implies that the size is fixed to k, the first k elements are added to the reservoir and then a new element will replace one element of the reservoir chosen uniformly with probability p. However, a drawback of the random selection approach is that hubs may not be found, as the data follows a long-tail distribution. There are not many hubs and they may not be sampled. The sample may not have the hubs properties that a typical brain connectivity matrix would have. Moreover, it has been shown that uniformly sampling the columns or rows at random works for some data, but may remove some important information in other cases.

Stratified sampling takes samples from identifiable subgroups (strata), and thus even small groups can be represented. This sampling can be performed randomly. In an optimal selection, the number of sampled elements per strata depends on the strata size. This can be useful in order not to discard brain regions if a strata is defined by the anatomical hierarchical brain structures.

The presented dimensionality reduction methods, i.e., Isomap, PCA, SVD, FA, MDS and t-SNE (t-distributed stochastic neighbour embedding), are powerful. However the data is projected on a lower dimensional space and as such, we will not keep the values of the connectivity matrices. It would then not make sense to project connectivities, as we want to keep real and meaningful values. Moreover, these algorithms are rather costly, for instance t-SNE has quadratic computational time. Thus, these techniques cannot be applied in our case.

Clustering is an unsupervised approach aiming at finding related vectors, in our case columns or rows of the matrix. Through clustering, we can keep only the representatives of the found clusters. Different clustering methods exist, namely hierarchical clustering, partitioning clustering, density-based clustering, grid-based clustering, and micro-clustering.

Hierarchical clustering allows scientists to hierarchically refine clusters based on their intra-cluster similarity. We can choose clusters on different hierarchical levels. However, the best hierarchical clustering algorithm applicable in our case (CURE) runs in $O(N_{sample}^2 log(N_{sample}) + N_{sample} k)$, with N_{sample} being the number of randomly sampled elements before clustering, and k the number of final clusters. This complexity is too large for the algorithm to be applicable in our case.

Partitioning clustering methods aim at grouping together vectors that are similar into

clusters, with a total of k clusters. These methods could then be applicable to our case with k being the number of columns or rows that we want to select. CLARA is a partitioning clustering method well adapted for large datasets, but still runs in $O(k(40+k)^2 + k(N-k))$ steps.

Density-based clustering methods, contrary to partitioning clustering algorithms, can find an arbitrary number of clusters based on their density. This can discard outliers and low density clusters, as well as find clusters with shapes that are not spherical. However, they run in at least $O(NlogN)$ time.

Grid-based clustering methods first partition the dataset into cells, based on a grid-structure, and then sort the cells according to their densities in order to identify clusters. Although these algorithms can be rather fast, the data should first be divided into a grid. Micro-clustering techniques can be applied to streaming data and thus provide a real time summarization of these streams. An early work on micro-clustering in described by Aggarwal et al. [APHW03], who developed an algorithm called CluStream. It runs in two phases ; an online phase gathers summary statistics about the streaming data, and an offline phase performs clustering on these summary statistics. In the online phase, the algorithm uses a pyramidal time frame approach in order to store the statistics in form of micro-clusters. Micro-clusters store the sum and sum of square of times stamps, the sum and sum of square of the data values as well as the number of values within the cluster. The algorithm uses then a k-means clustering on the micro-clusters in the offline phase. There are also variants, using the same micro-clustering approach at first, and then applying other clustering algorithms on the micro clusters, such as Denstream (a density-based approach) [CEQZ06].

A drawback of the clustering approaches lies in the fact that not all columns can be stored while the clustering is performed. Indeed, the computation of the new centroid in every iteration needs the positions of the other points. As a result, clustering does not seem to be a solution for sampling.

Another approach would be to compress the data using wavelets. Wavelets have been widely used for compression purposes in signal processing theory. With a lossy compression, where we keep only the biggest wavelet coefficients, we can still achieve a good reconstruction accuracy while saving a lot of space. Some approaches are applicable to streaming data. The AWSOM framework proposed by Papadimitriou et al. [PBF16] uses first wavelets and then estimates the correlation between the different wavelet coefficients. It could be appropriate, as it requires only one pass over the data, and the updates when a new element arrives is in $O(1)$. This method provides a lossy compression and eliminates noise, thus the final sample is not the real data any more. Moreover, this approach would require a post-processing in our situation. The connectivity matrix has to be sampled, and then the sampled version has to be displayed. For instance, the wavelet method could be applied row-wise in a first step, storing the k main coefficients for every row, and then in a second step rows with a too low maximum coefficient could be discarded. Finally, the remaining rows should be reconstructed in order to be displayed. Moreover, it is not convenient to compute statistics or operations on wavelet-compressed data.

As an adjacency matrix represents a graph, graph-based sampling methods can also be applied in our case. For example, Tang et al. [TCM16] proposed a method, which in one pass realizes the summary of a graph using hash functions. In our situation, we would need a postprocessing to recover the rows, and this method does not really help in finding how to choose which of the connections to keep. Other traditional graph-based sampling methods are either based on random node selection, on random edge selection, or on exploration techniques [LF06]. As previously stated, random edge selection is not very appropriate in our case, because of the storage conditions, while random node selection and its variant random degree node sampling could be of use. However, while leading to good results in terms of graph approximation, the exploration techniques are too costly to be computed in our case, as they require several passes over the graph. The approaches presented by Ahmed et al. [ANK14, ADNK14] are also not very well suited in our case, because of the final matrix size, which should not be too high. A naive approach could be to read a row with a given probability and select among this row ids of neighbouring nodes to explore, adding these ids into a slack of nodes to examine. When a row is read, we first need to check if its id is in the ids slack and otherwise sample it with a given probability. The connections that we did not take into account are lost, and we would need a second pass to gather columns that are in the id list. Moreover, all ids might be in the id list, because the process is random, and it is biased toward order of arrival, i.e., a row may be discarded, while it would have been in the list later on.

Graph sparsification or matrix low-rank approximation methods could be applied in our case. These methods aim at keeping the most important information while reducing the storage requirement of the matrix. Not all low-rank approximations methods can be applied. For instance, the k-SVD approximation approach is not interpretative anymore, whereas we want to keep some of the data values. The column subset selection approach (also generalisable in the graph sparsification context) seems well adapted to our situation, especially the online row sampling algorithm presented by Cohen et al. [CMP16]. Some authors proposed streaming algorithms for this very problem, and rely on leverage scores, or on a regularized version called ridge leverage scores, of a new column to be added in order to decide whether or not to keep the column. However, these approaches require the computation of the inverse of a rather large matrix, as well as the product of matrices, which can be computationally expensive. As such, these approaches are not that suited for our purpose, even if they keep meaningful properties of the initial matrix.

Finally, matrix coarsening is another approach that is less costly than leverage score computation. It consists in computing the dot product of a vector with all other not yet sampled vectors, grouping together vectors that are similar, and marking these similar vectors as visited. This approach mathematically means that we group together groups of neurons that have rather similar firing patterns i.e., a similar out-degree distribution, and that we will try to keep groups of neurons that have rather different firing patterns.

Constraints

A guided visual tool had to be developed in order to cleanse connectivity matrices. This tool has to respect some constraints:

- the matrix can only be read once, since it takes already long to read it
- the data can only be read row-wise (rows representing outgoing connectivity)
- the chosen sampling scheme should be representative, i.e., we cannot discard entire regions from the brain, all regions should be represented
- the sample and all results from computation should be stored in less than 12 GB RAM, since a typical customer PC has 16 GB RAM
- there must be a consistency of the ids of the sampled rows and columns in order to display the sampled matrix in a visual tool, i.e., there should ideally be the same ids in the sampled rows and columns
- the computation time of both the sampling and the cleansing operations should not be too long, i.e., the sampling should take at most a few hours
- a visual tool should guide the user in the cleansing operations, providing feedback for all operations
- the sampled matrix should be displayed in a meaningful way

For the sampling step, we have to find a tradeof between representativeness, sampling, size, and sampling time complexity.

Applicability

With the above constraints in mind, we will now define the applicability of all proposed approaches. As we do not want to discard regions, a possible approach would be to sample within the hierarchy using stratified sampling, at a level specified by the user. Thus, stratified sampling seems a good approach. We should then define how the sampling is realized within the regions. Compared with random sampling, systematic sampling does not seem to offer any advantage, as a result this technique will be discarded.

Random sampling applies the same probability to all columns or rows of being selected. We should keep in mind that the sampled ids from the rows are also included in the ids sampled from the columns, but the approach is applicable. If it is combined with stratified sampling, where we want to keep a certain amount of data from all strata, reservoir sampling seems to be a better solution than simple random sampling. It ensures a fixed size of samples taken among the strata. So both these approaches are applicable.

Because of their rather high computational costs, clustering methods do not seem a good solution for our data. Moreover, a meaningful distance would have to be defined. However, they could be a solution for grouping together similar rows or columns later on.

Applying wavelets row by row does not seem a satisfying approach at first sight. We cannot rely on the order of the rows or columns in our connectivity matrices. As a result, there is no guarantee that in a row, neighbouring values would be similar. It highly depends on the structure of the data, i.e., if the data is sparse then it could work and would lead to very condensed representations of the rows. On the other hand, if the data is not sparse, which seems to be the case, we cannot assume anything about the similarity of neighbouring values. It would make more sense to compute the wavelet coefficients between rows of the same region of interest. In this case, values can only be compared on a one-to-one approach. As a result, row-based wavelets do not seem very applicable in our case.

The column subset selection proposed by Cohen et al. [CMP16] involves row sampling, and uses the already sampled rows to compute an approximate ridge leverage score of a newly incoming row (or a bucket of l incoming rows). The authors relied on the fact that the number of columns would be less than the number of rows, which does not hold in our case. The approach requires the computation of the product between the transposed sampled matrix and the initial matrix, which results in a square matrix with as many rows and columns as the number of columns in the initial matrix. The inverse of this matrix should also be computed, which is computationally too expensive, actually the matrix may not even fit into memory. As a result this method will be discarded.

As we want that all indices of groups of neurons in the sampled rows to be also in the sampled columns, the column/row subset selection is not really appropriate. We only decide whether or not to keep the rows based on the ridge leverage scores. For instance, there is no guarantee that the final matrix will actually be representative and preserve the properties of the initial matrix. We would need first to sample the rows, while keeping a rather large number of rows, and then refine this first sampled matrix by applying once more the column subset selection column-wise. The result may not be representative, because some important information might have been lost in the first pass, and we would also have to get a matrix with matching column and row ids.

The graph coarsening non-modified method would be impracticable. The modified version, where the dot product is computed among vectors that are already in the sample could be a solution, however it would still require a rather long time to compute. A possible solution would be to use an AMS sketch to compute the dot product, which would accelerate the process. A drawback of this approach lies in the fact that it only discards rows, and for instance in the case of our structural connectivity matrix, this would lead to an even greater unbalance between the row and column numbers. Moreover, if it is not applied in combination with stratified sampling, there is no guarantee that all brain regions would be equally represented.

There is globally a problem with row selection approaches, since we would like to keep

only columns that are sampled row-wise. If we compute a value for a row in order to decide whether or not we should keep it, we do not know which columns we should take into account. Let us assume that we want to sample rows that are different by computing their angle (graph coarsening approach), and that one of the rows has a rather large angle with the others rows, meaning this row is different, but that this difference only comes from a portion of its cells. For instance, its sampled version could be exactly the same as another sampled row, but with some non-sampled values that are very different. If the two rows are kept but the columns where the difference lies are discarded, then the two rows would be very similar in the sample whereas they are not similar in the initial matrix.

Table 2.3 provides a summary of all the presented approaches.

Algorithm type	Advantages	Drawbacks
Systematic sampling	very fast exact size guaranteed	may not be representative
Hierarchical clustering (CURE)	gives an a priori good partition	needs two passes over the data
	exact size guaranteed	rather expensive requires a number of clusters
Partitioning clustering (CLARA)	partitions the data into k similar clusters	k is fixed, the underlying distribution may not match the initial distribution
	exact size guaranteed (fixed k)	rather expensive
Density-based clustering	discards noise based on density	rather expensive
	finds clusters of arbitrary shape	no guarantee on the number of found clusters performs poorly if the density is not even
Grid-based clustering (FC)	fast, one pass	heavily relies on initialisation
	deals with noise and outliers finds clusters of arbitrary shape	
Wavelets	rather fast rather precise reconstruction possible denoising (discards lower weigths) compact storage	changes the data (denoised) requires post-processing
Matrix sparsification (Leverage scores)	can be a very good approximation and preserves important characteristics of the matrix	rather expensive

2.6.2 Visualization Cleansing Tool

Two main types of visualisation methods exist to display graphs, namely node-link diagrams and adjacency matrices. These two approaches have advantages and drawbacks. As a result hybrid approached were also developed. Adjacency matrices and in particular heatmaps are more suitable to represent large matrices. As these matrices can be very large, they also need to be aggregated in order to be displayed, such as in the MultiLayerMatrix approach [DCF16].

Connectivity matrices are not simple matrices, as they have a biological meaning. As a result, a spatial representation is also of interest, in order to know from where the data comes from within the brain. This can help to infer knowledge. In a heatmap display, the rows and columns have to be aggregated but also ordered, to be able to visualise specific patterns. There are several ordering possibilities, such as the Barycenter heuristic [MS05], the Optimal Leaf Ordering [BJGJ01], and the Spectral Ordering. Fekete [Fek15] implemented the Javascript framework Reorder.js. The brain has a very specific structure, where the connectivity within brain regions is usually higher than the connectivity between regions. As a result, an ordering of the data by the brain regions would also make sense.

2.6.3 Evaluations of the Method and of the Sampling

Different metrics exist in the literature to evaluate connectivity matrices. Connectivity matrices can also be seen as graphs with special properties, i.e., small-worldness, hubs, high clustering, efficiency, modularity, small average path length, long-tail in-degree, and out-degree distribution. The distribution of these properties could be evaluated using Kolgomorov-Smirnov D-Statistics or Skew Divergence. For instance, Betzel et al. [BAKG$^+$16] based their evaluation on the Kolgomorov-Smirnov statistics of the graph's degree, clustering, betweenness centrality, and edge length distributions. These are metrics of interest in order to evaluate our sampling. Some authors also mentioned the spectral properties of a graph. Some of these metrics can be rather complicated to evaluate, and we may not be able to evaluate them on the fly.

The different methods need to be tested and evaluated in order to decide on the final sampling size and to analyse how well the sampled matrix is representative of the initial matrix. We will differentiate two cases: a global evaluation of the method, and an evaluation of the sampling. The global evaluation does not have as many requirements as the sampling evaluation. We can perform the global evaluation using a server, whereas the sampling evaluation should be performed on the user's PC and keeps all memory and time constraints. We can perform more measures in the global evaluation than in the sampling evaluation. The Brain Connectivity Toolbox [bct] offers a lot of different evaluation metrics. The matrices should be square to compute these measures. Some interesting metrics require binary connectivity matrices. Our connectivity matrices are noisy and there are not that many zeros. It follows that it is not trivial to define how to binarize the connectivity matrix. The PAGANI Toolbox [DXZ$^+$18] can compute some network measures quite fast and will be used. It also requires square matrices.

In Table 2.2, we summarized interesting measures for brain connectivity matrices. These measures were described by Rubinov and Sporns [RS10]. This table is based on the neuron level, whereas in our case it would be more appropriate to speak of groups of neurons. A local measure is defined as a measure where each node has a value, while a global measure computes a single value for the whole graph.

CHAPTER 3

Methodology

The last chapter presented the literature review concerning the current topic. The present chapter first describes the specifications, see Section 3.1. A sampling method had to be engineered and a visual tool had to be developed. Their development is presented in Section 3.2. The choice of programming languages is described in Section 3.3 and the data models that were used is described in the Section 3.4.

3.1 Specifications

The specifications for the sampling are the following:

- the data and all computation should fit within 12 GB of RAM,
- entire brain regions should not be discarded,
- if a column is sampled based on his group of neurons index, the corresponding row should also be sampled if it exists,
- the sample should be representative of the initial data.

The specifications for the cleansing are the following:

- the computation for the cleansing on the sampled matrix should also fit along with the sampled matrix inside 12 GB of RAM,
- the user should get some visual feedback on the sampling operations,
- the user should be able to perform thresholding and merging of similar rows and columns,

- there should be a finalisation step where the user can get a recap of the cleansing parameters he or she used, as well as a possibility to either get a Python script to cleanse the initial matrix later, or directly cleanse the initial matrix.

3.2 Concepts

The current section presents the developed concepts regarding the sampling and the visualisation methods. Based on the results of the literature review, it was decided to implement two sampling approaches based on reservoir random sampling a "naive" sampling approach based on reservoir sampling, detailed in subsection 3.2.1, and a more sophisticated one based on the anatomical hierarchy, see subsection 3.2.2.

Figure 3.1 presents the three main steps of the work:

1. sample the initial matrix
2. visualize the sample and define cleansing parameters on it
3. apply the cleansing parameters on the initial matrix.

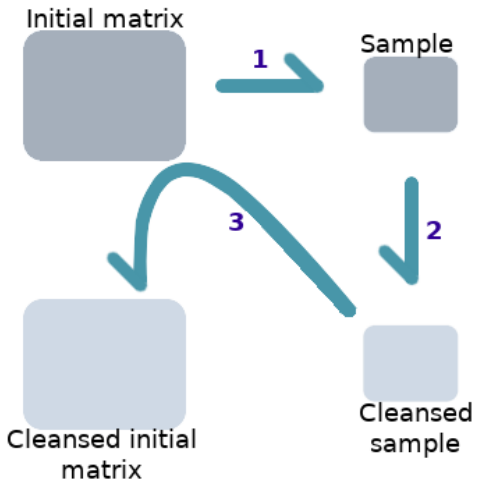


Figure 3.1: Main Concept of the Work

3.2.1 Developing the Naive Random Sampling Method

The first idea was to sample randomly using a reservoir. It would have worked as followings, see Algorithms 3.1 and 3.2. Algorithm 3.1 describes the overall matrix sampling process. Algorithm 3.2 summarizes the actual reservoir sampling step within

the overall sampling process in order to define if a column is kept in the sample or discarded.

Algorithm 3.1: Basic random sampling algorithm

```
Data: Big matrix M_{m×n}, desired sample storage size finalDesiredSize
Result: Sampled matrix S
// read first row: column ids
columnIds ← M[0];
// optimization step
finalNumberOfColumns ←
 optimization(M.colNumber, M.rowNumber, finalDesiredSize);
// reservoir sampling: define which columns to keep
sampledIds ← reservoirSampling(columnIds, finalNumberOfColumns);
// sample the matrix
currentIndex ← 0;
for i ← 1 to M.rowNumber do
    if M[i][0] ∈ sampledIds then
        if currentIndex < finalRowNumber then
            sampledRow.append(M[i][j])∀j ∈ sampledIds;
            sampledMatrix[currentIndex] ← sampledRow;
        end
        else
            randNb ← rand(0, currentIndex);
            if randNb < finalRowNumber then
                sampledMatrix[randNb] ← sampledRow;
            end
        end
        currentIndex ← currentIndex + 1;
    end
end
return sampledMatrix
```

However, after getting the data, further supplementary challenges also had to be taken into account

- the matrix is not necessarily squared (and the distribution of which rows ids are present is not uniform)
- the rows and columns ids orders may differ
- we do not know the actual number of rows
- we should not overflow the size of the reservoir during run-time

Algorithm 3.2: Reservoir sampling algorithm (reservoirSampling)

Data: list of ids of the columns $columnIds$, desired number of columns $finalNumberOfColumns$
Result: list of sampled ids $sampledIds$

```
sampledIds ← [];
for i ← 1 to finalNumberOfColumns do
    sampledIds.append(i);
end
for i ← finalNumberOfColumns + 1 to columnIds.size do
    randNb ← rand(0, i);
    if randNb < finalNumberOfColumns then
        sampledIds[randNb] ← i;
    end
end
return sampledIds
```

- rows have different sizes (in terms of memory storage)
- we need to keep the columns corresponding to the rows we keep (all kept row ids should be in the column ids)
- we are only allowed to read the matrix once

We do not know the ids of the rows. The matrices used for this thesis have at least as many columns as rows. As the column number may be higher than the row number, we have to perform sampling both row-wise and column-wise. The column-wise sampling has to be performed at the beginning, i.e., we need to define which columns we will keep for the whole matrix at the beginning. The problem is that we do not know which columns should be kept in the column-wise sampling, so that the ids of the kept columns are actually ids of rows of the initial matrix. In the structural connectivity matrix, there are around six times more columns that rows (the ratio between the row and column numbers is around 15 %). If the matrix was squared, it would be easy. One can simply sample the ids by reading the first row, i.e., the column ids, and keep the corresponding ids for both columns and rows, as presented as first idea.

However, as soon as the number of rows is largely inferior to the number of columns, there is no guarantee that the chosen indices will be present in the rows. For instance, the ratio between the number of rows and columns is around 15 % in the case of the ABMA structural connectivity matrix. In the worst case there may not be a single row kept. This worst case is highly improbable though. We can use statistics in order to define the number of rows and columns to keep, as presented in the following subsection.

We can compute statistics in order to sample first on the columns. We want to discard as many columns as possible, while keeping with high probability at least some portion of the

initial number of rows in the remaining columns. Then, we will only store rows that are in the remaining column indices until the reservoir is full. We can apply reservoir sampling if some rows remain. A small program was developed to perform the estimations of the best parameters with this method. For the structural connectivity matrix of resolution 450,000x67,500, and size 90 GB, we can discard around 300,000 columns, i.e., keep 32 % of the initial columns, if we want to keep at least 32 % of the initial rows with a 95 % chance.

It was decided to set the initial number of rows and columns as a user input, as he or she knows how many rows and columns there are in the file he or she is providing to the tool. It could have been possible to estimate the number of rows, by reading a few rows with an estimate of the size. It is however rather imprecise.

A high-level explanation of the naive sampling method was explained here. The method is later detailed in Chapter 4.

3.2.2 Developing the Stratified Random Sampling Method

The second idea was to sample randomly on the strata using a reservoir. For this task, we faced the same challenges as with the simple sampling approach. However, we also had to define what does a representative size of a strata means. We want the different strata to be represented according to their importance in the initial matrix. We have to choose to which extent the sampled strata size matches the initial one. We defined the strata size as the number of groups of neurons belonging to the strata, whose ids are among rows and columns. Having a final number of columns representative of the strata size, i.e., the number of columns belonging to the strata, seemed to be the best approach. Algorithm 3.3 describes the overall stratified sampling method. Algorithm 3.4 describes the reservoir sampling step within the overall approach in order to define if a column is kept in the sample or discarded.

Agorithm 3.3 works as following. The first line, containing the column ids, is read and processed with the 3D position mapping file in order to get the anatomical regions for all column ids. These anatomical regions are not necessarily at the level of interest, and need to be processed in order to get the final regions on the anatomical level of interest. Once the final anatomical region ids and the number of column ids per anatomical region are computed, the optimization step needs to be performed in order to get the final numbers of rows and columns per strata. This optimization step is described in the next subsection. A reservoir sampling is performed on the column ids in order to define which columns and rows should be kept. In this sampling, there is a reservoir per anatomical region in the level of interest, and columns are sampled in the reservoir corresponding to their anatomical region. Then, the matrix is read row by row. If a row belongs to the sampled ids, it will be kept if the reservoir of its corresponding strata is not already full. If it is already full, reservoir sampling will once again be performed and the row will either be discarded or it will replace one of the currently stored rows for the strata it belongs to.

Algorithm 3.3: Stratified random sampling algorithm

Data: Big matrix $M_{m \times n}$, anatomical level of interest $levelOfInterest$, anatomical mapping $anatMapping$, desired sample storage size $finalDesiredSize$
Result: Sampled matrix S

```
// read first row: column ids
columnIds ← M[0];
// get the anatomical regions ids: list of column ids for
   each anatomical region id on the level of interest
anatRegionsColumnIds ←
 getAnatRegionsIds(levelOfInterest, anatMapping, columnIds);
// optimization step
numberOfColumnsToKeepPerStrata ←
 optimization(M.colNumber, M.rowNumber, anatRegionsColumnIds, finalDesiredSize);

// reservoir sampling: define which columns to keep
sampledIdsPerStrata ←
 reservoirstrataampling(anatRegionsColumnIds, numberOfColumnsToKeepPerStrata);

sampledIds ← flatten(sampledIdsPerStrata);
// reservoir indices for the sampled matrix
currentIndexStrata ← [];
maxIndexStrata ← [];
currentId ← 1;
for i ← 1 to sampledIdsPerStrata.size do
    currentIndexStrata.append(currentId);
    maxIndexStrata.append(currentId + sampledIdsPerStrata[i].size);
    currentId ← currentId + sampledIdsPerStrata[i].size;
end
// sample the matrix
for i ← 1 to M.rowNumber do
    currentStrata ← anatRegionsColumnIds.find(i);
    if M[i][0] ∈ sampledIdsPerStrata[currentStrata] then
        currentIndex ← currentIndexStrata[currentStrata];
        if currentIndex < maxIndexStrata[currentStrata] then
            sampledRow.append(M[i][j]) ∀j ∈ sampledIds;
            sampledMatrix[currentIndex +
             currentIndexStrata[currentStrata − 1]] ← sampledRow;
        end
        else
            randNb ← rand(0, currentIndex);
            if randNb < maxIndexStrata[currentStrata] then
                sampledMatrix[randNb + currentIndexStrata[currentStrata −
                 1]] ← sampledRow;
            end
        end
        currentIndex ← currentIndex + 1;
    end
end
return sampledMatrix
```

Algorithm 3.4: Reservoir strata sampling algorithm (reservoirstrataampling)

Data: list of column ids for each anatomical region id $anatRegionsColumnIds$, list of number of columns to keep for each anatomical region $nbColsToKeepPerStrata$

Result: list of column ids to keep in the sampling $sampledIds$

```
1 sampledIds ← [];
2 for strata ← 1 to anatRegionIds.size do
3 |   colIdsCurrStrata ← anatRegionsIds[strata];
4 |   nbColsToKeepCurrStrata ← nbColsToKeepPerStrata[strata];
5 |   sampledIds.append(reservoirSampling(colIdsCurrStrata, nbColsToKeepCurrStrata));
6 end
7 return sampledIds
```

However, we want to be sure that there are at least $n_{rowStrata}$ columns belonging to rows left in the $n_{colStrata}$ columns left for the strata. This does not scale linearly with the size of the strata. As a result, a simple computation like the one for the simple random sampling method cannot be derived. However, we can use optimisation tools in order to have the best tradeof between final size and strata characteristics.

A high-level explanation of the stratified sampling method was explained here. The method is later detailed in Chapter 4.

3.2.3 Visualizing the Sampled Connectivity Matrix

We want to visualize the connectivity matrix in an efficient and intuitive way. However, there are several challenges for this task

- the sampled matrix is still large, the number of groups of neurons is important so that not everything can be displayed at the same time, we have to use different levels of detail
- we need to aggregate together the groups of neurons to get the different levels of details
- we have to handle the exploration of the connectome (i.e., we need to keep contextual information about the level of detail we are currently exploring)
- the computation time should not be too long

There are different required elements in the visualisation. There should be a representation of the sampled matrix. A mapping to the hierarchy to know to which brain regions the different groups of neurons belong is also required. Finally, there should be a spatial mapping to know where the groups of neurons from the matrix are located. Network

measures are interesting and should be displayed in order to see the effects of the operations on the sampled matrix.

The results from the literature review indicate that a heatmap is more suitable than other visualisation tools to represent large matrices. Thus the heatmap will be the main element of the visualisation. The rows and columns should be aggregated, because otherwise the matrix would be too large to display on a standard consumer screen. Like with MultiLayerMatrix [DCF16], it was decided to aggregate the rows and columns based on clustering. Hierarchical clustering seems particularly suitable to this case, because of the hierarchical structure of the brain and because it can allow the user to navigate in different levels of details (i.e., hierarchical levels). Two types of clustering could be of interest. The first one is based on similarity (i.e., how similar the firing patterns of different groups of neurons are). The second one is based on the anatomical regions and levels. Both can be interesting for the user, so he or she should be able to switch from one view to the other one.

3.3 Languages

A requirement for this thesis was to develop a web-based visualisation tool using ReactJS [rea], which is a JavaScript library developed by Facebook to build user interfaces. This library uses Components as graphical elements that have a state and should be updated when this state changes. ReactJS can be considered as the View part in the MVC model (Model View Controller). The Model and Controller parts can be handled through the Redux library [red], which stores the whole state of the application. All components only depend on one single so-called "store" where all state variables are stored. This avoids complex and unexpected behaviours when all properties are passed from a parent component to one of its children. As such, the front-end was developed in JavaScript using ReactJS and Redux. The webpack library was also useful for the development.

The application is divided into two parts. The back-end, on the server side, handles heavy computations and delivers information to the front-end. The front-end displays these informations and constitutes the user interface.

The choice of language for the back-end was free, but due to the requirements of memory consumption and of speed, C/C++ and Python were considered as main candidates. C/C++ is faster and has a better memory handling. However, it is a rather complicated language, which is not very appropriate for a proof of concept, and it does not interface well with the web. On the other hand, Python offers many powerful open source libraries, like SciPy[sci], NumPy[num] or Flask [fla]. The SciPy library was also used for its clustering tools: the hierarchical clusterings were computed using the SciPy linkage function. NumPy was the mostly used Python library for this thesis, since it offers very powerful numerical representations and computation tools and uses arrays that are contiguous in memory. It thus saves a lot of space compared to standard Python arrays. Finally, Flask is a web library that was used to create the back-end part. The Cython package [BBC+11] seemed to be a good compromise between C/C++ and Python, and

was thus used for the most critical parts. It has however a drawback. It needs to be compiled, but once compiled it can interface perfectly with Python code.

