Notational Notes

Bolding vectors and matrices. This note on bolding is to help you read my work, not a critique on your work. Statisticians have an obsession with bolding matrices and vectors and reserving non-bold variables for scalars when typing up work. This is purely stylistic and completely impractical both for handwritten work and for complex methodological investigations like those a mathematical probabilist (like me) does. You are certainly welcome to use bolding in this statistical style - encouraged, even! - but you should be prepared for some disciplines not to use this style and for me not to. I will tell you if something is a vector or matrix, not indicate it with bold.

Matrix transpose. Suppose A is a matrix. Traditionally, both A^T and A' denote the transpose of the matrix A, but you are not supposed to use them interchangeably within a single document. Two notations exist because a) they were developed in different contexts and b) there are situations where you need options to avoid ambiguity, e.g. A' is the derivative of operator A and so you use A^T as the transpose of A throughout a document, or maybe A^T is the exponential of matrix A by matrix T and so you use A' as the transpose of A throughout a document.

Either way, you've got to pick one and stick with it!! In the next version of your report, please only have A' or only have A^T notation. I prefer A^T because I've had to work more with operator derivatives than with matrix exponentials, so that's what I'll use in my notes here.

The vector e_i. Traditionally, the vector e_i is the standard unit vector whose jth element is δ^i_j, that is 1 in the ith place and 0 elsewhere. In discussion of orthogonal bases when ambiguity is not possible, the interpretation of e_i may be generalized to mean the ith basis vector, which is appropriate when talking about columns of a square orthogonal matrix. However, when there is ambiguity, you should reserve e_i for the standard unit vector and denote your orthogonal basis vectors / orthogonal eigenvectors by another letter, such as v_i or u_i.

For example, we know that if E is a square orthogonal matrix, then $E^TE = I$. The ith column vector of I is e_i, and if we denote the ith column vector of E by v_i, then $E^Tv_i = e_i$. Notice how we're talking both about the ith eigenvector and about the ith standard unit vector, so we reserve use of e_i for the standard unit vector. Note that what I've said here is in contradiction to your assertion that $Ee_i = e_i$ when $\lambda_i \neq 0$ in the methods. In fact whether or not $\lambda_i = 0$, $Ee_i = v_i$ as v_i is the ith column of E and $E^Tv_i = e_i$ as E is orthogonal.

Methods

Do you want to prove M is idempotent?

Do you want to prove the 1 vector is an eigenvector of MCM with eigenvalue 0?

Matrix form of Moran's coefficient. Let M be the centering matrix and y be a vector with mean value $\bar{y}$. Denote by $\vec{1}$ the vector of 1's. As M is the centering matrix, $[My]_j = y_j - \bar{y}$. Thus, knowing M is idempotent ($M^2 = M$) and symmetric ($M^T = M$), we can compute

$$\vec{1}^T I \vec{1} = \sum_{i=1}^{n}\sum_{j=1}^{n}[\vec{1}]_i[I]_{ij}[\vec{1}]_j = \sum_{i=1}^{n}\sum_{j=1}^{n}1\delta_i^j 1 = n$$

$$\vec{1}^T C \vec{1} = \sum_{i=1}^{n}\sum_{j=1}^{n}[\vec{1}]_i[C]_{ij}[\vec{1}]_j = \sum_{i=1}^{n}\sum_{j=1}^{n}1c_{ij}1 = \sum_{i=1}^{n}\sum_{j=1}^{n}c_{ij}$$

$$\begin{aligned} y^T My &= y^T M^T IMy = (My)^T I(My) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}[My]_i[I]_{ij}[My]_j = \sum_{i=1}^{n}\sum_{j=1}^{n}(y_i-\bar{y})\delta_i^j(y_j-\bar{y}) = \sum_{i=1}^{n}(y_i-\bar{y})^2 \end{aligned}$$

