5
Big Brain Network Data

In this chapter, we explore the main characteristics of big brain network data that offer unique computational challenges. The brain networks are biologically expected to be both sparse and hierarchical. Such unique characterizations put specific topological constraints onto statistical approaches and models we can use effectively. We explore the limitations of the current approaches used in the field and offer alternative approaches and explain new challenges as well as providing the computional solutions.

5.1 Big data

Wikipedia defines *big data* as data sets that are so large or complex that traditional data processing application software is inadequate to deal with them (`en.wikipedia.org/wiki/Big_data`). Big data is not just about the size of the data although that is the main obstacle of using traditional statistical approaches. Big data usually include data sets with sizes beyond the ability of standard software tools to process and analyze within a reasonable time limit. Even 100MB of data can be big if existing computing resources can only handle 1MB of data at a time. Thus, the size of the data is a *relative* quantity respect to the available computing resources.

If we pick any article in big data literature these days, chances are that we often encounter hardware solutions to solving big data problems. They often suggest increasing more central processing units (CPU) or graphical processing units (GPU) and emphasize the need for cluster or parallel computing. For instance, Boubela et al. (2016) suggests to use parallel computing as a way to compute large-scale Pearson correlation coefficients for 390GB of data in the Human Connectome Project (HCP) but did not suggest any other simpler algorithmic approaches that can be implemented in a limited computing resource environment. Simply adding more hardware is not necessarily an effective but costly strategy for big data. Such hardware approaches often do not provide a venue for more interesting statistical problems. Further, the access to fast computational resources is not necessarily given to everyone. Many biological laboratories still do not have technical expertise of using cluster or parallel computing. Therefore, it is often necessary to develop more algorithmic and statistical approaches in addressing big data at least for biological sciences.

Big brain image and network data that offer unique computational challenges. The brain networks are biologically expected to be both sparse and hierarchical. Such unique characterizations put specific topological constraints onto statistical approaches and models we can use effectively (Chung, 2018).

5.1.1 Large-scale brain images

Many big datasets introduce unique computational and statistical challenges that include scalability, storage bottleneck, data representation visualization, and computation mostly related to sample sizes (Fan et al., 2014). However, the challenges in big brain imaging datasets such as HCP and Alzheimer's Disease Neuroimaging Initiative (ADNI; `adni.loni.usc.edu`) are slightly different.

The majority of functional and structural connectivity studies in brain imaging are usually performed following the standard analysis framework (Gong et al., 2009; Hagmann et al., 2007; Fornito et al., 2010; Zalesky et al., 2010). From 3D whole brain images, n regions of interest (ROI) are identified and serve as the nodes of the brain network. Measurements at ROIs are then correlated in a pairwise fashion to produce the connectivity matrix of size $p \times p$. The connectivity matrix is then thresholded to produce the adjacency matrix consisting of zeros and ones that define the link between two nodes. The binarized adjacency matrix is then used to construct the brain network. However, for a large number of nodes, this brute force approach has a serious computational bottleneck of manipulating huge number of connections and storing them. Even if we solve the computational problem, biomedical interpretation of network will be difficult with the huge number of links. For example, for 3×10^5 voxels in an image, we can possibly have a total of 9×10^{10} edges in the graph.

Further, there are substantially more number of voxels (p) per image than the number of images (n) in the datasets. Even at 3mm low resolution, functional magnetic resonance images (fMRI) has more than 25000 voxels (Chung et al., 2017a). Unless the dataset consists of more than 25000 images, brain imaging is often the problem of *small-n large-p*, which is different from the usual big data setting where n is often big. HCP and ADNI have n in the range of a thousands, far smaller than the number of voxels.

Traditionally, numerical accuracy has been less of concerns in brain imaging particularly due to spatial and temporal smoothing often done in images to smooth out various image processing artifacts and physiological noises. Due to the increased sample size and the central limit theorem, which is further reinforced by smoothing, the statistical distribution of the data might become less of a concern in big imaging data (Salmond et al., 2002).

In the traditional mass univariate approaches (Chung et al., 2015a; Worsley et al., 1992), where statistical inference is done at each voxel, the problem of *small-n large-p* is not critical. Further, spatial smoothing has the effect of reducing the number of *resolution element* (RESEL), so we have far less number of effective p (Worsley et al., 1992). Smoothing also reduces the effect of image registration errors and high frequency noise. Gaussian kernel smoothing introduces continuous hierarchical structure through scale space (Worsley et al., 1996a). However, small-n large-p problems become critical in brain network modeling, where we need to correlate different voxels. In the small-n large-p setting, the sample covariance and correlation matrices are no longer positive definite. Subsequently, up to $p - n$ nodes are statistically dependent although there might be *no* true dependency at all. Thus, there is need to constrain the the covariance or correlation matrices by regularization methods such as sparse network models. Unfortunately, for large p, many sparse models have severe computational bottlenecks (Chung et al., 2015a).

There begin to emerge large-scale brain networks with more than 25000 nodes, where each voxel is taken as a network node (Figure 5.1) (Chung et al., 2017a; Eguíluz et al., 2005; Hagmann et al., 2007; Taylor et al., 2017). The size of such large-scale brain networks can easily match publicly available network data such as Stanford Large Network Dataset (`snap.stanford.edu/data`). In such large-scale networks, the small-n large-p problem will be more severe. This type of big data requires more *scalable* and *robust* solutions. possibly using sparse penalties.

Many traditional statistical method that perform well for reasonable sample size do not scale well with big data. Many likelihood or L_1-norm optimization techniques do not scale well with big data. Methods should likely to scale linearly or at least polynomially to even able to compute in a tolerable time limit. Any method that scales exponentially with increased sample size is not going to be useful. Permutation test is one such method. Any method that requires the complete data set as an input is not going to be useful as well. The following are the desirable properties of efficient computational methods for big data.

