Spectral Statistics of Random Matrices

A Thesis
Presented to
The Division of Mathematics and Natural Sciences
Reed College

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Arts

Ali Taqi

May 2021

Approved for the Division
(Mathematics)

Jonathan M. Wells

Table of Contents

List of Tables

Table of Random Matrix Distributions

Distribution	Notation ($\mathcal{D}$)	Parameters	Class
Normal	$\mathcal{N}(\mu, \sigma)$	$\mu \in \mathbb{R}, \sigma \in \mathbb{R}^+$	Explicit
Uniform	$\text{Unif}(a, b)$	$a, b \in \mathbb{R}$	Explicit
Hermite-β	$\mathcal{H}(\beta)$	$\beta \in \mathbb{N}$	Implicit
Erdos-p	$\text{ER}(p)$	$p \in [0, 1]$	Implicit

Table of Spectrum Schema

Scheme	Matrix	Notation	Ordering
Sign-Ordered	P	$\sigma_S(P)$	$\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_N$
Norm-Ordered	P	$\sigma_N(P)$	$\|\lambda_1\| \geq \|\lambda_2\| \geq ... \geq \|\lambda_N\|$
Singular	$P \cdot P^T$	$\sigma_+(P)$	$\sqrt{\lambda_1} \geq \sqrt{\lambda_2} \geq ... \geq \sqrt{\lambda_N}$

Table of Dispersion Metrics

Metric*	Notation	Formula	Symmetric	Parameters
Standard Norm	δ	$\|z' - z\|$	True	-
β-Norm	δ_β	$\|z' - z\|^\beta$	True	$\beta \in \mathbb{N}$
Difference of Absolutes	δ_{abs}	$\|z'\| - \|z\|$	False	-
Identity Difference	δ_{id}	$z' - z$	False	-

Table of Pairing Schema

Scheme	Notation	Formula
Lower	$\Pi_<$	$\{(i, j) \mid i < j \text{ for } i, j \in \mathbb{N}_N\}$
Upper	$\Pi_>$	$\{(i, j) \mid i > j \text{ for } i, j \in \mathbb{N}_N\}$
Consecutive	Π_C	$\{(i, j) \mid i = j + 1 \text{ for } i, j \in \mathbb{N}_N\}$
All	Π_0	$\{(i, j) \mid i, j \in \mathbb{N}_N\}$

Abstract

On their own, random variables exude deterministic properties regarding their uncertainty. The same generalization can be made for random matrices, which are matrices whose entries are random variables. One particular statistic worth investigating is the distribution of a matrix ensemble's eigenvalues, or its spectrum. In this thesis, there will be an exploration of various classes of random matrices and relevant spectral statistics like their spectra and mixing times.

Dedication

For my mother.

Introduction

So, what are *spectral statistics*? Do they have to do with rainbows? Sceptres? No, they don't, but they're almost as colorful and regal. The word spectral is borrowed from the spectral-like patterns observed in statistical physics - whether it may be atomic spectra or other quantum mechanical phenomena. The borrowing is loose and not literal, but still somewhat well founded.

The field of Random Matrix Theory was extensively developed in the 1930s by the nuclear physicist Eugene Wigner. He found connections between the deterministic properties of atomic nuclei and their random and stochastic behaviors. The link? Random matrices.

So in the context of this thesis, *Spectral statistics* will be an umbrella term for random matrix statistics that somehow involve that matrix's eigenvalues and eigenvectors.

To explore these spectral statistics, this thesis will use the **RMAT** package. This package was developed alongside this thesis in order to facilitate the simulation of these random matrices and spectral statistics. As such, there is a large simulation component to this thesis. To showcase the methodology of the simulations, code snippets will be sprinkled about the thesis. It will use code derived from the package RMAT which can be found on GitHub; minimal source code will also be available in the appendix, as well as pre-simulated data available on the Reed database. All code examples in this thesis are reproducible. Simply use the RMAT package (or equivalent code) and use *set.seed*(23).

Chapter 1

Random Matrices

As discussed in the introduction, this thesis will be an exploration of spectral statistics of random matrices. This means that we must first be able to understand what random matrices are. At a fundamental level, random matrices are simply matrices whose entries are randomly distributed in accordance to some distribution or method. To formalize all these notions, we will define what random matrices are and what it means for them to be $\mathcal{D}$-distributed.

Prior to beginning the discussion on $\mathcal{D}$-distributions, the reader should be familiar or at least accquainted with the notion of random variables and what they are. A summary is available in the appendix in A.x.

When it comes to random simulation, there must always be a rule to which our randomness must conform, regardless of complexity. For example, sampling a vector from a distribution is a rudimentary example of this. For random matrices, there will be a few methods of generating their entries that are not just sampling from theoretical distributions. As such, we motivate the $\mathcal{D}$-distribution.

1.1 $\mathcal{D}$-Distributions

Now, we motivate the D-distribution. A formalization on how to initialize a random matrix.

Definition 1.1.1 ($\mathcal{D}$-distribution)**.** *When we define a random matrix that is $\mathcal{D}$-distributed, we say that $P \sim \mathcal{D}$. In the simplest of terms, $\mathcal{D}$ is essentially the algorithm that generates the random matrix P. We define two primary methods of distribution: explicit distribution and implicit distribution. If $\mathcal{D}$ is an explicit distribution, then every entry of P is homogenously sampled from that distribution. Otherwise, if it is implicit, we utilize an algorithm that imposes an implicit distribution on the entries.*

1.1.1 Explicit Distributions

The simplest case is homogenous, explicitly distributed random matrices. If $\mathcal{D}$ is an explicit distribution, then we overload the notation $\mathcal{D}$ to mean a probability distribution in the classical sense (see Appendix A.x). So, if $\mathcal{D}$ is a probability distribution, the matrix $P \sim \mathcal{D}$ when $p_{ij} \sim \mathcal{D}$. In otherwords, we simply perform entry-wise sampling from that distribution.

Take for example the following explicit distributions which we cover in this thesis.

1. If $\mathcal{D} = \mathcal{N}(0,1)$, then $p_{ij} \sim \mathcal{N}(0,1)$.
2. If $\mathcal{D} = \text{Unif}(0,1)$, then $p_{ij} \sim \text{Unif}(0,1)$.

Remark (Formalization). *Explicit distributions can be formalized in scope of the standard notation in probability theory. At the heart of the formalization is the usage of index hacking to collapse the array's indices from two dimensions to one. This way, our random matrix has a representation as a random vector, which we commonly encounter in probability theory as a (i.i.d) sequence of random variables! Generally, an* $N \times N$ *random matrix that is explicity and homogenously* $\mathcal{D}$*-distributed is essentially a sequence of* N^2 *i.i.d random variables sampled from* $\mathcal{D}$*.*

Example (Formalization). *For example, suppose* P *is a* 2×2 *random matrix with* $\mathcal{D} = \mathcal{N}(0,1)$*. Then, we have four random variables to initialize. In the random matrix representation, we need to initialize* $P_{11}, P_{12}, P_{21},$ *and* P_{22} *by sampling them from* $\mathcal{D}$*. In the vector representation, we just say we are sampling four i.i.d random variables from* $\mathcal{D}$*. The matrix indexing is intrinsic and preservable by using index hacking with some modular math.*

1.1.2 Implicit Distributions

In the latter case, we are concerned less about the distribution of the matrix entries and moreso about its holisitic properties.

Consider for example, the following implicit distributions.

1. If $\mathcal{D}$ = Stochastic, then the matrix is a row of random stochastic rows. (See Algorithm B.x)
2. If $\mathcal{D}$ is any distribution (implicit or explicit), then $\mathcal{D}^\dagger$ is the Symmetric/Hermitian version of $\mathcal{D}$. (See Algorithm B.x)

1.1.3 Random Matrices

Definition 1.1.2 (Random Matrix). *Assuming* $\mathcal{D}$ *is an explicit distribution, a random matrix is any matrix over the field* $\mathbb{F}$ *is a matrix* $M \in \mathbb{F}^{N \times N}$ *is a matrix whose entries are i.i.d random variables. So, if a random matrix* $M = (m_{ij})$ *is* $\mathcal{D}$*-distributed, then we say* $m_{ij} \sim \mathcal{D}$*. In the scope of this thesis, assume every random matrix to be homogenously distributed. Otherwise, if* $\mathcal{D}$ *is an implicit distribution, then* P *is a matrix whose entries are determined by the algorithm imposed by* $\mathcal{D}$*.*

Sometimes, we want our matrix to have complex entries. We notate this by specifying $\mathbb{F} = \mathbb{C}$.

Remark (Complex Entries)**.** *To say that a random matrix is explicitly $\mathcal{D}$-distributed over $\mathbb{C}$ would mean that its entries take the form $a + bi$ where $a, b \sim \mathcal{D}$ are random variables. In other words, if we allow the matrix to have complex entries by setting $\mathbb{F} = \mathbb{C}$, then we must sample the real and imaginary component as $\mathcal{D}$-distributed i.i.d. random variables.*

Below, we can see code on how to generate a standard normal random matrix using the **RMAT** package.

Code Example (Standard Normal Matrix)**.** *Let $\mathcal{D} = \mathcal{N}(0, 1)$. We can generate $P \sim \mathcal{D}$, a 4×4 standard normal matrix, as such:*

```
library(RMAT)
P <- RM_norm(N = 4, mean = 0, sd = 1)
# Outputs the following
P
           [,1]       [,2]       [,3]        [,4]
[1,]  0.1058257 -1.0835598 -0.7031727  1.01608625
[2,] -0.2170453  1.8206070 -0.4539230  0.06828296
[3,]  1.3002145  0.1254992 -0.5214005 -0.61516174
[4,] -1.0398587  0.1975445 -0.8511950  0.86366082
```

Summary Table of $\mathcal{D}$-Distributions

Table of Random Matrix Distributions			
Distribution	Notation ($\mathcal{D}$)	Parameters	Class
Normal	$\mathcal{N}(\mu, \sigma)$	$\mu \in \mathbb{R}, \sigma \in \mathbb{R}^+$	Explicit
Uniform	Unif(a, b)	$a, b \in \mathbb{R}$	Explicit
Hermite-β	$\mathcal{H}(\beta)$	$\beta \in \mathbb{N}$	Implicit
Erdos-p	ER(p)	$p \in [0, 1]$	Implicit

1.2 The Crew: Ensembles

With a random matrix well defined, we may now motivate one of the most important ideas - the random matrix ensemble. One common theme in this thesis will be that random matrices on their own provide little information. When we consider them at the ensemble level, we start to obtain more fruitful results. Without further ado, we motivate the random matrix ensemble.

Definition 1.2.1 (Random Matrix Ensemble). *A $\mathcal{D}$-distributed random matrix ensemble $\mathcal{E}$ over $\mathbb{F}^{N\times N}$ of size K is defined as a set of $\mathcal{D}$-distributed random matrices $\mathcal{E} = \{P_i \sim \mathcal{D} \mid P_i \in \mathbb{F}^{N\times N}\}_{i=1}^{K}$. In simple words, it is simply a collection of K iterations of a specified class of random matrix.*

So, for example, we could compute a simple ensemble of matrices as follows.

Code Example (Standard Normal Hermitian Ensemble). *Let $\mathcal{D} = \mathcal{N}(0,1)^{\dagger}$. We can generate $\mathcal{E} \sim \mathcal{D}$ over $\mathbb{C}$, an ensemble of 4×4 complex Hermitian standard normal matrices of size 10 as such:*

```
library(RMAT)
# By default, mean = 0 and sd = 1.
ensemble <- RME_norm(N = 4, cplx = TRUE, herm = TRUE, size = 10)
# Outputs the following
ensemble
...
[[10]]
                    [,1]                 [,2]                 [,3]
[1,] -0.59931+1.24286i  1.29457+0.66058i  0.83539-0.16662i
[2,]  1.29457-0.66058i  0.78841+0.09818i -1.16592+1.14666i
[3,]  0.83539+0.16662i -1.16592-1.14666i -0.51256+0.17750i
```

With this in mind, we will gloss over, characterize, and briefly discuss a few special recurring ensembles in this thesis.

1.2.1 Hermite β-Ensembles

The Hermite β-Ensembles will be one of the primary ensembles discussed in this thesis. This ensemble will be characterized, motivated, and defined more thoroughly in **Chapter 4**. However, we will give a brief introduction to the ensemble.

At a practical level, all one would need to know is that the matrices are generated in accordance to an algorithm found in the appendix (Algorithm B.x) and in Chapter 4.

1.2.2 Erdos-Renyi p-Ensembles

The Hermite β-ensembles are a normal-like class of random matrices. Now, we will veer away from the normal distribution as a whole and switch to a different class of matrices: stochastic matrices. Stochastic matrices, in short, are matrices that represent Markov Chains (see A.x). We can also think of them as the matrix representation of a specific setup of a walk on a random graph.

A particular class of random graphs that we will consider are the Erdos-Renyi random graphs. Essentially, these are graphs whose vertices are connected with a uniform probability p. We can interpret this as saying an Erdos-Renyi graph is a simple random walk on a graph with parameterized sparsity (given by p). Without further ado, we motivate the Erdos-Renyi graph:

Definition 1.2.2 (Erdos-Renyi Graph)**.** *An Erdos-Renyi graph is a graph $G = (V, E)$ with a set of vertices $V = \{1, \dots, N\}$ and edges $E = \mathbb{1}_{i,j \in V} \sim Bern(p_{ij})$. It is homogenous if $p_{ij} = p$ is fixed for all i, j.*

Essentially, an Erdos-Renyi graph is a graph whose 'connectedness' is parameterized by a probability p (assuming it's homogenous, which this document will unless otherwise noted). As $p \to 0$, we say that graph becomes more sparse; analogously, as $p \to 1$ the graph becomes more connected.

Recall from probability theory that a sum of i.i.d Bernoulli random variables is a Binomial variable. As such, we may alternatively say that the degree of each vertex v is distributed as $deg(v) \sim Bin(N, p)$ where N is the number of vertices. This makes simulating the graphs much easier.

