2015

High-Throughput Data Analysis: Application to Micronuclei Frequency and T-cell Receptor Sequencing

Mateusz Makowski
Virginia Commonwealth University

Follow this and additional works at: https://scholarscompass.vcu.edu/etd

Part of the Biostatistics Commons, Genetic Processes Commons, and the Other Immunology and Infectious Disease Commons

© The Author
High-Throughput Data Analysis: Application to Micronuclei Frequency and T-cell Receptor Sequencing

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at Virginia Commonwealth University.

by

Mateusz Makowski

B.S. Biology and Mathematics, Dickinson College, USA, 2007

Director: Kellie J. Archer, Ph.D., Professor, Department of Biostatistics

Department of Biostatistics
Virginia Commonwealth University
Richmond, Virginia, USA
August, 2015
Acknowledgements

I would like to thank my adviser, Dr. Kellie J. Archer, for her patience in guiding me through the process of my doctoral work. She has been a wealth of information and support and I could not have finished without her. I would like to thank my committee members Dr. Nitai Mukhopadhyay and Dr. Roy Sabo for their time, patience and feedback related to statistics. I would like to thank Dr. Amir Toor and Dr. Masoud Manjili for serving on my committee, providing data for my dissertation and most importantly providing background on the biological aspects of my dissertation.

I also would like to thank my peers in the Department of Biostatistics, we learned as much from each other as from our instructors. I would like to thank the administration of the department for helping guide me through the process as well.

I would like to thank my friends in Richmond, especially the tennis community, for keeping me from doing nothing but sitting in front of my computer. I would like to thank my undergraduate roommates, Andy and Andre, who have always managed to stay in touch despite the many miles separating us. They have always been a shoulder to lean on.

Lastly, but most importantly, I want to thank my family. First I would like to thank my mother, Urszula, for her love and support during this long process. I would also like to thank my dad, Sam, and brother, Marty, for being supportive as well. I would also like to thank all my extended family that always showed interest in my research and progress. Finally, I would like to thank my wife, Maunette, for her keeping me sane during this process and always being there for me. Thank you for being by my side for the last 11 years and the many years to come.
Contents

1 T-Cell Receptors and the Adaptive Immune System 1
   1.1 T-cell Biology .................................................. 1
   1.2 Technology for Measuring TCRβ Clones ......................... 3
   1.3 ImmunoSEQ Precision, Accuracy and Sensitivity ............... 5
   1.4 Unseen Species and Estimating the TCRβ Repertoire using ImmunoSEQ data
      1.4.1 Unseen Species Problem ................................ 6
      1.4.2 TCRβ CDR3 Diversity ................................... 11
      1.4.3 Models for Estimating Population T-cell Diversity ....... 13
   1.5 Application of TCRβ Studies ................................... 15
      1.5.1 Hematopoietic stem cell transplantation .................. 15
      1.5.2 Case Studies .............................................. 15
   1.6 Overview of Statistical Methods for CDR3 Data ............... 17
      1.6.1 CDR3 Distribution Perturbation .......................... 18
      1.6.2 OligoScores .............................................. 19
      1.6.3 Simpson’s Diversity Index ................................ 20
      1.6.4 Kullback-Leibler Divergence Method ..................... 21
      1.6.5 Non-parametric Method .................................. 24
      1.6.6 Proportion Logit Transformation ........................... 25
      1.6.7 Tabular Summary of Statistical Methods for CDR3 data .. 26
      1.6.8 Multiple Comparisons ................................... 27
   1.7 Discussion ...................................................... 29

2 Applications to TCR sequencing data 30
   2.1 Introduction ..................................................... 30
   2.2 Reanalysis of Published TCR data .............................. 30
   2.3 Data ............................................................ 31
2.4 Permutation Tests ................................................................. 32
2.5 CDR3 Distribution Perturbation Results ................................. 33
2.6 Oligoscores Results ............................................................... 40
2.7 Simpson’s Diversity Index Results ........................................... 42
2.8 Kullback-Leibler Divergence Results ....................................... 47
2.9 Non-parametric Method Results ............................................. 49
2.10 Proportion Logit Transformation ............................................ 52
2.11 Discussion ............................................................................. 54

3 Compositional Data in Bioscience ............................................ 55
3.1 Introduction ............................................................................ 55
3.2 Definition of Compositional Data ........................................... 56
3.3 Representations of a Composition .......................................... 57
3.4 Statistical Modelling ............................................................... 59
3.5 Current Practices and Their Pitfalls ........................................ 61
3.6 Treatment of Zeros in Compositional Data ............................... 67
3.7 Application of Compositional Methods to TCR data ................ 69
3.8 Discussion ............................................................................. 71

4 Generalized monotone incremental forward stagewise method for modeling count data: Application predicting micronuclei frequency ......................................................... 73
4.1 Introduction ............................................................................ 73
4.1.1 Cytokinesis-block Micronucleus Assay ............................... 76
4.2 Methods ................................................................................. 77
4.2.1 Statistical Methods ............................................................ 77
4.2.2 Poisson Regression ............................................................ 77
4.2.3 Generalized Monotone Incremental Forward Stagewise Poisson Model ................................................................. 79
4.2.4 Comparative Method: Penalized Linear Regression ......... 81
List of Figures

1 VDJ Recombination: A schematic of how separate V,D and J segments are recombinated into a functional TCR protein (Adapted from Janeway et al. [2001]).

2 CDR3 Region: A diagram of where the CDR3 region occurs in a recombined VDJ segment.

3 TCR distributions: (A) Shows a typical Gaussian polyclonal distribution. (B) Shows a non-Gaussian distribution that is also polyclonal. (C) shows a non-Gaussian distribution that is monoclonal [Miqueu et al., 2007] with permission.

4 Spectratype Schematic Shows steps taken to run to generate spectratype data. (A) shows using electrophoresis to separate the mRNA by size. (B) shows measuring the band intensity. (C and D) show translating the band intensities into a distribution by size [Miqueu et al., 2007] with permission.

5 Sequencing Assay Strategy Schematic A generic TCRβ CDR3 PCR product is shown. Primers include parts of the V and J segments and also the Genome Analyzer Cluster station (GA F and GA R). The NDN region represents the CDR3 region. The primers were designed to capture just enough of the J and V segments to allow identification [Robins et al., 2012] with permission.
6 **Reproducibility Plots** The plots show frequencies of unique sequences from different samples. .................................................. 31

7 Distribution of perturbation from matched donor statistic for the 2 significantly different VDJ combinations. ........................................... 36

8 Distribution of perturbation from mean donor statistic for the 3 significantly different VDJ combinations. ................................................... 37

9 Distributions of CDR3 lengths from VDJ combination TRBJ2-6 TRBV9 TRBD1-2 donors. ................................................................. 38

10 Distributions of CDR3 lengths from VDJ combination TRBJ2-6 TRBV9 TRBD1-2 recipients. ................................................................. 39

11 Distribution of CDR3 length for highest Oligoscore in GVHD group. .......... 41

12 Distribution of CDR3 length for highest Oligoscore in No GVHD group. .......... 42

13 Distribution of Simpson’s Diversity Index statistic for the 5 significantly different VDJ combinations. ................................................ 43

14 Distribution of Shannon’s Diversity Index statistic for the 6 VDJ combinations having the smallest p-values. ........................................... 46

15 CDR3 length distribution for each patient of VDJ combination: TRBJ2-1, TRBV18, TRBD1-1. ................................................................. 48

16 CDR3 length distribution for each patient of VDJ combination: TRBJ1-2, TRBV3-1, TRBD1-1. ................................................................. 49

17 Distribution of the non-parametric \( \alpha \) statistic for the significantly different VDJ combination. ................................................................. 51

18 Distribution of the logit transformed proportion of shared sequences for the 2 VDJ combinations having the smallest p-values. ........................................... 53

19 Cluster Dendrogram of the 13 HSCT patients. ................................................................. 71
20 Number of Predictors Correctly Identified as Non-zero: This figure shows the
distribution of the number of predictors correctly identified as non-zero over
100 simulations. There were 5 predictors had non-zero coefficients. Boxplots
are separated by the type of distribution used to generate the data and number
of observations.

21 Number of Predictors Incorrectly Identified as Non-zero: This figure shows
the distribution of the number of predictors incorrectly identified as non-zero
over 100 simulations. There were 95 predictors that had coefficients set to
zero. Boxplots are separated by the type of distribution used to generate the
data and number of observations.

22 Accuracy of Count Predictions: This figure shows the distribution of the sum
of residuals squared over 100 simulations using a learning data set and an
independent test data set. Boxplots are separated by the type of distribution
used to generate the data and number of observations. The results for glmpath
with Gaussian family using an offset are not displayed because both values
are above 50,000.

23 Histogram of MN Counts for cord blood data set.

24 Plot of actual MN counts versus predicted MN counts using GMIFS.

25 Plot of actual MN counts versus predicted MN counts using glmpath.

26 Plot of actual MN counts versus predicted MN counts using glmpath using
17 non-zero coefficients.

27 Histogram of MN Counts for breast cancer data set.

28 Histogram of bud frequencies in the breast cancer data set.

29 Plot of actual MN counts versus predicted MN counts using GMIFS for breast
cancer data.
Abstract

High-Throughput Data Analysis: Application to Micronuclei Frequency and T-cell Receptor Sequencing

by Mateusz Makowski

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at Virginia Commonwealth University.

Virginia Commonwealth University, 2015

Major Director: Kellie J. Archer, Ph.D., Professor, Department of Biostatistics

The advent of high-throughput sequencing has brought about the creation of an unprecedented amount of research data. Analytical methodology has not been able to keep pace with the plethora of data being produced. Two assays, ImmunoSEQ and the cytokinesis-block micronucleus (CBMN), that both produce count data and have few methods available to analyze them are considered.

ImmunoSEQ is a sequencing assay that measures the β T-cell receptor (TCR) repertoire. The ImmunoSEQ assay was used to describe the TCR repertoires of patients’ that have undergone hematopoietic stem cell transplantation (HSCT). Several different methods for spectratype analysis were extended to the TCR sequencing setting then applied to these data to demonstrate different ways the data set can be analyzed. The different methods include CDR3 distribution perturbation, Oligoscores, Simpson’s diversity, Shannon diversity, Kullback-Leibler divergence, a non-parametric method and a proportion logit transformation method. Herein we also demonstrate adapting compositional data analysis methods to the TCR sequencing setting. The various methods were compared when analyzing a set of 13 subjects who underwent hematopoietic stem cell transplantation. The eight subjects who developed graft versus host disease were compared to the five who did not. There was no little overlap in the results of the different methods showing that researchers must choose the appropriate method for their research question of interest.
The CBMN assay measures the rate of micronuclei (MN) formation in a sample of cells and can be paired with gene expression or methylation assays to determine association between MN formation and other genetic markers. Herein we extended the generalized monotone incremental forward stagewise (GMIFS) method to the situation where the response is count data and there are more independent variables than there are samples. Our Poisson GMIFS method was compared to a popular alternative, glmpath, by using simulations and applying both to real data. Simulations showed that both methods perform similarly in accurately choosing truly significant variables. However, glmpath appears to overfit compared to our GMIFS method. Finally, when both methods were applied to two data sets GMIFS appeared to be more stable than glmpath.
T-Cell Receptors and the Adaptive Immune System

1.1 T-cell Biology

The adaptive immune system is responsible for recognizing foreign and self antigens and mounting the proper response. This is known as acquired immunity because the body “remembers” the specific antigen and can mount a quicker and more potent response each subsequent time the body encounters that antigen. T and B cells are responsible for maintaining an acquired immunity. T cells are a type of cell that maintains the body’s immunity by activating genes that create cell surface receptors that recognize foreign and self antigens. These genes and subsequently receptor proteins are passed on to future generations of cells during cell division. T-cell receptors (TCR) are heterodimers that contain an α and a β chain or a γ and a δ chain. The chains are made up of several different regions; a constant region, a variable region, a transmembrane domain and a short cytoplasmic tail. Further, the variable region of the β chain of TCR’s is made up of the variable (V), joining (J) and diversity (D) gene segments [Robins et al., 2009, Rudolph et al., 2006].

T-cells are activated when the TCR recognizes an antigen bound to a major histocompatibility complex (MHC) molecule presented to it by an antigen presenting cell (APC) [Avent, 2012]. Once the antigen is recognized an immune response is launched. The first step in a memory response is to have a TCR that recognizes a specific antigen. Because there is a large diversity of antigens the human body can come in contact with, the immune system must have a way to generate a large diversity of TCR’s. Producing a large diversity of TCR’s is accomplished by a process called VDJ recombination (Figure 1). VDJ recombination is the joining of different gene segments to create unique amino acid sequences. Specifically, V, D and J segments that are spread across human chromosomes 2, 14 and 22 are combined to make a unique VDJ combination.

In this review we are only interested in VDJ recombination of the β chain of TCR’s (TCRβ). There are 52 known V gene segments, 2 D gene segments and 13 J gene segments
Janeway et al., 2001 in the TCRβ chain. All possible VDJ combinations yield $13 \times 2 \times 52 = 1352$ possible combinations. This is not a large variety and further diversity in TCR’s is added by the amino acid sequence encoded in the third complementary-determining region (CDR3) Robins et al., 2009. The CDR3 region occurs at the VDJ junction. As can be seen in Figure 2, this junction includes part of the V segment, all of the D segment and the J segment. The CDR3 region undergoes template-independent additions and deletions of nucleotides during TCR gene rearrangement. This greatly increases TCRβ diversity. Arstila et al. 1999 reported estimates of $10^6$ unique TCRβ CDR3 sequences per person. By measuring the abundance of each clone a better understanding of the human immune response can be gained Carlson et al., 2013.

Figure 1: VDJ Recombination: A schematic of how separate V,D and J segments are recombined into a functional TCR protein (Adapted from Janeway et al., 2001).
Figure 2: CDR3 Region: A diagram of where the CDR3 region occurs in a recombined VDJ segment.

### 1.2 Technology for Measuring TCRβ Clones

Historically, measuring TCR abundance was done by spectratype analysis, a process that measures relative abundance of TCR clones. Unfortunately, spectratyping can only distinguish a particular V, D or J segment. For a specific V or J gene the clones are separated by size rather than sequence [Kepler et al., 2005]. This is useful in general terms to monitor TCR diversity. A more detailed review of spectratype analysis methodology can be found in section 1.6.

Figure 4 shows a schematic of how spectratyping assay is done. First a sample is collected and the T-cells are isolated. PCR is then used to replicate only a specific TCRβ gene segment (V, D or J) by using primers specific to a TCRβ segment. The mixture of CDR3 region clones is then separated by size using electrophoresis. The results are then presented on histograms with number of TCR’s at each specific size [Kepler et al., 2005]. Note that spectratyping separates CDR3 regions by size, if two different CDR3 sequences are of the same length they will be counted in the same bin. Figure 3 shows examples of some common CDR3 length distributions that result from spectratype analysis.

Recent improvements in sequencing technology allow for much greater detail in measuring TCR diversity [Robins et al., 2009]. Adaptive Biotechnologies developed an immune profiling assay, ImmunoSEQ, that uses parallel sequencing to capture 48 possible V segments, 13 J segments and 2 D segments by simultaneous use of PCR primers for each segment [Robins et al., 2009]. This allows for simultaneous sequencing of a large portion of the human T-cell
TCR distributions: (A) Shows a typical Gaussian polyclonal distribution. (B) Shows a non-Gaussian distribution that is also polyclonal. (C) shows a non-Gaussian distribution that is monoclonal [Miqueu et al., 2007] with permission.

The assay maximizes throughput by only amplifying and sequencing the region of the TCR that contains the CDR3 region. This allows for short reads, 60 nucleotides in length, to be used to identify each sequence. Further, the assay can be used on both cDNA and genomic DNA [Robins et al., 2009]. Recently, Adaptive Biotechnologies developed a new method to correct for PCR bias caused by using so many different primers [Carlson et al., 2013]. By using sequencing, researchers can distinguish TCRβ clones down to the level of unique CDR3 clones.

Figure 4: Spectratype Schematic Shows steps taken to run to generate spectratype data. (A) shows using electrophoresis to separate the mRNA by size. (B) shows measuring the band intensity. (C and D) show translating the band intensities into a distribution by size [Miqueu et al., 2007] with permission.
Figure 5 shows a schematic of a generic PCR product used by the ImmunoSEQ assay. 13 reverse J primers and 48 forward V primers were designed to capture $13 \times 2 \times 48 = 1248$ possible VDJ combinations. All sequences contain small portions of the V and J segments as well as the CDR3 region. This allows not only for gene segment family identification but also unique CDR3 identification.

**Figure 5: Sequencing Assay Strategy Schematic** A generic TCRβ CDR3 PCR product is shown. Primers include parts of the V and J segments and also the Genome Analyzer Cluster station (GA F and GA R). The NDN region represents the CDR3 region. The primers were designed to capture just enough of the J and V segments to allow identification [Robins et al., 2012] with permission.

1.3 ImmunoSEQ Precision, Accuracy and Sensitivity

[Robins et al. 2012] conducted experiments using the ImmunoSEQ platform to determine its precision, accuracy and specificity. Samples containing spiked-in clones of known concentration ranging from 10 to 100,000 cells per million were created. The experiments show that the assay can detect clones reliably at a level of one cell per 100,000.

The spike-in experiments were designed to test the assay’s accuracy and sensitivity. Different mixtures of five CD4+ T-cell clones specific for GAD65 were spiked into a background of one million CD4+ T cells. Three different mixtures were made using these five clones. Two other T-cell clones that targeted tumor-associated NY-ESO-1 antigen were also used. These were spiked directly into autologous peripheral blood mononuclear cells instead of separated cells. The titrations that were used were 1/100,000, 1/10,000, 1/1000 and 1/100. The different mixtures were then sequenced using the method described in section 1.2. The raw data results in counts for each unique CDR3 region. Finally, a nearest neighbor algorithm was used to merge closely related sequences to remove PCR and sequencing errors. The pro-
cessed data were then used to estimate the frequencies of each spiked in clone. The results show that 6 out of 7 of the clones had an observed frequency very close to the expected. The 7th clone was not detected at all because it contained a 16 nucleotide deletion and the assay is designed to capture all clones with deletions of 14 nucleotides or fewer (this deletion length accounts for 99% of TCRβ’s) [Robins et al., 2012]. Thus, the assay is extremely accurate in detecting clones even at low concentrations.

Finally, the reproducibility of the assay was tested. Blood was drawn from one subject and DNA was extracted two independent times (DNA_A and DNA_B). The DNA_A sample was then PCR amplified two independent times (PCR_A1 and PCR_A2) while the DNA_B sample was only PCR amplified once (PCR_B1). Finally, sample PCR_A1 was sequenced two independent times (SeqA1a and SeqA1b) while samples PCR_A2 and PCR_B1 were only sequenced once (SeqA2a and SeqB1a). [Robins et al., 2012] showed that sequencing from the the same PCR library is highly reproducible. [Robins et al., 2012] showed that sequencing from different PCR libraries is also reproducible but not quite as much. Finally, [Robins et al., 2012] showed that sequencing different DNA extractions is also reproducible but again not quite as much as the other two conditions. In the different blood draw samples, approximately 55% of all reads were obtained from shared sequences. This is probably due to a blood draw being a finite sample so each one is a random subset of all clones [Robins et al., 2012].

1.4 Unseen Species and Estimating the TCRβ Repertoire using ImmunoSEQ data

1.4.1 Unseen Species Problem

The unseen species problem is one that has been tackled several times over the years. [Efron and Thisted, 1976] described a method for estimating unseen species. The unseen species problem was first considered in ecology, where researchers needed to estimate the number
of different types of species in a given area assuming they could not sample every organism. That is, if a given area is sampled and \( n \) species are found, the unseen species problem is to estimate how many more unique species are in the area that were missed by the sample. A different application that may help elucidate this concept is to consider William Shakespeare’s vocabulary. Given all the different words he used in his writing, we may want to estimate how many words Shakespeare actually knew assuming he did not use all of them in his works. The unseen species estimation problem in this case is, if a new work from Shakespeare is found, to estimate the number of new words not used in his previous works. Shakespeare’s works are comprised of 884,647 total words, 31,534 of which are unique [Efron and Thisted, 1976]. Let \( n_x \) represent the number of words occurring \( x \) times (\( x = 1 \) to \( \infty \)). The method described by [Efron and Thisted, 1976] says, assuming the new works are the same size as the old, 11,460 new words would be expected. The basic model uses an empirical Bayes approach and is described later in this section. Recently, this method has been modified to genome sequencing [Ionita-Laza et al., 2009] and T-cell receptor sequencing [Robins et al., 2009].

The methodology described by [Efron and Thisted, 1976] can be thought about using species trapping terminology. Suppose there exist \( S \) species and after trapping for one unit of time we capture \( x_s \) members of species \( s \). We then assume that \( x_s \) has a Poisson distribution with a mean of \( \lambda_s \). It is convenient to consider the time period to be \([-1,0]\) and we wish to extrapolate to a time \( t \) in the future. Let \( x_s(t) \) be the number of times species \( s \) appears in \([-1,t]\).

Before moving on, it is of relevance to review the Poisson distribution. The Poisson distribution is a widely used discrete distribution for modeling count data. For example, the Poisson distribution can be used to model the number of occurrences of a phenomenon in a given time period [Casella and Berger, 2002]. In this case, the number of species seen in a given area in a given time period. The basic assumption the Poisson distribution is built on is that the probability of occurrence is proportional to length of waiting time. So the Poisson
distribution has one parameter $\lambda$, sometimes called the intensity parameter \cite{Casella and Berger, 2002}. A random variable, $X$ has a Poisson($\lambda$) distribution if

$$
P(X = x|\lambda) = \frac{e^{-\lambda} \lambda^x}{x!}, x = 0, 1, 2, ...$$

where the $\mathbb{E}(X) = \lambda$ and var($X$) = $\lambda$. Two useful characteristics of the Poisson distribution are that the sum of independent Poisson random variables is itself Poisson distributed and a Poisson random variable conditioned on a sum of Poisson variables is binomial. More formally,

**Theorem 1.** If $X \sim \text{Poisson}(\theta)$ and $Y \sim \text{Poisson}(\lambda)$ and $X$ and $Y$ are independent, then $X + Y \sim \text{Poisson}(\theta + \lambda)$ \cite{Casella and Berger, 2002}.

The Poisson distribution assumption implies, using Theorem 1, that $x_s(t)$ is distributed Poisson($\lambda_s(1+t)$). Furthermore, we can say that

$$
x_s|x_s(t) \sim \text{Binomial} \left( x_s(t), \frac{1}{1+t} \right)
$$

*Proof.* Assuming

$$
x_s \sim \text{Poisson}(\lambda_s)
$$

Then

$$
x_s(t) \sim \text{Poisson}(\lambda_s(1+t))
$$

given there are $1 + t$ time points in the $[-1, t]$ and by Theorem 1. Next to show $x_s|x_s(t)$ is binomially distributed, let $x_s(t) = x_s + x_s(t-1)$ where $x_s(t-1)$ does not include the interval $[-1, 0]$. Note that $\Pr(x_s(t-1) = x_s(t) - x_s | x_s) = \Pr(x_s(t-1) = x_s(t) - x_s)$ because $x_s(t-1)$ does not include the interval $[-1, 0]$ and therefore is not dependent on $x_s$. So we have
\[
\begin{align*}
\Pr(x_s \mid x_s(t)) &= \frac{\Pr(x_s, x_s(t) = x_s + x_s(t - 1))}{\Pr(x_s(t) = x_s + x_s(t - 1))} \\
&= \frac{\Pr(x_s) \Pr(x_s(t - 1) = x_s(t) - x_s \mid x_s)}{\Pr(x_s(t) = x_s + x_s(t - 1))} \\
&= \frac{\Pr(x_s) \Pr(x_s(t - 1) = x_s(t) - x_s)}{\Pr(x_s(t) = x_s + x_s(t - 1))} \\
&= \frac{\exp(-\lambda_s)x^x_s}{x_s!} \times \frac{\exp(-\lambda_s t)(\lambda_s t)^{x_s(t-1)}}{x_s(t-1)!} \times \frac{x_s(t)!}{\exp(-\lambda_s(1 + t))(\lambda_s(1 + t))^{x_s(t)}} \\
&= \left(\frac{x_s(t)}{x_s}\right) \frac{\exp(-\lambda_s)x^x_s}{x_s!} \frac{\exp(-\lambda_s t)(\lambda_s t)^{x_s(t)-x_s}}{\exp(-\lambda_s(1 + t))(\lambda_s(1 + t))^{x_s(t)}} \\
&= \left(\frac{x_s(t)}{x_s}\right) \frac{\lambda_s^x_s(\lambda_s t)^{x_s(t)-x_s}}{(\lambda_s(1 + t))^{x_s(t)}} \\
&= \left(\frac{x_s(t)}{x_s}\right) \frac{\lambda_s^x_s(\lambda_s t)^{x_s(t)-x_s}}{(\lambda_s(1 + t))^{x_s(t)-x_s}} \\
&= \left(\frac{x_s(t)}{x_s}\right) \left(\frac{1}{1 + t}\right)^{x_s} \left(\frac{t}{1 + t}\right)^{x_s(t)-x_s} \\
&= \left(\frac{x_s(t)}{x_s}\right) \left(\frac{1}{1 + t}\right)^{x_s} \left(1 - \frac{1}{1 + t}\right)^{x_s(t)-x_s}
\end{align*}
\]

This is a binomial distribution with \(n = x_s(t)\) and \(p = \frac{1}{1 + t}\). \(\square\)

Let \(G(\lambda)\) be the empirical cumulative distribution function of \(\lambda_1, \ldots, \lambda_S\). If \(n_x\) is the number of species observed \(x\) times in \([-1, 0]\), let

\[
\eta_x = E(n_x) = \int_{0}^{\infty} \frac{e^{-\lambda_S^x}}{x!} dG(\lambda), \quad (1)
\]

where \(\lambda\) is the \(\lambda\) associated with \(G(\lambda)\). Next, let \(\Delta(t)\) be the expected number of species in \((0, t]\) but not in \([-1, 0]\). Note that using equation [1] the probability of not seeing a species in a given interval (observing \(x = 0\)) is given by \(\exp(-\lambda)\). Likewise, the probability of not seeing a species in the interval \([-1, 0]\) is \(\exp(-\lambda)\) so that the probability of seeing a species in the interval \((0, t]\) is \(1 - \exp(-\lambda t)\). Thus,
\[ \Delta(t) = S \int_0^\infty \exp^{-\lambda}(1 - \exp^{-\lambda t})dG(\lambda). \]

To estimate \( \Delta(t) \) we can use a Taylor series expansion of \( 1 - \exp^{-\lambda t} \),

\[ 1 - \exp^{-\lambda t} = \lambda t - \frac{\lambda^2 t^2}{2!} + \frac{\lambda^3 t^3}{3!} - \ldots \]

and substituting the right hand expression into the expression for \( \Delta(t) \) we get

\[ \Delta(t) = S \int_0^\infty \exp^{-\lambda} \left( \lambda t - \frac{\lambda^2 t^2}{2!} + \frac{\lambda^3 t^3}{3!} - \ldots \right) dG(\lambda) \]

which can be reexpressed as

\[ \Delta(t) = S \int_0^\infty \left( \frac{\exp^{-\lambda} \lambda t}{1!} - \frac{\exp^{-\lambda} \lambda^2 t^2}{2!} + \frac{\exp^{-\lambda} \lambda^3 t^3}{3!} - \ldots \right) dG(\lambda). \]

Substituting equation [1] into the right hand side we get

\[ \Delta(t) = \eta_1 t - \eta_2 t^2 + \eta_3 t^3 - \ldots \]

Thus an estimate for \( \Delta(t) \) is

\[ \hat{\Delta}(t) = n_1 t - n_2 t^2 + n_3 t^3 - \ldots \]

This estimate works well when \( t = 1 \) but when \( t > 1 \) the \( t^x \) values cause wild oscillations which may cause convergence issues. Euler’s transformation is one way to force the series to converge. Euler’s transformation is defined [Knopp 2013] as

\[ \sum_{k=0}^{\infty} (-1)^k a_k = \sum_{n=0}^{\infty} \frac{\Delta^n a_0}{2^{n+1}} \]

where
\[ \Delta^k a_0 = \sum_{m=0}^{k} (-1)^m \binom{k}{m} a_{k-m} \]

which is the form that the unseen species estimate takes. That is

\[ \sum_{k=0}^{\infty} (-1)^k a_k = \sum_{x=1}^{\infty} (-1)^{x+1} \eta_x t^x. \]

Application of this transformation is discussed further in section 1.4.2.

### 1.4.2 TCR\(\beta\) CDR3 Diversity

Robins et al. [2009] used the unseen species method described in section 1.4.1 to estimate the human TCR\(\beta\) repertoire. Using the same terminology as in 1.4.1, the total number of unique TCR\(\beta\) CDR3 sequences in the repertoire is represented by \(S\). Let \(x_s\) represent the number of a specific CDR3 sequence, \(s\), observed in an experiment. Further, assuming T-cells circulate freely in the blood, it can be assumed that each CDR3 count, \(x_s\), is distributed \(Poisson(\lambda_s)\). Let \(x_s(t)\) represent the number of times a CDR3 sequence was observed in sequencing experiment \(t\) and every previous experiment. Here, \(t\) represents the numbered experiment, 1, 2, ..., that was sequenced. Finally, let \(n_x\) represent the number of CDR3 sequences observed exactly \(x\) times. Once again, we need to estimate \(\Delta(t)\), the number of CDR3 sequences expected to be observed in all sequencing experiments except the first one, \(t = 1\). This gives us an estimate of the total number of unique CDR3 sequences that could be observed in one subject. Once again the estimate for \(\Delta(t)\) is

\[ \hat{\Delta}(t) = n_1 t - n_2 t^2 + n_3 t^3 - \ldots \]

By letting \(t = \frac{u}{2-u}\) the following relationship is derived using Euler’s transformation from section 1.4.1.
\[
\sum_{x=1}^{\infty} (-1)^{x+1} \eta_x t^x = \sum_{y=1}^{\infty} \xi_y u^y
\]

where

\[
\xi_y = \sum_{x=1}^{\infty} \left( \frac{y-1}{x-1} \right) \frac{(-1)^{x+1}}{2^y} \eta_x = \frac{1}{2^y} \delta^y(\eta_1)
\]

and

\[\delta^0 = \eta_1, \delta^1 = \eta_1 - \eta_2, \delta^2 = \eta_1 - 2\eta_2 + \eta_3, \ldots.\]

Let

\[
\Delta^{x_0}(t) = \sum_{x=1}^{\infty} (-1)^{x+1} \eta_x t^x
\]

\[
\Delta^{x_0}(u) = \sum_{y=1}^{\infty} \xi_y u^y
\]

\[
\Delta(t) = \lim_{x_0 \to \infty} \Delta^{x_0}(t)
\]

\[
\Delta(u) = \lim_{x_0 \to \infty} \Delta^{x_0}(u).
\]

By definition \(\Delta(t) = \Delta(u)\) if both limits exist. Here \(\Delta^{x_0}(u)\) converges much more quickly to the common limit Efron and Thisted [1976]. We can estimate \(\xi_y\) by substituting \(n_x\) for \(\eta_x\) and then estimate \(\Delta(t)\) using

\[
\Delta^{x_0}(u) = \sum_{y=1}^{\infty} \xi_y u^y = \sum_{y=1}^{x_0} \frac{1}{2^y} \delta^y(\eta_1) u^y, u = \frac{2t}{1+t}.
\]

Using the results from their sequencing experiments, Robins et al. [2009] estimated the human TCR\(\beta\) repertoire to be \(10^5\) CDR3 sequences. This result is similar to that obtained by Arstila et al. [1999] who estimated it to be \(10^6\).
1.4.3 Models for Estimating Population T-cell Diversity

Section 1.6 describes several T-cell receptor analysis methods in detail. This section will briefly describe several other methods present in the literature. Specifically, models to estimate overall T-cell diversity rather than just sample T-cell diversity will be presented.

Sepúlveda et al. [2010] described a series of Poisson abundance models that can be used to estimate the diversity in the original T-cell population. The clonal size distribution can be described by the following Multinomial law

\[ P[m_x | D, \eta] = \frac{D!}{(D-M) \prod_{x=1}^{n} m_x!} \prod_{x=1}^{n} [p_\eta(x)]^{m_x}, \]  

where \( D \) is the number of distinct clonotypes in the population which is the population diversity in this case. \( m_x \) is the number of clonotypes with \( x \) copies in the sample and \( p_\eta(x) \) is the probability of a clonotype being sampled \( x \) times and is described by a model with parameter vector \( \eta \). \( M \) is the sample diversity and \( n \) is the sample size. By assuming \( p_\eta \) follows a Poisson distribution, one can obtain the class of Poisson abundance models.

The simplest Poisson abundance model is assuming a homogeneous Poisson distribution for all clonotypes. That is

\[ p_\lambda(x) = \frac{e^{-\lambda} \lambda^x}{x!} \]  

where \( \lambda \) is the sampling rate. This can then be plugged back into equation 2 for \( p_\eta \). However, it is very simplistic to assume all clonotypes have the same sampling rate. Thus, different distribution can be obtained by applying a distribution to \( \lambda \).

The first model to consider is the Poisson-Gamma model where \( \lambda \) follows the Gamma distribution. This assumes clonotypes have different sampling rates based on their clonal size (the number of times a clonotype appears in the sample). By mixing equation 3 with a Gamma distribution for \( \lambda \) we get
\[ p_{\alpha,\beta}(x) = \int_0^\infty \frac{e^{-\lambda} \lambda^x}{x!} \frac{\beta^\alpha \lambda^{\alpha-1} e^{-\lambda \beta}}{\Gamma(\alpha)} d\lambda = \frac{\Gamma(x + \alpha)}{\Gamma(x + 1) \Gamma(\alpha)} \left( \frac{\beta}{\beta + 1} \right)^\alpha \left( \frac{1}{\beta + 1} \right)^x, \quad (4) \]

where \( \alpha \) and \( \beta \) are the shape and scale parameters of the Gamma distribution.

Another model is the Poisson-lognormal model. This model assumes that \( \lambda \) follows a lognormal distribution with parameters \( \mu \) and \( \sigma^2 \). Using this model the sampling probability is

\[ p_{\mu,\sigma^2}(x) = \int_0^\infty \frac{e^{-\lambda} \lambda^x}{x!} \frac{e^{(\log \lambda - \mu)^2/2\sigma^2}}{\sqrt{2\pi\sigma^2}} d\lambda. \quad (5) \]

This does not have a closed form and a numerical algorithm can be used such as the Newton-Raphson to calculate the probabilities.

The final model assumes \( \lambda \) follows an exponential distribution with parameter \( \mu = e^{-\omega}/(1-e^{-\omega}) \). Further, \( \omega \) follows an exponential distribution with parameter \( \rho \). Thus we get the following sampling distribution

\[ p_\rho(x) = \rho \frac{\Gamma(x + 1) \Gamma(\rho + 1)}{\Gamma(x + \rho + 2)}. \quad (6) \]

Using the sampling probability equations, the diversity, \( D \), can be estimated by maximizing equation 2 [Rempala et al. 2011] and Greene et al. 2013 expand these models to the multivariate setting to estimate population diversity. To choose the model that best fits the data goodness of fit tests can be computed such as the Pearson chi-squared test and AIC measure. Once an appropriate model is chosen, reportoires can be compared. The diversities can be compared by tests such as the Chi-squared or Kolmogorov-Smirnov tests. Because these tests are very sensitive to sample size, Mehr et al. 2012 suggest using machine learning methods such as repertoire clustering by similarity.
1.5 Application of TCRβ Studies

1.5.1 Hematopoietic stem cell transplantation

Hematopoietic stem cell transplantation (HSCT) is used to treat hematological disorders such as blood cancers. The procedure involves replacing diseased host cells with healthy donor cells. A result of this procedure is graft-versus-tumor effect (GVT), which is when the reconstituted T-cells recognize the blood cancer and mount an immune response. Conversely, the reconstituted T-cells can recognize the new host and mount an autoimmune response known as graft-versus-host-disease (GVHD) [Avent, 2012]. Because GVHD is potentially lethal, it is extremely important to match donors to recipients to minimize the risk of donor T-cells recognizing the recipients cells. The ability to measure TCRβ clone abundance can greatly help in understanding the immune system’s requirement for initiating GVT versus GVHD.

1.5.2 Case Studies

Several studies have used the ImmunoSEQ assay to describe the T-cell repertoire in a variety of circumstances such as monitoring minimal residual disease (MRD), measuring diversity pre- and post-treatment and describing the T-cell repertoire during viral infection. This section will briefly describe six studies that have utilized the ImmunoSEQ assay.

[Weng et al.] [2013] studied the ability of high-throughput sequencing to monitor MRD in T-cell lymphoma patients. Samples from 10 patients were examined using the ImmunoSEQ assay. Tumor clones were identified prior to HSCT treatment and monitored using high-throughput sequencing. By using the ImmunoSEQ, [Weng et al.] [2013] were able to monitor recurrence of tumor clones that were a precursor of disease relapse using plots of percentage of malignant clone. Spike-in experiments were also performed that showed the assay could detect tumor clones with a sensitivity of 1:150,000.

[Cooper et al.] [2013] studied how BRAF inhibition affects the T-cell repertoire in patients
with metastatic melanoma. Serial biopsies from 8 patients were sequenced prior to treatment and 10-14 days after treatment using the ImmunoSEQ assay. T-cell diversity was calculated pre- and post-treatment and the study found that treatment with \textit{BRAF} inhibitors (\textit{BRAFi}) resulted in an increase in T-cell diversity in 7 out of 8 patients. The study also found that approximately 80\% of the clones found in patients post-treatment were not present before treatment suggesting \textit{BRAFi} treatment results in an influx of T-cells. Finally, the study found that the proportion of dominant clones present both pre- and post-treatment had an association with patient outcome. Patients with a high-proportion of pre-existing clones post-treatment had a better clinical outcome. It was unclear what specific statistical tests were performed.

\textcite{Robert et al. 2014} also studied patients with metastatic melanoma but investigated treatment with drugs that inhibit cytotoxic T-lymphocyte-associated protein 4 (\textit{CTLA4}) instead of \textit{BRAFi}. \textit{CTLA4} inhibition can result in tumor remissions due to a T-cell response but this response has not been fully described. This study had two major conclusions. First, T-cell diversity increases post-treatment but does not seem to be related to the patients’ ability to fight cancer. Diversity was calculated using the Shannon diversity index which is defined as

$$H = - \sum_{i=1}^{L} p_i \log(p_i)$$  \hspace{1cm} (7)

where $p_i$ is the probabilty of CDR3 length \textit{i}. Second, patients with toxicity had a significantly higher number of unique productive sequences after treatment. Because the authors had a large enough sample the Wilcoxon signed-rank test was used to compare diversity measures pre- and post-treatment. The authors concluded that these results point to a general activation of the immune system rather than a specific anti-cancer immune response.

\textcite{Muraro et al. 2014} studied the T cell repertoire post-HSCT in 24 multiple sclerosis (MS) patients. HSCT can be used as a way to reset the immune system in MS patients that have autoimmune disease (an organisms immune response to its own cells). The end point
of interest is a non-autoimmune repertoire. Samples were sequenced prior to treatment, two months after treatment and one year after treatment. The study found that patients that had a higher T-cell diversity at two months had a better clinical outcome. In this case diversity was the percentage of T cell repertoire that the top 100 clones included. It was unclear what statistical tests were performed to test for differences.

Finally, Zhu et al. [2013] and Neller et al. [2012] used the ImmunoSEQ assay to examine the T-cell repertoire in patients with herpes virus infection. Both studies showed that virus specific clonotypes are persistent in the patient’s T-cell repertoire. This suggests that there is a mechanism to promote a prompt response to viral recurrence.

1.6 Overview of Statistical Methods for CDR3 Data

Spectratype analysis is universally accepted as a procedure to monitor and calculate CDR3 length distributions. However, no agreement has been reached on a method to statistically compare individual repertoires. There are two major hurdles when analyzing CDR3 spectratype data. The first is extracting information from each CDR3 distribution and the second is integrating all of the information from all the CDR3 distributions [Miqueu et al., 2007]. Initially, spectratype analysis was done by “eye-balling” the distribution and placing it into a descriptional category. However, this methodology suffers from variation due to researchers having different ideas about which categories different distributions should fit into. Visual inspection and categorization is also an extremely time consuming method. Hence, researchers have used several computational methods to describe CDR3 distributions. Gorochov et al. [1998] described a method for measuring CDR3 distribution perturbation from a control distribution. This method is applied to T-cell repertoires during progression of AIDS and describes the distributions as a whole. Collette et al. [2003] took this a step further and described an OligoScore which describes each peak individually. This method can be used to identify recurrent peaks in different samples. Venturi et al. [2007] adapted a method from ecology called Simpson’s diversity index to measure the diversity in a CDR3
distribution. This method was applied to assess the T-cell repertoire recovery after HSCT by van Heijst et al. [2013]. All these methods assign scores to which statistical methods can then be applied. Kepler et al. [2005] described a statistical method that used the Kullback-Leibler divergence as a measure of how different CDR3 length distributions are from each other. Most recently, Bolkhovskaya et al. [2014] described a non-parametric method for comparing TCR distributions. In the following subsections each method is clearly described.

1.6.1 CDR3 Distribution Perturbation

The CDR3 length distribution measure was applied to measure T-cell repertoires during progression to AIDS. This is a measure of how different TCR distributions are from each other. Gorochov et al. [1998] used data that was translated into a probability distribution from Immunoscope analysis. The area under each peak was measured to calculate a probability using the following formula

$$p_i = \frac{A_i}{\sum_i A_i}$$

where $A$ is the area under each peak (CDR3 length) and $i = 1, \ldots, L$ where $L$ is the number of amino acid lengths in the data. Modifying this notation, let $p_{ik}^f$ represent the probability of peak $i$ of TCR$\beta$ family $f$ (VDJ combination) from sample $k$. The method then calls for a control distribution for each TCR$\beta$ family. This was done by taking the average of the control samples denoted by

$$p_{ic}^f = \frac{\sum_{k \in c} p_{ik}^f}{|c|}$$

where $c$ represents the indices of the control samples and $|c|$ is the cardinality of $c$. The perturbation of each peak from the control for sample $k$ is then defined as

$$D_{ik}^f = p_{ik}^f - p_{ic}^f.$$
The sum of the absolute value of the peak perturbations is then

\[ D^f_k = 100 \times \sum_i \frac{|D^f_{ik}|}{2} \quad (11) \]

for sample \( k \) and TCR\( \beta \) family \( f \) which yields an overall measure of perturbation for that TCR profile in percentages for each TCR\( \beta \) family. This is the generalized Hamming distance and measures the distance between two strings of equal size. This measure ranges from 0 to 100 where 0 represents no difference from the control and 100 represents no overlap with the control distribution. A further measure of overall perturbation for a sample \( k \) is defined as

\[ D_k = \sum_{f=1}^{F} \frac{D^f_k}{F} \quad (12) \]

where \( F \) is the number of TCR\( \beta \) families.

Different statistical tests can then be applied to the measures of perturbation to compare different groups. Gorochov et al. [1998] chose to use the Wilcoxon Rank sum test to compare groups. However, depending on sample size and distributional assumptions, a t-test may be appropriate as well. The null hypothesis for this test, assuming a Gaussian distribution, would be that the mean \( D_k \) is the same across all groups and the alternative hypothesis is that at least one group’s mean is different. The hypotheses would be the same when using \( D^f_k \) except there would be many tests done, one for each TCR\( \beta \) family. In this case a multiple comparisons adjustment needs to be applied. For a more detailed discussion of multiple comparisons see section 1.6.8.

### 1.6.2 OligoScores

Collette et al. [2003] described a score to measure each peak within a group. The OligoScore is defined for the \( i^{th} \) peak of the \( k^{th} \) sample as

\[ OligoScore(i) = \exp(1) \times \sqrt{\prod_{k=1}^{N} p_{ik} \exp(-n_k)} \quad (13) \]
where $p_{ik}$ is probability of the $i^{th}$ peak for the $k^{th}$ sample (as described in section 1.6.1), $n_k$ is the number of peaks in each sample and $N$ the number of samples [Collette and Six 2002]. This score gives a measure of each peak within a group for each TCR$\beta$ family, $f$. Oligoscores can be interpreted as a score that measures if the distribution has a major peak or not (higher scores mean a peak is major). Collette and Six [2002] did not do any hypothesis testing and instead ranked all oligoscores within each group. The highest oligoscore peaks were then looked at graphically. This score can be adapted to compute peaks for each VDJ combination.