1) An effective method should be unbiased and sufficiently fast. The *itera-*

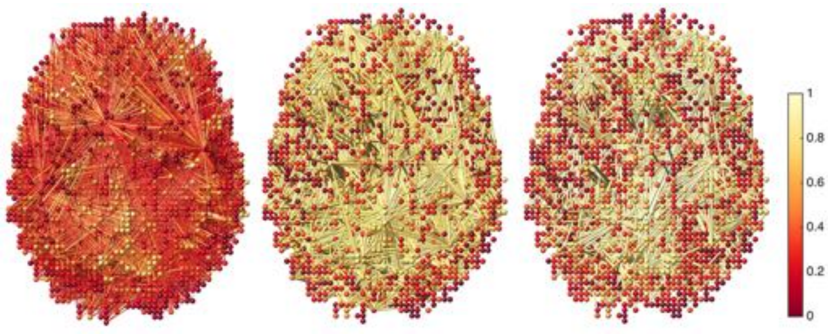

Figure 5.1 Left: Dense fMRI correlation network consisting of more than 25000 nodes (Chung et al., 2017a). The network is so dense, simply displaying all the nodes and edges of the network is not very informative. It is necessary to represent such dense network more sparsely. The sparse correlation network model with sparse parameters $\lambda = 0.7$ (middle) and $\lambda = 0.8$ (right). It can be shown that they form a nested hierarchy called the graph filtration.

tive residual fitting algorithm, originally developed for estimating more than 20000 spherical harmonic coefficients per brain, may offer one such solution (Chung et al., 2008a). The algorithm sequentially breaks down one gigantic problem into smaller problems in an iterative fashion.

2) An effective method should require only a small subset of the full data in accurately estimating the underlying model. Online algorithms, which we will discuss later, are such methods.

5.1.2 Large-scale brain networks

Purely data-driven approaches for large-scale brain networks are not going to be computationally efficient or effective. It is often necessary to incorporate the first-order principles of brain networks into models to possibly reduce computational bottlenecks. Large-scale brain networks are often characterized by sparsity and hierarchy (Chung, 2018). The sparsity and hierarchy is highly relevant topological characterization to other types of big network data such as social networks (Christakis and Fowler, 2007), World Wide Web (WWW) (Adamic, 1999) and genomic regulatory networks (Luscombe et al., 2004). Given any type of real world network, it is unlikely that all the nodes are densely connected to each other. It is expected that the network to have sufficient sparsity. Many large scale networks such as social networks and WWW show scale-free characteristic, which is the main characteristic of hierarchical networks. Although we don't expect all networks to be hierarchical or sparse, these aspect of brain network should be applicable to other big network data. In this section, we will explore two main characterizations of brain networks that should be utilized in even small data settings. We will further explore various statistical challenges related to such characterizations.

5.2 Sparsity

At the microscopic level, the activation of cortical neurons in the brain show *sparse* and widely distributed patterns (Histed and Reid, 2009). At the macroscopic level, diffusion tensor imaging (DTI) can produce up to a half million white matter fiber tracts per brain. Even then not every part of the brain is anatomically connected to other parts of the brain but sparsely connected (Chung et al., 2017b). This can be seen from Figure 5.2, where the brain is parcellated into 116 disjoint regions and the number of white matter fiber tracts passing between the regions is used in constructing the structural connectivity matrix (Chung et al., 2017b). Even though the white matter fibers are very dense, the resulting connectivity matrix is sparse. For 116×116 connectivity matrix, 60% of entries are zeros. As we increases the number of parcellations, the sparsity increases while the the total degree of all nodes decreases (Figure 5.3). Note the degree of nodes counts the number of connections at a node. Thus, it also measures the sparsity of the network.

In fMRI studies, functional connectivity, which measures the dependency of brain activity in one region to another region, is often measured by correlation, covariance or spectral coherence of fMRI time time series. Since the brain does not activate everywhere simultaneously (Chung et al., 2017a), functional connectivity is also expected to be not dense but sparsely clustered. It is reasonable to assume both functional and anatomical brain networks are sparsely connected at the both microscopic and macroscopic levels. Thus, there is strong biological justifications for modeling brain networks sparsely.

The small-n large-p problem in brain imaging often produces under-determined models with infinitely many possible solutions. Such problems are usually remedied by regularizing the systems with additional sparse penalties. Sparse models used in brain imaging include compressed sensing (CS) (Lee et al., 2011c), sparse correlations (Chung et al., 2017a), least absolute shrinkage and selection operator (LASSO) (Huang et al., 2009; Lee et al., 2011c), sparse canonical correlations (Avants et al., 2010) and graphical-LASSO (Chung et al., 2015a; Huang et al., 2009). Most of these sparse models require optimizing $L1$-norm penalties, which has been the major computational bottleneck for solving large-scale problems in brain imaging. Thus, almost all sparse brain

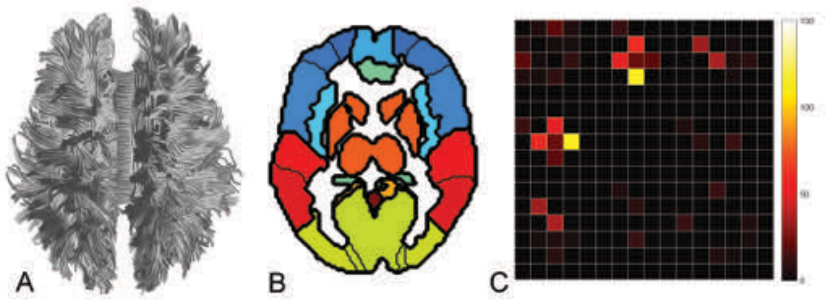

Figure 5.2 A. White matter fiber tracts obtained from a tractography algorithm. B. The brain is parcellated into 116 disjoint regions. C. Connectivity matrix showing how each region is connected to other regions. Even thoroughly fiber tracts are very dense, the resulting connective matrix is always sparse since not every part of brain is connected to each other (Chung et al., 2017b).

network models have been restricted to a few hundreds nodes or less. 2527 MRI features used in a LASSO model for Alzheimer's disease (Xin et al., 2015) is probably the largest number of features used in any sparse model in the brain imaging literature. Recently, a more scalable large-scale sparse brain network models, where each voxel is a network node, are begin to emerge (Chung et al., 2017a). For such large-scale network construction, faster scalable algorithms are needed.

There are few previous studies at speeding up the computation for sparse models. By identifying block diagonal structures in the estimated inverse covariance matrix, it is possible to reduce the computational burden in the penalized log-likelihood method (Mazumder and Hastie, 2012). In (Chung et al., 2017a), the computational bottleneck of $L1$-optimization is overcame by simplifying the sparse network problem into an orthogonal design in LASSO (least absolute shrinkage and selection operator) (Tibshirani, 1996). Other promising methods include a constrained L_1-minimization estimator (CLIME) (Wang et al., 2016a) and faster computations for graphical-LASSO (Witten et al., 2011) although they were never applied to large-scale brain networks yet.

