Immunology Workshop

Aly A. Khan

8/10/2020

Rmd Supplement

This is the code supplement to the Immunology Workshop lecture PDF in a R Markdown format.

Exercise 1.1 — Data wrangling Let's begin by downloading some scRNA-seq data. For this exercise we will be using data published as part of a study examining certain B cells in humans after influenza vaccination (Neu et al., JCI, 2019). We will use pre-processed supplementary data from the GEO database: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE116500. Scroll to the bottom of the page to locate the supplementary data. Alternatively, you can use the file in the data folder on github.

```
# load data
geo <- read.csv("GSE116500_Limma_adj_4.csv.gz", row.names = 1, header = TRUE)
```

How many genes and cells are in the data? Let's try looking at the dimensions of the data:

```
# number of rows (Genes) and columns (cells)
dim(geo)
```

```
## [1] 11895   295
```

```
num_genes <- dim(geo)[1]
num_cells <- dim(geo)[2]
```

To made things easier for subsequent steps, let's take the transpose of the data so that each row denotes a cell and each column denotes a gene. The transpose of a matrix is an operation which flips a matrix over its diagonal; that is, it switches the row and column indices. This can facilitate certain types of linear algebraic operations and calculations which operate on columns by default.

```
# current data with genes by cells
dim(geo)
```

```
## [1] 11895   295
```

```
# transpose data to cells by genes
geo <- t(geo)
dim(geo)
```

```
## [1]   295 11895
```

Our assumption is that most gene expression in scRNA-seq data contains random noise due to technical variation. We aim to focus our analyses on genes with high variability, which may have a biological basis. For the purposes of this exercise we simply calculate variance of gene expression as a way to rank genes, but we note other methods (e.g. coefficient of variation) can be used as well.

```
# Let's identify highly variable genes based on variance.
SDs <- apply(geo, 2, var)
```

```r
# we usually pick the top 1000 - 2000 highly variable genes (HVGs)
hvgs <- names(sort(SDs, decreasing = TRUE))[1:2000]
geo.hvg <- as.data.frame(geo[, hvgs])
dim(geo.hvg)
```

```
## [1]  295 2000
```

We should now have a matrix of 295 cells with the top 2000 most highly variable genes. One fact that we did not reveal earlier is that these cells are plasmablasts, which are B cells that secrete antibodies.

```r
# There are multiple genes that encode subclasses of the two isotypes in this
# data: IgA and IgG

# Grab max IgA values across all genes
IgA <- cbind(geo.hvg$IGHA1, geo.hvg$IGHA2)
IgA_max <- as.matrix(apply(IgA, 1, max))
# Set column name to 'IgA'
colnames(IgA_max) <- "IgA"

# Grab max IgG values across all genes
IgG <- cbind(geo.hvg$IGHG1, geo.hvg$IGHG2, geo.hvg$IGHG3, geo.hvg$IGHG4)
IgG_max <- as.matrix(apply(IgG, 1, max))
# Set column name to 'IgG'
colnames(IgG_max) <- "IgG"

# Determine if IgA is higher or IgG is higher
Ig <- cbind(IgA_max, IgG_max)
Ig_max <- colnames(Ig)[(apply(Ig, 1, which.max))]
```

We have now classified cells as IgA or IgG expressing plasmablasts.

Exercise 1.2 — Data Visualization Principal component analysis (PCA) is a linear dimensionality reduction algorithm that is often the first-step in visualizing high-dimensional data. PCA takes an input of correlations between cells based on gene expression data, and identifies principal components corresponding to linear combinations of genes, which cumulatively capture the variability of the total dataset. When the data is projected against the first few components, which account for the largest amount of variation, distinct populations can be visually and biologically interpreted. Let's use the top two principal components to visualize our data:

```r
# Let's perform PCA
geo.pca <- prcomp(geo.hvg, center = TRUE, scale = TRUE)
plot_pc_data <- data.frame(PC1 = geo.pca[["x"]][, "PC1"], PC2 = geo.pca[["x"]][,
    "PC2"])
```