Hence, the application is divided into a web front-end (JavaScript, ReactJS, Redux, webpack) and a back-end (Python). Most interactions with the user will require a call from the client (front-end) to the server (back-end), using the HTTP protocol (POST and GET requests). As a result, all computationally expensive operations will be performed by the server, which will then send the appropriate results to the client so that the results can be displayed for the user.

3.4 Data Models

During this thesis, a directed structural connectivity matrix of size 67,500 x 450,000 (around 90 GB) with a 100 microns resolution from the Allen Mouse Brain Connectivity Atlas was used, as well as a lower resolution one (200 microns, 5266 x 60,009, around 1 GB size). The rows in these matrices represent the injection sites, and the columns represent the locations in the brain. There is a non-zero value between a row and a column if the neurons in the injection site projects to the region in the column. We only have around 15 % of all possible injection sites. A connectivity matrix should be squared, but in our case it is not, since we have missing rows. Thus the sampling scheme should not consider a squared matrix. Moreover, the rows and columns have an id corresponding to a voxel id. The order of ids in rows and columns may differ. The voxel id corresponds to a position in a volume of 132 x 80 x 114 voxels, where the value of each voxel references the id of a region at the lowest level of anatomical regions within the hierarchical representation of brain regions. This volume can map the columns of the connectivity matrix to their corresponding brain regions.

We will also use a correlated gene expression network that quantifies tissue-tissue relationships across genes [LHA$^+$07, RAM$^+$15] and consists of a 56,585 x 56,585 undirected matrix, with a 200 microns resolution, and positive correlation coefficients between zero and one. This matrix also has a corresponding 67 x 41 x 58 volume.

CHAPTER 4

Suggested Solution/Implementation

The current chapter presents the suggested solution and all implementation details. Section 4.1 describes the sampling methods, which were already mentioned in the previous chapter. Implementation details are presented, as well as methods to define the best row and column ratio. Section 4.2 explains the different design steps for the final visualisation tool, and also describes the algorithms used in the visualisation tool.

4.1 Implementation of the Sampling Methods

Two sampling methods were implemented as detailed in the Concepts (see Section 3.2). The naive random sampling is based on node sampling. The stratified random sampling approach is based on random sampling within strata from the brain anatomical hierarchy.

Both approaches were implemented in Python, using NumPy and SciPy [JOP]. In particular, the optimization package of SciPy with the *brenth* function was used to compute the best rows and columns (as well as strata) ratios to fit within a given size.

The naive random sampling approach was the first to be developed. It was followed by the stratified approach. Actually, the naive random sampling approach is a particular case of the stratified approach, where there is only one strata (i.e., where the anatomical level of interest would be 0, corresponding to the root of the anatomical hierarchy). As such, only one method remained at the end.

As described in the Concepts (see Section 3.2), the first line, containing the column ids, is read and processed with the 3D position mapping file in order to get the anatomical regions for all column ids. There is a first reservoir sampling on the column ids, taking their anatomical region into account, to decide which columns should be kept. The

matrix is then read row by row and rows are either kept or discarded depending on their anatomical region.

In order to facilitate the following processing, it would be convenient to have the same order for all rows and columns, based on the ordering in the anatomical file. An ordering of the rows and columns is thus processed. The columns can actually be ordered while being stored in the final matrix. Indeed, the stored columns will remain until the end. However, due to the reservoir sampling of the rows, the later can only be ordered by strata. Then, the final rows have to be ordered, and this can take into account the pre-ordering per strata in order not to use too much memory.

The reservoir sampling was optimized in the memory aspect using NumPy arrays of floats stored in four bytes. It was also optimized on the time aspect, since the final user does not want to wait too long for the sampling of the connectivity matrix. The bottleneck was in the parsing of the rows, the Python parser was very long. The Cython package [BBC+11] was used in order to speed up the parsing and the computation generally.

4.1.1 Computation of the Best Row and Column Ratios

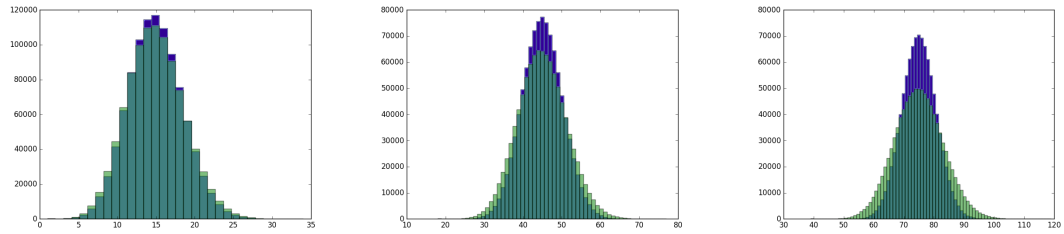
The number of rows and columns of the initial matrix may differ, as for the structural connectivity matrix. We need to decide at the beginning which columns will be kept in the sample, based on their id. However, we do not know the row ids before reading the matrix, and we want to keep the rows corresponding to the kept columns. We can define the size of the sample in terms of space, but then we also need to define how many columns and how many rows the sample will have. Here, we define strategies to define how many rows and how many columns should be kept for a given final sample size.

Naive Sampling Method

If the number of rows and columns of the initial matrix differ, it may be interesting to keep different ratios of the initial number of rows and columns. While discarding some column ids, we have to take care there are still enough ids that belong to the rows. We can use statistics for this purpose. The choice on which ids to keep and which ones to discard is based on the columns ids, based on random for statistical validity. This is an unordered sampling without replacement. An id will either be selected once or will not be selected. As a result, the corresponding probability model is the hypergeometric model. The hypergeometric model can be approximated by a binomial model if the number of samples is under 10 % of the initial population.

Figure 4.1 summarizes the differences between the binomial sampling (green) and hypergeometric sampling (blue) with a 15 % percentage of success cases. With a 10 % sampling rate, the binomial sampling is rather well approximated by the hypergeometric one. If the sampling rate increases, the approximation is not possible anymore.

The hypergeometric model is a Gaussian with the same mean as the corresponding binomial model, but with a smaller variance. The mean of the hypergeometric model is



(a) 10 % sample rate (b) 30 % sample rate (c) 50 % sample rate

Figure 4.1: Differences between the binomial and hypergeometric laws with different sample rates

$\mu = \frac{n_h m_h}{N_h}$ and the variance is $\sigma^2 = \frac{n_h m_h (N_h - n_h)(N_h - m_h)}{(N_h - 1) N_h^2}$, with n_h being the number of trials, m_h is the number of success items in the population, N_h is the total population.

We can use the standard normal distribution (Gaussian with a zero mean and a variance of one) to approximate the p-value in order to get 95 % of confidence that there will remain $p_{row} * n_{rows}$ actual row ids in the sampled column ids. Lahiri et al. [LCM07] studied the normal approximation of the hypergeometric distribution and provided some guidelines to approximate the hypergeometric distribution by the standard normal distribution in finite samples. They showed that the approximation error was rather large. The value of the absolute difference between the standardised hypergeometric distribution $P((X - EX)/Var(X) \leq x)$ and the normal distribution is small, when the percentage of individuals with desired characteristics is maximum and the ratio between the initial population size and the sample size is close to 1. Lahiri et al. found that the absolute error reaches 17.87 % in this case. In our case, the individuals with desired characteristics corresponds to the percentage p of columns that are rows. The ratio between the initial population size and the sample size is here the percentage of sampled columns p_c. However, in our case, p_c will most certainly be smaller than 0.5, because we want to reduce the size by at least a coefficient of 4. By reading the error table 4.2, we can see that the corresponding error is under 10 %.

The table 4.2 comes from their publication and displays the values.

By denoting by n_{cols} the number of columns, and p the percentage of columns that are in the rows (i.e., $n_{rows} = p n_{cols}$), we want to keep p_c ratio of the columns (i.e., $n_{columnsSample} = n_{cols} p_c$), and p_r ratio of the rows (i.e., $n_{rowsSample} = n_{cols} p p_r$). We also want to store each value as a 4 bytes float, so that the total size of the sampled matrix is $size_f$ (final size), such that

$$size_f = 4 n_{columnsFinal} n_{rowssFinal} = 4 n_{cols}^2 p p_c p_r \tag{4.1}$$

Moreover, we want to be sure with high probability (95 %) that there are still $n_{rowsSample}$ rows ids in the remaining $n_{columnsSample}$ columns. We can use the standard normal

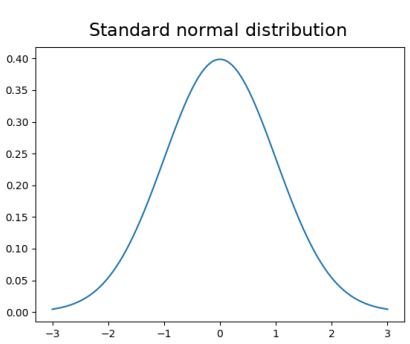
Table 1
Error of Normal approximation

p	$f=0.5$	$f=0.6$	$f=0.7$	$f=0.8$	$f=0.9$
0.5	0.0562	0.0574	0.0613	0.0701	0.0929
0.6	0.0574	0.0593	0.0641	0.0743	0.0995
0.7	0.0613	0.0641	0.0702	0.0822	0.1112
0.8	0.0701	0.0743	0.0822	0.0972	0.1321
0.9	0.0929	0.0995	0.1112	0.1321	0.1787

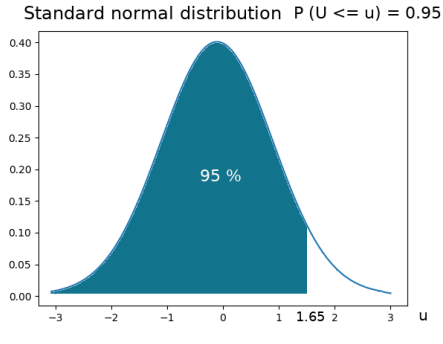
Values of the absolute difference $|P((X-E(X))/Var(X) \leqslant x) - \Phi(x)|$ at $x=0$ for different values of the parameters p and f, where $X \sim \text{Hyp}(n; M, N)$ and $\Phi(\cdot)$ is the cdf of the N(0, 1) distribution. Here, $N = 200$ and $M = Np$ and $n = Nf$.

Figure 4.2: Error table between the standardised hypergeometric distribution and the standard normal distribution from Lahiri et al. [LCM07]

distribution for this purpose ($P(U \leq u)$), using the mean and variance of a hypergeometric model $\mu = \frac{n_h m_h}{N_h}$ and $\sigma^2 = \frac{n_h m_h (N_h - n_h)(N_h - m_h)}{(N_h - 1)N_h^2}$. n_h corresponds to the number of trials, i.e., here the number of remaining columns: $n_h = n_{cols} p_c$. m_h is the number of success items in the population, i.e., the rows in the columns in our case: $m_h = n_{cols}.p$. N_h is the total population, i.e., the number of columns so $N_h = n_{cols}$. We have $\mu = \frac{n_{cols} p_c n_{cols} p}{n_{cols}} = n_{cols} p p_c$ and $\sigma^2 = \frac{n_{cols} p_c n_{cols} p (n_{cols} - n_{cols} p_c)(n_{cols} - n_{cols} p)}{(n_{cols} - 1) n_{cols}^2} = \frac{n_{cols}^2 p p_c (1-p)(1-p_c)}{n_{cols} - 1}$. We also want to have more than $x = n_{cols} p p_r$ remaining rows. The standard normal law gives $u = \frac{x - \mu}{\sigma} = \frac{n_{cols} p p_r - n_{cols} p p_c}{\sqrt{\frac{n_{cols}^2 p p_c (1-p)(1-p_c)}{n_{cols} - 1}}}$. The table of the standard normal law gives us $P(U \leq u) = 0.95 \Leftrightarrow u = 1.65$, with U following the standard normal law, and u the z-score. In our case, we want to have $P(U > u)$. Using the addition probability law, we have $P(U > u) = 1 - P(U \leq u) = 0.95$. As a result, we want to have $P(U \leq u) = 0.05$, which z-score is the symmetric of $P(U \leq u) = 0.95$ about the y axis. As a result, the z-score we are interested in is $u = -1.65$. See Figure 4.3.



(a) Standard normal distribution



(b) Standard normal distribution $P(U \leq u) = 0.95$

Figure 4.3: Standard normal distribution

Thus, we have

$$\frac{x-\mu}{\sigma} = u \tag{4.2}$$

and

$$\frac{x-\mu}{\sigma} = \frac{n_{cols}pp_r - n_{cols}pp_c}{\sqrt{\frac{n_{cols}^2 pp_c(1-p)(1-p_c)}{n_{cols}-1}}} \tag{4.3}$$

$$\begin{aligned}
u &= \frac{p(p_r - p_c)}{\sqrt{\frac{p(1-p)p_c(1-p_c)}{n_{cols}-1}}} \\
u^2 &= \frac{p^2(p_r - p_c)^2(n_{cols}-1)}{p(1-p)p_c(1-p_c)} \\
u^2 &= \frac{p(p_r - p_c)^2(n_{cols}-1)}{(1-p)p_c(1-p_c)} \\
u^2\frac{1-p}{p(n_{cols}-1)}(1-p_c)p_c &= (p_r - p_c)^2
\end{aligned}$$

Using Equation 4.1, we have $p_r = \frac{size_f}{4pn_{cols}^2 p_c}$, we obtain

$$\begin{aligned}
u^2\frac{1-p}{p(n_{cols}-1)}(1-p_c)p_c &= \left(\frac{size_f}{4pn_{cols}^2 p_c} - p_c\right)^2 \\
u^2\frac{1-p}{p(n_{cols}-1)}(1-p_c)p_c^3 &= \left(\frac{size_f}{4pn_{cols}^2} - p_c^2\right)^2 \\
u^2\frac{1-p}{p(n_{cols}-1)}p_c^3 - u^2\frac{1-p}{p(n_{cols}-1)}p_c^4 &= p_c^4 - 2\frac{size_f}{4pn_{cols}^2}p_c^2 + \left(\frac{size_f}{4pn_{cols}^2}\right)^2
\end{aligned}$$

We finally obtain

$$\therefore \left(1 + \frac{u^2(1-p)}{p(n_{cols}-1)}\right)p_c^4 - \frac{u^2(1-p)}{p(n_{cols}-1)}p_c^3 - \frac{size_f}{2pn_{cols}^2}p_i^2 + \left(\frac{size_f}{4pn_{cols}^2}\right)^2 = 0 \tag{4.4}$$

This equation was solved with the optimize library of SciPy, in order to compute p_c and then p_r.

Stratified Sampling Method

The sampled matrix size is upper-bounded by an user-defined size $size_f$ in terms of memory space (mega bytes). The final sampled matrix size will be defined by the number of rows and columns in the sample. We want to define an ideal number of rows and columns that will be kept, so that the sampled matrix size will be lower than $size_f$, but the closest as possible of $size_f$.

We note $\vec{N} = \begin{bmatrix} n_1 \\ n_2 \\ ... \\ n_S \end{bmatrix}$ the vector of size S (S being the total number of strata) containing the sizes of all strata. We note $Nb\vec{C}ols = \begin{bmatrix} n_1.p_{c,1} \\ n_2.p_{c,2} \\ ... \\ n_S/p_{c,S} \end{bmatrix}$ and $Nb\vec{R}ows = \begin{bmatrix} n_1.p.p_{r,1} \\ n_2.p.p_{r,2} \\ ... \\ n_S.p.p_{r,S} \end{bmatrix}$ the vectors of size S containing respectively the number of sampled columns and the number of sampled rows for all strata, with $p_{c,i}$ and $p_{r,i}$ respectively the percentage of kept columns and rows for strata i. Finally, we note $\vec{\mathbf{1}} = \begin{bmatrix} 1 \\ 1 \\ ... \\ 1 \end{bmatrix}$ the one-vector of size S, and $\vec{\mathbf{0}} = \begin{bmatrix} 0 \\ 0 \\ ... \\ 0 \end{bmatrix}$ the zero-vector of size S.

We want to define the $p_{c,i}$ and $p_{r,i}$ for all $i \in [0, S]$ such that the total size with 4 bytes per stored value does not exceed $size_f$, i.e., $\sum_{i=0}^{S} Nb\vec{R}ows_i . \sum_{i=0}^{S} Nb\vec{C}ols_i \leq \frac{size_f}{4}$. We also want to keep at least one column per strata (see Constraint 4.7). The number of kept columns per strata should be representative of the size of the strata, hence proportional to it. For very small strata, we still want to keep at least one column, this could overestimates a bit the representativeness of the strata (see Constraint 4.8). Moreover, we want to guarantee with a 95 % probability that the corresponding kept rows are in the remaining columns. We can use the equation from the simple random sampling case (see Equation 4.3), where we are only interested on the strata scale. We consider $p_{c,i}$, $p_{r,i}$ and n_i instead of p_c, p_r and n respectively. Indeed, we want to be sure that *within* the strata, we keep at least $n_i.p.p_{r,i}$ columns in the rows with 95 % probability. The simple random sampling is a particular case with one unique strata, thus we have Constraint 4.10. Moreover, the number of rows kept will always be lower or equal to the number of kept columns per strata (see Constraint 4.9). Finally, all values in $Nb\vec{R}ows$ and $Nb\vec{C}ols$ should be integers, since they represent a row or columns number. The table of the standard normal law gives us $P(U \leq u) = 0.95 \Leftrightarrow u = 1.65$, with U following the standard normal law, and u the z-score. In our case, we want to have $P(U > u)$. Using the addition probability law, we have $P(U > u) = 1 - P(U \leq u) = 0.95$. As a result, we

want to have $P(U \leq u) = 0.05$, which z-score is the symmetric of $P(U \leq u) = 0.95$ about the y axis. As a result, the z-score we are interested in is $u = -1.65$.

We want to maximize the final sample size within the upper bound $size_f$, i.e., minimize $size_f - finalSampleSize$. In our optimization, we define a minimization function that takes as parameters independent variables. Here, $Nb\vec{C}ols$ and $Nb\vec{R}ows$ are independent variables since they are the values we are looking for.

This leads to the following minimization function

$$f(Nb\vec{C}ols, Nb\vec{R}ows) = \frac{size_f}{4} - \sum_{i=0}^{S} Nb\vec{R}ows_i . \sum_{i=0}^{S} Nb\vec{C}ols_i \tag{4.5}$$

Subject to the following constraints:

$$f(NbCols, NbRows) \leq 0 \tag{4.6}$$

$$\vec{\mathbf{1}} \leq NbCols \leq \vec{N} \tag{4.7}$$

$$Nb\vec{C}ols \propto \vec{N} \tag{4.8}$$

$$\vec{\mathbf{0}} \leq Nb\vec{R}ows \leq Nb\vec{C}ols \tag{4.9}$$

$$\frac{n_{r,i} - p.n_{c,i}}{\sqrt{\frac{p.(1-p).n_s.p_{c,i}.(n_s - n_{c,i})}{n_i - 1}}} = u \tag{4.10}$$

In fact, the $p_{c,i}$ and $p_{r,i}$ are linked. We can thus simplify the constraints and reduce the number of independent variables that have to be optimized.

Constraint 4.10 can be rewritten in the following way:

$$\begin{aligned}
(n_{r,i} - p.n_{c,i})^2 &= u^2.p.\frac{1-p}{n_i - 1}.n_{c,i}.(n_i - n_{c,i}) \\
n_{r,i}^2 - 2.p.n_{c,i}.n_{r,i} + p^2.n_{c,i}^2 &= u^2.p.\frac{1-p}{n_i - 1}.n_{c,i}.(n_i - n_{c,i}) \\
n_{r,i}^2 - n_{r,i}.2.p.n_{c,i} + p^2.n_{c,i}^2 - u^2.p.\frac{1-p}{n_i - 1}.n_{c,i}.(n_i - n_{c,i}) &= 0 \\
n_{r,i}^2 - n_{r,i}.2.p.n_{c,i} + n_{c,i}^2.\left(p^2. + u^2.p.\frac{1-p}{n_i - 1}\right) - u^2.p.\frac{1-p}{n_i - 1}.n_i.n_{c,i} &= 0
\end{aligned}$$

We can solve this second order equation ($\Delta = b^2 - 4ac$ with $ax^2 + bx + c = 0$)

$$\Delta = (-2.p.n_{c,i})^2 - 4.\left(n_{c,i}^2.\left(p^2. + u^2.p.\frac{1-p}{n_i-1}\right) - u^2.p.\frac{1-p}{n_i-1}.n_i.n_{c,i}\right)$$
$$\Delta = 4.p^2.n_{c,i}^2 - 4.\left(p^2.n_{c,i}^2 + n_{c,i}^2.u^2.p.\frac{1-p}{n_i-1} - u^2.p.\frac{1-p}{n_i-1}.n_i.n_{c,i}\right)$$
$$\Delta = -4.\left(n_{c,i}^2.u^2.p.\frac{1-p}{n_i-1} - u^2.p.\frac{1-p}{n_i-1}.n_i.n_{c,i}\right)$$
$$\Delta = -4.\left(u^2.p.\frac{1-p}{n_i-1}.(n_{c,i}^2. - n_i.n_{c,i})\right)$$
$$\Delta = 4.u^2.p.\frac{1-p}{n_i-1}.n_{c,i}.(n_i - n_{c,i}))$$
$$n_i \geq n_{c,i} \Rightarrow \Delta \geq 0$$

There are two solutions

$$n_{r,i,1} = p.n_{c,i} + |u|.\sqrt{\frac{p.(1-p)}{n_i-1}.n_{c,i}.(n_i - n_{c,i})} \tag{4.11}$$

and

$$n_{r,i,2} = p.n_{c,i} - |u|.\sqrt{\frac{p.(1-p)}{n_i-1}.n_{c,i}.(n_i - n_{c,i})} \tag{4.12}$$

However, we need to have $p_{r,i} \leq p_{c,i} \Leftrightarrow n_i.p_{r,i} \leq n_i.p_{c,i} \Leftrightarrow p.n_i.p_{r,i} \leq p.n_i.p_{c,is} \Leftrightarrow n_{r,i} \leq p.n_{c,i}$. Moreover, we have $u = -1.65$. As a result, we obtain

$$n_{r,i,2} = p.n_{c,i} + u.\sqrt{\frac{p.(1-p)}{n_i-1}.n_{c_i}.(n_i - n_{c,i})} \tag{4.13}$$

We should also have $p_{r,i} \geq 0$. $p_{c,i}$ is always positive, but $p.n_{c,i} + u.\sqrt{\frac{p.(1-p)}{n_i-1}.n_{c,i}.(n_i - n_{c,i})}$ may not be. We also want to have integers values, so we can take the ceil value, and then we have

$$\therefore n_{r,i,2} = max\left(\left\lceil p.n_{c,i} + u.\sqrt{\frac{p.(1-p)}{n_i-1}.n_{c,i}.(n_i - n_{c,i})}\right\rceil, 0\right) \tag{4.14}$$

We can reduce the number of independent variables. Since the $p_{r,i}$ and $p_{c,i}$ are linked, so are $Nb\vec{C}ols$ and $Nb\vec{R}ows$. The optimisation function becomes:

$$f(Nb\vec{C}ols) = \frac{size_f}{4} - \sum_{i=0}^{S} max\left(\left\lceil p.Nb\vec{C}ols_i + u\sqrt{\frac{p.(1-p)}{(\vec{N}_i-1)}}\sqrt{Nb\vec{C}ols_i.(\vec{N}_i - Nb\vec{C}ols_i)}\right\rceil, 0\right) . \sum_{i=0}^{S} Nb\vec{C}ols_i \tag{4.15}$$

We want to have a final number of columns per strata that is proportional to the number of initial columns per strata (see constraint 4.8). We can rewrite the $Nb\vec{C}ols$ as a function of a global p_c, such that $n_{c,i} = \lceil max(1, p_c.n_i) \rceil$. The optimisation problem becomes an univariate optimisation problem, where the variable of interest is p_c. This leads finally to the following optimisation function 4.16:

$$min_{p_c} f(p_c) = \frac{size_f}{4} - \sum_{i=0}^{S} max\left(\left\lceil \lceil max(1, p_c.n_i) \rceil + u\sqrt{\frac{p.(1-p)}{(n_i-1)}}\sqrt{\lceil max(1, p_c.n_i) \rceil.(n_i - \lceil max(1, p_c.n_i) \rceil)} \right\rceil, 0\right) . \sum_{i=0}^{S} \lceil max(1, p_c.n_i) \rceil \quad (4.16)$$

subject to the following constraints:

$$f(p_c) \leq 0 \quad (4.17)$$

$$0 < p_c < 1 \quad (4.18)$$

This optimization problem was solved using the optimize library of SciPy, in order to define the number of rows and columns to sample per strata.

4.2 Development of the Tool for the Matrix Visualization

After sampling the connectivity matrix, the sample should be displayed in a visual tool. The purpose of this tool is to help the user in the definition of cleansing operations. The tool should thus provide the user with feedback on cleansing operations applied to the sample, in order to get an idea of how the defined cleansing threshold would operate on the initial data. The current section describes the development of the design of this visual tool.

4.2.1 Design of the Tool

While the content of the sample is interesting to display, the rows and columns ids are also an useful information for the user. However, displaying only the ids gives a poor information to the user. Instead, the hierarchical anatomical information linked to the id should be displayed. The similarity between the rows and between the columns are also interesting to display.

A similarity-based hierarchical clustering is usually displayed as a dendrogram. However, such structures can be costly in terms of space (i.e., it may be too large to display).

The visualisation can become quickly messy. An idea to overcome this problem was to "flatten" the dendrogram using clusters that are on the same hierarchical similarity level, i.e., display on the same line clusters with similarity in the same range, and on the line below, clusters of similarity value in a higher range (with higher similarity). As can be seen in Figure 4.4 with an anatomical hierarchical level of seven, there is a separation between the anatomical and the similarity-based clustering. The depth of the anatomical part represents its position within the anatomical hierarchy. The depth of the similarity part represents the similarity level of the cluster. The deeper the bar, the higher the similarity within the cluster represented. The hierarchical clusters (anatomical or similarity-based) will be the row and column headers of the sampled matrix and will be expandable by clicking on them.

It can also be of interest to see how the elements within the clusters displayed in the similarity hierarchy relate to each other in a single cluster, i.e., how similar the cluster's subelements are. It was thus decided to add a similarity view (view C on the final visualisation Figure 4.5) on the right of the matrix. It follows the same pattern as in MultiLayerMatrix [DCF16], and will display the similarity within the selected cluster. It should be noted that the rows and columns will not have the same hierarchical clustering, since the clusters are based on similarity. A tree hierarchy (view A in Figure 4.5) was also added on the left of the matrix (view B in Figure 4.5). This tree hierarchy displays the names of the anatomical structures and associates them with the corresponding anatomical structure's colour in the row and column hierarchy. The chosen colours are the ones already in use by the neuroscientists for the given anatomical brain levels.

This tool is aimed at defining parameters for matrix cleansing operations, i.e., thresholding, merging, data imputation, and finalisation. The finalisation step corresponds to the cleansing, i.e., the application of the cleansing operations with the chosen parameters. An operation panel (view F in Figure 4.5) was added on the upper right corner, with different tabs: one for each operation. By clicking on an operation tab, the corresponding control panel appears and the current operation is defined as the clicked one. During the thresholding step, the user can select a threshold and apply the operation using the operation panel (view F in Figure 4.5). In order to help the user to visualise the correspondance between the heatmap values and the chosen threshold (view B in Figure 4.5), a small square is displayed next to the threshold value and is coloured in the same colour scale as in the matrix, according to the selected threshold value. Below the operation control panel (view F in Figure 4.5), a 2D projection represents the spatial location of the sampled data (view G in Figure 4.5), i.e., where the neurons represented by the rows and columns are. The voxels on this projection can be coloured according to different rules, either by their anatomical region colour, or by the effect of the current operation on them. For the thresholding, the effect can be which rows or columns were thresholded or to which extent the line or the column was thresholded. For the merging, the effect can be which rows or columns were merged together. For the data imputation, the effect can be the display of the initially missing values.

During the merging step, the user can select a similarity threshold and apply the operation.