Any sparse brain network model is usually parameterized by a tuning parameter that controls the sparsity of the solution. Increasing the sparse parameter makes the solution more sparse. Thus, sparse models are inherently multiscale, where the scale of the model is determined by the sparsity. Many existing sparse network models use a fixed parameter λ that may not be optimal in other datasets or studies. Depending on the choice of the sparse parameter, the final network structure will be different (Chung et al., 2015a; Lee et al.,

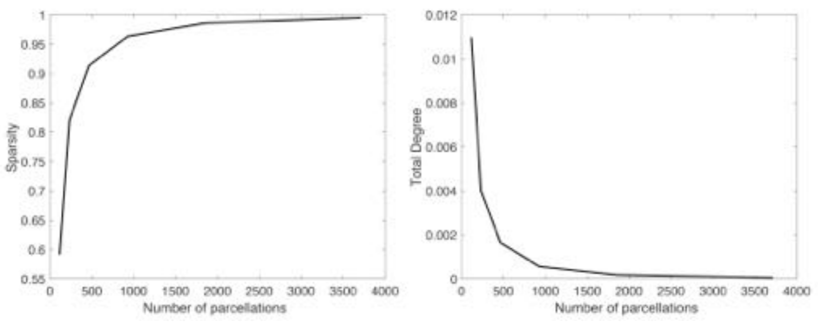

Figure 5.3 Left: plot of sparsity over the number of pacellations. The sparsity is measures as the ratio of zero entries over all entries in the connectivity matrix. Right: plot of total degree of nodes over the number of pacellations. The vertical axis measures the ratio of the total number of connections over every possible connection. The plots all show the sparse nature of brain networks at any spatial scale.

2012). There is a need to develop a multiscale sparse network model that provide a consistent analysis results and interpretation regardless of the choice of parameter (Chung et al., 2015a, 2017a).

5.3 Hierarchy

Brain networks are fundamentally *multiscale*. Intuitive and palatable biological hypothesis is that brain networks are organized into *hierarchies* (Betzel and Bassett, 2017). A brain network at any particular sale might be subdivided into subnetworks, which can be further subdivided into smaller subnetworks in an iterative fashion. There have been various attempts at modeling brain networks at multiple scales (Betzel and Bassett, 2017; Chung et al., 2015a, 2017a; Lee et al., 2012) . Unfortunately, many multiscale models give raise to conflicting topological structures of the networks from one scale to the next. For instance, the estimated modular structure in the multiscale community detection problem usually do not have continuity over different resolution parameters (Betzel and Bassett, 2017). The topological structure of parcellation at one particular scale may not carry over to different scales (Zalesky et al., 2010; Betzel and Bassett, 2017). There is a need to develop a hierarchical parcellation scheme that provide a consistent network analysis results and interpretation regardless of the choice of scale.

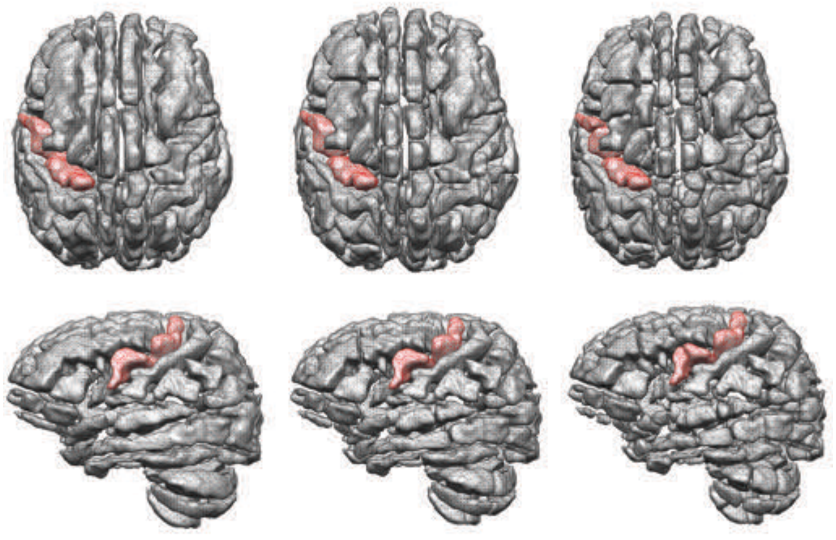

Figure 5.4 Left: AAL parcellation with 116 predefined ROI. Each parcellation is displayed as a disconnected mesh surface. Each ROI serves as a node in 116×116 connectivity matrix. For visualization purpose, we have added artificial empty gaps between mesh surfaces. Red region is the left precentral gyrus. Middle: the second layer of the hierarchical parcellation with 2×116 regions. Each AAL parcellation is subdivided into two disjoint regions. Right: the third layer of the hierarchical parcellation with 4×116 regions.

5.3.1 Hierarchical structural parcellation

It is possible to come up with a new hierarchical parcellation scheme based on the Courant nodal domain theorem (Courant and Hilbert, 1953). The method is related to graph cuts (Shen et al., 2010) and spectral clustering (Craddock et al., 2012; Pepe et al., 2015) based parcellation schemes previous used in parcellating the resting-state functional magnetic resonance images. However, unlike previous parcellation approaches, it is possible to provides hierarchical nestedness and, thus, preserves topology across different spatial resolutions. This can be achieved through Courant nodal domain theorem.

Courant nodal domain theorem. For Laplacian Δ in a compact domain $\mathcal{M} \subset \mathbb{R}^3$, consider eigenvalues

$$0 = \lambda_0 < \lambda_1 \leq \lambda_2 \leq \cdots$$

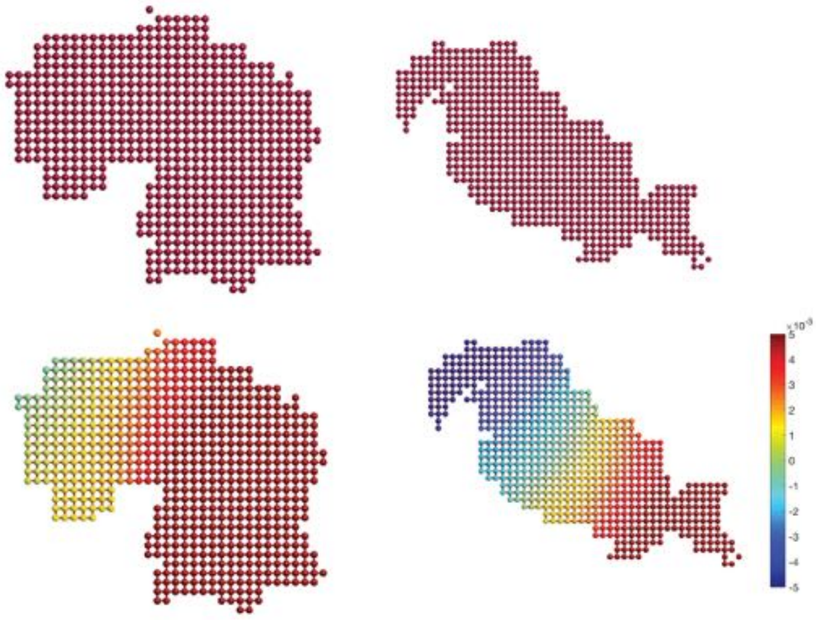

Figure 5.5 Top: A 3D binary image is converted to a 3D graph based on the 18-connected neighbor scheme. Here only 2D image slice result is shown. Bottom: The Fidler's vector, which is the first nontrivial eigenvector of the graph Laplacian is then computed. By clustering the graph depending on the sign of the Fidler's vector, we can split the graph into two disjoint regions.