In order to visualize our scRNA-seq data using PCA, we will need to use some functions from ggplot2. Let's load the R package:

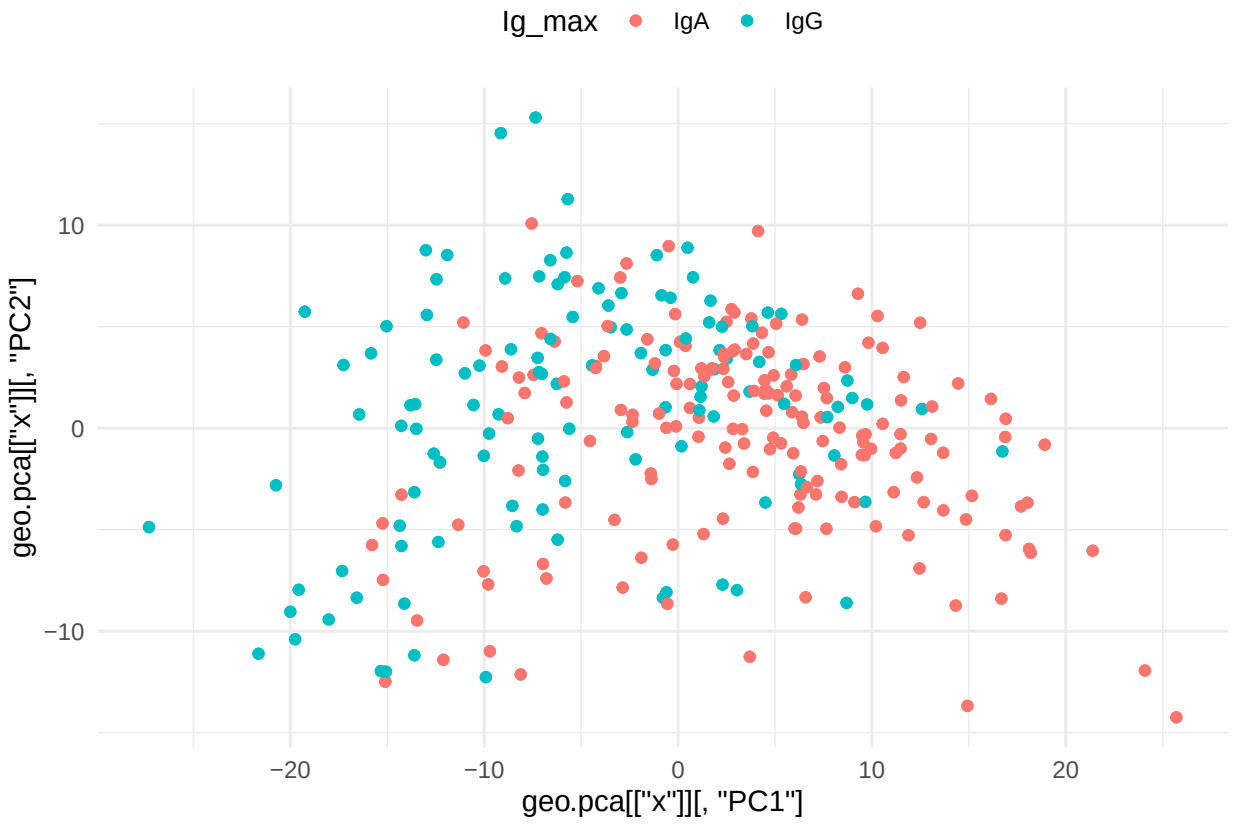
```r
library("ggplot2")
```

As a general rule of thumb, if you get an error message saying there is no package titled ggplot2 you may need to first install the appropriate package:

```r
# install.packages('ggplot2') library('ggplot2')
```

```r
# plot PCA results with Ig status
ggplot(plot_pc_data, aes(x = geo.pca[["x"]][, "PC1"], y = geo.pca[["x"]][, "PC2"],
```

```
    color = Ig_max)) + geom_point(shape = 1) + theme_minimal() + geom_point(aes(color = Ig_max)) +
    theme(legend.position = "top")
```