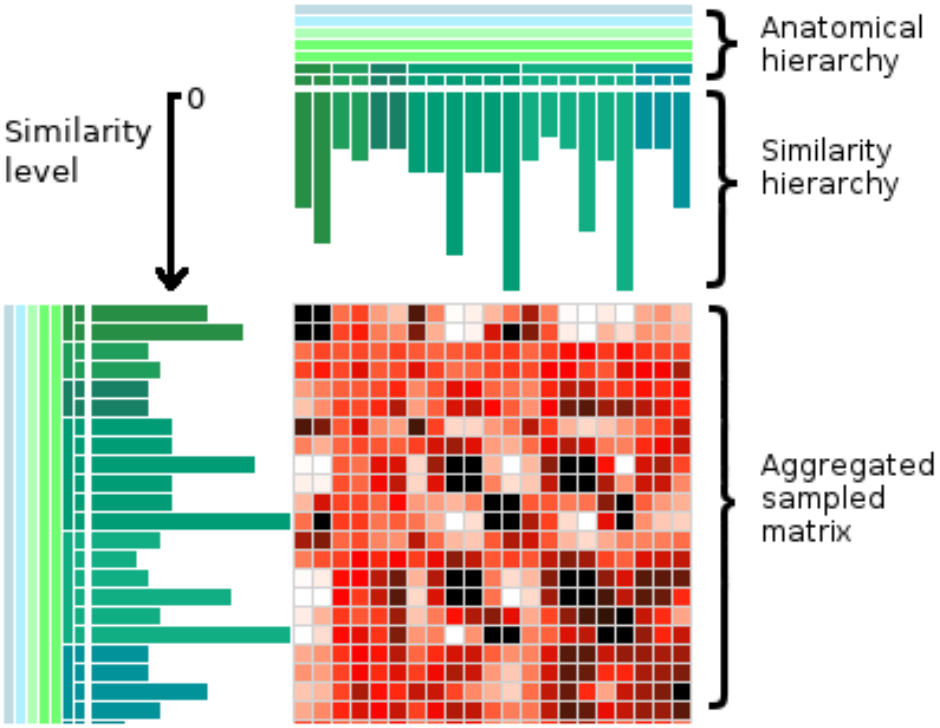


Figure 4.4: Row and column headers for an anatomical level of interest of seven. The header colors are the reference colors of the corresponding anatomical region

The same scheme as for the thresholding is used in order to give him an idea of the similarity value: a small box is displayed next to the similarity threshold input, whose colour is on the same green scale as the values displayed on the similarity view. For the data imputation part, the user can select the way the data should be imputed, i.e., set as zero or set as mean value of the values of groups of neurons belonging to the same anatomical region.

The user can see the effects of his or her operations on the 2D spatial projection (view G in Figure 4.5), but also on the network measures (views H and I in Figure 4.5) and on a note under the similarity view matrix (view C in Figure 4.5) that sums up the effects of all operations. For the thresholding, this note diplays how many voxels were thresholded. For the merging, the note tells how many rows and how many columns were merged and into how many rows and columns. For the data imputation part, it shows the number of missing values (view E in Figure 4.5).

Lastly, a finalisation tab sums up the found settings (threshold, similarity threshold, and data imputation scheme). It allows the user to apply the given settings to the initial matrix, or to get a Python script to apply the operations on the matrix by himself later.

Finally, some network measures of interest are displayed at the bottom (views H and I in Figure 4.5). Some network measures can be very long to compute and are not compatible

with real time use. That is why only the simplest network measures (i.e., node degree and strength) will be displayed in the form of histograms or scatter plots (view H in Figure 4.5), where the colour represent the corresponding anatomical structure. The measures for the initial matrix file, as well as for the sampled matrix and for the sampled matrix after the operations should be displayed to allow comparison.

Several design iterations occurred to get a satisfying visualisation. On the first iteration (see Figure 4.6), the tab bar was not separated from the matrix visualisation and the content of the tab bar panel was rather messy. On the first subfigure (see Subfigure 4.6a), a detail view allows the user to zoom within the matrix. By clicking on a cell on the matrix, the aggregated values of the lower hierarchical level are displayed. The matrix headers' colors of the detail view correspond to the clicked cell of the matrix. However, it seemed redundant since the user can click on the row and column clusters to expand them. A chord diagram is located below this detail view and represents the different relations between the anatomical regions, i.e., how these anatomical regions are connected (strong lines for strong relations). However, chord diagrams are quite heavy to compute and might be superfluous. As such they are not present in the final iteration. In the second subfigure (see Subfigure 4.6b), the rows and columns of the matrix are ordered by similarity. A similarity view replaces the chord diagram, because chord diagrams do not make sense without anatomical regions. The matrix headers' colors of the similarity view correspond to the clicked cluster. This similarity view displays how the clusters of the lower hierarchical level are similar to one another. The diagonal values represent the within similarity of each cluster. On both subfigures, the threshold and similarity threshold are set through a slider. This was changed to an input field, since the user most of the time expects the view to change automatically while the slider is changed, whereas, due to the heaviness of the operations to perform this cannot be made in real time (the similarity clustering has to be recomputed after every operation). Moreover, the finalisation tab was missing. Finally, the spatial projection views were too large and actually only one projection view is really needed. Thus, on the next iterations, only one projection view remained.

The next iteration (see Figure 4.7) displays almost all features that were described above. Minor changes were made, such as changing the matrix colour scale to a white to red scale instead of the blue to red colour scale. Indeed, this can be misleading and make the user think the values are negative. A checkbox allows the user to display the matrix in a log scale colour in order to better visualise the values. It was also decided to base the colour scale of the matrix and the corresponding legend bar based on the maximum and minimum view of the aggregated matrix, because otherwise the colour visualization would be poor. Indeed, the maximum value of the actual sample is way higher than the one of the aggregated matrix since the aggregation is aggregated based on the mean. Finally, in order to get an even better visualization colour showing more contrast, instead of taking the maximum value of the aggregated matrix, the 95 % percentile of the sampled matrix was used, which clearly improved the visualization: instead of having mostly almost white cells and a few red ones, the repartition within the red scale was way better.

The final iteration (see Figure 4.8) has minor changes compared to iteration 2. The matrix colour changes, and another 2D view is added on the network measures panel (view I in Figure 4.5), where the measures can also be displayed on a spatial view. The intensity of the measures are encoded in grayscale. All mockups can be found in the annexes.

4.2.2 Clustering

As described in the previous subsection, two types of clusterings are performed in order to aggregate columns and rows: the anatomical clustering and the similarity-based clustering. Both are described in the next paragraphs. This clustering is used for the aggregation of the sampled matrix in order to display it on the user screen, but also for the display of the row and column headers.

The Anatomical Clustering

The anatomical clustering is based on the brain hierarchy and is performed to some extent during the stratified sampling algorithm: as the strata represent anatomical structures at the level of interest, the anatomical clustering is done until the hierarchical level of interest. The columns can be fully ordered according to the lowest hierarchy resolution, but the rows cannot as they are added on the strata of the hierarchy level of interest on the fly to the reservoir. They could be if the matrix was squared or if we knew in advance the row ids. The strata of the hierarchy level of interest are ordered, but within these strata, the rows are not ordered according to lower hierarchical levels. A reordering post-processing scheme was implemented to reorder the rows in the strata. However, depending on the size of the strata, there can be a memory overhead.

Straightforward Solution for the Similarity Clustering

The most straightforward approach to cluster based on similarity would be to apply a basic hierarchical clustering (i.e., linkage from scipy.cluster.hierarchy [lin]). The hierarchical clustering function from the Python library fastclustering does not seem to provide better results (in terms of computational time) than the SciPy function.

Different methods can be applied. In the nearest point algorithm, the distance between two clusters corresponds to the minimum distance between two points belonging to these clusters. For the furthest point algorithm, the distance between two clusters corresponds to the maximum distance between two points belonging to these clusters. The UPGMA algorithm [Sok58] is another option, where the distance between two clusters corresponds to the average distance between two points belonging to these clusters. The UPGMC algorithm [ELLS11] is another option, where the distance between two clusters corresponds to the Euclidean distance between the centroids of the clusters, and the centroid of the newly formed cluster is recomputed over all points of the new cluster. The WPGMC algorithm is similar to the UPGMC algorithm [ELLS11], but the centroid of the new cluster is computed as the average of the centroids of the merged clusters.

Finally, the incremental algorithm uses the Ward variance minimization algorithm [WJ63] to compute the distances between clusters. All these methods are implemented in the linkage package.

Different metrics can be applied, such as the Euclidean or the cosine distance. The later seems better in our case, as the rows and columns can be seen as high dimensional points.

However, a problem with the hierarchical clustering is its complexity in $O(N^2)$, which makes it not feasible with large datasets. As an example, clustering a 350 MB reduced matrix takes around 70 seconds. Moreover, it creates a large memory overhead (around 1200 MB of RAM are then required), while it is only the clustering part.

Thus, it is often advised to employ a two-phase clustering. A fast algorithm is used in the first place to split the large dataset into smaller similar subsets, and then hierarchical clustering is applied on the subsets. K-means is often used as first algorithm, because it is rather fast. However, it still needs some time to be computed, especially if we want to split the dataset into many subsets.

Our data has a particular structure. Indeed, it represents connectivity in the brain, and groups of neurons within brain regions are usually more similar than to groups of neurons of different brain regions. Instead of using a clustering algorithm, we can also rely on the brain regions to create subsets on which to apply hierarchical clustering, and then combine the resulting clusters in order to get a final hierarchy.

In order to speed up the computation, instead of taking a pure similarity-based clustering, we can use a hybrid anatomical - similarity approach. The user defines a hierarchical level on which to compute the hierarchical clustering. Above this level, we keep the anatomical hierarchy. Below this level, we compute hierarchical clustering. This will save computation time while also having a real meaning. Indeed, the user may not want to merge rows or columns of the sampled matrix if they are coming from very different regions.

During tests, it was discovered that the UPGMC algorithm is far slower than the UPGMA algorithm, clustering 17,164 columns (558 rows around 37 MB of final size) was ten times slower with the UPGMC algorithm as with the UPGMA algorithm. Thus, the UPGMA algorithm should be preferred. The reason is the following (from SciPy documentation, see [lin]): "For method 'single' an optimized algorithm based on minimum spanning tree is implemented. It has time complexity $O(n^2)$. For methods 'complete', 'average', 'weighted' and 'ward' an algorithm called nearest-neighbours chain is implemented. It also has time complexity $O(n^2)$. For other methods a naive algorithm is implemented with time complexity $O(n^3)$".

Other Clustering Approaches

The CURE algorithm [GRS98] seemed promising at the beginning of this thesis. In fact it is more an aggregation approach based on hierarchy than a hierarchical clustering stricto sensu (i.e., it requires a number of final clusters and does not output the intermediate

lower hierarchical clusters). There is a pyclustering package [Nov18]) available. However the CURE algorithm from this package is very long to compute (20 minutes for a 350 MB matrix). It also triggers a huge memory overhead (up to almost 4 GB RAM for a 350 MB matrix). It does not output the intermediate clusters and requires a final number of clusters. As a result, this approach did not fit our needs and was discarded.

Final Hierarchical Clustering Approach

Finally, it was decided to use the correlation distance metric. This is the Pearson correlation, which measures the angle of expression vectors for genes or groups of neurons A and B around their mean expression level, thus emphasizing both over- and under-expression of genes (see the measures of similarity [sim]).

4.2.3 Final Visualisation Design

The final visualisation tool can be seen in Figure 4.9. In this example, the gene connectivity matrix was sampled to a final size of 3 GB at the sixth anatomical hierarchical level. The screenshot was taken at the beginning, before defining any cleansing thresholds.

4.3 Implementation of the Cleaning Operations

For all the cleaning operations, two approaches had to be developed. Rather simple ones to apply on the sampled matrix, and more complicated ones to apply on the initial matrix. After each cleansing operation on the sampled matrix, the hierarchical clustering is recomputed.

4.3.1 Coding the Thresholding Methods

Thresholding the Sampled Matrix

Thresholding the sampled matrix is a trivial task. All values that are under the given threshold need to be set to zero. The number of values that are set to zero (and were not zeros) as well as the current number of zeros in the matrix can be of interest for the user, in order to know how many values were changed and to evaluate how much the initial matrix will be compressed after the cleansing.

Thresholding the Initial Matrix

Thresholding the initial matrix is not as straightforward as thresholding the sampled one, for which we can use NumPy for the matrix stored in RAM. The initial matrix has to be read line by line, these lines have to be parsed and thresholded before writing them back in a new temporary file.

4.3.2 Coding the Merging Methods

Merging the Sampled Matrix

In order to merge the sampled matrix, the user inputs a similarity threshold between zero and one. Then, the SciPy [JOP] hierarchical row and column clusterings are explored recursively to gather the indices of all row and column clusters with similarity above the given similarity threshold. These clusters are then merged together by mean aggregation, row and column wise.

Merging the Initial Matrix

Merging the initial matrix requires the computation of a hierarchical clustering, which cannot be performed if the matrix is not completely in memory. The matrix can only be read line by line and only a few lines can be stored in memory. Thus, a strategy had to be developed to handle the computation of the hierarchical clustering. It is based on the same scheme as the hybrid anatomical / similarity clustering principle: cluster according to the anatomical hierarchy until the anatomical level of interest and then apply similarity clustering in the anatomical hierarchy clusters of the anatomical level of interest. The row and column indices are first preprocessed in order to get their anatomical regions. As the matrix can only be read line by line, the row and column steps have to be performed separately. The row ids are first grouped by anatomical regions of an adaptive level given a maximum number of rows per region parameter. The region will be the region of highest anatomical level that has less rows than the maximum number of rows, or the region of lowest anatomical level if there are more rows than the maximum number of rows. Fixing a hierarchical anatomical level can lead to way too large regions on which the similarity computation becomes infeasible.

Once the regions and corresponding indices are computed, for each region, the matrix will be read to gather the rows belonging to the current region. Then, the hierarchical clustering will be computed on these rows. The merging will be applied based on this clustering. The clustering will be used to gather the clusters with similarity above the given similarity value. Then the rows belonging to one cluster will be merged by mean aggregation, and the row id of the first row in the cluster will be the index of the aggregated cluster. The row ids are stored while the merged rows are written in a temporary file. However, the ordering of the rows is then changed, so for consistency we keep the position of the first row of all clusters so that to reorder them according to the initial order afterwards.

Then, in order to apply the column hierarchical clustering, the matrix is transposed, and the same procedure is applied. The lines (representing the columns) are then reordered according to the initial column order. The matrix is transposed again and the matrix is ordered according to the initial row order. The ordering was chosen to be in the last steps since as the merging reduces the size of the matrix, it is more computationally efficient to do it this way.

4.3.3 Coding the Imputation Methods

There was not enough time to develop a data imputation method. Moreover, the matrices that were used for the development and for testing did not contain missing values, only missing lines in the case of the structural connectivity.

The suggested solution was presented in the current chapter. Implementation details were given to fully explain how the sampling was performed and how the visualisation as well as the final cleansing work. The next chapter presents a critical reflection about the results.

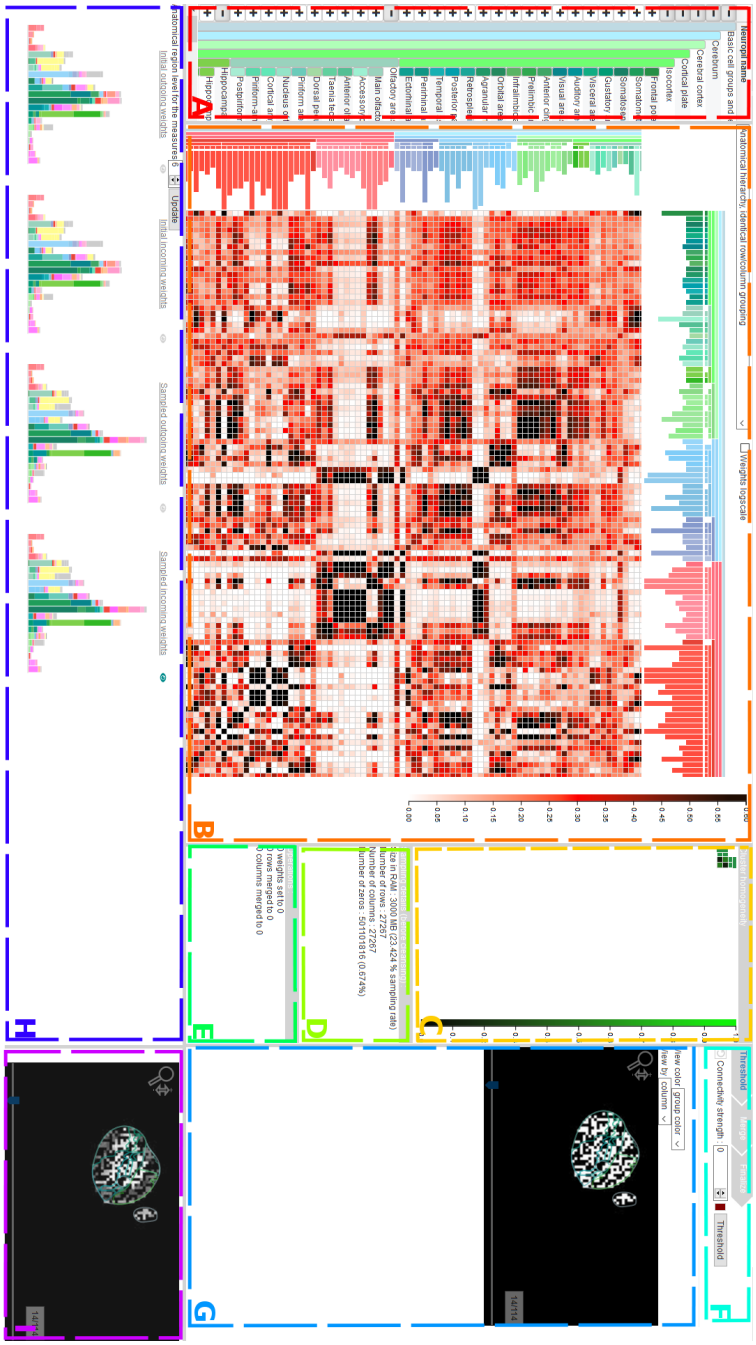
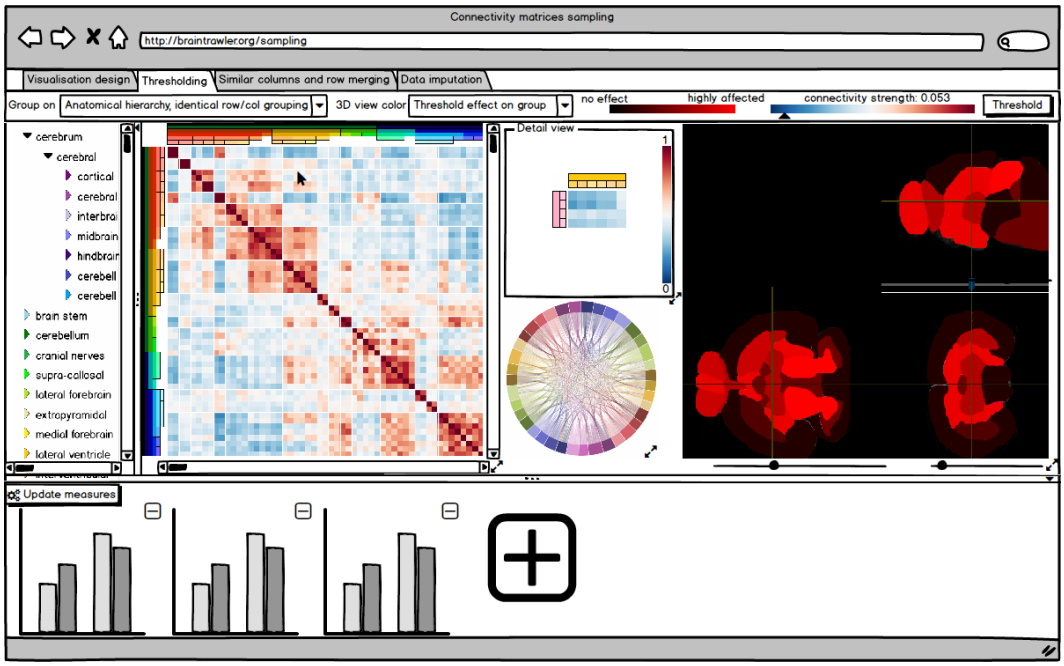
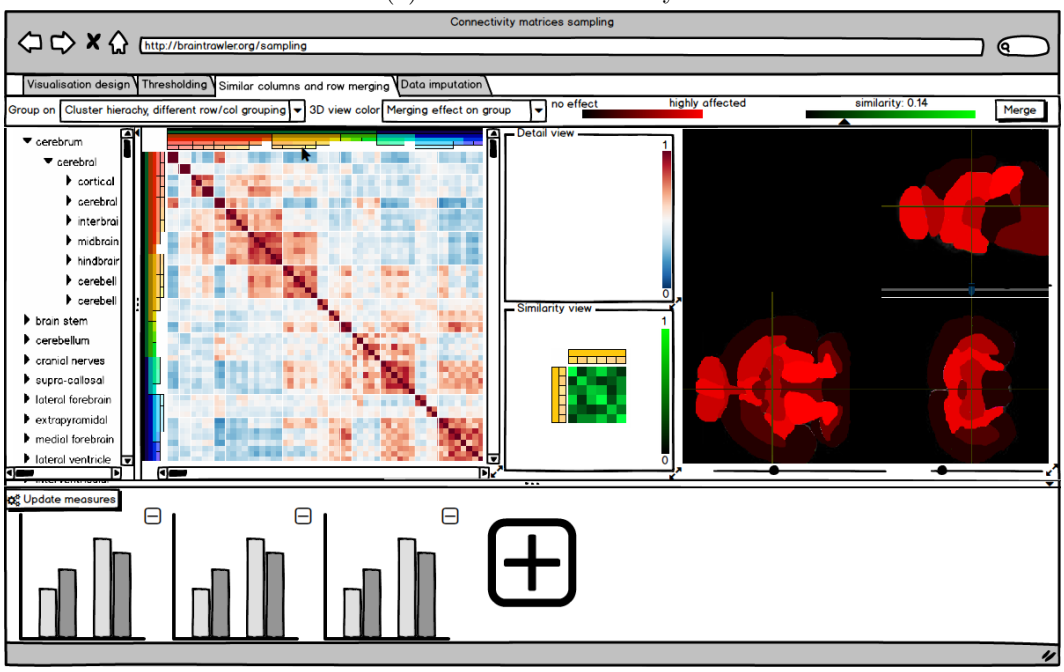


Figure 4.5: Final visualisation tool



(a) Anatomical hierarchy



(b) Similarity hierarchy

Figure 4.6: Mockups, Iteration 1

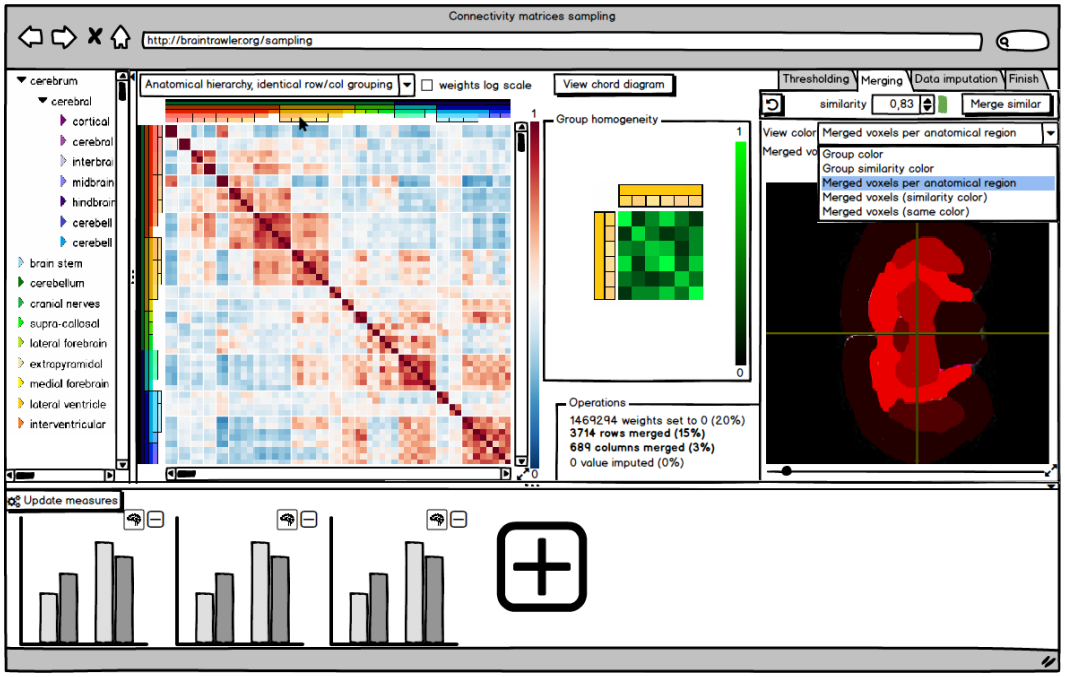


Figure 4.7: Mockups, Iteration 2, after the merging of similar rows and columns, matrix ordered by the anatomical hierarchy

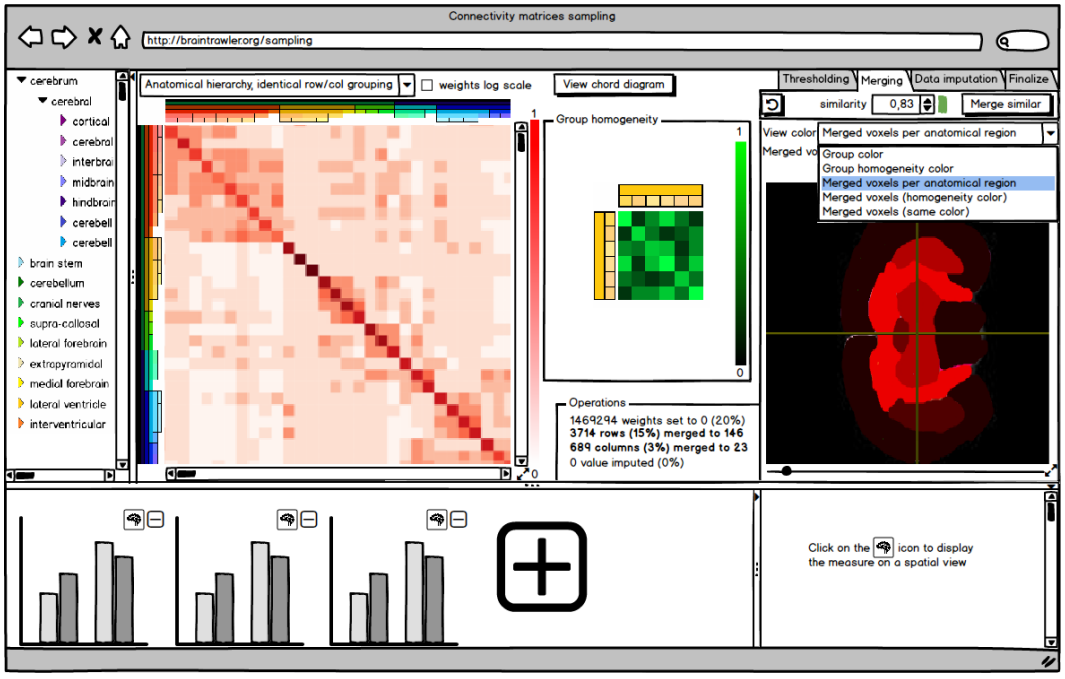


Figure 4.8: Mockups, Iteration 3, after the merging of similar rows and columns, matrix ordered by the anatomical hierarchy

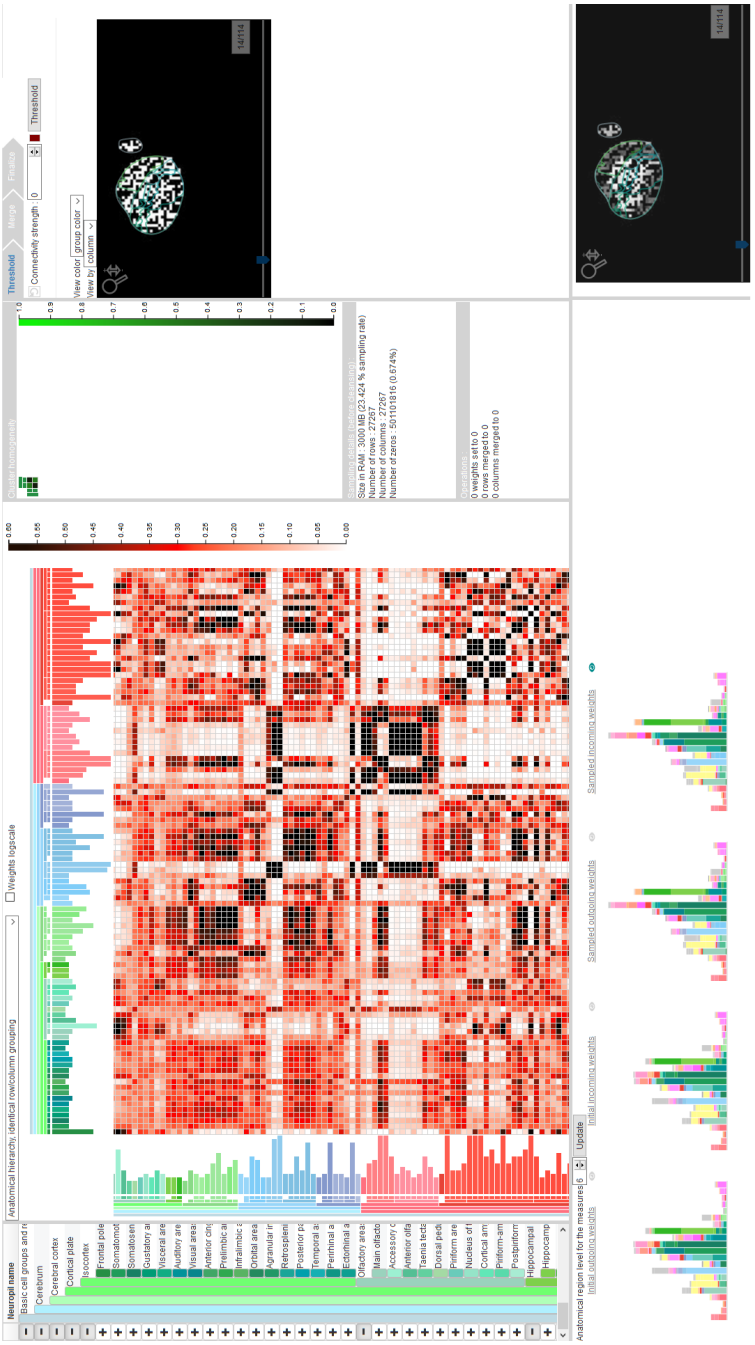


Figure 4.9: Final visualisation tool with legend

CHAPTER 5

Critical Reflection

The tool was developed according to what was presented in Chapter 4. The current chapter presents a critical reflection on the results of the developed tool. Section 5.1 describes how the tool was evaluated. Section 5.2 provides a discussion about the results, and summarizes the opened issues. Section 5.3 puts the results in relation to the literature.