### 1.6.3 Simpson’s Diversity Index

Slow T-cell recovery and low TCR diversity are risk factors for infection, cancer relapse and graft-versus-host-disease (GVHD) after hematopoietic stem cell transplantation (HSCT) [van Heijst et al. 2013]. See section 1.5.1 for a detailed description of GVHD and HSCT. Hence, measuring TCR diversity post-HSCT is an important way to monitor patient health. van Heijst et al. [2013] suggested using the inverse Simpson’s diversity index ($\frac{1}{D_S}$). This measure ranges from 1 to $\infty$ where 1 is no diversity and $\infty$ is high diversity. The Simpson’s diversity index in terms of TCR diversity using unique CDR3 length is the probability that any two TCR’s chosen at random from the sample will have different CDR3 lengths and is defined as

$$D_S = 1 - \sum_{i=1}^{L} \frac{n_{ikf}(n_{ikf} - 1)}{n_{kf}(n_{kf} - 1)} \quad (14)$$

where $n_{ikf}$ is the size of the $i^{th}$ TCR CDR3 length (the number of copies of each TCR within a CDR3 length and equivalent to the peak described in section 1.6.1) for the $k^{th}$ sample in TCR$\beta$ family, $f$. $L$ is the number of different TCR CDR3 lengths in the sample and $n_{kf}$ is the total number of TCR sequences sampled in sample $k$ of TCR$\beta$ family, $f$. The diversity index is computed for each TCR$\beta$ family. This measure can then be used to compare groups using t-tests or other inferential methods. Further, it can be adjusted to be computed for
van Heijst et al. [2013] were not interested in comparing groups but time points. Using paired t-tests they found the patient TCR diversity did not change over time. However, using correlations they did find that the composition of the TCR distribution changed drastically. For example, the dominant peak from one time point to the next could be completely different depending on what antigens the patient was in contact with recently.

1.6.4 Kullback-Leibler Divergence Method

Kepler et al. [2005] described this method directly for spectratype data which results in relative abundance measures as described in section 1.2. Let’s consider the multinomial distribution in terms of CDR3 length distribution. In this case, $L$ is the number of unique CDR3 lengths, $q$ is a vector of probabilities of each CDR3 length occurring, $m_i$ is the number of counts of the $i^{th}$ CDR3 length and $n_k$ is the total number of counts within a sample. Thus, the probability mass function (pmf) of the relative abundance of the lengths of CDR3 clones, $p = \frac{m}{n}$ is

$$f_p(p|n_k, q) = n_k^{L-1}C_{n_k}(n_k p) \prod_{i=1}^{L} q_i^{m_i n_k},$$

(15)

where $C_{n_k}(n_k p)$ is the multinomial coefficient and $q$ is a vector of probabilities corresponding to each category $i$.

$$C_{n_k}(n_k p) = \frac{\Gamma(n_k + 1)}{\prod_{i=1}^{L} \Gamma(n_k p_i + 1)}$$

(16)

for $\sum_{i=1}^{L} n_k p_i = n$ and zero otherwise.

The Stirling approximation can then be applied to the multinomial coefficient to simplify its form assuming that for each $i$, $n_k p_i$ is large enough.
\[
\log C_{n_k}(n_k \mathbf{p}) = \log \Gamma(n_k + 1) - \sum_{i=1}^{L} \log \Gamma(p_i n_k + 1)
\]

\[
= \sum_{i=1}^{L} \left[-n_k p_i \log p_i - \frac{1}{2} \log p_i \right] - \frac{L - 1}{2} \log 2\pi n_k + O\left(\frac{1}{n_k}\right) \tag{17}
\]

Plugging this back into the pmf for \( \mathbf{p} \) we get

\[
f_r(\mathbf{p}|n_k, \mathbf{q}) = \frac{n_k^{(L-1)/2}}{\sqrt{(2\pi)^{L-1}\prod_{i=1}^{L} p_i}} e^{-n_k D(\mathbf{p}; \mathbf{q})} \delta(0) \tag{19}
\]

where \( \delta \) is the Dirac delta, \( p. = \sum p_i \) and \( D(\mathbf{p}; \mathbf{q}) \) is the Kullback-Leibler divergence.

\[
D(\mathbf{p}; \mathbf{q}) = \sum_{i=1}^{L} p_i \log \frac{p_i}{q_i} \tag{20}
\]

where \( p. = q. = 1. \)

To determine the distribution of the Kullback-Leibler divergence we can apply a cumulative generating function method where

\[
h(s) = \log \int d^{L} p e^{s D(\mathbf{p}; \mathbf{q})} f_r(\mathbf{p}|n_k, \mathbf{q})
\]

\[
= -\lambda \log \left(1 - \frac{s}{n_k}\right) \tag{21}
\]

This is the cumulative generating function of a gamma random variable with \( \lambda = (L - 1)/2 \) and scale \( = \frac{1}{n_k} \). The the distribution of the Kullback-Leibler divergence is given by

\[
f_D(D|n_k) = \frac{n_k^{\lambda}}{\Gamma(\lambda)} D^{\lambda - 1} e^{-n_k D} \tag{23}
\]

Using the above derivations we can compare group distributions of the CDR3 length. We will be testing to see if the population parameter \( \mathbf{q} \) is identical in all groups. Thus, our null
hypothesis is that the distribution of CDR3 lengths is identical in all groups.

Let \( p_{ijk} \) represent the relative frequency counts of CDR3 length \( i = 1, \ldots, L \), where \( j = 1, \ldots, G \) represents the group and \( k = 1, \ldots, n_{ij} \) the \( k^{th} \) subject of the \( j^{th} \) group for CDR3 length \( i \). Then the mean of group \( j \) for CDR3 length \( i \) is defined as

\[
\bar{p}_{ij} = \frac{1}{n_i} \sum_{k=1}^{n_j} p_{ijk}
\]

where \( n_i \) is the number of counts of CDR3 length \( i \) and \( n_j \) is the number of subjects in group \( j \). If \( G \) is the number of groups then the grand sample mean is

\[
\bar{p}_{i.} = \frac{1}{n.} \sum_{j=1}^{G} n_i \bar{p}_{ij}
\]

where \( n. \) is the total number of subjects.

If we denote \( p_{jk} \) as the vector that has components \( p_{ijk} \) we can partition the total divergence as

\[
D_{tot} = \sum_j \sum_k D(p_{jk}; q) = \sum_j \sum_k [D(p_{jk}; \bar{p}_{k.}) + D(\bar{p}_{k.}; \bar{p}_{..}) + D(\bar{p}_{..}; q)]
\]

Under the null hypothesis that \( q \) is the same in all groups the expected values of the partitioned divergences are

\[
E[D(p_{jk}; \bar{p}_{k.})] = \frac{L - 1}{n_k}(1 - \frac{G - 1}{n.})
\]

\[
E[D(\bar{p}_{k.}; \bar{p}_{..})] = \frac{L - 1}{n_k n.}(1 - \frac{1}{G})
\]

\[
E[D(\bar{p}_{..}; q)] = \frac{L - 1}{n_k n_k}
\]
Finally, because the divergences are distributed Gamma, the following statistic is asymptotically distributed as Fisher’s F with \((L-1)(G-1)\) numerator and \((L-1)(n.-G+1)\) denominator degrees of freedom under the null hypothesis.

\[
F = \frac{(n. - G + 1)\Sigma_k D(\bar{p}_k; \bar{p}.)}{(G - 1)\Sigma_{jk} D(p_{jk}; \bar{p}_k)}
\]  

(31)

Notice that the numerator is a between group divergence and the denominator is within group divergence. This is analogous to the typical ANOVA test. This statistic can be used to compare different groups of subjects that have TCR data collected for them at each VDJ combination, \(f\).

### 1.6.5 Non-parametric Method

[Bolkhovskaya et al. 2014] described a non-parametric method for assessing the TCR length distribution. To incorporate statistical uncertainty, a non-parametric kernel distribution estimator is constructed for the cumulative frequency distribution, \(F_f(p^*) \equiv \text{Prob}(p \geq p^*)\)

\[
\hat{F}_f(p) = \frac{1}{L} \sum_{i=1}^{L} \Phi(p - p_i),
\]

where \(L\) is the number of TCR lengths, \(p_i\) is the frequency of each TCR CDR3 length, \(p\) is a pre-chosen frequency value and \(\Phi(\cdot)\) is the cumulative normal distribution with mean zero and standard deviation

\[
\sigma_{p_i} = \left[ \sum_{s=1}^{S} \sigma_{p_i,s}^2 \right]^{\frac{1}{2}}
\]

(33)

where \(\sigma_{p_i,s}^2\) is the standard deviation present at each of \(S\) steps of repertoire profiling. Three steps (S) during the sampling process that add to statistical error are sampling T-cells, PCR and sequencing. Each of these steps introduce variation to the final TCR frequency. [Bolkhovskaya et al. 2014] assume the sampling process at each step takes on a binomial distribution so

24
\[ \sigma_{p_{i,s}}^2 = \sqrt{\frac{p_i(1-p_i)}{N_s}} \]  

(34)

where \( N_s \) is the total number of samples at step \( s \). However, this variation can only be measured at the sequencing stage so \( S = 1 \). From here the probability density distribution can be estimated by

\[ \hat{W}(p) = \frac{d}{dp} \hat{F}_f(p). \]  

(35)

Finally, the upper 95% confidence interval can be computed using

\[ p_0 = 1 - (1 - 0.95)^z, \quad z = \sum_{s=1}^{S} \frac{1}{N_s}. \]  

(36)

Bolkhovskaya et al. [2014] applied this method to patient data and found that \( F_f(p) \) follows a power law decay

\[ F_f(p) \propto p^{\alpha_i}. \]  

(37)

The power law represents a relationship between two variables where one variable varies as the power of another. It can be seen in scenarios where a few values dominate. Many times a TCR CDR3 length distribution is dominated by one or two lengths. Thus, the power may be an appropriate way to compare TCR CDR3 length distributions between groups. This can be done by calculating and comparing \( \alpha \) values between groups.

### 1.6.6 Proportion Logit Transformation

The proportion of shared sequences between donor and recipient can be considered as a measure of how closely the donor and recipient are matched. The question posed is if there is a difference in the proportion of shared sequences between donor and recipient between patients with GVHD and no GVHD. Hence, we can match recipient and donor
TCR sequencing data by amino acid sequence and calculate the proportion of unique and shared TCR sequences. Proportions do not follow a Gaussian distribution. However, by applying a logit transformation to the proportions, they can be transformed into a value that follows a Gaussian distribution. If we let $\pi$ be the proportion of shared sequences, the logit transformation is defined as:

$$\text{logit}(\pi) = \log\left(\frac{\pi}{1 - \pi}\right) \quad (38)$$

where $\pi_{kf}$ is the proportion of shared sequences for subject $k$ within VDJ family, $f$. Once $\pi_{kf}$ is logit transformed, t-tests can be applied to test for group mean differences. Since each VDJ combination is to be tested a multiple comparisons correction is needed. A discussion of multiple comparisons can be found in section 1.6.8.

### 1.6.7 Tabular Summary of Statistical Methods for CDR3 data

Table 1 shows a summary of the methods described and test that is being performed.

<table>
<thead>
<tr>
<th>Method</th>
<th>Summaries</th>
<th>Alternative Hypothesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perturbation using Matched Donor</td>
<td>Each VDJ combination</td>
<td>Group perturbations are different</td>
</tr>
<tr>
<td>Perturbation using Mean Distribution</td>
<td>Each VDJ combination</td>
<td>Group perturbations are different</td>
</tr>
<tr>
<td>Oligoscores</td>
<td>Each CDR3 length within a VDJ combination</td>
<td>N/A</td>
</tr>
<tr>
<td>Simpson’s Diversity Index</td>
<td>Each VDJ combination</td>
<td>Group diversity measures are different</td>
</tr>
<tr>
<td>Kullback-Leibler Divergence</td>
<td>Each VDJ combination</td>
<td>Group distributions are different</td>
</tr>
<tr>
<td>Non-parametric Method</td>
<td>Each VDJ combination</td>
<td>Group $\alpha$ values are different</td>
</tr>
<tr>
<td>Proportion Logit Transformation</td>
<td>Each VDJ combination</td>
<td>Group proportions are different</td>
</tr>
</tbody>
</table>

Table 1: Table of Methods for Analyzing TCR Sequencing Data
1.6.8 Multiple Comparisons

Testing many independent hypotheses runs the danger of identifying significant differences that are the result of random chance. Methods such as the Bonferroni correction aim to control the probability of making a Type I error. This means that these methods also control the simultaneous correctness of all rejections [Benjamini and Hochberg, 2000]. However, the Bonferroni method is conservative and results in a substantial loss in power for individual tests. For high-throughput technologies it is more important to find all possible differences at the cost of getting a few hypotheses wrong.

The false discovery rate is defined as the expected ratio of erroneous rejections to the number of rejected hypotheses. Benjamini and Hochberg [1995] showed that this procedure has a substantial increase in power over classic $\alpha$ controlling methods. The procedure was shown to control the FDR at a lower level than desired so the adaptive procedure was created [Benjamini and Hochberg, 2000].

Let $m$ represent the total hypotheses that we wish to test and $m_0$ represent those that are true. Let $R$ define the total number of hypotheses rejected and $V$ denote the number of those hypotheses that were rejected incorrectly. Note that $V$ is an unobservable variable. Using these definitions the effective error rate is $E\left[ \frac{V}{m_0} \right]$ and the type I error rate is $P(V \geq 1)$. By testing each hypothesis at level $\alpha$ we guarantee $E\left[ \frac{V}{m_0} \right] \leq \alpha$ but not $P(V \geq 1) \leq \alpha$. Using a procedure such as the Bonferroni by testing each hypothesis at $\frac{\alpha}{m}$ guarantees $P(V \geq 1) \leq \alpha$. However, in the FDR procedure the error of falsely rejecting a null hypothesis is captured by the random variable $Q = \frac{V}{R}$, the proportion of false-positives [Benjamini and Hochberg, 1995]. Thus,

$$FDR = E(Q) = E\left( \frac{V}{R} \right).$$

Benjamini and Hochberg [2000] showed that when $m_0 < m$ the FDR is smaller than or equal to the Type I error rate. Thus any procedure that controls the Type I error rate also
controls the FDR. But if you are only controlling the FDR then an improvement in power is expected. The following procedure is shown to control the FDR at level $q$ and improve power:

1. Let $P_1 \leq \ldots \leq P_i \leq \ldots \leq P_m$ be the ordered p-values.
2. Let $k$ be the largest $i$ for which $P_i \leq \frac{i}{m} q$.
3. Reject all $k$ ordered hypotheses.

The proof that this procedure controls the FDR is based on the lemma that says for $m_1 = m - m_0$ false null hypotheses the test procedure satisfies the inequality \[ E(Q|P_{m_0+1} = p_1, \ldots, p_m = p_{m_1}) \leq \frac{m_0}{m} q \]

Thus, when some of the null hypotheses are rejected the FDR is controlled at a level less than $q$. So there is once again room to improve power in this procedure. If $m_0$ was known then $E(V) = m_0 \alpha$ and $r(\alpha) = v + s$ where $s$ is the number of null hypotheses rejected correctly. So the new estimate of the FDR can be $\hat{Q}_e(\alpha) = \frac{\hat{v}(\alpha)}{\hat{r}(\alpha)} = \frac{\hat{m}_0 \alpha}{\hat{r}(\alpha)}$.

Then to control the FDR at level $q$ we can follow the following procedure

1. Let $k$ be the largest $i$ for which $P_i \leq \frac{i}{m_0} q$.
2. Reject all $k$ ordered hypotheses.

The final problem is how to estimate $m_0$. A graphical method can be used. If you consider the p-values to be independent, under the null hypothesis they should follow a uniform[0,1] distribution. Thus, a quantile plot of the $p_i$’s should have a linear relationship through the
origin and slope of $S = \frac{1}{m+1}$. When there are false null hypotheses they tend to stack at the lower end. However, the upper end should continue with a linear relationship. Thus, we can use the slope of the higher p-values to solve for $m_0$. Benjamini and Hochberg [2000] suggest the estimate of $m_0$ to be calculated as follows.

1. Calculate slopes $S_i = \frac{1-p_i}{m+1-i}$ and continue as long as $S_i \geq S_{i-1}$

2. Stop once $S_j < S_{j-1}$.

3. Then $\hat{m}_0 = \min(\frac{1}{S_j} + 1, m)$.

1.7 Discussion

In this chapter we discussed T-cell biology and the ImmunoSEQ platform used to analyze the T-cell repertoire. Further, different methodologies were described to analyze the high-throughput sequencing data generated by the ImmunoSEQ assay. In the next chapter, we will apply these methodologies to a hematopoietic stem cell transplantation data set. In Chapter 3, we will discuss how high-throughput sequencing data can be considered compositional data and the analytical challenges that can present. Finally, the same HSCT data from chapter 2 will be analyzed as compositional data.
2 Applications to TCR sequencing data

2.1 Introduction

In this chapter, HSCT data (described in section 2.3) were analyzed using the methodologies described in chapter 1. To determine what pre-processing steps other researchers used, previously published data were reanalyzed. Finally, the differences in the methods were discussed.

2.2 Reanalysis of Published TCR data

We acquired the data from Dr. Harlan Robins’ lab that were analyzed in the 2012 paper titled “Ultra-sensitive detection of rare T-cell clones” [Robins et al., 2012]. This article describes the Adaptive Technologies ImmunoSEQ technology and its ability to detect T-cell clones. Thus, it was useful to reanalyze the data to determine the exact process used to perform the analysis. Specifically, we acquired replicate data for one patient but were unable to obtain the spike-in data used in the experiments. The data that we acquired included samples DNA A, DNA B, PCR A1, PCR A2, SeqA1b and SeqA1a.

We performed a similar analysis and compared our results to that reported by [Robins et al., 2012]. In the publication, the authors were not specific about their data processing procedures so several different attempts were made to replicate their results. The first step required deciding whether to use all the sequences or a subset. The sequences are flagged by Adaptive Biotechnologies software as productive (usable proteins) or non-productive and through trial and error we discovered both productive and non-productive sequences were used for the analysis performed by [Robins et al., 2012]. Next, in order to compare counts between replicates the files needed to be merged. The CDR3 sequences can be merged either by amino acid or nucleotide sequence and through trial and error we discovered [Robins et al., 2012] merged by nucleotide sequence.

Comparing Figure 6 to Figure 2 reported in the [Robins et al., 2012] paper, we see that the
log-log plots are almost identical. Thus we established in our reanalysis the specific process the authors undertook. We noted a small difference between the total number of shared sequences between samples SeqA1a and SeqA1b. The authors identified 150,612 sequences but our analysis identified 150,567.

Figure 6: Reproducibility Plots The plots show frequencies of unique sequences from different samples.

2.3 Data

Thirteen patients with hematological disorders who underwent HSCT were included in this study. This was a randomized phase II trial conducted at Virginia Commonwealth University
of a reduced intensity condition regimen for patients undergoing HSCT [Meier et al., 2013]. Human leukocyte antigen (HLA) matching was done and only patients with 7 of 8 or 8 of 8 matches were eligible for transplantation. HLA is a protein found on most cells in the body and there are 8 major types. The best transplant outcomes happen when patient and donor HLA’s are closely matched. Note that the TCR’s present are also determined by the protein they recognize along with the HLA’s present. Thus, the sample data could be skewed despite the close HLA matching. Peripheral blood stem cells were collected from the donor and recipient whole blood samples were collected post-HSCT. The goal of this analysis was to identify differences at the molecular level between T-cell repertoires of patients who suffered from GVHD and those that did not. Eight patients suffered from GVHD and five did not. T-cell sequencing data was generated using the ImmunoSEQ technology described in section 1.2. The methods described in section 1.6 were applied to the data and the results are described in the following sections.

All data analyses were performed using the R programming environment version 3.1.1 [R Core Team, 2014]. Only sequences that could result in usable proteins (productive sequences) were used in the analyses. Further, reads per million (RPM) were calculated for each sample and used in the analyses. RPM were calculated by dividing each CDR3 frequency by the total number of CDR3 sequences and multiplying by one million. VDJ combinations were filtered out to keep only combinations that had counts for at least 7 patients, leaving 947 VDJ combinations for analysis. CDR3 length summaries were obtained by summing all the CDR3 RPM’s of the same length within each VDJ combination. Further data management applied to a specific method will be described in the following sections.

2.4 Permutation Tests

Because of the low sample size (13 patients) permutation tests were used to compare groups. In this case, there are two samples that have been drawn from two distributions that may or many not be the same. That is the no GVHD group is drawn from a distribution, $F$, 


independent of the GVHD group from a distribution, $G$. Thus, the null hypothesis to test is that the two distributions are equal and the alternative is that they are not equal. That is

$$H_0 : F = G$$

$$H_a : F \neq G$$

Under the null hypothesis, any of the 13 observations could have come from either distribution $F$ or $G$. All 13 observations were considered as a single set of values. A sample of size five was drawn without replacement from this set of values to represent the no GVHD group. The remaining eight observations represent the GVHD group. 1000 random samples were drawn in this manner. The absolute difference between group means, $|\hat{\mu}^*|$, was calculated for each permutation. The p-value is then calculated as follows

$$p-value = \frac{\#(|\hat{\mu}^*| \geq |\hat{\mu}|)}{1000}$$

(39)

where $|\hat{\mu}|$ is the absolute difference in means using the original sample. Thus, this is the probability that the statistic was greater than that which was observed using the original sample [Efron and Tibshirani [1994]]. P-values were calculated for all 947 VDJ combinations considered for analysis. The p-values were then corrected for multiple comparisons using an FDR threshold of 0.05. All permutation tests and FDR corrections were implemented in the R programming environment.

### 2.5 CDR3 Distribution Perturbation Results

The CDR3 Distribution Perturbation method described in section 1.6.1 was adapted for TCR sequencing data and applied to the study data. Each possible VDJ combination, $f$, was examined individually. There were 947 perturbations calculated for each sample after filtering. The perturbations were calculated both as a perturbation from each donor’s T-cell
distribution, $D^f_{kd}$, and as a perturbation from an average donor T-cell distribution, $D^f_{ka}$. $D^f_{kd}$ measures the perturbation from each donor and can be considered as a measure of how similar the recipient is to the donor. $D^f_{ka}$ measures the perturbation from a control distribution and can be considered to measure how different the samples are from each other. Depending on which calculation is used the interpretation of the difference between the GVHD and no GVHD groups is slightly different. Permutation tests described in section 2.4 were used to test for differences between the recipient GVHD and no GVHD groups.

The probability of each CDR3 length, $i$, was calculated using equation 8. The RPM were summed for each unique CDR3 length and then divided by the total RPM present within the VDJ combination of interest to calculate the CDR3 length probability, $A_i$. First, the perturbation for each recipient was calculated as a perturbation from their matched donor. The donor CDR3 length probabilities were calculated in the manner described above. The perturbations were also calculated as a perturbation from a mean distribution of the 13 donors. The mean distribution was calculated by averaging the probability of each CDR3 length across all 13 donors. The perturbations were calculated using equation 10, where $k$ is each sample and $c$ is the control distribution. The control distribution is either an average donor distribution or the matched donor distribution.

Then statistics were calculated using each matched donor for each VDJ combination and tested for differences between the GVHD and no GVHD groups. After adjusting for multiple comparisons using the Benjamini and Hochberg FDR method described in section 1.6.8, there were two significant differences (FDR < 0.001). The distribution of the statistic for the two significantly different VDJ combinations can be seen in Figure 7. The figure shows boxplots where the median, 25th and 75th percentiles make up the box. The whiskers extend to 1.5 times the interquartile range (IQR) and if there are outliers they are represented by dots outside the whiskers. Table 2 shows the median perturbations and observed FDR for the two VDJ combinations. It was of interest to determine why the differences are in opposite directions. Looking at Figures 9 and 10 it appears as though the VDJ combination TRBJ2-
6 TRBV9 TRBD1-2 is not present in a large number of the samples. Figure 9 shows the CDR3 length distributions in the donors for VDJ combination TRBJ2-6 TRBV9 TRBD1-2 and Figure 10 shows the CDR3 length distributions in the recipients for VDJ combination TRBJ2-6 TRBV9 TRBD1-2. Matched donors and recipients are in the same positions in the 2 figures. The low prevalence of this VDJ combination in the sample population could be causing the results of the test to be skewed.

Second the statistics were calculated using a mean donor distribution as a control for each VDJ combination. After adjusting for multiple comparisons using the Benjamini and Hochberg FDR method, there were three significant differences (FDR < 0.001). The distribution of the statistic for the three significantly different VDJ combinations can be seen in Figure 8. This is a measure of how different the No GVHD and GVHD groups are from each other. Table 3 shows the median perturbations and observed FDR for the three VDJ combinations. Note that the two analyses have no VDJ combinations in common.
Figure 7: Distribution of perturbation from matched donor statistic for the 2 significantly different VDJ combinations.
Figure 8: Distribution of perturbation from mean donor statistic for the 3 significantly different VDJ combinations.

Table 2: Median Perturbation Using Matched Donor Control

<table>
<thead>
<tr>
<th>JGene</th>
<th>VGene</th>
<th>DGene</th>
<th>Median No GVHD Perturbation</th>
<th>Median GVHD Perturbation</th>
<th>Observed FDR &lt; 0.05</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRBJ2-3</td>
<td>TRBV7-9</td>
<td>TRBD1-1</td>
<td>37.946</td>
<td>22.513</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>TRBJ2-6</td>
<td>TRBV9</td>
<td>TRBD1-2</td>
<td>50.000</td>
<td>60.310</td>
<td>&lt;0.001</td>
</tr>
</tbody>
</table>
Table 3: Median Perturbation Using Mean Control

<table>
<thead>
<tr>
<th>JGene</th>
<th>VGene</th>
<th>DGene</th>
<th>Median No GVHD</th>
<th>Median GVHD</th>
<th>Observed FDR &lt; 0.05</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRBJ2-7</td>
<td>TRBV7-9</td>
<td>TRBD1-1</td>
<td>47.001</td>
<td>14.364</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>TRBJ1-1</td>
<td>TRBV9</td>
<td>TRBD1-1</td>
<td>51.051</td>
<td>33.529</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>TRBJ2-1</td>
<td>TRBV7-7</td>
<td>TRBD1-2</td>
<td>46.154</td>
<td>23.571</td>
<td>&lt;0.001</td>
</tr>
</tbody>
</table>

Figure 9: Distributions of CDR3 lengths from VDJ combination TRBJ2-6 TRBV9 TRBD1-2 donors.
Figure 10: Distributions of CDR3 lengths from VDJ combination TRBJ2-6 TRBV9 TRBD1-2 recipients.
2.6 Oligoscores Results

The Oligoscore method described in section 1.6.2 was adapted for TCR sequencing data and applied to the study data. Each possible VDJ combination, \( f \), was examined individually. An Oligoscore was calculated for each CDR3 length, \( i \), within each VDJ combination using equation 13. The RPM were summed for each unique CDR3 length and then divided by the total RPM present within the VDJ combination of interest to calculate the CDR3 length probability. Further, the scores were calculated for both groups, GVHD and no GVHD, individually. Each CDR3 length, \( i \), was considered that ranged from 1 to \( n_k \), the number of peaks for sample \( k \). Because this is a measure to rank interesting peaks, the maximum Oligoscore from each group was examined graphically. Looking at Figure 11 we see that CDR3 length 39 shows up as a major peak in every patient with GVHD but only 2 of the No GVHD patients. Similarly, looking at Figure 12 we see that CDR3 length 33 shows up in 4 out of 5 patients with no GVHD and 3 out of 8 with GVHD. Further, Fisher’s exact test was calculated to test for a difference in number of subjects that had the two peaks as major peaks. Calculating the Fisher’s exact test of the peak with the highest oligoscore in the GVHD group showed a significant difference between the two groups (\( p=0.035 \)). Calculating the Fisher’s exact test of the peak with the highest Oligoscore in the no GVHD group showed a non-significant difference between the two groups (\( p=0.266 \)).
Figure 11: Distribution of CDR3 length for highest Oligoscore in GVHD group.
2.7 Simpson’s Diversity Index Results

The Simpson’s diversity index method described in section 1.6.3 was adapted for TCR sequencing data and applied to the study data. Each possible VDJ combination, $f$, was examined individually. There were 947 diversity indexes calculated for each sample using equation 14. Each CDR3 length, $i$, was considered that ranged from 1 to $L$. $L$ could vary for each VDJ combination, $n_{kf}$ was the total RPM for each sample within a VDJ combination and $n_{ikf}$ was the RPM for the $i^{th}$ CDR3 length in the $k^{th}$ sample of VDJ combination,
Permutation tests described in section 2.4 were used to test for differences in Diversity between the GVHD and no GVHD groups.

Simpson’s diversity indices were calculated for every VDJ combination in each sample. After adjusting for multiple comparisons using the Benjamini and Hochberg FDR method, there were five significant differences (FDR < 0.001). The distribution of the statistic for the five significantly different VDJ combinations can be seen in Figure 13. Table 4 shows the median diversity and observed FDR for the five VDJ combinations.

Figure 13: Distribution of Simpson’s Diversity Index statistic for the 5 significantly different VDJ combinations.
Table 4: Median Simpson’s Diversity

<table>
<thead>
<tr>
<th>JGene</th>
<th>VGene</th>
<th>DGene</th>
<th>Median No GVHD Diversity</th>
<th>Median GVHD Diversity</th>
<th>Observed FDR &lt; 0.05</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRBJ2-2</td>
<td>TRBV11-1</td>
<td>TRBD1-1</td>
<td>0.458</td>
<td>0.544</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>TRBJ2-6</td>
<td>TRBV5-5</td>
<td>TRBD1-1</td>
<td>0.470</td>
<td>0.034</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>TRBJ2-4</td>
<td>TRBV5-5</td>
<td>TRBD1-2</td>
<td>0.715</td>
<td>0.000</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>TRBJ1-3</td>
<td>TRBV5-6</td>
<td>TRBD1-2</td>
<td>0.649</td>
<td>0.155</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>TRBJ2-6</td>
<td>TRBV7-6</td>
<td>TRBD1-2</td>
<td>1.088</td>
<td>0.619</td>
<td>&lt;0.001</td>
</tr>
</tbody>
</table>
For comparison Shannon’s diversity indices (see section 1.5.2) were also calculated for every VDJ combination in each sample. After adjusting for multiple comparisons using the Benjamini and Hochberg FDR method, there were no significant differences (FDR > 0.453). However, there were 34 p-values before adjusting below the $\alpha = 0.05$ threshold. For illustrative purposes, we examined the 6 VDJ combinations with the smallest p-values for differences between the diversity index statistic. The distribution of the statistic can be seen in Figure 14. Table 5 shows the median diversity and p-values for the 6 VDJ combinations having the smallest p-values.
Figure 14: Distribution of Shannon’s Diversity Index statistic for the 6 VDJ combinations having the smallest p-values.

Table 5: Median Shannon Diversity

<table>
<thead>
<tr>
<th>JGene</th>
<th>VGene</th>
<th>DGene</th>
<th>Median No GVHD Diversity</th>
<th>Median GVHD Diversity</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRBJ2-4</td>
<td>TRBV20-1</td>
<td>TRBD1-2</td>
<td>-6722.642</td>
<td>-2237.406</td>
<td>0.005</td>
</tr>
<tr>
<td>TRBJ2-1</td>
<td>TRBV16</td>
<td>TRBD1-1</td>
<td>&gt;-0.001</td>
<td>-52.182</td>
<td>0.007</td>
</tr>
<tr>
<td>TRBJ1-6</td>
<td>TRBV20-1</td>
<td>TRBD1-2</td>
<td>-3413.701</td>
<td>-699.385</td>
<td>0.012</td>
</tr>
<tr>
<td>TRBJ1-1</td>
<td>TRBV10-3</td>
<td>TRBD1-2</td>
<td>-4.448</td>
<td>-562.484</td>
<td>0.013</td>
</tr>
<tr>
<td>TRBJ2-5</td>
<td>TRBV10-1</td>
<td>TRBD1-2</td>
<td>-100.164</td>
<td>-2.094</td>
<td>0.014</td>
</tr>
<tr>
<td>TRBJ2-5</td>
<td>TRBV5-1</td>
<td>TRBD1-1</td>
<td>-33045.878</td>
<td>-9146.991</td>
<td>0.015</td>
</tr>
</tbody>
</table>
2.8 Kullback-Leibler Divergence Results

The Kullback-Leibler (KL) Divergence method described in section 1.6.4 was adapted for TCR sequencing data and applied to the study data. Each possible VDJ combination, \( f \), was examined individually. There were 947 F-statistics calculated using equation 31 and the K-L divergence was calculated using equation 20. Each CDR3 length, \( i \), was considered that ranged from 1 to \( L \). \( L \) could vary for each VDJ combination. For the equation, \( n = 13 \) was the total number of subjects and \( G = 2 \) represents the GVHD and no GVHD groups. Further, \( \Sigma_k D(\bar{p}_k; \bar{p}_.) \) is the within group divergence and \( \Sigma_{jk} D(p_{jk}; \bar{p}_k) \) is the between group divergence. The K-L divergence was calculated using equation 20 and \( p_i \) is the overall relative abundance of CDR3 length \( i \). Group differences were tested using equation 31. After adjusting for multiple comparisons using the Benjamini and Hochberg FDR method, there were 947 significant differences. This is all possible differences so only the lowest p-values were looked at graphically. As Figures 15 and 16 show, there is considerable variability within a group in the CDR3 length distribution. Thus, an average distribution would not be very informative in this particular data set.
Figure 15: CDR3 length distribution for each patient of VDJ combination: TRBJ2-1, TRBV18, TRBD1-1.
2.9 Non-parametric Method Results

The non-parametric method described in section 1.6.5 was adapted for TCR sequencing data and applied to the study data. Each possible VDJ combination, \( f \), was examined individually. There were 947 \( \alpha \) values calculated for each subject, \( k \). Each CDR3 length, \( i \), was considered that ranged from 1 to \( L \). \( L \) could vary for each VDJ combination. First, standard deviation was calculated for each CDR3 length, \( i \), using equation 33, where the
probability of each CDR3 length was calculated by summing the RPM for each unique CDR3 length and then dividing by the total RPM present within the VDJ combination of interest. Only one step of variation was considered, the final TCR sampling step, so \( S = 1 \) and equation 34 was used to calculate the standard deviation. This standard deviation was then used to estimate the complementary cumulative frequency distribution from equation 32. An \( \alpha \) was then calculated using equation 37 by setting \( p = (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9) \) and taking the log of both sides to estimate a slope. Permutation tests described in section 2.4 were used to test for differences between the \( \alpha \) values of the GVHD and no GVHD groups.

The \( \alpha \) values were calculated for every VDJ combination in each sample. After adjusting for multiple comparisons using the Benjamini and Hochberg FDR method, there was one significant difference (FDR < 0.001). The distribution of the statistic for the significantly different VDJ combination can be seen in Figure 17. Table 6 shows the median \( \alpha \) values for the VDJ combination and the observed FDR.
Figure 17: Distribution of the non-parametric $\alpha$ statistic for the significantly different VDJ combination.

Table 6: Median Alpha Value

<table>
<thead>
<tr>
<th>JGene</th>
<th>VGene</th>
<th>DGene</th>
<th>Median No GVHD Alpha</th>
<th>Median GVHD Alpha</th>
<th>Observed FDR $&lt; 0.05$</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRBJ2-2</td>
<td>TRBV25-1</td>
<td>TRBD1-1</td>
<td>-0.032</td>
<td>-0.074</td>
<td>&lt;0.001</td>
</tr>
</tbody>
</table>
2.10 Proportion Logit Transformation

The procedure described in section [1.6.6] was applied to the study data. Recipient data was matched to donor data by amino acid sequences. Proportions for shared sequences were then calculated for each VDJ combination and logit transformed. Because logit transformed proportions are assumed to follow a Gaussian distribution, Student’s t-tests were applied to each VDJ combination to test for difference between the GVHD and no GVHD groups. After adjusting for multiple comparisons using the Benjamini and Hochberg FDR method, there were no significant differences found (FDR $> 0.659$). However, there were 2 p-values before adjusting below the $\alpha = 0.05$ threshold. For illustrative purposes, we looked at the 2 VDJ combinations with the smallest p-values for differences between the proportions. The distribution of the logit transformed proportions can be seen in Figure [18]. Table [7] shows the median logit transformed proportion values for each group and the corresponding p-values.
Figure 18: Distribution of the logit transformed proportion of shared sequences for the 2 VDJ combinations having the smallest p-values.

Table 7: Median of the logit transformed proportion of shared sequences

<table>
<thead>
<tr>
<th>JGene</th>
<th>VGene</th>
<th>DGene</th>
<th>Median No GVHD Proportion</th>
<th>Median GVHD Proportion</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRBJ2-6</td>
<td>TRBV19</td>
<td>TRBD1-1</td>
<td>5.496</td>
<td>0.271</td>
<td>0.033</td>
</tr>
<tr>
<td>TRBJ2-1</td>
<td>TRBV30</td>
<td>TRBD1-1</td>
<td>68.000</td>
<td>2.748</td>
<td>0.037</td>
</tr>
</tbody>
</table>
2.11 Discussion

This chapter looked at several different methods to analyze TCR data. Permutation tests were then used to compare GVHD and no GVHD groups. Four methods identified several significantly different VDJ combinations after adjusting for multiple comparisons. None of the these methods found the same significant VDJ combinations. This could be because the methods attempt to quantify different aspects of the T-cell repertoire. The CDR3 perturbation method measures either how different the recipient distributions are from the donor or from a control distribution. On the other hand, the diversity measure simply describes how diverse the recipient’s repertoire is at that point in time. Oligoscores simply look for major peaks within groups. The Kullback-Leibler divergence measures difference in distributions but assumes an underlying distribution so appears to be more sensitive to changes than the perturbation method. The non-parametric method assumes the distribution follows a power law and can be interpreted as a measure of diversity. Finally, the proportion method compares patient and donor similarity. Some of these methods measure similar aspects of the repertoire while some measure very different aspects. So the method that bests corresponds to the research question should be chosen. For this data set, Simpson’s diversity index would be an appropriate choice since T-cell diversity has been shown to be a good measure of T-cell repertoire health.
3 Compositional Data in Bioscience

In the previous chapters several different methods were applied to high-throughput sequencing data. However, none of these methods consider the unit sum constraint on the data. This constraint can have large effects on data analysis and data of this type is called compositional. This chapter will describe what effects compositional data can have on analysis and how to analyze compositional data.

3.1 Introduction

To motivate what compositional data are and how they can affect data analyses consider two scenarios that [Pawlowsky-Glahn and Buccianti, 2011] described in their textbook on compositional data analysis. The first scenario pertains to counting the number of different messenger RNAs (mRNA) that emerge from the nucleus of a cell in a given time interval. The production of one type of mRNA does not necessarily affect the production of another mRNA. This means each mRNA can be considered independent of another mRNA and traditional analytical approaches are appropriate. The second scenario pertains to counting total mRNA within a cell that has reached its maximum capacity of mRNA. If one type of mRNA increases within the cell then another mRNA must decrease by some mechanism such as degradation. This is the sum-constrained property of compositional data that can have large effects on both the information gleaned and the interpretation of results.

There are quite a large amount of compositional data being generated in using “omics” technologies (genomics, transcriptomics, etc.) but little awareness to the fact that these are compositional data and should be treated as such. A lot of ground has been gained in the geosciences in dealing with compositional data [Pawlowsky-Glahn and Buccianti, 2011]. However, there are some big differences between the subject matters, the main one being that sequencing data produces compositions with hundreds-of-thousands of components whereas geoscience data usually produces tens to hundreds of components. This lower dimensionality
makes the data much easier to work with. For example, we can look at TCR sequencing data which appears to provide the absolute abundance of different TCR transcripts but, in reality, TCR sequencing provides only relative information for the specific sample. Note that depending on when and how the sample is prepared the sequencing can be dominated by several specific transcripts. This can be seen in experiments performed by van Heijst et al. [2013] in monitoring T-cell repertoire recovery after HSCT over time. In this case, one patient had a large spike in a TCR related to Epstein-Barr virus at day 145 which was undetectable at days 138 and 194. Furthermore, low frequency transcripts can be highly affected by changes in high frequency transcripts because only a limited number of transcripts can be sequenced. Even worse, if an experiment is repeated, the subset of low frequency transcripts detected can change with each run [Robins et al. 2009]. During TCR sequencing, a fixed amount of sample is procured and then a fixed amount of that sample is sequenced. The data are then “standardized” (ie reads per million) which results in a relative abundance. This constraint has not always been taken into account in current analytical practices. In the following sections, we will define compositional data and propose that compositional methods are more appropriate to apply to sequencing data, specifically TCR sequencing.

3.2 Definition of Compositional Data

Compositional data are defined as data where the elements of the composition are non-negative and sum to unity. This data structure often arises from non-negative data, such as counts, that has been scaled by the total of the components [Pawlowsky-Glahn and Buccianti 2011]. Compositional data analysis was most recently defined by Egozcue [2009]. The formulation he proposes is separated into a definition and three principles for a total of four parts.

**Definition** A compositional vector, or simply a composition, of $D$ parts is a positive real vector of $D$ components, describing quantitatively the parts of some whole, which carry exclusively relative information between the parts.
**Principle A** Scale Invariance: The information in a composition does not depend on the particular units in which the composition is expressed. Proportional positive vectors represent the same composition. Any sensible characteristic of a composition should be invariant under scale.

**Principle B** Permutation Invariance: Permutation of components of a composition does not alter information conveyed by the compositional vector.

**Principle C** Subcompositional Coherence: Information conveyed by a composition of $D$ parts should not be in contradiction with that coming from a sub-composition containing $d$ parts, $d \leq D$.

### 3.3 Representations of a Composition

To represent a composition so that the representation follows the principles in section 3.2, Aitchison [1986] proposed that all compositions be treated on a log transformed scale. This allows for most classical multivariate methods to be applied to compositional data. Log contrasts are scale invariant log ratios generically defined as

$$\sum_{i=1}^{D} \alpha_i \log(x_i), \quad \text{where} \Sigma_{i=1}^{D} \alpha_i = 0 \quad (40)$$

where $\alpha_i$ is a transformation coefficient, $x_i$ is the compositional data and $D$ is the size of the simplex. The condition on the coefficients guarantees scale invariance. Three different transformations are proposed [Pawlowsky-Glahn and Buccianti 2011] for use depending on the hypothesis to be tested. The first is called the additive-log-ratio transformation (alr). If $x$ is a composition in the $D$-part simplex $S^D$,

$$alr(x) = \log \left( \frac{x_1}{x_D}, \frac{x_2}{x_D}, ..., \frac{x_{D-1}}{x_D} \right) \quad (41)$$
where the natural logarithm is applied componentwise. The transformation is easily inverted to get the composition back. However, it is not invariant under permutation of components which may cause problems with some statistical tests. To remedy this problem, [Aitchison 1986] proposed another transformation called the centered log-ratio transformation (clr) that is defined as

$$clr(x) = \log\left(\frac{x_1}{g_m(x)}, \frac{x_2}{g_m(x)}, \ldots, \frac{x_D}{g_m(x)}\right), g_m(x) = \left(\prod_{i=1}^{D} x_i\right)^{\frac{1}{D}}$$

(42)

where $g_m$ is the geometric mean. However, in this case the components change when working with a subcomposition (defined in section 3.2) which is a violation of principle C. The components of a subcomposition using the clr transformation change because the geometric mean changes. Because of the sum-constrained property of compositional data, a D-part composition can only be represented with D-1 coefficients but the clr transformation tries to do this using D coefficients. The clr representation can be used as a measure of distance, called the Aitchison distance, that is discussed in more detail in section 3.4. The final representation is called the isometric log-ratio transformation (ilr) [Egozcue et al., 2003]. The major differences between the clr and ilr transformation are that the ilr transformation results in D-1 components and there are procedures that allow for this transformation to be easily interpretable. Simply, for the ilr transformation we are looking for an orthonormal basis of $S^D$ using a set of compositions, $e_1, e_2, \ldots, e_{D-1}$ such that $\langle e_i, e_j \rangle_a = 0$ for $i \neq j$, and $\|e_i\|_a = 1$. Where $\langle \cdot, \cdot \rangle_a$ and $\| \cdot \|_a$ are the Aitchison inner product and norm. Where $\langle x, y \rangle_a = \langle clr(x), clr(y) \rangle$ and $\|x\|_a = \|clr(x)\|$ and $\| \cdot \|$ and $\langle \cdot, \cdot \rangle$ are the Euclidean norm and inner product respectively. Note, a full discussion of the Aitchison metric is presented in Section 3.5. Using this terminology the ilr transformation is defined as

$$ilr(x) = (\langle x, e_1 \rangle_a, \langle x, e_2 \rangle_a, \ldots, \langle x, e_{D-1} \rangle_a).$$

(43)
3.4 Statistical Modelling

Compositional data do not lend themselves well to traditional statistical approaches. Issues such as spurious correlations can affect analytical results. For this reason some kind of transformation is necessary prior to performing statistical analysis. Further, compositional data are not part of the natural sample space and do not follow Euclidean geometry. The observations can be moved to a new sample space or represented using an orthonormal basis [Pawlowsky-Glahn and Buccianti 2011]. The “move method” is the log-ratio transformation method and the “stay method” includes the alr, clr and ilr transformations that were mentioned in section 3.3. These transformations place the compositional data in terms of its basis and corresponding coordinates. These coordinates, based on linear algebra theory, are part of the real space and standard multivariate techniques can be applied to them.