and eigenfunctions $\psi_0, \psi_1, \psi_2, \cdots$ satisfying

$$\Delta\psi_j(p) = \lambda_j \psi_j(p).$$

We then have $\psi_0(p) = 1/\sqrt{\mu(\mathcal{M})}$, where $\mu(\mathcal{M})$ is the volume of $\mathcal{M}$. From the orthogonality of eigenfunctions, we have

$$\int_{\mathcal{M}} \psi_0(p)\psi_1(p)\, d\mu(p) = 0,$$

Thus, ψ_1 must be take positive and negative values. The Courant nodal domain theorem (Courant and Hilbert, 1953) further states that ψ_1 divides $\mathcal{M}$ into two disjoint regions by the nodal surface boundary $\psi_1(p) = 0$. When the domain is discretized as a 3D graph, the second eigenfunction ψ_1 is called the Fiedler vector. It is often used in spectral clustering and graph cuts (Shen et al., 2010; Chung et al., 2011b). Applying iteratively the nodal domain theorem, we can hierarchically partition $\mathcal{M}$ in a nested fashion. Once domain $\mathcal{M}$ is divided into

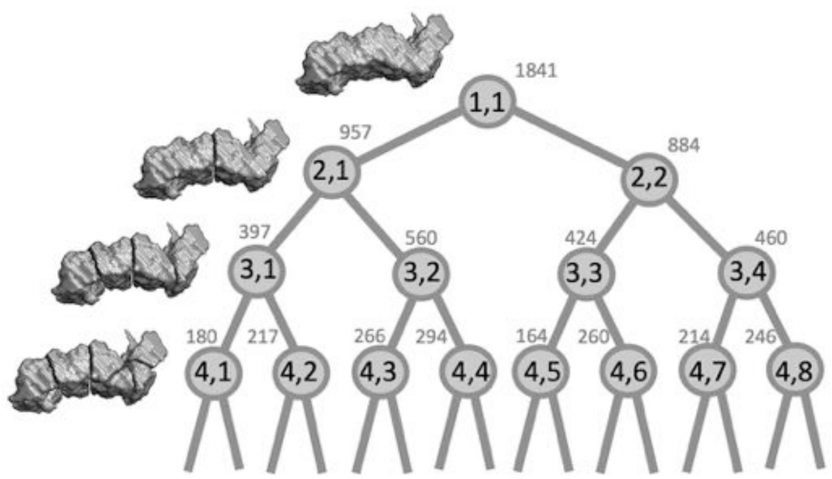

Figure 5.6 The hierarchical parcellation of a AAL ROI (left precentral gyrus shown in Figure 5.7) up to the fourth layer. At the fourth layer, we have 2^3 subparcellations. The hierarchical parcellation is represented as a binary tree and indexed using two numbers in computer implementation (layer, indexing for parcellation). The root node (1,1) is the AAL parcellation. At the next layer, nodes (2,1), (2,2) correspond to the partition of the AAL parcellation.

$\mathcal{M}_1$ and $\mathcal{M}_2$ based on the sign of the second eigenfunction, we iteratively recompute the second eigenfunction of Laplacian restricted to $\mathcal{M}_1$ and $\mathcal{M}_2$. Then proceed with subpartitioning $\mathcal{M}_1$ and $\mathcal{M}_2$ further.

The Courant nodal domain theorem can be discretely applied to the AAL parcellation as follows. We first convert the binary volume of each parcellation in AAL into a 3D graph by taking each voxel as a node and connecting neighboring voxels. Using the 18-connected neighbor scheme, we connect two voxels only if they touch each other on their faces or edges. If voxels are only touching at their corner vertices, they are not considered as connected. This results in an adjacency matrix and the 3D graph Laplacian. The computed Fiedler vector is then used to partition each AAL parcellation into two disjoint regions (Figures 5.4 and 5.7). For each disjoint subregion, we further recompute the Fiedler vector restricted to the subregion. This binary partition process iteratively continues till all the partitions are voxels. We are doubling the number of parcellations at each iteration. There are a total of $p = 116$ parcellations in layer 1 and $2 \cdot 115$ parcellations in layer 2. At the i-th layer, there are $2^{i-1} \cdot 116$ parcellations. This way, we can construct 20-layer nested hierarchical parcellations all the way to the voxel-level. Since the number of voxels are not uniform

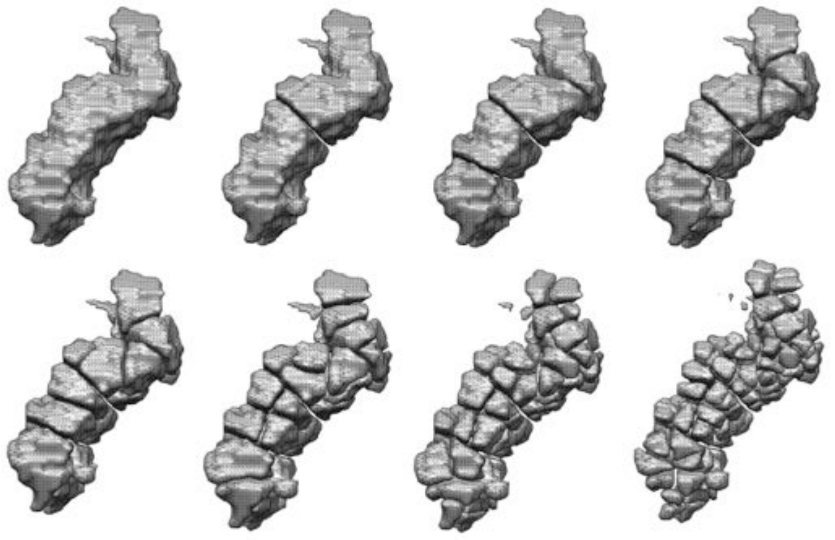
Figure 5.7 Hierarchical parcellation of the left precentral gyrus shown in Figure 5.4) up to the 8-th layer. At the 8-th layer, we have $2^{8-1} = 128$ parcellations of the gyrus. The hierarchical parcellation continues till every voxel is a parcellation.

across AAL parcellations, we are *approximately* doubling the number of parcellations at each iteration.