5.1 Evaluation

5.1.1 General method

At first, it was planned to use the brain connectivity toolbox [RS10] for evaluation. However, some metrics took too long to compute on such large matrices. The PAGANI Toolbox [DXZ+18] is a new tool, which uses GPU to compute some interesting measures in the brain connectivity context, and is rather fast. As a result, this toolbox was used for the evaluation. However, not all interesting measures are implemented in this toolbox, so only the ones that are available were used for evaluation, namely betweeness centrality, clustering coefficient, characteristic path length, degree and node efficiency. The characteristic path length is inversely linked to the node efficiency. The node efficiency seemed more interesting to explore, as it is a nodal distribution while the other one is only a global value. As a result, the characteristic path length was not studied.

However, most of these measures require a squared matrix, and only the gene connectivity matrix was squared. The evaluation was thus performed only on this matrix.

The evaluation was performed following for different sample sizes (namely 2 GB, 1 GB, 500 MB, and 250 MB) and with different anatomical hierarchy levels. The tool was used in order to define a threshold and a merging criterion. Usually, the threshold and merging criterion were rather similar for all sample sizes and anatomical hierarchy levels. Then, cleansed versions of the sampled matrices as well as of the initial matrix were generated

using the defined threshold and the merging threshold. The network measures were computed on all matrices: initial matrix, sampled versions, cleansed sampled versions, and cleansed initial versions. The measures distributions were compared in order to analyse how the tool performed. For the evaluation and the use of the PAGANI toolbox, the cleansed versions had to be harmonized with regard to the similarity grouping. The cleansed version contains only one row for all rows that were merged together, but this row represents several other rows. In order to compute measures, we need to take all these rows into account. Thus we had to duplicate the row as many times as the number of rows represented by the current row (i.e., harmonize). However, due to time issues, only the not-harmonized versions measures could be computed for the initial matrix.

At first, it was planned to use the Kolgomorov-Smirnov test in order to compare the distributions. The null hypothesis is rejected for this test if $D_{n,m} > c(\alpha)\sqrt{\frac{n+m}{nm}}$, with α the required alpha level, $c(\alpha) = \sqrt{-\frac{1}{2}ln(\frac{\alpha}{2})}$, $D_{n,m} = sup_x|F_{1,n}(x) - F_{2,m}(x)|$, with $F_{1,n}$ and $F_{2,m}$ being the empirical distribution functions of the first and second samples of size n and m respectively. Hence, $c(\alpha)$ ranges between 1.22 and 1.95 for $\alpha \in [0.1, 0.001]$. In our case, n and m (the sizes of the samples) are very large, hence $c(\alpha)\sqrt{\frac{n+m}{nm}}$ is small and if there is a point where the distributions are not that similar while the rest is similar, the value of the Kolgomorov-Smirnov statistic $D_{n,m}$ will reject the null hypothesis and we would conclude that the two samples do not come from the same sample. For instance, if we generate a random sample of 8000 elements between 0 and 1000 a and have $b = a * 1.05$ (hence b being very similar to a), the p-value equals 6.9267169858876584e-08 (ks_2samp function from SciPy). Hence the hypothesis will be rejected while the two samples were very similar (only 5 percent of difference). Thus, due to the large sample sizes, this test was not adapted and led to wrong results.

As a result, instead of using this test, a correlation between initial and modified matrix measures was computed. These measures are nodal, so the results of the measures are vector of size the number of rows in the matrix. However, the correlation needs inputs with same size, and the initial and modified matrix do not have the same number or rows, so the measures for the initial and modified matrix do not have the same size. The chosen solution to overcome this problem was to randomly sample the distribution vector of the measure of interest of the initial matrix without replacement a high number of times (1000 times) to be of the same size as the distribution vector of the measure of interest of the sample matrix, and then take the mean of all correlation values. Then, the Spearman correlation was computed. However, the order of the nodes in the initial and modified matrices may differ. The values were therefore ordered before computing the correlation. The mean and standard deviation also seemed interesting to compute for the modified matrix.

As a result, the following procedure was applied

- apply the current measure to the initial and to the modified matrices, resulting in the vector measures I and M for the initial and modified matrices respectively,

- if the initial matrix measure vector I is bigger than the modified matrix measure vector M, sample randomly without replacement 1000 times the vector I to I', with I' being the same size as M. Otherwise set I'=I,
- sort M to M_{sorted} and I' to I'_{sorted} according to their values,
- compute the Spearman correlation S between M_{sorted} and I'_{sorted},
- if the initial matrix measures vector I was bigger than the modified matrix measure vector M, take the mean of all Spearman correlation for all I' as final value,
- compute the mean and standard deviation for the initial and modified measure vectors I and M,
- plot the normalized histograms of the vector measures M and I.

It was planned at first to evaluate the method on both the mouse's gene connectivity matrix and a small squared version of the mouse's structural connectivity matrix. However, due to lack of time and computational resources, the method was only evaluated on the mouse's gene connectivity matrix. The mouse's structural connectivity matrix had indeed several problems. First, it was not squared, so rows with zeros had to be added to get a squared matrix for the evaluation, while these are not the real values. Second, the matrix is not symmetric, and the PAGANI Toolkit only provides results row wise, so the matrices should then have been transposed to also get the column-wise results, and due to lack of time, this could not be performed.

Using the sampling tool, it was observed that a good threshold value is between 0.29 and 0.32, while a good similarity threshold is between 0.995 and 0.999. A similarity threshold of 0.9995 led to even better results, but then only a few rows and columns were merged. These values were thus used for generating the cleansed initial versions, as well as the cleansed sampled versions.

The following subsections describe the different evaluation steps. First, the sampling method should be evaluated, i.e., we want to know if the sample displayed in the tool represents well the initial matrix. Then, the threshold and merging criterion found on the sample should be evaluated, i.e., we want to know if the cleansed version of the sample still represents well the initial sample data. Finally, the cleansing of the initial matrix should be evaluated, i.e., we want to know if the cleansed version of the initial matrix still represents well the initial data.

5.1.2 Initial Matrix - Sampled Matrix Evaluation

This subsection evaluates the sampling method. The comparison of the initial and sampled matrix evaluates how well the sampling preserves the intrinsic characteristics of the initial matrix, and how the sampling affected these characteristics. This evaluation can help to define a good compromise in terms of size ratio between initial and sampled

matrices and characteristics preservation. Two parameters could be tuned, i.e., the sample size and the anatomical hierarchy level.

According to the results, the anatomical hierarchy level was not a very important parameter and only had a small effect. It is thus recommended to use a high anatomical hierarchy level, so that the similarity computation is fastened.

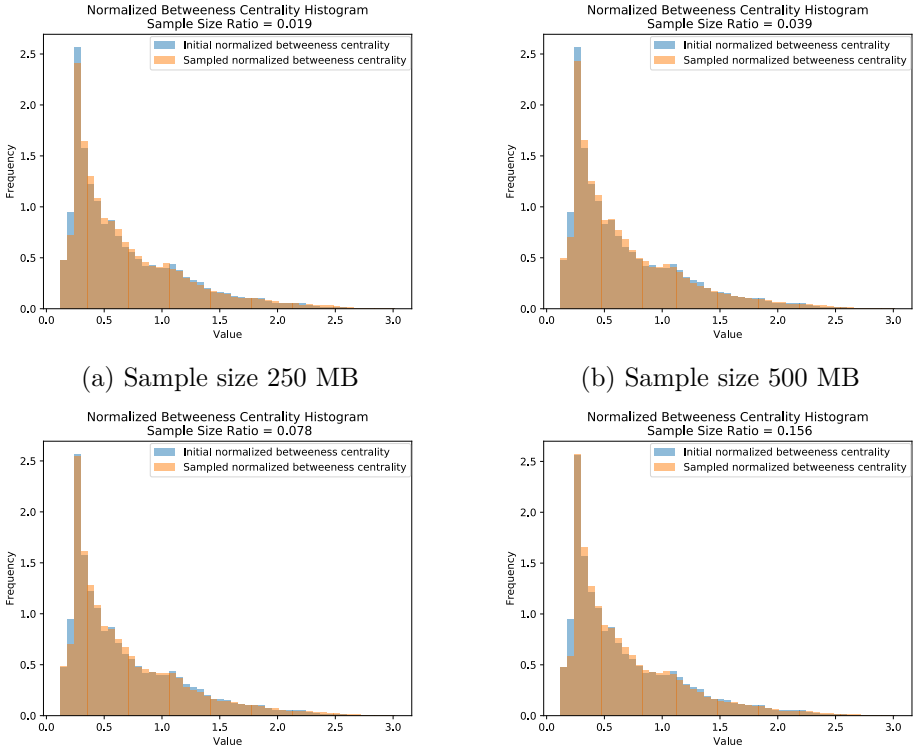
The sample size and hence size ratio between initial and sample matrices had a rather important effect. The bigger the sample size, the better characteristics preservation.

Betweeness Centrality

The distribution of the normalized betweeness centrality was highly similar with the initial betweeness centrality for all sampling sizes (see Figure 5.1). Figure 5.1 plots the normalized betweeness centrality histogram for different size ratio, i.e., the betweeness centrality divided by the number of nodes for different sample sizes (the size ratio being the ratio between initial matrix size and sample matrix size). The Spearman Correlation of the betweeness centrality was rather high and lied between 0.9999999938 (250 MB sampling size) and 0.9999999979 (2 GB sampling size). The betweeness centrality actually depends on the current number of nodes and the matrix size is proportional to the squared number of nodes. As can be seen on the Figure **??**, there is a square root correlation between the size and the betweeness centrality mean on the initial and sampled matrices. The sampled and initial means were normalized so that they can be compared. The error rate of the normalized mean was computed and is small: under 0.35 % for a sample size of 250 MB, and a bit decreasing with the size (see Subfigure 5.2b). The variance was also interpolated and led to a quasi linear interpolation (see Subfigure 5.2c). The betweeness centrality distribution was divided by the number of nodes (i.e., rows) before computing the variance (normalized variance). The results of the variance were not as good as the ones for the mean though, and the error rate of the normalized variance led to strange results where the error rate oscillates between 8 % and 9.5 % and was the best for a sample size of 1 GB (see Subfigure 5.2d). These results may be due to the rather low number of sampled matrices (for each final size, there was one sampled matrix per anatomical hierarchical level).

Clustering Coefficient

The distribution of the clustering coefficient is also rather similar between the initial and the sample matrices, though not being as good fitting as for the betweeness centrality distribution. The best fit seems to be the one with the sample size of 2 GB, but the difference is not very important (see Figure 5.3). The mean and variance of the sampled matrices seem close to the reference one (initial matrix). The error rate of the mean lies under 0.4 % for all values (see Subfigures 5.4a and 5.4b). The error rate of the variance lies under 3.5 % (see Subfigures 5.4c and 5.4d). The Spearman Correlation value is not as good as for the betweeness centrality (see Subfigure 5.4e).



(a) Sample size 250 MB

(b) Sample size 500 MB

(c) Sample size 1000 MB

(d) Sample size 2000 MB

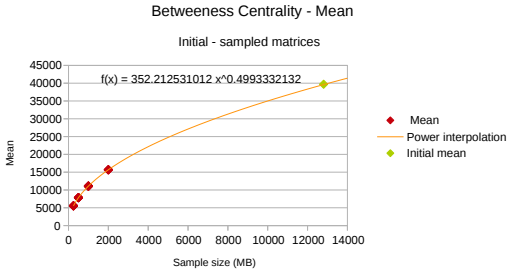
Figure 5.1: Normalized betweeness centrality distribution comparison

Degree

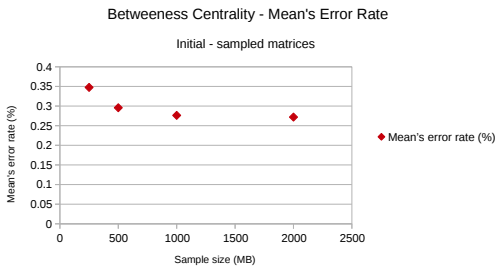
The normalized degree distribution of the sampled matrices looks rather similar to the initial one, though a bit right-shifted (see Figure 5.5). As for the betweeness centrality, there is an almost square root interpolation between mean degree and sample size, which fits rather well the initial's mean normalized degree (see Subfigure 5.6a). The error rate of the normalized mean lies under 0.9 %, which is not as good as for the other measures but still rather good (see Subfigure 5.6b). However, the initial variance is not fitted that well by the interpolation (see Subfigure 5.6c). This bad fit was also illustrated in the error rate of the normalized variance: the error rate lies between 20 % and 24 %, which is very high (see Subfigure 5.6d). The Spearman Correlation is rather similar to the one of the clustering coefficient.

Node Efficiency

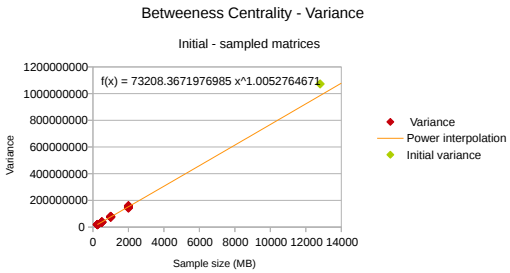
The node efficiency measure was normalized by the PAGANI Toolkit. However, as can be seen on the comparisons of the node efficiency distribution with the initial matrix's node efficiency distribution, the sampled node efficiency distributions are right-shifted in



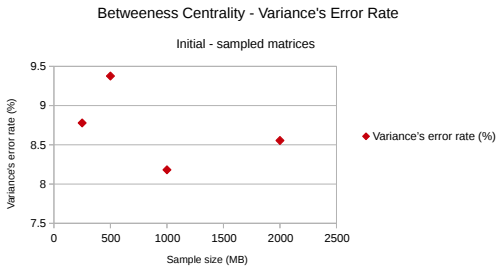
(a) Mean



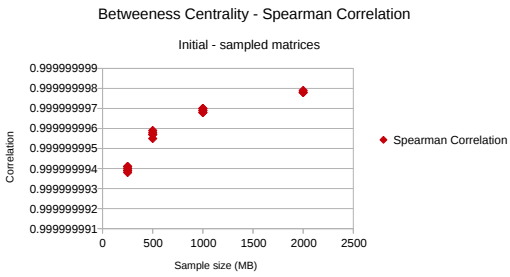
(b) Normalized Mean's Error Rate



(c) Variance

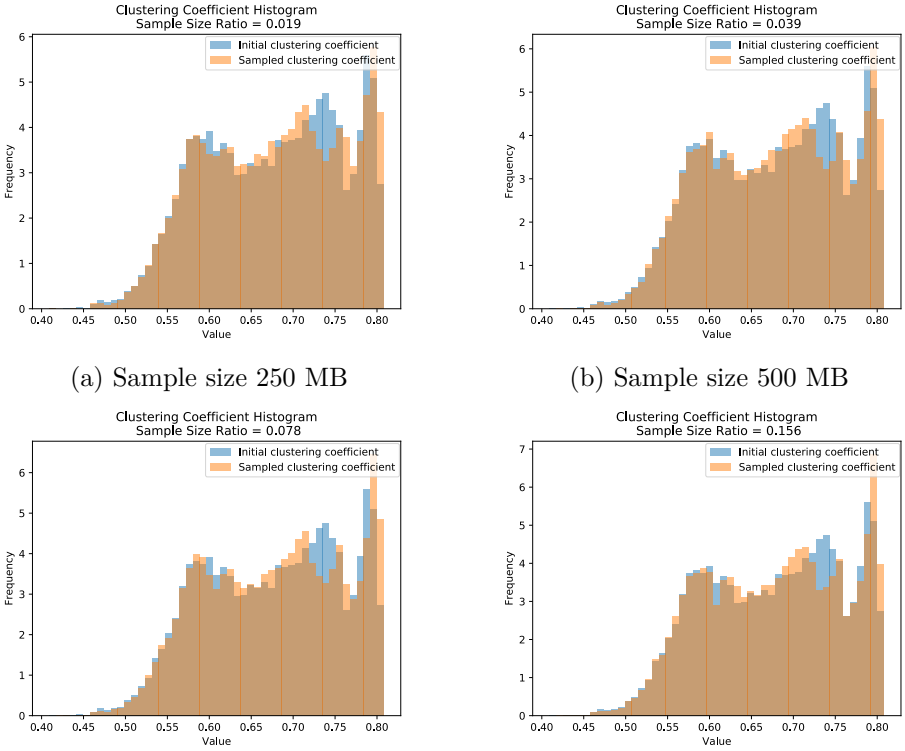


(d) Normalized Variance's Error Rate



(e) Spearman Correlation

Figure 5.2: Betweeness Centrality of initial and sample matrices



(a) Sample size 250 MB

(b) Sample size 500 MB

(c) Sample size 1000 MB

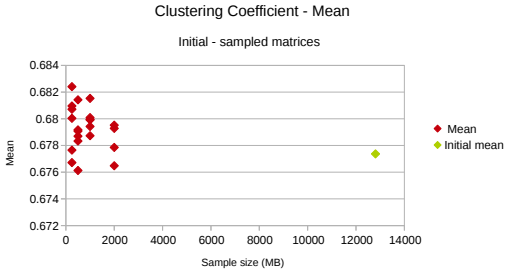
(d) Sample size 2000 MB

Figure 5.3: Clustering coefficient distribution comparison

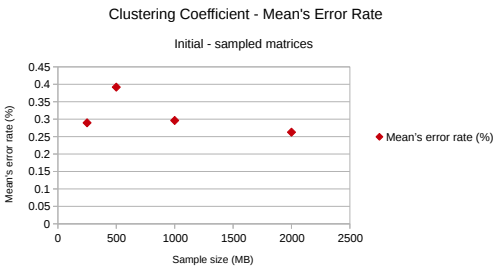
comparison with the initial matrix (see Figure 5.7). However, the overall distribution shape remains very similar. This right-shift can also be found in the mean value (see Subfigure 5.8a) and in the error rate of the mean, which lies between 1.5 % and 3 % (see Subfigure 5.8b). As for the variance, the sampled variance is a bit higher than the initial one (see Subfigure 5.8c). Its error rate lies between 10 % and 15 %, which is rather high (see Subfigure 5.8d). The Spearman Correlation lies above 0.999999993 for all sizes between 250 MB and 2 GB (see Subfigure 5.8e), and is slightly better than the one for the degree and clustering coefficient measures.

Summary

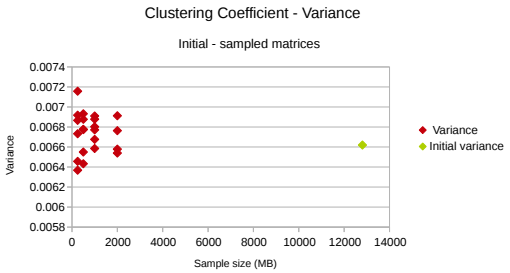
As can be seen on the previous figures, the sampling sizes of 250 MB and 500 MB are too small, even if the measures distributions are somehow similar to the original one. The sampling size of 1 GB (initial size of 12 GB) is already better and 2 GB is even better in terms of characteristics preservation. Above 2 GB sampling size, depending on the anatomical hierarchical level, we can reach the 12 GB RAM limit that was set as criterion for the cleansing tool. This is due to the fact that the similarity computation requires the storage of large working matrices by the SciPy library. The higher the hierarchical



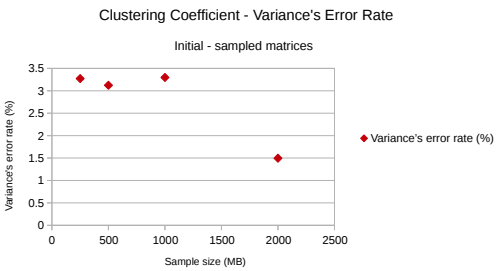
(a) Mean



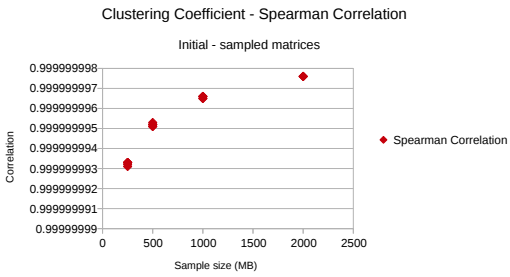
(b) Mean's Error Rate



(c) Variance

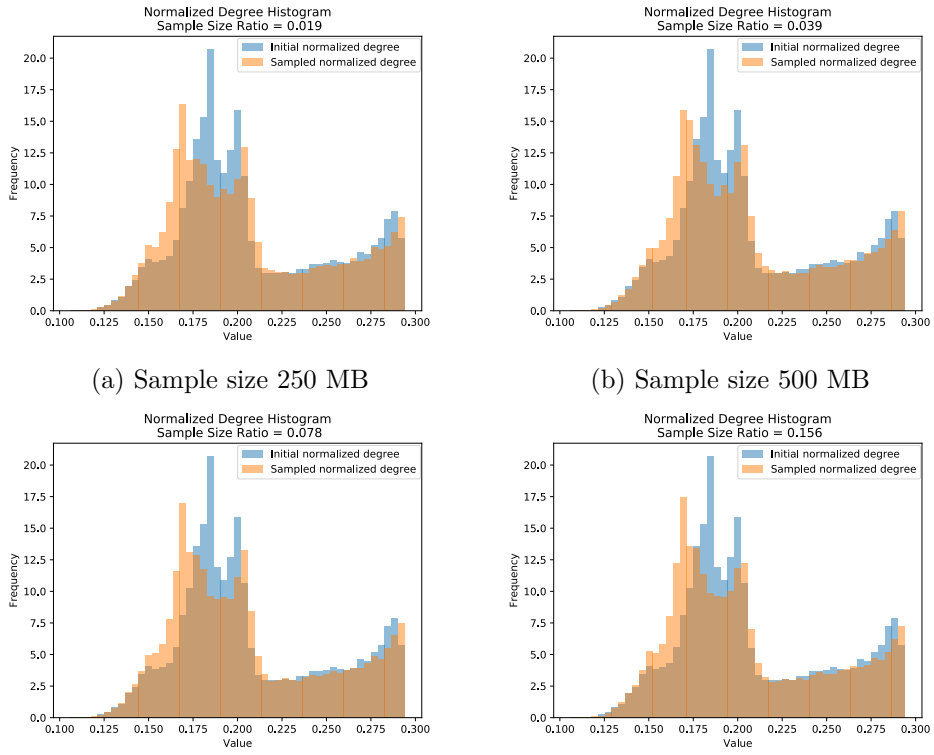


(d) Variance's Error Rate



(e) Spearman Correlation

Figure 5.4: Clustering Coefficient of initial and sample matrices



(a) Sample size 250 MB

(b) Sample size 500 MB

(c) Sample size 1000 MB

(d) Sample size 2000 MB

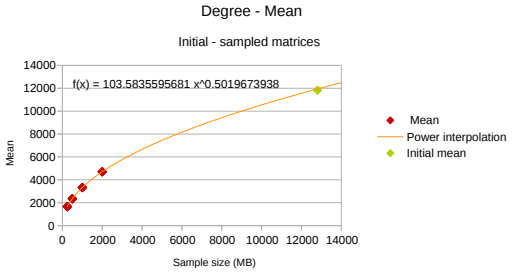
Figure 5.5: Normalized degree distribution comparison

level, the smaller the working matrices and the lower is the computation time. It is thus recommended to use a high anatomical hierarchy level of interest (six or more), with a sampling size of at least 1 GB, 2 GB being even better.

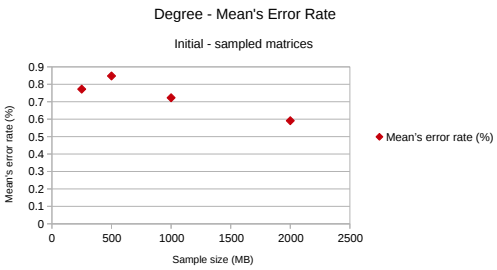
5.1.3 Sampled Matrix - Cleansed Sampled Matrix Evaluation

This subsection evaluates how the chosen thresholds (threshold and similarity threshold) affected the sampled matrix. The thresholding step was chosen so that it did not change the data much, and the difference between the tested thresholds (0.29 to 0.32) was barely visible. On the other hand, the similarity threshold (0.995, 0.999, and 0.9995) had a large influence on the matrix. While a similarity threshold of 0.9995 did not change the data much, a similarity threshold of 0.995 had a more tangible impact on the intrinsic characteristics of the matrix.

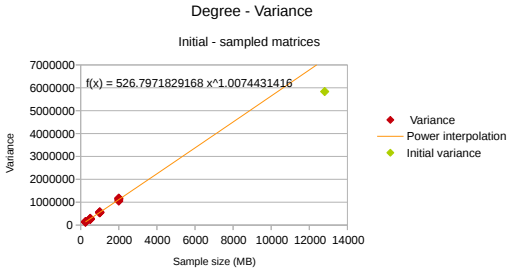
All sampled matrices of the previous step were thresholded and grouped according to the aforementioned thresholds. To compare the results, the mean and variance of the different measures of the sampled cleansed matrices were compared with the mean and variance of the different measures of the corresponding sampled matrices, via an error rate. The



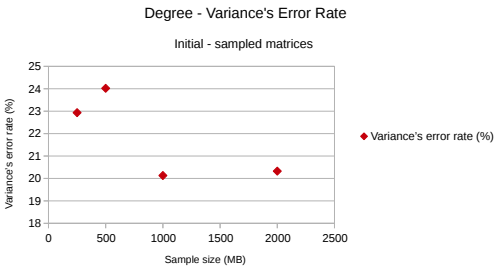
(a) Mean



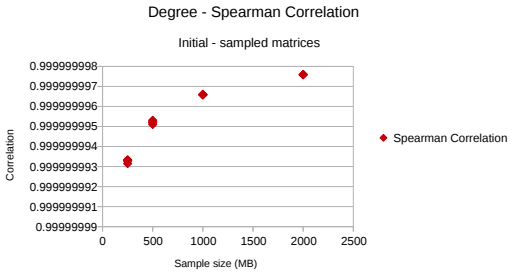
(b) Normalized Mean's Error Rate



(c) Variance

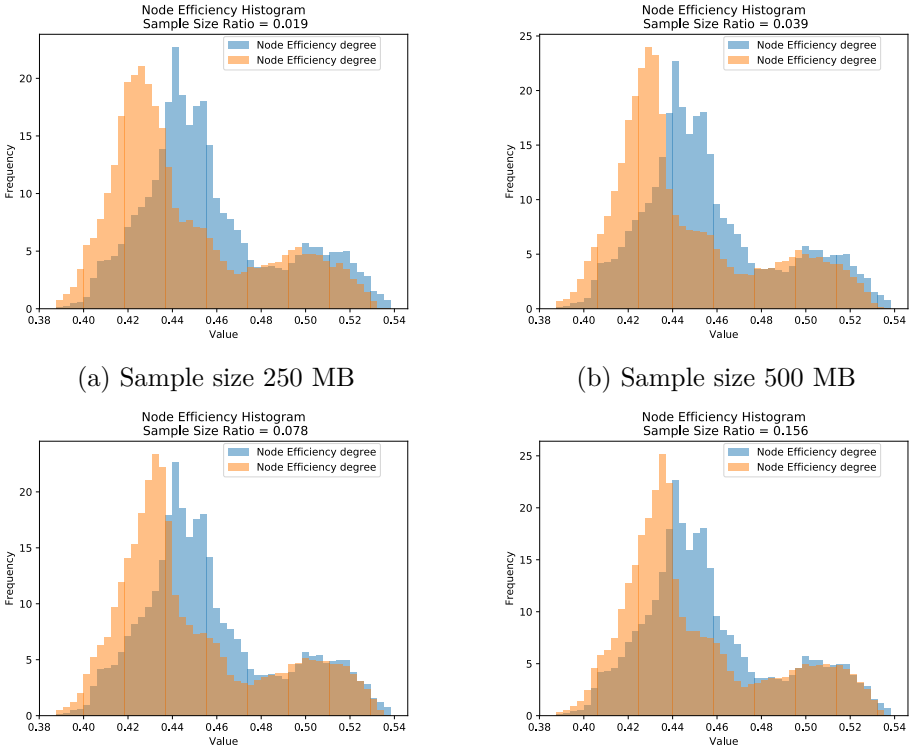


(d) Normalized Variance's Error Rate



(e) Spearman Correlation

Figure 5.6: Degree of initial and sample matrices



(a) Sample size 250 MB

(b) Sample size 500 MB

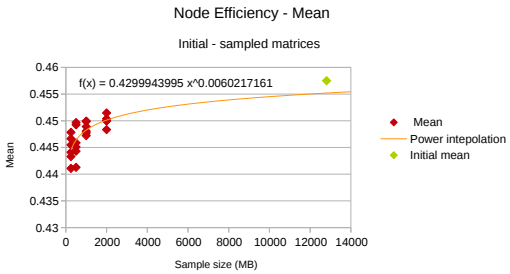
(c) Sample size 1000 MB

(d) Sample size 2000 MB

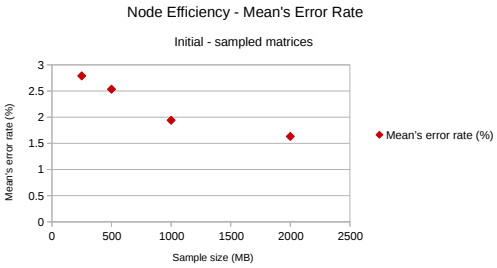
Figure 5.7: Node efficiency distribution comparison

percentage of the error rate compared to the mean and variance of the different measures of the corresponding sampled matrices were computed, in order to have a normalized error and to be able to compare the results.