Code Example (Erdos-Renyi p = 0.5 Ensemble)**.** *Let $\mathcal{D} = ER(p = 0.5)$. We can generate $\mathcal{E} \sim \mathcal{D}$, an ensemble of 4×4 Erdos-Renyi matrices ($p = 0.5$) of size 10 as such:*

```
library(RMAT)
ensemble <- RME_erdos(N = 4, p = 0.5, size = 10)
# Outputs the following
ensemble
...
[[10]]
          [,1]      [,2]      [,3]     [,4]
[1,] 0.0000000 0.1729581 0.8270419 0.000000
[2,] 0.0000000 0.0000000 1.0000000 0.000000
[3,] 0.2557890 0.3766740 0.0000000 0.367537
[4,] 0.2151029 0.3929580 0.3919391 0.000000
```

Chapter 2

Spectra

2.1 Introduction

So, what are *spectral statistics*? Do they have to do with rainbows? Sceptres? No, they don't, but they're almost as colorful and regal. The word spectral is borrowed from the spectral-like patterns observed in statistical physics - whether it may be atomic spectra or other quantum mechanical phenomena. The borrowing is loose and not literal, but still somewhat well founded. In fact, the field of Random Matrix Theory was extensively developed in the 1930s by the nuclear physicist Eugene Wigner. He found connections between the deterministic properties of atomic nuclei and their random and stochastic behaviors. The link? Random matrices.

So in the context of this thesis, *spectral statistics* will be an umbrella term for random matrix statistics that somehow involve that matrix's eigenvalues and eigenvectors. That being said, if we fix a *random matrix*, we can study its features by studying its eigenvalues - fundemental numbers that tell us a lot about the matrix. They are quite important for many reasons. For instance in statistical physics, many processes are represented by operators or matrices, and as such, their behaviours could be partially determined by the eigenvalues of their corrosponding matrices. The study of eigenvalues and eigenvectors primarily falls in the scope of Linear Algebra, but their utility is far-reaching. So, what exactly are *eigenvalues* exactly?

2.1.1 The Quintessential Spectral Statistic: the Eigenvalue

Given any standard square matrix $P \in \mathbb{F}^{N\times N}$, its *eigenvalues* are simply the roots of the characteristic polynomial $\mathrm{char}_P(\lambda) = \det(P - \lambda I)$. By the Fundamental Theorem of Algebra, we know that there is always have as many complex eigenvalues $\lambda \in \mathbb{C}$ as the dimension of the matrix.

That being said, when our random matrix has a specified distribution (say, standard normal), we can see patterns in the eigenvalue distributions. So, an eigenvalue is a **spectral statistic** of a random matrix! To talk about a matrix's eigenvalues in a more formal and concise manner, we motivate what is the eigenvalue spectrum.

Definition 2.1.1 (Spectrum)**.** *Suppose $P \in \mathbb{F}^{N \times N}$ is a square matrix of size N over $\mathbb{F}$. Then, the (eigenvalue) spectrum of P is defined as the multiset of its eigenvalues and it is denoted $\sigma(P) = \{\lambda_i \in \mathbb{C}\}_{i=1}^{N}$. Note that it is important to specify that a spectrum is a multiset and not just a set; eigenvalues could be repeated due to algebraic multiplicity and we opt to always have N eigenvalues.*

For example, consider the following code example from the RMAT package.

Code Example (Spectrum of a Standard Normal Matrix)**.** *Let $P \sim \mathcal{N}(0,1)$ be a 4×4 standard normal random matrix. We can generate the spectrum of P, $\sigma(P)$ as follows:*

```
library(RMAT)
P <- RM_norm(N = 5, mean = 0, sd = 1)
spectrum_P <- spectrum(P)
# Outputs the following
spectrum_P
...
        Re       Im   Norm Order
 1 -0.5434  1.3539 1.4589     1
 2 -0.5434 -1.3539 1.4589     2
 3  0.2255  1.4250 1.4427     3
 4  0.2255 -1.4250 1.4427     4
 5 -0.8678  0.0000 0.8678     5
```

While our definition for spectrum is nice and clean, there are a few caveats that we must take care of. First, how are spectra statistics? So we can even call them statistics, so there needs to be some formalization as to why we consider eigenvalues statistics. In this dialogue, we will call back on the same sort of formalization dialogue in the Random Matrices section. Before beginning, the reader is encouraged to review what a statistic is in the review appendix (A.x).

Recall that when we defined and motivated the $\mathcal{D}$-distribution framework of simulating random matrices,we always had one thing - a vector representation of random variables. Using this framework, the formalization is trivial.

Remark (Formalization)**.** *Suppose $P \sim \mathcal{D}$ is an $N \times N$ random matrix. Then, P has a representation as a sequence of N^2 random variables, denote it $\vec{X} = \{X_i \mid i = 1, 2, \ldots, N^2 - 1, N^2\}$. Then, the spectrum of the matrix P is simply a function of the vector $\vec{X}$. We can denote this $\sigma(\vec{X})$, where the operator σ is overloaded to mean the spectrum of a matrix* ***with respect to the vector representation****. The actual process for σ is not necessary to explicitly write out, since characterizing it will be sufficient for now. Essentially, σ is a function that would parse the random variables into the array form by index hacking. Then, it must compute the determinant of the matrix $P - \lambda I$, and solve for its roots. To summarize this, consider the flow chart below.*

To simplify the process, here is how we formalize the spectrum as a statistic. Suppose we sample $P \sim \mathcal{D}$. Then, we take its spectrum formally using σ as such:

$$P_{\text{array}} \to \vec{P} \to \sigma(\vec{P}) \to \text{index magic} \to \det(P - \lambda I) \to \sigma(P_{\text{array}})$$

2.1.2 Interlude: Ensembles

While the spectrum of a matrix provides a good summary of the matrix, a matrix is only considered a single point/observation in random matrix theory. Additionally, simulating large matrices and computing their eigenvalues becomes harder and more computationally expensive as $N \to \infty$. As such, to obtain more eigenvalue statistics efficiently, another dimension is introduced by motiving the *spectrum of a random matrix ensemble.*

Definition 2.1.2 (Ensemble Spectrum)**.** *If we have an ensemble $\mathcal{E}$, then we can naturally extend the definition of $\sigma(\mathcal{E})$. To take the spectrum of an ensemble, simply take the union of the spectra of each of its matrices. In other words, if $\mathcal{E} = \{P_i \sim \mathcal{D} \mid P_i \in \mathbb{F}^{N\times N}\}_{i=1}^{K}$, then $\sigma(\mathcal{E}) = \bigcup_{i=1}^{K} \sigma(P_i)$.*

For example, consider the following code example from the RMAT package.

Code Example (Spectrum of a Standard Normal Matrix Ensemble)**.** *Let $\mathcal{E} \sim \mathcal{N}(0,1)$ be an ensemble of 3×3 standard normal random matrices of size 3. We can generate the spectrum of $\mathcal{E}$, $\sigma(\mathcal{E})$ as follows:*

```
library(RMAT)
ens <- RME_norm(N = 3, mean = 0, sd = 1, size = 3)
spectrum_ens <- spectrum(ens)
# Outputs the following
spectrum_ens
...
        Re      Im   Norm Order
 1  1.7581  0.0000 1.7581     1
 2 -0.2614  1.0012 1.0347     2
 3 -0.2614 -1.0012 1.0347     3
 4  1.2327  0.4227 1.3032     1
 5  1.2327 -0.4227 1.3032     2
 6 -0.8504  0.0000 0.8504     3
 7 -0.5296  1.0508 1.1767     1
 8 -0.5296 -1.0508 1.1767     2
 9  0.7357  0.0000 0.7357     3
```

A common theme in this thesis will be that singleton matrices do not provide insightful information on their own. Rather, it is the collective behavior of a $\mathcal{D}$-distributed ensemble that tells us about how $\mathcal{D}$ impacts our spectral statistics. So in a way, ensemble statistics are the engine of this research.

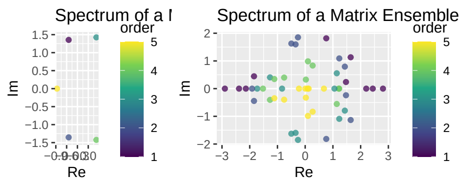

Figure 2.1: Spectrum of a Matrix versus an Ensemble

2.2 Ordered Spectra

2.2.1 Eigenvalue Ordering

When we motivate the idea of matrix dispersion in the next section, we will consider order statistics of that matrix's eigenvalues in tandem with its dispersion. However, to do so presupposes that we have a sense of what *ordered* eigenvalues means. Take a matrix P and its *unordered* spectrum $\sigma(P) = \{\lambda_j\}$. It is paramount to know what ordering scheme $\sigma(P)$ is using, because otherwise, the eigenvalue indices are meaningless! So, to eliminate confusion, we add an index to σ that indicates how the spectrum is ordered. Often, the ordering context will be clear and the indexing will be omitted. Consider the two following *ordering schema*:

Standard definitions of an ordered spectrum follows the standard ordering in the reals; denote this as the ordering by the **sign scheme**. Note that because total-ordering is only well-defined on the reals, we can only use this scheme when on a spectrum with real entries. So, we write the *sign-ordered spectrum* as follows:

$$\sigma_S(P) = \{\lambda_j : \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N\}_{j=1}^{N}$$

Alternatively, we can motivate a different scheme that properly handles complex eigenvalues. We could sort the spectrum by the norm of its entries; denote this as ordering using the *norm scheme*. This way, all the eigenvalues are mapped to a real value, in which we could use the sign-scheme of ordering. Without further ado, we write the *norm-ordered spectrum* as follows:

$$\sigma_N(P) = \{\lambda_j : |\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_N|\}_{j=1}^{N}$$

Note that when we take the norms of the eigenvalues, we essentially ignore "rotational" features of the eigenvalues. Signs of eigenvalues indicate reflection or rotation, so when we take the norm, we essentially become more concerned with scaling.

For example, consider the following plots, showing the difference in using the sign and norm ordering schemes for the same spectrum.

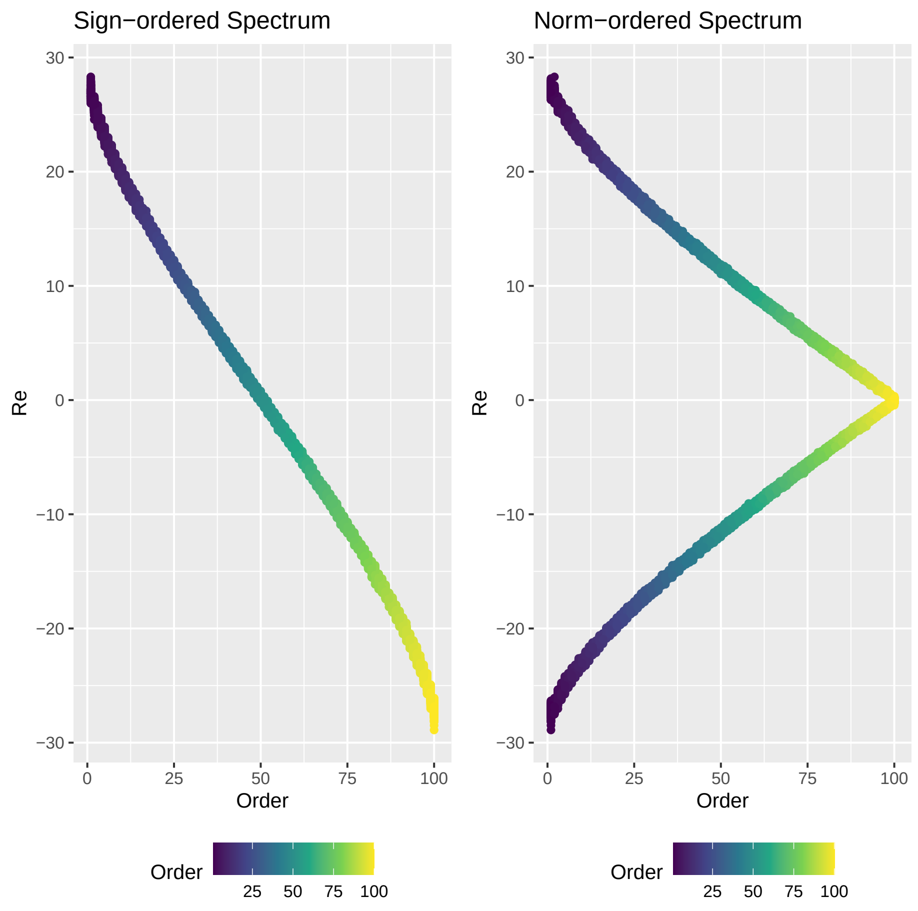

Figure 2.2: Spectrum displaying two different ordering scheme

Singular Values

An aternative to using the norm ordering scheme is using the singular values of the matrix. If a matrix is symmetric, the singular values are simply the norm of the eigenvalues. We ignore rotational features and focus solely on scale when we do so.

Suppose P is a random matrix. Then, we can take it singular values as such.

Definition 2.2.1 (Singular Values)**.** *The singular values of a matrix P are given by the square root of the eigenvalues of the corrosponding product of that matrix and its transpose. That is, $\sigma_+(P) = \sqrt{\sigma(P \cdot P^T)}$.*

Summary Table of Spectrum Schema

Table of Spectrum Schema			
Scheme	Matrix	Notation	Ordering
Sign-Ordered	P	$\sigma_S(P)$	$\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_N$
Norm-Ordered	P	$\sigma_N(P)$	$\lvert\lambda_1\rvert \geq \lvert\lambda_2\rvert \geq ... \geq \lvert\lambda_N\rvert$
Singular	$P \cdot P^T$	$\sigma_+(P)$	$\sqrt{\lambda_1} \geq \sqrt{\lambda_2} \geq ... \geq \sqrt{\lambda_N}$

2.2.2 Order Statistics

With eigenvalue ordering unambiguous and well-defined, we may proceed to start talking about their order statistics. In short, given a random sample of fixed size, order statistics are random variables defined as the value of an element conditioning on its rank within the sample. (See A.x)

In general, order statistics are quite useful and tell us a lot about how the eigenvalues distribute given a distribution. They tell us how the eigenvalues space themselves and give us useful upper and lower bounds.