The simplex can be defined as follows: Let \( x = (x_1, x_2, ..., x_D) \) denote a D-part composition, then

\[
S^D = \left\{ x = (x_1, x_2, ..., x_D) : x_i > 0 \ (i = 1, 2, ..., D), \sum_{i=1}^{D} x_i = \kappa \right\}
\]  

(44)

where \( \kappa \) is a positive constant that is typically 1 or 100 depending on how the composition is designed. The simplex has two basic operations called perturbation and powering as its internal and external operations.

\[
x \oplus x^* = C(x_1 x_1^*, ..., x_D x_D^*),
\]  

(45)

\[
\alpha \odot x = C(x_1^\alpha, ..., x_D^\alpha),
\]  

(46)

where \( C \) denotes the closure operation that normalises any vector to a constant sum. Furthermore, we can define an inner product, norm and distance to define a (D-1) Euclidean vector space structure on the simplex;
\[(x, x^*)_a = \frac{1}{D} \sum_{i<j} \log \frac{x_i}{x_j} \log \frac{x_i^*}{x_j^*}; \quad (47)\]

\[\| x \|_a = \sqrt{\frac{1}{D} \sum_{i<j} \left( \log \frac{x_i}{x_j} \right)^2}; \quad (48)\]

\[d_a(x, x^*) = \sqrt{\frac{1}{D} \sum_{i<j} \left( \log \frac{x_i}{x_j} - \log \frac{x_i^*}{x_j^*} \right)^2}; \quad (49)\]

Note that these equations are applied to the original simplex and are called the Aitchison geometry, hence the \(a\) subscript.

An appropriate basis for the simplex needs to be determined. Because compositions are a subset of the real space they can be expressed in terms of the canonical basis. However, this does not follow Euclidean space rules. Egozcue et al. [2003] suggested one orthonormal basis \(\{e_1, e_2, ..., e_{D-1}\}\);

\[e_1 = C \left( \exp \left( \frac{1}{\sqrt{i(i+1)}} \right),..., \exp \left( \frac{1}{\sqrt{i(i+1)}} \right), \exp \left( -\sqrt{\frac{i}{i+1}} \right), 1, ..., 1 \right) \quad (50)\]

and coordinates

\[x = (y_1 \odot e_1) \oplus (y_2 \odot e_2) \oplus ... \oplus (y_{D-1} \odot e_{D-1}), \quad (51)\]

with

\[y_i = \frac{1}{\sqrt{i(i+1)}} \log \left[ \frac{x_1 x_2 ... x_i}{(x_{i+1})^i} \right], \quad i = 1, 2, ..., D - 1. \quad (52)\]

Once the basis is selected typical multivariate methods can be applied such as Principal Component Analysis.
3.5 Current Practices and Their Pitfalls

TCRβ sequencing data are multivariate. Most methods applied to sequence data are univariate, such as the t-tests (moderated), with extra procedures that adjust for multiple-testing. However, the effect of the sum constraint on univariate analysis results is questionable. Consider a simple thought exercise and imagine how the sum constraint can affect univariate statistics. Specifically, for a simple two-part composition, \( x = (x_1, x_2) \) derived from a basis \( w = (w_1, w_2) \) through its closure:

\[
x = C(w) = \left( \frac{w_1}{w_1 + w_2}, \frac{w_2}{w_1 + w_2} \right).
\]

Consider two different scenarios that have the same result. First, consider what is necessary for \( w_2 \) to double \( x_2 \) if \( w_1 \) is fixed,

\[
w' = (w_1, w'_2)
\]

\[
x' = (x'_1, x'_2) = (x'_1, 2x_2)
\]

\[
\therefore \frac{w'_2}{w_1 + w'_2} = 2 \cdot \frac{w_2}{w_1 + w_2}
\]

\[
w'_2 = 2w_2 \cdot \frac{w_1}{w_1 - w_2}
\]

\[
\approx 2w_2, \text{ if } w_1 >> w_2
\]

Next, consider what is necessary for \( w_1 \) if we fixed \( w_2 \) to double \( x_2 \).

\[
w' = (w'_1, w_2)
\]
\[
x' = (x'_1, x'_2) = (x'_1, 2x_2)
\]

\[
\therefore \frac{w'_2}{w'_1 + w_2} = 2 \cdot \frac{w_2}{w_1 + w_2}
\]

\[
w'_1 = \frac{1}{2}(w_1 - w_2)
\]

\[
\approx \frac{1}{2}w_1, \text{ if } w_1 >> w_2
\]

This example demonstrates that there are two different mechanisms that can result in a doubling of \(x_2\), but we can not determine from the data alone which mechanism produced such a doubling. When working with a composition that consists of more than two parts, such as TCR\(\beta\) sequencing, \(x_1\) can be considered as an aggregation of everything except \(x_2\). In this scenario, “\(x_2 \rightarrow kx_2\)” implies \(w_2 \rightarrow kw_2\)” only if certain conditions apply:

1. \(x_2\) is a relatively small component

2. the rest of the basis stays the same

3. \(\log k\) is small

These three conditions are what, in practice, occur in ‘spike-in’ experiments but not necessarily in sequencing experiments. ‘Spike-in’ experiments are when a known concentration is added to an unknown mixture usually to assess assay sensitivity. \(x_2\) represents the ‘spike-in’ portion of the sample and is usually small compared to the rest of the sample. The known mixtures are tested while the rest of the sample stays unchanged. In this case, it is safe to use univariate statistics since the compositions are not changing dramatically and follow the three conditions above. It can be assumed that samples in sequencing experiments are much
more variable than in controlled ‘spike-in’ experiments. Thus, using these kinds of experiments as proof of concept to be applied to more variable data, such as TCRβ sequencing, can be misleading.

Multivariate methods are commonly applied in the analysis of high-throughput data. Specifically, clustering methods are applied to identify homogeneous groups. Clustering methods require that a distance measure be calculated between samples. Lovell et al. [2010] described four different distance metrics that can be used as measures of distance for multivariate methods: the Aitchison distance, Euclidean distance, the Euclidean distance between log values, and the Kullback-Leibler divergence. Knowing how these metrics behave can help select the appropriate metric for analysis.

Consider two compositions, \( x \) and \( y \), with positive components \( x_i \) and \( y_i \), where \( i = 1, ... D \) and \( D \) is the number of components of the composition. For TCRβ sequencing the \( D \) components are the unique nucleotide sequences, unique amino acid sequences or unique CDR3 lengths. The analysis will depend on the research question and can be done for the overall components or separated by VDJ family. Their perturbation difference, \( z = x \ominus y \), is defined by

\[ z = C \left( \frac{x_1}{y_1}, \frac{x_2}{y_2}, ..., \frac{x_D}{y_D} \right) \]

where \( C \) is the closure.

The Aitchinson distance can be written as
\[ d_a(\mathbf{x}, \mathbf{y}) = \sqrt{\| \text{clr}(\mathbf{x}) - \text{clr}(\mathbf{y}) \|} \] (53)

\[ = \sqrt{\| \text{clr}(\mathbf{z}) \|} \] (54)

\[ = \sqrt{\sum_i (\log \frac{x_i}{g_m(\mathbf{x})} - \log \frac{y_i}{g_m(\mathbf{y})})^2} \] (55)

\[ = \sqrt{\sum_i (\log \frac{z_i}{g_m(\mathbf{z})})^2} \] (56)

where \( g_m \) is the geometric mean. By writing the Aitchinson distance in terms of \( \mathbf{z} \), it can be seen that the Aitchinson distance is a measure of relative distances. In comparison, the Euclidean distance between \( \mathbf{x} \) and \( \mathbf{y} \) is written as

\[ d_e(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_i (x_i - y_i)^2} \] (57)

\[ = \sqrt{\sum_i \left(\frac{x_i}{y_i} - 1\right)^2 y_i^2} \] (58)

Notice that this distance depends not only on the ratio of the two compositions but on one of the compositions itself. Also, the Euclidean distance is bounded by \( \sqrt{2} \) whereas the Aitchison distance has no upper limit. The Aitchison distance would do a much more appropriate job because it is not constrained by an upper limit and can give more information about relative differences. Further, the Aitchinson distance is not constrained to lie on a simplex and does not have a sum-constraint.

High-throughput genomic data are frequently \( \log_2 \) transformed. For illustrative purposes [Pawlowsky-Glahn and Buccianti 2011] showed how the Aitchison distance compares to the Euclidean distance between logged values. The Euclidean distance between log transformed values is
\[ d_e(\log(x), \log_2(y)) = \sqrt{\sum_i (\log(x_i) - \log(y_i))^2} \]  
(59)

\[ = \sqrt{\sum_i \left( \log\left( \frac{x_i}{g_m(x)} \right) - \log\left( \frac{y_i}{g_m(y)} \right) + \log\left( \frac{g_m(x)}{g_m(y)} \right) \right)^2} \]  
(60)

\[ = \sqrt{d_a^2(x, y) + D \log^2 \left( \frac{g_m(x)}{g_m(y)} \right)} \]  
(61)

\[ \geq \sqrt{d_a^2(x, y)} \]  
(62)

This is very similar to Aitchison’s distance with an extra component, \( D \log^2 \left( \frac{g_m(x)}{g_m(y)} \right) \).

Because high throughput sequencing data are typically \( \log_2 \) transformed, similar results should be observed when comparing analyses performed using Aitchison’s distance and the \( \log_2 \) transformed Euclidean distance [Lovell et al., 2010].

Finally, the Kullback-Leibler (K-L) divergence, which is not specifically a distance measure [Lovell et al., 2010], is typically used to measure the difference between two probability distributions. Distributions are similar to compositions in the sense that they sum to 1. Therefore, the K-L divergence is another method that can be used when comparing compositions. The K-L divergence between compositions depends on the ratios of the composition components, their absolute values of components, and the dimensionality of the compositions. Recall that the K-L divergence (see Section 1.6.4) between discrete probability distribution \( P \) and probability distribution \( Q \), both indexed by \( i = 1 \) to \( D \) where \( D \) is the size of the simplex, is

\[ D_{KL}(P|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \]  
(63)

The effect that compositional data has on estimates of covariance and correlation should be used to guide the decision of which metric to use. [Lovell et al., 2010] showed the relationship between the log of two components in a composition. Consider two basis vectors...
and their closures

\[ \mathbf{w} = (w_1, w_2) \]

\[ \mathbf{W} = (W_1, W_2) \]

\[ \mathbf{x} = \mathcal{C}(\mathbf{w}) \]

\[ \mathbf{X} = \mathcal{C}(\mathbf{W}). \]

The following result will also be used

\[ \text{cov}(A + C, B + C) = \text{cov}(A, B) + \text{cov}(A, C) + \text{cov}(B, C) + \text{var}(C). \]

Applying this result to the log of the compositions and the log of the basis we get

\[ \text{cov}(\log x_1, \log x_2) = \text{cov}(\log(\frac{w_1}{t}), \log(\frac{w_2}{t})) \]

\[ = \text{cov}(\log w_1 - \log t, \log w_2 - \log t) \]

\[ = \text{cov}(\log w_1, \log w_2) - \text{cov}(\log w_1, \log t) - \text{cov}(\log w_2, \log t) + \text{var}(\log t) \]

where \( t \) is the size of the basis. Thus, the covariance of the log of two components of interest in a composition is equal to the covariance between the log of those components in the basis plus or minus some terms related to the size of the basis. If the terms related to the size of the basis are large, the covariance estimates could become very different between using the components and the basis. Similarly, since correlations are functions of variance and covariance they can also be very different between compositional data and those in the
basis. By definition the correlation is defined as

\[
\text{corr}(A + C, B + C) = \frac{\text{cov}(A + C, B + C)}{\sqrt{\text{var}(A + C) \text{var}(B + C)}}
\]

Unfortunately, there is no nice representation of correlation to show how the components and the basis are related.

### 3.6 Treatment of Zeros in Compositional Data

As previously described, compositional data analysis is performed using a log ratio scale. Therefore, consideration must be given with respect to how to appropriately handle observed zeros. Aitchison et al. [2003] identified three processes that may give rise to observed zero values. The process that gave rise to the zero values should guide the analyst as to how to deal with the zero values. First, a zero value may truly represent a zero in the data. This is known as a structural zero or essential zero. Second, a zero might occur because of an underlying discrete process such as space on a sequencing chip. For example, in TCR sequencing data, CDR3 sequences with low counts may not be sequenced because clonotypes with high frequency used up all the space on the chip. Third, zeroes can occur because values fall below the limit of detection.

Rounded zeros are zeros when very small values are rounded to zero because of rounding error or the value falls below the limit of detection. In compositional data, these zeros fall under the category of not missing at random (NMAR) and require special models. Generally, methods for working with rounded zeroes can be classified into either non-parametric or parametric techniques. The non-parametric methods are based on different imputation methods. On the other hand, the parametric methods rely on parametric models such as the normal distribution on the simplex.

Counts are values that occur when the number of times an event occurs is measured. In the case of TCR sequencing data the number of times a particular CDR3 sequence occurs is
being counted. Since TCR sequencing data are transformed into RPM (reads per million), a measure of relative abundance, they should be considered in the compositional framework. Pierotti et al. [2009] suggest a Bayesian-multiplicative approach.

The multiplicative part of the approach is based on an imputation strategy for rounded zeros [Martín-Fernández et al., 2003]. This strategy involves replacing the zeros with an appropriately small value $\delta_j$ and then modifying the non-zeros in a multiplicative way, where $j = 1, ..., n$ the number of observations. The specific modification is based on the sum constraint for imputation data. Let us consider a $D$-composition, $x \in S^D$ that has rounded zeros. This composition is replaced by a new composition $r \in S^D$ according to the formula

$$r_j = \begin{cases} 
\delta_j & \text{if } x_j = 0 \\
x_j \left(1 - \frac{\sum_{k: x_k = 0} \delta_k}{c}\right) & \text{if } x_j > 0
\end{cases} \quad (64)$$

where $c$ is the constant for the sum constraint and $\delta_j$ is a chosen value below the limit of detection. This method mixed with a Bayesian approach can be used to impute zero values in count data.

Now let $x$ be a count vector with $D$ categories in data set $X$. Let $N$ be the total count in $x$ and $\theta$ be its vector of probabilities from a multinomial distribution. Then the prior distribution for $\theta$ is a Dirichlet iid parameter vector $\alpha$ where $\alpha_k = st_k$, where $k = 1, ..., D$. The vector $t$ is the prior expectation for $\theta$ and $s$ is known as the strength of that prior. The parameter $s$ can be chosen to be between 0 and 1. From Bayes theorem the posterior estimation for $\theta$ is

$$\hat{\theta} = \frac{x_k + st_k}{N + s} \quad (65)$$

There are several different proposed priors but all come from the Dirichlet distribution. Then the chosen prior can be used to apply the multiplicative part of the method to actually impute zero values using a modification of the rounded zeros formula.
$$r_j = \begin{cases} \frac{\alpha_j}{N+s} & \text{if } x_j = 0 \\ x_j \left(1 - \sum_{k|x_k=0} \frac{\alpha_j}{N+s}\right) & \text{if } x_j > 0 \end{cases}$$ (66)

In this way we are able to preserve the ratios of the non-zero values while having a data set that contains no zeros. This new data set also preserves the unit constraint.

### 3.7 Application of Compositional Methods to TCR data

The data described in section 2.3 were treated as compositional data. All compositional calculations were done using the coin package in the R programming environment [Zeileis et al., 2008]. First each VDJ combination was treated as one part of a 947 part composition. The top 10% most variable VDJ combinations were considered for clustering. Clustering was then done using the Aitchison distance and Ward’s minimum variance method. Figure 19 shows the results of the clustering. Notice that other than the 2 No GVHD patients there seems to be no obvious grouping pattern.

Next, the CDR3 length distribution within each VDJ combination was considered as an $L$-part compositions. To compare groups linear models of the following form were run

$$Y_i = a \ominus X_i \odot b + \epsilon_i$$ (67)

$$\text{ilr}(Y_i) = \text{ilr}(a) + X_i\text{ilr}(b) + \text{ilr}(\epsilon_i)$$ (68)

where $a$ and $b$ are compositional constants, $Y_i$ is a random composition, $X_i$ is the group indicator and $\epsilon_i$ is a compositional random variable with compositional expectation $(1, ..., 1)/L$ and constant variance [Zeileis et al. 2008]. There were 947 models run to compare groups at each VDJ combination using this model. Zero values were replaced by a below limit of detection value of $1e - 6$. After adjusting for multiple comparisons using the Benjamini and
Hochberg FDR method, there were no significant differences found (FDR > 0.534). However, there were 20 p-values before adjusting below the $\alpha = 0.05$ threshold. Table 8 shows the 20 p-values.

Table 8: Compositional Linear Model p-values

<table>
<thead>
<tr>
<th>JGene</th>
<th>VGene</th>
<th>DGene</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRBJ2-3</td>
<td>TRBV10-2</td>
<td>TRBD1-1</td>
<td>0.001</td>
</tr>
<tr>
<td>TRBJ2-4</td>
<td>TRBV5-6</td>
<td>TRBD1-1</td>
<td>0.001</td>
</tr>
<tr>
<td>TRBJ2-4</td>
<td>TRBV28</td>
<td>TRBD1-1</td>
<td>0.002</td>
</tr>
<tr>
<td>TRBJ2-7</td>
<td>TRBV11-1</td>
<td>TRBD1-2</td>
<td>0.005</td>
</tr>
<tr>
<td>TRBJ2-5</td>
<td>TRBV6-1</td>
<td>TRBD1-2</td>
<td>0.005</td>
</tr>
<tr>
<td>TRBJ2-3</td>
<td>TRBV4-2</td>
<td>TRBD1-2</td>
<td>0.006</td>
</tr>
<tr>
<td>TRBJ2-4</td>
<td>TRBV7-6</td>
<td>TRBD1-1</td>
<td>0.010</td>
</tr>
<tr>
<td>TRBJ2-5</td>
<td>TRBV27</td>
<td>TRBD1-2</td>
<td>0.014</td>
</tr>
<tr>
<td>TRBJ2-6</td>
<td>TRBV11-2</td>
<td>TRBD1-2</td>
<td>0.015</td>
</tr>
<tr>
<td>TRBJ2-3</td>
<td>TRBV28</td>
<td>TRBD1-2</td>
<td>0.019</td>
</tr>
<tr>
<td>TRBJ2-5</td>
<td>TRBV4-3</td>
<td>TRBD1-1</td>
<td>0.022</td>
</tr>
<tr>
<td>TRBJ2-7</td>
<td>TRBV27</td>
<td>TRBD1-2</td>
<td>0.022</td>
</tr>
<tr>
<td>TRBJ2-7</td>
<td>TRBV7-3</td>
<td>TRBD1-2</td>
<td>0.027</td>
</tr>
<tr>
<td>TRBJ2-5</td>
<td>TRBV13</td>
<td>TRBD1-1</td>
<td>0.029</td>
</tr>
<tr>
<td>TRBJ2-1</td>
<td>TRBV5-5</td>
<td>TRBD1-1</td>
<td>0.032</td>
</tr>
<tr>
<td>TRBJ2-7</td>
<td>TRBV14</td>
<td>TRBD1-1</td>
<td>0.033</td>
</tr>
<tr>
<td>TRBJ2-5</td>
<td>TRBV9</td>
<td>TRBD1-1</td>
<td>0.034</td>
</tr>
<tr>
<td>TRBJ2-5</td>
<td>TRBV27</td>
<td>TRBD1-1</td>
<td>0.035</td>
</tr>
<tr>
<td>TRBJ2-1</td>
<td>TRBV16</td>
<td>TRBD1-1</td>
<td>0.037</td>
</tr>
<tr>
<td>TRBJ2-5</td>
<td>TRBV28</td>
<td>TRBD1-1</td>
<td>0.049</td>
</tr>
</tbody>
</table>
3.8 Discussion

In this chapter we discussed what compositional data are and why sequencing data falls in that category. The compositional methodology was then applied to the TCR sequencing data described in section 2.3. The compositional models identified different VDJ combinations compared to all the methods applied in chapter 2. This shows that choosing the appropriate analysis strategy can have large effects on results. The research question of interest in this study was to identify VDJ combinations that differ between the GVHD and no GVHD
groups. This seems to best be addressed by Simpson’s Diversity measure since it has a biological interpretation that more diversity is correlated with a healthier TCR repertoire. On the other hand, tests such as the Kullback-Leibler divergence method test for the equality of individual peak distributions which is not really the research question. It was of further interest to determine if the different methods identified similar V or J genes individually instead of specific VDJ combinations. Table 9 shows a summary of V or J genes that were identified by multiple methods.

Table 9: Common V or J Gene Using Different Analyses

<table>
<thead>
<tr>
<th>Gene</th>
<th>Individual Perturbation</th>
<th>Mean Perturbation</th>
<th>Simpson’s Diversity</th>
<th>Compositional Analysis</th>
<th>Alpha Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRBJ2-6</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>TRBJ1-1</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>TRBJ2-1</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>TRBJ2-2</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>TRBJ1-3</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>TRBV7-9</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>TRBV9</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRBV7-7</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>TRBV7-6</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>TRBV25-1</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
4 Generalized monotone incremental forward stagewise method for modeling count data: Application predicting micronuclei frequency

This chapter considers count data that are left as counts and not transformed like sequencing data. Specifically, this chapter looks at a method to use high-throughput data to predict a count response. Because many methodologies cannot be applied to scenarios where there is a larger number of predictors than samples, a generalized monotone forward stagewise (GMIFS) method is considered.

4.1 Introduction

Micronuclei (MN) are small nuclear bodies that are formed in cells that are in the process of division but are not part of the nucleus. Therefore MN can only be found in cells that have undergone nuclear division at least once and appear as small extranuclear bodies. When two daughter nuclei are formed during cell division, these bodies are placed into a smaller nucleus not part of the main nuclei, hence the term micronuclei [Kirsch-Volders et al., 2011]. Once the micronuclei are formed the cell has several different response options. MN can stay within the cell, if it has functional DNA, as its own entity or be reabsorbed into the main nucleus. If the DNA is non-functional the MN may be expelled from the cell or the whole cell may be destroyed through apoptosis. Since MN can be expelled from the cell, they can be used as a mechanism to remove extra chromosomes from the cell [Kirsch-Volders et al., 2011].

Micronuclei can form spontaneously or they can be induced by mutagens. Some spontaneous MN are actually beneficial to the organism. An example is in the mouse cerebral cortex where MN formation adds diversity to the nervous system [Kirsch-Volders et al., 2011]. However, the large majority of MN are caused by mutagens and may play a role in carcino-
genesis. Depending on the fate of the MN, the end result could be a variety of different DNA and chromosome cell contents. This variety could result in an accumulation of DNA changes and instability that could result in cancer \cite{Kirsch-Volders2011}. Several studies have shown that higher MN counts result in a higher risk of cancer in the future \cite{Kirsch-Volders2011}. Thus, using the cytokinesis-block micronucleus (CBMN) assay as a risk assessment for cancer has potential clinical benefits. Further, combining CBMN with other high-throughput technologies such as gene expression and methylation analyses may help identify factors related to micronucleation.

Quantifying MN in patient samples has been shown to be a good measure of genetic damage. MN scoring, counting the number of MN present in a sample, is a popular tool for testing genotoxicity mostly because of its simplicity, accuracy, applicability to different cell types and ease of automation. Cancer cells show a loss of genetic control which can be caused by DNA damage so they are a good candidate for MN testing. The CBMN assay has successfully been used and validated to score MN. The CBMN assay uses Cytochalasin-B which stops cells from performing cytokinesis but does not stop nuclear division giving rise to cells that are binucleated \cite{Fenech1993,Fenech2000}. Furthermore, the Organisation for Economic Co-operation and Development (OECD) has developed a set of guidelines for running the CBMN assay to have the most consistent and reliable results \cite{Kirsch-Volders2011}.

Guidelines for the process of scoring MN have been presented by the HUman Micronucleus (HUMN) project. This is an international collaborative project aimed at improving the application of the CBMN assay. One of the HUMN projects main goals is to identify methodological variables in the scoring of the assay to minimise confounding effects \cite{Fenech2003}. The HUMN project compiled 6583 subjects from 25 laboratories from 16 countries and looked at background MN frequency using the CBMN assay. The goal of the study was to identify variables that affect the background MN frequency. Scoring criteria was found to account for 47% of the observed variability thus standardized scoring criterion were
developed and described by Fenech et al. [2003]. The guideline includes scoring 2000 cells in order to accurately estimate MN frequency.

Because these guidelines were developed for assay performance they do not address how to statistically analyze the data generated by the assay. This has led to the application of various statistical methods that may render different interpretations and conclusions. In a review article examining analytical methods, Ceppi et al. [2010] reviewed 63 studies that statistically analyzed MN data and developed recommendations for selecting an appropriate analytical method. The review included studies that applied both parametric and non-parametric tests. The non-parametric tests included Kruskall-Wallis, Friedman, Wilcoxon and Mann-Whitney U-tests. Although these tests do not require an underlying distributional assumption, they are unable to adjust for confounding factors. There were a variety of parametric tests performed that assume normality such as ANOVA, ANCOVA, and multivariable linear models which can adjust for confounding factors. Other methods such as correlations and Student’s t-test were also used. However, applying these methods to MN data, which are rarely normally distributed, could result in inappropriate inferences. Although the data could be transformed to better adhere to a Gaussian distribution prior to applying such parametric tests, few studies applied any type of transformation. Further, Student’s t-tests and Pearson’s correlation cannot adjust for confounding variables. The common non-Gaussian models used were log-linear, Poisson, negative binomial and logistic regression. The logistic and log-linear models account for categories whereas Poisson and negative binomial directly model count data. For this reason Ceppi et al. [2010] recommend using negative binomial or Poisson models for MN data analysis. Another advantage of these count models is that they can adjust for confounding variables such as age, gender and smoking status. Finally, Ceppi et al. [2010] recommended 2000 or more cells be scored for best model performance. If less than 2000 cells are scored, a zero-inflated Poisson model is recommended. Ceppi et al. [2010].

When trying to identify molecular features related to MN frequency, high-throughput
genomic assays can be used. However, the previously described methods cannot be applied in settings where there are more predictor variables than samples. Therefore, in this section we extended the generalized monotone forward stagewise (GMIFS) method to the Poisson regression setting and apply it to a breast cancer study where we were interested in predicting MN frequency using features from the Illumina HumanMethylation 450K assay and to a cord blood data set where we were interested in predicting MN frequency using features from the Agilent 4x44k human oligonucleotide microarray.

4.1.1 Cytokinesis-block Micronucleus Assay

Micronuclei assays can only be effective if dividing cells can be identified. This is because MN only occur in cells that have undergone at least one nuclear division after DNA damage has occurred. Cytokinesis is the step in the cell division process that involves the splitting of the cytoplasm to form two cells. Cytochalasin-B stops cells from performing cytokinesis but does not stop nuclear division giving rise to cells that are binucleated [Fenech, 1993, Fenech et al., 1999]. However, only cells in the sample that are going to divide will take on a binucleated appearance. The sample is heterogeneous with cells that tried to divide and those that did not. Thus, there will be a mixture of mononucleated cells along with binucleated cells. Mononucleated cells in the CBMN assay can arise from several possibilities. There could be cells that never divide, cells which replicate DNA but escape nuclear division, cells that escape the cytochalasin B block or cells that are in an early stage of apoptosis or necrosis [Kirsch-Volders and Fenech, 2001]. Thus it can be considered that mononucleated cells indicate chromosome damage in vivo, present prior to the assay being done, and binucleated cells indicate DNA damage present before culture as well as damage expressed in vitro during culture. Thus, what is scored is dependent on the protocol followed and what type of hypothesis is being tested.
4.2 Methods

4.2.1 Statistical Methods

There are many available methods that can model count data. However, these methods require independence of explanatory variables \((p)\) and that the number of samples \((n)\) does not exceed the number of explanatory variables. The incremental forward stagewise regression method for linear regression and the Generalized Monotone Incremental Forward Stagewise (GMIFS) for a logistic regression model have been previously described [Hastie et al., 2007]. The GMIFS method for modeling ordinal response data has also been described [Archer et al., 2014]. To motivate our extension to the Poisson regression setting, we first review Poisson regression. We subsequently describe our GMIFS method for fitting Poisson regression models when \(n < p\).

4.2.2 Poisson Regression

Poisson regression is commonly used to model count data. Let \(i = 1, ..., n\) be the number of observations and \(y_i\) represent a Poisson distributed random variable. Let the expected value of \(y_i\) be written as

\[
\mathbb{E}(y_i) = \lambda_i.
\]

Then the conditional probability is given by

\[
P(y_i|\lambda_i) = \frac{e^{-\lambda_i} \lambda_i^{y_i}}{y_i!}
\]

for each observation \(i\). The likelihood is represented by

\[
L(\lambda|y) = \prod_{i=1}^{n} \frac{e^{-\lambda_i} \lambda_i^{y_i}}{y_i!}.
\]

Mathematically it is easier to maximize the log-likelihood which is given by
\[ \ell(\lambda|y) = \sum_{i=1}^{n} (y_i \log \lambda_i - \lambda_i - \log(y_i!)). \]

Thus, we are looking for the value of \( \lambda \) that maximizes the log-likelihood above. Further, an offset is used if the response variable can be considered a rate. For example, MN frequency is scored from a larger number of total cells. So if the total number of cells examined varies by subject an offset is appropriate. In this case the expected value is

\[ E(y_i) = t_i \lambda_i \]

where \( t_i \) is the offset value. The conditional probability is then given by

\[ P(y_i|\lambda_i) = \frac{e^{-t_i \lambda_i} (t_i \lambda_i)^{y_i}}{y_i!} \]

for each observation \( i \). The likelihood is represented by

\[ L(\lambda|y) = \prod_{i=1}^{n} \frac{e^{-t_i \lambda_i} (t_i \lambda_i)^{y_i}}{y_i!}. \]

Again, mathematically it is easier to maximize the log-likelihood which is given by

\[ \ell(\lambda|y) = \sum_{i=1}^{n} (y_i \log(t_i \lambda_i) - t_i \lambda_i - \log(y_i!)). \]

Once again we are looking for the \( \lambda \) value that maximizes the log-likelihood. These log-likelihoods are used to model predictor variables. In Poisson regression the model assumes that the expected value can be modeled by a linear combination of predictors. In this case the natural log of \( t_i \) is entered as an offset in the model estimation. The natural log of the expected value is

\[ \log(E(y_i|x_i)) = \log(t_i) + \theta^T x_i \]

78
where $x_i$ is a vector of predictor variables and $\theta$ is a vector of coefficients. The estimated coefficients can be exponentiated to determine how the response changes with the predictor. By using the estimated linear combination of coefficient estimates and taking the exponent we can calculate the estimated response of that particular subject.

4.2.3 Generalized Monotone Incremental Forward Stagewise Poisson Model

The GMIFS method was previously described for the logistic regression scenario by Hastie et al. 2007 and here we adapted it to a Poisson regression model. For the proposed method we consider three types of parameters that $\theta$ from section 4.2.2 can be separated into along with an offset ($t_i$). The parameters are the intercept ($\alpha$), those corresponding to an unpenalized subset of predictors ($\gamma$), and those corresponding to a set of penalized predictors ($\beta$). The design matrix, $x$, consists of two parts, $x_j$ and $x_k$, where $j = 1, \ldots, J$ is the set of unpenalized predictors, $k = 1, \ldots, K$ is the set of penalized predictors and $J + K = P$ is the total number of predictors. The unpenalized predictors are those that we wish to force into the model, such as gender, age and smoking status which researchers consider important predictors of MN frequency [Ceppi et al., 2010] and their values are in the $x_{ij}$ design matrix for subject $i$. The penalized variables (thousands of features from a high-throughput genomic experiment) are those that the model will choose for us and are considered to be the investigative predictors and their values are in the $x_{ik}$ design matrix for subject $i$.

The algorithm proceeds in an iterative fashion and updates one of the penalized covariates by a small incremental amount at each step. To determine which penalized covariate is to be updated next the largest negative gradient is used. Thus we need to calculate the first derivative of the log-likelihood corresponding to each penalized predictor. The log-likelihood written in terms of $\alpha$, $\beta$ and $\gamma$ is
\[ \ell(\alpha, \beta, \gamma|y, x_j, x_k) = \sum_{i=1}^{n} (y_i(\alpha + \log(t_i) + \gamma x_{ij} + \beta x_{ik}) - \exp(\alpha + \log(t_i) + \gamma x_{ij} + \beta x_{ik}) - \log(y_i!)) \]

and the first derivative written in terms of \( \alpha, \beta \) and \( \gamma \) in matrix notation is

\[ \frac{\partial \ell}{\partial \beta} = X'(y - \exp(\alpha + \log(t_i) + \gamma x_j + \beta x_k)). \]

Once we know which covariate to update, we need to determine in what direction to update the covariate. In order to know the direction of the update the second derivative would need to be calculated which is a cumbersome process. [Hastie et al. 2007] showed that to avoid having to calculate the second derivative an expanded covariate space can be used. Using the notation mention previously where \( x_j \) are the unpenalized variables and \( x_k \) are the penalized subset, the expanded covariate space is \( \tilde{x} = [x_j : x_k : -x_k] \). For example, let \( \beta_1, \ldots, \beta_p \) be the positive coefficient estimates and \( \beta_{p+1}, \ldots, \beta_{2p} \) be the coefficient estimates of the negative version of \( x_k \). Then \( \beta \) is calculated by subtracting the pairs, \( \beta_1 - \beta_{p+1}, \ldots, \beta_p - \beta_{2p} \). Our proposed GMIFS algorithm using the expanded covariate set is

1. Initialize the components of \( \hat{\beta}^{(s)} = 0 \) at step \( s = 0 \).

2. Initialize the intercept \( \alpha \) and the unpenalized coefficients \( \gamma_j \) where \( j = 1, \ldots, J \) using a maximization algorithm of the log-likelihood.

3. Considering \( \alpha \) and \( \gamma \) fixed, find the predictor \( x_m \) where \( m = \arg\min_{2K} \left( -\frac{\partial \ell}{\partial \beta_k} \right) \) at the current estimate \( \hat{\beta} = \hat{\beta}^{(s)} \).

4. Update the corresponding coefficient \( \hat{\beta}_{m}^{(s+1)} = \hat{\beta}_{m}^{(s)} + \epsilon \) to yield a new vector of parameter estimates.
5. Update $\alpha$ and the unpenalized coefficients, $\gamma_j$, by maximum likelihood considering the $\hat{\beta}^{s+1}$ from step 4 as fixed.

6. Repeat steps 3-5 until the difference between successive log-likelihoods is less than a pre-specified tolerance, $\tau$. The defaults for the GMIFS algorithm are $\epsilon = 0.001$ and $\tau = 0.00001$.

4.2.4 Comparative Method: Penalized Linear Regression

A penalized linear regression model can be fit by adding a penalty term to the sums of squares. Specifically, the glmpath algorithm uses a linear combination of the $L_1$ and $L_2$ norm penalizations. The glmpath algorithm is based on a previous algorithm called LASSO. LASSO minimizes the typical sum of squares with an added constraint. Specifically for linear regression LASSO minimizes \cite{tibshirani2011}

$$
\sum_{i=1}^{N} (y_i - \sum_j x_{ij} \hat{\beta}_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|
$$

where $x_{ij}$ are the standardized predictors and $y_i$ is the set of centered responses for $i = 1, ..., N$ and $j = 1, ..., p$. Because of the form of the constraint, LASSO does both variable selection and shrinkage. The glmpath algorithm modifies this slightly by first considering the typical generalized linear model formula

$$
\hat{\beta} = \arg\max_{\beta} L(y; \beta)
$$

where $L$ denotes the appropriate likelihood function. The glmpath algorithm then adds an analagous LASSO penalty term to help with variable selection when $p > n$

$$
\hat{\beta}(\lambda) = \arg\min_{\beta} \{- \log L(y; \beta) + \lambda ||\beta||_1\}
$$

where $\lambda > 0$ is the regularization parameter. The glmpath algorithm computes coefficient
estimates as $\lambda$ varies. The algorithm starts with the largest $\lambda$ that makes $\hat{\beta}(\lambda)$ nonzero with each step using a smaller $\lambda$. Each optimization consists of three parts: determining the step size in $\lambda$, predicting the corresponding change in the coefficients, and correcting the error in the previous prediction \cite{Park2007}. The algorithm continues finding the next largest $\lambda$ that will change the coefficient estimates until no further predictors can be found. However, when the predictors are strongly correlated, the coefficient estimates become highly unstable using the $L_1$ norm penalization \cite{Hastie2007}. Thus, the glmpath algorithm adds a quadratic penalty term and computes the solution to

$$\hat{\beta}(\lambda_1) = \arg\min_\beta \left\{ -\log L(y; \beta) + \lambda_1 ||\beta||_1 + \frac{\lambda_2}{2} ||\beta||_2^2 \right\}$$

where $\lambda_1 \in (0, \infty)$ and $\lambda_2$ is a fixed, small, positive constant. By adding this quadratic penalty, the effects of the strong correlations do not affect the stability of the fit. Further, when the correlations are not strong, the effects of the quadratic penalty are negligible \cite{Hastie2007}. Thus, the glmpath algorithm uses both the $L_1$ and $L_2$ penalties as its default method.

The glmpath algorithm uses a default binomial distribution with a logit link and $\lambda_2 = .00001$. The algorithm also allows for a Poisson distribution with a log link and Gaussian distribution with an identity link. The algorithm then computes the regularization path for generalized linear models with L1 penalty.

4.2.5 Simulations

Simulations are a useful technique to test how well a new methodology performs. In this case, we wished to quantify how accurately the GMIFS method estimated true non-zero coefficients and predicted count data. Furthermore, we wished to determine how the GMIFS method compared to the glmpath method in predicting the count outcome and simulations provide a good platform to accomplish this comparison. Several general steps must be considered in the simulation process; how to simulate the response, how to simulate the
predictors associated with the response and how to simulate the predictors not associated with the response. Furthermore, we wished to examine how the methods perform under ideal situations and non-ideal situations such as when distributional assumptions are met and are not met respectively. Note that all simulations were performed using the R programming environment (version 3.1.1) \cite{R-Core-Team-2014}.

First, we considered the situation where the response is Poisson distributed and the user fits a Poisson regression model. Then we generated the response to follow a Poisson distribution where an offset (a) was and (b) was not used. The uniform distribution was used to generate the predictors. The steps involved in simulating the data under these conditions were as follows:

1. Randomly generate $P$ variables, $x_{i1}, x_{i2}, ..., x_{iP}$ where $i = 1$ to $n$, using the uniform distribution on the [0,1] interval.

2. Choose $P_1$ of the $P$ variables to be associated with the response.

3. Assign the $P_1 \beta$ values associated with the response and the intercept value, $\alpha$. If the offset is to vary, then a uniform distribution was used with maximum 2200 and minimum 1800 and rounded to the nearest integer. This range was selected because it is recommended to score MN using 2000 cells.

4. Generate the $\lambda$ values for the Poisson distribution using the following formula:

$$\lambda_i = \exp(\alpha + \log(\text{offset}_i) + \sum_{k=1}^{P_1} \beta_k x_{ik}).$$

5. Randomly generate $Y_i \sim \text{Poisson}(\lambda_i)$.

6. Fit a Poisson GAMIFS model and fit a \textsf{glimpath} model.

7. Repeat steps 1-6 $r$ times.
This simulation method was adjusted in several places. In this case we chose $n = 30$ and $n = 80$ which are roughly the sample sizes of our cord blood and breast cancer datasets, respectively. We studied the models letting $P = 100$ predictor variables, $P_1 = 5$ predictor variables associated with the response, and $r = 100$ simulations were used. The intercept ($\alpha$) and the five predictor variables associated with the response ($\beta_1, \beta_2, \beta_3, \beta_4, \beta_5$) were set to -5, 0.3, 0.2, -0.7, 0.5 and 0.1 respectively for data simulated using no offset. For data simulated using an offset, $\alpha$ was set to -7. This was done to keep lambda values low so the Gaussian approximation for the Poisson distribution is not appropriate. To compare the two different statistical models, the following three outcomes were examined:

A. The number of true predictors that have a non-zero coefficient;

B. The number of false predictors that have a non-zero coefficient;

C. Accuracy of count predictions from the model (sum of squared residuals) when applied to an independent test set.

The methods were compared with and without the use of an offset during the simulation process. Furthermore, the \texttt{glmpath} method allows for the use of Gaussian and Poisson distributions. Thus, those options were also used to see what effects user error had on the results. Thus, a total of three models were compared when the true distribution was Poisson:

1. Poisson GMIFS model;

2. \texttt{glmpath} using “poisson” family option and lambda2=0 which fits a LASSO model; and

3. \texttt{glmpath} using “gaussian” family option and lambda2=0 which fits a LASSO model.
4.3 Results

4.3.1 Simulations

Simulations were performed as described in section 4.2.5 and Figures 20-22 show the results of the simulations. Figure 20 shows the distribution of the number of predictors correctly identified as non-zero over 100 simulations and the types of models used. The data were generated using both \( n = 30 \) and \( n = 80 \) observations. The median number of correctly identified non-zero coefficients with no offset using GMIFS is 1 (range=0, 3) for \( n = 30 \) and 2 (range=0, 4) for \( n = 80 \). Similarly, the median number of correctly identified non-zero coefficients with no offset using \texttt{glmpath} with Poisson family is 1 (range=0, 5) for \( n = 30 \) and 2 (range=0, 4) for \( n = 80 \). This number increases slightly when using the \texttt{glmpath} with Gaussian family to a median of 2 (range=0, 5) for \( n = 30 \) and 4 (range=2, 5) for \( n = 80 \). All the numbers are similar when an offset is used to generate the data. The median correct using GMIFS is 0 (range=0, 3) for \( n = 30 \) and 1 (range=0, 4) for \( n = 80 \). The median correct using \texttt{glmpath} with Poisson family is 0 (range=0, 3) for \( n = 30 \) and 1 (range=0, 4) for \( n = 80 \). Once again the median values increase when using the \texttt{glmpath} with Gaussian family to 2 (range=0, 5) for \( n = 30 \) and 4 (range=2, 5) for \( n = 80 \).

Figure 21 shows the distribution of the number of predictors incorrectly identified as non-zero over 100 simulations and the types of models used. The data were generated using both \( n = 30 \) and \( n = 80 \) observations. The median number of incorrectly identified non-zero coefficients with no offset using GMIFS is 3 (range=0, 15) for \( n = 30 \) and 7 (range=0, 28) for \( n = 80 \). Similarly, the median number of incorrectly identified non-zero coefficients with no offset using \texttt{glmpath} with Poisson family is 3 (range=0, 17) for \( n = 30 \) and 7 (range=0, 41) for \( n = 80 \). This number increases when using the \texttt{glmpath} with Gaussian family to a median of 26 (range=23, 28) for \( n = 30 \) and 74 (range=73, 76) for \( n = 80 \). All results are similar when an offset is used to generate the data. The median incorrect using GMIFS is 2 (range=0, 14) for \( n = 30 \) and 5 (range=0, 26) for \( n = 80 \). The median incorrect using
The `glmpath` with Poisson family is 2 (range=0, 24) for \( n = 30 \) and 4.5 (range=0, 31) for \( n = 80 \). Once again the median values increase when using the `glmpath` with Gaussian family to 26 (range=23, 28) for \( n = 30 \) and 74 (range=72, 76) for \( n = 80 \).

Figure 22 shows the distribution of the sum of residuals squared as a measure of the model prediction accuracy. The data were generated using both \( n = 30 \) and \( n = 80 \) observations. For both sample sizes a learning data set was used to estimate coefficients and then the model was applied to an independent test data set. The median accuracy with no offset using GMIFS is 133 (range=68, 240) for \( n = 30 \) and 325 (range=188, 699) for \( n = 80 \). Similarly, the median accuracy with no offset using `glmpath` with Poisson family is 142 (range=55, 254) for \( n = 30 \) and 333 (range=185, 1666) for \( n = 80 \). The median accuracy with no offset using `glmpath` with Gaussian family is 206 (range=90, 383) for \( n = 30 \) and 1,503 (range=535, 3772) for \( n = 80 \). The numbers are different when an offset is used to generate the data. The median accuracy using GMIFS is 80 (range=30, 185) for \( n = 30 \) and 205 (range=137, 367) for \( n = 80 \). The median accuracy using `glmpath` with Poisson family is 80 (range=33, 805) for \( n = 30 \) and 206 (range=126, 339) for \( n = 80 \). The median accuracy with an offset using `glmpath` with Gaussian family for both sample sizes is above 50,000.
Figure 20: Number of Predictors Correctly Identified as Non-zero: This figure shows the distribution of the number of predictors correctly identified as non-zero over 100 simulations. There were 5 predictors had non-zero coefficients. Boxplots are separated by the type of distribution used to generate the data and number of observations.
Figure 21: Number of Predictors Incorrectly Identified as Non-zero: This figure shows the distribution of the number of predictors incorrectly identified as non-zero over 100 simulations. There were 95 predictors that had coefficients set to zero. Boxplots are separated by the type of distribution used to generate the data and number of observations.
Figure 22: Accuracy of Count Predictions: This figure shows the distribution of the sum of residuals squared over 100 simulations using a learning data set and an independent test data set. Boxplots are separated by the type of distribution used to generate the data and number of observations. The results for glmpath with Gaussian family using an offset are not displayed because both values are above 50,000.

4.3.2 Gene Expression Analysis

The cord blood data were collected as part of the Norwegian Mother and Child Cohort Study (Moba) [Magnus et al., 2006]. The target population of Moba is all women who give birth in Norway. The overall goal of this study was to collect data on pregnant women and their children to estimate the association between exposures and diseases. Specifically, the data are part of a subcohort called BraMat, which translates to “good food” in English. This subcohort concentrates on what effect a pregnant woman’s diet has on her child. Umbilical cord blood samples were collected immediately after birth from 200 babies. After quality
control and other exclusions, 111 samples were hybridized to Agilent 4x44k human oligonucleotide microarrays to measure gene expression. Of the 111 subjects, 29 also had MN data collected. The MN were scored using the procedure described by [Decordier et al. 2009]. Further, demographics such as gender were collected for all subjects. Data were downloaded from Gene Expression Omnibus (GSE31836). Sample processing, image analysis, normalization, background correction and filtering are described in [Hochstenbach et al. 2012]. For this analysis the data were further filtered to only include genes that had no missing values, leaving 8497 genes for statistical modeling.