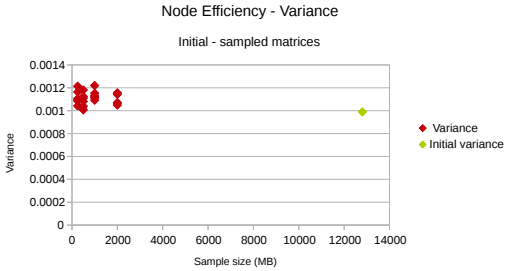
Globally, the size and strata level of the sampled matrices did not have a big impact, but with a 0.995 similarity threshold, the size shows a moderate effect. Globally, the bigger the sampled matrix, the higher the mean's and variance's squared error rate of the non cleansed sampled matrix. The higher the hierarchical level, the lower the mean's and variance's squared error percentage. This second effect can be explained by the fact that the higher the hierarchical level, the lower the number of grouped rows and columns is. Indeed, if the hierarchical level decreases, there are less strata and some rows and columns can be grouped together, while they do not belong to the same stratum in an upper hierarchical level. There are thus more rows and columns grouped together. The more rows and columns are grouped together, the higher is the chance that the measures will differ from the initial matrix, especially if the grouping similarity threshold is too low, hence the hierarchical level can have some effect.



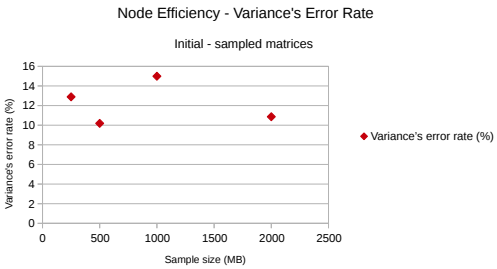
(a) Mean



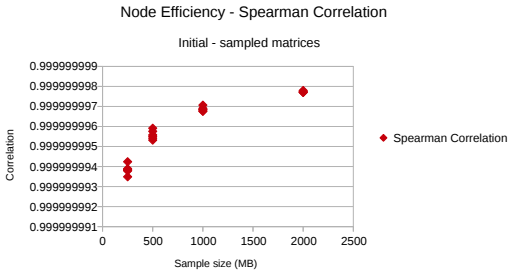
(b) Mean's Error Rate



(c) Variance



(d) Variance's Error Rate



(e) Spearman Correlation

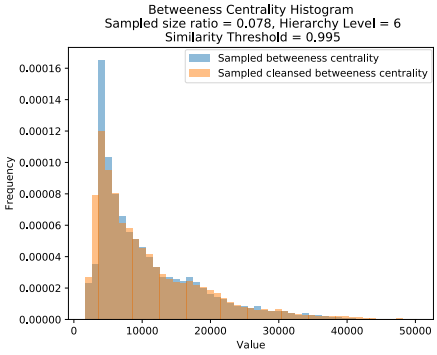
Figure 5.8: Node Efficiency of initial and sample matrices

Betweeness Centrality

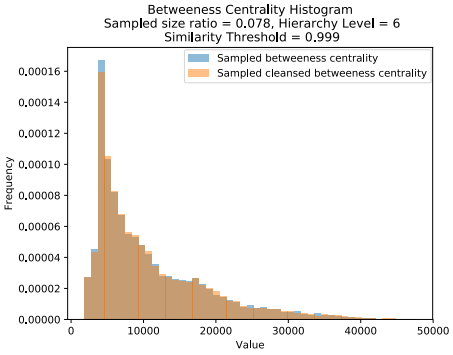
The betweeness centrality was rather similar between the sampled matrices and their cleansed versions. The similarity threshold of 0.995 seems a bit too low. The other cleansed matrices for the 0.999 and 0.9995 similarity threshold have a betweeness centrality Spearman Correlation of almost one (mean values of 0.9999999942 and 0.9999999953 respectively), the Spearman Correlation is not higher than 0.9999999 (see Subfigures 5.9h, and 5.9i). The histogram distribution with sample size 1 GB and a hierarchy level of six depicted for similarity thresholds 0.995, 0.999, and 0.9995 is rather similar to the non cleansed corresponding sampled matrix (see Subfigures 5.9a, 5.9b, and 5.9c respectively). It is rather clear that the distribution with similarity threshold of 0.995, while still keeping a rather similar shape, is not as good as the ones with similarity thresholds of 0.999 and 0.9995. The error rate of the mean is always under 5 % of the expected one for all similarity thresholds, and is even better for the 0.999 and 0.9995 similarity thresholds by not being higher than 3 % (see Subfigures 5.9d and 5.9e). The error rate of the variance is rather high and can reach 50 % for one sample (1 GB, hierarchy level 2) of similarity 0.995. The mean of the variance error rate for the 0.995 similarity threshold is of 24 %. It does not exceed 30 % for the 0.999 and 0.9995 similarity threshold, with a mean of 9 % (See subfigures 5.9f and 5.9g). The sample size does not really seem to have an important impact on the goodness of fit for the betweeness centrality measures.

Clustering Coefficient

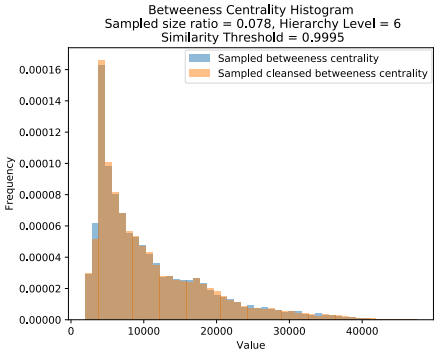
The distribution of the clustering coefficient for the cleansed sampled matrix with similarity threshold 0.995 is not that good, while it is good for the similarity thresholds 0.999 and 0.9995 (see Subfigures 5.10a, 5.10b, and 5.10c). This can be seen in the Spearman Correlation graphs (see Subfigures 5.10h, and 5.10i). While the two other thresholds have a Spearman Correlation of almost one, it can go almost as low as 0.999999 for some cases with a similarity threshold of 0.995. The value of 0.999999 is still rather high, and the global shape is indeed rather similar. The error rate for the mean is very low: under 0.4 %, with a mean error rate for the mean of 0.24 % for the 0.995 similarity threshold, and under 0.15 % with a mean error rate for the mean of respectively 0.034 % and 0.027 % for the 0.999 and 0.9995 similarity thresholds (see Subfigures 5.10d, and 5.10e). The error rate for the variance is higher but does not exceed 14 % for the 0.995 similarity thresholds. The mean error rate for the variance is around 9 %, and does not exceed 6 % for the 0.999 and 0.9995 similarity thresholds, with a mean of 1.9 % and 1.5 % respectively (see Subfigures 5.10f, and 5.10g). The Spearman Correlation for the 0.995 similarity threshold is rather good, always exceeding 0.9999992, but not as good as the Spearman Correlation for the 0.999 and 0.9995 similarity thresholds (mean value of 0.999999991 and 0.999999989 respectively). Once again, the sampling size does not seem to impact the results for the clustering coefficient.



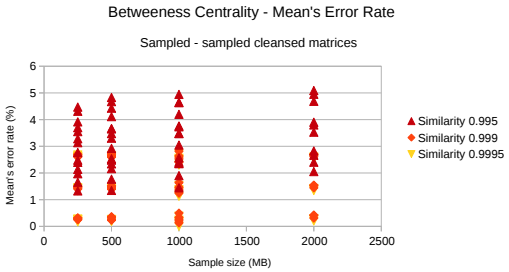
(a) Distribution similarity 0.995



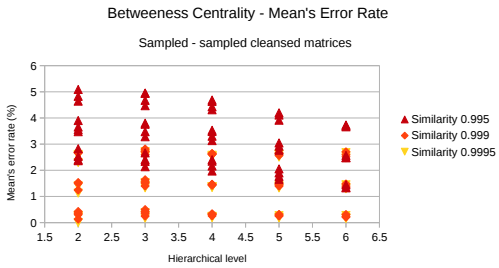
(b) Distribution similarity 0.999



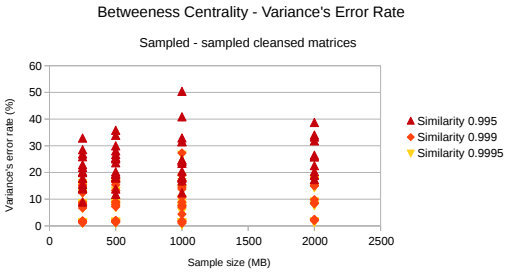
(c) Distribution similarity 0.9995



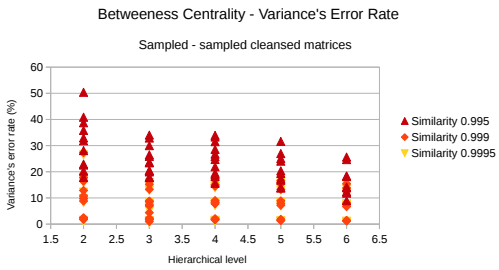
(d) Mean's Error Rate



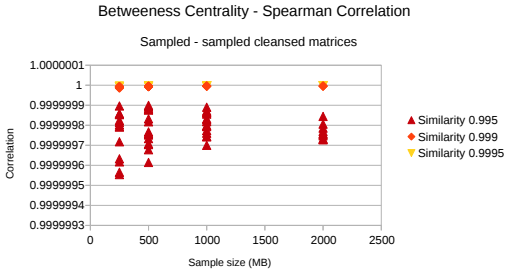
(e) Mean's Error Rate



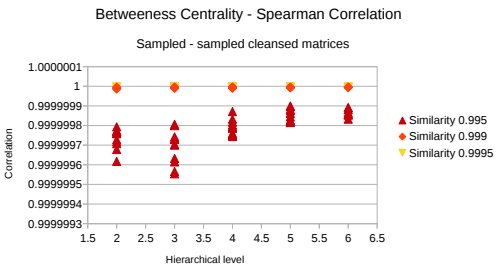
(f) Variance's Error Rate



(g) Variance's Error Rate

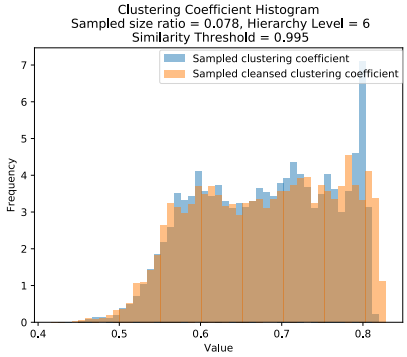


(h) Spearman Correlation

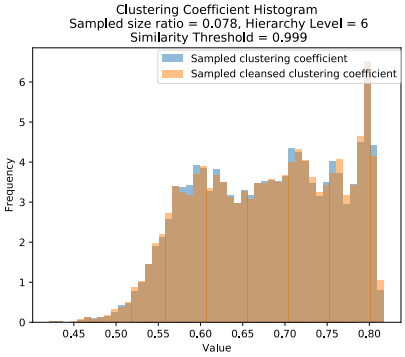


(i) Spearman Correlation

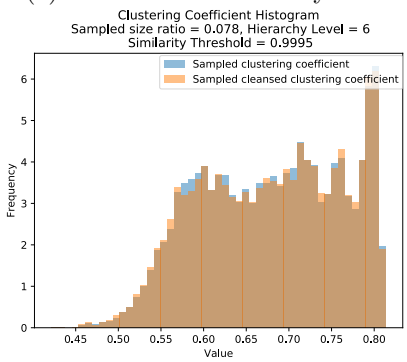
Figure 5.9: Betweeness Centrality of sample and cleansed sample matrices



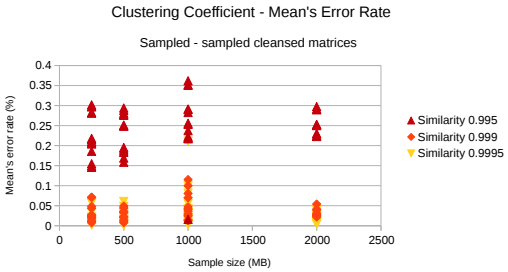
(a) Distribution similarity 0.995



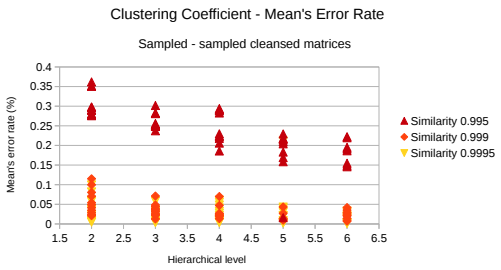
(b) Distribution similarity 0.999



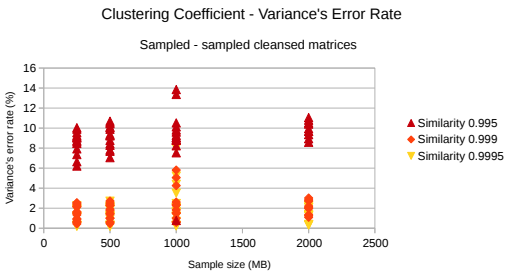
(c) Distribution similarity 0.9995



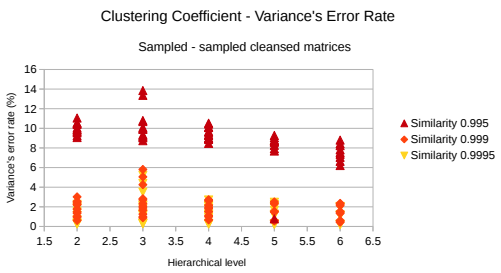
(d) Mean's Error Rate



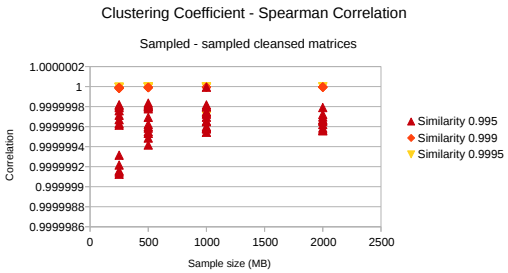
(e) Mean's Error Rate



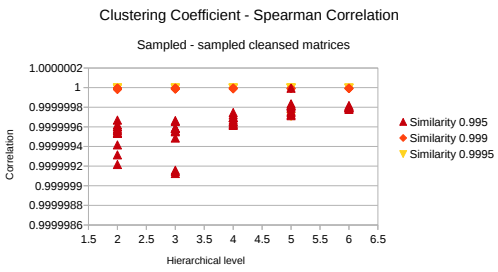
(f) Variance's Error Rate



(g) Variance's Error Rate



(h) Spearman Correlation



(i) Spearman Correlation

Figure 5.10: Clustering Coefficient of sample and cleansed sample matrices

Degree

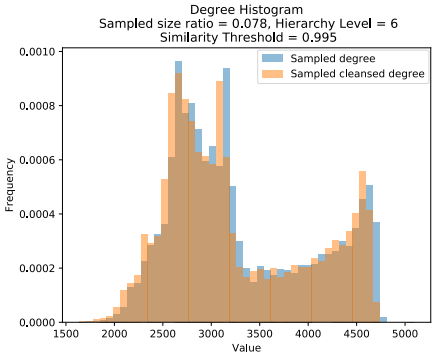
The degree distribution looks rather similar (see Subfigures 5.11a, 5.11b, and 5.11c). It is obviously better for the 0.999 and 0.9995 similarity thresholds than for the 0.995 similarity threshold. The error rate of the mean does not exceed 3.5 %, with a mean error rate of 2 %, 0.84 %, and 0.79 % for the 0.995, 0.999, and 0.9995 similarity thresholds respectively (see Subfigures 5.11d and 5.11e). The error rate of the variance is rather similar for all similarity thresholds, always lying under 2.5 %, its mean value being 0.51 %, 0.7 %, and 0.77 % for the 0.995, 0.999, and 0.9995 similarity thresholds respectively (see Subfigures 5.11f and 5.11g). The Spearman Correlation always exceeds 0.999999, with mean values of 0.9999996344, 0.9999999916, and 0.9999999948 for the 0.995, 0.999, and 0.9995 similarity thresholds respectively. Once again, it is clear that the 0.995 similarity threshold is not as good as the two others (see Subfigures 5.11h and 5.11i). Again, the sampling size has a low impact on the goodness of fit of the cleansed sampled matrices in comparison with their corresponding sampled matrices.

Node Efficiency

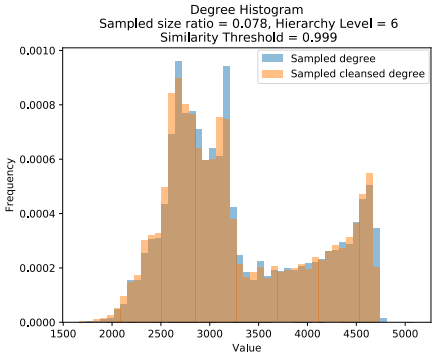
The node efficiency distribution is always very similar to the original one, as can be seen for instance with a sample size of 1 GB and a hierarchical level of six in the Subfigures 5.12a, 5.12b, and 5.12c. Once again, the Spearman Correlation always exceeds 0.999999 and has a mean value of 0.9999996752. It is truly better with the 0.999 and 0.9995 similarity thresholds with mean values of 0.9999999922 and 0.9999999963 respectively (see Subfigures 5.12h, and 5.12i). The error rate of the mean is low, under 0.9 %, with mean values of 0.25, 0.044, and 0.037 for the 0.995, 0.999, and 0.9995 similarity thresholds respectively. The error rate of the variance is rather low, always under 5 %, with mean values of 1.52 %, 0.32, % and 0.29 % for the 0.995, 0.999, and 0.9995 similarity thresholds respectively. Once again, the 0.999 and 0.9995 similarity thresholds clearly preserve better the node efficiency than the 0.995 similarity threshold. The sample size has a low impact on the goodness of fit. It is perhaps slightly better with 250 MB and 500 MB in cases of error rate for the mean and variance with similarity 0.995. It is more dispersed for the 250 MB and 500 MB than for the 1 GB or 2 GB sample sizes, in case of Spearman Correlation for a similarity threshold of 0.995.

Summary

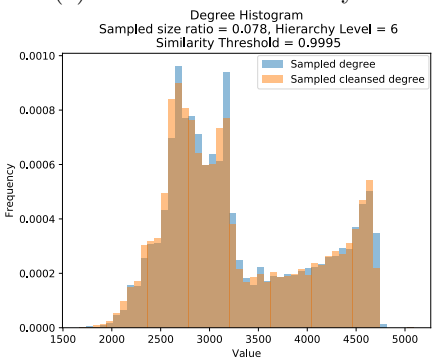
As a result, the 0.999 similarity threshold seems good. Indeed, there is a very small difference to the 0.9995 similarity threshold and the size reduction will be more important. The sample size does not seem to have an important effect on the goodness of fit between sampled and cleansed sampled matrices, which can be of interest when deciding the sample size.



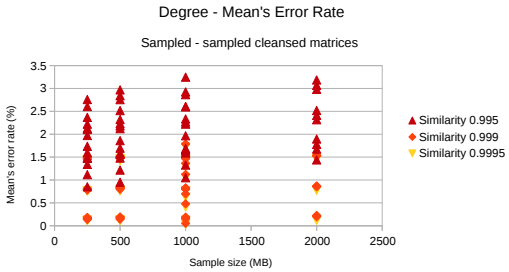
(a) Distribution similarity 0.995



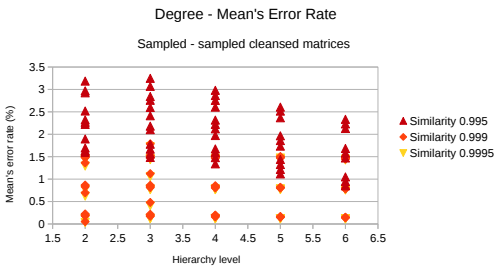
(b) Distribution similarity 0.999



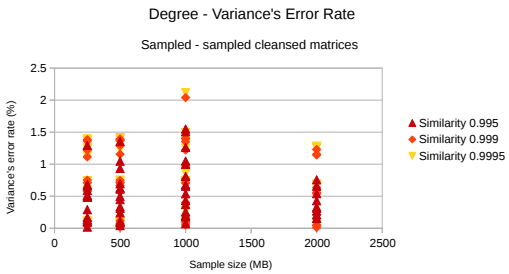
(c) Distribution similarity 0.9995



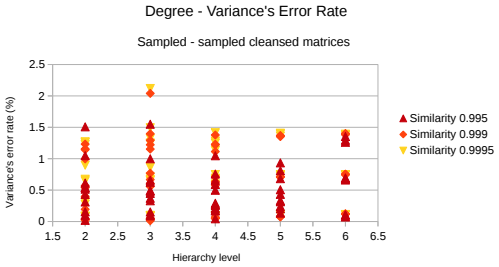
(d) Mean's Error Rate



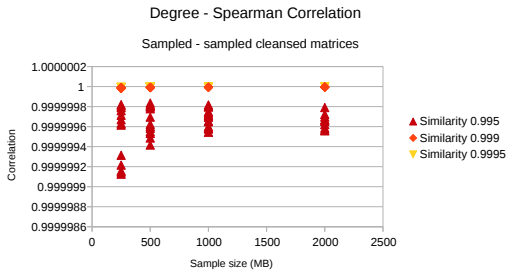
(e) Mean's Error Rate



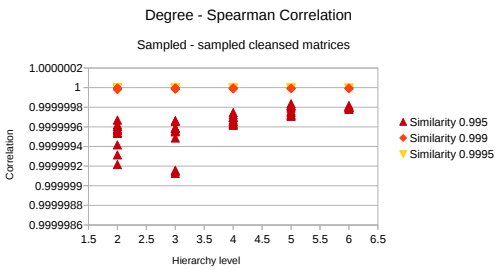
(f) Variance's Error Rate



(g) Variance's Error Rate

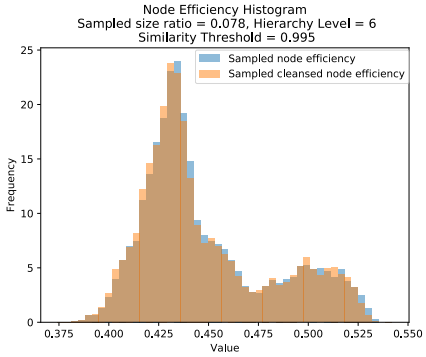


(h) Spearman Correlation

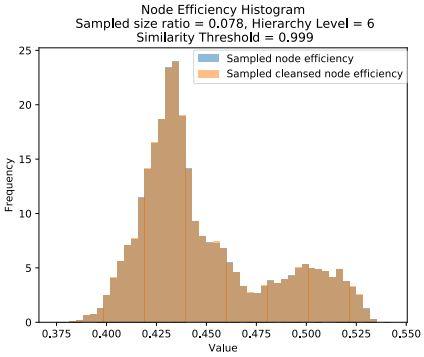


(i) Spearman Correlation

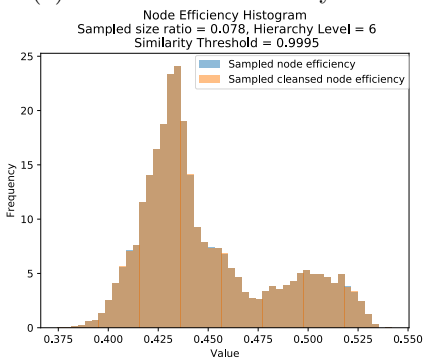
Figure 5.11: Degree of sample and cleansed sample matrices



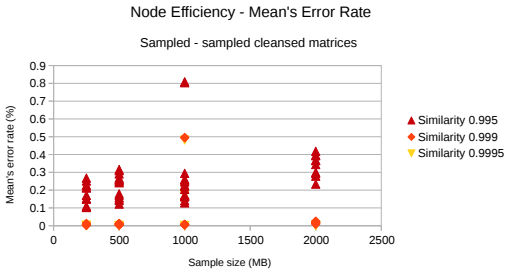
(a) Distribution similarity 0.995



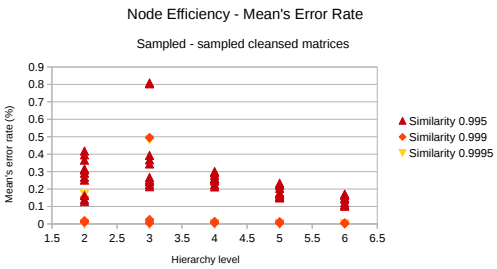
(b) Distribution similarity 0.999



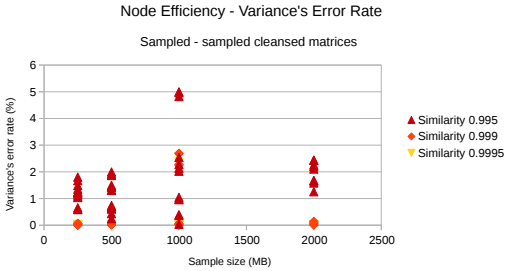
(c) Distribution similarity 0.9995



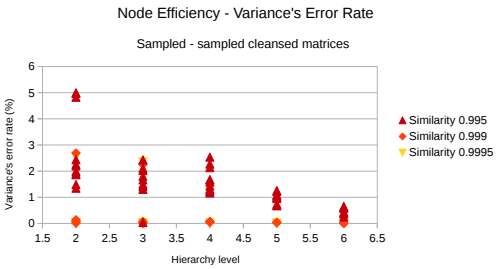
(d) Mean's Error Rate



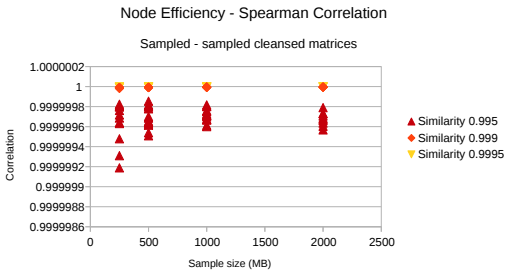
(e) Mean's Error Rate



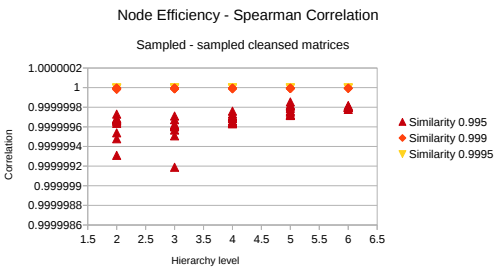
(f) Variance's Error Rate



(g) Variance's Error Rate



(h) Spearman Correlation



(i) Spearman Correlation

Figure 5.12: Node Efficiency of sample and cleansed sample matrices

5.1.4 Initial Matrix - Cleansed Initial Matrix Evaluation

This subsection evaluates how the chosen thresholds (threshold and similarity threshold) on the sampled matrix affected the initial matrix. It should be noted that the measures presented here were not computed on harmonized versions, because of lack of time and computational resources.

The initial matrix was thus cleansed using the thresholds found in the sampling step. The threshold lays between 0.29 and 0.32, and the similarity threshold lays between 0.995 and 0.999. There was no difference between the results of the different thresholdings. It may be due to the fact that these threshold values are very close and that only a few values in the matrices were below the given thresholds. The similarity threshold had an important impact. A similarity threshold of 0.995 reduced the data up to 80 % of its initial size (the final data size was around 2.4 GB). The similarity threshold of 0.997 reduced the data to 72 % of its initial size (the final cleansed matrix being around 3.25 GB big), and the similarity threshold of 0.999 reduced the data to 62 % of its initial size (around 4.6 GB big), see Figure 5.13. This seems to be a very big size reduction, in comparison with the sampled matrix data reduction, which barely exceeded 40 % for the 0.999 similarity threshold. This can be due to the fact that another hierarchy scheme was used, depending on the number of rows and columns per stratum instead of fixing a given anatomical hierarchical level of interest as in the sampling part. The lower the hierarchy level, the more important the size reduction. Rows/columns that do not belong to the same stratum on a given hierarchy level belong to the same on a lower hierarchy level and can then be merged together with the same similarity threshold.

While the 0.995 similarity threshold led to rather poor results (although still preserving some important network characteristics), the 0.997 similarity threshold was slightly better and the 0.999 similarity threshold seems rather good. The different values for the initial matrix were plotted with similarity threshold one for comparison purposes only. The initial matrix was not cleansed with a similarity threshold of one.