For example, the maximum of a sample is an order statistic concerned with the highest ranked element. In our case, this could corrospond the largest eigenvalue of a spectrum. After all, a spectrum is a random sample of fixed size, so this statistic is well-defined.

Example (The Largest Eigenvalue)**.** *Suppose we have seek the largest eigenvalue distribution for a ensemble distribution $\mathcal{D}$, we would simulate an ensemble $\mathcal{E}$ and observe λ_1 for each of its matrices. Then, we can set the distribution of the largest eigenvalue for $\mathcal{D}$ by observing the distribution of λ_1.*

So, we fill consider the conditional order statistics $\mathbb{E}(\lambda_i \mid i)$ and $\text{Var}(\lambda_i \mid i)$.

2.3 Symmetric and Hermitian Matrices

2.3.1 Introduction

A very important class of matrices in Linear Algebra is that of Symmetric or Hermitian matrices (See A.x). Simply put, those are matrices which are equal to their conjugate transpose.

Note: Since real numbers are their own conjugate transpose, every Symmetric matrix is Hermitian. However, we will still delineate the two terms to avoid confusion.

In any case, one critical result in Linear Algebra that will be extensively wielded in this thesis is the fact that a matrix is Symmetric or Hermitian if and only if it has real eigenvalues. In other words:

$$P = \overline{P^T} \iff \sigma(P) = \{\lambda_i \mid \lambda_i \in \mathbb{R}\}$$

Having a complete set of real eigenvalues yields many great properties. For instance, if all eigenvalues are real, we have the option of observing either the sign-ordered spectrum or the norm-ordered spectrum. This way, we can preserve negative signs and we would not lose the rotational aspect of the eigenvalue when we study its statistics. That is just one reason out of many more why having real eigenvalues is quite nice.

2.3.2 Wigner's Semicircle Distribution

The eigenvalues of Hermitian matrices obey Wigner's Semicircle distribution. Since Hermitian matrices have real eigenvalues, then we can be more precise and generally say that the real component of the eigenvalues follow the semicircle distribution.

Definition 2.3.1 (Semicircle Distribtion)**.** *If a random variable X is semicircle distributed with radius $R \in \mathbb{R}^+$, then we say $X \sim SC(R)$. X has the following probability density function:*

$$P(X = x) = \frac{2}{\pi R^2}\sqrt{R^2 - x^2} \text{ for } x \in [-R, R]$$

Remark (Radius and Matrix Dimension)**.** *The dimension of the matrix determines the radius of the eigenvalues. Namely, if a Hermitian matrix P is $N \times N$, then its eigenvalues are semicircle distributed with radius $R = 2\sqrt{N}$. That is, $P^\dagger$ has a spectrum $\sigma(P) \sim SC(2\sqrt{R})$.*

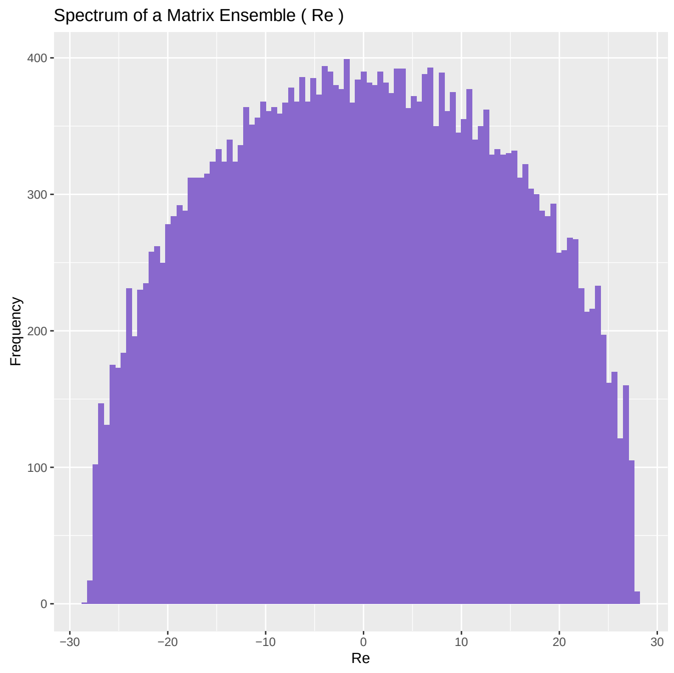

Figure 2.3: Eigenvalues of a Symmetric Matrix displaying the Semicircle Distribution

2.4 Findings

Normal matrices tend to have eigenvalues distributed about the complex disk.

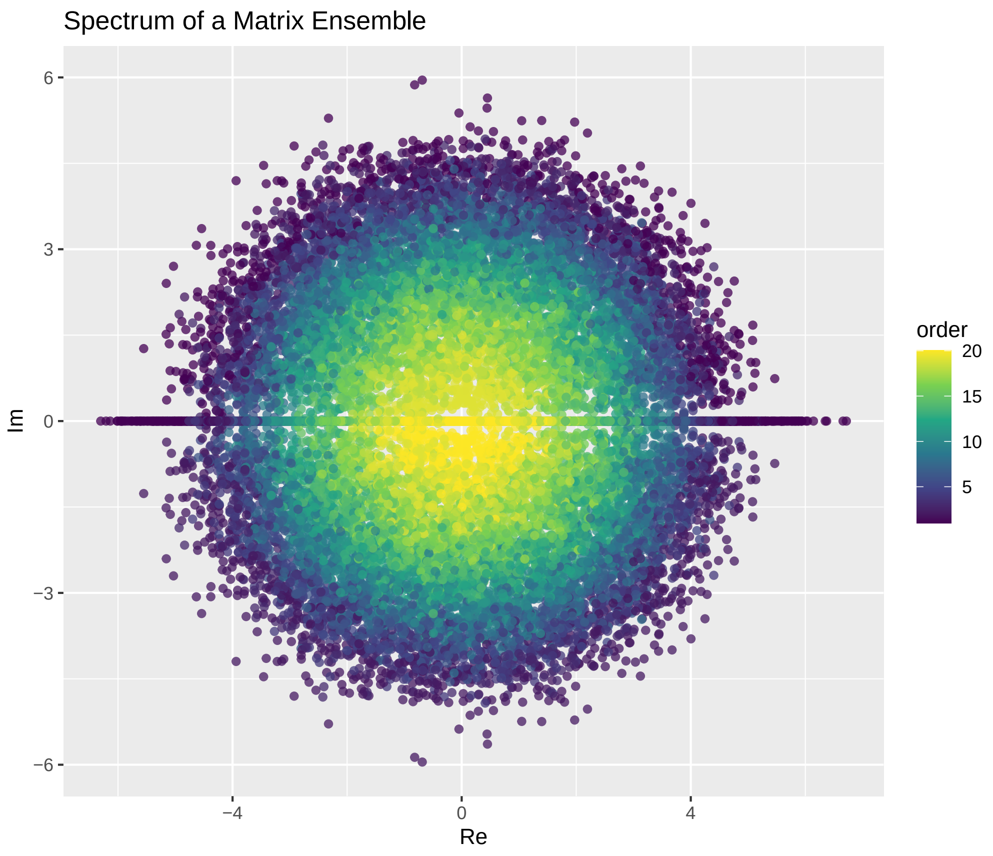

Figure 2.4: Spectrum of a Standard Normal Matrix ensemble

Chapter 3

Dispersions

3.1 Introduction

In this section, we define the final spectral statistic studied in this chapter: eigenvalue dispersions. As the name suggests, these statistics are concerned with the distirbution of the spacings between the eigenvalues. Interestingly, this is almost as literal as it gets when we use the word "spectral". In physics and chemistry, atomic spectra are essentially differences between energy levels or quanta, so the translation is close.

In any case, we will begin this chapter by first motivating a few definitions and formalisms in this section. Then, once our setup is ready, we will motivate the definition of a matrix's eigenvalue dispersion and formalize it as a statistic. To outline the section, we will first define two things: the dispersion metric and the pairing scheme. In simple terms, we formalize **what** "eigenvalue spacings" are and **which** eigenvalue pairs' spacings to consider. So, to begin, we first formally define an eigenvalue pair.

Definition 3.1.1 (Eigenvalue Pair). *Suppose P is a matrix and $\sigma(P)$ is its ordered spectrum. Then, an eigenvalue pair with respect to this ordered spectrum is denoted π_{ij}. It is defined as the ordered pair $\pi_{ij} = (\lambda_i, \lambda_j)$.*

Without further ado, we now motivate the dispersion metric.

3.1.1 Dispersion Metrics

Before we may even start to consider studying dispersions of eigenvalues, we must first formalize and make clear what "metric" of spacing we are using. To do so, we motivate the dispersion metric. In simple words, a dispersion metric is a function that takes in a pair of eigenvalues and returns a positive real number that represents some metric of dispersion or spacing.

Definition 3.1.2 (Dispersion Metric). *A dispersion metric $\delta : \mathbb{C} \times \mathbb{C} \to \mathbb{R}^+$ is defined as a function from the space of pairs of complex numbers to the positive reals. In simple terms, it is a way of measuring "space" between two complex numbers - our eigenvalues.*

In the scope of this thesis, we consider the following dispersion metrics below.

1. The standard norm: $\delta(z, z') = |z' - z|$
2. The β-norm: $\delta_\beta(z, z') = |z' - z|^\beta$
3. The difference of absolutes: $\delta_{\text{abs}}(z, z') = |z'| - |z|$

Remark (Symmetric Metrics)**.** *Note that the standard norm and the β-norm are symmetric operations. This means that switching the order of arguments will have no effect on the function's output. Otherwise, the difference of absolutes metric is not symmetric.*

While we have defined disperison metrics to be functions from $\mathbb{C}^2$, there is one special case where we can make an exception so that the domain of δ is not $\mathbb{R}^+$.

Remark (Identity Difference Heuristic)**.** *Suppose we take the arithmetic difference of two complex numbers. Then, the range of δ is $\mathbb{C}$. For this reason, we won't consider the arithmetic difference a formal dispersion metric, but we will honor it as a dispersion huerestic. As such, we will denote this as the "identity difference" huerestic and call it δ_{id}. So, we define $\delta_{id} : (z, z') \mapsto z' - z$*

Summary Table of Dispersion Metrics

For every dispersion metric, assume the functions' order of arguments is $\delta(z, z')$.

Table of Dispersion Metrics

Metric*	Notation	Formula	Symmetric	Parameters
Standard Norm	δ	$\|z' - z\|$	True	-
β-Norm	δ_β	$\|z' - z\|^\beta$	True	$\beta \in \mathbb{N}$
Difference of Absolutes	δ_{abs}	$\|z'\| - \|z\|$	False	-
Identity Difference	δ_{id}	$z' - z$	False	-

*Note that the identity difference is a heurestic, and not a formal metric.

3.1.2 Pairing Schema

The next thing we need to motivate before talking about eigenvalue dispersions are pairing schema. In simple terms, pairing schema are templates (for some $N \in \mathbb{N}$) for which eigenvalue pairs to pick. There are many subtle reasons why this is important, which will be covered in detail later. Without further ado, we motivate the pairing schema.

Before we may begin talking about pairing scheme, we define an auxilliary object - the spectral pairs of a matrix, which we denote $\sigma^{(2)}$. Since we are now talking about eigenvalue pairs, it is helpful to define this object before proceeding.

Definition 3.1.3 (Spectral Pairs)**.** *Suppose P is an $N \times N$ random matrix. Then, taking the spectral pair of P, denoted $\sigma^{(2)}(P)$ is equivalent to taking the Cartesian product of its ordered spectrum $\sigma(P)$. That is, $\sigma^{(2)}(P) = \sigma(P) \times \sigma(P) = \{(\lambda_i, \lambda_j) \mid i, j \in \mathbb{N}_N\}$.*

Now, to select eigenvalue pairs, we motivate the pairing scheme - this is what tells us which indices to select.

Definition 3.1.4 (Pairing Scheme)**.** *Suppose P is any $N \times N$ matrix and $\sigma^{(2)}(P)$ are its spectral pairs. A pairing scheme for the matrix P is a subset of* $\mathbb{N}_N \times \mathbb{N}_N$ *- a subset of pairs of numbers from $\mathbb{N}_N = \{1, \ldots, N\}$. In other words, it is a subset of pair indices for N objects - in our case, eigenvalues. We denote a pairing scheme as a set $\Pi = \{(\alpha, \beta) \mid \alpha, \beta \in \mathbb{N}_N\} \subseteq \mathbb{N}_N \times \mathbb{N}_N$. To take a matrix's spectral pairs with respect to* Π*, we simply take the set of eigenvalue pairs with the matching indices, $\sigma^{(2)}(P \mid \Pi) = \{(\lambda_\alpha, \lambda_\beta) \mid (\alpha, \beta) \in \Pi\}$.*

The reader might find this definition slightly obscure, and rightfully so. The definition is a mere formality, as we will usually only consider a few specific pairing schema. Seeing the explicit examples will hopefully make things more clear. With all the technical details aside, we can just say that a pairing scheme tells us which subset of eigenvalue pairs to consider. If one visualizes an array with N objects on two axes, we are simply choosing a subset of that plane. In fact, consider the following.

Remark (Proper Subset)**.** *If $\sigma^{(2)}(P)$ is the spectral pairs of the matrix P, then for any pairing scheme* Π*, we will find that* $\sigma^{(2)}(P \mid \Pi) \subseteq \sigma^{(2)}(P)$*. By definition, taking a spectral pair with respect to some pairing scheme constricts which pairs to select - meaning it is a proper subset of the general spectral pairs.*

Common Pairing Schema

Suppose P in an $N \times N$ square matrix, and $\sigma^{(2)}(P)$ are its spectral pairs. For every pairing scheme, assume $i, j \in \mathbb{N}_N$.

1. The unique pair combinations schema are two complementary pair schema. By specifying **either** $i > j$ or $i < j$, we characterize this scheme to entail all unique pair combinations of eigenvalues without repeats. The reason we call them upper and lower pair combinations is an allusion to the indices of the upper and lower triangular matrices.