Both GMIFS and glmpath models were applied to the cord blood gene expression data set described above. For glmpath, the Poisson family option was used and the lambda2 option was set to zero. For GMIFS, the default options were chosen. The response in the model was MN counts and the predictors were the gene expression intensities. Gender was included in the model as part of the unpenalized subset. Based on Figure 23, a Poisson distribution was assumed for both models since the data appear skewed. The final model parameters were chosen using the minimum AIC. The GMIFS model identified 17 non-zero gene expression coefficients as associated with MN count and glmpath with Poisson family identified 23. Out of the genes that were identified, 10 were common to both models. Figures 24 (Sum of squared residuals = 101.7) and 25 (Sum of squared residuals = 1.8) show that both models seem to predict MN relatively well. Because simulations showed that glmpath seems to overfit, glmpath’s ability to predict MN frequency was looked at using only 17 non-zero coefficients. Figure 26 (Sum of squared residuals = 80.0) shows that when the same number of coefficients are used both GMIFS and glmpath predict MN frequency with similar accuracy. Table 10 shows the genes that both models identified as being associated with MN count and the types of cancer with which they are linked. Nine out of the ten genes are linked to some type of cancer.
Table 10: Genes Identified as Associated with MN Count by both GMIFS and glmpath

<table>
<thead>
<tr>
<th>Probe ID</th>
<th>Gene Symbol</th>
<th>Gene Name</th>
<th>Associated with Cancer</th>
<th>GMIFS</th>
<th>glmpath</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-23-P100196</td>
<td>USP10</td>
<td>ubiquitin specific peptidase 10</td>
<td>Glioblastoma multiforme [Grunda et al., 2006]</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>A-23-P138967</td>
<td>SDHD</td>
<td>succinate dehydrogenase complex</td>
<td>Tumor Suppressor [King et al., 2006]</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>A-23-P42331</td>
<td>HMGA1</td>
<td>high mobility group AT-hook 1</td>
<td>Pancreatic Adenocarcinoma [Liu et al., 2008]</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>A-23-P9293</td>
<td>TIP2</td>
<td>tight junction protein 2</td>
<td>Breast [Martin et al., 2004]</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>A-24-P19410</td>
<td>CBX7</td>
<td>chromobox homolog 7</td>
<td>Carcinoma [Federico et al., 2009]</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>A-24-P214588</td>
<td>TREML2</td>
<td>triggering receptor expressed on myeloid cells-like 2</td>
<td>Pancreatic [Loos et al., 2009]</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>A-24-P2463</td>
<td>WHSC1</td>
<td>Wolf-Hirschhorn syndrome candidate 1</td>
<td>Carcinogenesis [Toyokawa et al., 2011]</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>A-24-P397584</td>
<td>TBCC</td>
<td>tubulin folding cofactor C</td>
<td>None Found</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>A-24-P398604</td>
<td>KIAA0258</td>
<td>None Found</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>A-32-P18547</td>
<td>C21orf57</td>
<td>chromosome 21 open reading frame 57</td>
<td>Breast [Smeets et al., 2011]</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>A-23-P103824</td>
<td>FAU</td>
<td>Finkel-Biskis-Reilly murine sarcoma virus (FBR-MuSV) ubiquitously expressed</td>
<td>None Found</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>A-23-P209394</td>
<td>CFLAR</td>
<td>CASP8 and FADD-like apoptosis regulator</td>
<td>Human cancers [Fulda, 2013]</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>A-23-P79911</td>
<td>PSMP1</td>
<td>proteasome (prosome, macropain) inhibitor subunit 1 (PSMI)</td>
<td>Breast [Kuznetsova et al., 2006]</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>A-24-P202567</td>
<td>ITPKC</td>
<td>inositol 1,4,5-trisphosphate 3-kinase C</td>
<td>Cervical [Yang et al., 2012]</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>A-24-P31235</td>
<td>EIF5A</td>
<td>eukaryotic translation initiation factor 5A</td>
<td>Chronic myeloid leukemia [Balabanov et al., 2007]</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>A-24-P405054</td>
<td>CLRORF144</td>
<td>chromosome 1 open reading frame 144</td>
<td>Mantle cell lymphoma [Schraders et al., 2008]</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>A-32-P156549</td>
<td>CLRORF144</td>
<td>None Found</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>A-25-P118313</td>
<td>GABARAPL2</td>
<td>GABA(A) receptor-associated protein-like 2</td>
<td>Lung [Borcziuk et al., 2003]</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>A-23-P143817</td>
<td>MYLK</td>
<td>myosin, light polypeptide kinase</td>
<td>Gastric [Chen et al., 2012]</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>A-23-P156809</td>
<td>LOC642880</td>
<td>similar to PKSG62</td>
<td>None Found</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>A-25-P394304</td>
<td>PDZK1P1</td>
<td>PDZK1 interacting protein 1</td>
<td>Thyroid [Di Marco et al., 2014]</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>A-23-P39665</td>
<td>SLC11A1</td>
<td>solute carrier family 11, member 1</td>
<td>Esophageal [Zaahl et al., 2005]</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>A-23-P67529</td>
<td>KCNN4</td>
<td>potassium intermediate/small conductance calcium-activated channel, subfamily N, member 4</td>
<td>Colorectal [Lai et al., 2011]</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>A-24-P594683</td>
<td>LOC645592</td>
<td>similar to peptidylprolyl isomerase A isoform 1</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>A-24-P708161</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>A-24-P98086</td>
<td>GNA12</td>
<td>guanine nucleotide binding protein</td>
<td>Oral [Gan et al., 2011]</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>A-32-P10067</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>A-32-P137849</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>A-32-P169754</td>
<td>LOC145221</td>
<td>EST</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>A-32-P208078</td>
<td>MTHFR</td>
<td>5,10-methylenetetrahydrofolate reductase (NADPH)</td>
<td>Breast [Chen et al., 2005]</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Figure 23: Histogram of MN Counts for cord blood data set.

Figure 24: Plot of actual MN counts versus predicted MN counts using GMIFS.
4.3.3 Methylation Analysis

The breast cancer data were collected as part of an ongoing prospective study titled, Epigenetics and psychoneurologic symptoms in women with breast cancer (R01NR012667) and
includes 70 women diagnosed with breast cancer having both MN frequencies and methylation profiles collected at baseline. MN frequency was typically scored from among 2000 cells with some variation in the total number of cells scored. Specifically, the cells were scored in four sets of 500 cells and mononucleated, binucleated and trinucleated cells were considered in the total. The subjects also all have methylation data available that can be paired with the MN data to test for associations. For each subject MN frequency was calculated as the number of binucleated cells containing one or more nuclei. Further, demographic variables such as age and smoking status were collected for all subjects.

Epigenetics is the “study of heritable changes in gene expression that occur independent of changes in the primary DNA sequence” [Sharma et al., 2010]. The epigenetic gene expression profile is affected by modifications such as DNA methylation, histone modifications, nucleosome positioning and microRNA. Cancer cells undergo drastic changes in the epigenomic landscape compared to normal cells [Sharma et al., 2010]. Alterations in epigenetic mechanisms can lead to silencing of tumor suppressor genes, activation of oncogenes and genetic instability. The first epigenetic alterations identified in cancer were DNA methylation changes. A typical cancer epigenome shows overall global hypomethylation and in contrast hypermethylation of CpG islands [Sharma et al., 2010].

There are several mechanisms that are used to upregulate or downregulate gene expression, one of which is DNA methylation. Essentially this process involves adding a methyl group to a CpG site. A CpG site is a cystine followed by a guanine connected by a phosphate backbone (hence CpG) on a single strand of DNA. There are two important features of CpG sites to consider. The first is rarity; a CpG combination occurs only about one fifth of the expected frequency. The second feature is methylation; CpG sites can be methylated and this accounts for most, if not all, of the methylcytosine in the vertebrate genome [Bird, 1985]. CpG sites can be part of CpG islands, a section of DNA that is rich in CpG sites and are generally present near gene promoters. The methylation status of adjacent CpG sites has been shown to be highly correlated [Bibikova et al., 2011]. This implies that CpG islands are
often methylated or unmethylated as a group. This large scale methylation of CpG islands near promoters has been shown to inhibit transcription of genes. CpG islands are found in approximately 60% of gene promoters [Dedeurwaerder et al., 2011].

Hypomethylation and hypermethylation of specific DNA regions work together to promote tumor growth. Hypomethylation of the genome at repetitive elements, retrotransposons, introns and gene deserts results in the increase of genomic instability. By opening these regions to modification, such as chromosomal rearrangement and transposon translocation, hypomethylation promotes genomic instability which is implicated in many cancers such as thymic lymphomas [Howard et al., 2007]. Similarly, hypomethylation of CpG poor promoters can result in activation of growth promoting genes, such as $R$-Ras and MAPSIN in gastric cancer [Sharma et al., 2010]. By activating these types of genes, hypomethylation promotes cancer development and progression. On the other hand, hypermethylation downregulates tumor suppressor genes such as $Rb$, a gene associated with retinoblastoma [Sharma et al., 2010]. Many other genes have been shown to be suppressed by hypermethylation that control functions such as DNA repair, cell cycle, cell adhesion, apoptosis and angiogenesis. All of these mechanisms have been shown to affect cancer development and progression. Thus, studying the epigenomic changes in cancer such as DNA methylation is of great interest. Having a better understanding of these changes can result in possible targets for cancer therapies.

The downregulation of genes with methylated CpG islands can be seen in studies performed on the inactive X chromosome. Females have two X chromosomes and only one needs to be transcriptionally active. Studies have found that many genes on the X chromosome, such as HPRT and G6PD, are made inactive by the methylation of CpG islands [Bird, 1985].

One way to study DNA methylation profiles is to use the Illumina HumanMethylation 450K assay. This assay uses bisulfite conversion followed by hybridization of the sample to an array that interrogates specific CpG sites. Bisulfite treatment of DNA converts unmethylated cytosines into uracils but leaves methylated cytosines unchanged. Since uracil binds to
adenine and cytosine binds to guanine, the bisulfite converted DNA can be differentiated between methylated and unmethylated loci. This can be done by using probes for specific CpG sites and using amplification with marked amino acids to measure whether an adenine or guanine was added. The Illumina 450K methylation assay consists of a chip that contains probes for 482,421 CpG sites [Bibikova et al., 2011]. It is worth mentioning that bisulfite treatment is a harsh process but has been shown to have good reproducibility and precision [Ogino et al., 2006]

The Illumina 450K chip covers 98.9% of UCSC RefGenes with an average of 17.2 probes per gene. Similarly, the probes target a high proportion of CpG islands. Using in vitro created standards, [Bibikova et al., 2011] were able to show that $\beta$ values (intensity of methylated allele/(intensity of methylated allele + intensity of unmethylated allele + 100) ) cluster around the expected value. [Bibikova et al., 2011] also tested reproducibility and correlation with whole-genome bisulfite sequencing (WGBS). The 450K assay showed high reproducibility between technical replicates with an $R^2$ of 0.992. Similarly, there was a high correlation between whole genome bisulfite sequencing (WGBS) and the 450K array. In normal lung cells the correlation was 0.95 and in tumor lung cells the correlation was 0.96 [Bibikova et al., 2011]. This shows that 450K methylation assay provides an accurate and cost-effective method to measure large scale DNA methylation levels.

The numerical summaries per beadtype are $\beta$ values which can be interpreted as the proportion methylation for that CpG site. Given that there are over 480,000 CpG sites interrogated, it is often necessary to filter the dataset prior to analysis. For example, CpG sites for which all samples have $\beta < 20\%$ are considered completely unmethylated and CpG sites for which all samples have $\beta > 80\%$ are considered completely methylated and both can be filtered from downstream analysis [Zou et al., 2014].

We intended to model MN frequency in the breast cancer data set, however when looking at Figure 27 the MN counts appear to be normally distributed. Thus, we modeled bud frequency, which appear Poisson distributed (Figure 28). Specifically, we were interested in
identifying a multivariable model where methylation levels served as predictor and bud frequency of binucleated cells was the outcome. Buds are similar to MN but do not completely separate from the main nucleus. Buds were calculated in the breast cancer data set as number of cells containing at least one bud. Age and smoking status were included in the model as part of the unpenalized subset. The default options were chosen for GMIFS. The total number of binucleated cells scored was included in the model as an offset. Prior to modeling, methylation sites that were all above 80% methylated and below 20% methylated were filtered out. Further, univariate Poisson models were run on each CpG site where bud frequency was the outcome to further retain only those that were significantly related to buds. CpG sites that had a p-value $\leq 0.05$ were included in the GMIFS model, leaving 10,860 CpG sites. The final model parameters were chosen using the minimum AIC. The final GMIFS model included 25 non-zero methylation loci coefficients. The model predicts that these 25 loci and subsequently the genes they are associated with Bud formation. Table [I] shows the 25 loci and the genes with which they are associated. Looking at Figure [29] the GMIFS model seems to predict bud frequency relatively well. When glmpath was applied to the data, a fatal error occured and the model would not run.

Figure 27: Histogram of MN Counts for breast cancer data set.
Figure 28: Histogram of bud frequencies in the breast cancer data set.

Figure 29: Plot of actual MN counts versus predicted MN counts using GMIFS for breast cancer data.
Table 11: Methylation Loci Identified as Associated with Bud Formation

<table>
<thead>
<tr>
<th>Methylation Loci</th>
<th>Associated Gene</th>
<th>Gene Name</th>
<th>Associated with Cancer</th>
</tr>
</thead>
<tbody>
<tr>
<td>cg21271015</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cg23791325</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cg18598929</td>
<td>HCG27</td>
<td>HLA Complex Group 27</td>
<td></td>
</tr>
<tr>
<td>cg06213635</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cg02402208</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cg06558765</td>
<td>ZFY</td>
<td>Zinc Finger Protein, Y-Linked</td>
<td></td>
</tr>
<tr>
<td>cg21605248</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cg12099459</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cg17594670</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cg13819787</td>
<td>SP9</td>
<td>Sp9 Transcription Factor</td>
<td></td>
</tr>
<tr>
<td>cg26443646</td>
<td>NCRNA00171</td>
<td>ZNRD1 Antisense RNA 1</td>
<td>Stomach [Hong et al., 2004]</td>
</tr>
<tr>
<td>cg10511429</td>
<td>CTHRC1</td>
<td>Collagen Triple Helix Repeat Containing 1</td>
<td>Solid Cancers [Tang et al., 2006]</td>
</tr>
<tr>
<td>cg26952662</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cg14140673</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cg15357934</td>
<td>TSPAN31</td>
<td>Tetraspanin 31</td>
<td>Colorectal [Kuhn et al., 2007]</td>
</tr>
<tr>
<td>cg15744240</td>
<td>FGF14</td>
<td>Fibroblast Growth Factor 14</td>
<td>Many [Turner and Grose, 2010]</td>
</tr>
<tr>
<td>cg12281446</td>
<td>SNAP23</td>
<td>Synaptosomal-Associated Protein</td>
<td></td>
</tr>
<tr>
<td>cg12788108</td>
<td>C17orf64</td>
<td>Chromosome 17 Open Reading Frame 64</td>
<td>Breast [Natrajan et al., 2009]</td>
</tr>
<tr>
<td>cg05190096</td>
<td>SHC2</td>
<td>SHC Transforming Protein 2</td>
<td>Breast [Dankort et al., 2001]</td>
</tr>
<tr>
<td>cg07054927</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cg21100328</td>
<td>DKKPz434J0226</td>
<td>Uncharacterized LOC93429</td>
<td></td>
</tr>
<tr>
<td>cg23343875</td>
<td>OR10H4</td>
<td>Olfactory Receptor, Family 10, Subfamily H, Member 4</td>
<td></td>
</tr>
<tr>
<td>cg07219314</td>
<td>ZDHHC8P</td>
<td>Zinc Finger, DHHC-Type Containing 8 Pseudogene 1</td>
<td></td>
</tr>
<tr>
<td>cg09692492</td>
<td>ZDHHC8P</td>
<td>Zinc Finger, DHHC-Type Containing 8 Pseudogene 1</td>
<td></td>
</tr>
</tbody>
</table>

4.4 Discussion

We described the generalized monotone incremental forward stagewise method for modeling a count response when we want to (1) coerce some variables into the model and (2) perform automatic variable selection and model estimation by penalizing predictors. High-throughput data contains more predictors than there are samples so traditional methods are not appropriate in this setting. The GMIFS method was compared to glmpath, a popular penalization algorithm. Simulations showed that both methods performed similarly when identifying predictors known to be non-zero. GMIFS appeared to slightly outperform glmpath in the sense that GMIFS included fewer predictors that are truly unimportant in the model. This implies glmpath tends to overfit. Similarly, when applied to an independent data set GMIFS appeared to have higher predictive accuracy. Also, glmpath seems to be unstable and crashed on all attempts to run on our breast cancer data set. Thus, it appears
that GMIFS is more generalizable than \texttt{glmpath} to independent data sets.

Both methods were applied to a cord blood gene expression data set. Gene expression profiles were used to predict MN frequency. Both models identified a similar number of genes as related to MN frequency. Further, ten of those genes were common to both models. Nine out of the ten genes have been shown to be associated with different types of cancers. Since MN count is a measure of DNA damage, genes associated with MN frequency would be expected to be linked to cancer. Similarly, both methods were applied breast cancer methylation data set. Both models identified a similar number of methylation sites and 51 of those sites were common to both models.

Both models appear to identify genes linked to cancer. As in the simulations, \texttt{glmpath} identified more genes as non-zero compared to GMIFS. In the simulations, this was because \texttt{glmpath} was including more predictors incorrectly. However, there is no way to know if this is also the case in the cord blood data set given that these data are observational and no further confirmatory studies can be performed on the samples.

When the models were applied to a breast cancer data set, GMIFS found 25 methylation loci associated with bud formation. Because bud formation is a measure of DNA damage it is a reasonable assumption that these loci are related to genes that are involved in DNA stability. \texttt{glmpath}, on the other hand, crashed when the model was run.
5 Conclusions and Future Work

5.1 TCR Analysis Conclusions and Future Work

The first part of this thesis proposed several methods to analyze TCR repertoire data. Specifically, methods to differentiate between patients with and without GVHD after HSCT. The study data consisted of 13 patients that had undergone HSCT. Five patients suffered for GVHD and eight were GVHD-free. Six methods were described and applied that attempted to differentiate the two groups at the VDJ combination level. Three of these methods (CDR3 Distribution Perturbation, Simpson’s Diversity Index and Non-parametric Method) calculated different measures for each VDJ combination of the recipients and then permutation tests were used to test for differences. The CDR3 perturbation method calculated using a mean reference and individual reference found three and two significant differences after adjusting for multiple comparisons, respectively. Simpson’s diversity index found five significant differences after adjusting for multiple comparisons. The non-parametric method found one significant difference after adjusting for multiple comparisons. The Kullback-Leibler Divergence method was used to compare the difference between CDR3 length distribution at each VDJ combination. This method found almost all VDJ combinations to be statistically different between groups. However, looking at histograms it is difficult to tell what is causing those differences. The proportion logit method used both donor and recipient data to calculate the proportion of shared sequences between donor and recipient at each VDJ combination. Student’s T-tests were then used to test for differences. Once again no differences were found between the groups after adjusting for multiple comparison. There were two VDJ combinations that were significant before multiple comparison adjustment. Finally, the Oligoscore method calculated a score for each CDR3 length within a VDJ combination for each group. This method identified several peaks that seem to differ between groups when looked at graphically. These methods were all applied to data collected shortly after the HSCT procedure when the immune systems is not very well reconstituted. Future
work should be done to test for differences in later time points after treatment and prior to treatment. The measures can also be modified to analyze nucleotide or amino acid sequences rather than CDR3 lengths. The HSCT data were also treated as compositional data. No significant differences were found after adjusting for multiple comparisons. All these methods measure different aspects of the T-cell repertoire. Researchers should choose the best measure for the research question they wish to answer. Further, these methodologies should be tried on larger sample sizes to see how they perform in more ideal circumstances.

5.2 Generalized Monotone Incremental Forward Stagewise Method

Conclusions and Future Work

The GMIFS method was extended to analyze count data using the Poisson distribution. Chapter 4 described our GMIFS method and how it performed compared to a popular alternative, glmpath. Specifically, analyses related to gene expression and methylation were considered. In both cases, it is important to take into account the cell composition of the sample being tested [Jaffe and Irizarry, 2014]. This is because different cell types have different gene expression and methylation profiles. [Houseman et al., 2012] describe a statistical method using methylation data to estimate cell composition. Accounting for varying cell compositions is important because differences between groups could be due to the difference in sample cell composition rather than true biological differences. Not all count data follows a Poisson distribution and GMIFS can be extended to the negative binomial distribution as well. Finally, both the Poisson and negative binomial GMIFS should be extended to the longitudinal setting as well.
References


Yoav Benjamini and Yosef Hochberg. On the adaptive control of the false discovery rate


Jia Chen, Marilie D Gammon, Wendy Chan, Caroline Palomeque, James G Wetmur, Geoffrey C Kabat, Susan L Teitelbaum, Julie A Britton, Mary Beth Terry, Alfred I Neugut,


Ilse Decordier, Alexander Papine, Gina Plas, Sam Roesems, Kim Vande Loock, Jennifer Moreno-Palom, Eduardo Cemeli, Diana Anderson, Aleksandra Fucic, Ricardo Marcos,


M Fenech, Wushou P Chang, Micheline Kirsch-Volders, Nina Holland, Stefano Bonassi, and Errol Zeiger. HUMN project: detailed description of the scoring criteria for the


EB Kuznetsova, TV Kekeeva, SS Larin, VV Zemliakova, OV Babenko, MV Nemtsova,


Shuji Ogino, Takako Kawasaki, Mohan Brahmandam, Mami Cantor, Gregory J Kirkner, Donna Spiegelman, G Mike Makrigiorgos, Daniel J Weisenberger, Peter W Laird, Massimo Loda, et al. Precision and performance characteristics of bisulfite conversion and real-time


Harlan S. Robins, Paulo V. Campregher, Santosh K. Srivastava, Abigail Wacher, Cameron J. Turtle, Orsalem Kahsai, Stanley R. Riddell, Edus H. Warren, and Christopher S. Carl-


### 6 R Code

#### 6.1 T-cell Receptor Code

```r
# Change file directory to where TSV files are located
setwd("C:/Users/makowski/Desktop/Dissertation/TCRdata")

# Function to pull out on sequences marked as productive
# and put recipient and donor in one list
fileread <- function(name1, name2, file1, file2) {
  name1 <- subset(read.delim(files[file1]), sequenceStatus == "Productive")
  name2 <- subset(read.delim(files[file2]), sequenceStatus == "Productive")
  list(name1, name2)
}

# Run the function for each patient
pt11 <- fileread(name1 = pt11d, name2 = pt11r3, file1 = 1, file2 = 2)
pt12 <- fileread(name1 = pt12d, name2 = pt12r2, file1 = 3, file2 = 4)
pt13 <- fileread(name1 = pt13d, name2 = pt13r2, file1 = 5, file2 = 6)
pt14 <- fileread(name1 = pt14d, name2 = pt14r3, file1 = 7, file2 = 8)
pt16 <- fileread(name1 = pt16d, name2 = pt16r2, file1 = 9, file2 = 10)
pt24 <- fileread(name1 = pt24d, name2 = pt24r3, file1 = 11, file2 = 12)
pt3 <- fileread(name1 = pt3d, name2 = pt3r2, file1 = 13, file2 = 14)
pt4 <- fileread(name1 = pt4d, name2 = pt4r2, file1 = 15, file2 = 16)
```
25 pt5<-fileread(name1=pt5d, name2=pt5r2, file1=17, file2=18)
26 pt6<-fileread(name1=pt6d, name2=pt6r3, file1=19, file2=20)
27 pt7<-fileread(name1=pt7d, name2=pt7r3, file1=21, file2=22)
28 pt8<-fileread(name1=pt8d, name2=pt8r3, file1=23, file2=24)
29 pt9<-fileread(name1=pt9d, name2=pt9r3, file1=25, file2=26)
30
31 #function to calculate rpm for both patient and donor
32 rpm<-function(patient){
33 norm<-patient[[1]][,6]
34 depth<-sum(norm)
35 RPM<-(norm/depth)*1e6
36 donor<-cbind(patient[[1]],RPM)
37
38 norm<-patient[[2]][,6]
39 depth<-sum(norm)
40 RPM<-(norm/depth)*1e6
41 recipient<-cbind(patient[[2]],RPM)
42
43 list(donor,recipient)
44 }
45
46 #use function to calculate rpm for donor and recipient
47 pt11rpm<-rpm(pt11)
48 rm(pt11)
49 pt12rpm<-rpm(pt12)
50 rm(pt12)
51 pt13rpm<-rpm(pt13)
52 rm(pt13)
53 pt14rpm<-rpm(pt14)
54 rm(pt14)
55 pt16rpm<-rpm(pt16)
56 rm(pt16)
57 pt24rpm<-rpm(pt24)
58 rm(pt24)
59 pt3rpm<-rpm(pt3)
60 rm(pt3)
61 pt4rpm<-rpm(pt4)
62 rm(pt4)
63 pt5rpm<-rpm(pt5)
64 rm(pt5)
65 pt6rpm<-rpm(pt6)
66 rm(pt6)
67 pt7rpm<-rpm(pt7)
68 rm(pt7)
69 pt8rpm<-rpm(pt8)
70 rm(pt8)
71 pt9rpm<-rpm(pt9)
72 rm(pt9)

# possible V,D and J genes
74 # Note not all patients have all VDJ combinations interrogated
75 # by Immunoseq technology so this finds all of them
76 # I use both Dg and d later and easier to make both than change more code
J<-unique(c(levels(pt11rpm[[1]]$JGeneName), levels(pt12rpm[[1]]$JGeneName), levels(pt13rpm[[1]]$JGeneName), levels(pt14rpm[[1]]$JGeneName), levels(pt16rpm[[1]]$JGeneName), levels(pt24rpm[[1]]$JGeneName), levels(pt3rpm[[1]]$JGeneName), levels(pt4rpm[[1]]$JGeneName), levels(pt5rpm[[1]]$JGeneName), levels(pt6rpm[[1]]$JGeneName), levels(pt7rpm[[1]]$JGeneName), levels(pt8rpm[[1]]$JGeneName), levels(pt9rpm[[1]]$JGeneName)))

V<-unique(c(levels(pt11rpm[[1]]$VGeneName), levels(pt12rpm[[1]]$VGeneName), levels(pt13rpm[[1]]$VGeneName), levels(pt14rpm[[1]]$VGeneName), levels(pt16rpm[[1]]$VGeneName), levels(pt24rpm[[1]]$VGeneName), levels(pt3rpm[[1]]$VGeneName), levels(pt4rpm[[1]]$VGeneName), levels(pt5rpm[[1]]$VGeneName), levels(pt6rpm[[1]]$VGeneName), levels(pt7rpm[[1]]$VGeneName), levels(pt8rpm[[1]]$VGeneName), levels(pt9rpm[[1]]$VGeneName)))

Dg<-unique(c(levels(pt11rpm[[1]]$DGeneName), levels(pt12rpm[[1]]$DGeneName), levels(pt13rpm[[1]]$DGeneName), levels(pt14rpm[[1]]$DGeneName), levels(pt16rpm[[1]]$DGeneName), levels(pt24rpm[[1]]$DGeneName), levels(pt3rpm[[1]]$DGeneName), levels(pt4rpm[[1]]$DGeneName), levels(pt5rpm[[1]]$DGeneName), levels(pt6rpm[[1]]$DGeneName), levels(pt7rpm[[1]]$DGeneName), levels(pt8rpm[[1]]$DGeneName), levels(pt9rpm[[1]]$DGeneName)))

#remove undefined values

V<-V[-1]
d<-Dg[-1]

#save file for later use

setwd("C:/Users/makowski/Desktop/Archer Code and Files")
save.image("TcellRPM.RData")
Determine VDJ combinations present in each sample

setwd("C:/Users/makowski/Desktop/Archer Code and Files")
load("TcellRPM.RData")

# Sum all VDJ combinations
location<-expand.grid(J,V,d)
VDJcomp<-matrix(ncol=1248,nrow=13)
for (i in 1:1248){
y<-2
x<-which(J==location[i,1])
w<-which(V==location[i,2])
z<-which(d==location[i,3])
VDJcomp[,i]<-c(sum(pt11rpm[[y]]$RPM[which(pt11rpm[[y]]$JGeneName==J[x] & pt11rpm[[y]]$VGeneName==V[w] & pt11rpm[[y]]$DGeneName==d[z])]),
       sum(pt12rpm[[y]]$RPM[which(pt12rpm[[y]]$JGeneName==J[x] & pt12rpm[[y]]$VGeneName==V[w] & pt12rpm[[y]]$DGeneName==d[z])]),
       sum(pt13rpm[[y]]$RPM[which(pt13rpm[[y]]$JGeneName==J[x] & pt13rpm[[y]]$VGeneName==V[w] & pt13rpm[[y]]$DGeneName==d[z])]),
       sum(pt14rpm[[y]]$RPM[which(pt14rpm[[y]]$JGeneName==J[x] & pt14rpm[[y]]$VGeneName==V[w] & pt14rpm[[y]]$DGeneName==d[z])])}
sum(pt16rpm[[y]]$RPM[which(pt16rpm[[y]]$JGeneName==J[x] & pt16rpm[[y]]$VGeneName==V[w] & pt16rpm[[y]]$DGeneName==d[z])]),

sum(pt24rpm[[y]]$RPM[which(pt24rpm[[y]]$JGeneName==J[x] & pt24rpm[[y]]$VGeneName==V[w] & pt24rpm[[y]]$DGeneName==d[z])]),

sum(pt3rpm[[y]]$RPM[which(pt3rpm[[y]]$JGeneName==J[x] & pt3rpm[[y]]$VGeneName==V[w] & pt3rpm[[y]]$DGeneName==d[z])]),

sum(pt4rpm[[y]]$RPM[which(pt4rpm[[y]]$JGeneName==J[x] & pt4rpm[[y]]$VGeneName==V[w] & pt4rpm[[y]]$DGeneName==d[z])]),

sum(pt5rpm[[y]]$RPM[which(pt5rpm[[y]]$JGeneName==J[x] & pt5rpm[[y]]$VGeneName==V[w] & pt5rpm[[y]]$DGeneName==d[z])]),

sum(pt6rpm[[y]]$RPM[which(pt6rpm[[y]]$JGeneName==J[x] & pt6rpm[[y]]$VGeneName==V[w] & pt6rpm[[y]]$DGeneName==d[z])]),

sum(pt7rpm[[y]]$RPM[which(pt7rpm[[y]]$JGeneName==J[x] & pt7rpm[[y]]$VGeneName==V[w] & pt7rpm[[y]]$DGeneName==d[z])]),

sum(pt8rpm[[y]]$RPM[which(pt8rpm[[y]]$JGeneName==J[x] & pt8rpm[[y]]$VGeneName==V[w] & pt8rpm[[y]]$DGeneName==d[z])]),

sum(pt9rpm[[y]]$RPM[which(pt9rpm[[y]]$JGeneName==J[x] & pt9rpm[[y]]$VGeneName==V[w] & pt9rpm[[y]]$DGeneName==d[z])])

}

# check to see where 0's are

Diversity.check<-matrix(rep(0,(1248*13)),ncol=1248,nrow=13)

for (i in 1:13){
  for (j in 1:1248){

if (VDJcomp[i,j] == "0") {
    Diversity.check[i,j] <- 1
} else {
    Diversity.check[i,j] <- 0
}

# keep only VDJ combinations that 7 or more of the sample contain
NAs <- c(which(apply(Diversity.check, 2, sum) == 13),
         which(apply(Diversity.check, 2, sum) == 12),
         which(apply(Diversity.check, 2, sum) == 11),
         which(apply(Diversity.check, 2, sum) == 10),
         which(apply(Diversity.check, 2, sum) == 9),
         which(apply(Diversity.check, 2, sum) == 8),
         which(apply(Diversity.check, 2, sum) == 7),
         which(apply(Diversity.check, 2, sum) == 6))

save(NAs, file = "VDJNAs.RData")

# CDR3 Perturbation Test using mean
# loaded the needed file
setwd("C:/Users/makowski/Desktop/Archer Code and Files")
load("TcellRPM.RData")

# Function that does the perturbation test for the VDJ families; x is the J gene,
# w is V gene and z is d gene and y=1 is donor and y=2 is recipient
Perttest<-function(x,w,z,y){
    #this part calculates the number of unique CDR3 lengths
    L=length(unique(c(unique(pt11rpm[[y]]$cdr3Length[which(pt11rpm[[y]]$JGeneName==J[x] & pt11rpm[[y]]$VGeneName==V[w] & pt11rpm[[y]]$DGeneName==d[z]])),
              unique(pt12rpm[[y]]$cdr3Length[which(pt12rpm[[y]]$JGeneName==J[x] & pt12rpm[[y]]$VGeneName==V[w] & pt12rpm[[y]]$DGeneName==d[z]])),
              unique(pt13rpm[[y]]$cdr3Length[which(pt13rpm[[y]]$JGeneName==J[x] & pt13rpm[[y]]$VGeneName==V[w] & pt13rpm[[y]]$DGeneName==d[z]])),
              unique(pt14rpm[[y]]$cdr3Length[which(pt14rpm[[y]]$JGeneName==J[x] & pt14rpm[[y]]$VGeneName==V[w] & pt14rpm[[y]]$DGeneName==d[z]])),
              unique(pt16rpm[[y]]$cdr3Length[which(pt16rpm[[y]]$JGeneName==J[x] & pt16rpm[[y]]$VGeneName==V[w] & pt16rpm[[y]]$DGeneName==d[z]])),
              unique(pt24rpm[[y]]$cdr3Length[which(pt24rpm[[y]]$JGeneName==J[x] & pt24rpm[[y]]$VGeneName==V[w] & pt24rpm[[y]]$DGeneName==d[z]])),
              unique(pt3rpm[[y]]$cdr3Length[which(pt3rpm[[y]]$JGeneName==J[x] & pt3rpm[[y]]$VGeneName==V[w] & pt3rpm[[y]]$DGeneName==d[z]])),
              unique(pt4rpm[[y]]$cdr3Length[which(pt4rpm[[y]]$JGeneName==J[x] & pt4rpm[[y]]$VGeneName==V[w] & pt4rpm[[y]]$DGeneName==d[z]])),
              unique(pt5rpm[[y]]$cdr3Length[which(pt5rpm[[y]]$JGeneName==J[x] & pt5rpm[[y]]$VGeneName==V[w] & pt5rpm[[y]]$DGeneName==d[z]])),
              unique(pt6rpm[[y]]$cdr3Length[which(pt6rpm[[y]]$JGeneName==J[x] & pt6rpm[[y]]$VGeneName==V[w] & pt6rpm[[y]]$DGeneName==d[z]])),
              unique(pt7rpm[[y]]$cdr3Length[which(pt7rpm[[y]]$JGeneName==J[x] & pt7rpm[[y]]$VGeneName==V[w] & pt7rpm[[y]]$DGeneName==d[z]])),
              unique(pt8rpm[[y]]$cdr3Length[which(pt8rpm[[y]]$JGeneName==J[x] & pt8rpm[[y]]$VGeneName==V[w] & pt8rpm[[y]]$DGeneName==d[z]])),
              unique(pt9rpm[[y]]$cdr3Length[which(pt9rpm[[y]]$JGeneName==J[x] & pt9rpm[[y]]$VGeneName==V[w] & pt9rpm[[y]]$DGeneName==d[z]])),
              unique(pt10rpm[[y]]$cdr3Length[which(pt10rpm[[y]]$JGeneName==J[x] & pt10rpm[[y]]$VGeneName==V[w] & pt10rpm[[y]]$DGeneName==d[z]])),
              unique(pt11rpm[[y]]$cdr3Length[which(pt11rpm[[y]]$JGeneName==J[x] & pt11rpm[[y]]$VGeneName==V[w] & pt11rpm[[y]]$DGeneName==d[z]])),
              unique(pt12rpm[[y]]$cdr3Length[which(pt12rpm[[y]]$JGeneName==J[x] & pt12rpm[[y]]$VGeneName==V[w] & pt12rpm[[y]]$DGeneName==d[z]])),
              unique(pt13rpm[[y]]$cdr3Length[which(pt13rpm[[y]]$JGeneName==J[x] & pt13rpm[[y]]$VGeneName==V[w] & pt13rpm[[y]]$DGeneName==d[z]])),
              unique(pt14rpm[[y]]$cdr3Length[which(pt14rpm[[y]]$JGeneName==J[x] & pt14rpm[[y]]$VGeneName==V[w] & pt14rpm[[y]]$DGeneName==d[z]])),
              unique(pt16rpm[[y]]$cdr3Length[which(pt16rpm[[y]]$JGeneName==J[x] & pt16rpm[[y]]$VGeneName==V[w] & pt16rpm[[y]]$DGeneName==d[z]])),
              unique(pt24rpm[[y]]$cdr3Length[which(pt24rpm[[y]]$JGeneName==J[x] & pt24rpm[[y]]$VGeneName==V[w] & pt24rpm[[y]]$DGeneName==d[z]])),
              unique(pt3rpm[[y]]$cdr3Length[which(pt3rpm[[y]]$JGeneName==J[x] & pt3rpm[[y]]$VGeneName==V[w] & pt3rpm[[y]]$DGeneName==d[z]])),
              unique(pt4rpm[[y]]$cdr3Length[which(pt4rpm[[y]]$JGeneName==J[x] & pt4rpm[[y]]$VGeneName==V[w] & pt4rpm[[y]]$DGeneName==d[z]])),
    123
unique(pt9rpm[[y]]$cdr3Length[which(pt9rpm[[y]]$JGeneName==J[x] & pt9rpm[[y]]$VGeneName==V[w] & pt9rpm[[y]]$DGeneName==d[z])])

# print(L)

# this sets the perturbation statistic to zero if the VDJ combination is not present
if (L==0){
  AbsoluteDeviancei<-rep(0,13)
  AbsoluteDeviancei
}

else {
  # This list the peak lengths
  Peaks=unique(c(unique(pt11rpm[[y]]$cdr3Length[which(pt11rpm[[y]]$JGeneName==J[x] & pt11rpm[[y]]$VGeneName==V[w] & pt11rpm[[y]]$DGeneName==d[z]))

  ,
  unique(pt12rpm[[y]]$cdr3Length[which(pt12rpm[[y]]$JGeneName==J[x] & pt12rpm[[y]]$VGeneName==V[w] & pt12rpm[[y]]$DGeneName==d[z])]),
  unique(pt13rpm[[y]]$cdr3Length[which(pt13rpm[[y]]$JGeneName==J[x] & pt13rpm[[y]]$VGeneName==V[w] & pt13rpm[[y]]$DGeneName==d[z])]),
  unique(pt14rpm[[y]]$cdr3Length[which(pt14rpm[[y]]$JGeneName==J[x] & pt14rpm[[y]]$VGeneName==V[w] & pt14rpm[[y]]$DGeneName==d[z])]),
  unique(pt16rpm[[y]]$cdr3Length[which(pt16rpm[[y]]$JGeneName==J[x] & pt16rpm[[y]]$VGeneName==V[w] & pt16rpm[[y]]$DGeneName==d[z])]),
  unique(pt24rpm[[y]]$cdr3Length[which(pt24rpm[[y]]$JGeneName==J[x] & pt24rpm[[y]]$VGeneName==V[w] & pt24rpm[[y]]$DGeneName==d[z])])
}
unique(pt3rpm[[y]]$cdr3Length[which(pt3rpm[[y]]$JGeneName==J[x] &
    pt3rpm[[y]]$VGeneName==V[w] & pt3rpm[[y]]$DGeneName==d[z])]),

unique(pt4rpm[[y]]$cdr3Length[which(pt4rpm[[y]]$JGeneName==J[x] &
    pt4rpm[[y]]$VGeneName==V[w] & pt4rpm[[y]]$DGeneName==d[z])]),

unique(pt5rpm[[y]]$cdr3Length[which(pt5rpm[[y]]$JGeneName==J[x] &
    pt5rpm[[y]]$VGeneName==V[w] & pt5rpm[[y]]$DGeneName==d[z])]),

unique(pt6rpm[[y]]$cdr3Length[which(pt6rpm[[y]]$JGeneName==J[x] &
    pt6rpm[[y]]$VGeneName==V[w] & pt6rpm[[y]]$DGeneName==d[z])]),

unique(pt7rpm[[y]]$cdr3Length[which(pt7rpm[[y]]$JGeneName==J[x] &
    pt7rpm[[y]]$VGeneName==V[w] & pt7rpm[[y]]$DGeneName==d[z])]),

unique(pt8rpm[[y]]$cdr3Length[which(pt8rpm[[y]]$JGeneName==J[x] &
    pt8rpm[[y]]$VGeneName==V[w] & pt8rpm[[y]]$DGeneName==d[z])]),

unique(pt9rpm[[y]]$cdr3Length[which(pt9rpm[[y]]$JGeneName==J[x] &
    pt9rpm[[y]]$VGeneName==V[w] & pt9rpm[[y]]$DGeneName==d[z])])

#this calculates the total rpm in each peak

Peaksi<-matrix(ncol=13,nrow=L)

for (i in 1:L){

    Peaksi[i,]<-c(sum(pt11rpm[[y]]$RPM[which(pt11rpm[[y]]$JGeneName==J[x] &
        pt11rpm[[y]]$VGeneName==V[w] & pt11rpm[[y]]$DGeneName==d[z] &
        pt11rpm[[y]]$cdr3Length==Peaks[i])]),

    sum(pt12rpm[[y]]$RPM[which(pt12rpm[[y]]$JGeneName==J[x] &
        pt12rpm[[y]]$VGeneName==V[w] & pt12rpm[[y]]$DGeneName==d[z] &
        pt12rpm[[y]]$cdr3Length==Peaks[i])]),

    sum(pt13rpm[[y]]$RPM[which(pt13rpm[[y]]$JGeneName==J[x] &
        pt13rpm[[y]]$VGeneName==V[w] & pt13rpm[[y]]$DGeneName==d[z] &
        pt13rpm[[y]]$cdr3Length==Peaks[i])])}
sum(pt14rpm[[y]]$RPM[which(pt14rpm[[y]]$JGeneName==J[x] & pt14rpm[[y]]$VGeneName==V[w] & pt14rpm[[y]]$DGeneName==d[z] & pt14rpm[[y]]$cdr3Length==Peaks[i]])],
sum(pt16rpm[[y]]$RPM[which(pt16rpm[[y]]$JGeneName==J[x] & pt16rpm[[y]]$VGeneName==V[w] & pt16rpm[[y]]$DGeneName==d[z] & pt16rpm[[y]]$cdr3Length==Peaks[i]])],
sum(pt24rpm[[y]]$RPM[which(pt24rpm[[y]]$JGeneName==J[x] & pt24rpm[[y]]$VGeneName==V[w] & pt24rpm[[y]]$DGeneName==d[z] & pt24rpm[[y]]$cdr3Length==Peaks[i]])],
sum(pt3rpm[[y]]$RPM[which(pt3rpm[[y]]$JGeneName==J[x] & pt3rpm[[y]]$VGeneName==V[w] & pt3rpm[[y]]$DGeneName==d[z] & pt3rpm[[y]]$cdr3Length==Peaks[i]])],
sum(pt4rpm[[y]]$RPM[which(pt4rpm[[y]]$JGeneName==J[x] & pt4rpm[[y]]$VGeneName==V[w] & pt4rpm[[y]]$DGeneName==d[z] & pt4rpm[[y]]$cdr3Length==Peaks[i]])],
sum(pt5rpm[[y]]$RPM[which(pt5rpm[[y]]$JGeneName==J[x] & pt5rpm[[y]]$VGeneName==V[w] & pt5rpm[[y]]$DGeneName==d[z] & pt5rpm[[y]]$cdr3Length==Peaks[i]])],
sum(pt6rpm[[y]]$RPM[which(pt6rpm[[y]]$JGeneName==J[x] & pt6rpm[[y]]$VGeneName==V[w] & pt6rpm[[y]]$DGeneName==d[z] & pt6rpm[[y]]$cdr3Length==Peaks[i]])],
sum(pt7rpm[[y]]$RPM[which(pt7rpm[[y]]$JGeneName==J[x] & pt7rpm[[y]]$VGeneName==V[w] & pt7rpm[[y]]$DGeneName==d[z] & pt7rpm[[y]]$cdr3Length==Peaks[i]])],
sum(pt8rpm[[y]]$RPM[which(pt8rpm[[y]]$JGeneName==J[x] & pt8rpm[[y]]$VGeneName==V[w] & pt8rpm[[y]]$DGeneName==d[z] & pt8rpm[[y]]$cdr3Length==Peaks[i]])],
126
sum(pt9rpm[[y]]$RPM[which(pt9rpm[[y]]$JGeneName==J[x] & pt9rpm
[[y]]$VGeneName==V[w] & pt9rpm[[y]]$DGeneName==d[z] & pt9rpm
[[y]]$cdr3Length==Peaks[i])])
}
#total rpm per patient
Peakssum<-apply(Peaksi,2,sum)