It is possible to hieratically parcellate the brain regions into small subregions. This requires first converting the binary volume of each parcellation into an adjacency matrix, which represent how each voxel is connected to other voxels. Using the 18-connected neighbor scheme, we define two voxels are connected only if they touch each other on their faces or edges. If they are only touching at their corner vertices, they will not be considered as connected. Figure 5.5 shows the result of binary voxels converted to graphs on two representative slices. Depending on the scale of parcellation, the parameters of graph, which characterize graph topology, varies considerably up to 95$\%$ (Fornito et al., 2010; Zalesky et al., 2010). Thus, there is a need for parcellation-free brain network node identification methods.

5.3.2 Hierarchical structural connectivity

At the each layer of the hierarchical parcellation (Chung et al., 2018b), we count the total number of white matter fiber tracts connecting parcellations as a measure of connectivity. The resulting connectivity matrices form a *convolu-*

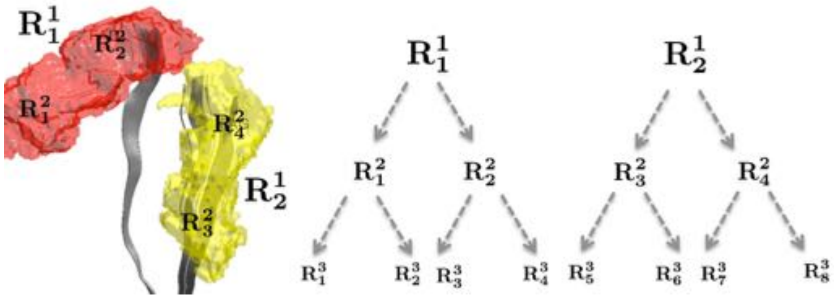

Figure 5.8 Two representative AAL parcellations $\mathbf{R}_1^1$ (right precentral gyrus) and $\mathbf{R}_2^1$ (left precentral gyrus) at the first layer will be partitioned into four subregions $\mathbf{R}_1^2, \mathbf{R}_2^2, \mathbf{R}_3^2, \mathbf{R}_4^2$ at the second layer. The fiber tracts will be counted between the parcellations.

tional network. Let S_{jk}^i denote the total number of tracts between parcellations $\mathbf{R}_j^i$ and $\mathbf{R}_k^i$ at the i-th layer (Figure 5.8). The connectivity S_{jk}^i at the i-th layer is then the sum of connectivities at the $(i+1)$-th layer (Figure 5.9), i.e.,

$$S_{jk}^i = \sum_{\mathbf{R}_l^{i+1} \subset \mathbf{R}_j^i} \sum_{\mathbf{R}_m^{i+1} \subset \mathbf{R}_k^i} S_{lm}^{i+1}.$$

The sum is taken over every subparcellation of $\mathbf{R}_j^i$ and $\mathbf{R}_k^i$. This provide a subject-level connectivity matrix. The connectivity matrix $S^i = (S_{jk}^i)$ is expected to be very sparse at any hierarchy (Figure 5.9). Note the diagonal elements of matrix $S^i = (S_{jk}^i)$, which defines node values, are unspecified yet. Then connectivity matrix S^i will also have a hierarchical structure.

Persistent homology may offer an effective framework in addressing the topological inconsistency in multiscale models. Instead of studying images and networks at a fixed scale, as usually done in traditional approaches, persistent homology summarizes the changes of topological features over different scales and identifies the most persistent topological features that are robust under different scales. This robust performance under different scales is needed for network models that are parameter and scale dependent. Instead of building networks at one fixed parameter that may not be optimal, persistent homological approaches exploit the topological structure of the data and models. In doing so, topologically consistent nested hierarchical networks called the *graph filtration* is obtained (Lee et al., 2012; Chung et al., 2015a). Such a nested hierarchical structure can further speed up various computations even

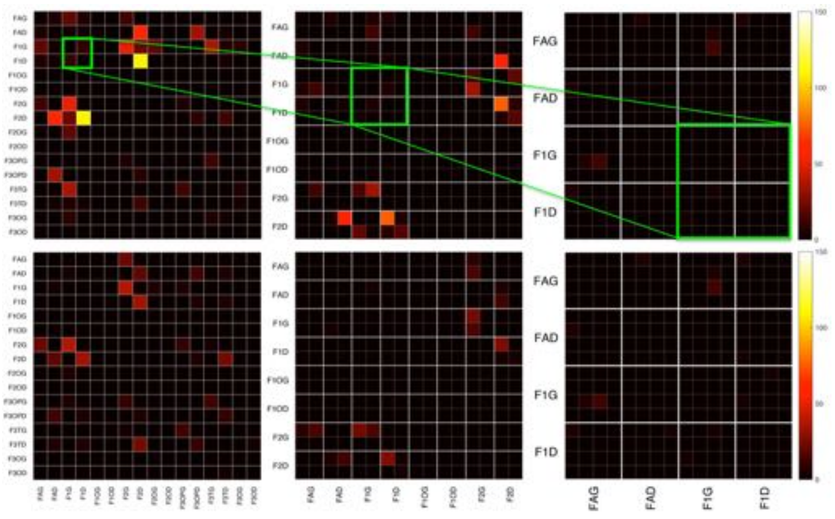

Figure 5.9 The hierarchical connectivity matrices of MZ- (top) and DZ-twins (bottom). The parts of connectivity matrices of the layers 1, 2 and 3 are shown. They form a layered convolutional network, where the convolution is defined as the sum of tracts between sub-parcellations.

for large-scale networks with a billions of connections (Chung et al., 2017a).

5.4 Computing large correlation matrices

For constructing large-scale correlation-based brain networks, different type of correlation computation technique is needed. Existing built-in functions in MATLAB and R packages for computing correlation matrices are not necessarily optimized for large-scale data. For large correlation matrices of size $p \times p$, where $p > 10000$, the matrix computation takes long time and requires significant memory. The built-in functions in MATLAB and R packages are often not written in a computationally efficient fashion. The pairwise computation of correlations is not efficient. A more efficient way is to scale and normalize the data matrice first and compute the correlation matrix as a matrix product (Chung et al., 2015a, 2017a).