Do IgA cells and IgG cells separate well using PCA?

t-distributed stochastic neighbor embedding (t-SNE) is a widely used nonlinear dimensionality reduction algorithm. Unlike PCA, which seeks to capture variance in data, t-SNE seeks to explicitly preserve the local structure of the original data. t-SNE constructs a probability distribution to describe the data set such that pairs of similar cells are assigned a high probability, while dissimilar pairs are assigned a much smaller probability. Thus, cells that are similar in the high-dimensional space will cluster together (due to high probability) in low-dimensional space. This ability to explicitly maintain clustering of similar cells is an advantage of t-SNE over direct linear transformation such as PCA. This approach is very effective with scRNA-seq data, and has been used to resolve transcriptionally distinct populations that are indistinguishable with PCA. We will need to use some functions from Rtsne, so let's load the R package:

```
library("Rtsne")
```

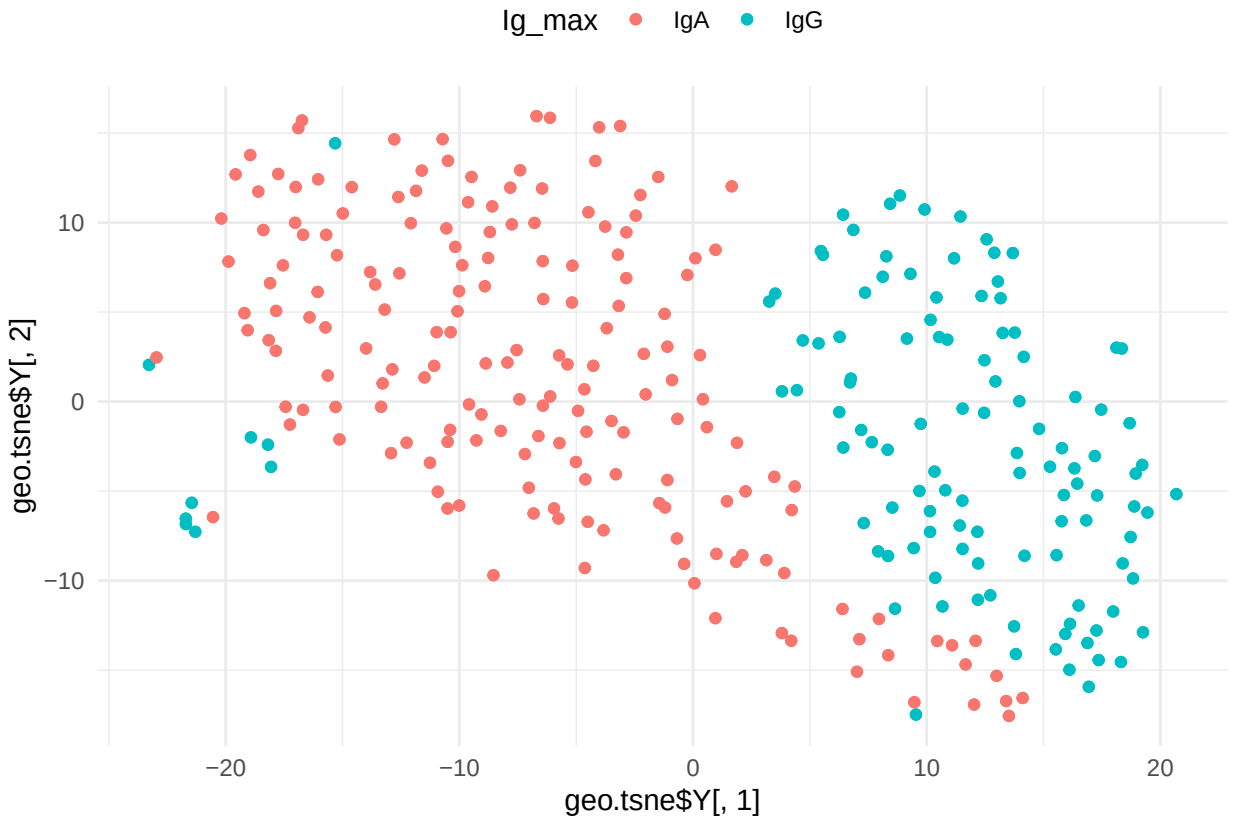
If you get an error message saying there is no package titled Rtsne you may need to first install the package.

```
# install.packages('Rtsne') library('Rtsne')
```

```
# Let's use the top 10 PC for t-SNE
geo.tsne <- Rtsne(geo.hvg, theta = 0.001, perplexity = 30, initial_dims = 10)
plot_tsne_data <- data.frame(tsne1 = geo.tsne$Y[, 1], tsne2 = geo.tsne$Y[, 2])
```