#calculates the probability of each peak
Probabilityi<-matrix(ncol=13,nrow=L)
for (j in 1:L) {
  Probabilityi[j,]<-Peaksi[j,]/Peakssum
}

#calculates total rpm in each donor peak
controlPeaksi<-matrix(ncol=13,nrow=L)
for (k in 1:L){
  controlPeaksi[k,]<-c(sum(pt11rpm[[1]]$RPM[which(pt11rpm[[1]]$JGeneName==J[x]
 & pt11rpm[[1]]$VGeneName==V[w] & pt11rpm[[1]]$DGeneName==d[z] & pt11rpm
[[1]]$cdr3Length==Peaks[k])]),
  sum(pt12rpm[[1]]$RPM[which(pt12rpm[[1]]$JGeneName==J[x] & pt12rpm[[1]]$VGeneName==V[w] & pt12rpm[[1]]$DGeneName==d[z]
 & pt12rpm[[1]]$cdr3Length==Peaks[k])]),
  sum(pt13rpm[[1]]$RPM[which(pt13rpm[[1]]$JGeneName==J[x] & pt13rpm[[1]]$VGeneName==V[w] & pt13rpm[[1]]$DGeneName==d[z]
 & pt13rpm[[1]]$cdr3Length==Peaks[k])])}
& pt13rpm[[1]]$cdr3Length==Peaks[k])]),
sum(pt14rpm[[1]]$RPM[which(pt14rpm[[1]]$JGeneName==J[x] &
    pt14rpm[[1]]$VGeneName==V[w] & pt14rpm[[1]]$DGeneName==d[z] &
    pt14rpm[[1]]$cdr3Length==Peaks[k])]),
sum(pt16rpm[[1]]$RPM[which(pt16rpm[[1]]$JGeneName==J[x] &
    pt16rpm[[1]]$VGeneName==V[w] & pt16rpm[[1]]$DGeneName==d[z] &
    pt16rpm[[1]]$cdr3Length==Peaks[k])]),
sum(pt24rpm[[1]]$RPM[which(pt24rpm[[1]]$JGeneName==J[x] &
    pt24rpm[[1]]$VGeneName==V[w] & pt24rpm[[1]]$DGeneName==d[z] &
    pt24rpm[[1]]$cdr3Length==Peaks[k])]),
sum(pt3rpm[[1]]$RPM[which(pt3rpm[[1]]$JGeneName==J[x] & pt3rpm
    [[1]]$VGeneName==V[w] & pt3rpm[[1]]$DGeneName==d[z] & pt3rpm
    [[1]]$cdr3Length==Peaks[k])]),
sum(pt4rpm[[1]]$RPM[which(pt4rpm[[1]]$JGeneName==J[x] & pt4rpm
    [[1]]$VGeneName==V[w] & pt4rpm[[1]]$DGeneName==d[z] & pt4rpm
    [[1]]$cdr3Length==Peaks[k])]),
sum(pt5rpm[[1]]$RPM[which(pt5rpm[[1]]$JGeneName==J[x] & pt5rpm
    [[1]]$VGeneName==V[w] & pt5rpm[[1]]$DGeneName==d[z] & pt5rpm
    [[1]]$cdr3Length==Peaks[k])]),
sum(pt6rpm[[1]]$RPM[which(pt6rpm[[1]]$JGeneName==J[x] & pt6rpm
    [[1]]$VGeneName==V[w] & pt6rpm[[1]]$DGeneName==d[z] & pt6rpm
    [[1]]$cdr3Length==Peaks[k])]),
sum(pt7rpm[[1]]$RPM[which(pt7rpm[[1]]$JGeneName==J[x] & pt7rpm
    [[1]]$VGeneName==V[w] & pt7rpm[[1]]$DGeneName==d[z] & pt7rpm
    [[1]]$cdr3Length==Peaks[k])]),
sum(pt8rpm[[1]]$RPM[which(pt8rpm[[1]]$JGeneName==J[x] & pt8rpm
    [[1]]$VGeneName==V[w] & pt8rpm[[1]]$DGeneName==d[z] & pt8rpm
[[1]]$cdr3Length==Peaks[k])
sum(pt9rpm[[1]]$RPM[which(pt9rpm[[1]]$JGeneName==J[x] & pt9rpm
[[1]]$VGeneName==V[w] & pt9rpm[[1]]$DGeneName==d[z] & pt9rpm
[[1]]$cdr3Length==Peaks[k])])

}  

# print(controlPeaksi)
# calculates total rpm for each donor
controlPeakssum<-apply(controlPeaksi,2,sum)

# print(controlPeakssum)
# calculates probability of each donor peak
controlProbabilityi<-matrix(ncol=13,nrow=L)
for (l in 1:L) {
controlProbabilityi[l,]<-controlPeaksi[l,]/controlPeakssum
}

# print(controlProbabilityi)
# sets division by zero calculations to zero instead of NaN
for (m in 1:L){
Probabilityi[m,which(Probabilityi[m,]=="NaN")]<-0
controlProbabilityi[m,which(controlProbabilityi[m,]=="NaN")]<-0
}

# calculates the deviance for mean donor or individual donor
Deviancei<-Probabilityi-apply(controlProbabilityi,1,mean)

# Change this line to make it a perturbation from each donor instead of mean
# Deviancei<-Probabilityi-controlProbabilityi

# calculates total deviance per subject
AbsoluteDeviance <- matrix(nrow=1, ncol=13)

for (o in 1:13) {
  AbsoluteDeviance[o] <- 100 * sum(abs(Deviance[,o]) / 2)
}

# print(AbsoluteDeviance)

AbsoluteDeviance

# lists all possible VDJ combinations
location <- expand.grid(J, V, d)

# a is VDJ combination from location object
# function calculates perturbation statistic for each VDJ combination
AbsoluteDeviance <- matrix(nrow=(length(J) * length(V) * length(d)), ncol=13)
for (a in 1:1248) {
  AbsoluteDeviance[a,] <- Perttest(which(J == location[a,1]), which(V == location[a,2]), which(d == location[a,3]), 2)
  print(a)
}

setwd("C:/Users/makowski/Desktop/Archer Code and Files")
save(AbsoluteDeviance, location, file="Perturbationtestmean.RData")
# Run the permutation tests for perturbation test using mean

# set the two groups
GroupGVT<-c(2,5,7,8,11)
GroupGVHD<-c(1,3,4,6,9,10,12,13)
# GVT is 0 and GVHD is 1
Group<-c(1,0,1,1,0,1,0,0,1,1,0,1,1)

# for referencing which VDJ combination
rownames(AbsoluteDeviance)<-c(1:1248)

# load NA values for tests
setwd("C:/Users/makowski/Desktop/Archer Code and Files")
load("VDJNAs.RData")

AbsoluteDeviance.NA<-AbsoluteDeviance[-NAs,]

# run the permutation test
set.seed(123)
B <- 1000
test.stat<-pval<-numeric()
test.stat.b<-matrix(nrow=dim(AbsoluteDeviance.NA)[1], ncol=B)
for (i in 1:dim(AbsoluteDeviance.NA)[1]) {
    test.stat[i]<-mean(AbsoluteDeviance.NA[i,Group==1]) - mean(AbsoluteDeviance.NA[i,Group==0])}
for (j in 1:B) {
    sample<-sample(Group, replace=FALSE)
    test.stat.b[i,j]<-mean(AbsoluteDeviance.NA[i,sample==1]) -
    mean(AbsoluteDeviance.NA[i,sample==0])
}

pval[i]<-sum(abs(test.stat.b[i,]>abs(test.stat[i])))/B
}

fdr<-cbind(p.adjust(pval, method="BH"),c(1:dim(AbsoluteDeviance.NA)[1]))

sum(fdr<0.05)

fdr.order<-fdr[order(pval),]

location.NA<-location[-NAs,]

Groupnames<-factor(Group,levels=c(0,1),labels=c("No GVHD","GVHD"))

#make some plots

png(file = "perttestmean.png", width = 1200, height = 1000, units = "px")
par(mfrow=c(2,2),mar=c(5.1,4.5,4.1,2.1))

plot(groupnames,AbsoluteDeviance.NA[fdr.order[1,2],],main=paste(location.NA[
    fdr.order[1,2],1],location.NA[fdr.order[1,2],2],location.NA[fdr.order
    [1,2],3]),cex.lab=2,cex.main=2,cex.axis=2,ylab="Perturbation")

plot(groupnames,AbsoluteDeviance.NA[fdr.order[2,2],],main=paste(location.NA[
    fdr.order[2,2],1],location.NA[fdr.order[2,2],2],location.NA[fdr.order
    [2,2],3]),cex.lab=2,cex.main=2,cex.axis=2,ylab="Perturbation")

plot(groupnames,AbsoluteDeviance.NA[fdr.order[3,2],],main=paste(location.NA[
    fdr.order[3,2],1],location.NA[fdr.order[3,2],2],location.NA[fdr.order
    [3,2],3]),cex.lab=2,cex.main=2,cex.axis=2,ylab="Perturbation")
dev.off()

library(xtable)

apply(t(AbsoluteDeviance[fdr.order[c(1:4),2],GroupGVT]),2,FUN=median)

perttest.table<-cbind(location.NA[fdr.order[c(1:3),2],2],
apply(t(AbsoluteDeviance.NA[fdr.order[c(1:3),2],GroupGVT]),2,FUN=median),
apply(t(AbsoluteDeviance.NA[fdr.order[c(1:3),2],GroupGVHD]),2,FUN=median),
order[c(1:3),1])

colnames(perttest.table)<-c("JGene","VGene","DGene", "Median No GVHD Perturbation", "Median GVHD Perturbation", "FDR")

xtable(perttest.table,digits=3)

#CDR3 Perturbation Test using individual control

#loaded the needed file

setwd("C:/Users/makowski/Desktop/Archer Code and Files")

load("TcellRPM.RData")

#Function that does the perturbation test for the VDJ families; x is the J gene, w is V gene and z is d gene and y=1 is donor and y=2 is recipient

Perttest<-function(x,w,z,y){
#this part calculates the number of unique CDR3 lengths

L = length(unique(c(unique(pt11rpm[[y]]$cdr3Length[which(pt11rpm[[y]]$JGeneName==J[x] & pt11rpm[[y]]$VGeneName==V[w] & pt11rpm[[y]]$DGeneName==d[z]])
    ,
    unique(pt12rpm[[y]]$cdr3Length[which(pt12rpm[[y]]$JGeneName==J[x] & pt12rpm[[y]]$VGeneName==V[w] & pt12rpm[[y]]$DGeneName==d[z]])
    ,
    unique(pt13rpm[[y]]$cdr3Length[which(pt13rpm[[y]]$JGeneName==J[x] & pt13rpm[[y]]$VGeneName==V[w] & pt13rpm[[y]]$DGeneName==d[z]])
    ,
    unique(pt14rpm[[y]]$cdr3Length[which(pt14rpm[[y]]$JGeneName==J[x] & pt14rpm[[y]]$VGeneName==V[w] & pt14rpm[[y]]$DGeneName==d[z]])
    ,
    unique(pt16rpm[[y]]$cdr3Length[which(pt16rpm[[y]]$JGeneName==J[x] & pt16rpm[[y]]$VGeneName==V[w] & pt16rpm[[y]]$DGeneName==d[z]])
    ,
    unique(pt24rpm[[y]]$cdr3Length[which(pt24rpm[[y]]$JGeneName==J[x] & pt24rpm[[y]]$VGeneName==V[w] & pt24rpm[[y]]$DGeneName==d[z]])
    ,
    unique(pt3rpm[[y]]$cdr3Length[which(pt3rpm[[y]]$JGeneName==J[x] & pt3rpm[[y]]$VGeneName==V[w] & pt3rpm[[y]]$DGeneName==d[z]])
    ,
    unique(pt4rpm[[y]]$cdr3Length[which(pt4rpm[[y]]$JGeneName==J[x] & pt4rpm[[y]]$VGeneName==V[w] & pt4rpm[[y]]$DGeneName==d[z]])
    ,
    unique(pt5rpm[[y]]$cdr3Length[which(pt5rpm[[y]]$JGeneName==J[x] & pt5rpm[[y]]$VGeneName==V[w] & pt5rpm[[y]]$DGeneName==d[z]])
    ,
    unique(pt6rpm[[y]]$cdr3Length[which(pt6rpm[[y]]$JGeneName==J[x] & pt6rpm[[y]]$VGeneName==V[w] & pt6rpm[[y]]$DGeneName==d[z]])
    ,
    unique(pt7rpm[[y]]$cdr3Length[which(pt7rpm[[y]]$JGeneName==J[x] & pt7rpm[[y]]$VGeneName==V[w] & pt7rpm[[y]]$DGeneName==d[z]])
    ,
    unique(pt8rpm[[y]]$cdr3Length[which(pt8rpm[[y]]$JGeneName==J[x] & pt8rpm[[y]]$VGeneName==V[w] & pt8rpm[[y]]$DGeneName==d[z]])
    ));
#this sets the perturbation statistic to zero if the VDJ combination is not present
if (L==0) {
  AbsoluteDeviancei<-rep(0,13)
  AbsoluteDeviancei
}
else {
  #This list the peak lengths
  Peaks=unique(c(unique(pt11rpm[[y]]$cdr3Length[which(pt11rpm[[y]]$JGeneName==J[x] & pt11rpm[[y]]$VGeneName==V[w] & pt11rpm[[y]]$DGeneName==d[z]])),
               unique(pt12rpm[[y]]$cdr3Length[which(pt12rpm[[y]]$JGeneName==J[x] & pt12rpm[[y]]$VGeneName==V[w] & pt12rpm[[y]]$DGeneName==d[z]])),
               unique(pt13rpm[[y]]$cdr3Length[which(pt13rpm[[y]]$JGeneName==J[x] & pt13rpm[[y]]$VGeneName==V[w] & pt13rpm[[y]]$DGeneName==d[z]])),
               unique(pt14rpm[[y]]$cdr3Length[which(pt14rpm[[y]]$JGeneName==J[x] & pt14rpm[[y]]$VGeneName==V[w] & pt14rpm[[y]]$DGeneName==d[z]])),
               unique(pt16rpm[[y]]$cdr3Length[which(pt16rpm[[y]]$JGeneName==J[x] & pt16rpm[[y]]$VGeneName==V[w] & pt16rpm[[y]]$DGeneName==d[z]])),
               unique(pt24rpm[[y]]$cdr3Length[which(pt24rpm[[y]]$JGeneName==J[x] & pt24rpm[[y]]$VGeneName==V[w] & pt24rpm[[y]]$DGeneName==d[z]])},
  print(L)
}

unique(pt3rpm[[y]]$cdr3Length[which(pt3rpm[[y]]$JGeneName==J[x] &
    pt3rpm[[y]]$VGeneName==V[w] & pt3rpm[[y]]$DGeneName==d[z])]),
unique(pt4rpm[[y]]$cdr3Length[which(pt4rpm[[y]]$JGeneName==J[x] &
    pt4rpm[[y]]$VGeneName==V[w] & pt4rpm[[y]]$DGeneName==d[z])]),
unique(pt5rpm[[y]]$cdr3Length[which(pt5rpm[[y]]$JGeneName==J[x] &
    pt5rpm[[y]]$VGeneName==V[w] & pt5rpm[[y]]$DGeneName==d[z])]),
unique(pt6rpm[[y]]$cdr3Length[which(pt6rpm[[y]]$JGeneName==J[x] &
    pt6rpm[[y]]$VGeneName==V[w] & pt6rpm[[y]]$DGeneName==d[z])]),
unique(pt7rpm[[y]]$cdr3Length[which(pt7rpm[[y]]$JGeneName==J[x] &
    pt7rpm[[y]]$VGeneName==V[w] & pt7rpm[[y]]$DGeneName==d[z])]),
unique(pt8rpm[[y]]$cdr3Length[which(pt8rpm[[y]]$JGeneName==J[x] &
    pt8rpm[[y]]$VGeneName==V[w] & pt8rpm[[y]]$DGeneName==d[z])]),
unique(pt9rpm[[y]]$cdr3Length[which(pt9rpm[[y]]$JGeneName==J[x] &
    pt9rpm[[y]]$VGeneName==V[w] & pt9rpm[[y]]$DGeneName==d[z])])
)

# print(Peaks)

# this calculates the total rpm in each peak

Peaksi<-matrix(ncol=13,nrow=L)

for (i in 1:L){
  Peaksi[i,]<-c(sum(pt11rpm[[y]]$RPM[which(pt11rpm[[y]]$JGeneName==J[x] &
    pt11rpm[[y]]$VGeneName==V[w] & pt11rpm[[y]]$DGeneName==d[z] & pt11rpm[[y]]$cdr3Length==Peaks[i])]),
    sum(pt12rpm[[y]]$RPM[which(pt12rpm[[y]]$JGeneName==J[x] &
    pt12rpm[[y]]$VGeneName==V[w] & pt12rpm[[y]]$DGeneName==d[z] &
    pt12rpm[[y]]$cdr3Length==Peaks[i])]),
    sum(pt13rpm[[y]]$RPM[which(pt13rpm[[y]]$JGeneName==J[x] &
    pt13rpm[[y]]$VGeneName==V[w] & pt13rpm[[y]]$DGeneName==d[z] &
    pt13rpm[[y]]$cdr3Length==Peaks[i])])
}
\text{sum}(pt14rpm[[y]]RPM[\text{which}(pt14rpm[[y]]JGeneName==J[x] & pt14rpm[[y]]VGeneName==V[w] & pt14rpm[[y]]DGeneName==d[z] & pt14rpm[[y]]cdr3Length==Peaks[i])]),

\text{sum}(pt16rpm[[y]]RPM[\text{which}(pt16rpm[[y]]JGeneName==J[x] & pt16rpm[[y]]VGeneName==V[w] & pt16rpm[[y]]DGeneName==d[z] & pt16rpm[[y]]cdr3Length==Peaks[i])]),

\text{sum}(pt24rpm[[y]]RPM[\text{which}(pt24rpm[[y]]JGeneName==J[x] & pt24rpm[[y]]VGeneName==V[w] & pt24rpm[[y]]DGeneName==d[z] & pt24rpm[[y]]cdr3Length==Peaks[i])]),

\text{sum}(pt3rpm[[y]]RPM[\text{which}(pt3rpm[[y]]JGeneName==J[x] & pt3rpm[[y]]VGeneName==V[w] & pt3rpm[[y]]DGeneName==d[z] & pt3rpm[[y]]cdr3Length==Peaks[i])]),

\text{sum}(pt4rpm[[y]]RPM[\text{which}(pt4rpm[[y]]JGeneName==J[x] & pt4rpm[[y]]VGeneName==V[w] & pt4rpm[[y]]DGeneName==d[z] & pt4rpm[[y]]cdr3Length==Peaks[i])]),

\text{sum}(pt5rpm[[y]]RPM[\text{which}(pt5rpm[[y]]JGeneName==J[x] & pt5rpm[[y]]VGeneName==V[w] & pt5rpm[[y]]DGeneName==d[z] & pt5rpm[[y]]cdr3Length==Peaks[i])]),

\text{sum}(pt6rpm[[y]]RPM[\text{which}(pt6rpm[[y]]JGeneName==J[x] & pt6rpm[[y]]VGeneName==V[w] & pt6rpm[[y]]DGeneName==d[z] & pt6rpm[[y]]cdr3Length==Peaks[i])]),

\text{sum}(pt7rpm[[y]]RPM[\text{which}(pt7rpm[[y]]JGeneName==J[x] & pt7rpm[[y]]VGeneName==V[w] & pt7rpm[[y]]DGeneName==d[z] & pt7rpm[[y]]cdr3Length==Peaks[i])]),

\text{sum}(pt8rpm[[y]]RPM[\text{which}(pt8rpm[[y]]JGeneName==J[x] & pt8rpm[[y]]VGeneName==V[w] & pt8rpm[[y]]DGeneName==d[z] & pt8rpm[[y]]cdr3Length==Peaks[i])]),

137
sum(pt9rpm[[y]]$RPM[which(pt9rpm[[y]]$JGeneName==J[x] & pt9rpm
[[y]]$VGeneName==V[w] & pt9rpm[[y]]$DGeneName==d[z] & pt9rpm
[[y]]$cdr3Length==Peaks[i])])

}

#total rpm per patient

Peakssum<-apply(Peaksi,2,sum)

#calculates the probability of each peak

Probabilityi<-matrix(ncol=13,nrow=L)

for (j in 1:L) {

Probabilityi[j,]<-Peaksi[j,]/Peakssum
}

#calculates total rpm in each donor peak

controlPeaksi<-matrix(ncol=13,nrow=L)

for (k in 1:L){

controlPeaksi[k,]<-c(sum(pt11rpm[[1]]$RPM[which(pt11rpm[[1]]$JGeneName==J[x]
 & pt11rpm[[1]]$VGeneName==V[w] & pt11rpm[[1]]$DGeneName==d[z] & pt11rpm
[[1]]$cdr3Length==Peaks[k])]),

sum(pt12rpm[[1]]$RPM[which(pt12rpm[[1]]$JGeneName==J[x] & pt12rpm[[1]]$VGeneName==V[w] & pt12rpm[[1]]$DGeneName==d[z]
 & pt12rpm[[1]]$cdr3Length==Peaks[k])]),

sum(pt13rpm[[1]]$RPM[which(pt13rpm[[1]]$JGeneName==J[x] & pt13rpm[[1]]$VGeneName==V[w] & pt13rpm[[1]]$DGeneName==d[z]
& pt13rpm[[1]]$cdr3Length==Peaks[k])]),
sum(pt14rpm[[1]]$RPM[which(pt14rpm[[1]]$JGeneName==J[x] & pt14rpm[[1]]$VGeneName==V[w] & pt14rpm[[1]]$DGeneName==d[z] & pt14rpm[[1]]$cdr3Length==Peaks[k])]),
sum(pt16rpm[[1]]$RPM[which(pt16rpm[[1]]$JGeneName==J[x] & pt16rpm[[1]]$VGeneName==V[w] & pt16rpm[[1]]$DGeneName==d[z] & pt16rpm[[1]]$cdr3Length==Peaks[k])]),
sum(pt24rpm[[1]]$RPM[which(pt24rpm[[1]]$JGeneName==J[x] & pt24rpm[[1]]$VGeneName==V[w] & pt24rpm[[1]]$DGeneName==d[z] & pt24rpm[[1]]$cdr3Length==Peaks[k])]),
sum(pt3rpm[[1]]$RPM[which(pt3rpm[[1]]$JGeneName==J[x] & pt3rpm[[1]]$VGeneName==V[w] & pt3rpm[[1]]$DGeneName==d[z] & pt3rpm[[1]]$cdr3Length==Peaks[k])]),
sum(pt4rpm[[1]]$RPM[which(pt4rpm[[1]]$JGeneName==J[x] & pt4rpm[[1]]$VGeneName==V[w] & pt4rpm[[1]]$DGeneName==d[z] & pt4rpm[[1]]$cdr3Length==Peaks[k])]),
sum(pt5rpm[[1]]$RPM[which(pt5rpm[[1]]$JGeneName==J[x] & pt5rpm[[1]]$VGeneName==V[w] & pt5rpm[[1]]$DGeneName==d[z] & pt5rpm[[1]]$cdr3Length==Peaks[k])]),
sum(pt6rpm[[1]]$RPM[which(pt6rpm[[1]]$JGeneName==J[x] & pt6rpm[[1]]$VGeneName==V[w] & pt6rpm[[1]]$DGeneName==d[z] & pt6rpm[[1]]$cdr3Length==Peaks[k])]),
sum(pt7rpm[[1]]$RPM[which(pt7rpm[[1]]$JGeneName==J[x] & pt7rpm[[1]]$VGeneName==V[w] & pt7rpm[[1]]$DGeneName==d[z] & pt7rpm[[1]]$cdr3Length==Peaks[k])]),
sum(pt8rpm[[1]]$RPM[which(pt8rpm[[1]]$JGeneName==J[x] & pt8rpm[[1]]$VGeneName==V[w] & pt8rpm[[1]]$DGeneName==d[z] & pt8rpm[[1]]$cdr3Length==Peaks[k])]),
430 sum(pt9rpm[[1]]$RPM[which(pt9rpm[[1]]$JGeneName==J[x] & pt9rpm[[1]]$VGeneName==V[w] & pt9rpm[[1]]$DGeneName==d[z] & pt9rpm[[1]]$cdr3Length==Peaks[k]])}

431 }

432 #print(controlPeaksi)
433 #calculates total rpm for each donor
434 controlPeakssum<-apply(controlPeaksi,2,sum)
435
436 #print(controlPeakssum)
437 #calculates probability of each donor peak
438 controlProbabilityi<-matrix(ncol=13,nrow=L)
439 for (l in 1:L) {
440  controlProbabilityi[l,]<-controlPeaksi[l,]/controlPeakssum
441 }

442 #print(controlProbabilityi)
443 #sets division by zero calculations to zero instead of NaN
444 for (m in 1:L){
445  Probabilityi[m,which(Probabilityi[m,]=="NaN")]<-0
446  controlProbabilityi[m,which(controlProbabilityi[m,]=="NaN")]<-0
447 }

449 #calculates the deviance for mean donor or individual donor
450 #Deviancei<-Probabilityi-apply(controlProbabilityi,1,mean)
451 #Change this line to make it a perturbation from each donor instead of mean
452 Deviancei<-Probabilityi-controlProbabilityi
453 #calculates total deviance per subject
AbsoluteDeviancei <- matrix(nrow=1,ncol=13)

for (o in 1:13) {
    AbsoluteDeviancei[o] <- 100*sum(abs(Deviancei[,o])/2)
}

# print(AbsoluteDeviancei)

AbsoluteDeviancei

# lists all possible VDJ combinations
location <- expand.grid(J,V,d)

# a is VDJ combination from location object
# function calculates perturbation statistic for each VDJ combination
AbsoluteDeviance <- matrix(nrow=(length(J)*length(V)*length(d)),ncol=13)

for (a in 1:1248) {
    AbsoluteDeviance[a,] <- Perttest(which(J==location[a,1]),which(V==location[a,2]),which(d==location[a,3]),2)
    print(a)
}

setwd("C:/Users/makowski/Desktop/Archer Code and Files")

save(AbsoluteDeviance, location, file="Perturbationtestdonor.RData")
#Run the permutation tests for perturbation test using individual control

#set the two groups
GroupGVT<-c(2,5,7,8,11)
GroupGVHD<-c(1,3,4,6,9,10,12,13)
#GVT is 0 and GVHD is 1
Group<-c(1,0,1,1,0,1,0,0,1,1,0,1,1)

#for referencing which VDJ combination
rownames(AbsoluteDeviance)<-c(1:1248)

#load NA values for tests
setwd("C:/Users/makowski/Desktop/Archer Code and Files")
load("VDJNAs.RData")

AbsoluteDeviance.NA<-AbsoluteDeviance[-NAs,]

#run the permutation test
set.seed(123)
B <- 1000

test.stat<-pval<-numeric()
test.stat.b<-matrix(nrow=dim(AbsoluteDeviance.NA)[1], ncol=B)
for (i in 1:dim(AbsoluteDeviance.NA)[1]) {
  test.stat[i]<-mean(AbsoluteDeviance.NA[i,Group==1]) - mean(AbsoluteDeviance.NA[i,Group==0])
for (j in 1:B) {
  sample <- sample(Group, replace=FALSE)
  test.stat.b[i,j]<-mean(AbsoluteDeviance.NA[i,sample==1]) - mean(AbsoluteDeviance.NA[i,sample==0])
}
pval[i]<-sum(abs(test.stat.b[i,]>abs(test.stat[i])))#/B
}

fdr<-cbind(p.adjust(pval, method="BH"),c(1:dim(AbsoluteDeviance.NA)[1]))
sum(fdr<0.05)
fdr.order<-fdr[order(pval),]
location.NA<-location[-NAs,]
groupnames<-factor(Group,levels=c(0,1),labels=c("No GVHD","GVHD"))

#make some plots

png(file = "perttestindiv.png", width = 1200, height = 1000, units = "px")
par(mfrow=c(1,2),mar=c(5.1,4.5,4.1,2.1))
plot(groupnames,AbsoluteDeviance.NA[fdr.order[1,2],],main=paste(location.NA[ fdr.order[1,2],1],location.NA[fdr.order[1,2],2],location.NA[fdr.order [1,2],3]),cex.lab=2,cex.main=2,cex.axis=2,ylab="Perturbation")
plot(groupnames,AbsoluteDeviance.NA[fdr.order[2,2],],main=paste(location.NA[ fdr.order[2,2],1],location.NA[fdr.order[2,2],2],location.NA[fdr.order [2,2],3]),cex.lab=2,cex.main=2,cex.axis=2,ylab="Perturbation")
dev.off()
library(xtable)

apply(t(AbsoluteDeviance.NA[fdr.order[c(1:2),2],GroupGVT]),2,FUN=median)
perttest.table<-cbind(location.NA[fdr.order[c(1:2),2],],apply(t(AbsoluteDeviance.NA[fdr.order[c(1:2),2],GroupGVT]),2,FUN=median),apply(t(AbsoluteDeviance.NA[fdr.order[c(1:2),2],GroupGVHD]),2,FUN=median),fdr.order[c(1:2),1])
colnames(perttest.table)<-c("JGene","VGene","DGene", "Median No GVHD Perturbation", "Median GVHD Perturbation", "FDR")
xtable(perttest.table,digits=3)

###############################################################################
#Calculation of Oligoscores
###############################################################################

#loaded the needed file
setwd("C:/Users/makowski/Desktop/Archer Code and Files")
load("TcellRPM.RData")

GroupGVT<-c(2,5,7,8,11)
GroupGVHD<-c(1,3,4,6,9,10,12,13)

#Function that does the oligoscore test for the VDJ families x is the J gene, w is V gene and z is d gene and y=1 is donor and y=2 is recipient
oligotest<-function(x,w,z,y){
  #this part calculates the number of unique CDR3 lengths
L = length(unique(c(unique(pt11rpm[[y]]$cdr3Length[which(pt11rpm[[y]]$JGeneName==J[x] & pt11rpm[[y]]$VGeneName==V[w] & pt11rpm[[y]]$DGeneName==d[z]])),
unique(pt12rpm[[y]]$cdr3Length[which(pt12rpm[[y]]$JGeneName==J[x] &
pt12rpm[[y]]$VGeneName==V[w] & pt12rpm[[y]]$DGeneName==d[z]])),
unique(pt13rpm[[y]]$cdr3Length[which(pt13rpm[[y]]$JGeneName==J[x] &
pt13rpm[[y]]$VGeneName==V[w] & pt13rpm[[y]]$DGeneName==d[z]])),
unique(pt14rpm[[y]]$cdr3Length[which(pt14rpm[[y]]$JGeneName==J[x] &
pt14rpm[[y]]$VGeneName==V[w] & pt14rpm[[y]]$DGeneName==d[z]])),
unique(pt16rpm[[y]]$cdr3Length[which(pt16rpm[[y]]$JGeneName==J[x] &
pt16rpm[[y]]$VGeneName==V[w] & pt16rpm[[y]]$DGeneName==d[z]])),
unique(pt24rpm[[y]]$cdr3Length[which(pt24rpm[[y]]$JGeneName==J[x] &
pt24rpm[[y]]$VGeneName==V[w] & pt24rpm[[y]]$DGeneName==d[z]])),
unique(pt3rpm[[y]]$cdr3Length[which(pt3rpm[[y]]$JGeneName==J[x] &
pt3rpm[[y]]$VGeneName==V[w] & pt3rpm[[y]]$DGeneName==d[z]])),
unique(pt4rpm[[y]]$cdr3Length[which(pt4rpm[[y]]$JGeneName==J[x] &
pt4rpm[[y]]$VGeneName==V[w] & pt4rpm[[y]]$DGeneName==d[z]])),
unique(pt5rpm[[y]]$cdr3Length[which(pt5rpm[[y]]$JGeneName==J[x] &
pt5rpm[[y]]$VGeneName==V[w] & pt5rpm[[y]]$DGeneName==d[z]])),
unique(pt6rpm[[y]]$cdr3Length[which(pt6rpm[[y]]$JGeneName==J[x] &
pt6rpm[[y]]$VGeneName==V[w] & pt6rpm[[y]]$DGeneName==d[z]])),
unique(pt7rpm[[y]]$cdr3Length[which(pt7rpm[[y]]$JGeneName==J[x] &
pt7rpm[[y]]$VGeneName==V[w] & pt7rpm[[y]]$DGeneName==d[z]])),
unique(pt8rpm[[y]]$cdr3Length[which(pt8rpm[[y]]$JGeneName==J[x] &
pt8rpm[[y]]$VGeneName==V[w] & pt8rpm[[y]]$DGeneName==d[z]])),
unique(pt9rpm[[y]]$cdr3Length[which(pt9rpm[[y]]$JGeneName==J[x] &
pt9rpm[[y]]$VGeneName==V[w] & pt9rpm[[y]]$DGeneName==d[z]]))))
#print(L)

#if VDJ combination not present set to zero

if (L==0){
    Peaks<-0
    OligoscoreiGVT<-0
    OligoscoreiGVHD<-0
    cbind(Peaks,OligoscoreiGVT,OligoscoreiGVHD)
}

#this calculates the number of peaks for each subject

else {
    nk=c(length(unique(pt11rpm[[y]]$cdr3Length[which(pt11rpm[[y]]$JGeneName==J[x] & pt11rpm[[y]]$VGeneName==V[w] & pt11rpm[[y]]$DGeneName==d[z])])),
         length(unique(pt12rpm[[y]]$cdr3Length[which(pt12rpm[[y]]$JGeneName==J[x] & pt12rpm[[y]]$VGeneName==V[w] & pt12rpm[[y]]$DGeneName==d[z])])),
         length(unique(pt13rpm[[y]]$cdr3Length[which(pt13rpm[[y]]$JGeneName==J[x] & pt13rpm[[y]]$VGeneName==V[w] & pt13rpm[[y]]$DGeneName==d[z])])),
         length(unique(pt14rpm[[y]]$cdr3Length[which(pt14rpm[[y]]$JGeneName==J[x] & pt14rpm[[y]]$VGeneName==V[w] & pt14rpm[[y]]$DGeneName==d[z])])),
         length(unique(pt16rpm[[y]]$cdr3Length[which(pt16rpm[[y]]$JGeneName==J[x] & pt16rpm[[y]]$VGeneName==V[w] & pt16rpm[[y]]$DGeneName==d[z])])),
         length(unique(pt24rpm[[y]]$cdr3Length[which(pt24rpm[[y]]$JGeneName==J[x] & pt24rpm[[y]]$VGeneName==V[w] & pt24rpm[[y]]$DGeneName==d[z])])),
         length(unique(pt3rpm[[y]]$cdr3Length[which(pt3rpm[[y]]$JGeneName==J[x] & pt3rpm[[y]]$VGeneName==V[w] & pt3rpm[[y]]$DGeneName==d[z])])),
         length(unique(pt4rpm[[y]]$cdr3Length[which(pt4rpm[[y]]$JGeneName==J[x] & pt4rpm[[y]]$VGeneName==V[w] & pt4rpm[[y]]$DGeneName==d[z)]]),

         146}
length(unique(pt5rpm[[y]]$cdr3Length[which(pt5rpm[[y]]$JGeneName==J[x]
 & pt5rpm[[y]]$VGeneName==V[w] & pt5rpm[[y]]$DGeneName==d[z])])),
length(unique(pt6rpm[[y]]$cdr3Length[which(pt6rpm[[y]]$JGeneName==J[x]
 & pt6rpm[[y]]$VGeneName==V[w] & pt6rpm[[y]]$DGeneName==d[z])])),
length(unique(pt7rpm[[y]]$cdr3Length[which(pt7rpm[[y]]$JGeneName==J[x]
 & pt7rpm[[y]]$VGeneName==V[w] & pt7rpm[[y]]$DGeneName==d[z])])),
length(unique(pt8rpm[[y]]$cdr3Length[which(pt8rpm[[y]]$JGeneName==J[x]
 & pt8rpm[[y]]$VGeneName==V[w] & pt8rpm[[y]]$DGeneName==d[z])])),
length(unique(pt9rpm[[y]]$cdr3Length[which(pt9rpm[[y]]$JGeneName==J[x]
 & pt9rpm[[y]]$VGeneName==V[w] & pt9rpm[[y]]$DGeneName==d[z])]))
# print(nk)

# This list the peak lengths

Peaks=unique(c(unique(pt11rpm[[y]]$cdr3Length[which(pt11rpm[[y]]$JGeneName==J[x] & pt11rpm[[y]]$VGeneName==V[w] & pt11rpm[[y]]$DGeneName==d[z])]),
unique(pt12rpm[[y]]$cdr3Length[which(pt12rpm[[y]]$JGeneName==J[x] & pt12rpm[[y]]$VGeneName==V[w] & pt12rpm[[y]]$DGeneName==d[z])]),
unique(pt13rpm[[y]]$cdr3Length[which(pt13rpm[[y]]$JGeneName==J[x] & pt13rpm[[y]]$VGeneName==V[w] & pt13rpm[[y]]$DGeneName==d[z])]),
unique(pt14rpm[[y]]$cdr3Length[which(pt14rpm[[y]]$JGeneName==J[x] & pt14rpm[[y]]$VGeneName==V[w] & pt14rpm[[y]]$DGeneName==d[z])]),
unique(pt16rpm[[y]]$cdr3Length[which(pt16rpm[[y]]$JGeneName==J[x] & pt16rpm[[y]]$VGeneName==V[w] & pt16rpm[[y]]$DGeneName==d[z])]),
unique(pt24rpm[[y]]$cdr3Length[which(pt24rpm[[y]]$JGeneName==J[x] & pt24rpm[[y]]$VGeneName==V[w] & pt24rpm[[y]]$DGeneName==d[z])]),

147
unique(pt3rpm[[y]]$cdr3Length[which(pt3rpm[[y]]$JGeneName==J[x] & pt3rpm[[y]]$VGeneName==V[w] & pt3rpm[[y]]$DGeneName==d[z]))),
unique(pt4rpm[[y]]$cdr3Length[which(pt4rpm[[y]]$JGeneName==J[x] & pt4rpm[[y]]$VGeneName==V[w] & pt4rpm[[y]]$DGeneName==d[z]))),
unique(pt5rpm[[y]]$cdr3Length[which(pt5rpm[[y]]$JGeneName==J[x] & pt5rpm[[y]]$VGeneName==V[w] & pt5rpm[[y]]$DGeneName==d[z]))),
unique(pt6rpm[[y]]$cdr3Length[which(pt6rpm[[y]]$JGeneName==J[x] & pt6rpm[[y]]$VGeneName==V[w] & pt6rpm[[y]]$DGeneName==d[z]))),
unique(pt7rpm[[y]]$cdr3Length[which(pt7rpm[[y]]$JGeneName==J[x] & pt7rpm[[y]]$VGeneName==V[w] & pt7rpm[[y]]$DGeneName==d[z]))),
unique(pt8rpm[[y]]$cdr3Length[which(pt8rpm[[y]]$JGeneName==J[x] & pt8rpm[[y]]$VGeneName==V[w] & pt8rpm[[y]]$DGeneName==d[z]))),
unique(pt9rpm[[y]]$cdr3Length[which(pt9rpm[[y]]$JGeneName==J[x] & pt9rpm[[y]]$VGeneName==V[w] & pt9rpm[[y]]$DGeneName==d[z])))
}

# this calculates the total rpm in each peak

Peaksi<-matrix(ncol=13,nrow=L)
for (i in 1:L){
  Peaksi[i,]<-c(sum(pt11rpm[[y]]$RPM[which(pt11rpm[[y]]$JGeneName==J[x] & pt11rpm[[y]]$VGeneName==V[w] & pt11rpm[[y]]$DGeneName==d[z] & pt11rpm[[y]]$cdr3Length==Peaks[i]])),
  sum(pt12rpm[[y]]$RPM[which(pt12rpm[[y]]$JGeneName==J[x] & pt12rpm[[y]]$VGeneName==V[w] & pt12rpm[[y]]$DGeneName==d[z] & pt12rpm[[y]]$cdr3Length==Peaks[i]])),
  sum(pt13rpm[[y]]$RPM[which(pt13rpm[[y]]$JGeneName==J[x] & pt13rpm[[y]]$VGeneName==V[w] & pt13rpm[[y]]$DGeneName==d[z] & pt13rpm[[y]]$cdr3Length==Peaks[i]]),
}
sum(pt14rpm[[y]]$RPM[which(pt14rpm[[y]]$JGeneName==J[x] & pt14rpm[[y]]$VGeneName==V[w] & pt14rpm[[y]]$DGeneName==d[z] & pt14rpm[[y]]$cdr3Length==Peaks[i]]]),

sum(pt16rpm[[y]]$RPM[which(pt16rpm[[y]]$JGeneName==J[x] & pt16rpm[[y]]$VGeneName==V[w] & pt16rpm[[y]]$DGeneName==d[z] & pt16rpm[[y]]$cdr3Length==Peaks[i]])),

sum(pt24rpm[[y]]$RPM[which(pt24rpm[[y]]$JGeneName==J[x] & pt24rpm[[y]]$VGeneName==V[w] & pt24rpm[[y]]$DGeneName==d[z] & pt24rpm[[y]]$cdr3Length==Peaks[i]])),

sum(pt3rpm[[y]]$RPM[which(pt3rpm[[y]]$JGeneName==J[x] & pt3rpm[[y]]$VGeneName==V[w] & pt3rpm[[y]]$DGeneName==d[z] & pt3rpm[[y]]$cdr3Length==Peaks[i]])),

sum(pt4rpm[[y]]$RPM[which(pt4rpm[[y]]$JGeneName==J[x] & pt4rpm[[y]]$VGeneName==V[w] & pt4rpm[[y]]$DGeneName==d[z] & pt4rpm[[y]]$cdr3Length==Peaks[i]])),

sum(pt5rpm[[y]]$RPM[which(pt5rpm[[y]]$JGeneName==J[x] & pt5rpm[[y]]$VGeneName==V[w] & pt5rpm[[y]]$DGeneName==d[z] & pt5rpm[[y]]$cdr3Length==Peaks[i]])),

sum(pt6rpm[[y]]$RPM[which(pt6rpm[[y]]$JGeneName==J[x] & pt6rpm[[y]]$VGeneName==V[w] & pt6rpm[[y]]$DGeneName==d[z] & pt6rpm[[y]]$cdr3Length==Peaks[i]])),

sum(pt7rpm[[y]]$RPM[which(pt7rpm[[y]]$JGeneName==J[x] & pt7rpm[[y]]$VGeneName==V[w] & pt7rpm[[y]]$DGeneName==d[z] & pt7rpm[[y]]$cdr3Length==Peaks[i]])),

sum(pt8rpm[[y]]$RPM[which(pt8rpm[[y]]$JGeneName==J[x] & pt8rpm[[y]]$VGeneName==V[w] & pt8rpm[[y]]$DGeneName==d[z] & pt8rpm[[y]]$cdr3Length==Peaks[i]])),

149
sum(pt9rpm[[y]]$RPM[which(pt9rpm[[y]]$JGeneName==J[x] & pt9rpm[[y]]$VGeneName==V[w] & pt9rpm[[y]]$DGeneName==d[z] & pt9rpm[[y]]$cdr3Length==Peaks[i])])
}

# total rpm per patient
Peakssum<-apply(Peaksi,2,sum)

# calculates the probability of each peak
Probabilityi<-matrix(ncol=13,nrow=L)
for (j in 1:L) {
  Probabilityi[j,]<-Peaksi[j,]/Peakssum
}

# sets 0 values and NaN values to a low detection threshold as recommended
for (m in 1:L){
  Probabilityi[m,which(Probabilityi[m,]"NaN")]<-1e-6
  Probabilityi[m,which(Probabilityi[m,]==0)]<-1e-6
}

# calculate oligoscore for GVT group
OligoscoreiGVT<-matrix(nrow=L,ncol=1)
for (k in 1:L){
  OligoscoreiGVT[k]<-exp(1)*((prod(Probabilityi[k,GroupGVT]*exp(-nk[GroupGVT]))
  )^(1/length(GroupGVT)))
}
#calculate Oligoscore for GVHD group
OligoscoreGVHD<-matrix(nrow=L,ncol=1)
for (l in 1:L){
  OligoscoreGVHD[l]<-exp(1)*((prod(Probabilityi[l,GroupGVHD]*exp(-nk[GroupGVHD])))^(1/length(GroupGVHD)))
}

#make list of VDJ combos
location<-expand.grid(J,V,d)
#calculate oligoscore for each vdj combination
Oligoscores<-matrix(0,ncol=6)
colnames(Oligoscores)<-c("JGeneName","VGeneName","DGeneName","CDR3Length","GVT","GVHD")
for (a in 1:1248){
  Oligoscoresa<-cbind(location[a,],oligotest(which(J==location[a,1]),which(V==location[a,2]),which(d==location[a,3]),2))
colnames(Oligoscoresa)<-c("JGeneName","VGeneName","DGeneName","CDR3Length","GVT","GVHD")
Oligoscores<-rbind(Oligoscores,Oligoscoresa)
print(a)
Oligoscores<-Oligoscores[-1,]

setwd("C:/Users/makowski/Desktop/Archer Code and Files")
save(Oligoscores,location,file="Oligoscores.RData")