   (a) Let $\Pi_>$ be the lower-pair combinations of ordered eigenvalues. This will be the standard unique pair combination scheme used in lieu of the argument orders of our dispersion metrics (more later). In this pairing scheme, the eigenvalue with the lower rank is always listed first, and the higher rank second.
   $$\sigma^{(2)}(P \mid \Pi_>) = \{\pi_{ij} = (\lambda_i, \lambda_j) \mid i > j\}_{i=1}^{N-1}$$

   (b) For completeness, we will also define the upper-pair scheme. $\Pi_<$ is the set of upper-pair unique combinations of ordered eigenvalues. Wont be used because we want bigger - smaller to make positive definite.
   $$\sigma^{(2)}(P \mid \Pi_<) = \{\pi_{ij} = (\lambda_i, \lambda_j) \mid i < j\}_{i=1}^{N-1}$$

   Benefits: Solves the issue of repeated pairs for symmetric dispersion metrics.

2. Let Π_C be the consecutive pairs of eigenvalues in a spectrum.
   $$\sigma^{(2)}(P \mid \Pi_C) = \{\tilde{\pi}_j = (\lambda_{j+1}, \lambda_j)\}_{j=1}^{N-1}$$

   Benefits: This pairing scheme gives us the minimal information needed to express important bounds and spacings in terms of its elements.

3. Let Π_0 be all the pairs in a spectrum. For completeness, we define this pairing scheme as the implicit pairing scheme for the spectral pairs of a matrix.
   $$\sigma^{(2)}(P \mid \Pi_0) = \sigma^{(2)}(P) = \{\pi_{ij} = (\lambda_i, \lambda_j) \mid i, j \in \mathbb{N}_N\}$$

Consecutive Pairs

In the definition of the consecutive pairing scheme Π_C, a new notation of eigenvalue pair with one index is introduced; it takes the form $\tilde{\pi}_j$. This notation is used to denote the consecutive eigenvalue pair for a given matrix. The consecutive eigenvalue pairs are so special that we denote them with a unique notation for convenience. It also makes the discussion regarding order statistics more intrinisic.

Definition 3.1.5 (Consecutive Pairs)**.** *Suppose P is a matrix and $\sigma(P)$ is its ordered spectrum. Then, let $\tilde{\pi}_j$ denote a pair of consecutive eigenvalues, the largest of the two being the j^{th} largest eigenvalue.*

Since the consecutive pair scheme can be sufficiently indexed by one index (j), we will use the convention of omitting the index of the smaller eigenvalue to be more concise and idiomatic. So, whenever one sees the notation $\tilde{\pi}_j$, think the j^{th} largest eigenvalue and its smaller neighbour.

Example (The Largest Eigenvalues)**.** *With this notation at hand, we say that $\tilde{\pi}_1$ represents the pair of the two largest eigenvalues in an ordered spectrum.*

$$\tilde{\pi}_1 = (\lambda_2, \lambda_1)$$

Remark (Order Statistics)**.** *Since the pairing scheme assumes the eigenvalues are given in an ordered spectrum, our analysis will be mostly leveraging this fact by using the indices. However, as notated above, some pairing schemes are indexed with i and j, and some like the consecutive pairs only with j. These indices are interchangable with "order statistic", so in the next section, we will devise a synthetic order statistic that uses two indices.*

Summary Table of Pairing Schema

Suppose P is an $N \times N$ matrix. Then, its spectral pairs $\sigma^{(2)}(P)$ may take on the following pairing schemes.

Table of Pairing Schema

Scheme	Notation	Formula
Lower	$\Pi_<$	$\{(i,j) \mid i < j \text{ for } i,j \in \mathbb{N}_N\}$
Upper	$\Pi_>$	$\{(i,j) \mid i > j \text{ for } i,j \in \mathbb{N}_N\}$
Consecutive	Π_C	$\{(i,j) \mid i = j+1 \text{ for } i,j \in \mathbb{N}_N\}$
All	Π_0	$\{(i,j) \mid i,j \in \mathbb{N}_N\}$

Another way we can define the pairing scheme is defining an auxilliary object: the eigenvalue (pair) matrix.

An Alternative Represetation: Eigenvalue Matrix

Definition 3.1.6 (Eigenvalue Matrix)**.** *Suppose P is an $N \times N$ square matrix and $\sigma^{(2)}(P)$ are its spectral pairs. Then, the eigenvalue matrix of P, given by $\Lambda(P)$ is the matrix with entries $\pi_{ij} = (\lambda_i, \lambda_j)$ for $\pi_{ij} \in \sigma^{(2)}(P)$. Again, it is given that the eigenvalues λ_i come from some* ***ordered*** *spectrum $\sigma(P)$ as per the definition of spectral pairs.*

Using the eigenvalue matrix, we have an alternative, potentially more intrinsic, equivalent representation of pairing schemes. For example, the entire matrix represents all pairs. The upper and lower schemes represent the upper-triangular matrix and the lower-triangular respectively. Lastly, the consecutive pairs are simply the nearest off-diagonal in the lower-triangle of the matrix.

Example (Eigenvalue Matrix for a 5×5 Matrix)**.** *Suppose we have a 5×5 matrix P and its corrosponding spectral pairs $\sigma^{(2)}(P)$. Then, its eigenvalue matrix has the following strucutre:*

$$\begin{bmatrix} - & \pi_{12} & \pi_{13} & \pi_{14} & \pi_{15} \\ \tilde{\pi}_1 & - & \pi_{23} & \pi_{24} & \pi_{25} \\ \pi_{31} & \tilde{\pi}_2 & - & \pi_{34} & \pi_{35} \\ \pi_{41} & \pi_{42} & \tilde{\pi}_3 & - & \pi_{45} \\ \pi_{51} & \pi_{52} & \pi_{53} & \tilde{\pi}_4 & - \end{bmatrix}$$

With the matrix analogy, describing the pairing schemes can be much simpler. For instance, the **lower pairs** are given by the indices of the entries in the lower triangle of the matrix. Analogously, the **upper pairs** by those of the upper triangle of the matrix. The **consecutive pairs**, given our default lower pair scheme, are given by the lower main off-diagonal band of the matrix.

3.1.3 Dispersions

Alas, with dispersion metrics and pairing schemes defined, we are finally able to motivate the definition of a matrix dispersion for both singleton matrices and their ensemble counterparts.

Definition 3.1.7 (Dispersion)**.** *Suppose P is an $N \times N$ matrix, and $\sigma^{(2)}(P)$ are its spectral pairs. The dispersion of P with respect to the pairing scheme Π and dispersion metric δ is denoted by $\Delta_\delta(P \mid \Pi)$ and it is given by the following:*

$$\Delta_\delta(P \mid \Pi) = \{\delta(\pi_{ij}) \mid \pi_{ij} \in \sigma^{(2)}(P \mid \Pi)\}$$

Consider the following code example, generating the dispersion of standard normal 5×5 matrix with respect to the consecutive pairing scheme.

Code Example (Consecutive Pair Dispersion of a Standard Normal Matrix)**.** *In our notation, we are simulating the disperison of $P \sim \mathcal{N}(0, 1)$ where $P \in \mathbb{R}^{5\times5}$. Specifically, we are simulating $\Delta(P \mid \Pi_C)$. In the code implementation, we obtain an array for every dispersion metric δ.*

```
library(RMAT)
P <- RM_norm(N = 5, mean = 0, sd = 1)
disp_P <- dispersion(P, pairs = "consecutive")
# Outputs the following
disp_P
...
i  j  eig_i       eig_j       id_diff     iddiff_norm abs_diff diff_ij
2  1 -0.54-1.35i -0.54+1.35i  0.00+2.71i 2.71          0.00     1
3  2  0.23+1.43i -0.54-1.35i -0.77-2.78i 2.88          0.02     1
4  3  0.23-1.43i  0.23+1.43i  0.00+2.85i 2.85          0.00     1
5  4 -0.87+0.00i  0.23-1.43i  1.09-1.43i 1.80          0.57     1
```

Next, we extend the definition of dispersion for an ensemble as we usually do.

Definition 3.1.8 (Ensemble Dispersion)**.** *If we have an ensemble $\mathcal{E}$, then we can naturally extend the definition of $\Delta_\delta(\mathcal{E} \mid \Pi)$. To take the dispersion of an ensemble, simply take the union of the dispersions of each of its matrices. In other words, if $\mathcal{E} = \{P_i \sim \mathcal{D}\}_{i=1}^{K}$, then its dispersion is given by:*

$$\Delta_\delta(\mathcal{E} \mid \Pi) = \bigcup_{i=1}^{K} \Delta_\delta(P_i \mid \Pi)$$

Consider the following code example, generating the dispersion of a beta ensemble with respect to the consecutive pairing scheme.

Code Example (Consecutive Pair Dispersions of a Beta Ensemble)**.** *In our notation, we are simulating the disperison of $\mathcal{E} \sim \mathcal{H}(\beta = 4)$ where $P \in \mathbb{R}^{5\times5}$. Specifically, we are simulating $\Delta(P \mid \Pi_C)$. In the code implementation, we obtain an array for every dispersion metric δ.*

```
library(RMAT)
ens <- RME_beta(N = 4, beta = 4, size = 3)
disp_ens <- dispersion(ens, pairs = "consecutive")
# Outputs the following
disp_ens
...
i j eig_i    eig_j    id_diff  iddf_norm abs_diff diff_ij
2 1 -3.78+0i 4.00+0i   7.78+0i 7.78      0.22     1
3 2 2.06+0i  -3.78+0i -5.84+0i 5.84      1.72     1
4 3 0.19+0i  2.06+0i   1.88+0i 1.88      1.88     1
2 1 3.80+0i  -4.00+0i -7.80+0i 7.80      0.20     1
3 2 -1.80+0i 3.80+0i   5.60+0i 5.60      2.00     1
4 3 0.89+0i  -1.80+0i -2.69+0i 2.69      0.92     1
2 1 3.51+0i  -3.53+0i -7.04+0i 7.04      0.03     1
3 2 1.35+0i  3.51+0i   2.16+0i 2.16      2.16     1
4 3 -0.67+0i 1.35+0i   2.02+0i 2.02      0.68     1
```

3.2 Dispersion Analysis

With dispersions well defined, we provide a few guidelines for analyzing a dispersion of a matrix. We will synthesize all the techniques, notations, and remarks in the previous section to provide a comprehensive manual on how to analyze the dispersion of a matrix.

Conjugate Pairs

Often, we find that our eigenvalues come in conjuagte pairs. This is a result of the conjugate pair root theorem, which states that polynomials tend to have roots that come in conjugate pairs. This follows because the characteristic polynomial is just like any other. That being said, we can notice conjugate pairs in our dispersion table in a few ways.

First, we can directly see them by comparing (eig-i) and (eig-j). A better way of spotting conjugate pairs is by looking at either the abs-diff column or the id-diff column. The absolute difference of two eigenvalues which are conjugate pairs is zero since they have the same size. Alternatively, we can characterize conjugate pairs by their purely imaginary identity difference. Although these measures do not ensure that the eigenvalues at hand are conjugate pairs, they almost always do.

3.2.1 Order Statistics

With eigenvalue dispersions and eigenvalue orderings well-defined, we may proceed to start talking about their order statistics.

In **Chapter 2**, we observed simple order statistics regarding the eigenvalues. This time around, we have two eigenvalues being compared at once, so there needs to be further setup with regards to dispersion order statistics.

However, prior to introducing any new concepts, there is one scenario where using one index is sufficient in terms of discussing dispersion order statistics.

Remark (Consecutive Pair Dispersions). *When we consider the dispersion with repsect to the consecutive pairs, things are much simpler since our pairs are intrinsically defined by one index (j). This is because we are observing the j^{th} eigenvalue and its smaller neighbour. As such, we will consider dispersion statistics in the form of $\mathbb{E}(\tilde{\pi}_j \mid j)$ and $Var(\tilde{\pi}_j \mid j)$.*

Otherwise, we introduce a new synthetic order statistic called the **ranking difference class**. Since we are no longer observing a single eigenvalue at a given rank, we will need a way to standardize observing a pair of eigenvalues at a time. To do so, we introduce a new **eqivalence relation** (see A.x) called the **ranking difference**. As the name suggests, it is precisely the integer difference of the eigenvalue orders.

Definition 3.2.1 (Ranking Difference). *The ranking difference is a function $\rho : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ which takes the index of two ordered eigenvalues and returns their difference. In other words, it is the function $\rho : (\lambda_i, \lambda_j) \mapsto (i - j)$.*

With the ranking difference, we may now take the spectral pairs of some random matrix with respect to either the lower ($\Pi_<$) or upper ($\Pi_>$) pairing scheme and partition the eigenvalues into distinct equivalence classes.

Remark (Equivalence Relation). *Formally speaking, we may define $\sim_\rho$ as an equivalence relation. It satsifies all three properties of reflexivity, symmetry, and transitivity. More precisely, it is an equivalence class which partitions $\mathbb{N}_N \times \mathbb{N}_N$ into $N-1$ equivalence classes given by $[r]_\rho$ for $r \in \{1, \dots, N-1\}$. We would define the equivalence relation $\sim_\rho$ as follows:*

$$(\lambda_a, \lambda_b) \sim_\rho (\lambda_c, \lambda_d) \iff (a - b) = (c - d)$$

So, two eigenvalue pairs would belong to the same class $[r]_\rho$ if their ranking difference is the same. That is,

$$[r]_\rho = \{(\lambda_\alpha, \lambda_\beta) \mid \alpha - \beta = r\}$$

Remark (Matrix Analogy). *Using the eigenvalue matrix analogy, $\sim_\rho$ partitions the matrix into diagonal bands. The diagonal represents $[0]_\rho$, the first off-diagonal represents $[1]_\rho$, and so on... Notice that this means every equivalence class has a different size.*

With this partition in mind, we can consider various statistics conditioning on the value of r. Conditioning on r will be especially useful in the cases where we are considering matrices like the Hermite-β matrices; the eigenvalues of those matrices tend to *repel*, so to speak, and we can observe these patterns using r.

Consider the following example.