Betweeness Centrality

Only the 0.999 similarity threshold seemed good enough to keep the important characteristics of the betweeness centrality distribution (see Subfigures 5.14a, and 5.14b, 5.14c). It is the only distribution which has an error rate for the normalized variance under 20 % (see Subfigure 5.14g) with a variance rather close to the original one (see Subfigure 5.14f). The two other distributions have a rather similar mean compared to the original one: under 1.2 % error rate for all three distributions, with an error rate of around 0.6 % (see Subfigures 5.14d and 5.14e). The Spearman Correlation is always above 0.9999999983 and reaches 0.9999999988 for the 0.999 similarity threshold cleansed matrices (see Subfigure 5.14h).

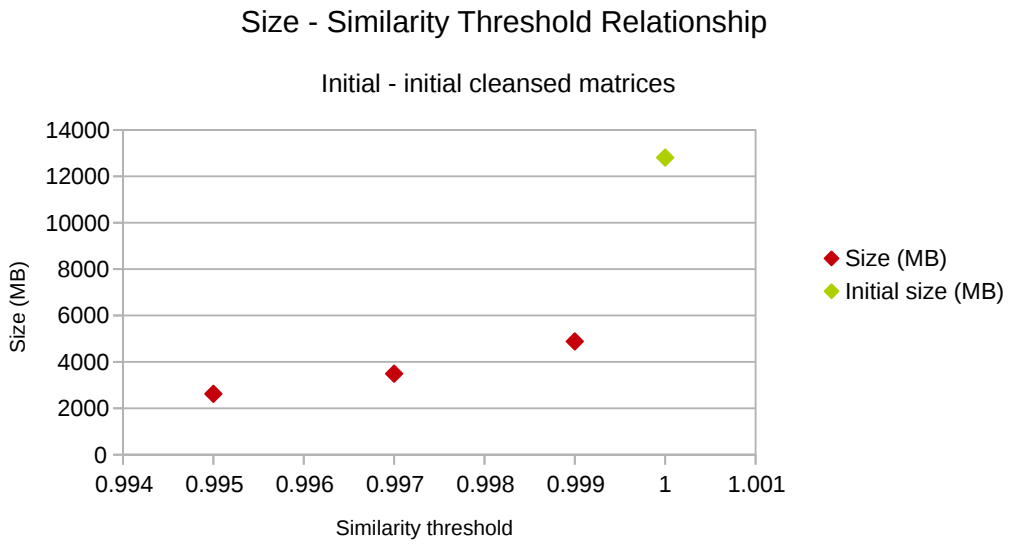


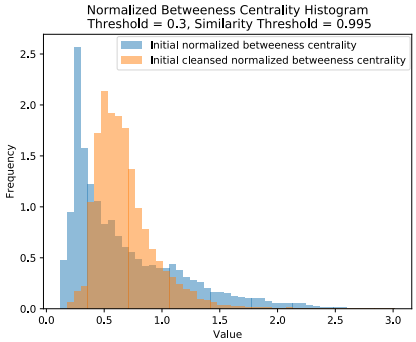
Figure 5.13: Size

Clustering Coefficient

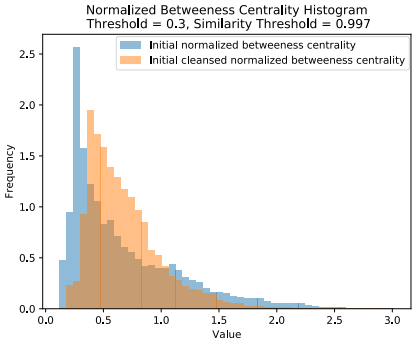
The clustering coefficient distribution is rather bad for all similarity thresholds, even if the 0.999 similarity threshold is rather close to the original one, in a horizontally scaled shape (see Subfigures 5.15a, 5.15b, and 5.15c). This can partly be explained by the fact that the clustering coefficient is somehow normalized by the PAGANI Toolkit and that the cleansed matrices were not harmonized. It thus appears in the plots that these results are not as good as the ones for the betweeness centrality. The mean's error rate can reach almost 6 % for a 0.995 similarity threshold and is close to 1 % in the case of the 0.999 similarity threshold cleansing (see Subfigures 5.15d and 5.15e). The variance's error rate is rather similar to what was found for the betweeness centrality measures, i.e., it has a high variance's error rate, exceeding 15 % for all cleansed matrices (see Subfigures 5.15f and 5.15g). The Spearman Correlation increases with the similarity threshold, from around 0.9999999981 to almost 0.9999999987 (see Subfigure 5.15h).

Degree

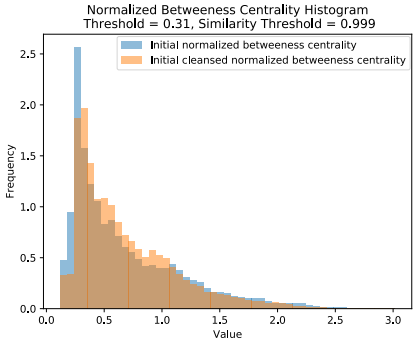
The global shape of the normalized degree distribution is rather well preserved for all thresholds. Interestingly, the distribution for the 0.997 similarity threshold seems to fit better the initial distribution than the 0.999 one (see Subfigures 5.16a, 5.16b and 5.16c). This can also be found in the variance measure of the normalized degree distribution. The variance for the distribution for the 0.997 similarity threshold is very close to the original one, with less than 2 % error rate (see Subfigures 5.16f and 5.16g). The variance's error rate of the distribution for the 0.999 similarity threshold reaches 31 % and is even



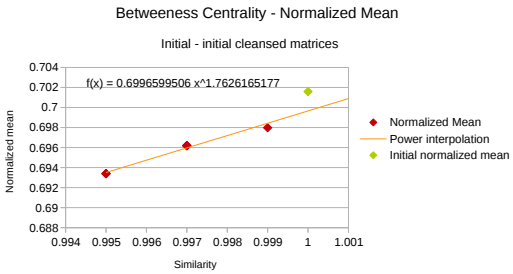
(a) Distribution similarity 0.995



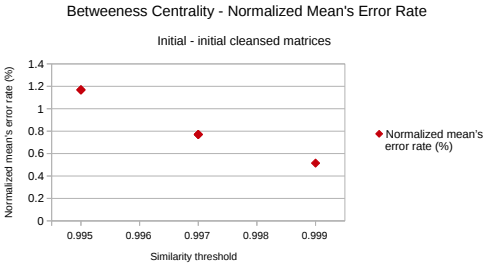
(b) Distribution similarity 0.997



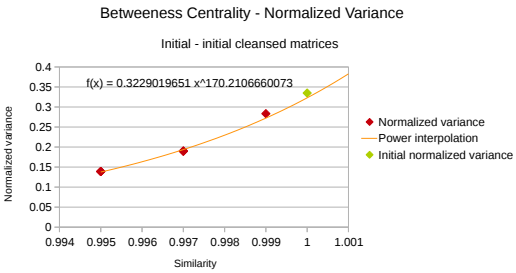
(c) Distribution similarity 0.999



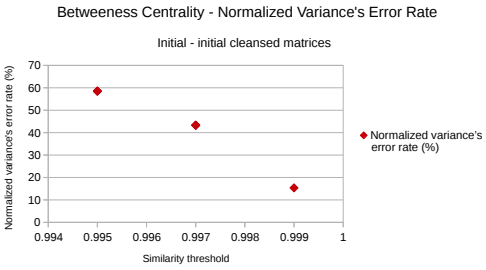
(d) Mean



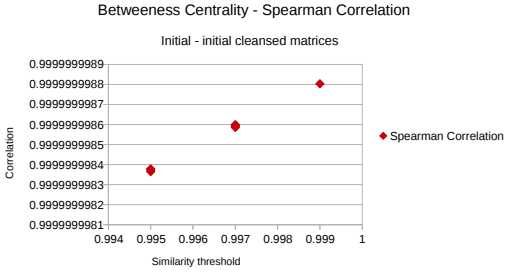
(e) Normalized Mean's Error Rate



(f) Variance

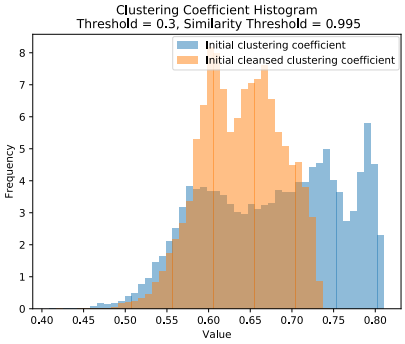


(g) Normalized Variance's Error Rate

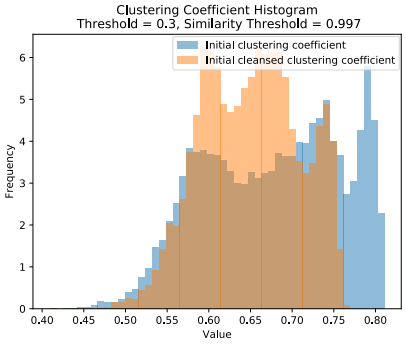


(h) Spearman Correlation

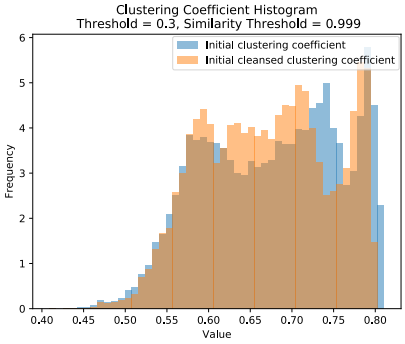
Figure 5.14: Betweeness Centrality of initial and cleansed initial matrices



(a) Distribution similarity 0.995

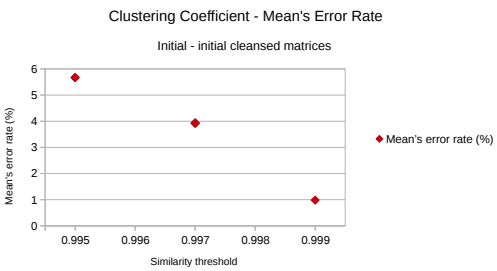


(b) Distribution similarity 0.997

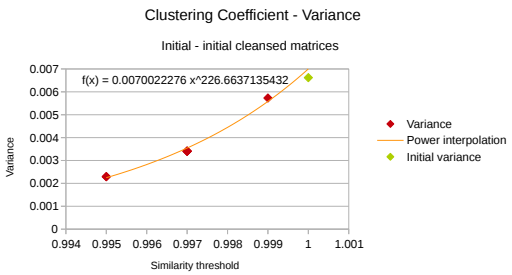


(c) Distribution similarity 0.999

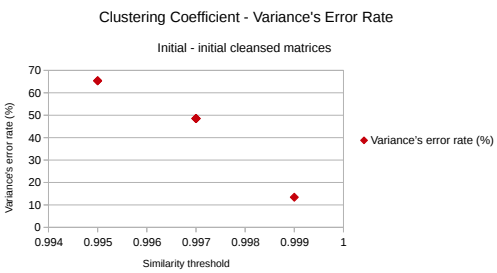
(d) Mean



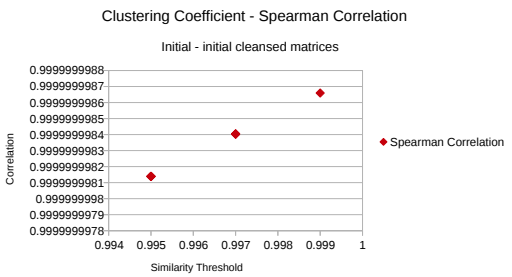
(e) Mean's Error Rate



(f) Variance



(g) Variance's Error Rate



(h) Spearman Correlation

Figure 5.15: Clustering Coefficient of initial and cleansed initial matrices

bigger than the variance's error rate for the 0.995-distribution (around 27 %). However, the mean is better fitted with the 0.999 similarity threshold than the others with an error rate of around 0.7 %. The 0.995 and 0.997 similarity threshold distributions have an error rate of around 2.4 % and 1 % respectively (see Subfigures 5.16d and 5.16e). Once again, the Spearman Correlation increases with the similarity threshold. The 0.999 similarity threshold distribution also has a better Spearman Correlation than the others, i.e., almost 0.9999999987, while the others have a Spearman Correlation of around 0.9999999984 and 0.9999999982 (see Subfigure 5.16h).

Node Efficiency

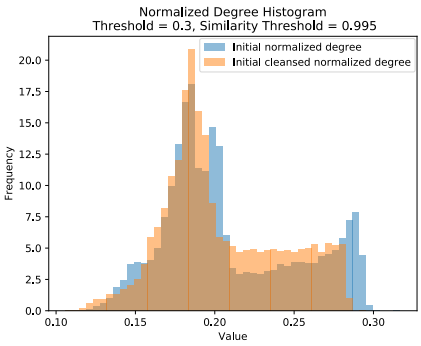
The node efficiency has a distribution rather close to the original one for all cleansed matrices, though a bit right-shifted. The cleansed matrices with a 0.999 similarity threshold have a better fitting distribution than the ones with lower similarity thresholds (see Subfigures 5.17a, 5.17b, and 5.17c). The mean of the node efficiency of the cleased versions is rather good: under 2.5 % for all similarity thresholds and around 0.7 % for a cleansing with the 0.999 similarity threshold (see Subfigures 5.17d and 5.17e). The error rate for the variance is globally high around 30 %, 17 %, and 18 % for the 0.995, 0.997, and 0.999 similarity threshold cleansed matrices respectively (see Subfigures 5.17f and 5.17g).

Summary

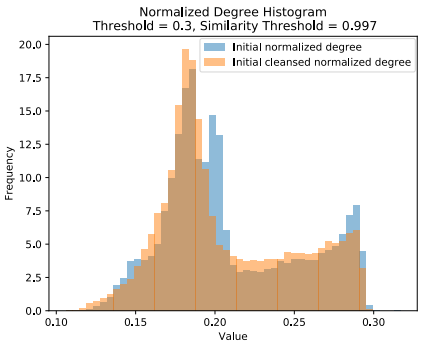
There was no real difference between the results of the different thresholds (0.29 to 0.32), as these thresholds were very close to another. However, the difference was significant for the similarity thresholds. 0.995 is clearly too low for this matrix. 0.997 is slightly better, but 0.999 is really the most appropriate similarity threshold among the three. All of these thresholds had a rather important impact on the final matrix size. This impact is really more important than the one on the sampled cleansed matrices. This can be explained by the different hierarchical scheme, which can be confusing for the user but was important since otherwise the computation of the cleansing (especially the similarity grouping) could be infeasible depending on the stratum sizes. The results presented here were not on harmonized matrices, where they would most certainly have been better, but seem already promising. A 0.999 similarity threshold seems rather appropriate for the cleansing. Another higher similarity threshold would give better results, of course, but then the gain in memory would not be as good. It could be interesting in the future to test with higher similarity thresholds and to define clear error margins that the results should not cross in order to still be called "similar to the initial matrix" by the neuroscientists.

5.2 Discussion and Open Issues

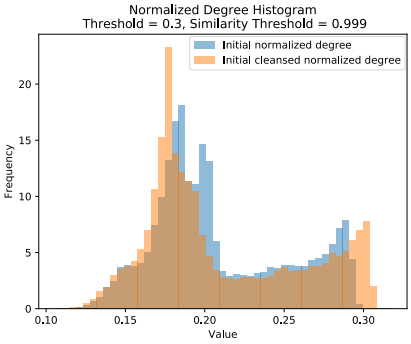
A sampling and visualisation tool was presented here, in order to define good cleansing thresholds and operations to cleanse large connectivity matrices. The sampling scheme relies on two parameters: a sampling size (or ratio of the initial size) and the anatomical



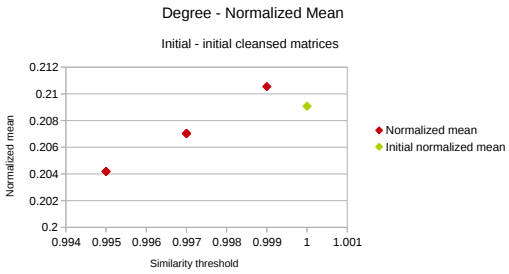
(a) Distribution similarity 0.995



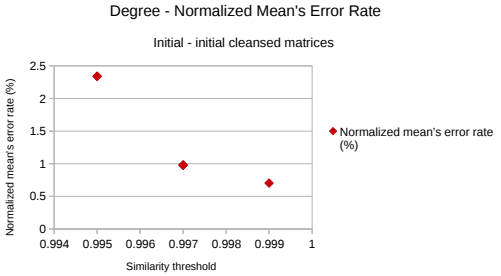
(b) Distribution similarity 0.997



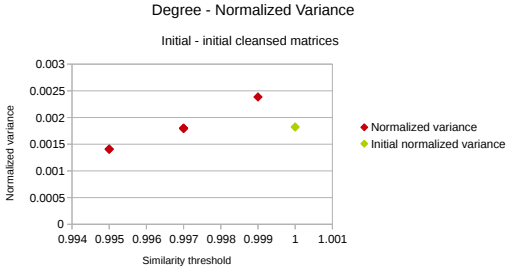
(c) Distribution similarity 0.999



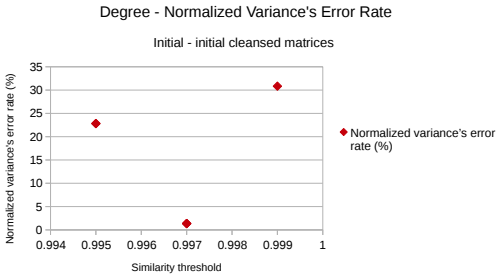
(d) Mean



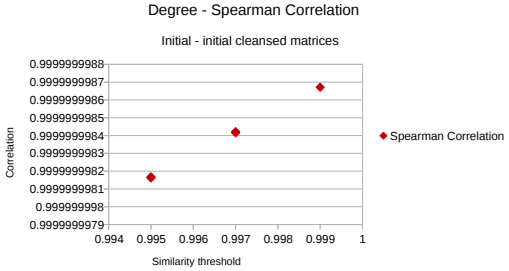
(e) Normalized Mean's Error Rate



(f) Variance

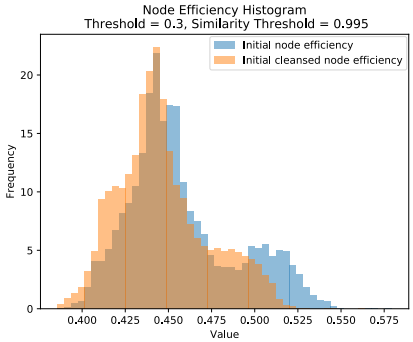


(g) Normalized Variance's Error Rate

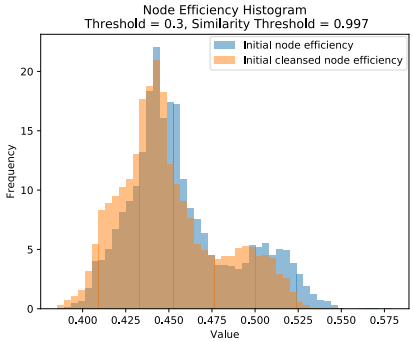


(h) Spearman Correlation

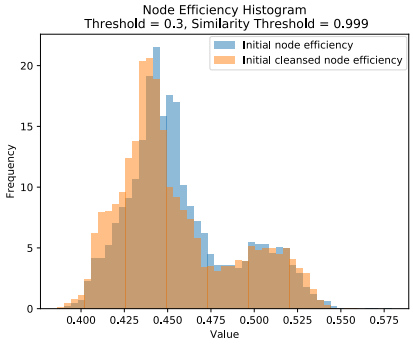
Figure 5.16: Degree of initial and cleansed initial matrices



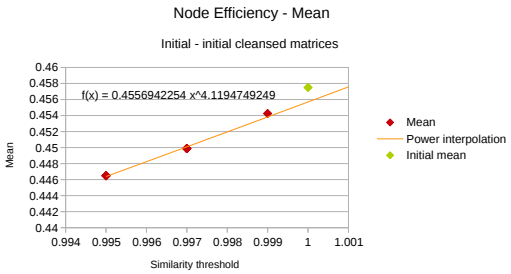
(a) Distribution similarity 0.995



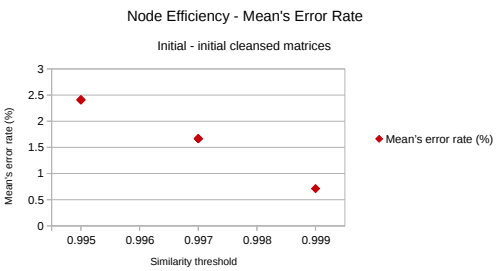
(b) Distribution similarity 0.997



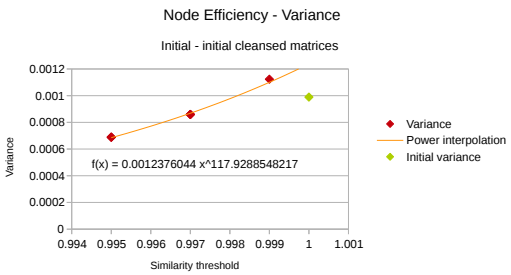
(c) Distribution similarity 0.999



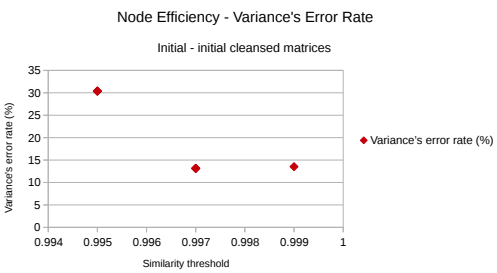
(d) Mean



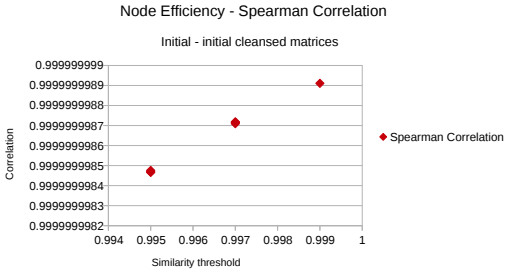
(e) Mean's Error Rate



(f) Variance



(g) Variance's Error Rate



(h) Spearman Correlation

Figure 5.17: Node Efficiency of initial and cleansed initial matrices

hierarchical level of interest. If the matrices are not squared, such as in case of the structural connectivity matrices which have a lot more columns than rows, the selection of rows and columns to keep is based on a hypergeometrical model. This model relies on three hypotheses. First, that the row space is a subspace of the column space (all row ids are also in the column ids, but not all column ids may be in the row ids). This first hypothesis was validated by our data, and would require a transposition if other custom data is used with this tool. Second, the model is built on the assumption that the row and column ids can be found in a volume of interest file that maps the ids to an anatomical region id and to the position of the corresponding neuron or groups of neurons in space. In order to be able to use this tool, this volume of interest mapping file is required. Last but not least, it relied on the hypothesis that the row ids are equi-distributed within the different anatomical hierarchy levels. This hypothesis is actually not always verified in practise. Although it is rather well verified for the first hierarchy levels (1 or 2), it becomes less and less accurate with the hierarchy level. For instance, for the structural connectivity matrix, there are a lot of rows belonging to the cortical region and its subregions, while the rows are not that numerous for other brain regions. This can lead to a reserved space in the reservoir while no row will ever be stored. In any case, a postprocessing is needed to order all rows and columns according to a complete brain hierarchy, but if the hypothesis is not verified, it also means that the sample is smaller than initially planed.

The visualisation tool was tested by a potential user, a computer scientist working in the field of neuro-informatics. He found the tool rather effective and easy to use. However, some operations can be long to compute, especially if the sampling size is large and the hierarchy level is low. It is recommended not to have a sampling size above 2 to 4 GB with a low hierarchical level such as one or two. It is recommended to use high hierarchical levels such as six. The cleansing step of the initial matrix can also be confusing, since the hierarchy scheme is not the same as in the sampled matrices and can lead to results that are a bit different from the ones that were acquired in the sampling cleansing. It could also have been interesting to store a sampled matrix on the hard-drive at each cleansing step. Thus, the user would be able to come back to the cleansing step results instead of coming back to the initial sampled matrix.

The evaluation was only performed on one initial gene connectivity matrix, because of lack of time and computational resources. Hence, the results may not be extrapolated to all neurobiological connectivity matrices. First of all, the sampled matrices were evaluated with regards to the initial matrix, to be able to define good sampling sizes and hierarchy levels. In order to get a good preservation of the characteristics of the initial gene connectivity matrix, it is recommended to have a sampling size of at least one GB (7.8 % of the initial size), or higher, with at least a hierarchical level of six. Otherwise the following operations would be too costly and long to perform. It should not exceed four GB since it would then cross the 12 GB RAM limit. However, it seems important to note that for all sample sizes that were tested, the same cleansing thresholds (threshold and similarity threshold) were found and the sampling size did not really impact the similarity

between sampled and cleansed sampled matrices. As a result, even if a sampled matrix with a lower sampling ratio does not fit the original matrix that well, the found cleansing operation thresholds still seem good. A similarity threshold of 0.999 seemed a good compromise between characteristics preservation and future matrix size, while 0.3 seemed good for the thresholding step. Finally, the cleansed initial matrix was evaluated with different similarity thresholds between 0.995 and 0.999. 0.999 is clearly the best choice here, but an even higher threshold would most probably have been better. However, the size reduction would not have been that big. These final results were not computed on harmonized matrices, where the results would have been better most probably.

5.3 Relation to the Literature

Usually, a first cleansing step occurs when getting the initial signal (Diffusion-tensor imaging (DTI), functional Magnetic Resonance Imaging, Diffusion-weighted Magnetic Resonance Imaging, proton density MRI, blood flow imaging...). The signal is noisy and needs to be thresholded. Then, the connectivity matrices need to be built. These matrices can still be noisy with redundant data. They can be up to several gigabytes big, and do not fit into the RAM usually. A reduction of size would lead to a higher processing speed, for instance by querying [GKHB18]. Ganglberger et al. [GKHB18] created a specific data structure in order to speed up aggregation queries in large brain networks, so that the user can get the results of the queries in real time. For this purpose the authors applied a row wise compression, in order to exploit potential sparseness of the data, after having reordered the rows and columns by a space filling curve (Hilbert curve) in order to preserve locality. The authors relied on the principle that local neural nodes tend to have similar characteristics. This row wise procedure compresses the zero values and relies on spatial relationships of rows and columns. The approach is not lossy and keeps all data, but the gain in space is limited by the number of zeros in the graph, and as this data structure is rather specific, the result requires adapted tools in order to be analysed.

An important point in this work is that the data is redundant and noisy and, when getting a new connectivity matrix, it is not trivial to define thresholds below which the values are considered as noise. This is where the work presented here has a role to play. Neuroscientists can use this cleansing tool in order to get an idea of which filters to apply to reduce the size of the matrices, by using and visualizing a sample of the matrices. This work can be inserted in the preprocessing workflow of building brain networks.

This cleansing tool provides a novel adjacency matrix representation, with aggregation based on similarity, such as in ZAME [EFD+08] or MultiLayerMatrix [DCF16]. It presents several similarities with the MultiLayerMatrix approach, among which the similarity window to explore the similarity between the elements of a row or column cluster. The approach of combining anatomical and similarity-based hierarchies is novel though, according to the author's best knowledge. This technique could be of use in other contexts when visualizing large biological datasets.

CHAPTER 6

Summary and Future Work

Understanding the principle organization of the brain and its function is a continuing quest in neuroscience and psychiatry. Understanding how the brain works, how it is functionally and structurally correlated as well as how the genes are expressed in the brains is one of the most important aims in neuroscience. Recent advances in neuro-imaging as well as brain initiatives such as the Human Brain Project [hum] or the Allen Institute [OHN+14] allowed the neuro-scientists to create important brain data resources with ever increasing precision. These resources are made available for the neuroscientists and researchers and can be used to create network graphs representing the different connectivities in the brain. These networks, represented as connectivity matrices, can be up to hundreds of gigabytes, and are too large to hold in current machines' memory. This is especially the case if one wants to fuse different connectivity matrices to better understand the underlying processes, i.e., on anatomical, structural and functional scales.

These networks, contain noisy and redundant data. A cleansing of the networks is required in order to be able to fit them in current machines memory. Operations can then be performed on them in order to infer new knowledge in neurosciences. The goal of this thesis was to realise a guided cleansing of a connectivity matrix in order to reduce the size of the initial matrix while keeping the most important information. A visual guided cleansing tool was thus implemented in order to reduce the connectivity data size. Hence, the result of this work aims at being a component in the large connectivity matrices preprocessing pipeline.

First, in order to be able to find good threshold and similarity threshold values, the matrices needed to be sampled to relevant and representative data. This was performed based on random sampling in the different anatomical hierarchies, with a fixed anatomical level and a sampling size chosen by the user.