#run Fisher’s exact test
GVHDmatrix<-matrix(c(8,0,2,3),nrow=2,ncol=2)
fisher.test(GVHDmatrix) #p-value = 0.03497
NoGVHDmatrix<-matrix(c(3,5,4,1),nrow=2,ncol=2)
fisher.test(NoGVHDmatrix) #p-value = 0.2657

########################################################
#This section calculated Simpson’s Diversity Index
########################################################

setwd("C:/Users/makowski/Desktop/Archer Code and Files")
load("TcellRPM.RData")

#Function that does the Diversity test for the VDJ families x is the J gene, w is V gene and z is d gene and y=1 is donor and y=2 is recipient

diversitytest<-function(x,w,z,y){
  #this part calculates the number of unique CDR3 lengths
L = length(unique(c(unique(pt11rpm[[y]]$cdr3Length[which(pt11rpm[[y]]$JGeneName==J[x] & pt11rpm[[y]]$VGeneName==V[w] & pt11rpm[[y]]$DGeneName==d[z]])),
unique(pt12rpm[[y]]$cdr3Length[which(pt12rpm[[y]]$JGeneName==J[x] & pt12rpm[[y]]$VGeneName==V[w] & pt12rpm[[y]]$DGeneName==d[z]])),
unique(pt13rpm[[y]]$cdr3Length[which(pt13rpm[[y]]$JGeneName==J[x] & pt13rpm[[y]]$VGeneName==V[w] & pt13rpm[[y]]$DGeneName==d[z]])),
unique(pt14rpm[[y]]$cdr3Length[which(pt14rpm[[y]]$JGeneName==J[x] & pt14rpm[[y]]$VGeneName==V[w] & pt14rpm[[y]]$DGeneName==d[z]])),
unique(pt16rpm[[y]]$cdr3Length[which(pt16rpm[[y]]$JGeneName==J[x] & pt16rpm[[y]]$VGeneName==V[w] & pt16rpm[[y]]$DGeneName==d[z]])),
unique(pt24rpm[[y]]$cdr3Length[which(pt24rpm[[y]]$JGeneName==J[x] & pt24rpm[[y]]$VGeneName==V[w] & pt24rpm[[y]]$DGeneName==d[z]])),
unique(pt3rpm[[y]]$cdr3Length[which(pt3rpm[[y]]$JGeneName==J[x] & pt3rpm[[y]]$VGeneName==V[w] & pt3rpm[[y]]$DGeneName==d[z]])),
unique(pt4rpm[[y]]$cdr3Length[which(pt4rpm[[y]]$JGeneName==J[x] & pt4rpm[[y]]$VGeneName==V[w] & pt4rpm[[y]]$DGeneName==d[z]])),
unique(pt5rpm[[y]]$cdr3Length[which(pt5rpm[[y]]$JGeneName==J[x] & pt5rpm[[y]]$VGeneName==V[w] & pt5rpm[[y]]$DGeneName==d[z]])),
unique(pt6rpm[[y]]$cdr3Length[which(pt6rpm[[y]]$JGeneName==J[x] & pt6rpm[[y]]$VGeneName==V[w] & pt6rpm[[y]]$DGeneName==d[z]])),
unique(pt7rpm[[y]]$cdr3Length[which(pt7rpm[[y]]$JGeneName==J[x] & pt7rpm[[y]]$VGeneName==V[w] & pt7rpm[[y]]$DGeneName==d[z]])),
unique(pt8rpm[[y]]$cdr3Length[which(pt8rpm[[y]]$JGeneName==J[x] & pt8rpm[[y]]$VGeneName==V[w] & pt8rpm[[y]]$DGeneName==d[z]])),
unique(pt9rpm[[y]]$cdr3Length[which(pt9rpm[[y]]$JGeneName==J[x] & pt9rpm[[y]]$VGeneName==V[w] & pt9rpm[[y]]$DGeneName==d[z]]))}
#print(L)
#this sets the perturbation statistic to zero if the VDJ combination is not present
if (L==0){
  Diversity<-rep(0,13)
}

else {
  #This list the peak lengths
  Peaks=unique(c(unique(pt11rpm[[y]]$cdr3Length[which(pt11rpm[[y]]$JGeneName ==J[x] & pt11rpm[[y]]$VGeneName==V[w] & pt11rpm[[y]]$DGeneName==d[z])]),
    unique(pt12rpm[[y]]$cdr3Length[which(pt12rpm[[y]]$JGeneName==J[x] & pt12rpm[[y]]$VGeneName==V[w] & pt12rpm[[y]]$DGeneName==d[z])]),
    unique(pt13rpm[[y]]$cdr3Length[which(pt13rpm[[y]]$JGeneName==J[x] & pt13rpm[[y]]$VGeneName==V[w] & pt13rpm[[y]]$DGeneName==d[z])]),
    unique(pt14rpm[[y]]$cdr3Length[which(pt14rpm[[y]]$JGeneName==J[x] & pt14rpm[[y]]$VGeneName==V[w] & pt14rpm[[y]]$DGeneName==d[z])]),
    unique(pt16rpm[[y]]$cdr3Length[which(pt16rpm[[y]]$JGeneName==J[x] & pt16rpm[[y]]$VGeneName==V[w] & pt16rpm[[y]]$DGeneName==d[z])]),
    unique(pt24rpm[[y]]$cdr3Length[which(pt24rpm[[y]]$JGeneName==J[x] & pt24rpm[[y]]$VGeneName==V[w] & pt24rpm[[y]]$DGeneName==d[z])]),
    unique(pt3rpm[[y]]$cdr3Length[which(pt3rpm[[y]]$JGeneName==J[x] & pt3rpm[[y]]$VGeneName==V[w] & pt3rpm[[y]]$DGeneName==d[z])]),
    unique(pt4rpm[[y]]$cdr3Length[which(pt4rpm[[y]]$JGeneName==J[x] & pt4rpm[[y]]$VGeneName==V[w] & pt4rpm[[y]]$DGeneName==d[z])]),
}
unique(pt5rpm[[y]]$cdr3Length[which(pt5rpm[[y]]$JGeneName==J[x] & pt5rpm[[y]]$VGeneName==V[w] & pt5rpm[[y]]$DGeneName==d[z])]),
unique(pt6rpm[[y]]$cdr3Length[which(pt6rpm[[y]]$JGeneName==J[x] & pt6rpm[[y]]$VGeneName==V[w] & pt6rpm[[y]]$DGeneName==d[z])]),
unique(pt7rpm[[y]]$cdr3Length[which(pt7rpm[[y]]$JGeneName==J[x] & pt7rpm[[y]]$VGeneName==V[w] & pt7rpm[[y]]$DGeneName==d[z])]),
unique(pt8rpm[[y]]$cdr3Length[which(pt8rpm[[y]]$JGeneName==J[x] & pt8rpm[[y]]$VGeneName==V[w] & pt8rpm[[y]]$DGeneName==d[z])]),
unique(pt9rpm[[y]]$cdr3Length[which(pt9rpm[[y]]$JGeneName==J[x] & pt9rpm[[y]]$VGeneName==V[w] & pt9rpm[[y]]$DGeneName==d[z])]))

# print(Peaks)
# this calculates the total rpm in each peak

Peaksi<-matrix(ncol=13,nrow=L)
for (i in 1:L){
  Peaksi[i,]<-c(sum(pt11rpm[[y]]$RPM[which(pt11rpm[[y]]$JGeneName==J[x] & pt11rpm[[y]]$VGeneName==V[w] & pt11rpm[[y]]$DGeneName==d[z] & pt11rpm[[y]]$cdr3Length==Peaks[i])]),
              sum(pt12rpm[[y]]$RPM[which(pt12rpm[[y]]$JGeneName==J[x] & pt12rpm[[y]]$VGeneName==V[w] & pt12rpm[[y]]$DGeneName==d[z] & pt12rpm[[y]]$cdr3Length==Peaks[i])]),
              sum(pt13rpm[[y]]$RPM[which(pt13rpm[[y]]$JGeneName==J[x] & pt13rpm[[y]]$VGeneName==V[w] & pt13rpm[[y]]$DGeneName==d[z] & pt13rpm[[y]]$cdr3Length==Peaks[i])]),
              sum(pt14rpm[[y]]$RPM[which(pt14rpm[[y]]$JGeneName==J[x] & pt14rpm[[y]]$VGeneName==V[w] & pt14rpm[[y]]$DGeneName==d[z] & pt14rpm[[y]]$cdr3Length==Peaks[i]))},

\[
\text{sum}(pt16rpm[[y]]\$RPM[\text{which}(pt16rpm[[y]]\$JGeneName==J[x] \& \linebreak pt16rpm[[y]]\$VGeneName==V[w] \& pt16rpm[[y]]\$DGeneName==d[z] \linebreak \& pt16rpm[[y]]\$cdr3Length==Peaks[i]])], \\
\text{sum}(pt24rpm[[y]]\$RPM[\text{which}(pt24rpm[[y]]\$JGeneName==J[x] \& \linebreak pt24rpm[[y]]\$VGeneName==V[w] \& pt24rpm[[y]]\$DGeneName==d[z] \linebreak \& pt24rpm[[y]]\$cdr3Length==Peaks[i]])], \\
\text{sum}(pt3rpm[[y]]\$RPM[\text{which}(pt3rpm[[y]]\$JGeneName==J[x] \& pt3rpm \linebreak [[y]]\$VGeneName==V[w] \& pt3rpm[[y]]\$DGeneName==d[z] \& pt3rpm \linebreak [[y]]\$cdr3Length==Peaks[i]])], \\
\text{sum}(pt4rpm[[y]]\$RPM[\text{which}(pt4rpm[[y]]\$JGeneName==J[x] \& pt4rpm \linebreak [[y]]\$VGeneName==V[w] \& pt4rpm[[y]]\$DGeneName==d[z] \& pt4rpm \linebreak [[y]]\$cdr3Length==Peaks[i]])], \\
\text{sum}(pt5rpm[[y]]\$RPM[\text{which}(pt5rpm[[y]]\$JGeneName==J[x] \& pt5rpm \linebreak [[y]]\$VGeneName==V[w] \& pt5rpm[[y]]\$DGeneName==d[z] \& pt5rpm \linebreak [[y]]\$cdr3Length==Peaks[i]])], \\
\text{sum}(pt6rpm[[y]]\$RPM[\text{which}(pt6rpm[[y]]\$JGeneName==J[x] \& pt6rpm \linebreak [[y]]\$VGeneName==V[w] \& pt6rpm[[y]]\$DGeneName==d[z] \& pt6rpm \linebreak [[y]]\$cdr3Length==Peaks[i]])], \\
\text{sum}(pt7rpm[[y]]\$RPM[\text{which}(pt7rpm[[y]]\$JGeneName==J[x] \& pt7rpm \linebreak [[y]]\$VGeneName==V[w] \& pt7rpm[[y]]\$DGeneName==d[z] \& pt7rpm \linebreak [[y]]\$cdr3Length==Peaks[i]])], \\
\text{sum}(pt8rpm[[y]]\$RPM[\text{which}(pt8rpm[[y]]\$JGeneName==J[x] \& pt8rpm \linebreak [[y]]\$VGeneName==V[w] \& pt8rpm[[y]]\$DGeneName==d[z] \& pt8rpm \linebreak [[y]]\$cdr3Length==Peaks[i]])], \\
\text{sum}(pt9rpm[[y]]\$RPM[\text{which}(pt9rpm[[y]]\$JGeneName==J[x] \& pt9rpm \linebreak [[y]]\$VGeneName==V[w] \& pt9rpm[[y]]\$DGeneName==d[z] \& pt9rpm \linebreak [[y]]\$cdr3Length==Peaks[i])]) 
\]
Peakssum<-apply(Peaksi,2,sum)
# calculates the diversity statistic
Diversity<-matrix(nrow=1,ncol=13)
for (j in 1:13) {
  Diversity[j]<-1-sum((Peaksi[,j]*(Peaksi[,j]-1))/(Peakssum[j]*(Peakssum[j]-1))
}
# list of VDJ combinations
location<-expand.grid(J,V,d)
# calculates diversity statistic for each VDJ combination
Diversity<-matrix(nrow=(length(J)*length(V)*length(d)),ncol=13)
for (a in 1:1248){
  Diversity[a,]<-diversitytest(which(J==location[a,1]),which(V==location[a,2]),
  which(d==location[a,3]),2)
  print(a)
}
setwd("C:/Users/makowski/Desktop/Archer Code and Files")
save(Diversity,location,file="Diversity.RData")

#############################################
# Run the permutation tests for the Diversity index
#############################################

GroupGVT<-c(2,5,7,8,11)
GroupGVHD<-c(1,3,4,6,9,10,12,13)

# GVT is 0 and GVHD is 1
Group<-c(1,0,1,0,1,0,1,1,0,1,0,1,1)

# for referencing which VDJ combination
rownames(Diversity)<-c(1:1248)

# load NA values for tests
setwd("C:/Users/makowski/Desktop/Archer Code and Files")
load("VDJNAs.RData")

Diversity.NA<-Diversity[-NAs,]

# replace NaN with NA
for (i in 1:13){
  for (j in 1:dim(Diversity.NA)[1]){  
if (Diversity.NA[j,i]=="NaN"){
  Diversity.NA[j,i]<-NA
}
```r
set.seed(123)
B <- 1000
test.stat<-pval<-numeric()
test.stat.b<-matrix(nrow=dim(Diversity.NA)[1], ncol=B)
for (i in 1:dim(Diversity.NA)[1]) {
  test.stat[i]<-mean(Diversity.NA[i,Group==1],na.rm=TRUE) - mean(Diversity.NA[i,Group==0],na.rm=TRUE)
  for (j in 1:B) {
    sample<-sample(Group, replace=FALSE)
    test.stat.b[i,j]<-mean(Diversity.NA[i,sample==1],na.rm=TRUE) - mean(Diversity.NA[i,sample==0],na.rm=TRUE)
  }
  pval[i]<-sum(abs(test.stat.b[i,]>abs(test.stat[i])))#/B
}
fdr<-cbind(p.adjust(pval, method="BH"),c(1:dim(Diversity.NA)[1]))
length(which(fdr[,1]<0.05))
fdr.order<-fdr[order(pval),]
location.NA<-location[-NAs,]
groupnames<-factor(Group,levels=c(0,1),labels=c("No GVHD","GVHD"))
#make some plots
```
```r
png(file = "Diversitytest.png", width = 1200, height = 1000, units = "px")
par(mfrow=c(3,2),mar=c(5.1,4.5,4.1,2.1))
plot(groupnames, Diversity.NA[fdr.order[1,2],], main=paste(location.NA[fdr.
    order[1,2],1], location.NA[fdr.order[1,2],2], location.NA[fdr.order
    [1,2],3]), cex.lab=2, cex.main=2, cex.axis=2, ylab="Diversity")
plot(groupnames, Diversity.NA[fdr.order[2,2],], main=paste(location.NA[fdr.
    order[2,2],1], location.NA[fdr.order[2,2],2], location.NA[fdr.order
    [2,2],3]), cex.lab=2, cex.main=2, cex.axis=2, ylab="Diversity")
plot(groupnames, Diversity.NA[fdr.order[3,2],], main=paste(location.NA[fdr.
    order[3,2],1], location.NA[fdr.order[3,2],2], location.NA[fdr.order
    [3,2],3]), cex.lab=2, cex.main=2, cex.axis=2, ylab="Diversity")
plot(groupnames, Diversity.NA[fdr.order[4,2],], main=paste(location.NA[fdr.
    order[4,2],1], location.NA[fdr.order[4,2],2], location.NA[fdr.order
    [4,2],3]), cex.lab=2, cex.main=2, cex.axis=2, ylab="Diversity")
plot(groupnames, Diversity.NA[fdr.order[5,2],], main=paste(location.NA[fdr.
    order[5,2],1], location.NA[fdr.order[5,2],2], location.NA[fdr.order
    [5,2],3]), cex.lab=2, cex.main=2, cex.axis=2, ylab="Diversity")
dev.off()

library(xtable)
apply(t(Diversity.NA[fdr.order[1:5],2], GroupGVT)), 2, FUN=median, na.rm=TRUE)
perttest.table<-cbind(location.NA[fdr.order[1:5],2], apply(t(Diversity.NA
    [fdr.order[1:5],2], GroupGVT)), 2, FUN=median, na.rm=TRUE), apply(t( Diversity.NA[fdr.order[1:5],2], GroupGVHD)), 2, FUN=median, na.rm=TRUE),
    fdr.order[1:5], 1])
```
colnames(perttest.table)<-c("JGene","VGene","DGene", "Median No GVHD Perturbation", "Median GVHD Perturbation", "FDR")
xtable(perttest.table,digits=3)

#This section calculated Shannon Diversity Index

setwd("C:/Users/makowski/Desktop/Archer Code and Files")
load("TcellRPM.RData")

#Function that does the Diversity test for the VDJ families x is the J gene, w is V gene and z is d gene and y=1 is donor and y=2 is recipient
diversitytest<-function(x,w,z,y){
  #this part calculates the number of unique CDR3 lengths
  L=length(unique(c(unique(pt11rpm[[y]]$cdr3Length[which(pt11rpm[[y]]$JGeneName==J[x] & pt11rpm[[y]]$VGeneName==V[w] & pt11rpm[[y]]$DGeneName==d[z]])),
             unique(pt12rpm[[y]]$cdr3Length[which(pt12rpm[[y]]$JGeneName==J[x] & pt12rpm[[y]]$VGeneName==V[w] & pt12rpm[[y]]$DGeneName==d[z]])),
             unique(pt13rpm[[y]]$cdr3Length[which(pt13rpm[[y]]$JGeneName==J[x] & pt13rpm[[y]]$VGeneName==V[w] & pt13rpm[[y]]$DGeneName==d[z]])),
             unique(pt14rpm[[y]]$cdr3Length[which(pt14rpm[[y]]$JGeneName==J[x] & pt14rpm[[y]]$VGeneName==V[w] & pt14rpm[[y]]$DGeneName==d[z]])),
             unique(pt15rpm[[y]]$cdr3Length[which(pt15rpm[[y]]$JGeneName==J[x] & pt15rpm[[y]]$VGeneName==V[w] & pt15rpm[[y]]$DGeneName==d[z]])),
unique(pt16rpm[[y]]$cdr3Length[which(pt16rpm[[y]]$JGeneName==J[x] & pt16rpm[[y]]$VGeneName==V[w] & pt16rpm[[y]]$DGeneName==d[z])]),
unique(pt24rpm[[y]]$cdr3Length[which(pt24rpm[[y]]$JGeneName==J[x] & pt24rpm[[y]]$VGeneName==V[w] & pt24rpm[[y]]$DGeneName==d[z])]),
unique(pt3rpm[[y]]$cdr3Length[which(pt3rpm[[y]]$JGeneName==J[x] & pt3rpm[[y]]$VGeneName==V[w] & pt3rpm[[y]]$DGeneName==d[z])]),
unique(pt4rpm[[y]]$cdr3Length[which(pt4rpm[[y]]$JGeneName==J[x] & pt4rpm[[y]]$VGeneName==V[w] & pt4rpm[[y]]$DGeneName==d[z])]),
unique(pt5rpm[[y]]$cdr3Length[which(pt5rpm[[y]]$JGeneName==J[x] & pt5rpm[[y]]$VGeneName==V[w] & pt5rpm[[y]]$DGeneName==d[z])]),
unique(pt6rpm[[y]]$cdr3Length[which(pt6rpm[[y]]$JGeneName==J[x] & pt6rpm[[y]]$VGeneName==V[w] & pt6rpm[[y]]$DGeneName==d[z])]),
unique(pt7rpm[[y]]$cdr3Length[which(pt7rpm[[y]]$JGeneName==J[x] & pt7rpm[[y]]$VGeneName==V[w] & pt7rpm[[y]]$DGeneName==d[z])]),
unique(pt8rpm[[y]]$cdr3Length[which(pt8rpm[[y]]$JGeneName==J[x] & pt8rpm[[y]]$VGeneName==V[w] & pt8rpm[[y]]$DGeneName==d[z])]),
unique(pt9rpm[[y]]$cdr3Length[which(pt9rpm[[y]]$JGeneName==J[x] & pt9rpm[[y]]$VGeneName==V[w] & pt9rpm[[y]]$DGeneName==d[z])]))

# print(L)
# this sets the perturbation statistic to zero if the VDJ combination is not present
if (L==0){
  Diversity<-rep(0,13)
  Diversity
}
else {
# This list the peak lengths

```
Peaks = unique(c(unique(pt11rpm[[y]]$cdr3Length[which(pt11rpm[[y]]$JGeneName == J[x] & pt11rpm[[y]]$VGeneName == V[w] & pt11rpm[[y]]$DGeneName == d[z]])),

unique(pt12rpm[[y]]$cdr3Length[which(pt12rpm[[y]]$JGeneName == J[x] & pt12rpm[[y]]$VGeneName == V[w] & pt12rpm[[y]]$DGeneName == d[z]]),

unique(pt13rpm[[y]]$cdr3Length[which(pt13rpm[[y]]$JGeneName == J[x] & pt13rpm[[y]]$VGeneName == V[w] & pt13rpm[[y]]$DGeneName == d[z]]),

unique(pt14rpm[[y]]$cdr3Length[which(pt14rpm[[y]]$JGeneName == J[x] & pt14rpm[[y]]$VGeneName == V[w] & pt14rpm[[y]]$DGeneName == d[z]]),

unique(pt16rpm[[y]]$cdr3Length[which(pt16rpm[[y]]$JGeneName == J[x] & pt16rpm[[y]]$VGeneName == V[w] & pt16rpm[[y]]$DGeneName == d[z]]),

unique(pt24rpm[[y]]$cdr3Length[which(pt24rpm[[y]]$JGeneName == J[x] & pt24rpm[[y]]$VGeneName == V[w] & pt24rpm[[y]]$DGeneName == d[z]]),

unique(pt3rpm[[y]]$cdr3Length[which(pt3rpm[[y]]$JGeneName == J[x] & pt3rpm[[y]]$VGeneName == V[w] & pt3rpm[[y]]$DGeneName == d[z]]),

unique(pt4rpm[[y]]$cdr3Length[which(pt4rpm[[y]]$JGeneName == J[x] & pt4rpm[[y]]$VGeneName == V[w] & pt4rpm[[y]]$DGeneName == d[z]]),

unique(pt5rpm[[y]]$cdr3Length[which(pt5rpm[[y]]$JGeneName == J[x] & pt5rpm[[y]]$VGeneName == V[w] & pt5rpm[[y]]$DGeneName == d[z]]),

unique(pt6rpm[[y]]$cdr3Length[which(pt6rpm[[y]]$JGeneName == J[x] & pt6rpm[[y]]$VGeneName == V[w] & pt6rpm[[y]]$DGeneName == d[z]]),

unique(pt7rpm[[y]]$cdr3Length[which(pt7rpm[[y]]$JGeneName == J[x] & pt7rpm[[y]]$VGeneName == V[w] & pt7rpm[[y]]$DGeneName == d[z]]),

unique(pt8rpm[[y]]$cdr3Length[which(pt8rpm[[y]]$JGeneName == J[x] & pt8rpm[[y]]$VGeneName == V[w] & pt8rpm[[y]]$DGeneName == d[z]]),
```

unique(pt9rpm[[y]]$cdr3Length[which(pt9rpm[[y]]$JGeneName==J[x] &
pt9rpm[[y]]$VGeneName==V[w] & pt9rpm[[y]]$DGeneName==d[z])]))

# print(Peaks)

# this calculates the total rpm in each peak

Peaksi<-matrix(ncol=13,nrow=L)

for (i in 1:L){
  Peaksi[i,]<-c(sum(pt11rpm[[y]]$RPM[which(pt11rpm[[y]]$JGeneName==J[x] &
pt11rpm[[y]]$VGeneName==V[w] & pt11rpm[[y]]$DGeneName==d[z] &
pt11rpm[[y]]$cdr3Length==Peaks[i])]),
  sum(pt12rpm[[y]]$RPM[which(pt12rpm[[y]]$JGeneName==J[x] &
pt12rpm[[y]]$VGeneName==V[w] & pt12rpm[[y]]$DGeneName==d[z] &
pt12rpm[[y]]$cdr3Length==Peaks[i])]),
  sum(pt13rpm[[y]]$RPM[which(pt13rpm[[y]]$JGeneName==J[x] &
pt13rpm[[y]]$VGeneName==V[w] & pt13rpm[[y]]$DGeneName==d[z] &
pt13rpm[[y]]$cdr3Length==Peaks[i])]),
  sum(pt14rpm[[y]]$RPM[which(pt14rpm[[y]]$JGeneName==J[x] &
pt14rpm[[y]]$VGeneName==V[w] & pt14rpm[[y]]$DGeneName==d[z] &
pt14rpm[[y]]$cdr3Length==Peaks[i])]),
  sum(pt16rpm[[y]]$RPM[which(pt16rpm[[y]]$JGeneName==J[x] &
pt16rpm[[y]]$VGeneName==V[w] & pt16rpm[[y]]$DGeneName==d[z] &
pt16rpm[[y]]$cdr3Length==Peaks[i])]),
  sum(pt24rpm[[y]]$RPM[which(pt24rpm[[y]]$JGeneName==J[x] &
pt24rpm[[y]]$VGeneName==V[w] & pt24rpm[[y]]$DGeneName==d[z] &
pt24rpm[[y]]$cdr3Length==Peaks[i])]),
  sum(pt3rpm[[y]]$RPM[which(pt3rpm[[y]]$JGeneName==J[x] &
pt3rpm[[y]]$VGeneName==V[w] & pt3rpm[[y]]$DGeneName==d[z] &
pt3rpm[[y]]$cdr3Length==Peaks[i])]),
}

164


```r
sum(pt4rpm[[y]]$RPM[which(pt4rpm[[y]]$JGeneName==J[x] & pt4rpm[[y]]$VGeneName==V[w] & pt4rpm[[y]]$DGeneName==d[z] & pt4rpm[[y]]$cdr3Length==Peaks[i]])],
sum(pt5rpm[[y]]$RPM[which(pt5rpm[[y]]$JGeneName==J[x] & pt5rpm[[y]]$VGeneName==V[w] & pt5rpm[[y]]$DGeneName==d[z] & pt5rpm[[y]]$cdr3Length==Peaks[i]])],
sum(pt6rpm[[y]]$RPM[which(pt6rpm[[y]]$JGeneName==J[x] & pt6rpm[[y]]$VGeneName==V[w] & pt6rpm[[y]]$DGeneName==d[z] & pt6rpm[[y]]$cdr3Length==Peaks[i]])],
sum(pt7rpm[[y]]$RPM[which(pt7rpm[[y]]$JGeneName==J[x] & pt7rpm[[y]]$VGeneName==V[w] & pt7rpm[[y]]$DGeneName==d[z] & pt7rpm[[y]]$cdr3Length==Peaks[i]])],
sum(pt8rpm[[y]]$RPM[which(pt8rpm[[y]]$JGeneName==J[x] & pt8rpm[[y]]$VGeneName==V[w] & pt8rpm[[y]]$DGeneName==d[z] & pt8rpm[[y]]$cdr3Length==Peaks[i]])],
sum(pt9rpm[[y]]$RPM[which(pt9rpm[[y]]$JGeneName==J[x] & pt9rpm[[y]]$VGeneName==V[w] & pt9rpm[[y]]$DGeneName==d[z] & pt9rpm[[y]]$cdr3Length==Peaks[i]])])
}

# total rpm per patient
Peakssum<-apply(Peaksi,2,sum)

# calculates the diversity statistic
Diversity<-matrix(nrow=1,ncol=13)
for (j in 1:13) {
  Diversity[j]<-(-1)*sum((Peaksi[,j]*log(Peaksi[,j])),na.rm=TRUE)
}
```

---

165
#print(Diversity)

Diversity

#list of VDJ combinations

location<-expand.grid(J,V,d)

#calculates diversity statistic for each VDJ combination

# a is VDJ combination

Diversity<-matrix(nrow=(length(J)*length(V)*length(d)),ncol=13)

for (a in 1:1248){
  Diversity[a,]<-diversitytest(which(J==location[a,1]),which(V==location[a,2]),
          which(d==location[a,3]),2)

  print(a)
}

save(Diversity,location,file="ShannaonDiversity.RData")

#Run the permutation tests for the Diversity index

GroupGVT<-c(2,5,7,8,11)

GroupGVHD<-c(1,3,4,6,9,10,12,13)
# GVT is 0 and GVHD is 1
Group<-c(1,0,1,0,1,0,0,1,1,0,1,1)

# for referencing which VDJ combination
rownames(Diversity)<-c(1:1248)

# load NA values for tests
setwd("C:/Users/makowski/Desktop/Archer Code and Files")
load("VDJNAs.RData")

Diversity.NA<-Diversity[-NAs,] }

set.seed(123)
B <- 1000
test.stat<-pval<-numeric()
test.stat.b<-matrix(nrow=dim(Diversity.NA)[1], ncol=B)
for (i in 1:dim(Diversity.NA)[1]) {
  test.stat[i]<-mean(Diversity.NA[i,Group==1],na.rm=TRUE) - mean(
    Diversity.NA[i,Group==0],na.rm=TRUE)
  for (j in 1:B) {
    sample<-sample(Group, replace=FALSE)
    test.stat.b[i,j]<-mean(Diversity.NA[i,sample==1],na.rm=TRUE)
    - mean(Diversity.NA[i,sample==0],na.rm=TRUE)
  }
  pval[i]<-sum(abs(test.stat.b[i,]>abs(test.stat[i]))) / B
}
fdr<-cbind(p.adjust(pval, method="BH"),c(1:dim(Diversity.NA)[1]))

length(which(fdr[,1]<0.05))

fdr.order<-fdr[order(pval),]

location.NA<-location[-NAs,]

groupnames<-factor(Group,levels=c(0,1),labels=c("No GVHD","GVHD"))

#make some plots

png(file = "DiversityShannon.png", width = 1200, height = 1000, units = "px"
  )

par(mfrow=c(3,2),mar=c(5.1,4.5,4.1,2.1))

plot(groupnames,Diversity.NA[fdr.order[1,2],],main=paste(location.NA[fdr.
    order[1,2],1],location.NA[fdr.order[1,2],2],location.NA[fdr.order
    [1,2],3]),cex.lab=2,cex.main=2,cex.axis=2,ylab="Diversity")

plot(groupnames,Diversity.NA[fdr.order[2,2],],main=paste(location.NA[fdr.
    order[2,2],1],location.NA[fdr.order[2,2],2],location.NA[fdr.order
    [2,2],3]),cex.lab=2,cex.main=2,cex.axis=2,ylab="Diversity")

plot(groupnames,Diversity.NA[fdr.order[3,2],],main=paste(location.NA[fdr.
    order[3,2],1],location.NA[fdr.order[3,2],2],location.NA[fdr.order
    [3,2],3]),cex.lab=2,cex.main=2,cex.axis=2,ylab="Diversity")

plot(groupnames,Diversity.NA[fdr.order[4,2],],main=paste(location.NA[fdr.
    order[4,2],1],location.NA[fdr.order[4,2],2],location.NA[fdr.order
    [4,2],3]),cex.lab=2,cex.main=2,cex.axis=2,ylab="Diversity")

plot(groupnames,Diversity.NA[fdr.order[5,2],],main=paste(location.NA[fdr.
    order[5,2],1],location.NA[fdr.order[5,2],2],location.NA[fdr.order
    [5,2],3]),cex.lab=2,cex.main=2,cex.axis=2,ylab="Diversity")
plot(groupnames,Diversity.NA[fdr.order[6,2],],main=paste(location.NA[fdr.order[6,2],1],location.NA[fdr.order[6,2],2],location.NA[fdr.order[6,2],3]),cex.lab=2,cex.main=2,cex.axis=2,ylab="Diversity")

dev.off()

library(xtable)

apply(t(Diversity.NA[fdr.order[c(1:5),2],GroupGVT]),2,FUN=median,na.rm=TRUE)
perttest.table<-cbind(location.NA[fdr.order[c(1:6),2],],apply(t(Diversity.NA[fdr.order[c(1:6),2],GroupGVT]),2,FUN=median,na.rm=TRUE),apply(t(Diversity.NA[fdr.order[c(1:6),2],GroupGVHD]),2,FUN=median,na.rm=TRUE),pval[fdr.order[c(1:6),2]])
colnames(perttest.table)<-c("JGene","VGene","DGene", "Median No GVHD Perturbation", "Median GVHD Perturbation", "p-value")
xtable(perttest.table,digits=3)

#########################################################
#This section computes the K-L Divergence test statistics
#########################################################

setwd("C:/Users/makowski/Desktop/Archer Code and Files")
load("TcellRPM.RData")

#Function that does the distribution test for the VDJ families x is the J
gene v is the Vgene and w is the d gene and y=1 is donor and y=2 is
VDJKLtest<-function(x,v,w,y){

#this part calculates the number of unique CDR3 lengths
L=length(unique(c(unique(pt11rpm[[y]]$cdr3Length[which(pt11rpm[[y]]$JGeneName==J[x] & pt11rpm[[y]]$VGeneName==V[v] & pt11rpm[[y]]$DGeneName==d[w]]),
unique(pt12rpm[[y]]$cdr3Length[which(pt12rpm[[y]]$JGeneName==J[x] & pt12rpm[[y]]$VGeneName==V[v] & pt12rpm[[y]]$DGeneName==d[w]]),
unique(pt13rpm[[y]]$cdr3Length[which(pt13rpm[[y]]$JGeneName==J[x] & pt13rpm[[y]]$VGeneName==V[v] & pt13rpm[[y]]$DGeneName==d[w]]),
unique(pt14rpm[[y]]$cdr3Length[which(pt14rpm[[y]]$JGeneName==J[x] & pt14rpm[[y]]$VGeneName==V[v] & pt14rpm[[y]]$DGeneName==d[w]]),
unique(pt16rpm[[y]]$cdr3Length[which(pt16rpm[[y]]$JGeneName==J[x] & pt16rpm[[y]]$VGeneName==V[v] & pt16rpm[[y]]$DGeneName==d[w]]),
unique(pt24rpm[[y]]$cdr3Length[which(pt24rpm[[y]]$JGeneName==J[x] & pt24rpm[[y]]$VGeneName==V[v] & pt24rpm[[y]]$DGeneName==d[w]]),
unique(pt3rpm[[y]]$cdr3Length[which(pt3rpm[[y]]$JGeneName==J[x] & pt3rpm[[y]]$VGeneName==V[v] & pt3rpm[[y]]$DGeneName==d[w]]),
unique(pt4rpm[[y]]$cdr3Length[which(pt4rpm[[y]]$JGeneName==J[x] & pt4rpm[[y]]$VGeneName==V[v] & pt4rpm[[y]]$DGeneName==d[w]]),
unique(pt5rpm[[y]]$cdr3Length[which(pt5rpm[[y]]$JGeneName==J[x] & pt5rpm[[y]]$VGeneName==V[v] & pt5rpm[[y]]$DGeneName==d[w]]),
unique(pt6rpm[[y]]$cdr3Length[which(pt6rpm[[y]]$JGeneName==J[x] & pt6rpm[[y]]$VGeneName==V[v] & pt6rpm[[y]]$DGeneName==d[w]]),
unique(pt7rpm[[y]]$cdr3Length[which(pt7rpm[[y]]$JGeneName==J[x] & pt7rpm[[y]]$VGeneName==V[v] & pt7rpm[[y]]$DGeneName==d[w]]),...
unique(pt8rpm[[y]]$cdr3Length[which(pt8rpm[[y]]$JGeneName==J[x] & pt8rpm[[y]]$VGeneName==V[v] & pt8rpm[[y]]$DGeneName==d[w])]),
unique(pt9rpm[[y]]$cdr3Length[which(pt9rpm[[y]]$JGeneName==J[x] & pt9rpm[[y]]$VGeneName==V[v] & pt9rpm[[y]]$DGeneName==d[w])])).

#print(L)

#files2 <- ls(pattern="rpm$")
#files2<-files2[c(-1,-15)]
#keeps only VDJ combinations that have at least 2 peaks
if (L<2){
VDJdist<-c("NA","NA","NA")
list(VDJdist[1],VDJdist[2],VDJdist[3])
}

else {

#This lists the CDR3 lengths

cdr3L<-unique(c(unique(pt11rpm[[y]]$cdr3Length[which(pt11rpm[[y]]$JGeneName==J[x] & pt11rpm[[y]]$VGeneName==V[v] & pt11rpm[[y]]$DGeneName==d[w])]),
unique(pt12rpm[[y]]$cdr3Length[which(pt12rpm[[y]]$JGeneName==J[x] & pt12rpm[[y]]$VGeneName==V[v] & pt12rpm[[y]]$DGeneName==d[w])]),
unique(pt13rpm[[y]]$cdr3Length[which(pt13rpm[[y]]$JGeneName==J[x] & pt13rpm[[y]]$VGeneName==V[v] & pt13rpm[[y]]$DGeneName==d[w])]),
unique(pt14rpm[[y]]$cdr3Length[which(pt14rpm[[y]]$JGeneName==J[x] & pt14rpm[[y]]$VGeneName==V[v] & pt14rpm[[y]]$DGeneName==d[w])]).
```
unique(pt16rpm[[y]]$cdr3Length[which(pt16rpm[[y]]$JGeneName==J[x] & pt16rpm[[y]]$VGeneName==V[v] & pt16rpm[[y]]$DGeneName==d[w])]),
unique(pt24rpm[[y]]$cdr3Length[which(pt24rpm[[y]]$JGeneName==J[x] & pt24rpm[[y]]$VGeneName==V[v] & pt24rpm[[y]]$DGeneName==d[w])]),
unique(pt3rpm[[y]]$cdr3Length[which(pt3rpm[[y]]$JGeneName==J[x] & pt3rpm[[y]]$VGeneName==V[v] & pt3rpm[[y]]$DGeneName==d[w])]),
unique(pt4rpm[[y]]$cdr3Length[which(pt4rpm[[y]]$JGeneName==J[x] & pt4rpm[[y]]$VGeneName==V[v] & pt4rpm[[y]]$DGeneName==d[w])]),
unique(pt5rpm[[y]]$cdr3Length[which(pt5rpm[[y]]$JGeneName==J[x] & pt5rpm[[y]]$VGeneName==V[v] & pt5rpm[[y]]$DGeneName==d[w])]),
unique(pt6rpm[[y]]$cdr3Length[which(pt6rpm[[y]]$JGeneName==J[x] & pt6rpm[[y]]$VGeneName==V[v] & pt6rpm[[y]]$DGeneName==d[w])]),
unique(pt7rpm[[y]]$cdr3Length[which(pt7rpm[[y]]$JGeneName==J[x] & pt7rpm[[y]]$VGeneName==V[v] & pt7rpm[[y]]$DGeneName==d[w])]),
unique(pt8rpm[[y]]$cdr3Length[which(pt8rpm[[y]]$JGeneName==J[x] & pt8rpm[[y]]$VGeneName==V[v] & pt8rpm[[y]]$DGeneName==d[w])]),
unique(pt9rpm[[y]]$cdr3Length[which(pt9rpm[[y]]$JGeneName==J[x] & pt9rpm[[y]]$VGeneName==V[v] & pt9rpm[[y]]$DGeneName==d[w])]),

#this calculates an overall abundance for each CDR3 length
overall<-function(z){
  sum(sum(pt11rpm[[y]]$RPM[which(pt11rpm[[y]]$JGeneName==J[x] & pt11rpm[[y]]$cdr3Length==cdr3L[z] & pt11rpm[[y]]$VGeneName==V[v] & pt11rpm[[y]]$DGeneName==d[w]])),
  sum(pt12rpm[[y]]$RPM[which(pt12rpm[[y]]$JGeneName==J[x] & pt12rpm[[y]]$cdr3Length==cdr3L[z] & pt12rpm[[y]]$VGeneName==V[v] & pt12rpm[[y]]$DGeneName==d[w]])),
```
sum(pt13rpm[[y]]$RPM[which(pt13rpm[[y]]$JGeneName==J[x] & pt13rpm[[y]]$cdr3Length==cdr3L[z] & pt13rpm[[y]]$VGeneName==V[v] & pt13rpm[[y]]$DGeneName==d[w])]),

sum(pt14rpm[[y]]$RPM[which(pt14rpm[[y]]$JGeneName==J[x] & pt14rpm[[y]]$cdr3Length==cdr3L[z] & pt14rpm[[y]]$VGeneName==V[v] & pt14rpm[[y]]$DGeneName==d[w])]),

sum(pt16rpm[[y]]$RPM[which(pt16rpm[[y]]$JGeneName==J[x] & pt16rpm[[y]]$cdr3Length==cdr3L[z] & pt16rpm[[y]]$VGeneName==V[v] & pt16rpm[[y]]$DGeneName==d[w])]),

sum(pt24rpm[[y]]$RPM[which(pt24rpm[[y]]$JGeneName==J[x] & pt24rpm[[y]]$cdr3Length==cdr3L[z] & pt24rpm[[y]]$VGeneName==V[v] & pt24rpm[[y]]$DGeneName==d[w])]),

sum(pt3rpm[[y]]$RPM[which(pt3rpm[[y]]$JGeneName==J[x] & pt3rpm[[y]]$cdr3Length==cdr3L[z] & pt3rpm[[y]]$VGeneName==V[v] & pt3rpm[[y]]$DGeneName==d[w])]),

sum(pt4rpm[[y]]$RPM[which(pt4rpm[[y]]$JGeneName==J[x] & pt4rpm[[y]]$cdr3Length==cdr3L[z] & pt4rpm[[y]]$VGeneName==V[v] & pt4rpm[[y]]$DGeneName==d[w])]),

sum(pt5rpm[[y]]$RPM[which(pt5rpm[[y]]$JGeneName==J[x] & pt5rpm[[y]]$cdr3Length==cdr3L[z] & pt5rpm[[y]]$VGeneName==V[v] & pt5rpm[[y]]$DGeneName==d[w])]),

sum(pt6rpm[[y]]$RPM[which(pt6rpm[[y]]$JGeneName==J[x] & pt6rpm[[y]]$cdr3Length==cdr3L[z] & pt6rpm[[y]]$VGeneName==V[v] & pt6rpm[[y]]$DGeneName==d[w])]),

sum(pt7rpm[[y]]$RPM[which(pt7rpm[[y]]$JGeneName==J[x] & pt7rpm[[y]]$cdr3Length==cdr3L[z] & pt7rpm[[y]]$VGeneName==V[v] & pt7rpm[[y]]$DGeneName==d[w])]),

173
sum(pt8rpm[[y]]$RPM[which(pt8rpm[[y]]$JGeneName==J[x] & pt8rpm[[y]]$cdr3Length==cdr3L[z] & pt8rpm[[y]]$VGeneName==V[v] & pt8rpm[[y]]$DGeneName==d[w])]),
sum(pt9rpm[[y]]$RPM[which(pt9rpm[[y]]$JGeneName==J[x] & pt9rpm[[y]]$cdr3Length==cdr3L[z] & pt9rpm[[y]]$VGeneName==V[v] & pt9rpm[[y]]$DGeneName==d[w])]),
}

# create matrix to store all these values
rbar<-matrix(nrow=L)

for (i in 1:L){
  rbar[i]<-overall(i)/13
}

# print(rbar)

# these two functions calculate the relative abundances for each group
GVT<-function(z){
  sum(sum(pt12rpm[[y]]$RPM[which(pt12rpm[[y]]$JGeneName==J[x] & pt12rpm[[y]]$cdr3Length==cdr3L[z] & pt12rpm[[y]]$VGeneName==V[v] & pt12rpm[[y]]$DGeneName==d[w])]),
  sum(pt16rpm[[y]]$RPM[which(pt16rpm[[y]]$JGeneName==J[x] & pt16rpm[[y]]$cdr3Length==cdr3L[z] & pt16rpm[[y]]$VGeneName==V[v] & pt16rpm[[y]]$DGeneName==d[w])]),
  sum(pt3rpm[[y]]$RPM[which(pt3rpm[[y]]$JGeneName==J[x] & pt3rpm[[y]]$cdr3Length==cdr3L[z] & pt3rpm[[y]]$VGeneName==V[v] & pt3rpm[[y]]$DGeneName==d[w])]),
GVHD <- function(z) {

  sum(sum(pt11rpm[[y]]$RPM[which(pt11rpm[[y]]$JGeneName==J[x] & pt11rpm[[y]]$
cdr3Length==cdr3L[z] & pt11rpm[[y]]$VGeneName==V[v] & pt11rpm[[y]]$
DGeneName==d[w]])），

  sum(pt13rpm[[y]]$RPM[which(pt13rpm[[y]]$JGeneName==J[x] & pt13rpm[[y]]$
cdr3Length==cdr3L[z] & pt13rpm[[y]]$VGeneName==V[v] & pt13rpm[[y]]$
DGeneName==d[w]])），

  sum(pt14rpm[[y]]$RPM[which(pt14rpm[[y]]$JGeneName==J[x] & pt14rpm[[y]]$
cdr3Length==cdr3L[z] & pt14rpm[[y]]$VGeneName==V[v] & pt14rpm[[y]]$
DGeneName==d[w]])），

  sum(pt24rpm[[y]]$RPM[which(pt24rpm[[y]]$JGeneName==J[x] & pt24rpm[[y]]$
cdr3Length==cdr3L[z] & pt24rpm[[y]]$VGeneName==V[v] & pt24rpm[[y]]$
DGeneName==d[w]])），

  sum(pt5rpm[[y]]$RPM[which(pt5rpm[[y]]$JGeneName==J[x] & pt5rpm[[y]]$
cdr3Length==cdr3L[z] & pt5rpm[[y]]$VGeneName==V[v] & pt5rpm[[y]]$
DGeneName==d[w]])），

  sum(pt6rpm[[y]]$RPM[which(pt6rpm[[y]]$JGeneName==J[x] & pt6rpm[[y]]$
cdr3Length==cdr3L[z] & pt6rpm[[y]]$VGeneName==V[v] & pt6rpm[[y]]$
DGeneName==d[w]]))，