$$\begin{aligned} y^T MCMy &= (My)^T C(My) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}[My]_i[C]_{ij}[My]_j = \sum_{i=1}^{n}\sum_{j=1}^{n}(y_i-\bar{y})c_{ij}(y_j-\bar{y}) \end{aligned}$$

$$\begin{aligned} \mathcal{MC}(y) &= \frac{n}{\sum_{i=1}^{n}\sum_{j=1}^{n}c_{ij}} \frac{\sum_{i=1}^{n}(y_i-\bar{y})\sum_{j=1}^{n}c_{ij}(y_j-\bar{y})}{\sum_{i=1}^{n}(y_i-\bar{y})^2} \\ &= \frac{\vec{1}^T I\vec{1}}{\vec{1}^T C\vec{1}} \frac{(My)^T C(My)}{(My)^T I(My)} \\ &= \frac{n}{\vec{1}^T C\vec{1}} \frac{y^T MCMy}{y^T My} \end{aligned}$$

There are a few things I would like you to do with this information:

1. Go through the derivation on your own without referring to this write-up to make sure you're comfortable going back and forth between matrix products and sums of scalars. You should be able to simplify what I've done as computations / steps involving I were not strictly needed.

2. Study the new representation of the Moran's coefficient a little more closely: $\mathcal{MC}(y) = \frac{\vec{1}^T I\vec{1}}{\vec{1}^T C\vec{1}} \frac{(My)^T C(My)}{(My)^T I(My)}$. This is the representation that makes more clear why this is an autocorrelation. The term $\frac{(My)^T I(My)}{\vec{1}^T I\vec{1}}$ is the covariance of y with itself along an independence structure I (i.e. the usual variance). The term $\frac{(My)^T C(My)}{\vec{1}^T C\vec{1}}$ is the covariance of y with itself along spatial structure C (i.e. a spatial variance). $\mathcal{MC}$ thus normalizes a spatial variance by the usual variance to get a spatial autocorrelation. You could easily get covariances and correlations in place of variances and autocorrelations by using two images / vectors x and y rather than using one image / vector y twice.

3. $\mathcal{MC}$ is a type of correlation, and so should be 0 when there is no correlation. My guess is that the assertion it is $\frac{-1}{n-1}$ comes from asserting a distribution on y with no true

spatial correlation and computing an expected value of observed spatial correlation and finding that we expect to observe negative spatial correlation. Saying $\mathcal{MC}$ has expected value $\frac{-1}{n-1}$ when there is no correlation in the assumed model is distinct from saying $\mathcal{MC}$ takes value 0 when there is no correlation in the input. Make sure to be precise here.

Eigenvalue Decomposition and Key Properties

Suppose M and C are square symmetric matrices. Then MCM has an eigendecomposition

$$MCM = E\Lambda E^T$$

into diagonal matrix Λ of eigenvalues and orthogonal matrix E of eigenvectors of MCM. Denote by λ_i and v_i the ith eigenvalue and eigenvector respectively.

Claim: If M is idempotent ($M^2 = M$) and $\lambda_i \neq 0$, then $Mv_i = v_i$.

Proof:

$$\begin{aligned} MCMv_i &= \lambda_i v_i \\ M^2CMv_i &= \lambda_i Mv_i \\ MCMv_i &= \lambda_i Mv_i \\ \lambda_i v_i &= \lambda_i Mv_i \\ v_i &= Mv_i \qquad\qquad \text{for } \lambda_i \neq 0 \end{aligned}$$

Note: The proof you gave can be fixed but is a little more finicky as you're handling all eigenvectors at once, including those with 0-eigenvalue:

$$\begin{aligned} MCME &= E\Lambda E^T E = E\Lambda \\ M^2CME &= ME\Lambda \\ MCME &= ME\Lambda \\ E\Lambda &= ME\Lambda \\ \forall i: \quad \lambda_i v_i &= E\Lambda e_i = ME\Lambda e_i = \lambda_i Mv_i \\ \forall i \text{ such that } \lambda_i \neq 0: \quad v_i &= Mv_i \end{aligned}$$