Example (Ranking Differences for a 4×4 Matrix). *Suppose P is a 4×4 random matrix. Then, if we take the spectral pairs with respect to the lower pairs $\sigma^{(2)}(P \mid \Pi_<)$ and partition it by ρ, we get the following classes:*

1. $[1]_\rho = \{\tilde{\pi}_1, \tilde{\pi}_2, \tilde{\pi}_3\}$
2. $[2]_\rho = \{\pi_{31}, \pi_{42}\}$
3. $[3]_\rho = \{\pi_{41}\}$

3.2.2 Conditional Statistics

We will considering the conditional statistics $\mathbb{E}(\rho_{ij} \mid \rho)$ and $\operatorname{Var}(\rho_{ij} \mid \rho)$.

[Plots]

3.3 Analytical Results

3.3.1 Case Study: Wigner's Surmise

Wigner's surmise is a result found by Richard Wigner regarding the limiting distribution of eigenvalue spacings of for symmetric matrices. To start talking about this, we must talk about normalized spacings, which are the precise items considered in the distribution. Before, we can talk about the normalized spacing, we define the mean spacing. Recall that the mean spacing showed up in the normalization of the largest eigenvalue in the *Tracy-Widom distribution.* This is precisely the same notion.

Definition 3.3.1 (Mean Spacing)**.** *Suppose P is an $N \times N$ symmetric matrix. Then, the mean (eigenvalue) spacing, denoted $\langle s \rangle = \langle \lambda_j - \lambda_i \rangle$ is given by taking the consecutive pairs of the sign-ordered spectrum of P and taking the mean of their difference. That is, $\langle s \rangle = \mathbb{E}(\lambda_j - \lambda_{j-1})$ for $j = 1, \dots, N-1$).*

So, with the mean spacing defined, we now define the normalized spacing between a pair of consecutive eigenvalues below.

Definition 3.3.2 (Normalized Spacing)**.** *Suppose P is an $N \times N$ symmetric matrix, and $\sigma(P)$ are its real, sign-ordered eigenvalues. Then, the normalized spacing of the j^{th} pair of eigenvalues, denoted s_j is given by the following formula. Note that in the alternate notation $\delta = \delta_{id}$.*

$$s_j = \frac{(\lambda_j - \lambda_{j+1})}{\langle s \rangle} = \frac{\delta(\tilde{\pi}_j)}{\mathbb{E}[\delta(\tilde{\pi}_j)]}$$

The analytical results for the Hermite β-ensembles for $\beta = 1, 2, 4$, denoted by w_β, are stated below.

$$w_1(s) = \frac{\pi}{2} \exp\Big(-\frac{\pi}{4} s^2\Big)$$
$$w_2(s) = \frac{32}{\pi^2} s^2 \exp\left(-\frac{4}{\pi} s^2\right)$$
$$w_4(s) = \frac{2^{18}}{3^6 \pi^3} s^4 \exp\left(-\frac{64}{9\pi} s^2\right)$$

[Plot]

Chapter 4

β-Ensembles

4.1 Introduction

In this chapter, we will talk about the Hermite β-ensembles more in depth. The beta ensembles have wide applications in statistical physics, eningeering, and many other places. They are defined by the joint density of their eigenvalues, and have a special characterization discussed in the next subsection.

4.1.1 Hermite β-Ensembles

The Hermite β-ensembles, also called the Gaussian ensembles, are an important class of random matrix ensembles studied in engineering, statistical physics, and probability theory. Parameterized by $\beta \in \mathbb{N}$ through the Dyson index, this ensemble is charecterized by a few things.

- The Dyson index β corrosponds to the number of real number of components the subject matrices have.
- The subject matrices are classically defined for $\beta = 1, 2, 4$ and they corrospond to matrices with real, complex, and quaternionic entries. The corrosponding fields are $\mathbb{R}$, $\mathbb{C}$, and $\mathbb{H}$.
- The matrices in this ensemble, most importantly, have a feature called conjugation invariance. With respect to the conjugation by the respective group of matrices.
- Most importantly, the eigenvalues are determined by the joint probability density function given below.

Definition 4.1.1 (Hermite β-ensembles)**.** *The Hermite β-ensembles, commonly known as the Gaussian ensembles, are an ensemble of random matrices parameterized by β, and their eigenvalues have the joint probability density function:*

$$f_\beta(\Lambda) = c_H^\beta \prod_{i<j} |\lambda_i - \lambda_j|^\beta e^{-1/2 \sum_i \lambda_i^2}$$

where the normalization constant c_H^β is given by:

$$c_H^\beta = (2\pi)^{-n/2} \prod_{j=1}^{n} \frac{\Gamma(1+\beta/2)}{\Gamma(1+\beta j/2)}$$

To simulate matrices from the β-ensemble, we will be using a recent result published in "Matrix Models for Beta Ensembles" **?**. This makes the β-ensemble a canonical example of an implicity distributed matrix; we do not care about the actual distribution of the entries, but rather the effect they have on the eigenvalues (trace) of the matrices. The algorithm used is directly cited from the results of Dumitriu's paper, and can be found in Algorithms B.1.3.

Code Example (Hermite Beta = 2 Ensemble)**.** *Let $\mathcal{D} = \mathcal{H}(\beta = 2)$. We can generate $\mathcal{E} \sim \mathcal{D}$, an ensemble of 4×4 Hermite matrices ($\beta = 2$) of size 10 as such:*

```
library(RMAT)
ensemble <- RME_beta(N = 4, beta = 2, size = 10)
# Outputs the following
ensemble
...
[[10]]
          [,1]      [,2]        [,3]     [,4]
[1,] 0.7246302 1.8893868  0.00000000 0.000000
[2,] 1.8893868 1.5278221  0.68840045 0.000000
[3,] 0.0000000 0.6884004 -0.03876104 1.944495
[4,] 0.0000000 0.0000000  1.94449533 1.042741
```

4.1.2 Dimitriu's Matrix Model

To generate Hermite β matrices, we consider the result of Dimitriu's paper. We obtain the following algorithm.

Algorithm 4.1.1 (Dimitriu's Beta Matrix)**.**

1. *To simulate an $N \times N$ beta matrix, fix $N \in \mathbb{N}$.*
2. *Start by taking a diagonal of $\mathcal{N}(0, 2)$ variables.*
3. *Set both of the nearest off-diagonals to the row that samples from a $\chi(df = c_j)$ where $c_j = \beta \cdot j$ for columns spanning $j = 1, \ldots, n-1$.*

4.2 Spectra

We know that β-matrices must be symmetric. So, their eigenvalues must be real. Any imaginary component we observe is simply computational error, and we may safely ignore it. It is good to see that the error is uniform and small.

Remark (Implicit Distribution)**.** *Note that the beta ensemble is an ensemble characterized by some joint density function on its eigenvalues. So, this is a specific instance of an implcitly distributed matrix. There are many things to consider about* ***identifiability*** *that are subtle, but important. Recall that in our formalization of a specturm as a formal statistic, we vectorized the matrix then tooks the determinant of the characteristic polynomial that came about it. For this reason, we can see that while Dimitriu's model provides an explicit formula, there is an issue of identifiability. That is, given some eigenvalues, there is no injective function to the characteristic polynomial that produced it. Rather, there are infinitly many equivalence classes of characteristic polynomials (and such, random matrices) that surjectively produce a given multiset of eigenvalues.*

Remark (Floating Point Errors)**.** *Because the model involves diving by $\sqrt{2}$, floating point errors and algorithmic systemic error will yield small, but negligible imaginary components.*

4.3 Dispersions

Consider the following plot of the sign-sorted eigenvalue dispersions below.

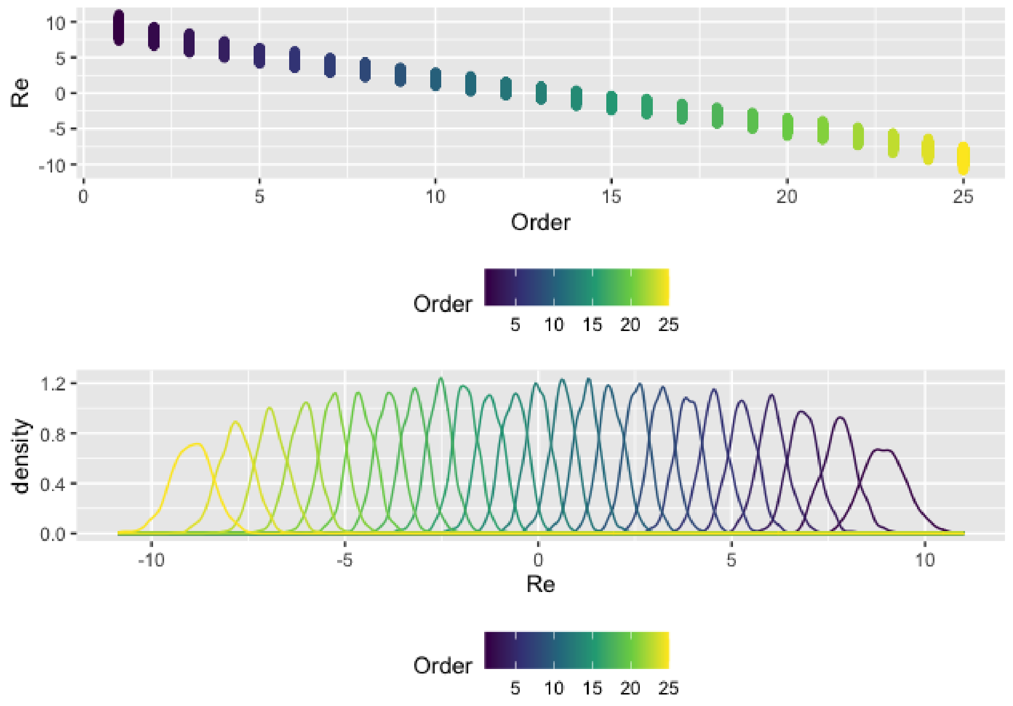

Figure 4.1: Dispersions of a $\beta = 4$ matrix with respect to ranking difference

Appendix A

Math Review

A.1 Linear Algebra

A.1.1 Matrices

Definition A.1.1 (Eigenvalue). *Suppose $P \in \mathbb{F}^{n\times n}$ is a square matrix. Then, the eigenvalues of the matrix P are precisely the roots of the characteristic polynomial of P, given by $char_P(\lambda) = \det(P - \lambda I)$. The polynomial $char_P(\lambda)$ has degree n. So by the Fundamental Theorem of Algebra, P has a multiset of n eigenvalues.*

Definition A.1.2 (Inverse Matrix). *Suppose there is a matrix $P \in \mathbb{F}^{m\times n}$. Then, P^{-1} is its inverse matrix iff multiplying it by P returns the identity matrix. That is, the inverse matrix must satisfy:*

$$P^{-1}P = I$$

Definition A.1.3 (Transpose Matrix). *Suppose there is a matrix $P = (p_{ij}) \in \mathbb{F}^{m\times n}$. Then, its transpose matrix, $P^T = (q_{ij}) = (p_{ji}) \in \mathbb{F}^{n\times m}$ matrix whose columns are the rows of the original matrix.*

Definition A.1.4 (Conjugate Transpose Matrix). *Suppose there is a matrix $P = (p_{ij}) \in \mathbb{C}^{m\times n}$. Then, its conjugate transpose matrix, $P^\dagger = (q_{ij}) = (\overline{p_{ji}}) \in \mathbb{F}^{n\times m}$ is a matrix whose columns are the rows of the original matrix.*

Definition A.1.5 (Symmetric Matrix). *A matrix P is symmetric iff it is equal to its transpose:*

$$P = P^T$$

Definition A.1.6 (Hermitian Matrix). *A matrix P is Hermitian iff it is equal to its conjugate transpose:*

$$P = P^\dagger$$

Definition A.1.7 (Orthogonal Matrix). *A matrix P is called orthogonal iff its transpose is its inverse:*

$$P^T = P^{-1}.$$

Theorem A.1.1 (Orthogonal Group)**.** *The set of* all *unitary matrices in* $\mathbb{F}^{n\times n}$ *is a matrix group. (It is called the orthogonal group.)*

Definition A.1.8 (Unitary Matrix)**.** *A matrix P is called unitary iff its conjugate transpose is its inverse:*

$$P^{\dagger} = P^{-1}.$$

Theorem A.1.2 (Unitary Group)**.** *The set of* all *unitary matrices in* $\mathbb{C}^{n\times n}$ *is a matrix group. (It is called the unitary group.)*

A.1.2 Proof: Real Symmetric Matrices have Real Eigenvectors

Notation. For notational convenience, for any $N \in \mathbb{N}$, let $\widetilde{N} = \{1, \ldots, N\}$.

In this document, we prove that for any $M \times M$ real symmetric matrix, S_M, there exists for some eigenvalue λ, a corrosponding **real** eigenvector $\vec{v} \in \mathbb{R}^M$. Prior to starting the main proof, we begin with a lemma.

Lemma. Suppose we have a $M \times M$ real symmetric matrix with a some eigenvalue λ. If there we have a corrosponding eigenvector $v \in \mathbb{C}^M$, then every entry of v, say v_i is equal to a **real** linear combination of the other entries $v_j \mid j \neq i$.

So, we will show that:

$$\forall i \in \widetilde{M} : v_i = \sum_{j \neq i} c_j v_j \quad (c_j \in \mathbb{R})$$

Proof of Lemma. Begin by taking a real symmetric matrix S_M for some $M \in \mathbb{N}$. Suppose we have an eigenvalue λ. Then, if we have some eigenvector v, we know that:

$$(1) : \forall i \in \widetilde{M} : a_1 v_1 + \cdots + d_i v_i + \cdots + a_{m-1} v_m = \lambda v_i \quad (a_j \in \mathbb{R})$$

We obtain (1) by expanding the equality $Av = \lambda v$ and noticing that every row of Av is expressible as the sum of the non-diagonal entries multiplied by $v_j \mid j \neq i$ plus $d_i v_i$. Note that since our matrix is symmetric, for some rows, some of the constants a_j are not distinct but this should not raise any issues. Next, we collect the terms:

$$\forall i \in \widetilde{M} : a_1 v_1 + \cdots + a_{m-1} v_m = v_i(\lambda - d_i)$$

Since S_M is a real symmetric matrix, the a_j terms are real so we can say:

$$\forall i \in \widetilde{M} : v_i(\lambda - d_i) = \sum_{j \neq i} a_j v_j \quad (a_j \in \mathbb{R})$$

Finally, divide both sides by $(\lambda - d_i)$. Since S_M is a real symmetric matrix, we know $\lambda \in \mathbb{R}$ then also $(\lambda - d_i) \in \mathbb{R}$. On the right hand side, the coefficients of the v_j become $\frac{a_j}{(\lambda - d_i)}$. Since $a_j \in \mathbb{R}$, then also $\frac{a_j}{(\lambda - d_i)} \in \mathbb{R}$. Letting $c_j = \frac{a_j}{(\lambda - d_i)}$, we obtain:

$$\forall i \in \widetilde{M} : v_i = \sum_{j \neq i} c_j v_j \quad (\forall j : c_j \in \mathbb{R})$$

Thus, for any $M \in \mathbb{N}$, a real symmetric matrix with eigenvalue λ must have a corrosponding eigenvector v such that each of its entries is expressible as a real linear combination of the other entries. □

Now, we will prove the main theorem.