  ...
sum(pt8rpm[[y]]$RPM[which(pt8rpm[[y]]$JGeneName==J[x] & pt8rpm[[y]]$cdr3Length==cdr3L[z] & pt8rpm[[y]]$VGeneName==V[v] & pt8rpm[[y]]$DGeneName==d[w])]),
sum(pt9rpm[[y]]$RPM[which(pt9rpm[[y]]$JGeneName==J[x] & pt9rpm[[y]]$cdr3Length==cdr3L[z] & pt9rpm[[y]]$VGeneName==V[v] & pt9rpm[[y]]$DGeneName==d[w])]),
}

G=2

# get average for each group
rgroup<-matrix(nrow=L,ncol=G)
for (i in 1:L){
  rgroup[i,1]<-GVT(i)/5
  rgroup[i,2]<-GVHD(i)/8
}
# print(rgroup)
# this calculated abundances for each subject
indiv<-function(z){
cbind(sum(pt11rpm[[y]]$RPM[which(pt11rpm[[y]]$JGeneName==J[x] & pt11rpm[[y]]$cdr3Length==cdr3L[z] & pt11rpm[[y]]$VGeneName==V[v] & pt11rpm[[y]]$DGeneName==d[w])]),
sum(pt12rpm[[y]]$RPM[which(pt12rpm[[y]]$JGeneName==J[x] & pt12rpm[[y]]$cdr3Length==cdr3L[z] & pt12rpm[[y]]$VGeneName==V[v] & pt12rpm[[y]]$DGeneName==d[w])]),
sum(pt13rpm[[y]]$RPM[which(pt13rpm[[y]]$JGeneName==J[x] & pt13rpm[[y]]$cdr3Length==cdr3L[z] & pt13rpm[[y]]$VGeneName==V[v] & pt13rpm[[y]]$DGeneName==d[w])],
sum(pt14rpm[[y]]$RPM[which(pt14rpm[[y]]$JGeneName==J[x] & pt14rpm[[y]]$cdr3Length==cdr3L[z] & pt14rpm[[y]]$VGeneName==V[v] & pt14rpm[[y]]$DGeneName==d[w])]),
sum(pt16rpm[[y]]$RPM[which(pt16rpm[[y]]$JGeneName==J[x] & pt16rpm[[y]]$cdr3Length==cdr3L[z] & pt16rpm[[y]]$VGeneName==V[v] & pt16rpm[[y]]$DGeneName==d[w])]),
sum(pt24rpm[[y]]$RPM[which(pt24rpm[[y]]$JGeneName==J[x] & pt24rpm[[y]]$cdr3Length==cdr3L[z] & pt24rpm[[y]]$VGeneName==V[v] & pt24rpm[[y]]$DGeneName==d[w])]),
sum(pt3rpm[[y]]$RPM[which(pt3rpm[[y]]$JGeneName==J[x] & pt3rpm[[y]]$cdr3Length==cdr3L[z] & pt3rpm[[y]]$VGeneName==V[v] & pt3rpm[[y]]$DGeneName==d[w])]),
sum(pt4rpm[[y]]$RPM[which(pt4rpm[[y]]$JGeneName==J[x] & pt4rpm[[y]]$cdr3Length==cdr3L[z] & pt4rpm[[y]]$VGeneName==V[v] & pt4rpm[[y]]$DGeneName==d[w])]),
sum(pt5rpm[[y]]$RPM[which(pt5rpm[[y]]$JGeneName==J[x] & pt5rpm[[y]]$cdr3Length==cdr3L[z] & pt5rpm[[y]]$VGeneName==V[v] & pt5rpm[[y]]$DGeneName==d[w])]),
sum(pt6rpm[[y]]$RPM[which(pt6rpm[[y]]$JGeneName==J[x] & pt6rpm[[y]]$cdr3Length==cdr3L[z] & pt6rpm[[y]]$VGeneName==V[v] & pt6rpm[[y]]$DGeneName==d[w])]),
sum(pt7rpm[[y]]$RPM[which(pt7rpm[[y]]$JGeneName==J[x] & pt7rpm[[y]]$cdr3Length==cdr3L[z] & pt7rpm[[y]]$VGeneName==V[v] & pt7rpm[[y]]$DGeneName==d[w])]),
sum(pt8rpm[[y]]$RPM[which(pt8rpm[[y]]$JGeneName==J[x] & pt8rpm[[y]]$cdr3Length==cdr3L[z] & pt8rpm[[y]]$VGeneName==V[v] & pt8rpm[[y]]$DGeneName==d[w])]),
```r
sum(pt9rpm[[y]]$RPM[which(pt9rpm[[y]]$JGeneName==J[x] & pt9rpm[[y]]$cdr3Length==cdr3L[z] & pt9rpm[[y]]$VGeneName==V[v] & pt9rpm[[y]]$DGeneName==d[w]])
```

# puts the values for each subject into a matrix

```r
rindiv<-matrix(ncol=13,nrow=L)
for (i in 1:L) {
  rindiv[i,]<-indiv(i)
}
```

# print(rindiv)

# set 0 values to a minimum detection threshold

```r
for (m in 1:L){
  rindiv[m,which(rindiv[m,]==0)]<-1e-6
}
```

```r
for (m in 1:L){
  rgroup[m,which(rgroup[m,]==0)]<-1e-6
}
```

```r
for (m in 1:L){
  rbar[which(rbar==0)]<-1e-6
}
```

# Calculates the between group and within group divergence

```r
178
```
bgd<-abs(sum(rgroup*log(rgroup/cbind(rbar,rbar))))

# print(bgd)

wgd<-sum(sum(rindiv[,c(2,5,11,7,8)]*log(rindiv[,c(2,5,11,7,8)]/cbind(rgroup[,1],rgroup[,1],rgroup[,1],rgroup[,1],rgroup[,1])),
        rindiv[,c(-2,-5,-11,-7,-8)]*log(rindiv[,c(-2,-5,-11,-7,-8)]/cbind(rgroup[,2],rgroup[,2],rgroup[,2],rgroup[,2],rgroup[,2],rgroup[,2],rgroup[,2],rgroup[,2],rgroup[,2]),na.rm=TRUE),na.rm=TRUE)

# print(wgd)

ndot=13

# uses all the parts to run the F-test

Fstat<-((ndot-G+1)*bgd)/((G-1)*wgd)

Ftest<-Fstat/(qf(c(.025,.975), (L-1)*(G-1), (L-1)*(ndot-G+1)))

pval<- pf(c(.05), (L-1)*(G-1), (L-1)*(ndot-G+1))

list(Fstat,Ftest,pval)

}

}

VDJdist<-matrix(nrow=(length(J)*length(V)*length(d)),ncol=2)
colnames(VDJdist)<-c("Fstat","pval")

# list of VDJ combinations

location<-expand.grid(J,V,d)
# a is VDJ combination
# runs KL divergence test for each VDJ combination
for (a in 1:1248) {
  VDJKL<- VDJKLtest(which(J==location[a,1]),which(V==location[a,2]),which(d==
    location[a,3]),2)
  VDJdist[a,1]<-VDJKL[[1]]
  VDJdist[a,2]<-VDJKL[[3]]
  print(a)
}
setwd("C:/Users/makowski/Desktop/Archer Code and Files")
save(VDJdist,location,file="KLtest.RData")
#save(VDJdist,file="VDJKLtest.RData")
#remove missing values
load("VDJNAs.RData")
rownames(VDJdist)<-c(1:1248)
#run the FDR correction on pvalues
results.dist<-p.adjust(as.numeric(VDJdist[-NAs,2]), method="BH")
results.dist<-data.frame(pval=results.dist,index=c(1:(1248-length(NAs))))
results.dist<-results.dist[order(as.numeric(VDJdist[-NAs,2])),]
#This section calculates the non-parametric test

```r
setwd("C:/Users/makowski/Desktop/Archer Code and Files")
load("TcellRPM.RData")

#Function that does the non-parametric test for the VDJ families x is the J gene, w is V gene and z is d gene and y=1 is donor and y=2 is recipient
alphatest<-function(x,w,z,y){
  L=length(unique(c(unique(pt11rpm[[y]]$cdr3Length[which(pt11rpm[[y]]$JGeneName==J[x] & pt11rpm[[y]]$VGeneName==V[w] & pt11rpm[[y]]$DGeneName==d[z]])),
            unique(pt12rpm[[y]]$cdr3Length[which(pt12rpm[[y]]$JGeneName==J[x] & pt12rpm[[y]]$VGeneName==V[w] & pt12rpm[[y]]$DGeneName==d[z]])),
            unique(pt13rpm[[y]]$cdr3Length[which(pt13rpm[[y]]$JGeneName==J[x] & pt13rpm[[y]]$VGeneName==V[w] & pt13rpm[[y]]$DGeneName==d[z]])),
            unique(pt14rpm[[y]]$cdr3Length[which(pt14rpm[[y]]$JGeneName==J[x] & pt14rpm[[y]]$VGeneName==V[w] & pt14rpm[[y]]$DGeneName==d[z]])),
            unique(pt16rpm[[y]]$cdr3Length[which(pt16rpm[[y]]$JGeneName==J[x] & pt16rpm[[y]]$VGeneName==V[w] & pt16rpm[[y]]$DGeneName==d[z]])),
            unique(pt24rpm[[y]]$cdr3Length[which(pt24rpm[[y]]$JGeneName==J[x] & pt24rpm[[y]]$VGeneName==V[w] & pt24rpm[[y]]$DGeneName==d[z]])),
            unique(pt3rpm[[y]]$cdr3Length[which(pt3rpm[[y]]$JGeneName==J[x] & pt3rpm[[y]]$VGeneName==V[w] & pt3rpm[[y]]$DGeneName==d[z]]))
  )
```

#this part calculates the number of unique CDR3 lengths
L=length(unique(c(unique(pt11rpm[[y]]$cdr3Length[which(pt11rpm[[y]]$JGeneName==J[x] & pt11rpm[[y]]$VGeneName==V[w] & pt11rpm[[y]]$DGeneName==d[z]])),
             unique(pt12rpm[[y]]$cdr3Length[which(pt12rpm[[y]]$JGeneName==J[x] & pt12rpm[[y]]$VGeneName==V[w] & pt12rpm[[y]]$DGeneName==d[z]])),
             unique(pt13rpm[[y]]$cdr3Length[which(pt13rpm[[y]]$JGeneName==J[x] & pt13rpm[[y]]$VGeneName==V[w] & pt13rpm[[y]]$DGeneName==d[z]])),
             unique(pt14rpm[[y]]$cdr3Length[which(pt14rpm[[y]]$JGeneName==J[x] & pt14rpm[[y]]$VGeneName==V[w] & pt14rpm[[y]]$DGeneName==d[z]])),
             unique(pt16rpm[[y]]$cdr3Length[which(pt16rpm[[y]]$JGeneName==J[x] & pt16rpm[[y]]$VGeneName==V[w] & pt16rpm[[y]]$DGeneName==d[z]])),
             unique(pt24rpm[[y]]$cdr3Length[which(pt24rpm[[y]]$JGeneName==J[x] & pt24rpm[[y]]$VGeneName==V[w] & pt24rpm[[y]]$DGeneName==d[z]])),
             unique(pt3rpm[[y]]$cdr3Length[which(pt3rpm[[y]]$JGeneName==J[x] & pt3rpm[[y]]$VGeneName==V[w] & pt3rpm[[y]]$DGeneName==d[z]]))
```
unique(pt4rpm[[y]]$cdr3Length[which(pt4rpm[[y]]$JGeneName==J[x] & pt4rpm[[y]]$VGeneName==V[w] & pt4rpm[[y]]$DGeneName==d[z])]),
unique(pt5rpm[[y]]$cdr3Length[which(pt5rpm[[y]]$JGeneName==J[x] & pt5rpm[[y]]$VGeneName==V[w] & pt5rpm[[y]]$DGeneName==d[z])]),
unique(pt6rpm[[y]]$cdr3Length[which(pt6rpm[[y]]$JGeneName==J[x] & pt6rpm[[y]]$VGeneName==V[w] & pt6rpm[[y]]$DGeneName==d[z])]),
unique(pt7rpm[[y]]$cdr3Length[which(pt7rpm[[y]]$JGeneName==J[x] & pt7rpm[[y]]$VGeneName==V[w] & pt7rpm[[y]]$DGeneName==d[z])]),
unique(pt8rpm[[y]]$cdr3Length[which(pt8rpm[[y]]$JGeneName==J[x] & pt8rpm[[y]]$VGeneName==V[w] & pt8rpm[[y]]$DGeneName==d[z])]),
unique(pt9rpm[[y]]$cdr3Length[which(pt9rpm[[y]]$JGeneName==J[x] & pt9rpm[[y]]$VGeneName==V[w] & pt9rpm[[y]]$DGeneName==d[z])]))

# print(L)
# keeps only VDJ combinations that have at least 2 non-zero peaks
if (L<2){
alphai<-rep(0,13)
alphai
}
else {
# list present peaks
Peaks=unique(c(unique(pt11rpm[[y]]$cdr3Length[which(pt11rpm[[y]]$JGeneName==J[x] & pt11rpm[[y]]$VGeneName==V[w] & pt11rpm[[y]]$DGeneName==d[z])]),
unique(pt12rpm[[y]]$cdr3Length[which(pt12rpm[[y]]$JGeneName==J[x] & pt12rpm[[y]]$VGeneName==V[w] & pt12rpm[[y]]$DGeneName==d[z])]),

182
unique(pt13rpm[[y]]$cdr3Length[which(pt13rpm[[y]]$JGeneName==J[x] &
pt13rpm[[y]]$VGeneName==V[w] & pt13rpm[[y]]$DGeneName==d[z])]),
unique(pt14rpm[[y]]$cdr3Length[which(pt14rpm[[y]]$JGeneName==J[x] &
pt14rpm[[y]]$VGeneName==V[w] & pt14rpm[[y]]$DGeneName==d[z])]),
unique(pt16rpm[[y]]$cdr3Length[which(pt16rpm[[y]]$JGeneName==J[x] &
pt16rpm[[y]]$VGeneName==V[w] & pt16rpm[[y]]$DGeneName==d[z])]),
unique(pt24rpm[[y]]$cdr3Length[which(pt24rpm[[y]]$JGeneName==J[x] &
pt24rpm[[y]]$VGeneName==V[w] & pt24rpm[[y]]$DGeneName==d[z])]),
unique(pt3rpm[[y]]$cdr3Length[which(pt3rpm[[y]]$JGeneName==J[x] &
pt3rpm[[y]]$VGeneName==V[w] & pt3rpm[[y]]$DGeneName==d[z])]),
unique(pt4rpm[[y]]$cdr3Length[which(pt4rpm[[y]]$JGeneName==J[x] &
pt4rpm[[y]]$VGeneName==V[w] & pt4rpm[[y]]$DGeneName==d[z])]),
unique(pt5rpm[[y]]$cdr3Length[which(pt5rpm[[y]]$JGeneName==J[x] &
pt5rpm[[y]]$VGeneName==V[w] & pt5rpm[[y]]$DGeneName==d[z])]),
unique(pt6rpm[[y]]$cdr3Length[which(pt6rpm[[y]]$JGeneName==J[x] &
pt6rpm[[y]]$VGeneName==V[w] & pt6rpm[[y]]$DGeneName==d[z])]),
unique(pt7rpm[[y]]$cdr3Length[which(pt7rpm[[y]]$JGeneName==J[x] &
pt7rpm[[y]]$VGeneName==V[w] & pt7rpm[[y]]$DGeneName==d[z])]),
unique(pt8rpm[[y]]$cdr3Length[which(pt8rpm[[y]]$JGeneName==J[x] &
pt8rpm[[y]]$VGeneName==V[w] & pt8rpm[[y]]$DGeneName==d[z])]),
unique(pt9rpm[[y]]$cdr3Length[which(pt9rpm[[y]]$JGeneName==J[x] &
pt9rpm[[y]]$VGeneName==V[w] & pt9rpm[[y]]$DGeneName==d[z])]))

# print(Peaks)
# calculates total rpm per peak
Peaksi<-matrix(ncol=13,nrow=L)
for (i in 1:L){
\[\text{Peaksi}[i] \leftarrow \text{c(sum}(\text{pt11rpm}[[y]]\$RPM[\text{which}(\text{pt11rpm}[[y]]\$JGeneName==J[x] \& pt11rpm[[y]]\$VGeneName==V[w] \& pt11rpm[[y]]\$DGeneName==d[z] \& pt11rpm[[y]]\$cdr3Length==\text{Peaks}[i]))),\]
\[\text{sum}(\text{pt12rpm}[[y]]\$RPM[\text{which}(\text{pt12rpm}[[y]]\$JGeneName==J[x] \& pt12rpm[[y]]\$VGeneName==V[w] \& pt12rpm[[y]]\$DGeneName==d[z] \& pt12rpm[[y]]\$cdr3Length==\text{Peaks}[i]))),\]
\[\text{sum}(\text{pt13rpm}[[y]]\$RPM[\text{which}(\text{pt13rpm}[[y]]\$JGeneName==J[x] \& pt13rpm[[y]]\$VGeneName==V[w] \& pt13rpm[[y]]\$DGeneName==d[z] \& pt13rpm[[y]]\$cdr3Length==\text{Peaks}[i]))),\]
\[\text{sum}(\text{pt14rpm}[[y]]\$RPM[\text{which}(\text{pt14rpm}[[y]]\$JGeneName==J[x] \& pt14rpm[[y]]\$VGeneName==V[w] \& pt14rpm[[y]]\$DGeneName==d[z] \& pt14rpm[[y]]\$cdr3Length==\text{Peaks}[i]))),\]
\[\text{sum}(\text{pt16rpm}[[y]]\$RPM[\text{which}(\text{pt16rpm}[[y]]\$JGeneName==J[x] \& pt16rpm[[y]]\$VGeneName==V[w] \& pt16rpm[[y]]\$DGeneName==d[z] \& pt16rpm[[y]]\$cdr3Length==\text{Peaks}[i]))),\]
\[\text{sum}(\text{pt24rpm}[[y]]\$RPM[\text{which}(\text{pt24rpm}[[y]]\$JGeneName==J[x] \& pt24rpm[[y]]\$VGeneName==V[w] \& pt24rpm[[y]]\$DGeneName==d[z] \& pt24rpm[[y]]\$cdr3Length==\text{Peaks}[i]))),\]
\[\text{sum}(\text{pt3rpm}[[y]]\$RPM[\text{which}(\text{pt3rpm}[[y]]\$JGeneName==J[x] \& pt3rpm[[y]]\$VGeneName==V[w] \& pt3rpm[[y]]\$DGeneName==d[z] \& pt3rpm[[y]]\$cdr3Length==\text{Peaks}[i]))),\]
\[\text{sum}(\text{pt4rpm}[[y]]\$RPM[\text{which}(\text{pt4rpm}[[y]]\$JGeneName==J[x] \& pt4rpm[[y]]\$VGeneName==V[w] \& pt4rpm[[y]]\$DGeneName==d[z] \& pt4rpm[[y]]\$cdr3Length==\text{Peaks}[i]))),\]
\[\text{sum}(\text{pt5rpm}[[y]]\$RPM[\text{which}(\text{pt5rpm}[[y]]\$JGeneName==J[x] \& pt5rpm[[y]]\$VGeneName==V[w] \& pt5rpm[[y]]\$DGeneName==d[z] \& pt5rpm[[y]]\$cdr3Length==\text{Peaks}[i]))),\]

184
sum(pt6rpm[[y]]$RPM[which(pt6rpm[[y]]$JGeneName==J[x] & pt6rpm[[y]]$VGeneName==V[w] & pt6rpm[[y]]$DGeneName==d[z] & pt6rpm[[y]]$cdr3Length==Peaks[i])]),

sum(pt7rpm[[y]]$RPM[which(pt7rpm[[y]]$JGeneName==J[x] & pt7rpm[[y]]$VGeneName==V[w] & pt7rpm[[y]]$DGeneName==d[z] & pt7rpm[[y]]$cdr3Length==Peaks[i])]),

sum(pt8rpm[[y]]$RPM[which(pt8rpm[[y]]$JGeneName==J[x] & pt8rpm[[y]]$VGeneName==V[w] & pt8rpm[[y]]$DGeneName==d[z] & pt8rpm[[y]]$cdr3Length==Peaks[i])]),

sum(pt9rpm[[y]]$RPM[which(pt9rpm[[y]]$JGeneName==J[x] & pt9rpm[[y]]$VGeneName==V[w] & pt9rpm[[y]]$DGeneName==d[z] & pt9rpm[[y]]$cdr3Length==Peaks[i])])

)

#print(Peaksi)

# calculates total rpm for patient with VDJ combination

Peakssum<-apply(Peaksi,2,sum)

# print(Peakssum)

# calculates probability of each peak

Probabilityi<-matrix(ncol=13,nrow=L)

for (j in 1:L) {
    Probabilityi[j,]<-Peaksi[j,]/Peakssum
}

# set NaN to zero

for (m in 1:L){
    Probabilityi[m,which(Probabilityi[m,]=="NaN")]<-0
}


# print(Probabilityi)

# calculate variance of each peak
sigma <- sqrt(Probabilityi * (1 - Probabilityi))
# print(sigma)

# set p for linear model
p <- matrix(rep(c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9), 1), nrow=9, ncol=1)

cdf <- matrix(nrow=9, ncol=13)

# calculate cdf for each peak
for(k in 1:13){
  for(o in 1:9){
    cdf[o, k] <- (1/L) * sum(pnorm(p[o] - Probabilityi[, k], 0, sigma[, k]))
  }
}

# print(cdf)

alphai <- matrix(nrow=1, ncol=13)

# calculate alpha value using lm
for(l in 1:13){
  alpha <- lm(log10(cdf[, l]) ~ log10(p))
  alphai[l] <- alpha$coef[1]
}

# print(alphai)
#plot(log10(p),log10(cdf[,1]))

alphai
}
}

# list of VDJ combinations
location<-expand.grid(J,V,d)

# a is VDJ combination
# calculates alpha values for each VDJ combination
alphapower<-matrix(nrow=(length(J)*length(V)*length(d)),ncol=13)
for (a in 1:1248){
  alphapower[a,]<-alphatest(which(J==location[a,1]),which(V==location[a,2]),
                              which(d==location[a,3]),2)
  print(a)
}

setwd("C:/Users/makowski/Desktop/Archer Code and Files")
save(alphapower,location,file="AlphaTest.RData")
# Run the permutation tests for group differences

GroupGVT<-c(2,5,7,8,11)
GroupGVHD<-c(1,3,4,6,9,10,12,13)

# GVT is 0 and GVHD is 1
Group<-c(1,0,1,1,0,1,0,0,1,1,0,1,1)

# for referencing which VDJ combination
rownames(alphapower)<-c(1:1248)

# load NA values for tests
setwd("C:/Users/makowski/Desktop/Archer Code and Files")
load("VDJNAs.RData")

alphapower.NA<-alphapower[-NAs,]

set.seed(123)
B <- 1000
test.stat<-pval<-numeric()
test.stat.b<-matrix(nrow=dim(alphapower.NA)[1], ncol=B)
for (i in 1:dim(alphapower.NA)[1]) {
    test.stat[i]<-mean(alphapower.NA[i,Group==1],na.rm=TRUE) - mean(alphapower.NA[i,Group==0],na.rm=TRUE)
for (j in 1:B) {
  sample<-sample(Group, replace=FALSE)
  test.stat.b[i,j]<-mean(alphapower.NA[i,sample==1],na.rm=TRUE) - mean(alphapower.NA[i,sample==0],na.rm=TRUE)
}
pval[i]<-sum(abs(test.stat.b[i,]>abs(test.stat[i]))) / B
}

fdr<-cbind(p.adjust(pval, method="BH"),c(1:dim(alphapower.NA)[1]))
length(which(fdr[,1]<0.05))
fdr.order<-fdr[order(pval),]
location.NA<-location[-NAs,]
groupnames<-factor(Group,levels=c(0,1),labels=c("No GVHD","GVHD"))
# make some plots
png(file = "nonparametrictest.png", width = 1200, height = 1000, units = "px")
par(mfrow=c(1,1),mar=c(5.1,4.5,4.1,2.1))
plot(groupnames,alphapower.NA[fdr.order[1,2],],main=paste(location.NA[fdr.order[1,2],1],location.NA[fdr.order[1,2],2],location.NA[fdr.order[1,2],3]),cex.lab=2,cex.main=2,cex.axis=2,ylab="Alpha")
dev.off()

library(xtable)
apply(t(alphapower.NA[fdr.order[c(1:1),2],GroupGVT]),2,FUN=median,na.rm=TRUE)

perttest.table<-cbind(location.NA[fdr.order[c(1:1),2],],median(alphapower.NA[fdr.order[c(1:1),2],GroupGVT],na.rm=TRUE),median(alphapower.NA[fdr.order[c(1:1),2],GroupGVHD],na.rm=TRUE),fdr.order[c(1:1),1])

colnames(perttest.table)<-c("JGene","VGene","DGene", "Median No GVHD Perturbation", "Median GVHD Perturbation", "FDR")
xtable(perttest.table,digits=3)

#This section runs the logit transformation

countrpm<-function(patient){
  counta<-matrix(nrow=624,ncol=5)
  for (i in 1:48) {
    for (j in 1:13) {
      for (d in 2:2) {
        pt3d.VJ<-subset(patient[[1]],VGeneName==V[i] & JGeneName==J[j] & DGeneName==Dg[d])
      }
    }
  }
  pt3r.VJ<-subset(patient[[2]],VGeneName==V[i] & JGeneName==J[j] & DGeneName==Dg[d])
}
pt3<-merge(pt3d.VJ,pt3r.VJ,by.x="aminoAcid",by.y="aminoAcid",all.x=TRUE,all.y=TRUE)
unique.3<-sum(pt3$RPM.y[which(!is.na(pt3$RPM.y)=="TRUE" & !is.na(pt3$RPM.x)=="FALSE")])
shared.3<-sum(pt3$RPM.y[which(!is.na(pt3$RPM.y)=="TRUE" & !is.na(pt3$RPM.x)=="TRUE")])
counta[((i-1)*13)+j,]<-cbind(V[i],J[j],Dg[d],unique.3,shared.3)
}
}
}
countb<-matrix(nrow=624,ncol=5)
for (i in 1:48) {
for (j in 1:13) {
for (d in 3:3) {
pt3d.VJ<-subset(patient[[1]],VGeneName==V[i] & JGeneName==J[j] & DGeneName==Dg[d])
pt3r.VJ<-subset(patient[[2]],VGeneName==V[i] & JGeneName==J[j] & DGeneName==Dg[d])
pt3<-merge(pt3d.VJ,pt3r.VJ,by.x="aminoAcid",by.y="aminoAcid",all.x=TRUE,all.y=TRUE)
unique.3<-sum(pt3$RPM.y[which(!is.na(pt3$RPM.y)=="TRUE" & !is.na(pt3$RPM.x)=="FALSE")])
shared.3<-sum(pt3$RPM.y[which(!is.na(pt3$RPM.y)=="TRUE" & !is.na(pt3$RPM.x)=="TRUE")])
countb[((i-1)*13)+j,]<-cbind(V[i],J[j],Dg[d],unique.3,shared.3)
}
count <- rbind(counta, countb)
colnames(count) <- c("VGene", "JGene", "Dgene", "Unique", "Shared")
list(count)

count3 <- countrpm(pt3rpm)
count4 <- countrpm(pt4rpm)
count5 <- countrpm(pt5rpm)
count6 <- countrpm(pt6rpm)
count7 <- countrpm(pt7rpm)
count8 <- countrpm(pt8rpm)
count9 <- countrpm(pt9rpm)
count11 <- countrpm(pt11rpm)
count12 <- countrpm(pt12rpm)
count13 <- countrpm(pt13rpm)
count14 <- countrpm(pt14rpm)
count16 <- countrpm(pt16rpm)
count24 <- countrpm(pt24rpm)

save(count3, count4, count5, count6, count7, count8, count9, count11, count12, count13, count14, count16, count24, file = "TcellCount.RData")

# function to do calculate proportion and logit transformation
fam2<-function(fam){
  fam<-cbind(fam[[1]],as.numeric(fam[[1]][,4])/(as.numeric(fam[[1]][,4])+as.numeric(fam[[1]][,5])))
  colnames(fam)[6]<-"proportion"
  fam<-cbind(fam,as.numeric(fam[,6])/(1-as.numeric(fam[,6])))
  colnames(fam)[7]<-"transform"
  return(fam)
}

fam11<-fam2(count11)
fam12<-fam2(count12)
fam13<-fam2(count13)
fam14<-fam2(count14)
fam16<-fam2(count16)
fam24<-fam2(count24)
fam3<-fam2(count3)
fam4<-fam2(count4)
fam5<-fam2(count5)
fam6<-fam2(count6)
fam7<-fam2(count7)
fam8<-fam2(count8)
fam9<-fam2(count9)

famVDJ<-cbind(as.numeric(fam11[,7]),
               as.numeric(fam12[,7]),
               as.numeric(fam13[,7]),
               as.numeric(fam14[,7]),
as.numeric(fam16[,7]),
as.numeric(fam24[,7]),
as.numeric(fam3[,7]),
as.numeric(fam4[,7]),
as.numeric(fam5[,7]),
as.numeric(fam6[,7]),
as.numeric(fam7[,7]),
as.numeric(fam8[,7]),
as.numeric(fam9[,7]))

colnames(famVDJ)<-cbind("fam11$transform",
"fam12$transform",
"fam13$transform",
"fam14$transform",
"fam16$transform",
"fam24$transform",
"fam3$transform",
"fam4$transform",
"fam5$transform",
"fam6$transform",
"fam7$transform",
"fam8$transform",
"fam9$transform")

#replace inf and NaN with 0
for (i in 1:1248){

for (j in 1:13) {
    if (is.nan(famVDJ[i,j]) == "TRUE") {
        famVDJ[i,j] <- 0
    } else if (famVDJ[i,j] == "Inf") {
        famVDJ[i,j] <- 0
    }
}

# Reorder rows to match other analyses
load("VDJobjects.RData")
location <- expand.grid(J, V, d)

order.logit <- matrix(nrow = 1248, ncol = 1)
for (i in 1:1248) {
    order.logit[i] <- which(location[,1] == count3[[1]][i,2] & location[,2] == count3
                           [[1]][i,1] & location[,3] == count3[[1]][i,3])
}

# For referencing which VDJ combination
rownames(famVDJ) <- c(1:1248)

# Load NA values for tests
setwd("C:/Users/makowski/Desktop/Archer Code and Files")
load("VDJNAs.RData")

famVDJ.NAs <- famVDJ[-NAs,]
# group indicator

group<-c("GVHD","GVT","GVHD","GVHD","GVT","GVT","GVHD","GVHD","GVT","GVHD","GVHD")

# run linear model to test differences
res.lm<-matrix(nrow=1248-length(NAs),ncol=2)
for (i in 1:(1248-length(NAs))){
  prop.lm<-lm(famVDJ.NAs[i,]~as.factor(group))
  pv.lm<-summary(prop.lm)$coef[2, "Pr>|t|"]
  cv.lm<-coef(prop.lm)[2]
  res.lm[i,]<-cbind(pv.lm,cv.lm)
}

colnames(res.lm)<-c("pvalue","coefficient")

fdr<-cbind(p.adjust(res.lm[,1], method="BH"),c(1:dim(famVDJ.NAs)[1]))
length(which(fdr[,1]<0.05))
fdr.order<-fdr[order(res.lm[,1]),]
location.NA<-location[-NAs,]
#GVT is 0 and GVHD is 1
Group<-c(1,0,1,1,0,1,0,0,1,1,0,1,1)
groupnames<-factor(Group,levels=c(0,1),labels=c("No GVHD","GVHD"))

# make some plots
png(file = "famVDJ.png", width = 1200, height = 1000, units = "px")
par(mfrow=c(1,2),mar=c(5.1,4.5,4.1,2.1))
plot(groupnames,famVDJ.NAs[fdr.order[1,2],],main=paste(location.NA[fdr.order[1,2],1],location.NA[fdr.order[1,2],2],location.NA[fdr.order[1,2],3]),
cex.lab=2,cex.main=2,cex.axis=2,ylab="Logit Proportion Shared")

plot(groupnames,famVDJ.NAs[fdr.order[2,2],],main=paste(location.NA[fdr.order
[2,2],1],location.NA[fdr.order[2,2],2],location.NA[fdr.order[2,2],3]),
cex.lab=2,cex.main=2,cex.axis=2,ylab="Logit Proportion Shared")

dev.off()

GroupGVT<-c(2,5,7,8,11)
GroupGVHD<-c(1,3,4,6,9,10,12,13)

library(xtable)

apply(t(famVDJ.NAs[fdr.order[c(1:2),2],GroupGVT]),2,FUN=median,na.rm=TRUE)
perttest.table<-cbind(location.NA[fdr.order[c(1:2),2],],apply(t(famVDJ.NAs[
fdr.order[c(1:2),2],GroupGVT]),2,FUN=median,na.rm=TRUE),apply(t(famVDJ.
NAs[fdr.order[c(1:2),2],GroupGVHD]),2,FUN=median,na.rm=TRUE),res.lm[fdr.
order[c(1:2),2],1])
colnames(perttest.table)<-c("JGene","VGene","DGene", "Median No GVHD
Perturbation", "Median GVHD Perturbation", "FDR")
xtable(perttest.table,digits=3)
6.2 Compositional Code

```
setwd("C:/Users/makowski/Desktop/Archer Code and Files")
load("TcellRPM.RData")
load("VDJobjects.RData")
library(compositions)

# Sum all VDJ combinations
location<-expand.grid(J,V,d)
VDJcomp<-matrix(ncol=1248,nrow=13)
for (i in 1:1248){
y<-2
x<-which(J==location[i,1])
w<-which(V==location[i,2])
z<-which(d==location[i,3])
VDJcomp[,i]<-c(sum(pt11rpm[[y]]$RPM[which(pt11rpm[[y]]$JGeneName==J[x] & pt11rpm[[y]]$VGeneName==V[w] & pt11rpm[[y]]$DGeneName==d[z])]),
sum(pt12rpm[[y]]$RPM[which(pt12rpm[[y]]$JGeneName==J[x] & pt12rpm[[y]]$VGeneName==V[w] & pt12rpm[[y]]$DGeneName==d[z])]),
sum(pt13rpm[[y]]$RPM[which(pt13rpm[[y]]$JGeneName==J[x] & pt13rpm[[y]]$VGeneName==V[w] & pt13rpm[[y]]$DGeneName==d[z])]),
sum(pt14rpm[[y]]$RPM[which(pt14rpm[[y]]$JGeneName==J[x] & pt14rpm[[y]]$VGeneName==V[w] & pt14rpm[[y]]$DGeneName==d[z])]),
```

198
sum(pt16rpm[[y]]$RPM[which(pt16rpm[[y]]$JGeneName==J[x] &
    pt16rpm[[y]]$VGeneName==V[w] & pt16rpm[[y]]$DGeneName==d[z])]),

sum(pt24rpm[[y]]$RPM[which(pt24rpm[[y]]$JGeneName==J[x] &
    pt24rpm[[y]]$VGeneName==V[w] & pt24rpm[[y]]$DGeneName==d[z])]),

sum(pt3rpm[[y]]$RPM[which(pt3rpm[[y]]$JGeneName==J[x] & pt3rpm
[[y]]$VGeneName==V[w] & pt3rpm[[y]]$DGeneName==d[z])]),

sum(pt4rpm[[y]]$RPM[which(pt4rpm[[y]]$JGeneName==J[x] & pt4rpm
[[y]]$VGeneName==V[w] & pt4rpm[[y]]$DGeneName==d[z])]),

sum(pt5rpm[[y]]$RPM[which(pt5rpm[[y]]$JGeneName==J[x] & pt5rpm
[[y]]$VGeneName==V[w] & pt5rpm[[y]]$DGeneName==d[z])]),

sum(pt6rpm[[y]]$RPM[which(pt6rpm[[y]]$JGeneName==J[x] & pt6rpm
[[y]]$VGeneName==V[w] & pt6rpm[[y]]$DGeneName==d[z])]),

sum(pt7rpm[[y]]$RPM[which(pt7rpm[[y]]$JGeneName==J[x] & pt7rpm
[[y]]$VGeneName==V[w] & pt7rpm[[y]]$DGeneName==d[z])]),

sum(pt8rpm[[y]]$RPM[which(pt8rpm[[y]]$JGeneName==J[x] & pt8rpm
[[y]]$VGeneName==V[w] & pt8rpm[[y]]$DGeneName==d[z])]),

sum(pt9rpm[[y]]$RPM[which(pt9rpm[[y]]$JGeneName==J[x] & pt9rpm
[[y]]$VGeneName==V[w] & pt9rpm[[y]]$DGeneName==d[z])])
}

#check to see where 0's are

Diversity.check<-matrix(rep(0,(1248*13)),ncol=1248,nrow=13)

for (i in 1:13){
    for (j in 1:1248){
if (VDJcomp[i,j]=="0"){
    Diversity.check[i,j]<-1
}
else {
    Diversity.check[i,j]<-0
}
}

# keep only VDJ combinations that 7 or more of the sample contain
NAs<-c(which(apply(Diversity.check,2,sum)==13),which(apply(Diversity.check,2,sum)==12),which(apply(Diversity.check,2,sum)==11),which(apply(Diversity.check,2,sum)==10),which(apply(Diversity.check,2,sum)==9),which(apply(Diversity.check,2,sum)==8),which(apply(Diversity.check,2,sum)==7),which(apply(Diversity.check,2,sum)==6))

# save(NAs, file="VDJNAs.RData")

VDJcomp.NA<-VDJcomp[, -NAs]

# set zeros to 1e-6
for (i in 1:13){
    for (j in 1:(1248-length(NAs))){
        if (VDJcomp.NA[i,j]=="0")
            VDJcomp.NA[i,j]<-1e-6
    }
}

200
#top 10% most variable VDJ combinations

var.top<-order(apply(VDJcomp.NA,2,var))[852:947]

#tell package we are working with compositional data

VDJcomp.NA.a<-acomp(VDJcomp.NA)
rownames(VDJcomp.NA.a)<-groupnames
VDJcomp.mean<-mean(VDJcomp.NA.a)

#run clustering on the top 10% most variable VDJ combinations

Group<-c(1,0,1,0,1,0,1,0,1,1,0,1,1)
groupnames<-factor(Group,levels=c(0,1),labels=c("No GVHD","GVHD"))

dd<-dist(VDJcomp.NA.a[,var.top])

hc = hclust(dd, method="ward.D")

png(file = 'compcluster.png', width = 1200, height = 1000, units = "px")
plot(hc,xlab="Group",ylab="Aitchison Distance",main="")
dev.off()

save.image("compositions.RData")


Function that does the composition test for the VDJ families; x is the J gene, w is V gene and z is d gene and y=1 is donor and y=2 is recipient

prob.calc<-function(x,w,z,y){

#this part calculates the number of unique CDR3 lengths
L = length(unique(c(unique(pt11rpm[[y]]$cdr3Length[which(pt11rpm[[y]]$JGeneName==J[x] & pt11rpm[[y]]$VGeneName==V[w] & pt11rpm[[y]]$DGeneName==d[z]])),
unique(pt12rpm[[y]]$cdr3Length[which(pt12rpm[[y]]$JGeneName==J[x] & pt12rpm[[y]]$VGeneName==V[w] & pt12rpm[[y]]$DGeneName==d[z]])),
unique(pt13rpm[[y]]$cdr3Length[which(pt13rpm[[y]]$JGeneName==J[x] & pt13rpm[[y]]$VGeneName==V[w] & pt13rpm[[y]]$DGeneName==d[z]])),
unique(pt14rpm[[y]]$cdr3Length[which(pt14rpm[[y]]$JGeneName==J[x] & pt14rpm[[y]]$VGeneName==V[w] & pt14rpm[[y]]$DGeneName==d[z]])),
unique(pt16rpm[[y]]$cdr3Length[which(pt16rpm[[y]]$JGeneName==J[x] & pt16rpm[[y]]$VGeneName==V[w] & pt16rpm[[y]]$DGeneName==d[z]])),
unique(pt24rpm[[y]]$cdr3Length[which(pt24rpm[[y]]$JGeneName==J[x] & pt24rpm[[y]]$VGeneName==V[w] & pt24rpm[[y]]$DGeneName==d[z]])),
unique(pt3rpm[[y]]$cdr3Length[which(pt3rpm[[y]]$JGeneName==J[x] & pt3rpm[[y]]$VGeneName==V[w] & pt3rpm[[y]]$DGeneName==d[z]])),
unique(pt4rpm[[y]]$cdr3Length[which(pt4rpm[[y]]$JGeneName==J[x] & pt4rpm[[y]]$VGeneName==V[w] & pt4rpm[[y]]$DGeneName==d[z]])),
unique(pt5rpm[[y]]$cdr3Length[which(pt5rpm[[y]]$JGeneName==J[x] & pt5rpm[[y]]$VGeneName==V[w] & pt5rpm[[y]]$DGeneName==d[z]])),
unique(pt6rpm[[y]]$cdr3Length[which(pt6rpm[[y]]$JGeneName==J[x] & pt6rpm[[y]]$VGeneName==V[w] & pt6rpm[[y]]$DGeneName==d[z]])),
unique(pt7rpm[[y]]$cdr3Length[which(pt7rpm[[y]]$JGeneName==J[x] & pt7rpm[[y]]$VGeneName==V[w] & pt7rpm[[y]]$DGeneName==d[z]])),
unique(pt8rpm[[y]]$cdr3Length[which(pt8rpm[[y]]$JGeneName==J[x] & pt8rpm[[y]]$VGeneName==V[w] & pt8rpm[[y]]$DGeneName==d[z]])),
unique(pt9rpm[[y]]$cdr3Length[which(pt9rpm[[y]]$JGeneName==J[x] & pt9rpm[[y]]$VGeneName==V[w] & pt9rpm[[y]]$DGeneName==d[z]]))))

202


97  #print(L)
98
99  #this sets the perturbation statistic to zero if the VDJ combination is not

100  if (L==0){
101    AbsoluteDeviancei<-rep(0,13)
102    AbsoluteDeviancei
103  }
104
105  else {
106    #This list the peak lengths
107    Peaks=unique(c(unique(pt11rpm[[y]]$cdr3Length[which(pt11rpm[[y]]$JGeneName==J[x] &

108      pt11rpm[[y]]$VGeneName==V[w] & pt11rpm[[y]]$DGeneName==d[z]))

109      ,

110      unique(pt12rpm[[y]]$cdr3Length[which(pt12rpm[[y]]$JGeneName==J[x] &

111        pt12rpm[[y]]$VGeneName==V[w] & pt12rpm[[y]]$DGeneName==d[z]))),

112      unique(pt13rpm[[y]]$cdr3Length[which(pt13rpm[[y]]$JGeneName==J[x] &

113        pt13rpm[[y]]$VGeneName==V[w] & pt13rpm[[y]]$DGeneName==d[z]))),

114      unique(pt14rpm[[y]]$cdr3Length[which(pt14rpm[[y]]$JGeneName==J[x] &

115        pt14rpm[[y]]$VGeneName==V[w] & pt14rpm[[y]]$DGeneName==d[z]))),

116      unique(pt16rpm[[y]]$cdr3Length[which(pt16rpm[[y]]$JGeneName==J[x]&

117        pt16rpm[[y]]$VGeneName==V[w] & pt16rpm[[y]]$DGeneName==d[z]]),

118      unique(pt24rpm[[y]]$cdr3Length[which(pt24rpm[[y]]$JGeneName==J[x]&

119        pt24rpm[[y]]$VGeneName==V[w] & pt24rpm[[y]]$DGeneName==d[z]]),

120      unique(pt3rpm[[y]]$cdr3Length[which(pt3rpm[[y]]$JGeneName==J[x] &

121        pt3rpm[[y]]$VGeneName==V[w] & pt3rpm[[y]]$DGeneName==d[z]])

203
unique(pt4rpm[[y]]$cdr3Length[which(pt4rpm[[y]]$JGeneName==J[x] &
    pt4rpm[[y]]$VGeneName==V[w] & pt4rpm[[y]]$DGeneName==d[z]))),
unique(pt5rpm[[y]]$cdr3Length[which(pt5rpm[[y]]$JGeneName==J[x] &
    pt5rpm[[y]]$VGeneName==V[w] & pt5rpm[[y]]$DGeneName==d[z]))),
unique(pt6rpm[[y]]$cdr3Length[which(pt6rpm[[y]]$JGeneName==J[x] &
    pt6rpm[[y]]$VGeneName==V[w] & pt6rpm[[y]]$DGeneName==d[z]))),
unique(pt7rpm[[y]]$cdr3Length[which(pt7rpm[[y]]$JGeneName==J[x] &
    pt7rpm[[y]]$VGeneName==V[w] & pt7rpm[[y]]$DGeneName==d[z]))),
unique(pt8rpm[[y]]$cdr3Length[which(pt8rpm[[y]]$JGeneName==J[x] &
    pt8rpm[[y]]$VGeneName==V[w] & pt8rpm[[y]]$DGeneName==d[z]))),
unique(pt9rpm[[y]]$cdr3Length[which(pt9rpm[[y]]$JGeneName==J[x] &
    pt9rpm[[y]]$VGeneName==V[w] & pt9rpm[[y]]$DGeneName==d[z]))))

# print(Peaks)

# this calculates the total rpm in each peak

Peaksi<-matrix(ncol=13,nrow=L)
for (i in 1:L){
  Peaksi[i,]<-c(sum(pt11rpm[[y]]$RPM[which(pt11rpm[[y]]$JGeneName==J[x] &
    pt11rpm[[y]]$VGeneName==V[w] & pt11rpm[[y]]$DGeneName==d[z] & pt11rpm[[y]]$cdr3Length==Peaks[i])]),
    sum(pt12rpm[[y]]$RPM[which(pt12rpm[[y]]$JGeneName==J[x] &
    pt12rpm[[y]]$VGeneName==V[w] & pt12rpm[[y]]$DGeneName==d[z] & pt12rpm[[y]]$cdr3Length==Peaks[i])]),
    sum(pt13rpm[[y]]$RPM[which(pt13rpm[[y]]$JGeneName==J[x] &
    pt13rpm[[y]]$VGeneName==V[w] & pt13rpm[[y]]$DGeneName==d[z] & pt13rpm[[y]]$cdr3Length==Peaks[i])]),
    sum(pt14rpm[[y]]$RPM[which(pt14rpm[[y]]$JGeneName==J[x] &
    pt14rpm[[y]]$VGeneName==V[w] & pt14rpm[[y]]$DGeneName==d[z] &
    pt14rpm[[y]]$ VaderName==V[w] & pt14rpm[[y]]$DGeneName==d[z])]}
\[\text{sum}(pt14rpm[[y]]\text{RPM}[\text{which}(pt14rpm[[y]]\text{JGeneName}=J[x] \&
\quad pt14rpm[[y]]\text{VGeneName}=V[w] \& pt14rpm[[y]]\text{DGeneName}=d[z] \&
\quad pt14rpm[[y]]\text{cdr3Length}=\text{Peaks}[i]])],
\]

\[\text{sum}(pt16rpm[[y]]\text{RPM}[\text{which}(pt16rpm[[y]]\text{JGeneName}=J[x] \&
\quad pt16rpm[[y]]\text{VGeneName}=V[w] \& pt16rpm[[y]]\text{DGeneName}=d[z] \&
\quad pt16rpm[[y]]\text{cdr3Length}=\text{Peaks}[i]])],
\]

\[\text{sum}(pt24rpm[[y]]\text{RPM}[\text{which}(pt24rpm[[y]]\text{JGeneName}=J[x] \&
\quad pt24rpm[[y]]\text{VGeneName}=V[w] \& pt24rpm[[y]]\text{DGeneName}=d[z] \&
\quad pt24rpm[[y]]\text{cdr3Length}=\text{Peaks}[i]])],
\]

\[\text{sum}(pt3rpm[[y]]\text{RPM}[\text{which}(pt3rpm[[y]]\text{JGeneName}=J[x] \& pt3rpm
\quad [[y]]\text{VGeneName}=V[w] \& pt3rpm[[y]]\text{DGeneName}=d[z] \& pt3rpm
\quad [[y]]\text{cdr3Length}=\text{Peaks}[i]])],
\]

\[\text{sum}(pt4rpm[[y]]\text{RPM}[\text{which}(pt4rpm[[y]]\text{JGeneName}=J[x] \& pt4rpm
\quad [[y]]\text{VGeneName}=V[w] \& pt4rpm[[y]]\text{DGeneName}=d[z] \& pt4rpm
\quad [[y]]\text{cdr3Length}=\text{Peaks}[i]])],
\]

\[\text{sum}(pt5rpm[[y]]\text{RPM}[\text{which}(pt5rpm[[y]]\text{JGeneName}=J[x] \& pt5rpm
\quad [[y]]\text{VGeneName}=V[w] \& pt5rpm[[y]]\text{DGeneName}=d[z] \& pt5rpm
\quad [[y]]\text{cdr3Length}=\text{Peaks}[i]])],
\]

\[\text{sum}(pt6rpm[[y]]\text{RPM}[\text{which}(pt6rpm[[y]]\text{JGeneName}=J[x] \& pt6rpm
\quad [[y]]\text{VGeneName}=V[w] \& pt6rpm[[y]]\text{DGeneName}=d[z] \& pt6rpm
\quad [[y]]\text{cdr3Length}=\text{Peaks}[i]])],
\]

\[\text{sum}(pt7rpm[[y]]\text{RPM}[\text{which}(pt7rpm[[y]]\text{JGeneName}=J[x] \& pt7rpm
\quad [[y]]\text{VGeneName}=V[w] \& pt7rpm[[y]]\text{DGeneName}=d[z] \& pt7rpm
\quad [[y]]\text{cdr3Length}=\text{Peaks}[i]])],
\]

\[\text{sum}(pt8rpm[[y]]\text{RPM}[\text{which}(pt8rpm[[y]]\text{JGeneName}=J[x] \& pt8rpm
\quad [[y]]\text{VGeneName}=V[w] \& pt8rpm[[y]]\text{DGeneName}=d[z] \& pt8rpm
\quad [[y]]\text{cdr3Length}=\text{Peaks}[i]])],
\]

\[\text{sum}(pt9rpm[[y]]\text{RPM}[\text{which}(pt9rpm[[y]]\text{JGeneName}=J[x] \& pt9rpm
\quad [[y]]\text{VGeneName}=V[w] \& pt9rpm[[y]]\text{DGeneName}=d[z] \& pt9rpm
\quad [[y]]\text{cdr3Length}=\text{Peaks}[i]])],
\]

205
```r
[[y]$cdr3Length==Peaks[i])]])