Whitening Transformations

This section is written with respect to a linear model predicting images from covariates. Your simulation work currently predicts labels from images, which is a very different context. I want you to understand whitening transformations in both contexts, and it's easier to start with linear models predicting images from covariates. When you feel like you have a solid understanding of this section, I want you to adapt this knowledge to your simulation context. In particular, when predictors have strong and complex correlation structure, it can impede model performance. For example, a) coefficient estimates may not be stable because of multicollinearity, impeding generalizability, or b) if the relevant signal is the signal inducing strong correlation among covariates, it may be that signal is too diffuse to identify with good power, increasing data needs for good performance.

Standard Linear Models

Suppose you are fitting a standard linear model and your desired variable v for the outcome has substantial right skew. You might choose to use $\log v$ as the outcome instead. This changes some of your assumptions when fitting the model:

No logarithm	Logarithm
v is linearly related to covariates	$\log v$ is linearly related to covariates
v produces normally distributed residuals	$\log v$ produces normally distributed residuals
v results in homoscedasticity	$\log v$ results in homoscedasticity

Generally speaking, none of these assumptions are true. True relationships are almost never perfectly linear and error distributions are almost never perfectly normal and homoscedastic. When we fit a linear model, we want these assumptions to be close enough to true that we can reasonably use the model. When we apply a logarithm to a right skew variable before using it as an outcome in a linear model, we do so because we think the assumptions in the right column are more true than those in the left column.

Linear models of images

Suppose you have a sample of images all registered to the same template and you want to fit a linear model predicting the images y with s pixels from p covariates x (say age and disease status). The model you would likely use, described at the individual pixel level and also at the individual level, is

$$y_{ij} = \sum_{k=1}^{p} x_{ik}\beta_{kj} + \epsilon_{ij}; \qquad y_{i\cdot} = x_{i\cdot}\beta + \epsilon_{i\cdot}$$

where i indexes the image, j indexes the pixel, and k indexes the covariate, and where error vectors $\epsilon_{i\cdot} = [\epsilon_{ij}]$ are independent with distribution $\mathcal{N}(0, \Sigma)$. Here, image, covariate, and error vectors $y_{i\cdot}$, $x_{i\cdot}$, $\epsilon_{i\cdot}$ are horizontal vectors. The parameters of this distribution are the $p \times s$ matrix β and $s \times s$ matrix Σ, and: *you've assumed a particular type of linear relationship between images and covariates, you've assumed independence of images, you've assumed joint normality of the errors, and you've assumed a particular shared covariance structure Σ of the errors.*

While independence of images may be true depending on the context you work in, these remaining assumptions are not true in the same way as the assumptions of a standard linear model are not true. We assume them anyway if they're close enough to true to be useful. It's really important to keep this in mind, because this is the crux of your misunderstanding. When we fit a model, we are making incorrect assumptions because they are close enough to true to be useful.

Now, in order to assess whether our assumptions are close enough to true to be useful, we need to know what useful means. In the context of image analysis, useful usually involves answering yes to the following questions: 1) can we fit the model in a computationally efficient and stable way, 2) do we have enough data to estimate all of the parameters, and 3) have we adequately captured the structure of the data?

Assumption of linear relationship: Linear relationships have computationally efficient and stable estimators, require little data to estimate, and are often adequate for capturing data structure in preliminary work. You can always add a quadratic term later, for example. Assuming a linear relationship is generally useful.

Assumption of independence of images: Depends on context. If the images are from different people with no relation, it's probably adequately capturing data structure. It is certainly an assumption supporting efficiency, stability, and low data needs in fitting.

Assumption of joint normality of errors: This assumption supports computational efficiency, stability, and low data needs in fitting, but may not adequately capture the structure of the data. For example, chest CT scans are going to have a distribution reflecting the quantities and densities of air, water/tissue, and bone in the image, which certainly doesn't result in a normal distribution. When we violate this normality assumption, we increase our data needs (law of large numbers allows us to fudge the normality assumption). This assumption is nice to address, but not vital if sample sizes are moderate to large.