Theorem (**Taqi**). Suppose we have a $M \times M$ real symmetric matrix, S_M. Then, we will show that there exists for some eigenvalue λ, a corrosponding **real** eigenvector $\vec{v} \in \mathbb{R}^M$.

Proof. For this proof we will induct on the dimension of the matrix, M. So let the inductive statement be

$$f(M) : S_M \text{ has a real eigenvector } v \text{ corrosponding to an eigenvalue } \lambda$$

Base Case. Take the base case $M = 2$. Then by **Zoom Meeting 11.12**, we know $f(2)$ is true.

Inductive Step. For our inductive step, we need to show that $f(M) \Rightarrow f(M{+}1)$. So, let us assume $f(M)$. This means that we can assume any real symmetric matrix S_M has a real eigenvector $v \in \mathbb{R}^M$ corrosponding to λ.

Next, we will write S_{M+1} as the matrix S_M augmented by some $u \in \mathbb{R}^M$ as follows:

$$S_{M+1} = \left[\begin{array}{c|c} S_M & u \\ \hline u^T & d_{M+1} \end{array}\right]$$

From our lemma, we use the fact that S_{M+1} is symmetric and our assumption of $f(M)$ to obtain:

$$(1) : \forall i \in \{1, \ldots, m+1\} : v_i = \sum_{j \neq i} c_j v_j \quad (c_j \in \mathbb{R})$$

$$(2) : \forall i \in \tilde{M} : v_i \in \mathbb{R}$$

In particular for (2), we know that $v_i = \left(\sum_{j \neq i} \frac{a_j}{d_i - \lambda} v_j\right)$.

From (1), we know that for row $i = m+1$: $v_{m+1} = \sum_{j \neq m+1} c_j v_j \quad (c_j \in \mathbb{R})$ By (2), this is a linear combination of real entries v_i. Since $v_{m+1} \in \mathbb{R}$, it follows that:

$$\forall i \in \{1, \ldots, m+1\} : v_i \in \mathbb{R}$$

So, we have established that $f(m) \Rightarrow f(M+1)$.

By the induction, the theorem is proved. $\square$.

A.2 Probability Theory

Definition A.2.1 (Random Variable)**.** *A random variable $X : \Omega \to \mathbb{R}$ is a function from some sample space $\Omega = \{s_i\}_{i=1}^{n}$ to the real numbers $\mathbb{R}$. The sample space is taken to be any set of events such that the probability function corrosponding to the random variable, p_X exhausts over all the events in Ω. In other words, we expect $\int_\Omega p_X(s) = 1$.*

Definition A.2.2 (Statistic)**.** *A statistic is formally defined as a function of a vector of random variables. So, for instance f is a statistic of a vector of random variables $\bar{X}$. Its observed value is given by $f(\vec{X})$.*

Example (Mean Statistic)**.** *For instance, the mean of a random sample $\vec{X}$ is a statistic. Formally, we would define the statistic $f : \vec{X} \to \frac{\sum_i x_i}{N}$. So the mean of the random sample is defined as $\bar{x} = f(\vec{X})$.*

Definition A.2.3 (Order Statistic)**.** *Suppose $\vec{X} = \{X_i\}_{i=1}^{N}$ is an ordered vector of random variables. Then, the i^{th} order statistic of X is given by X_i.*

Example (Order Statistic)**.** *Suppose $X = (20, 7, 2, 1)$. The smallest (fourth) order statistic is 1. The second order statistic is 7. The largest (first) order statistic is 20.*

A.3 Markov Chains

Definition A.3.1 (Markov Chain)**.** *Say a set of random variables X_i each take a value in a set, called the state space, $S_M = \{1, 2, \dots, M\}$. Then, a sequence of such random variables $X_0, X_1, \dots, X_n$ is called a Markov Chain if the following conditions are satisifed:*

- $\forall X_i : X_i$ has support and range $S_M = \{1, 2, ..., M\}$.
- *(Markov Property)* The transition probability from state $i \to j$, $\mathrm{P}(X_{n+1} = j \mid X_n = i)$ is conditionally independent from all past events in the sequence $X_{n-1} = i', X_{n-2} = i'', \dots, X_0 = i^{(n-1)}$, excluding the present/last event in the sequence. In other words, given the present, the past and the future are conditionally independent.

  $$\forall i, j \in S_M : \mathrm{P}(X_{n+1} = j \mid X_n = i) = \mathrm{P}(X_{n+1} = j \mid X_n = i, X_{n-1} = i', \dots, X_0 = i^{(n-1)})$$

Definition A.3.2 (Transition Matrixx)**.** *Let $X_0, X_1, \dots, X_M$ be a Markov Chain with state space S_M. Letting $q_{ij} = P(X_{n+1} = j \mid X_n = i)$ be the transition probability from $i \to j$, then the matrix $Q \in \mathcal{M}_{\mathbb{R}^+}[M \times M] : Q = (q_{ij})$ is the transition matrix of the chain. Q must satisfy the following conditions to be a valid transition matrix:*

Definition A.3.3 (Transition Matrix)**.** *Take a Markov Chain with states $1, \dots, M$. Letting $q_{ij} = P(X_{n+1} = j \mid X_n = i)$ be the transition probability from $i \to j$, then the matrix $Q = (q_{ij})$ is the transition matrix of the chain. For this transiton matrix to be valid, its rows have to be stochastic, meaning their entries sum to 1; $\forall i \in 1, \dots, M : \sum_{j \in 1,\dots,M} q_{ij} = 1$.*

- Q is a non-negative matrix. That is, note that $Q \in \mathcal{M}_{\mathbb{R}^+}[M \times M]$ so every $q_{ij} \in \mathbb{R}^+$. This follows because probabilities are necessarily non-negative values.
- The entries of every row i of Q must sum up to 1. This may be understood as applying the law of total probability to the event of transitioning from any given state $\forall i \in S_M$. In other words, the chain has to go somewhere with probability 1.

  $$\forall i \in S_M : \sum_{j \in S_M} q_{ij} = 1$$

- Note, it is NOT necessary that the converse holds. The columns of our transition matrix need not sum to 1 for it to be a valid transition matrix.

Definition A.3.4 (n-Step Transition Probability)**.** *The $n-$step transition probability of $i \to j$ is the probability of being at j exactly n steps after being at i. We denote this value $q_{ij}^{(n)}$:*

$$q_{ij}^{(n)} : \mathrm{P}(X_n = j \mid X_0 = i)$$

Realize:

$$q_{ij}^{(2)} = \sum_{k \in S_M} q_{ik} \cdot q_{kj}$$

Because by definition, a Markov Chain is closed under a support/range of S_M so the event $i \to j$ may have taken any intermediate step $k \in S_M$. Realize by notational equivalence, $Q^2 = (q_{ij}^{(2)})$. Inducting over n, we then obtain that:

$$q_{ij}^{(n)} \text{ is the } (i,j) \text{ entry of } Q^n$$

Definition A.3.5 (Marginal Distribution of Xn)**.** *Let* $\boldsymbol{t} = (t_1, t_2, \ldots, t_M)$ *such that* $\forall i \in S_M : t_i = P(X_0 = i)$*So,* $\boldsymbol{t} \in \mathcal{M}_{\mathbb{R}}[1, M]$. *Then, the marginal distribution of* X_n *is given by the product of the vector* $\boldsymbol{t}Q^n \in \mathcal{M}_{\mathbb{R}}[1, M]$. *That is, the* j^{th} *component of that vector is* $P(X_n = j)$ *for any* $j \in S_M$. *We may call* $\boldsymbol{t}$ *an initial state distribution.*

A.3.1 Classification of states

- A state $i \in S_M$ is said to be **recurrent** if starting from i, the probability is 1 that the chain will *eventually* return to i. If the chain is not recurrent, it is **transient**, meaning that if it starts at i, there is a non-zero probability that it never returns to i.

- Caveat: As we let $n \to \infty$, our Markov chain will gurantee that all transient states will be left forever, no matter how small the probability is. This can be proven by letting the probability be some ε, then realizing that by the support of Geom(ε) is always some finite value, then the equivalence between the Markov property and independent Geometric trials gurantees the existence of some finite value such that there is a success of never returning to i.

Definition A.3.6 (Reducibility)**.** *A Markov chain is said to be* ***irreducible*** *if for any* $i, j \in S_M$*, it is possible to go from* $i \to j$ *in a finite number of steps with positive probability. In other words:*

$$\forall i, j \in S_M : \exists n \in \mathbb{N} : q_{ij}^{(n)} > 0$$

- From our quantifier formulation of irreducible Markov chains, note that we can equivalently say that a chain is irreducible if there is integer $n \in \mathbb{N}$ such that the (i, j) entry of Q^n is positive for any i, j.

- A Markov chain is **reducible** if it is not **irreducible**. Using our quantifier formulation, it means that it suffices to find transient states so that:

$$\exists i, j \in S_M : \nexists n \in \mathbb{N} : q_{ij}^{(n)} > 0$$

Appendix B

Algorithm Appendix

B.1 Implicit $\mathcal{D}$-Matrices

Algorithm B.1.1 (Stochastic Row)**.**

1. *To sample a row of size N, fix $N \in \mathbb{N}$.*
2. *Sample a vector X with N entries between $[0, 1]$. So, generate $X = (x_i)_{i=1}^{N}$ where $x_i \sim$ Unif$(0, 1)$.*
3. *Normalize the rows by diving each entry by the row sum; so assign $x_i \leftarrow x_i \cdot \frac{1}{\sum x_i}$*
4. *Return the stochastic row X.*

Algorithm B.1.2 (Stochastic Matrix)**.**

1. *To generate a stochastic square matix P of size N, fix $N \in \mathbb{N}$.*
2. *Then, for every row of P, randomly sample a stochastic row and assign it to P.*
3. *Return the stochastic matrix P.*

Algorithm B.1.3 (Symmetric Stochastic Matrix)**.**

1. *To sample a symmetric stochastic matrix P of size N, fix $N \in \mathbb{N}$.*
2. *Sample a random stochastic matrix Q of size N.*
3. *Choosing one of the triangles of Q, set both the upper and lower of triangles of P to be that triangle.*
4. *Set the diagonal of the matrix P to be equal to 1 minus the sum of the non-diagonal entries.*
5. *Return the symmetric stochastic matrix P.*

Algorithm B.1.4 (Transition Matrix of an Erdos-Renyi Graph)**.**

1. *Fix* $N \in \mathbb{N}$ *and* $p \in [0,1]$*.*
2. *Generate a matrix* Q *such that every entry* $i, j \in 1, \ldots, N$ *is* $x_{ij} \sim \mathit{Unif}(0,1)$*.*
3. *For each* v_i *in* $\{1, \ldots, N\}$*, generate* $\mathit{deg}(v_i) \sim \mathit{Bin}(N,p)$*.*
4. *Randomly chose* $1 - \mathit{deg}(v_i)$ *vertices, set the entries* x_{ij} *in the* j *columns to 0.*
5. *Renormalize the matrix by dividing each row by its sum; let* $(x_i) \leftarrow (x_i)/\sum_j (x_i)$*.*

B.2 Explicit $\mathcal{D}$-Matrices

Algorithm B.2.1 (Explicit $\mathcal{D}$-Matrix)**.**

1. *To simulate a* $\mathcal{D}$*-distributed square matrix* P *of size* N*, fix* $N \in \mathbb{N}$*.*
2. *Sample a vector* X *with* N *entries from* $\mathcal{D}$*. So, generate* $X = (x_i)_{i=1}^N$ *where* $x_i \sim \mathcal{D}$*.*
3. *Assign the vector* X *as a row of the matrix* P*. Repeat for every other row.*
4. *Return the* $\mathcal{D}$*-distributed matrix* P*.*

Algorithm B.2.2 (Dimitriu's Beta Matrix)**.**

1. *To simulate an* $N \times N$ *beta matrix, fix* $N \in \mathbb{N}$*.*
2. *Start by taking a diagonal of* $\mathcal{N}(0,2)$ *variables.*
3. *Set both of the nearest off-diagonals to the row that samples from a* $\chi(\mathit{df} = c_j)$ *where* $c_j = \beta \cdot j$ *for columns spanning* $j = 1, \ldots, n-1$*.*

Appendix C

Code Appendix

The RMAT Package

The RMAT package is composed of two primary modules, the random matrix module (matrices.R) and the spectral statistics module (spectrum.R) and (dispersion.R). They can be further subdivided into smaller modules as follows.

1. **Random Matrix Module**
   (a) Explicitly Distributed Matrices
   (b) Implicitly Distributed Matrices
   (c) Ensemble Extensions
2. **Spectral Statistics Module**
   (a) Spectrum
   (b) Dispersions
   (c) Parallel Extensions
   (d) Visualizations

Matrices

Random matrices can either be explicitly or implicitly distributed. If they are explicitly distributed, their entries have a specific distribution. Otherwise, the entries have an implicit distribution imposed by generative algorithm the matrix uses.

Explicitly Distributed

For (homogenous) explicitly distributed matrices, we can use a "function factory" method to be concise. The actual implementation is more verbose for the purposes of argument documentation, but the following code is minimal and fully functional. Additionally, there are the beta matrices, which use the matrix model provided by the algorithm in Dimitriu's paper.