137  
138  #print(Peaksi)
139  
140  #total rpm per patient
141  Peakssum<--apply(Peaksi,2,sum)
142  
143  #print(Peakssum)
144  
145  #calculates the probability of each peak
146  Probabilityi<-matrix(ncol=13,nrow=L)
147  for (j in 1:L) {
148    Probabilityi[j,]<-Peaksi[j,]/Peakssum
149  }
150  
151  #replace zero values with below limit of detection value 1e-6
152  for (m in 1:L){
153    Probabilityi[m,which(Probabilityi[m,]==0)]<-1e-6
154  }
155  
156  Probabilityi
157  }
158  
159  Group<-c(1,0,1,1,0,1,0,0,1,1,0,1,1)
160  groupnames<-factor(Group,levels=c(0,1),labels=c("No GVHD","GVHD"))
161  ```
# Run linear model for each VDJ combination

comp.pval<-matrix(nrow=1,ncol=(1248-length(NAs)))

location.NA<-location[-NAs,]

for (a in 1:947){
  tryCatch({
    prob.test<-prob.calc(which(J==location.NA[a,1]),which(V==location.NA[a,2]),
                         which(d==location.NA[a,3]),2)
    prob.test.a<-acomp(prob.test)
    (model = lm(ilr(t(prob.test.a))~groupnames))
    comp.pval[a]<-anova(model)[2,6]
    print(a)},error=function(e){})
}

save(comp.pval,file="comppval.RData")

# identify significant values

fdr<-cbind(p.adjust(comp.pval, method="BH"),c(1:dim(comp.pval)[2]))

length(which(fdr[,1]<0.05))

fdr.order<-fdr[order(comp.pval),]

location.NA<-location[-NAs,]

# make table

GroupGVT<-c(2,5,7,8,11)

GroupGVHD<-c(1,3,4,6,9,10,12,13)

load("VDObjects.RData")

location<-expand.grid(J,V,d)

library(xtable)
diversitytest.table<-cbind(location.NA[fdr.order[c(1:20),2]],comp.pval[fdr.
    order[c(1:20),2]])

colnames(diversitytest.table)<-c("JGene","VGene","DGene","p-value")
xtable(diversitytest.table,digits=3)
6.3 GMIFS code

6.3.1 GMIFS and Related Functions

1 poisson.theta <- function (par, w, x, y, offset, beta) {
2   if (is.null(x)) {
3     if (!is.null(offset)) {
4       Xb <- cbind(offset, w) %*% c(1, par)
5     } else {
6       Xb <- w %*% par
7     }
8   } else {
9     if (!is.null(offset)) {
10    Xb <- cbind(offset, w, x) %*% c(1, par, beta)
11   } else {
12    Xb <- cbind(w, x) %*% c(par, beta)
13   }
14  }
15
16  contri.LL <- y * Xb - exp(Xb) - lgamma(y + 1) # likelihood function
17  loglik <- sum(contri.LL)
18  loglik
19 }

20 poisson.gmifs <- function (formula, data, x=NULL, offset, subset, epsilon = 0.001, tol=1e-5, scale=TRUE, verbose=FALSE, ...) {
21   mf <- match.call(expand.dots = FALSE)
22   cl <- match.call()
23   m <- match(c("formula", "data", "subset", "offset"), names(mf), 0L)
24   ...
mf <- mf[c(1L, m)]
mf[[1L]] <- as.name("model.frame")
mf <- eval(mf, parent.frame())
mt <- attr(mf, "terms")
y <- model.response(mf)
w <- model.matrix(mt, mf)
offset <- model.offset(mf)
if (!is.null(x)) {  
  if (missing(subset))
    r <- TRUE
  else {
    e <- substitute( subset)
    r <- eval( e, data)
    if (!is.logical(r))
      stop("'subset' must evaluate to logical")
    r <- r & !is.na(r)
  }
  if (class(x)=="character") {
    nl <- as.list( 1:ncol(data))
    names(nl) <- names( data)
    vars <- eval(substitute(x), nl, parent.frame())
    x <- data [r , vars, drop=FALSE ]
    x <- as.matrix(x )
  } else if (class(x)== "matrix" || class(x)== "data.frame") {
    x <- x[r,, drop =FALSE]
    x <- as.matrix(x)
  }
}
data <- data.matrix(data)
theta <- rep(0, dim(w)[2]) # initiate non-penalized coefficients
if (!is.null(x)) {
  vars <- dim(x)[2]
  oldx <- x
  if (scale)
    x <- scale(x, center = TRUE, scale = TRUE)
  }
  x <- cbind(x, -1 * x)
  beta <- rep(0, dim(x)[2])
  names(beta) <- dimnames(x)[[2]]
  theta.update <- matrix(theta, ncol = length(theta))
  step <- 0
  Estimates <- matrix(0, ncol=vars)
  Likelihood <- numeric()
  AIC <- numeric()
  # initialize theta and (Intercept)
  step <- 0
  # initialize <- nlm(poisson.theta, p=theta, w=w, x=x, y=y, offset=offset,
                  beta=beta)
  # theta <- initialize$estimate
  initialize <- glm(y ~ w - 1, offset=offset, family=poisson)
  theta <- coef(initialize)
  theta.update <- matrix(theta, ncol = length(theta))
  repeat {
    if (!is.null(offset)) {
      
    }}
Xb <- cbind(offset, w, x) %*% c(1, theta, beta)

} else {
Xb <- cbind(w, x) %*% c(theta, beta)
}

u <- t(x)%*%(y-exp(Xb)) #likelihood gradient value

update.j <- which.min(-u) #choose coefficient to update
if (-u[update.j] < 0) {
}
Estimates<-rbind(Estimates,beta[1:vars]-beta[(vars+1):length(beta)]) #keep track of beta changes

out <- optim(theta, poisson.theta, w=w, x=x, y=y, offset=offset, beta=beta,method="BFGS") #update intercept and non-penalized subset using new beta values
theta <- out$par
theta.update <- rbind(theta.update, theta) #keep track of new theta values

p <- sum(Estimates[step+2,!]=0) + length(theta)
Likelihood[step+1]<- LL1<- -out$value
if (verbose) cat("step = ", step, "\n")
if (step >= 1 && LL1 - LL0 < tol) { #decide when to stop repeat
  break
}
LL0 <- LL1
step <- 1 + step

beta.final <- Estimates[-1,]  # final values
theta<-theta.update[-1,]
if (dim(w)[2]==grep("(Intercept)",colnames(w))) {
  names(theta)<-colnames(w)
} else {
  colnames(theta)<-colnames(w)
}
model.select<-which.min(AIC)
output<-list(beta = beta.final, theta=theta, x=oldx, y=y, scale=scale,
              Likelihood=Likelihood, AIC=AIC, model.select=model.select,w=w,
              offset=offset)

} else {
# out <- nlm(poisson.theta, p=theta, w=w, x=NULL, y=y, offset=offset, beta=NULL)
# output<- out$estimate
out<-glm(y~w-1, offset=offset, family=poisson)
output <- coef(out)
}

class(output) <- "poisson.gmifs"
output
predict.poisson.gmifs <- function(fit, newx=NULL, newoffset=NULL, model.select=NA) {
  y<-fit$y

  if (is.null(newx)) {
    x<-fit$x
  } else { x<-newx
  }

  w<-fit$w

  if (is.null(newoffset)){
    offset<-fit$offset
  } else {offset<-newoffset
  }

  if (is.numeric(model.select)) {
    model.select = model.select
  } else if (is.na(model.select)) {
    model.select=dim(fit$beta)[1]
  }
else if (model.select=="AIC") {
    model.select = fit$model.select
}

if (is.null(dim(fit$theta))) {
    beta<-fit$beta[model.select,]
    theta<-fit$theta[model.select]
    offset<-fit$offset

    if (is.null(offset)) {
        y.pred <- exp(c(theta,beta) %*% t(cbind(w, x)))
    }
    else {
        offset<-fit$offset
        y.pred <- exp(c(1,theta,beta) %*% t(cbind(offset, w, x)))
    }
}

else if (is.numeric(fit$theta[,-1])){
    beta<-fit$beta[model.select,]
    theta<-fit$theta[model.select,]
    offset<-fit$offset

    if (is.null(offset)) {
        y.pred <- exp(c(theta,beta) %*% t(cbind(w, x)))
    }
}
else {
    y.pred <- exp(c(1,theta,beta) %*% t(cbind(offset, w, x)))
}

output<-list(pred=y.pred,theta=theta,beta=beta,offset=offset,w=w,x=x)
output

##############################
#Coefficient function######
##############################

coef.poisson.gmifs<- function(fit, model.select=NA) {

if (is.numeric(model.select)) {

  if (is.null(dim(fit$theta))) {
    beta<-fit$beta[,model.select]
    theta<-fit$theta[,model.select]
    c.coef<-cbind(theta,beta)
    colnames(c.coef)<- c("intercept",colnames(fit$w)[-1],colnames(fit$x))
    rownames(c.coef)<-as.character(1:dim(beta)[1])
  } else {
    beta<-fit$beta[,model.select]
    theta<-fit$theta[,model.select]
  }
} else {
  beta<-fit$beta[,model.select]
  theta<-fit$theta[,model.select]
}
c.coef<-cbind(theta,beta)
colnames(c.coef)<- c("intercept",colnames(fit$w)[-1],colnames(fit$x))
rownames(c.coef)<-as.character(1:dim(beta)[1])
}
}

else if (is.na(model.select)) {
  model.select=dim(fit$beta)[1]
  if (is.null(dim(fit$theta))) {
    beta<-fit$beta[model.select,]
    theta<-fit$theta[model.select]
    c.coef<-c(theta,beta)
  } else {
    beta<-fit$beta[model.select,]
    theta<-fit$theta[model.select]
    c.coef<-c(theta,beta)
  }
}
else if (model.select == "AIC") {
  model.select = fit$model.select
  if (is.null(dim(fit$theta))) {
    beta<-fit$beta[model.select,]
theta<-fit$theta[model.select]
c.coef<-c(theta,beta)

else {
beta<-fit$beta[model.select,]
theta<-fit$theta[model.select,]
c.coef<-c(theta,beta)
}

else if (model.select == "all") {

beta<-fit$beta
theta<-fit$theta
c.coef<-cbind(theta,beta)
colnames(c.coef)<- c("intercept",colnames(fit$w)[-1],colnames(fit$x))
rownames(c.coef)<-as.character(1:dim(beta)[1])
}

output<-list(coef=c.coef)
output
summary.poisson.gmifs<- function(fit, model.select=NA) {
  #browser()
  if (is.na(model.select)) {
    model.select=dim(fit$beta)[1]
    Likelihood<-fit$Likelihood[model.select]
    AIC<-fit$AIC[model.select]
    summary<-c(Likelihood,AIC)
    names(summary)<- c("Likelihood","AIC")
  }
  else if (model.select == "AIC") {
    model.select = fit$model.select
    Likelihood<-fit$Likelihood[model.select]
    AIC<-fit$AIC[model.select]
    summary<-c(Likelihood,AIC)
    names(summary)<- c("Likelihood","AIC")
  }
  else if (model.select=="all") {
    Likelihood<-fit$Likelihood
    AIC<-fit$AIC
    summary<-cbind(Likelihood,AIC)
    colnames(summary)<- c("Likelihood","AIC")
  }
}
else {
  Likelihood<-fit$Likelihood[model.select]
  AIC<-fit$AIC[model.select]
  summary<-cbind(Likelihood,AIC)
  colnames(summary)<- c("Likelihood","AIC")
}

output<-list(summary=summary)
output
}

#############################
#plot function#############
#############################

plot.poisson.gmifs<- function(fit, type,main=type,beta="All") {
  #browser()
  if (type=="coefficients") {
    if (beta=="All"){
      n<-which(fit$beta[dim(fit$beta)[1],] != 0)
      plot(1:dim(fit$beta)[1],fit$beta[,n[1]],xlab="Step",ylab="Beta",main=main,
           col=500+n[1],type="l",ylim=c(min(fit$beta[dim(fit$beta)[1],]),max(fit$beta[dim(fit$beta)[1],]))
      for (i in 2:length(n)) {
        lines(fit$beta[,n[i]],col=500+n[i])
      }
    }
  }
}
else {
    n <- beta
    plot(1:dim(fit$beta)[1], fit$beta[, n[1]], xlab="Step", ylab="Beta", main=main,
         col=500+n[1], type="l", ylim=c(min(fit$beta[dim(fit$beta)[1],]), max(fit$beta[dim(fit$beta)[1],])))
    for (i in 2:length(n)){
        lines(fit$beta[, n[i]], col=500+n[i])
    }
}

else if (type == "AIC") {
    plot(1:length(fit$AIC), fit$AIC, xlab="Step", ylab="AIC", main=main)
}

else { type = "Likelihood"
    plot(1:length(fit$Likelihood), fit$Likelihood, xlab="Step", ylab="-
         logLikelihood", main=main)
}

}
6.3.2 GMIFS Simulations

1 # No offset simulations

4 #create matrix to save results in
5 sims<-matrix(nrow=100,ncol=27)
6 colnames(sims)<-c("no.correct.gmifs","beta1.gmifs","beta2.gmifs","beta3.
    gmifs","beta4.gmifs","beta5.gmifs","no.incorrect.gmifs","sum.incorrect.
    gmifs","res.sqr.gmifs","no.correct.path","beta1.path","beta2.path","beta3.
    path","beta4.path","beta5.path","no.incorrect.path","sum.incorrect
    .path","res.sqr.path","no.correct.path.norm","beta1.path.norm","beta2.
    path.norm","beta3.path.norm","beta4.path.norm","beta5.path.norm","no.
    incorrect.path.norm","sum.incorrect.path.norm","res.sqr.path.norm")
7 y.sims<-matrix(nrow=30,ncol=100)
8 library(glmpath) #load glmpath library
9 #loop 100 times or some other number
10
11 for (i in 1:100) {
12    #number of samples
13    n<-30
14
15    #generate related coefficients - these can be changed
16    beta0<-1
17    beta1<-0.3
18    beta2<-0.2
19    beta3<- -0.7
20    beta4<-0.5
beta5<-0.1
beta<-cbind(beta1,beta2,beta3,beta4,beta5)

#generate covariates related to response
set.seed(i*23) #set seed to replicate sims
x1<-runif(n=n, min=0, max=1)
x2<-runif(n=n, min=0, max=1)
x3<-runif(n=n, min=0, max=1)
x4<-runif(n=n, min=0, max=1)
x5<-runif(n=n, min=0, max=1)

#generate lambda value using the five covariates related to response
mu<-exp(beta0 + (beta1*x1) + (beta2*x2) + (beta3*x3) + (beta4*x4) + (beta5 *x5))

#generate poisson variable using mu
y<-rpois(n=n, lambda=mu)
y.sims[,i]<-y

#create data frame
x<-cbind(x1,x2,x3,x4,x5)
data<-data.frame(y=y,x=x)

#create 95 covariates not related to response for a total of 100
x.unrelated<-matrix(nrow=n, ncol=95)
for (j in 1:95) {
  x.unrelated[,j]<-runif(n=n,min=0,max=1)
}

#final data frame
data<-data.frame(data,x.unrelated)

#############################
#Create test data set
#############################

#generate covariates related to response
#set seed to replicate sims
x1.test<-runif(n=n, min=0, max=1)
x2.test<-runif(n=n, min=0, max=1)
x3.test<-runif(n=n, min=0, max=1)
x4.test<-runif(n=n, min=0, max=1)
x5.test<-runif(n=n, min=0, max=1)

#generate lambda value using the five covariates related to response
mu.test<-exp(beta0 + (beta1*x1.test) + (beta2*x2.test) + (beta3*x3.test) +
(beta4*x4.test) + (beta5*x5.test))

#generate poisson variable using mu
y.test<-rpois(n=n, lambda=mu.test)

#create data frame
x.test<-cbind(x1.test,x2.test,x3.test,x4.test,x5.test)
data.test<-data.frame(y.test=y.test, x.test=x.test)

#create 95 covariates not related to response for a total of 100
x.unrelated.test<-matrix(nrow=n, ncol=95)
for (j in 1:95) {
x.unrelated.test[,j]<-runif(n=n, min=0, max=1)
}

#final data frame
data.test<-data.frame(data.test, x.unrelated.test)

#run gmifs
fit<-poisson.gmifs(y~1, x=data[, -1], data=data)
pred<-exp(c(fit$theta[fit$model.select],fit$beta[fit$model.select,])) %*% t(cbind(fit$w,data.test[,1:-1])) #get predicted values of test data

fit.path<-glm.path(x=as.matrix(data[,-1]),y=as.matrix(data[,1]),family=poisson,lambda2=0)
pred.path<- predict(fit.path, newx=as.matrix(data.test[,-1]), type="response") #get response values using test data
coeff.path<- predict(fit.path, newx=as.matrix(data[,-1]), type="coefficients"

fit.path.norm<- glm.path(x=as.matrix(data[,-1]),y=as.matrix(data[,1]),family=gaussian)
pred.path.norm<- predict(fit.path.norm, newx=as.matrix(data.test[,-1]), type ="response") #get response values using test data
coeff.path.norm<- predict(fit.path.norm, newx=as.matrix(data[,-1]), type="coefficients"

#different results to save from sims...

#keeping beta values, number nonzero that should be and number nonzero that should not be

no.correct<- sum(fit$beta[fit$model.select,1:5] != 0)
sims[i,1]<-no.correct
no.incorrect<- sum(fit$beta[fit$model.select,-(1:5)] != 0)
sims[i,7]<-no.incorrect
sims[i,8]<- sum(abs(fit$beta[fit$model.select,which(fit$beta[fit$model.select,,-(1:5)] != 0)+5]))
sims[i,9]<-sum((data.test[,1] - pred)^2)
no.correct.path <- sum(coef.path[which.min(summary(fit.path)$AIC),2:6] != 0)
sims[i,10] <- no.correct.path
no.incorrect.path <- sum(coef.path[which.min(summary(fit.path)$AIC),-(1:6)] != 0)
sims[i,16] <- no.incorrect.path
sims[i,17] <- sum(abs(coef.path[which.min(summary(fit.path)$AIC),which(coef.path[which.min(summary(fit.path)$AIC),-(1:6)] != 0)+6]))
sims[i,18] <- sum((data.test[,1] - pred.path[,which.min(summary(fit.path)$AIC)]).^2)

no.correct.path.norm <- sum(coef.path.norm[which.min(summary(fit.path.norm)$AIC),2:6] != 0)
sims[i,19] <- no.correct.path.norm
no.incorrect.path.norm <- sum(coef.path.norm[which.min(summary(fit.path.norm)$AIC),-(1:6)] != 0)
sims[i,25] <- no.incorrect.path.norm
sims[i,26] <- sum(abs(coef.path.norm[which.min(summary(fit.path.norm)$AIC),which(coef.path.norm[which.min(summary(fit.path.norm)$AIC),-(1:6)] != 0)+6]))
sims[i,27] <- sum((data.test[,1] - pred.path.norm[,which.min(summary(fit.path.norm)$AIC)]).^2)
print(i)
}


save(sims,y.sims,file="simsnoffset30independent.RData")

####################################
#Sims using an offset#
####################################
sims<-matrix(nrow=100,ncol=27)
colnames(sims)<-c("no.correct.gmis","beta1.gmis","beta2.gmis","beta3.gmis","beta4.gmis","beta5.gmis","no.incorrect.gmis","sum.incorrect.gmis","res.sqr.gmis","no.correct.path","beta1.path","beta2.path","beta3.path","beta4.path","beta5.path","no.incorrect.path","sum.incorrect.path","res.sqr.path","no.correct.path.norm","beta1.path.norm","beta2.path.norm","beta3.path.norm","beta4.path.norm","beta5.path.norm","no.incorrect.path.norm","sum.incorrect.path.norm","res.sqr.path.norm")
y.sims<-matrix(nrow=30,ncol=100)
library(glmpath) #load glmpath library

#loop 100 times or some other number
for (i in 1:100) {
  #number of samples
  n<-30
  set.seed(i*34)
  #generate related coefficients - these can be changed
  beta0<-7
  beta1<-0.3
  beta2<-0.2
  beta3<- -0.7
\begin{verbatim}
157 beta4<-0.5
158 beta5<-0.1
159 beta<-cbind(beta1,beta2,beta3,beta4,beta5)

161 # generate covariates related to response
162 x1<-runif(n=n, min=0, max=1)
163 x2<-runif(n=n, min=0, max=1)
164 x3<-runif(n=n, min=0, max=1)
165 x4<-runif(n=n, min=0, max=1)
166 x5<-runif(n=n, min=0, max=1)

168 # Generate offset
169 offset<- round(2000 + runif(n=n,min=-200,max=200))

171 # generate lambda value using the five covariates related to response
172 mu<-exp(beta0 + log(offset) + (beta1*x1) + (beta2*x2) + (beta3*x3) + (beta4*
         x4) + (beta5*x5))

175 # generate poisson variable using mu
176 y<-rpois(n=n, lambda=mu)
177 y.sims[,i]<-y

180 # create data frame
181 x<-cbind(x1,x2,x3,x4,x5)
182 data<-data.frame(y=y,x=x)
\end{verbatim}
#create 95 covariates not related to response for a total of 100

x.unrelated <- matrix(nrow = n, ncol = 95)

for (j in 1:95) {
  x.unrelated[, j] <- runif(n = n, min = 0, max = 1)
}

#final data frame

data <- data.frame(data, x.unrelated, offset = offset)

#Generate test data set

#generate covariates related to response

x1.test <- runif(n = n, min = 0, max = 1)
x2.test <- runif(n = n, min = 0, max = 1)
x3.test <- runif(n = n, min = 0, max = 1)
x4.test <- runif(n = n, min = 0, max = 1)
x5.test <- runif(n = n, min = 0, max = 1)

#Generate offset
offset.test<- round(2000 + runif(n=n,min=-200,max=200))

#generate lambda value using the five covariates related to response
mu.test<-exp(beta0 + log(offset.test) + (beta1*x1.test) + (beta2*x2.test) +
(beta3*x3.test) + (beta4*x4.test) + (beta5*x5.test))

#generate poisson variable using mu
y.test<-rpois(n=n, lambda=mu.test)

#create data frame
x.test<-cbind(x1.test,x2.test,x3.test,x4.test,x5.test)
data.test<-data.frame(y.test=y.test,x.test=x.test)

#create 95 covariates not related to response for a total of 100
x.unrelated.test<-matrix(nrow=n, ncol=95)
for (j in 1:95) {
x.unrelated.test[,j]<-runif(n=n,min=0,max=1)
}

#final test data frame
data.test<-data.frame(data.test,x.unrelated.test,offset.test=offset.test)

#test gmifs
fit<-poisson.gmifs(y~1, x=data[,,-c(1,102)], data=data, offset=log(offset))

offset.test<-data.test[,102]

pred<-exp(c(fit$theta[fit$model.select],1,fit$beta[fit$model.select,])) %*% t(cbind(fit$w,log(offset.test),data.test[,,-c(1,102)]))

offset<-fit$offset

#run glmpath

fit.path<-glmpath(x=as.matrix(data[,,-c(1,102)]),y=as.matrix(data[,1]),offset=offset,family=poisson,lambda2=0)
pred.path<- predict(fit.path, newx=as.matrix(data.test[,,-c(1,102)]),offset=offset.test, type="response")

coef.path<- predict(fit.path, newx=as.matrix(data[,,-c(1,102)]),offset=offset, type="coefficients")

fit.path.norm<- glmpath(x=as.matrix(data[,,-c(1,102)]),y=as.matrix(data[,1]), family=gaussian,offset=offset)
pred.path.norm<- predict(fit.path.norm, newx=as.matrix(data.test[,,-c(1,102)]),offset=offset.test, type="response")

coef.path.norm<- predict(fit.path.norm, newx=as.matrix(data[,,-c(1,102)]), offset=offset, type="coefficients")

#different results to save from sims...

#keeping beta values, number nonzero that should be and number nonzero that should not be

no.correct<- sum(fit$beta[fit$model.select,1:5] != 0)
sims[i,1]<-no.correct


no.incorrect<- sum(fit$beta[fit$model.select,-(1:5)] != 0)
sims[i,7]<-no.incorrect
sims[i,8]<- sum(abs(fit$beta[fit$model.select,which(fit$beta[fit$model.
    select,-(1:5)] != 0)+5]))
sims[i,9]<-sum((data[,1] - pred)^2)
####
no.correct.path<- sum(coef.path[which.min(summary(fit.path)$AIC),2:6] != 0)
sims[i,10]<-no.correct.path
no.incorrect.path<-sum(coef.path[which.min(summary(fit.path)$AIC),-(1:6)] !=
    0)
sims[i,16]<-no.incorrect.path
sims[i,17]<- sum(abs(coef.path[which.min(summary(fit.path)$AIC),which(coef.
    path[which.min(summary(fit.path)$AIC),-(1:6)] != 0)+6]))
sims[i,18]<-sum((data.test[,1] - pred.path[,which.min(summary(fit.path)$AIC)
    ])^2)
####
no.correct.path.norm<- sum(coef.path.norm[which.min(summary(fit.path.norm)$
    AIC),2:6] != 0)
sims[i,19]<-no.correct.path.norm
no.incorrect.path.norm<-sum(coef.path.norm[which.min(summary(fit.path.norm)$
    AIC),-(1:6)] != 0)
sims[i,25]<-no.incorrect.path.norm
sims[i,26]<- sum(abs(coef.path.norm[which.min(summary(fit.path.norm)$AIC),
    which(coef.path.norm[which.min(summary(fit.path.norm)$AIC),-(1:6)] != 0)
    +6]))
sims[i,27]<-sum((data.test[,1] - pred.path.norm[,which.min(summary(fit.path.
        norm)$AIC)])^2)

print(i)

}

save.image("simsoffset30alphaindependent.RData")
6.3.3 Gene Expression Code

# Set up Gene Expression Data

```
library("GEOquery")
data<-getGEO("GSE31836")[[1]]
setwd("C:/Users/makowski/Desktop/Dissertation/TCRdata/micronuclei/codes")
mn.data<-read.csv("MNdataCEBPpaper.csv")
pData(data)$Array<-substr(pData(data)$supplementary_file,93,98)
mn.data$ID<-paste(mn.data$Array,mn.data$Lane,sep="")
names(mn.data)[2]<"Chip"
pheno.frame<-merge(pData(data),mn.data,by.x="Array",by.y="ID",all.x=TRUE,all.y=TRUE)
pheno.sorted<-pheno.frame[match(pData(data)$geo_accession,pheno.frame$geo_accession),]
pData(data)$C_NMNCB_VUB<-pheno.sorted$C_NMNCB_VUB
pData(data)$Chip<-pheno.sorted$Chip
pData(data)$Lane<-pheno.sorted$Lane
mn.exprs.data<-data[,!is.na(pData(data)$C_NMNCB_VUB)]
hist(pData(data)$C_NMNCB_VUB)
pData(data)$characteristics_ch2.1 # gender
nas<-apply(exprs(mn.exprs.data),1,function(x) sum(is.na(x)))
mn.exprs<-mn.exprs.data[nas==0,]
mn.gender<-pData(data)$characteristics_ch2.1[which(pData(data)$C_NMNCB_VUB!="NA")]
all.equal(as.character(pData(mn.exprs)$geo_accession),sampleNames(mn.exprs))
```
# Fit model where pData(mn.exprs)$C_NMNCB_VUB ~ pData(data)$characteristics_ch2.1(not penalized) + expression (penalized)

gexpr<-t(exprs(mn.exprs))

mn.gender.num<-rep(0,29)
mn.gender.num[grep("female",mn.gender)]=1 # male=0 female=1

pheno<-as.data.frame(cbind(pData(mn.exprs)$C_NMNCB_VUB,mn.gender.num))
colnames(pheno)<-c("MN","gender")

GErun<-poisson.gmifs(MN~gender, data=pheno, x=gexpr, epsilon=0.001, tol=1e-5, scale=TRUE, verbose=TRUE) # GMIFS run

pred<-exp(c(GErun$theta[GErun$model.select,],GErun$beta[GErun$model.select,]) %*% t(cbind(GErun$w,scale(GErun$x,center=TRUE,scale=TRUE)))) # get the predicted values from GMIFS at AIC value

sum(coef.poisson.gmifs(GErun,model.select="AIC")$coef != 0) # how many are non-zero using function

summary.poisson.gmifs(GErun) # test of function

plot.poisson.gmifs(GErun,type="coefficients") # test of plot function

save(GErun,file="GErun.RData")
library(glmpath)
x< gexpr
2 y< pheno[,1]
x.nopenalty< cbind(pheno[,2],x)

fit.path< glmpath(x=as.matrix(x.nopenalty),y=y,nopenalty.subset=c(1),family= poisson,lambda2=0) # glmpath run

save.image("GErun.RData")

coef.path< predict(fit.path, newx=as.matrix(x.nopenalty), type="
coefficient") # get the coefficients from glmpath

hist(coef.path[which.min(summary(fit.path)$AIC),]) # look at histogram

sum(coef.path[which.min(summary(fit.path)$AIC),] != 0) # non-zero at AIC value

sum(coef.path[dim(coef.path)[1],] != 0) # non-zero at full model

# Comparison

glmcoef<-which(coef.path[which.min(summary(fit.path)$AIC),] != 0) # 23 non-zero coefficients
poisscoef <- which(coef.poisson.gmifs(GErun, model.select="AIC")$coef != 0) # 17 non-zero coefficients

length(intersect(glmcoef[-c(1,2)], poisscoef[-c(1,2)])) # 10 of them are in common

glmpathnon <- names(coef.path[which.min(summary(fit.path)$AIC),] [glmcoef[-c(1,2)]])) # get the names of the glmpath

gmifsnon <- names(coef.poisson.gmifs(GErun, model.select="AIC")$coef[poisscoef[-c(1,2)]]])

common <- intersect(names(coef.path[which.min(summary(fit.path)$AIC),] [glmcoef[-c(1,2)]]), names(coef.poisson.gmifs(GErun, model.select="AIC")$coef[poisscoef[-c(1,2)]]]))

union(names(coef.path[which.min(summary(fit.path)$AIC),] [glmcoef[-c(1,2)]]), names(coef.poisson.gmifs(GErun, model.select="AIC")$coef[poisscoef[-c(1,2)]]))

outersect <- function(x, y, ...) {
  big.vec <- c(x, y, ...)
  duplicates <- big.vec[duplicated(big.vec)]
  setdiff(big.vec, unique(duplicates))
}

gmifsonly <- outersect(gmifsnon, common)
glmpathtonly <- outersect(glmpathnon, common)
pred.path <- predict(fit.path, newx = as.matrix(x.nopenalty), type = "response")

pred.path[, which.min(summary(fit.path)$AIC)]

png(file = "glmpathplot.png", width = 1080, height = 768, units = "px")
par(mar = c(10, 10, 1, 1))
plot(y, pred.path[, which.min(summary(fit.path)$AIC)], xlab = "", ylab = "", cex.axis = 1.5, xlim = c(0, 25))
myylab = "Predicted MN count"
myxlab = "Actual MN count"
mtext(myylab, side = 2, line = 7, cex = 3)
mtext(myxlab, side = 1, line = 7, cex = 3)
abline(lm(pred.path[, which.min(summary(fit.path)$AIC)] ~ y)$coef)
dev.off()

png(file = "gmifsplot.png", width = 1080, height = 768, units = "px")
par(mar = c(10, 10, 1, 1))
plot(y, pred, xlab = "", ylab = "", cex.axis = 1.5, xlim = c(0, 25))
myylab = "Predicted MN count"
myxlab = "Actual MN count"
mtext(myylab, side = 2, line = 7, cex = 3)
mtext(myxlab, side = 1, line = 7, cex = 3)
114 abline(lm(pred[1,]~y)$coef)
115 dev.off()
116
117 png(file = "GEMNhist.png", width = 1200, height = 768, units = "px")
118 par(mar = c(8,7,4,2) + 0.1)
119 hist(y,main="",xlab="MN Frequency",ylab="Number of Subjects",cex.lab=2,cex.
   main=2,cex.axis=2)
120 dev.off()
121
122 SSR.glmpath=sum((y-pred.path[,which.min(summary(fit.path)$AIC)])^2)
123 SSR.GMIFS=sum((y-pred[1,])^2)
6.3.4 Buds Code

```
setwd("C:/Users/makowski/Desktop/Dissertation/TCRdata/micronuclei")
load("BreastCa.RData")

age<-phenoData$age
smoking.status<-phenoData$CurrentlySmoking

MNfreq<-apply(phenoData[,grep("mn_Bi_w_mn",names(phenoData))], 1, sum) + apply(phenoData[,grep("mn_MN",names(phenoData))], 1, sum)
BNcells<-apply(phenoData[,grep("mn_Binucleate",names(phenoData))], 1, sum) + MNfreq

head(phenoData[,grep("mn_Mononucleate_w_MN", names(phenoData))])
Mono.cells<-apply(phenoData[,grep("mn_Mononucleate_w_MN",names(phenoData))], 1, sum)

Tri.cells<-apply(phenoData[,grep("mn_Trinucleate",names(phenoData))], 1, sum)

Total.cells<-BNcells+Mono.cells+Tri.cells

Bud.cells<-apply(phenoData[,grep("mn_Buds",names(phenoData))], 1, sum)
Bridge.cells<-apply(phenoData[,grep("mn_Bridges",names(phenoData))], 1, sum)

Total.cells<-BNcells+Mono.cells+Tri.cells+Bud.cells+Bridge.cells

#check distribution of Bud and Bridge counts
#both look skewed but Bridge.cells has 54/73 0 values whereas Bud.cells only has 16/73 0 values...run analysis on Buds

hist(Bud.cells)
hist(Bridge.cells)

NAs<-which(is.na(Bud.cells))
```
#filter out CpG sites that are all above 80% or below 20%
x.eighty<-matrix(nrow=dim(beta.corrected)[1],ncol=1)
for (i in 1:dim(beta.corrected)[1]){
x.eighty[i,1]<-sum(beta.corrected[i,] > 0.8)
}
x.eighty.pos<-which(x.eighty[,1]==73)

x.twenty<-matrix(nrow=dim(beta.corrected)[1],ncol=1)
for (i in 1:dim(beta.corrected)[1]){
x.twenty[i,1]<-sum(beta.corrected[i,] < 0.2)
}
x.twenty.pos<-which(x.twenty[,1]==73)

x.twenty.eighty.pos<-c(x.twenty.pos,x.eighty.pos)
methyl.filter<-t(beta.corrected[-(x.twenty.eighty.pos),])

#Remove observations with NA's... 70 observations now
Bud.cells<-Bud.cells[-NAs]
age<-age[-NAs]
smoking.status<-smoking.status[-NAs]
BNcells<-BNcells[-NAs]
methyl.filter<-methyl.filter[-NAs,]
## Identify CpG sites associated with Buds for (1) general conclusions and (2) use for filtering in multivariable modeling step

```r
pvalue<-numeric()
for (i in 1:dim(methyl.filter)[2]) {
  fit<-glm(Bud.cells ~ age + smoking.status + methyl.filter[,i] +
          offset(log(BNcells)), family=poisson)
  summary.fit<-summary(fit)
  pvalue[i]<-summary.fit$coefficients[4, "Pr>|z|"]
  print(i)
}
pvalue<-cbind(pvalue,colnames(methyl.filter))
colnames(pvalue)<-c("p-value","CpG")

## Use a generous p-value threshold to filter CpG sites are associated with Buds
pval<-(as.numeric(pvalue[,1]))
sum(pval<0.05) #leaves 10860 CpG sites;
sum(pval<0.01) #would reduce to 1728

methyl.filter.pval<-methyl.filter[,which(pval < .05)]
```

---

243
smoke <- rep(1, 70)
smoke[which(smoking.status == "No")] <- 0

pheno <- data.frame(age = age, smoke = smoke, binucleate_total = BNcells, binucleate_Buds = Bud.cells)

# save image that has everything needed to do the test run including GMIFS function
save.image("filterBudsrun.RData")

# GMIFS run using Buds
system.time(Budsrun <- poisson.gmifs(binucleate_Buds ~ age + smoke, data = pheno,
  x = methyl.filter.pval, offset = log(pheno$binucleate_total), epsilon = 0.001,
  tol = 1e-5, scale = TRUE, verbose = TRUE))

# user system elapsed
# 12858.81 1689.74 14582.69

pred <- exp(c(Budsrun$theta[Budsrun$model.select,], 1, Budsrun$beta[Budsrun$model.select,]) %*% t(cbind(Budsrun$w, Budsrun$offset, scale(Budsrun$x, center = TRUE, scale = TRUE))))  # get the predicted values from GMIFS at AIC value

sum(coef.poisson.gmifs(Budsrun, model.select = "AIC")$coef != 0)  # how many are non-zero using function

coef.poisson.gmifs(Budsrun, model.select = "AIC")$coef[which(coef.poisson.gmifs(Budsrun, model.select = "AIC")$coef != 0)]
#plots

gmifsplotbuds.png, width = 1080, height = 768, units = "px"

par(mar = c(10,10,1,1))

plot(y,pred,xlab="",ylab="",cex.axis = 1.5,xlim=c(0,7))

Predicted Bud count

Actual Bud count

mtext("Predicted Bud count", side = 2, line = 7, cex = 3)

mtext("Actual Bud count", side = 1, line = 7, cex = 3)

abline(lm(pred[1,"y")$coef)

dev.off()

Budhist.png, width = 1200, height = 768, units = "px"

par(mar = c(8,7,4,2) + 0.1)

hist(y,main="",xlab="Buds Frequency",ylab="Number of Subjects",cex.lab=2,cex .main=2,cex.axis=2)

dev.off()
Vita

Department of Biostatistics  
Virginia Commonwealth University  
Richmond, VA 23298  
Phone: (717) 371-4439  
Email: makowskims@vcu.edu  
Date of Birth: April, 20, 1985  
Citizenship: United States of America  
Email: makowski85@gmail.com

Education

• August 2015 Ph.D., Biostatistics, Virginia Commonwealth University (expected)

• May 2007 B.S., Biology and Mathematics, Dickinson College

Academic Professional Experience

• Statistical Consultant, Technology Services, VCU 2010-2014

• Statistical Consultant, Technology Services, VCU 2010-2014

• Research Assistant, Dr. Kellie J. Archer Department of Biostatistics, VCU 2012-2015

• Teaching Assistant, Department of Biostatistics, VCU 2009-2014
  
  – Courses: ANOVA, Biostatistical Computing, Mathematical Statistics, Categorical Analysis, Biostatistical Methods

• Research Assistant, Dr. Mark Reimers, Department of Biostatistics, VCU 2010-2012
• Fellow, Biotherapy Section, Laboratory of Molecular Biology, National Cancer Institute, NIH 2007-2009

• Summer Intern, Biotherapy Section, Laboratory of Molecular Biology, National Cancer Institute, NIH 2006

• Tutor, Department of Chemistry, Dickinson College 2004
  – Course: General Chemistry

Honors and Awards

• 2007 Intramural Research Training Award, National Institutes of Health

• 2003 John Dickinson Scholar, Dickinson College

Presentations


Publications


• Riddle D, Kong X, Makowski M "Knee Osteoarthritis Worsening Across the Disease Spectrum and Future Knee Pain, Symptoms and Functioning: A Multisite Prospective Cohort Study." Arthritis Care and Research (2015) accepted