Assumption of a particular shared covariance structure Σ *of the errors:* This is the assumption that messes up all of our notions of useful. If Σ is fully general, we have horrible computational efficiency and stability and very large data needs to compensate, but we can capture the data structure flexibly. If Σ is diagonal, we have great computational efficiency and stability, but we've completely ignored the spatial structure of the data and lost a tremendous amount of power, thus substantially increasing our data needs to compensate. If Σ is structured, such as an exponential correlation structure, we may have even worse computational efficiency than for a fully general Σ, but more moderate stability, data needs, and structural adequacy of the model.

The assumption of a shared covariance structure Σ *of the errors is perhaps the most impactful assumption on the usefulness of the image model.* Pick something too complex and you can't fit the model. Pick something too simple and you can't learn anything from the model because you've lost power.

So, what do we do? Computational efficiency is so important in this context that an inefficient model may be impossible to fit because image data are so large and complex. Thus, a solution should reduce either the size of the data (dimension reduction) so that efficiency is less pivotal or reduce the complexity of the data so that we can both efficiently and adequately capture structure.

Dimension reduction is a popular and important approach, but it's not our focus here. Instead, we're interested in reducing the complexity of the data. A **whitening transfor-**

mation is a transformation we can apply to our multi-dimension outcome to substantially reduce correlation (reduce complexity) allowing for diagonal Σ to adequately capture structure. This is akin to a logarithm being a transformation we can apply to an outcome to reduce skew and better support the normality assumption.

Let's call this whitening transformation $\mathcal{F}$. If $\mathcal{F}$ is a linear transformation, then it is a matrix we can apply to vectorized images. In the context of our linear model, a whitening transformation $\mathcal{F}$ produces a new linear system for which our earlier assumptions are true enough to be useful:

$$\tilde{y}_{i\cdot} = x_{i\cdot}\tilde{\beta} + \tilde{\epsilon}_{i\cdot}$$

where $\tilde{y}_{i\cdot} = y_{i\cdot}\mathcal{F},\ \tilde{\beta} = \beta\mathcal{F},\ \tilde{\epsilon}_{i\cdot} = \epsilon_{i\cdot}\mathcal{F}$, and $\tilde{\epsilon}_{i\cdot}$ are independent with distribution $\mathcal{N}(0, \tilde{\Sigma})$ for diagonal covariance $\tilde{\Sigma} = \mathcal{F}^T\Sigma\mathcal{F}$.

By definition, the whitening transformation ensures that diagonal $\tilde{\Sigma}$ adequately captures structure. Because it is diagonal, we know we have computational efficiency, stability, and low data needs. As a bonus, since transformed outcomes $\tilde{y}$ are obtained as weighted averages of outcomes y, the normality assumption will also be better satisfied.

So, let's pause and summarize for a moment: In order to have a useful linear model of images, we need to pick our covariance matrix really carefully to ensure computational efficiency, stability, power, and adequate capturing of spatial structure. More general covariances are needed to adequately capture spatial structure of raw images, while diagonal covariances are best for computational efficiency, stability, and power. A whitening transformation is a transformation of the image data such that a diagonal covariance adequately captures structure of transformed data, allowing us to achieve all aspects of a useful linear model. Our starting question was how to pick a good linear model. Our revised question is how to pick a whitening transformation.

Picking a whitening transformation

Everything we just discussed is nice and all if we already have a whitening transformation $\mathcal{F}$ on hand. However, generally, we don't! We need to make one up and check that it really does 'whiten' the data, meaning make the transformed image data so little correlated that we can treat them as independent and use a diagonal covariance.