Then, the visual tool can give the user more insight into his data, as he or she can see have a preview of how the matrix looks like thanks to the heatmap, and of how

similar the columns and rows are. The user can also see where the neurons or groups of neurons represented by the rows and columns are anatomically located. Thanks to all this information, the user can define appropriate threshold and similarity threshold values. To this end, an aggregated version of the sampled matrix is displayed. The matrix has row and column headers displaying the anatomical hierarchy down to the chosen anatomical level of interest. Under this level, the rows and columns are aggregated according to their similarity. The rows and columns can be expanded by the user by clicking on them. The user can also visualize the aggregated matrix based purely on the anatomical hierarchy. In this case, the similarity within the different anatomical hierarchies is not displayed, this may be a step in future work. In order to have an idea of the impact of the chosen thresholds, the number of affected rows and columns are displayed for each cleansing step. Measures histograms for initial and sampled versions are also displayed. A spatial slice view is also displayed with the anatomical strata contours of the chosen hierarchy level and the position of the sampled voxels in order to know where the sampled nodes are spatially located.

In future work, it could be interesting to add another step before the finalisation. After using the tool to get an idea of threshold and similarity threshold ranges, this step would test the different range combinations in order to get the most appropriate pair. The results would then be stored on hard-drive and the tool would provide the user with a visual comparison of the results from the different combinations. This step would be rather long and could also include the calculation of network measures such as betweeness centrality, clustering coefficient or node efficiency instead of only degree, as it does not require any user interaction during the computation.

The evaluation could not be exhaustively performed due to a lack of time and computational resources. In future work, the evaluation could also be realised on the structural connectivity matrix, and on cleansed harmonized versions of the initial gene connectivity matrix. It was found that a threshold of 0.3 and a similarity threshold of 0.999 are rather good to cleanse the gene connectivity matrix. However, precisely quantifying what "good" means in terms of error rate for end users (i.e., neuroscientists and researchers) would also be important, with clearly defined limits that the error rates should not cross. It should also be noted that, although small sampling sizes (i.e., 2 % or 4 % of the initial size) were not as representative as bigger ones, they were still good enough to get an idea of appropriate cleansing thresholds. It is also recommended to use a high hierarchical level value such as six, to ease the computation and lower the memory costs of the similarity computation. This seems also reasonable, since rows and columns belonging to very different anatomical regions should not be merged together.

Due to lack of time, the imputation step to impute missing values was not implemented. However, the available matrices did not really contain missing values. This could also be implemented in future work.

Acknowledgements

This work was performed at the Competence Centre VRVis in context of the VRVis strategic research project Integrative Visual Computing and the FFG Research Headquarter (852936) at IMP Vienna funded by Boehringer Ingelheim and the Austrian Research Promotion Agency (FFG). VRVis is funded by BMVIT, BMDW, Styria, SFG and Vienna Business Agency in the scope of COMET - Competence Centers for Excellent Technologies (854174) which is managed by FFG.

APPENDIX A

Supplementary Figures

A.1 Mockups - Final Iteration

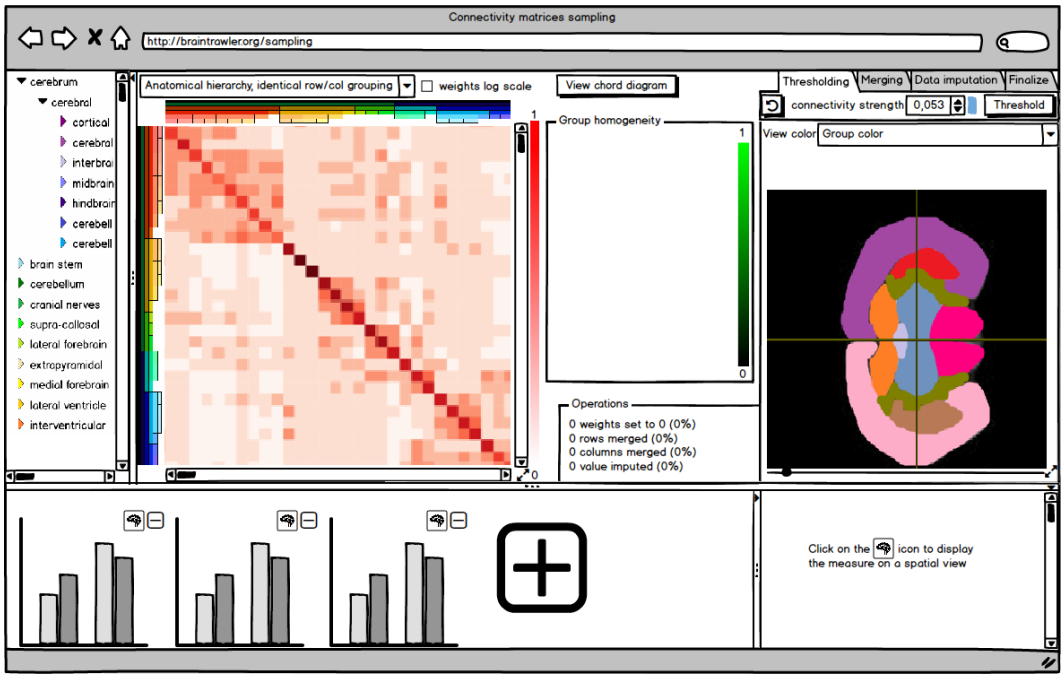


Figure A.1: Before thresholding

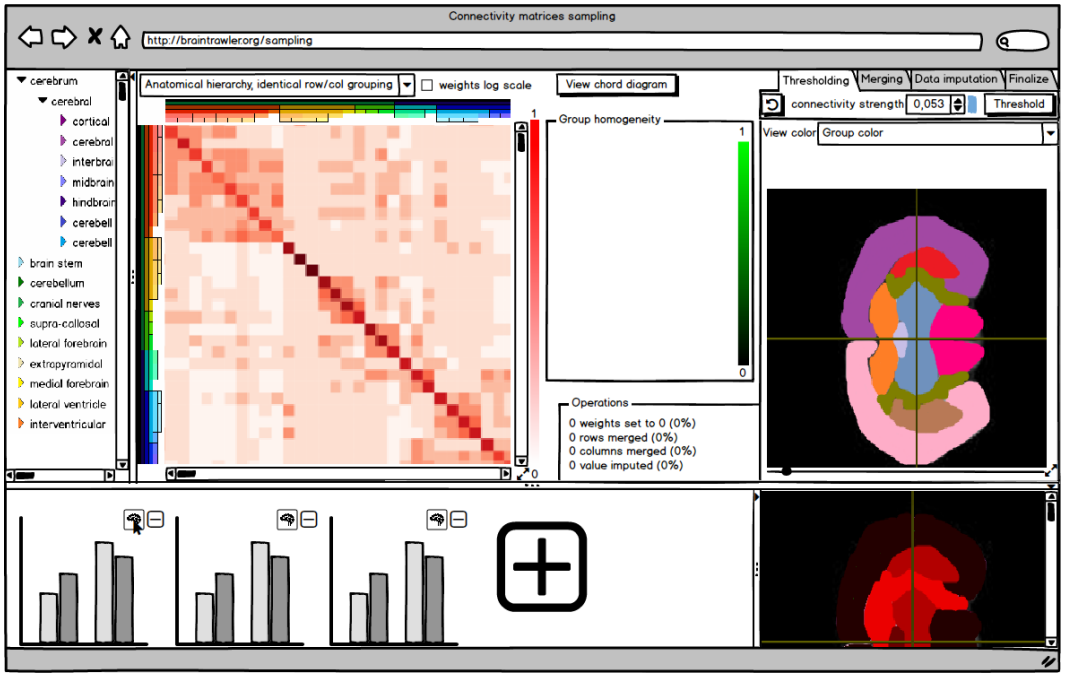


Figure A.2: Before thresholding, displaying the network measures on the 2D projection space

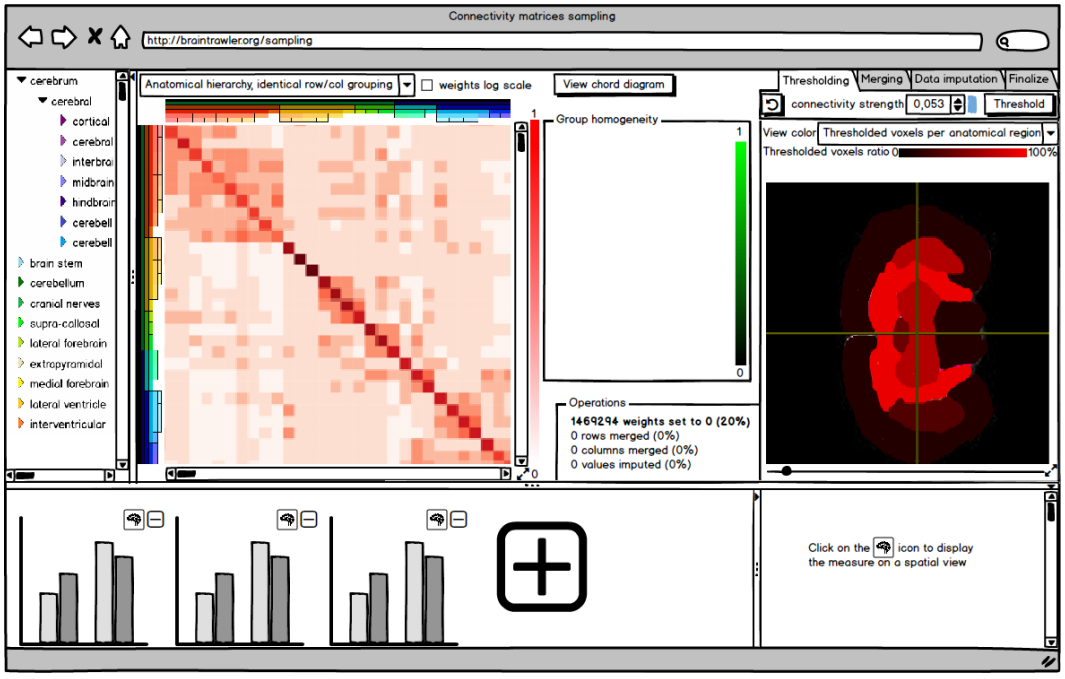


Figure A.3: After thresholding, colouring the 2D projection space according to the effect of thresholding on the different brain regions, operation panel is updated

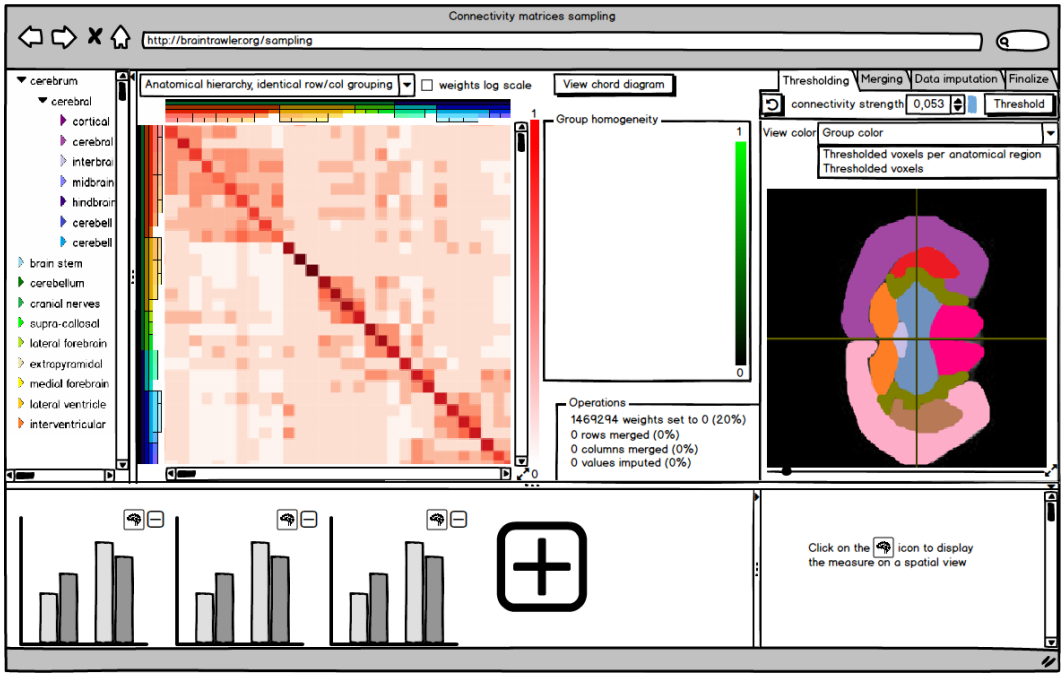


Figure A.4: After thresholding, colouring the 2D projection space according to the group colour, operation panel is updated

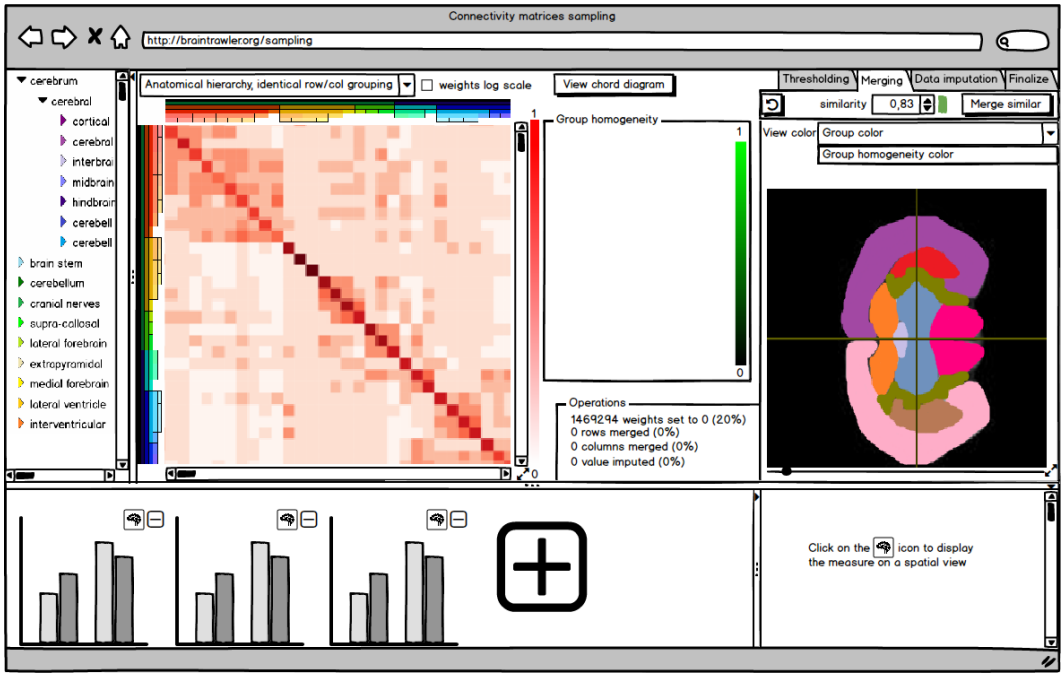


Figure A.5: Before merging the rows and columns

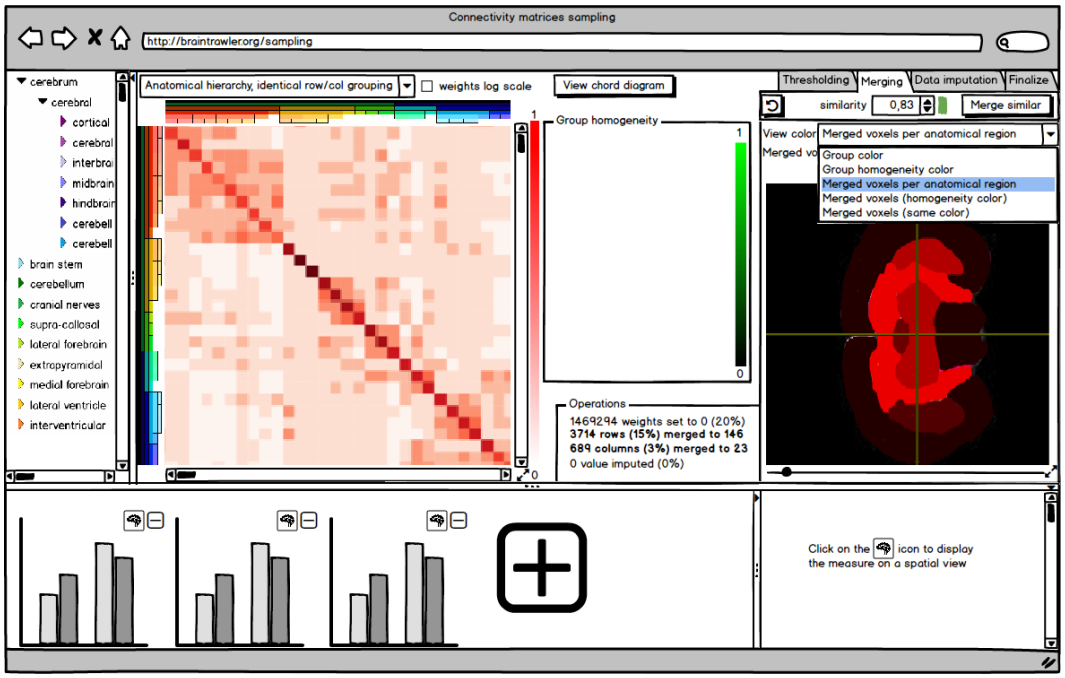


Figure A.6: After merging the rows and columns, choice of the different 2D projection view colour scheme, operation panel is updated

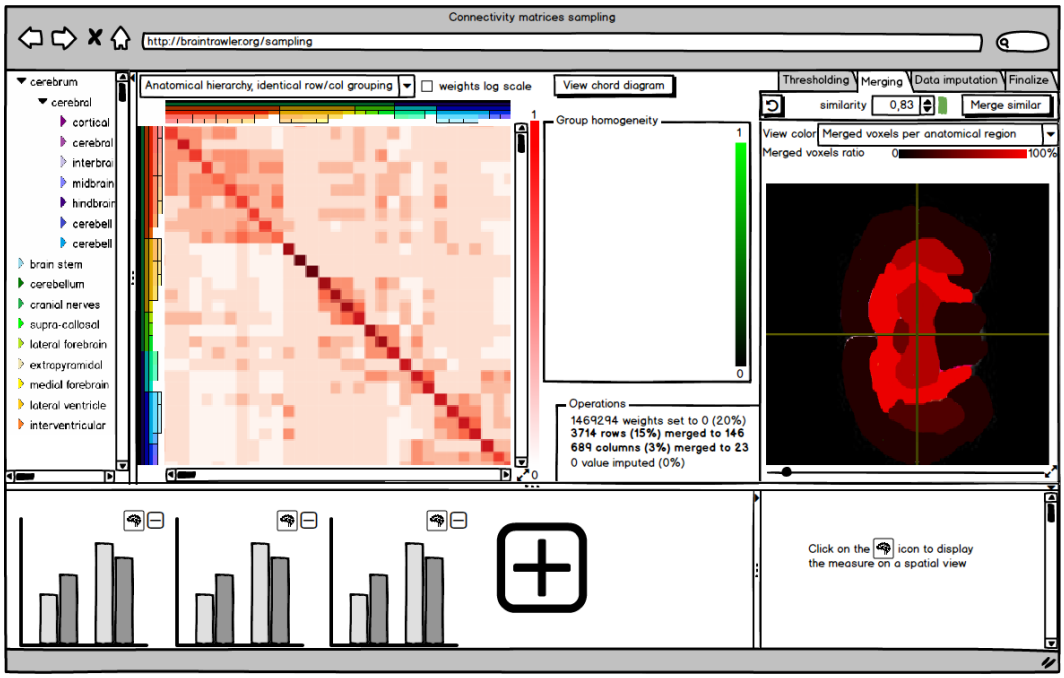


Figure A.7: After merging the rows and columns, colouring the 2D projection space according to the effect of merging on the different brain regions, operation panel is updated

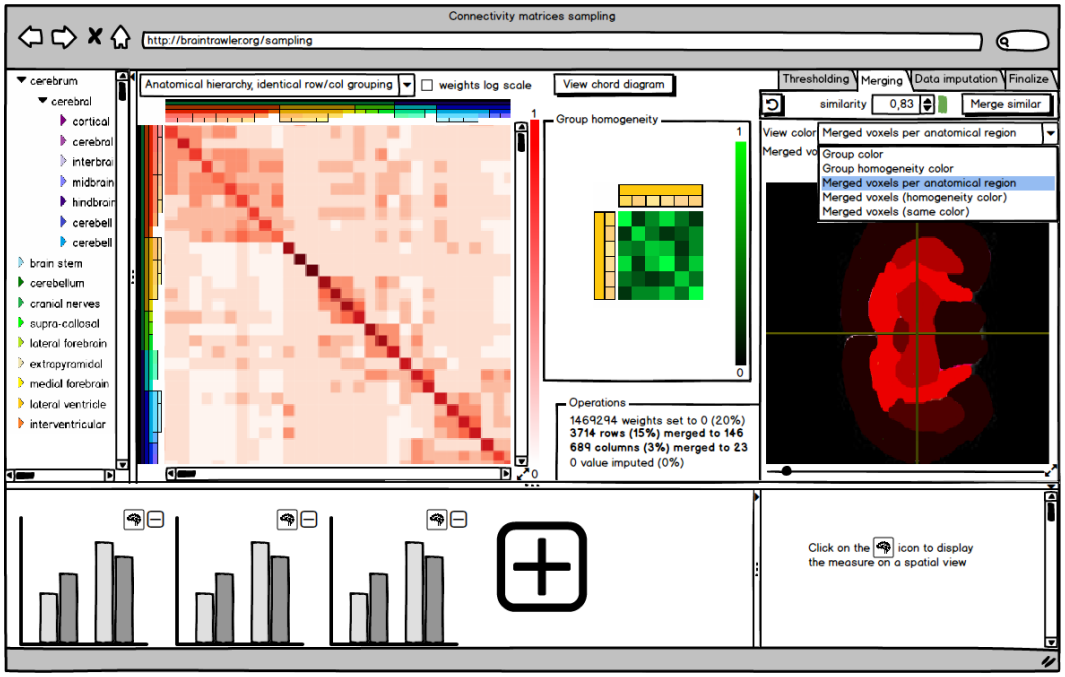


Figure A.8: After merging the rows and columns, clicking on a cluster to display its homogeneity (i.e., how the clusters of lower level are similar to each other), operation panel is updated

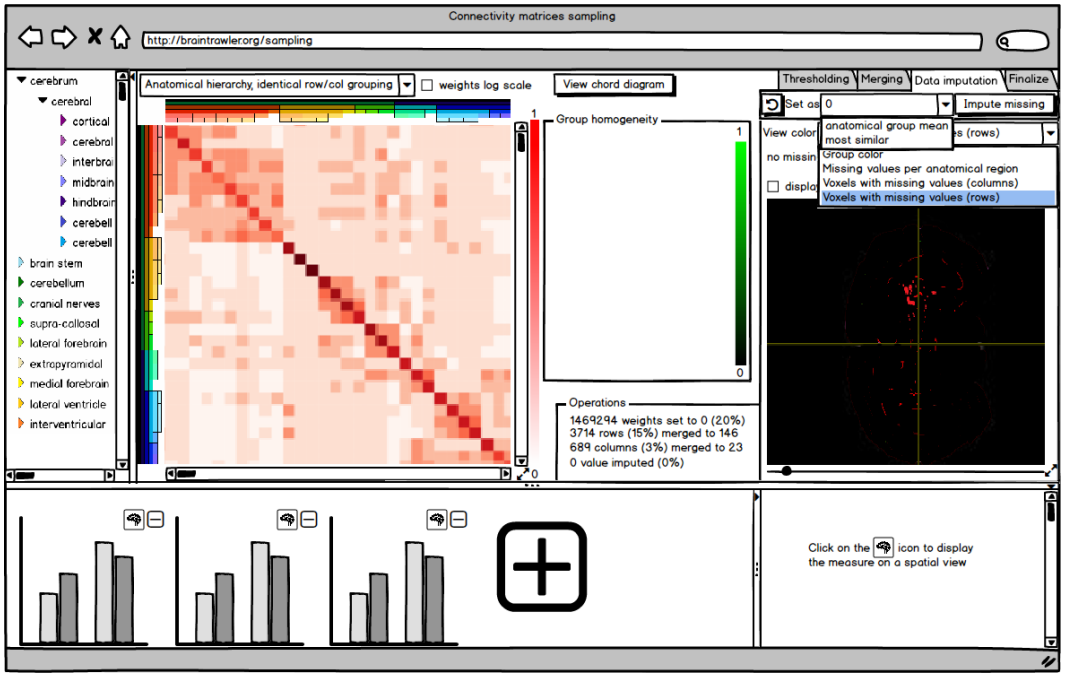


Figure A.9: Before data imputation

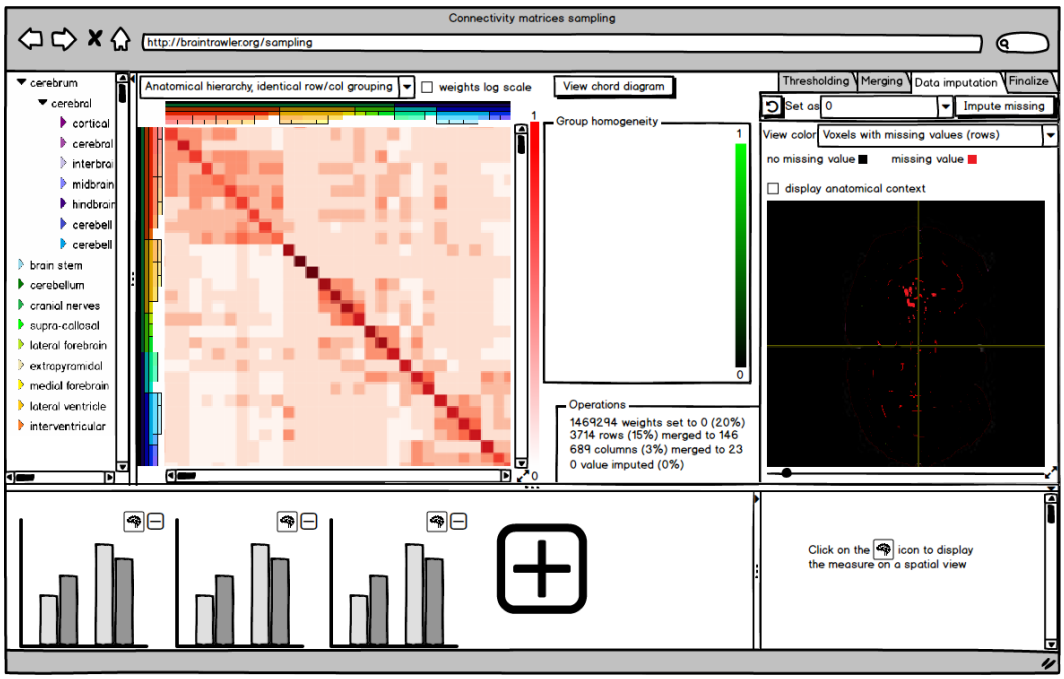


Figure A.10: Before data imputation, displaying missing values from the rows (red)

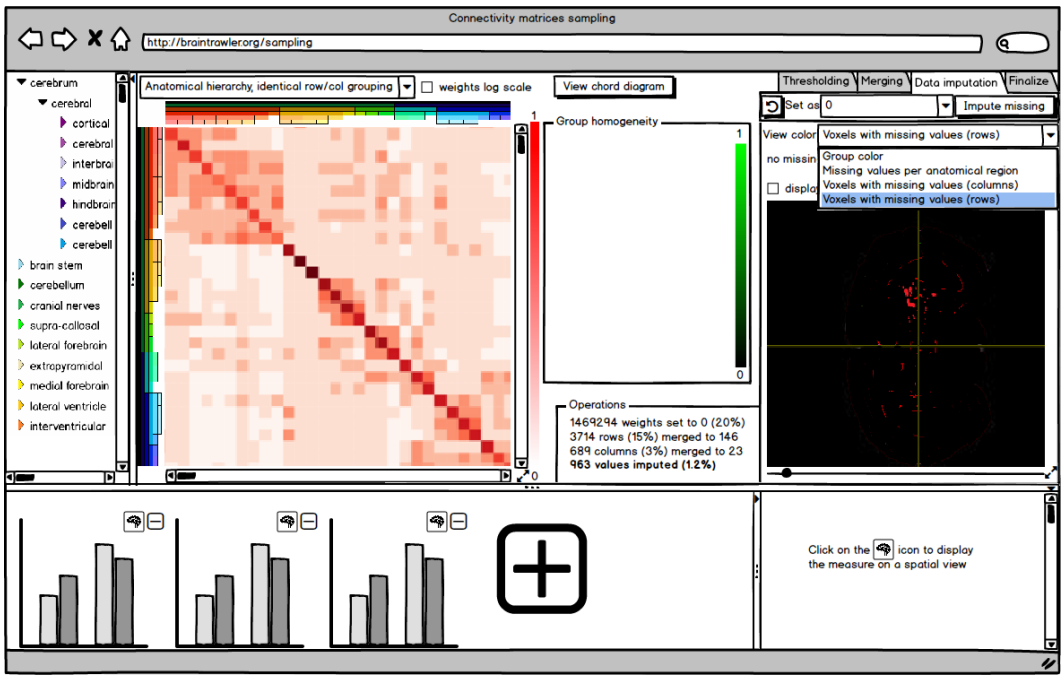


Figure A.11: After data imputation, operation panel is updated

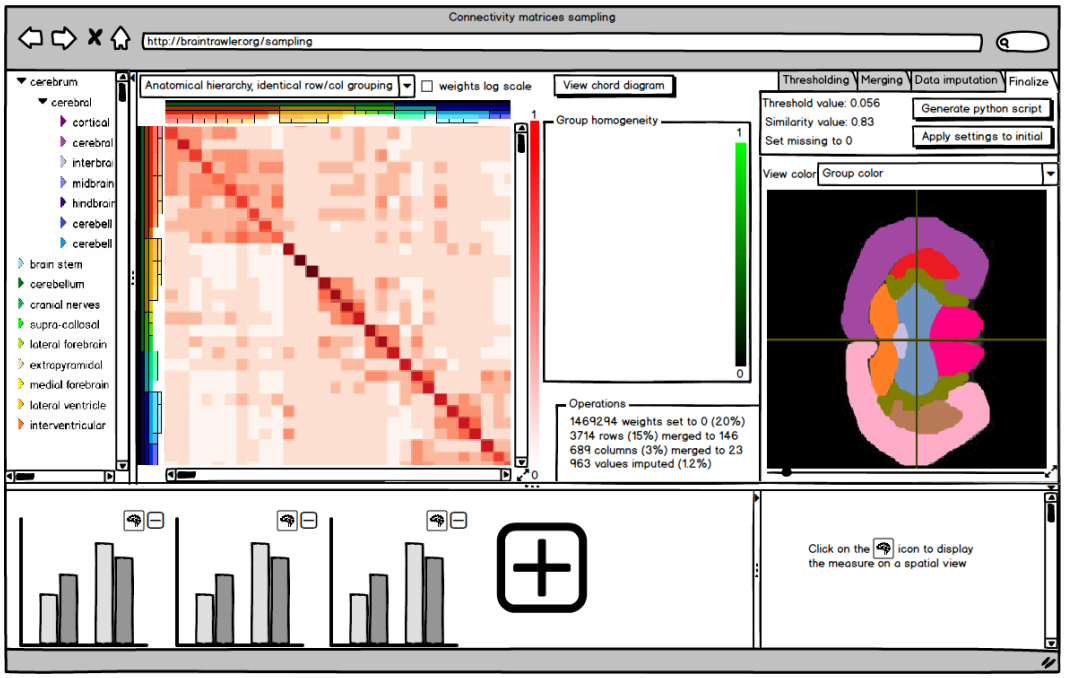


Figure A.12: Finalisation, the user can apply the cleansing operations directly to the initial matrix and/or get the cleansing Python script