Suppose we have $n \times p$ data matrices $X = (x_{ij})$ and $Y = (x_{ij})$, where n is the number of images and p is the number of voxels we are interested in

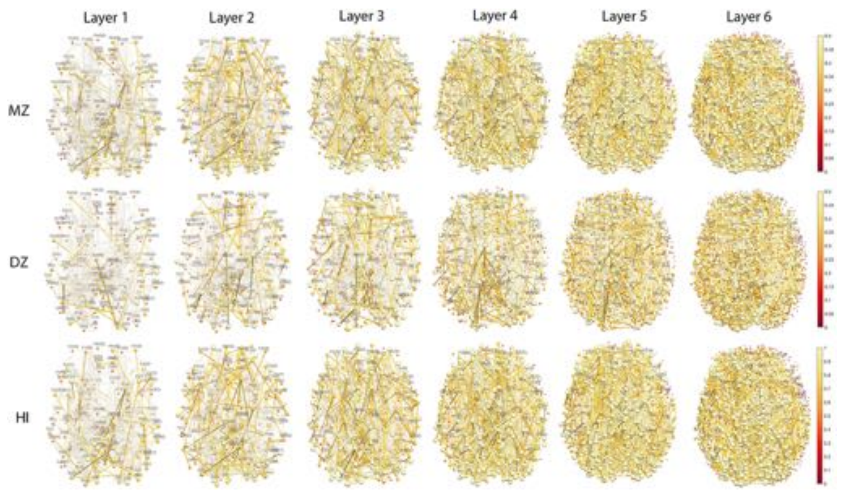

Figure 5.10 Top, middle: Edge colors are Spearman's rank correlations thresholded at 0.3 for MZ- and DZ-twins for different layers. Node colors are the maximum correlation of all the connecting edges. Bottom: Edge colors are the heritability index (HI). Node colors are the maximum HI of all the connecting edges. MZ-twins show higher correlations compared to DZ-twins. The node and edge sizes are proportionally scaled.

computing correlations. We scale and normalize along the columns such that

$$\sum_{i=1}^{n} x_{ij} = 0, \quad \sum_{i=1}^{n} x_{ij}^2 = 1, \tag{5.1}$$

$$\sum_{i=1}^{n} y_{ij} = 0, \quad \sum_{i=1}^{n} y_{ij}^2 = 1. \tag{5.2}$$

Then the correlation matrices of X and Y are defined as

$$corr(X) = X^\top X, \quad corr(Y) = Y^\top Y.$$

If $n \gg p$, the correlation matrices are full rank and invertible. If not, we no longer have a well defined correlation matrices and regularizations are possibly needed. The cross-correlation matrices of X and Y are defined as

$$corr(X, Y) = X^\top Y,$$

which is not symmetric. To make it a symmetric, we can perform a symmetrization:

$$(X^\top Y + Y^\top X)/2.$$

The above procedure can be implemented as

```
function Z = corr2norm(X)
   [n p]=size(X);
   meanX=mean(X,1);
   meanX=repmat(meanX,n,1);
   X1=X-meanX;
   diagX1 = sqrt(diag(X1'*X1));
   X1=X1./repmat(diagX1', n,1);
   Z=X1;
```

Once we normalize the data matrix, the cross-correlation matrix `C` is simply given as the product of martrices:

```
X= corr2norm(X);
Y = corr2norm(Y);
C = X'*Y;
```

This is perhaps the fastest way to compute large-scale correlation matrices. Due to scaling,

```
X'*X= 1
Y'*Y= 1
```

Often, data matrix `X` will likely to have missing values, which are often denoted as `NaN` in MATLAB. The above algorithm assumes there is no missing value. It may necessary to perform missing data treatment for any entry with `NaN`. Figure 5.11 displays the large-scale cross-correlation networks with 25972 nodes constructed using the method (Chung et al., 2017a).

5.5 Online algorithms

In terms of computation, many existing brain image analysis software such as SPM (`www.fil.ion.ucl.ac.uk/spm`) and AFNI (`afni.nimh.nih.gov`) are not effective for big data. The general statistical premise of such mainstream tools is that all the image measurements are available in the computer memory and statistics are computed using all the data. However, in the big data setting, it may not be possible to fit all of the imaging data in a computer's memory, making it necessary to perform the analysis by adding one image at a time in a sequential manner. We need a way to incrementally update the statistical analysis results without repeatedly running the entire analysis whenever new images or parts of images are added.

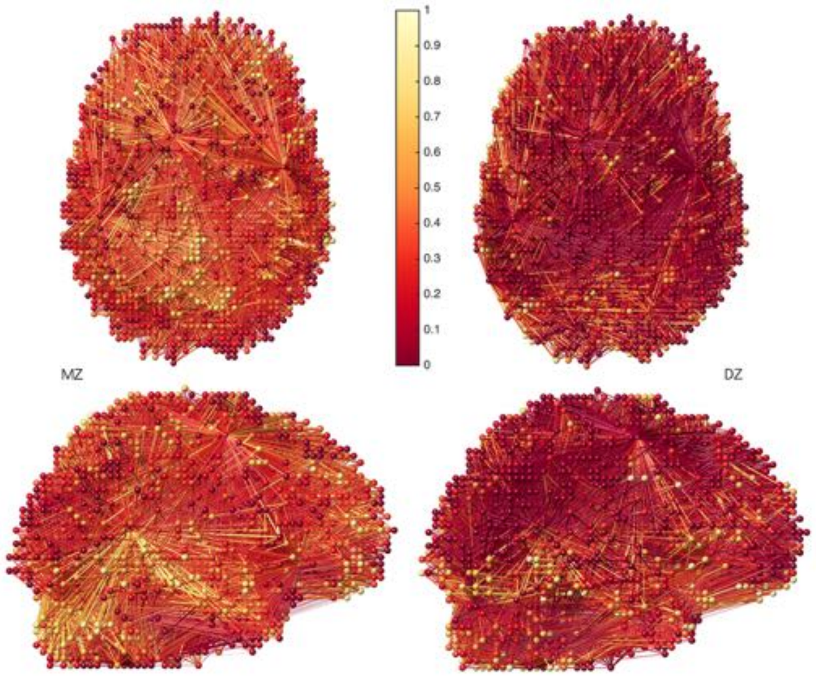

Figure 5.11 fMRI cross-correlation network of monozygotic (MZ) and same-sex dizygotic (DZ) twins with 25972 nodes (Chung et al., 2017a). Only positive correlations are shown.