Homogenously Distributed

```
# ... represents all the arguments taken in by the rdist function
RM_explicit <- function(rdist){
  function(N, ..., symm = FALSE){
    # Create an [N x N] matrix sampling the rows from rdist, passing ... to rdist
    P <- matrix(rdist(N^2, ...), nrow = N)
    # Make symmetric if prompted
    if(symm){P <- .makeHermitian(P)}
    # Return P
    P
  }
}

# A version where we add an imaginary component
RM_explicit_cplx <- function(rdist){
  RM_dist <- function(N, ..., symm = FALSE, cplx = FALSE, herm = FALSE){
    # Create an [N x N] matrix sampling the rows from rdist, passing ... to rdist
    P <- matrix(rdist(N^2, ...), nrow = N)
    # Make symmetric/hermitian if prompted
    if(symm || herm){P <- .makeHermitian(P)}
    # Returns a matrix with complex (and hermitian) entries if prompted
    if(cplx){
      # Recursively add imaginary components as 1i * instance of real-valued matrix.
      Im_P <- (1i * RM_dist(N, ...))
      # Make imaginary part Hermitian if prompted
      if(herm){P <- P + .makeHermitian(Im_P)}
      else{P <- P + Im_P}
    }
    P # Return the matrix
  }
}
```

With our function factories set up, we can quickly generate all the random matrix functions for all the distributions our hearts could desire.

```
RM_unif <- RM_explicit_cplx(runif)
RM_norm <- RM_explicit_cplx(rnorm)
```

Beta Matrices

For the β-ensemble matrices, we simply use the algorithm provided in Dimitriu's paper. Doing so, we get the function:

```r
RM_beta <- function(N, beta){
  # Set the diagonal as a N(0,2) distributed row.
  P <- diag(rnorm(N, mean = 0, sd = sqrt(2)))
  # Set the off-1 diagonals as chi squared variables with df(beta), as given in Dumitriu's model
  df_seq <- beta*(N - seq(1, N-1)) # Get degrees of freedom sequence for offdigonal
  P[row(P) - col(P) == 1] <- P[row(P) - col(P) == -1] <- sqrt(rchisq(N-1, df_seq)) # Generate tridiagon
  P <- P/sqrt(2) # Rescale the entries by 1/sqrt(2)
  P # Return the matrix
}
```

Implicitly Distributed

In the case of implicitly distributed matrices, we have various types of stochastic matrices.

Stochastic Matrices

For stochastic matrices, we require slightly more setup. First, we setup the row functions to sample probability vectors:

```
# Generates stochastic rows of size N
.stoch_row <- function(N){
  row <- runif(N,0,1) # Sample probability distribution
  row/sum(row) # Return normalized row
}
```

For random introduced sparsity, we define the following row function.

```
# Generates same rows as in r_stoch(N), but with introduced random sparsity
.stoch_row_zeros <- function(N){
  row <- runif(N,0,1)
  degree_vertex <- sample(1:(N-1), size = 1) # Sample a degree of at least 1, as to ensure row is stoch
  row[sample(1:N, size = degree_vertex)] <- 0 # Choose edges to sever and sever them
  row/sum(row) # Return normalized row
}
```

Once this is done, we can use this function iteratively. With some magic, we can incorporate an option to make the matrix symmetric, and we get the following function.

```
RM_stoch <- function(N, symm = F, sparsity = F){
  if(sparsity){row_fxn <- .stoch_row_zeros} else {row_fxn <- .stoch_row} # Choose row function
  # Generate the [N x N] stochastic matrix stacking N stochastic rows (using the chosen function)
  P <- do.call("rbind", lapply(X = rep(N, N), FUN = row_fxn))
  if(symm){ # Make symmetric (if prompted)
    P <- .makeHermitian(P) # Make lower and upper triangles equal to each other's conjugate transpose
    diag(P) <- rep(0, N) # Nullify diagonal
    for(i in 1:N){P[i, ] <- P[i, ]/sum(P[i, ])} # Normalize rows
    # Set diagonal to the diff. between 1 and the non-diagonal entry sums such that rows sum to 1
    diag <- vector("numeric", N)
    for(i in 1:N){diag[i] <- (1 - sum(.offdiagonalEntries(row = P[i, ], row_index = i)))}
    diag(P) <- diag
  }
  P # Return the matrix
}
```

Erdos-Renyi Matrices

For the Erdos-Renyi walks, we do something similar by defining a parameterized row function.

```
# Generates a stochastic row with parameterized sparsity of p
.stoch_row_erdos <- function(N, p){
  row <- runif(N,0,1) # Generate a uniform row of probabilites
  degree_vertex <- rbinom(1, N, 1-p) # Sample number of zeros so that degree of row/vertex i ~ Bin(n,p)
  row[sample(1:N, degree_vertex)] <- 0 # Choose edges to sever and sever them
  if(sum(row) != 0){row/sum(row)} else{row} # Return normalized row only if non-zero (cannot divide by
}
```

And we again use the row function iteratively to get the following function.

```r
RM_erdos <- function(N, p, stoch = T){
  # Generate an [N x N] Erdos-Renyi walk stochastic matrix by stacking N p-stochastic rows
  P <- do.call("rbind", lapply(X = rep(N, N), FUN = .stoch_row_erdos, p = p))
  # If the matrix is to be truly stochastic, map rows with all zeros to have diagonal entry 1
  if(stoch){
    # Set diagonal to ensure that rows sum to 1
    diag <- rep(0, N)
    for(i in 1:N){diag[i] <- (1 - sum(.offdiagonalEntries(row = P[i, ], row_index = i)))}
    diag(P) <- diag
  }
  P # Return the matrix
}
```

And as such, we have minimal, functional implementations of functions that sample random matrices! In total, we only needed two helper functions. The `.offdiagonalEntries` function was used to normalize the probabilities in `RM_stoch` and `RM_erdos`.

```r
# Manually make equate the entries in the upper triangle to the conjugate of those in the lower triangl
.makeHermitian <- function(P){
  # Run over entry of the matrix
  for(i in 1:nrow(P)){
    for(j in 1:ncol(P)){
      # Restrict view to one of the triangles (i < j): Lower Triangle
      if(i < j){P[i,j] <- Conj(P[j,i])} # Equalize lower and upper triangles, making conjugate if compl
    }
  }
  P # Return Hermitian Matrix
}

# Return the off-diagonal entries of row i
.offdiagonalEntries <- function(row, row_index){row[which(1:length(row) != row_index)]}
```

Ensemble Extensions

Lastly, we have the ensemble extensions. These functions are quite simple to implement user a "function factory' '. Again, the actual implementations are more verbose due to the argument descriptions, but otherwise, are exactly the same.

```
RME_extender <- function(RM_dist){
  # Function returns a list of replicates of the RM_dist function with ... as its arguments
  function(N, ..., size){
    lapply(X = rep(N, size), FUN = RM_dist, ...)
  }
}
```

Now, we extend the functions as follows, and we are done with the matrix module!

```
RME_unif <- RME_extender(RM_unif)
RME_norm <- RME_extender(RM_norm)
RME_beta <- RME_extender(RM_beta)
RME_stoch <- RME_extender(RM_stoch)
RME_erdos <- RME_extender(RM_erdos)
```

Spectral Statistics

Spectrum

```r
spectrum <- function(array, components = TRUE, sort_norms = TRUE, singular = FALSE, order = NA){
  digits <- 4 # Digits to round values to
  # Array is a matrix; call function returning eigenvalues for singleton matrix
  if(class(array) == "matrix"){
    .spectrum_matrix(array, components, sort_norms, singular, order, digits)
  }
  # Array is an ensemble; recursively row binding each matrix's eigenvalues
  else if(class(array) == "list"){
    purrr::map_dfr(array, .spectrum_matrix, components, sort_norms, singular, order, digits)
  }
}
```

```r
# Helper function returning tidied eigenvalue array for a matrix
.spectrum_matrix <- function(P, components, sort_norms, singular, order, digits = 4){
  # If prompted for singular values, then take the product of the matrix and its tranpose instead
  if(singular){P <- P %*% t(P)}
  # Get the sorted eigenvalue spectrum of the matrix
  eigenvalues <- eigen(P)$values # Compute the eigenvalues of P
  if(singular){eigenvalues <- sqrt(eigenvalues)} # Take the square root of the eigenvalues
  if(sort_norms){eigenvalues <- .sort_by_norm(eigenvalues)} # Order the eigenvalue spectrum by norm rat
  else{eigenvalues <- sort(eigenvalues, decreasing = TRUE)} # Else, sort by sign.
  # If uninitialized, get eigenvalues of all orders; Otherwise, concatenate so single inputs become vec
  if(class(order) == "logical"){order <- 1:nrow(P)} else{order <- c(order)}
  purrr::map_dfr(order, .resolve_eigenvalue, eigenvalues, components, digits) # Get the eigenvalues
}
```

```r
# Read and parse an eigenvalue from an eigen(P)$value array
.resolve_eigenvalue <- function(order, eigenvalues, components, digits){
  eigenvalue <- eigenvalues[order] # Read from eigen(P)$values
  # Get norm and order columns (will unconditionally be returned)
  norm_and_order <- data.frame(Norm = abs(eigenvalue), Order = order)
  # If components are requested, resolve parts into seperate columns and cbind to norm and order
  if(components){evalue <- cbind(data.frame(Re = Re(eigenvalue), Im = Im(eigenvalue)), norm_and_order)}
  else{evalue <- cbind(data.frame(Eigenvalue = eigenvalue), norm_and_order)}
  evalue <- round(evalue, digits) # Round entries
  evalue # Return resolved eigenvalue
}
```

Helper Functions

```r
# Sort an array of numbers by their norm (written for eigenvalue sorting)
.sort_by_norm <- function(eigenvalues){
  (data.frame(eigenvalue = eigenvalues, norm = abs(eigenvalues)) %>% arrange(desc(norm)))$eigenvalue
  }
```

Dispersions

```
dispersion <- function(array, pairs = NA, sort_norms = TRUE, singular = FALSE, norm_pow = 1){ #sortNorm
  digits <- 2 # Digits to round values to
  pairs <- .parsePairs(pairs, array) # Parse input and generate pair scheme (default NA), passing on ar
  # Array is a matrix; call function returning dispersion for singleton matrix
  if(class(array) == "matrix"){
    .dispersion_matrix(array, pairs, sort_norms, singular, norm_pow, digits)
  }
  # Array is an ensemble; recursively row binding each matrix's dispersions
  else if(class(array) == "list"){
    purrr::map_dfr(array, .dispersion_matrix, pairs, sort_norms, singular, norm_pow, digits)
  }
}
```

```
# Find the eigenvalue dispersions for a given matrix
.dispersion_matrix <- function(P, pairs, sort_norms, singular, norm_pow, digits = 4){
  eigenvalues <- spectrum(P, sort_norms = sort_norms, singular = singular) # Get the sorted eigenvalues
  norm_fn <- function(x){(abs(x))^norm_pow} # Generate norm function to pass along as argument (Euclide
  purrr::map2_dfr(pairs[,1], pairs[,2], .resolve_dispersion, eigenvalues, norm_fn, digits) # Evaluate t
}
```

```
# Read and parse a dispersion observation between eigenvalue i and j.
.resolve_dispersion <- function(i, j, eigenvalues, norm_fn, digits){
  ## Copmute dispersion metrics
  disp <- data.frame(i = i, j = j) # Initialize dispersion dataframe by adding order of eigenvalues com
  disp$eig_i <- .read_eigenvalue(i, eigenvalues); disp$eig_j <- .read_eigenvalue(j, eigenvalues) # Add
  disp$id_diff <- disp$eig_j - disp$eig_i # Get the identity difference dispersion metric
  ## Compute norm metrics
  disp$id_diff_norm <- norm_fn(disp$id_diff) # Take the norm of the difference
  disp$abs_diff <- norm_fn(disp$eig_j) - norm_fn(disp$eig_i) # Compute the difference of absolutes w.r.
  ## Prepare for return
  disp <- round(disp, digits) # Round digits
  disp$diff_ij <- disp$i - disp$j
  disp # Return resolved dispersion observation
}
```

Helper Functions

Warning There is a bug in this code chunk regarding coercion of complex numbers into numericals.

```
# Parses a matrix spectrum array for the eigenvalue at a given order as cplx type (for arithmetic)
.read_eigenvalue <- function(order, mat_spectrum){
  if(ncol(mat_spectrum) == 3){mat_spectrum[order, 1]} # If the components are not resolved, return valu
  else{ # Components are resolved; get components and make it a complex number for arithmetic prep
    evalue <- complex(real = mat_spectrum[order, 1], imaginary = mat_spectrum[order, 2])
    #if(Im(evalue) != 0){evalue} else{as.numeric(evalue)} # If it is real, coerce it into a numeric to
  }
}
```

Pairing Schema

```r
# Parse a string argument for which pairing scheme to utilize
.parsePairs <- function(pairs, array){
  valid_schemes <- c("largest", "lower", "upper", "consecutive", "all") # Valid schemes for printing if
  # Obtain the matrix by inferring array type; if ensemble take first matrix
  if(class(array) == "list"){P <- array[[1]]} else{P <- array}
  if(class(pairs) == "logical"){pairs <- "consecutive"} # Set default value to be the consecutive pair
  # Stop function call if the argument is invalid
  if(!(pairs %in% valid_schemes)){stop(paste("Invalid pair scheme. Try one of the following: ",paste(va
  # Obtain the dimension of the matrix
  N <- nrow(P)
  # Parse the pair string and evaluate the pair scheme
  if(pairs == "largest"){pair_scheme <- data.frame(i = 2, j = 1)}
  else if(pairs == "consecutive"){pair_scheme <- .consecutive_pairs(N)}
  else if(pairs == "lower"){pair_scheme <- .unique_pairs_lower(N)}
  else if(pairs == "upper"){pair_scheme <- .unique_pairs_upper(N)}
  else if(pairs == "all"){pair_scheme <- .all_pairs(N)}
  pair_scheme # Return pair scheme
}
```

```r
# The antisymmetric pair scheme (for assymetric dispersion metrics); essentially all the permutations
.all_pairs <- function(N){
  purrr::map_dfr(1:N, function(i, N){data.frame(i = rep(i, N), j = 1:N)}, N)
}

# The consecutive-value scheme (Sufficient such that no linear combiantions of the diseprsion metric ex
.consecutive_pairs <- function(N){
  purrr::map_dfr(2:N, function(i){data.frame(i = i, j = i - 1)})
}

# The triangular pair schema (for symmetric dispersion metrics); essentially all the combinations
# (Enumerate the pair combinations given N items with i > j)
.unique_pairs_lower <- function(N){
  is <- do.call("c", purrr::map(1:N, function(i){rep(i,N)}))
  js <- rep(1:N, N)
  do.call("rbind",purrr::map2(is, js, .f = function(i, j){if(i > j){c(i = i, j = j)}}))
}
# The triangular pair schema (for symmetric dispersion metrics); essentially all the combinations
# (Enumerate the pair combinations given N items with i < j)
.unique_pairs_upper <- function(N){
  is <- do.call("c", purrr::map(1:N, function(i){rep(i,N)}))
  js <- rep(1:N, N)
  do.call("rbind",purrr::map2(is, js, .f = function(i, j){if(i < j){c(i = i, j = j)}}))
}
```