List of Figures

Glossary

Blood Oxygen Level Dependent Blood oxygenation level dependent (BOLD) imaging is the standard technique used to generate images in functional MRI (fMRI) studies, and relies on regional differences in cerebral blood flow to delineate regional activity. 9

connectome A connectome is a comprehensive map of neural connections in the brain, and may be thought of as its "wiring diagram". 7–9

Count Min The Count-Min sketch is a probabilistic data structure based on hash functions to map events to frequencies, using sub-linear space. 38

degree The number of edges (or number of neighbours) for a given node. 16

Diffusion-tensor imaging Diffusion-tensor imaging (DTI) is a MRI technique that uses anisotropic diffusion to estimate the axonal (white matter) organisation of the brain. 9, 115

Diffusion-weighted Magnetic Resonance Imaging Diffusion weighted Magnetic Resonance Imaging is a form of MR imaging based upon measuring the random Brownian motion of water molecules in a voxel of tissue. Basically, it estimates the likelihood in white matter to have fiber bundles that go from one site to other parts of the brain. 9, 115

Factor Analysis Factor Analysis is a statistical method that aims at identifying a low-dimensional space that preserves the variance shared across the variables, thus describing the variability among the variables. 29, 30

Force-based layouts Based on a physical model of attraction and repulsion, a force-based layout aims at layouting the nodes of the graph in an optimal way. There are several force-based layout algorithms, most of which are rather time consuming. 44

Fraction of Labeled Neurons (FLNe) The number of labeled neurons in one region divided by the total number of labeled neurons less the number of labeled neurons intrinsic to this area.. 29

Frobenius Norm The Frobenius Norm, in linear algebra, is a matrix norm defined as the square root of the absolute squares of its elements. 42

functional Magnetic Resonance Imaging Functional magnetic resonance imaging or functional MRI (fMRI) measures brain activity by detecting changes associated with blood flow. 9, 115

Gaussian Mixture Model A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. 29

hub Nodes with high degrees. 17

Meta-Analytic Connectivity Modeling Meta-Analytic Connectivity Modeling measures the coupling between brain regions during specific tasks. 13

Multi-Dimensional Scaling Multi-Dimensional Scaling aims at finding a set of low-dimensional coordinates that best preserves pairwise distances in the original high-dimensional space so that to provide a visual representation of similarities within a dataset: for any pairwise dissimilarity, it reconstruct a map that preserves distances. 29, 30

Principal Component Analysis Principal Component Analysis is a statistical method to emphasize variation and detect strong patterns along so-called principal directions. It thus outputs an ordered set of orthogonal directions that captures the greatest variance in the data. 29, 30

Resting State Resting state networks are brain networks acquired through fMRI, in order to evaluate regional interactions when the subject is not performing any explicit task. 13

Scale Free Graph A scale-free network is a network whose degree distribution follows a power law, at least asymptotically, i.e., the majority of nodes only have a few connections to other nodes, whereas some nodes are connected to many others. 16

Singular Value Decomposition Singular Value Decomposition consists in the factorization of any (not necessarily squared) matrix M into the form M=U S V*, with S being a diagonal matrix whose values are the singular values. 29

Acronyms

dMRI Diffusion-weighted MRI. 12

FF Forest Fire. 39

LSH Locality-Sensitive Hashing. 36

RE Random Edge. 39

RN Random Node. 38, 39

RNE Random Node Edge. 39

RNN Random Node Neighbour. 39

RW Random Jump. 39

RW Random Walk. 39

Bibliography

[ABHR+13] B. Alper, B. Bach, N. Henry Riche, T. Isenberg, and J. D. Fekete. Weighted graph comparison techniques for brain connectivity analysis. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 483–492. ACM, 2013.

[ABKS99] M. Ankerst, M. M. Breunig, H. P. Kriegel, and J. Sander. Optics: ordering points to identify the clustering structure. In *ACM Sigmod record*, volume 28, pages 49–60. ACM, 1999.

[ADNK14] N. K. Ahmed, N. Duffield, J. Neville, and R. Kompella. Graph sample and hold: A framework for big-graph analytics. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1446–1455. ACM, 2014.

[AGGR98] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. *Automatic subspace clustering of high dimensional data for data mining applications*, volume 27. ACM, 1998.

[all] Allen brain atlas. `http://help.brain-map.org/display/mouseconnectivity/API`. [Online; accessed 2018-12-23].

[All15] M. Allanic. *Gestion et visualisation de données hétérogènes multidimensionnelles: application PLM à la neuroimagerie*. PhD thesis, Université de Technologie de Compiègne, 2015.

[ANK14] N. K. Ahmed, J. Neville, and R. Kompella. Network sampling: From static to streaming graphs. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(2):1–56, 2014.

[APHW03] C. C. Aggarwal, S. Y. Philip, J. Han, and J. Wang. A framework for clustering evolving data streams. In *Proceedings 2003 VLDB Conference*, pages 81–92. Elsevier, 2003.

[Aub04] D. Auber. Tulip—a huge graph visualization framework. In *Graph drawing software*, pages 105–126. Springer, 2004.

[BAKG+16] R. F. Betzel, A. Avena-Koenigsberger, J. Goñi, Y. He, M. A. De Reus, A. Griffa, P? E? Vértes, B. Mišic, J. P. Thiran, and P. Hagmann. Generative models of the human connectome. *NeuroImage*, 124:1054–1064, 2016.

[Bar12] C. I. Bargmann. Beyond the connectome: how neuromodulators shape neural circuits. *Bioessays*, 34(6):458–465, 2012.

[BB17] R. F. Betzel and D. S. Bassett. Multi-scale brain networks. *NeuroImage*, 160(August):73–83, 2017.

[BBC+11] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D.S. Seljebotn, and K. Smith. Cython: The best of both worlds. *Computing in Science Engineering*, 13(2):31 –39, 2011.

[BC00] D. Barbará and P. Chen. Using the fractal dimension to cluster datasets. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 260–264. ACM, 2000.

[bct] Brain connectivity toolbox. `https://sites.google.com/site/bctnet/`. [Online; accessed 2018-12-27].

[BHJ09] M. Bastian, S. Heymann, and M. Jacomy. Gephi: an open source software for exploring and manipulating networks. *Icwsm*, 8(2009):361–362, 2009.

[BJG+13] T. Blumensath, S. Jbabdi, M. F. Glasser, D. C. Van Essen, K. Ugurbil, T. E.J. Behrens, and S.n M. Smith. Spatially constrained hierarchical parcellation of the brain with resting-state fmri. *NeuroImage*, 76:313–324, 2013.

[BJGJ01] Z. Bar-Joseph, D. K. Gifford, and T. S. Jaakkola. Fast optimal leaf ordering for hierarchical clustering. *Bioinformatics*, 17(suppl_1):S22–S29, 2001.

[Bla] B. Blaus.

[BM98] V. Batagelj and A. Mrvar. Pajek-program for large network analysis. *Connections*, 21(2):47–57, 1998.

[BMKK14] A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. Krause. Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 671–680, New York, NY, USA, 2014. ACM.

[BMP+17] R. F. Betzel, J. D. Medaglia, L. Papadopoulos, G. L. Baum, R. Gur, R. Gur, D. Roalf, T. D. Satterthwaite, and D. S. Bassett. The modular organization of human anatomical brain networks: Accounting for the cost of wiring. *Network Neuroscience*, 1(1):42–68, 2017.

[bra]

[BRNL+10] P. Bellec, P. Rosa-Neto, O. C. Lyttelton, H. Benali, and A. C. Evans. Multi-level bootstrap analysis of stable clusters in resting-state fmri. *NeuroImage*, 51(3):1126–1139, 2010.

[BS09] E. Bullmore and O. Sporns. Complex brain networks: Graph theoretical analysis of structural and functional systems. 10:186–98, 03 2009.

[BS12] T. E.J. Behrens and O. Sporns. Human connectomics. *Current Opinion in Neurobiology*, 22(1):144 – 153, 2012. Neurotechnology.

[BSST13] J. Batson, D. Spielman, N. Srivastava, and S. H. Teng. Spectral sparsification of graphs: Theory and algorithms. 56:87–94, 08 2013.

[CD14] G. Cormode and N. Duffield. Sampling for big data: A tutorial. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1975–1975, New York, NY, USA, 2014. ACM.

[CEQZ06] F. Cao, M. Estert, W. Qian, and A. Zhou. Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM international conference on data mining*, pages 328–339. SIAM, 2006.

[CGS06] G. Cormode, M. Garofalakis, and D. Sacharidis. Fast approximate wavelet tracking on streams. In *International Conference on Extending Database Technology*, pages 4–22. Springer, 2006.

[CMM17] M. B. Cohen, C. Musco, and C. Musco. Input sparsity time low-rank approximation via ridge leverage score sampling. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, pages 1758–1777, Philadelphia, PA, USA, 2017. Society for Industrial and Applied Mathematics.

[CMP16] M. B. Cohen, C. Musco, and J. Pachocki. Online row sampling. In *The 19th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2016*, 2016.

[CY14] J. P. Cunningham and B. M. Yu. Dimensionality reduction for large-scale neural recordings. *Nature Neurosciences*, 17(11):1500–1509, 2014.

[DCF16] T. N. Dang, H. Cui, and A.G. Forbes. Multilayermatrix: Visualizing large taxonomic datasets. In *Proceedings of the 7th EuroVis Workshop on Visual Analytics (EuroVA)*, pages 55–59, 2016.

[DG09] J. S. Damoiseaux and M. D. Greicius. Greater than the sum of its parts: a review of studies combining structural connectivity and resting-state functional connectivity. *Brain Structure and Function*, 213(6):525–533, 2009.

[DMF15] T. N. Dang, P. Murray, and A. G. Forbes. Pathwaymatrix: visualizing binary relationships between proteins in biological pathways. In *BMC proceedings*, volume 9, page S3. BioMed Central, 2015.

[DXZ+18] H. Du, M. Xia, K. Zhao, X. Liao, H. Yang, Y. Wang, and Y. He. Pagani toolkit: Parallel graph-theoretical analysis package for brain network big data. *Human brain mapping*, 39(5):1869–1885, 2018.

[EFD+08] N. Elmqvist, J. D. Fekete, T. N. Do, H. Goodell, and N. Henry. ZAME: Interactive Large-Scale Graph Visualization. In *IEEE VGTC Pacific Visualization Symposium 2008*, 2008.

[EKSX96] M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996.

[ELLS11] B. S. Everitt, S. Landau, M. Leese, and D. Stahl. Hierarchical clustering. *Cluster analysis*, pages 71–110, 2011.

[Fek15] J. D. Fekete. Reorder.js: A javascript library to reorder tables and networks. `https://hal.inria.fr/hal-01214274`, Oct 2015. Poster.

[fla] Flask, a python microframework. `http://flask.pocoo.org/`. [Online; accessed 23-01-2019].

[For65] E. Forgey. Cluster analysis of multivariate data: Efficiency vs. interpretability of classification. *Biometrics*, 21(3):768–769, 1965.

[Fre78] L. C. Freeman. Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239, 1978.

[GCR15] H. Gualdron, R. L.F. Cordeiro, and J. F. Rodrigues. Structmatrix: large-scale visualization of graphs by means of structure detection and dense matrices. In *Data Mining Workshop (ICDMW), 2015 IEEE International Conference on*, pages 493–500. IEEE, 2015.

[GH05] S. Guha and B. Harb. Wavelet synopsis for data streams: minimizing non-euclidean error. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 88–97. ACM, 2005.

[GKHB18] F. Ganglberger, J. Kaczanowska, W. Haubensak, and K. Buehler. A data structure for real-time aggregation queries of big brain networks. *bioRxiv*, page 346338, 2018.

[GKP+17] F. Ganglberger, J. Kaczanowska, J. M. Penninger, A. Hess, and W. Haubensak. Predicting functional neuroanatomical maps from fusing brain networks with genetic information. *NeuroImage*, 2017.

[GN02] M. Girvan and M. E.J. Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

[GRS98] S. Guha, R. Rastogi, and K. Shim. Cure: an efficient clustering algorithm for large databases. In *ACM Sigmod Record*, volume 27, pages 73–84. ACM, 1998.

[GRS00] S. Guha, R. Rastogi, and K. Shim. Rock: A robust clustering algorithm for categorical attributes. *Information systems*, 25(5):345–366, 2000.

[GSM+15] S. Gu, T. D. Satterthwaite, J. D. Medaglia, M. Yang, R. E. Gur, R. C. Gur, and D. S. Bassett. Emergence of system roles in normative neurodevelopment. *Proceedings of the National Academy of Sciences*, 112(44):13681–13686, 2015.

[GV16] H. Gibson and P. Vickers. Using adjacency matrices to lay out larger small-world networks. *Applied Soft Computing*, 42:80 – 92, 2016.

[GZFA10] A. Goldenberg, A. X. Zheng, S. E. Fienberg, and E. M. Airoldi. A survey of statistical network models. *Foundations and Trends in Machine Learning*, 2(2):129–233, 2010.

[Haa16] P. J. Haas. Data-stream sampling: basic techniques and results. In *Data Stream Management*, pages 13–44. Springer, 2016.

[HBB+13] A. M. Hermundstad, D. S. Bassett, K. S. Brown, E. M. Aminoff, D. Clewett, S. Freeman, A. Frithsen, A. Johnson, C. M. Tipper, and M. B. Miller. Structural foundations of resting-state and task-based functional connectivity in the human brain. *Proceedings of the National Academy of Sciences*, 110(15):6169–6174, 2013.

[HCG+08] P. Hagmann, L. Cammoun, X. Gigandet, R. Meuli, C. J. Honey, V. J. Wedeen, and O. Sporns. Mapping the structural core of human cerebral cortex. *PLOS Biology*, 6(7):1–15, 07 2008.

[HF07] N. Henry and J. D. Fekete. Matlink: Enhanced matrix visualization for analyzing social networks. In *IFIP Conference on Human-Computer Interaction*, pages 288–302. Springer, 2007.

[HFM07] N. Henry, J. D. Fekete, and M. J. McGuffin. Nodetrix: a hybrid visualization of social networks. *IEEE Transactions on visualization and computer graphics*, 13(6):1302–1309, 2007.

[HG08] M. D. Humphries and K. Gurney. Network 'small-world-ness': a quantitative method for determining canonical network equivalence. *PloS one*, 3(4):e0002051, 2008.

[HK98] A. Hinneburg and D. A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *KDD*, volume 98, pages 58–65, 1998.

[HK99] A. Hinneburg and D. A. Keim. Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. pages 506–517, 1999.

[HMB+] M. Hinne, A. Meijers, R. Bakker, P. H. E. Tiesinga, M. Mørup, and M. A. J. Van Gerven. The missing link: Predicting connectomes from noisy and partially observed tract tracing data.

[Hol06] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on visualization and computer graphics*, 12(5):741–748, 2006.

[HSC+09] C. J. Honey, O. Sporns, L. Cammoun, X. Gigandet, J. P. Thiran, R. Meuli, and P. Hagmann. Predicting human resting-state functional connectivity from structural connectivity. *Proceedings of the National Academy of Sciences*, 106(6):2035–2040, 2009.

[HTG+15] Z. R. Hesabi, Z. Tari, A. Goscinski, A. Fahad, I. Khalil, and C. Queiroz. Data summarization techniques for big data—a survey. In *Handbook on Data Centers*, pages 1109–1152. Springer, 2015.

[hum] Human Brain Project. `https://www.humanbrainproject.eu/en/`. [Online; accessed 2018-01-24].

[JOP] E. Jones, T. Oliphant, and P. Peterson. Scipy: Open source scientific tools for python. `http://www.scipy.org/`, 2001–. [Online; accessed 22-06-2018].

[JP17] A. P. Joshi and B. V. Patel. Issues in real time knowledge discovery through data stream mining. *International Journal of Scientific Research in Science and Technology, 3*, pages 132–135, 2017.

[Kai11] M. Kaiser. A tutorial in connectome analysis: topological and spatial features of brain networks. *NeuroImage*, 57(3):892–907, 2011.

[KERKT16] K. Knoblauch, M. Ercsey-Ravasz, H. Kennedy, and Z. Toroczkai. The brain in space. In *Micro-, meso-and macro-connectomics of the brain*, pages 45–74. Springer, 2016.

[KF11] U. Kang and C. Faloutsos. Beyond caveman communities: Hubs and spokes for graph compression and mining. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 300–309. IEEE, 2011.

[KLKF14] U. Kang, J. Y. Lee, D. Koutra, and C. Faloutsos. Net-ray: Visualizing and mining billion-scale graphs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 348–361. Springer, 2014.

[KR90] L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction To Cluster Analysis*. John Wiley & Sons, 01 1990.

[KS05] S. H. Koslow and S. Subramaniam. *Databasing The Brain : From Data to Knowledge (Neuroinformatics)*. Wiley, 2005.

[LBK+18] F. Lekschas, B. Bach, P. Kerpedjiev, N. Gehlenborg, and H. Pfister. HiPiler: Visual Exploration of Large Genome Interaction Matrices with Interactive Small Multiples. *IEEE Transactions on visualization and computer graphics*, 24(1):522–531, 2018.

[LCM07] S. N. Lahiri, A. Chatterjee, and T. Maiti. Normal approximation to the hypergeometric distribution in nonstandard cases and a sub-gaussian berry–esseen theorem. *Journal of Statistical Planning and Inference*, 137(11):3570–3590, 2007.

[LF06] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 631–636, New York, NY, USA, 2006. ACM.

[LHA+07] E. S. Lein, M. J. Hawrylycz, N. Ao, M. Ayres, A. Bensinger, A. Bernard, and A. F. Boe. Genome-wide atlas of gene expression in the adult mouse brain. *Nature*, 445:168–176, 2007.

[lin] Scipy linkage hierarchical clustering. `https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html`. [Online; accessed 23-01-2019].

[LRRF11] A. Lancichinetti, F. Radicchi, J. J. Ramasco, and S. Fortunato. Finding statistically significant communities in networks. *PloS one*, 6(4):e18961, 2011.

[LRU14] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of massive datasets*. Cambridge university press, 2014.

[MBWG13] D. S. Margulies, J. Böttger, A. Watanabe, and K. J. Gorgolewski. Visualizing the human connectome. *NeuroImage*, 80:445–461, 2013.

[MERG+14] N. T. Markov, M. M. Ercsey-Ravasz, A. R. Ribeiro Gomes, C. Lamy, L. Magrou, J. Vezoli, P. Misery, A. Falchier, R. Quilodran, M. A. Gariel, J. Sallet, R. Gamanut, C. Huissoud, S. Clavagnier, P. Giroud, D. Sappey-Marinier, P. Barone, C. Dehay, Z. Toroczkai, K. Knoblauch, D. C. Van Essen, H. Kennedy, and Henry Kennedy. A Weighted and Directed Interareal Connectivity Matrix for Macaque Cerebral Cortex. *Cerebral Cortex*, 24:17–36, 2014.

[Mir17] B. Mirzasoleiman. Big data summarization using submodular functions, 2017.

[MMF+11] N. T. Markov, P. Misery, A. Falchier, C. Lamy, J. Vezoli, R. Quilodran, M.A. Gariel, P. Giroud, and al. Weight consistency specifies regularities of macaque cortical networks. *Cerebral Cortex*, 21:1254–1272, 2011.

[MNP17] S. McCurdy, V. Ntranos, and L. Pachter. Column subset selection for single-cell rna-seq clustering. *bioRxiv*, 2017.

[MORG91] J. W. McClurkin, L. M. Optican, B. J. Richmond, and T. J. Gawne. Concurrent processing and complexity of temporally encoded neuronal messages in visual perception. *Science*, 253(5020):675–677, 1991.

[MS05] E. Mäkinen and H. Siirtola. The barycenter heuristic and the reorderable matrix. *Informatica (Slovenia)*, 29(3):357–364, 2005.

[NH94] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 144–155, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[Nov18] A. Novikov. annoviko/pyclustering: pyclustering 0.8.1 release. `https://doi.org/10.5281/zenodo.1254845`, May 2018.

[num] Numpy library. `http://www.numpy.org/`. [Online; accessed 23-01-2019].

[Nyl17] J. Nylén. Exploring ways of visualizing functional connectivity, 2017.

[OHN+14] S. W. Oh, J. A. Harris, L. Ng, B. Winslow, and N. Cain. A mesoscale connectome of the mouse brain. *Nature*, 508(7495):207–214, 2014.

[OW14] S. C. Olhede and P. J. Wolfe. Network histograms and universality of blockmodel approximation. *Proceedings of the National Academy of Sciences*, 111(41):14722–14727, 2014.

[PBF16] S. Papadimitriou, A. Brockwell, and C. Faloutsos. Adaptive, automatic stream mining. In *Data Stream Management*, pages 499–528. Springer, 2016.

[PCN+11] J. D. Power, A. L. Cohen, S. M. Nelson, G. S. Wig, K. A. Barnes, J. A. Church, A. C. Vogel, T. O. Laumann, F. M. Miezin, and B. L. Schlaggar. Functional network organization of the human brain. *Neuron*, 72(4):665–678, 2011.

[PKB14] D. Papailiopoulos, A. Kyrillidis, and C. Boutsidis. Provable deterministic leverage score sampling. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 997–1006, New York, NY, USA, 2014. ACM.

[RAM+15] J. Richiardi, A. Altmann, A. C. Milazzo, C. Chang, M. Mallar Chakravarty, T. Banaschewski, G. J. Barker, A. L.W. Bokde, J. B. Poline, M. D. Greicius, and I. Consortium. Correlated gene expression supports synchronous activity in brain networks. *Science*, 348(6240):1241–1244, 2015.

[rea] React - a javascript library for building user interfaces. `https://reactjs.org/`. [Online; accessed 23-01-2019].

[red] Redux. a predictable state container for javascript apps. `https://redux.js.org/`. [Online; accessed 23-01-2019].

[RRG+09] P. Roca, D. Rivì Ere, P. Guevara, C. Poupon, and J. F. Mangin. Tractography - based Parcellation of the Cortex using a Spatially - informed Dimension Reduction of the Connectivity Matrix. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2009*, pages 935–942, 2009.

[RS10] M Rubinov and O. Sporns. Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage*, 52(3):1059 – 1069, 2010. Computational Models of the Brain.

[Sch96] E. Schikuta. Grid-clustering: An efficient hierarchical clustering method for very large data sets. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on Pattern Recognition*, volume 2, pages 101–105. IEEE, 1996.

[sci] Scipy library. `https://scipy.org/scipylib/`. [Online; accessed 23-01-2019].

[SCKH04] O. Sporns, D. R. Chialvo, M. Kaiser, and C. C. Hilgetag. Organization, development and function of complex brain networks. *Trends in cognitive sciences*, 8(9):418–425, 2004.

[SCZ00] G. Sheikholeslami, S. Chatterjee, and A. Zhang. Wavecluster: a wavelet-based clustering approach for spatial data in very large databases. *The VLDB Journal—The International Journal on Very Large Data Bases*, 8(3-4):289–304, 2000.

[SEKX98] J. Sander, M. Ester, H. P. Kriegel, and X. Xu. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data mining and knowledge discovery*, 2(2):169–194, 1998.

[SHL+10] J. L. Stein, X. Hua, S. Lee, A. J. Ho, A. D. Leow, A. W. Toga, A. J. Saykin, L. Shen, T. Foroud, and N. Pankratz. Voxelwise genome-wide association study (vgwas). *NeuroImage*, 53(3):1160–1174, 2010.

[sim] Measures of similarity.

[sma] Small-world network, wikipedia. `https://en.wikipedia.org/wiki/Small-world_network`. [Online; accessed 2018-06-17].

[SMO+03] P. Shannon, An. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research*, 13(11):2498–2504, 2003.

[Sok58] R. R Sokal. A statistical method for evaluating systematic relationship. *University of Kansas science bulletin*, 28:1409–1438, 1958.

[Spo16] O. Sporns. *Connectome networks: from cells to systems*. Springer, 2016.

[SS11] D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

[STK05] O. Sporns, G. Tononi, and R. Kötter. The human connectome: A structural description of the human brain. *PLOS Computational Biology*, 1(4), 09 2005.

[SWM05] M. P. H. Stumpf, C. Wiuf, and R. M. May. Subnets of scale-free networks are not scale-free: Sampling properties of networks. *Proceedings of the National Academy of Sciences*, 102(12):4221–4224, 2005.

[TB13] N. B. Turk-Browne. Functional interactions as big data in the human brain. *Science*, 342(6158):580–584, 2013.

[TCM16] N. Tang, Q. Chen, and P. Mitra. Graph stream summarization: From big bang to big crunch. In *SIGMOD Conference*, 2016.

[TM67] J. Travers and S. Milgram. The small world problem. *Psychology Today*, 1(1):61–67, 1967.

[US17] S. Ubaru and Y. Saad. Sampling and multilevel coarsening algorithms for fast matrix approximations. *arXiv preprint arXiv:1711.00439*, 2017.

[vdHS11] Martijn P. van den Heuvel and Olaf Sporns. Rich-club organization of the human connectome. *Journal of Neuroscience*, 31(44):15775–15786, 2011.

[WJ63] J. Ward and H. Joe. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

[WS98] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *nature*, 393(6684):440, 1998.

[WWZ+09] J. Wang, L. Wang, Y. Zang, H. Yang, H. Tang, Q. Gong, Z. Chen, C. Zhu, and Y. He. Parcellation-dependent small-world brain functional networks: a resting-state fmri study. *Human brain mapping*, 30(5):1511–1523, 2009.

[WYM97] W. Wang, J. Yang, and R. Muntz. Sting: A statistical information grid approach to spatial data mining. In *VLDB*, volume 97, pages 186–195, 1997.

[XEKS98] X. Xu, M. Ester, H. P. Kriegel, and J. Sander. A distribution-based clustering algorithm for mining in large spatial databases. In *Data Engineering, 1998. Proceedings., 14th International Conference on*, pages 324–331. IEEE, 1998.

[XLYE14] X. Xu, C. H. Lee, and D. Young Eun. A general framework of hybrid graph sampling for complex network analysis. In *INFOCOM, 2014 Proceedings IEEE*. IEEE, 2014.

[XYJ+15] T. Xu, Z. Yang, L. Jiang, X. X. Xing, and X. N. Zuo. A Connectome Computation System for discovery science of brain. *Science Bulletin*, 60(1):86–95, 2015.

[YAC+15] A. Q. Ye, O. A. Ajilore, G. Conte, J. GadElkarim, G. Thomas-Ramos, L. Zhan, S. Yang, A. Kumar, R. L. Magin, A. G. Forbes, and A. D. Leow. The intrinsic geometry of the human brain connectome. *Brain informatics*, 2:197–210, 2015.

[ZFB10] A. Zalesky, A. Fornito, and E. T. Bullmore. Network-based statistic: Identifying differences in brain networks. *NeuroImage*, 53(4):1197 – 1207, 2010.

[ZFB12] A. Zalesky, A. Fornito, and E. Bullmore. On the use of correlation as a measure of network connectivity. *NeuroImage*, 60(4):2096–2106, 2012.

[ZRL96] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases. *SIGMOD Rec.*, 25(2):103–114, June 1996.