An *online algorithm* is one that processes its inputted data in a sequential manner (Chung et al., 2017c). Instead of processing the entire set of imaging data from the start, an online algorithm processes one image at a time. That way, we can bypass the memory requirement, reduce numerical instability and increase computational efficiency. With the ever-increasing amount of large-scale brain imaging datasets such as ADNI and HCP, the development of various online statistical method is warranted (Chung et al., 2017c). Thus, here is an immediate need to develop the online version of sparse or hierarchical network models although there are no such available methods yet. Even large-scale Pearson correlation coefficients can be computed using an online algorithm.

Existing statistical analysis packages such as MATLAB and R also assume all measurements to be available in computer memory. Unless substantial modification to existing codes is made, we cannot even compute t-statistics for extremely large data that will not fit into the computer memory using the built-in functions. Thus, there is a strong need to develop online algorithms for big data beyond brain imaging.

5.5.1 Online two-sample t-test

Given data $x_1, \cdots, x_m$, an *online algorithm* for computing the sample mean μ_m is given by

$$\begin{aligned}\mu_m &= \frac{1}{m}\sum_{i=1}^{m} x_i \\ &= \mu_{m-1} + \frac{1}{m}(x_m - \mu_{m-1})\end{aligned}$$

for any $m \geq 1$. The algorithm updates the previous mean μ_{m-1} with new data x_m. This algorithm avoids accumulating large sums and tend to be numerically more stable (Finch, 2009).

An online algorithm for computing the sample variance σ_m^2 is algebraically involved (Chan et al., 1983; Knuth, 1981). After lengthy derivation, it can be shown that

$$\begin{aligned}\sigma_m^2 &= \frac{1}{m-1}\sum_{i=1}^{m}(x_i - \mu_m)^2 \\ &= \frac{m-2}{m-1}\sigma_{m-1}^2 + \frac{1}{m}(x_m - \mu_{m-1})^2\end{aligned}$$

for $m \geq 2$. The algorithm starts with the initial value $\sigma_1^2 = 0$.

For comparing a collection of data between groups, two-sample t-statistic can be used. Given measurements $x_1, \cdots, x_m \sim N(\mu^1, (\sigma^1)^2)$ in one group and $y_1, \cdots, y_n \sim N(\mu^1, (\sigma^2)^2)$ in the other group, the two-sample t-statistic for testing

$$H_0 : \mu^1(x) = \mu^2(x) \;\; vs. \;\; H_1 : \mu^1(x) > \mu^2(x)$$

at each x is given by

$$T_{m,n}(x) = \frac{\mu_m^1 - \mu_n^2 - (\mu^1 - \mu^2)}{\sqrt{(\sigma^1)_m^2/m + (\sigma^2)_n^2/n}},$$

where μ_m^1, μ_n^2, $(\sigma^1)_m^2, (\sigma^2)_m^2$ are sample means and variances in each group estimated using the online algorithm. $T_{m,n}$ is then sequentially computed as

$$T_{1,0} \to T_{2,0} \to \cdots \to T_{m,0} \to T_{m,1} \to \cdots \to T_{m,n}$$

in $m + n$ steps.

5.5.2 Online algorithm for linear regression

The online algorithm for linear regression is itself useful but additionally more useful in constructing an online algorithm for F-tests in the next section. Given data vector $\mathbf{y}_{m-1} = (y_1, \cdots, y_{m-1})^\top$ and design matrix Z_{m-1}, consider linear model

$$\mathbf{y}_{m-1} = Z_{m-1}\boldsymbol{\lambda}_{m-1}$$

with unknown parameter vector $\boldsymbol{\lambda}_{m-1} = (\lambda_1, \lambda_2, \cdots, \lambda_k)^\top$. Z_{m-1} is a matrix of size $(m-1) \times k$. Multiplying $Z_{m-1}^\top$ on the both sides we have

$$Z_{m-1}^\top \mathbf{y}_{m-1} = Z_{m-1}^\top Z_{m-1}\boldsymbol{\lambda}_{m-1} \tag{5.3}$$

Let $W_{m-1} = Z_{m-1}^\top Z_{m-1}$, which is a $k \times k$ matrix. In most applications, there are substantially more data than the number of parameters, i.e., $m \gg k$, and W_{m-1} is invertible. The least squares estimation (LSE) of $\boldsymbol{\lambda}_{m-1}$ is given by

$$\boldsymbol{\lambda}_{m-1} = W_{m-1}^{-1} Z_m^\top \mathbf{y}_{m-1}.$$

When new data y_m is introduced to the linear model (5.3), the model is updated to

$$\begin{pmatrix} \mathbf{y}_{m-1} \\ y_m \end{pmatrix} = \begin{pmatrix} Z_{m-1} \\ z_m \end{pmatrix} \boldsymbol{\lambda}_m,$$

where z_m is $1 \times k$ row vector. Subsequently, we have

$$\begin{aligned} (Z_{m-1}^\top \; z_m^\top) \begin{pmatrix} \mathbf{y}_{m-1} \\ y_m \end{pmatrix} &= (Z_{m-1}^\top \; z_m^\top) \begin{pmatrix} Z_{m-1} \\ z_m \end{pmatrix} \boldsymbol{\lambda}_m \\ Z_{m-1}^\top \mathbf{y}_{m-1} + z_m^\top y_m &= (W_{m-1} + z_m^\top z_m)\boldsymbol{\lambda}_m. \end{aligned}$$

From Woodbury formula (Deng, 2011), we can write

$$(W_{m-1} + z_m^\top z_m)^{-1} = W_{m-1}^{-1} - c_m W_{m-1}^{-1} z_m^\top,$$

where $c_m = 1/(1 + z_m W_{m-1} z_m^\top)$ is scalar. Then we have the explicit online algorithm for updating the parameter vector:

$$\boldsymbol{\lambda}_m = (I - W_{m-1}^{-1} z_m^\top y_m - c_m W_{m-1}^{-1} z_m^\top W_{m-1}^\top)\boldsymbol{\lambda}_{m-1} - c_m W_{m-1}^{-1} z_m^\top z_m^\top y_m,$$

where I is the identity matrix of size $k \times k$. Since the algorithm requires W_{m-1} to be invertible, the algorithm must start from

$$\boldsymbol{\lambda}_k \to \boldsymbol{\lambda}_{k+1} \to \cdots \to \boldsymbol{\lambda}_m.$$

At each iteration, we need to store $k \times k$ matrix W_{m-1}. In many applications, k will not be larger than 10 and most likely around 5 or less, which is manageable as far as computer memory is concerned.

A similar online algorithm for fitting a general linear model (GLM) was introduced for real-time fMRI (Bagarinao et al., 2006), where the Cholesky factorization was used to invert the covariance matrix in solving GLM. Our approach based on Woodbury formula does not require the factorization or inversion of matrices.