So, how might we pick one? For one thing, linear transformations are nice. For another, orthogonal transformations are nice because they're 'rigid' linear transformations (no rescaling; rotations and reflections) and very easy to invert ($\mathcal{F}^{-1} = \mathcal{F}^T$). For this reason, classically, whitening transformations are constructed via an eigendecomposition: a real symmetric matrix is eigendecomposed into a diagonal matrix of eigenvalues and an orthogonal matrix of eigenvectors, the latter of which we might use as a transformation. When we choose to use eigendecomposition, we refer to transformed image data as frequency data.

In graph signal processing (GSP), which is a generalization of Fourier analysis, the real symmetric matrix we eigendecompose is an adjacency matrix or associated Laplacian matrix C where the adjacency matrix encodes the structure of the image as a graph / network with nodes (pixels) and edges (quantifying closeness of pixels). In GSP, the orthogonal matrix from the eigendecomposition is used for the transformation and while the eigenvalues may be used for filtering, they don't really have meaning. In eigenvector spatial filtering

(ESF), we use the centered adjacency matrix (MCM) because this gives us an additional tool for understanding our transformation: Moran's coefficient. Moran's coefficient makes the eigenvalues meaningful as measures of spatial autocorrelation. We really like ESF for having Moran's coefficient because we feel it makes the frequency space more interpretable.

The previous question was how to pick a whitening transformation. The revised question is now how to pick an adjacency matrix / graph / network C.

Picking a network C

You've already seen that there are a lot of ways to pick networks for the LIIA baseline project. You've considered Pearson's, Spearman's, and distance correlation networks, soft-thresholding transformations of these to produce scale-free topology networks, and sparse GLASSO networks. These tend to be data-driven networks, though, and we want to be able to pick data-independent networks so that we don't double dip with our data: once to get $\mathcal{F}$, once to fit the linear model. There are three really easy ways to pick a network for images that we'll consider, two of which are data independent. In all three cases, every pixel is a node in the network. *In each case, the diagonal elements of adjacency C should be 0!*

1. (Unweighted; data independent) Two pixels share an edge if they are within a distance d of eachother in an image, e.g. within 3 pixels. The network would then be called a 3-neighbor graph or radius 3 proximity graph.

2. (Weighted; data independent) Every two pixels share an edge. The weight of the edge is a transformation of the distance d_{ij} between the pixels, e.g., $C_{ij} = \exp(-d_{ij})$. This often produces a correlation matrix C for non-diagonal elements, e.g., exponential.

3. (Weighted; data dependent) - This requires you have an independent but similar dataset to the one you want to analyze to avoid double dipping - Every two pixels share an edge. The weight of the edge is the correlation between those two pixels in the independent dataset. Alternatively, use another method to estimate a network C from this dataset (e.g., GLASSO).

We often pick option 2. and choose a transformation such that C is a standard correlation matrix (e.g. exponential). This is a fairly arbitrary choice though. We will be exploring all of these options over time. For now, when you revise the report, I want you to switch to using a 2-neighbor adjacency lattice (option 1.; see my old digit report for implementation), since using option 2. has been leading to confusion. When we start working with ADNI, we will use ADNI for option 3 so we can analyze Bri's LIIA data.

Checking model assumptions

Great, we've picked a whitening transformation in the hopes of identifying a linear model for which assumptions are true enough to make the model useful. That's a hope that needs to be checked!

To be a whitening transformation, we need the transformed image data (the frequency data) to be approximately uncorrelated. If your images are small, I would just compute

an empirical correlation matrix for the transformed image data and plot a histogram of the entries. This should produce a fairly tight histogram centered about 0. If not, the whitening transformation is no good.

While we're at it, we might want to check the normality assumption. For chest CT, I noticed that raw image data were fairly skewed because of the anatomy discussion from earlier, so I computed the skewness of each frequency and made another histogram. I wanted to see another tight histogram centered at 0 (no skew). If it was substantially off center or had a wide spread, the normality assumption might be failing. In that case, I might need to have a larger sample size to counteract the normality violation. Alternatively, I might investigate other whitening transformations with better normalizing effects. E.g. a sparse transformation may result in less normalizing than a transformation which is not sparse.