Parallel Extensions

Spectrum

```
spectrum_parallel <- function(array, components = TRUE, sort_norms = TRUE, singular = FALSE, order = NA
  digits <- 4 # Digits to round values to
  # Array is a matrix; call function returning eigenvalues for singleton matrix
  if(class(array) == "matrix"){
    .spectrum_matrix(array, components, sort_norms, singular, order, digits)
  }
  # Array is an ensemble; recursively row binding each matrix's eigenvalues
  else if(class(array) == "list"){
    furrr::future_map_dfr(array, .spectrum_matrix, components, sort_norms, singular, order, digits)
  }
}
```

Dispersion

```
dispersion_parallel <- function(array, pairs = NA, sort_norms = TRUE, singular = FALSE, norm_pow = 1){
  digits <- 4 # Digits to round values to
  pairs <- .parsePairs(pairs, array) # Parse input and generate pair scheme (default NA), passing on ar
  # Array is a matrix; call function returning dispersion for singleton matrix
  if(class(array) == "matrix"){
    .dispersion_matrix(array, pairs, sort_norms, singular, norm_pow, digits)
  }
  # Array is an ensemble; recursively row binding each matrix's dispersions
  else if(class(array) == "list"){
    furrr::future_map_dfr(array, .dispersion_matrix, pairs, sort_norms, singular, norm_pow, digits)
  }
}
```

Visualization Functions

Spectrum Visualization

```
spectrum.scatterplot <- function(array, ..., mat_str = ""){
  # Process spectrum of the matrix/ensemble
  if(class(array) == "list" || class(array) == "matrix"){
    array_spectrum <- spectrum(array, ...)
  }
  # Else, the array is a precomputed spectrum (avoid computational waste for multiple visualizations)
  else{
    array_spectrum <- array
  }
  # Infer plot title string from which type of array (matrix/ensemble)
  title_str <- .plot_title(class(array), prefix = "Spectrum", mat_str)
  # Plot parameters
  order <- array_spectrum[["Order"]]
  # Plot
  array_spectrum %>%
    ggplot() +
    geom_point(mapping = aes(x = Re, y = Im, color = order), alpha = 0.75) +
    scale_color_continuous(type = "viridis") +
    labs(x = "Re", y = "Im", title = paste(title_str,sep = "")) +
    coord_fixed()
}
spectrum.histogram <- function(array, ..., component = NA, bins = 100, mat_str = ""){
  # Process spectrum of the matrix/ensemble
  if(class(array) == "list" || class(array) == "matrix"){array_spectrum <- spectrum(array, ...)}
  else{array_spectrum <- array} # Else, the array is a precomputed spectrum (avoid computational waste
  # Infer plot title string from which type of array (matrix/ensemble)
  title_str <- .plot_title(class(array), prefix = "Spectrum", mat_str)
  # Plot parameters
  color0 <- "mediumpurple3"
  num_entries <- nrow(array) # Get number of entries to normalize
  if(class(component) == "logical"){component <- c("Re", "Im")} # Set default to both components
  # Plot lambda function
  component_plot <- function(component){
    # Plot parameters
    component_str <- paste(" (",component,")",collapse = "")
    # Plot
    array_spectrum %>%
      ggplot() +
      geom_histogram(mapping = aes_string(x = component), fill = color0, bins = bins) +
      labs(x = component, y = "Frequency", title = paste(title_str, component_str, sep = ""))
  }
  # Get list of plots
  plots <- purrr::map(component, component_plot)
  # If we have both components and patchwork is loaded, attach plots to each other
  if(length(plots) == 2){plots[[1]] / plots[[2]]} else if(length(plots) == 1){plots[[1]]}
  # Return the list of plots
  else{plots}
}

# Helper function returning appoporiate string for matrix/ensemble given a matrix type string and class
```

```r
.plot_title <- function(array_class, prefix, mat_str){
  if(mat_str != ""){pre_space <- " "} else{pre_space <- ""} # Format without given name
  # Infer plot title string from which type of array (matrix/ensemble)
  if(array_class == "matrix"){plot_str <- paste(pre_space, mat_str," Matrix", sep = "", collapse = "")}
  else if(array_class == "list"){plot_str <- paste(pre_space, mat_str," Matrix Ensemble", sep = "", col
  else{plot_str <- paste(pre_space, mat_str," Matrix Ensemble", sep = "", collapse = "")}
  paste(prefix," of a",plot_str, sep = "")
}
```

```r
order.scatterplot <- function(spectrum, component){
  spectrum %>%
    ggplot(aes(x = Order, y = {{ component }}, color = Order)) +
    geom_point() +
    scale_color_viridis_c() +
    theme(legend.position = "bottom")
}
order.density <- function(spectrum, component){
  spectrum %>%
    ggplot(mapping = aes(group = Order, x = {{ component }}, color = Order)) +
    geom_density() +
    scale_color_viridis_c() +
    theme(legend.position = "bottom")
}
order.summary <- function(spectrum, component){
  spectrum %>%
    group_by(Order) %>%
    summarize(
      Mean_Re = mean(Re), Mean_Im = mean(Im), Mean_Norm = mean(Norm),
      Variance_Re = var(Re), Variance_Im = var(Im), Variance_Norm = var(Norm)) %>%
    ggplot(mapping = aes(y = {{ component }}, x = Order, color = Order)) +
    geom_point() +
    geom_line() +
    scale_color_viridis_c() +
    theme(legend.position = "bottom")
}
```

Dispersion Visualization

```r
dispersion.histogram <- function(array, metric = NA, ..., bins = 100){
  valid_schemes <- c("id_diff","id_diff_norm","abs_diff") # Valid schemes for printing if user is unawa
  if(class(metric) == "logical"){stop("Please input a valid dispersion metric. Try one of the following
  # Process spectrum of the matrix/ensemble
  if(class(array) == "list" || class(array) == "matrix"){ disps_df <- dispersion(array, ...) }
  else{disps_df <- array} # Otherwise, the array is a precomputed dispersion dataframe
  num_entries <- nrow(disps_df) # Get number of entries
  # Plot parameters
  color0 <- "darkorchid4"
  # Return plot
  ggplot(data = disps_df, mapping = aes_string(x = metric)) +
    geom_histogram(mapping = aes(y = stat(count / num_entries)), bins = bins) +
    labs(title = "Distribution of Eigenvalue Spacings", y = "Probability")
}
```

```r
dispersion.scatterplot <- function(array, metric = "id_diff_norm", pairs = NA, ...){
  # Process dispersion of the matrix/ensemble; if array is a dispersion data frame, copy it.
  if(class(array) == "list" || class(array) == "matrix"){disps_df <- dispersion(array, pairs, ...)}
  else{disps_df <- array} # Otherwise, the array is a precomputed dispersion dataframe
  # Parse plotting aesthetics from pairs. diff_ij is more useful unless pairs = "consecutive" or "large
  if(pairs %in% c("consecutive","largest")){order_stat <- "j"} else{order_stat <- "diff_ij"}

  # Plot parameters
  color0 <- "darkorchid4"
  real_valued <- T
  # Scatterplot of dispersion metric
  if(real_valued){
    disps_df %>%
      ggplot(mapping = aes_string(x = metric, y = order_stat, color = order_stat)) +
      geom_point() +
      scale_color_continuous(type = "viridis") +
      labs(title = "Distribution of Eigenvalue Spacings by Order Statistic")
  } else{
      resolved <- .resolve_eigenvalues(disps_df)
      resolved %>%
        ggplot() +
        geom_point()
    }
}
```

Appendix D

Mixing Time Simulations

D.1 Introduction

In this chapter, we'll talk about ratio-mixing time simulations. Essentially, these simulations are a method of approximating the distribution of a random transition matrix's mixing time. There will be a fun exploration of the Erdos-Renyi matrix ensembles and we will computationally show that the parameterized ensemble has a mixing time inversely proportional to graph sparsity.

D.2 Mixing Time Simulations

With the Erdos-Renyi graph defined, we may now motivate the simulation of random walks on them. First, however, we need to generate their corrosponding transition matrices. An algorithm for this is outlined below.

Suppose we have simulated a transition matrix for an Erdos-Renyi graph called Q. Now, fixing some initial probability distribution $\vec{x} \in \mathbb{R}^M$, we may consider the evolution sequence of a random walk on this Erdos-Renyi graph by taking its evolution sequence $\mathcal{S}(Q, x)$.

Definition D.2.1 (Random Batches)**.** *Let $\mathbb{F}$ be a field, and fix some $M \in \mathbb{N}$. Let $\mathcal{B}_\lambda \subset \mathbb{F}^M$ be a uniformly random batch of points in the M-hypercube of length λ. That is,*

$$\mathcal{B}_\lambda = \{\vec{x} \mid x_i \sim \mathit{Unif}(-\lambda, \lambda) \text{ for } i = 1, \ldots, M\}$$

Note: *If $\mathbb{F} = \mathbb{C}$, then take $\vec{x} \in \mathcal{B}_\lambda$ to mean $\vec{x} = a + bi$ where $a, b \sim \mathit{Unif}(-\lambda, \lambda)$.*

Definition D.2.2 (Evolution Sequence)**.** *An evolution sequence of a vector $\vec{\pi}$ and a transition matrix Q is defined as the sequence $\mathcal{S}(Q, \pi) = (\pi'_n)_{n=1}^N$ where $\pi'_n = \pi Q^n$*

Definition D.2.3 (Finite Evolution Sequences)**.** *Suppose we sample a random point from $\mathcal{B}_\lambda$, emulating a random point $\vec{v} \in \mathbb{F}^M$. Additionally, let $Q \in \mathbb{F}^{M \times M}$ be a transition matrix over $\mathbb{F}$. Fixing a maximum power ('time') $T \in \mathbb{N}$, define the evolution sequence of $\vec{v}$ as follows:*

$$\mathcal{S}(v, Q, T) = (\alpha_n)_{n=1}^T \text{ where } \alpha_k = vQ^k$$

If we do not impose a finiteness constraint on the sequence, we consider powers for $n \in \mathbb{N}$ or $t = \infty$

Definition D.2.4 (Consecutive Ratio Sequences)**.** *Accordingly, define the consecutive ratio sequence (CST) of $\vec{v}$ as follows:*

$$\mathcal{R}(v, Q, T) = (r_n)_{n=2}^{T} \text{ where } (r_n)_j = \frac{(\alpha_n)_j}{(\alpha_{n-1})_j} \text{ for } j = 1, \ldots, M$$

*In other words, the consecutive ratio sequence of v can be obtained by performing **component-wise division** on consecutive elements of the evolution sequence of v.*

Definition D.2.5 (Near Convergence)**.** *Because these sequences may never truly converge to eigenvectors of the matrix, we formalize a notion of "near convergence". As a prelemenary, we first define ε-equivalence. Let $\mathbb{F}$ be a field, and fix $\varepsilon \in \mathbb{R}^+$. Suppose we have vectors $v, v' \in \mathbb{F}^M$. Then, $v \sim_\varepsilon v'$ if $||v - v'|| < \varepsilon$ where $|| \cdot ||$ is the norm on $\mathbb{F}$.*

Let $\varepsilon \in \mathbb{R}^+$, and suppose we have an evolution sequence $(a[\vec{v}])_n$. Then, a_n ε-converges at $N \in \mathbb{N}$ if:

$$\forall n \geq N \mid a_N \sim_\varepsilon a_n$$

D.3 Erdos-Renyi Ensemble Simulations

Definition D.3.1 (Erdos-Renyi Graph)**.** *An Erdos-Renyi graph is a graph $G = (V, E)$ with a set of vertices $V = 1, \ldots, M$ and edges $E = \mathbb{1}_{i,j \in V} \sim Bern(p_{ij})$. It is homogenous if $p_{ij} = p$ is fixed for all i, j.*

Essentially, an Erdos-Renyi graph is a graph whose 'connectedness' is parameterized by a probability p (assuming it's homogenous, which this document will unless otherwise noted). As $p \to 0$, we say that graph becomes more sparse; analogously, as $p \to 1$ the graph becomes more connected.

Recall from probability theory that a sum of i.i.d Bernoulli random variables is a Binomial variable. As such, we may alternatively say that the degree of each vertex v is distributed as $deg(v) \sim Bin(M, p)$. This is helpful to know because the process of simulating graphs becomes much simpler.

D.4 Questions

1. How are the entries of the CRS distributed? Are they normal, and if so, what is its mean?
2. Are the entries of the CRS i.i.d as $t \to \infty$?
3. For an Erdos-Renyi matrix, is the mixing time t dependent on the parameter p?
4. What impact does the running time parameter T have on σ (the variance of the distribution of the CRS entries)?

D.4.1 Cauchy Distributed Ratios

It seems to be the case that the **log-transformed** entries of the CRS are Cauchy distributed about $\log \lambda_1$ where $\lambda_1 = \max(\sigma(Q))$, the largest eigenvalue of Q. That is,

$$r_i \sim \text{Cauchy}(\ln \lambda_1) \text{ for } i = 1, \ldots, M$$