5.5.3 Online algorithm for F-test

An online algorithm for the F-test is involved but it is based on the online algorithm for linear regression. Let y_i be the i-th data, $\mathbf{x}_i = (x_{i1}, \cdots, x_{ip})^\top$ be the variables of interest and $\mathbf{z}_i = (z_{i1}, \cdots, z_{ik})^\top$ be nuisance covariates corresponding to the i-th data. We assume there are $m-1$ data to start with. Consider a general linear model (Chung, 2013)

$$\mathbf{y}_{m-1} = Z_{m-1}\boldsymbol{\lambda}_{m-1} + X_{m-1}\boldsymbol{\beta}_{m-1},$$

where $Z_{m-1} = (z_{ij})$ is $(m-1) \times k$ design matrix, $X_{m-1} = (x_{ij})$ is $(m-1) \times p$ design matrix. $\boldsymbol{\lambda}_{m-1} = (\lambda_1, \cdots, \lambda_k)^\top$ and $\boldsymbol{\beta}_{m-1} = (\beta_1, \cdots, \beta_p)^\top$ are unknown parameter vectors to be estimated at the $(m-1)$-th iteration. Consider hypotheses

$$H_0 : \boldsymbol{\beta} = 0 \text{ vs. } H_1 : \boldsymbol{\beta} \neq 0.$$

The reduced null model when $\boldsymbol{\beta} = 0$ is

$$\mathbf{y}_{m-1} = Z_{m-1}\boldsymbol{\lambda}^0_{m-1}.$$

The goodness-of-fit of the null model is measured by the sum of the squared errors (SSE):

$$\text{SSE}^0_{m-1} = (\mathbf{y}_{m-1} - Z_{m-1}\boldsymbol{\lambda}^0_{m-1})^\top(\mathbf{y}_{m-1} - Z_{m-1}\boldsymbol{\lambda}^0_{m-1}),$$

where $\boldsymbol{\lambda}^0_{m-1}$ is estimated using the online algorithm (5.4). This provide the sequential update of SSE under H_0:

$$\text{SSE}^0_k \to \text{SSE}^0_{k+1} \to \cdots \to \text{SSE}^0_m.$$

Similarly the fit of the alternate full model is measured by

$$\text{SSE}^1_{m-1} = (\mathbf{y}_{m-1} - \mathbb{Z}_{m-1}\boldsymbol{\gamma}^1_{m-1})^\top(\mathbf{y}_{m-1} - \mathbb{Z}_{m-1}\boldsymbol{\gamma}^1_{m-1}),$$

where $\mathbb{Z}_{m-1} = [Z_{m-1}X_{m-1}]$ is the combined design matrix and of size $(m-1) \times (k+p)$ and

$$\boldsymbol{\gamma}^1_{m-1} = \begin{pmatrix} \boldsymbol{\lambda}^1_{m-1} \\ \boldsymbol{\beta}^1_{m-1} \end{pmatrix}$$

is the combined parameter vector of size $(k+p)\times 1$. Similarly using the online algorithm (5.4), SSE under H_1 is given as

$$\text{SSE}^1_{k+p} \to \text{SSE}^1_{k+1} \to \cdots \to \text{SSE}^1_m.$$

Under H_0, the test statistic at the m-th iteration f_m is given by

$$f_m = \frac{(\text{SSE}_0 - \text{SSE}_1)/p}{\text{SSE}_0/(m-p-k)} \sim F_{p,m-p-k},$$

which is the F-statistic with p and $m-p-k$ degrees of freedom.

5.5.4 Online algorithm for correlation matrix

Using a similar iterative principle, it is possible to update Pearson correlation when new data is added. Bivariate data $\mathbf{x}_n = (x_1, x_2, \cdots, x_n)$ and $\mathbf{y}_n = (y_1, y_2, \cdots, y_n)$ are given. The Pearson correlation between $\mathbf{x}_n$ and $\mathbf{y}_n$ is given by $\rho(\mathbf{x}_n, \mathbf{y}_n)$. Suppose new data x_{n+1} and y_{n+1} are added to existing data $\mathbf{x}_n$ and $\mathbf{y}_n$ respectively. Then it is possible to compute the Pearson correlation between $\mathbf{x}_{n+1} = (\mathbf{x}_n, x_{n+1})$ and $\mathbf{y}_{n+1} = (\mathbf{y}_n, y_{n+1})$ as a function of $\rho(\mathbf{x}_n, \mathbf{y}_n)$ and x_n and y_n only. The numerical implementation will input existing correlation value $\rho(\mathbf{x}_n, \mathbf{y}_n)$ and new data x_{n+1} and y_{n+1}, then outputs $\rho(\mathbf{x}_{n+1}, \mathbf{y}_{n+1})$. One possible iterative solution is[1]

$$n\mathbb{V}(\mathbf{x}_{n+1}, \mathbf{y}_{n+1}) = (n-1)\mathbb{V}(\mathbf{x}_n, \mathbf{x}_n) + \frac{n}{n+1}(\mathbf{x}_{n+1} - \mu_n)(\mathbf{y}_{n+1} - \mu_n),$$

where μ_n is the online algorithm for the sample mean and $\mathbb{V}$ is the sample covariance. The normalization of the sample covariance is needed to obtain the online version of correlation. It is possible to have slightly different variations to the above formulation.

We can further construct an online algorithm for computing correlation matrix. Suppose n subjects are given. Each subject has p measurements. The i-th subject data vector can be denoted as

$$\mathbf{x}_i = (x_{i1}, x_{i2}, \cdots, x_{ip})^\top.$$

The $p \times p$ correlation matrix $C_n = (c^n_{ij})$ across subjects is given by $c^n_{ij} = \rho(\mathbf{x}_i, \mathbf{x}_j)$, the Pearson correlation between $\mathbf{x}_i$ and $\mathbf{x}_j$. Now we add the $(n+1)$-th subject with p measurements, denoted as $\mathbf{x}_{n+1}$. Then the correlation matrix C_{n+1} of all $(n+1)$ subjects can be computed using only C_n and $\mathbf{x}_{n+1}$. The resulting $p \times p$ correlation matrix C_{n+1} is a function of C_n and $\mathbf{x}_{n+1}$ only and the online algorithm is based on the iterative formula. One possible solution

[1] The formulation was derived by Tiankang Xie of University of Wisconsin-Madison and Zewei Lin of Renmin University.

can be derived by making the matrix version of the iterative algorithm for correlations.