Let's visualize our data:

```
# plot PCA results with Ig status
ggplot(plot_tsne_data, aes(x = geo.tsne$Y[, 1], y = geo.tsne$Y[, 2], color = Ig_max)) +
    geom_point(shape = 1) + theme_minimal() + geom_point(aes(color = Ig_max)) + theme(legend.position =
```



Do IgA cells and IgG cells separate well using t-SNE?

Discussion While single cell transcriptional profiles have high dimensionality due to the thousands of genes profiled, their intrinsic dimensionalities are typically much lower. Thus, unsupervised low dimensional projections can reveal salient structure in large-scale scRNA-seq datasets. However, the choice of dimensionality reduction algorithms used to visualize and to infer cell similarity needs careful thought for immunology applications.

Exercise 2.1 — Data wrangling IEDB is a public database of immune epitope information. The database contains data related to antibody and T cell epitopes for humans, non-human primates, rodents, and other animal species. In particular, the database contains extensive MHC class I binding data from a variety of different antigenic sources.} Let's begin by downloading some peptide-MHC binding data. High-throughout peptide screening through competition experiments have resulted in large datasets cataloging binding affinities between various MHC molecules and peptides. For this exercise we will be using data from the The Immune Epitope Database (IEDB). We will download pre-processed peptide-MHC binding data from the IEDB database: http://tools.iedb.org/static/main/binding_data_2013.zip (Alternatively, you can use the file in the data folder on github.)

Once the peptide-MHC binding data is downloaded, you can uncompress the file. The resulting tab-delimited file contains nearly 200,000 peptide-MHC combinations. Let's analyze the data again using R:

```
# load data
iedb <- read.csv("bdata.20130222.mhci.txt", header = TRUE, sep = "\t", as.is = TRUE)

# let's use head to view a snippet of the data
head(iedb)
```

```
##      species         mhc peptide_length sequence inequality       meas
```

```
## 1 chimpanzee Patr-A*0101             8 RRDYRRGL          =   778.5834
## 2 chimpanzee Patr-A*0101             8 YHSNVKEL          = 18806.1666
## 3 chimpanzee Patr-A*0101             8 AQFSPQYL          = 22203.1869
## 4 chimpanzee Patr-A*0101             8 GDYKLVEI          > 87128.7129
## 5 chimpanzee Patr-A*0101             8 RGYVFQGL          > 87128.7129
## 6 chimpanzee Patr-A*0101             8 RPPLGNWF          > 87128.7129
```

You should see the first few lines of the file, including the header for the columns. Let's take a moment to interpret what the values mean for each of the columns:

- species - This is the species from which a specific MHC allele was evaluated for peptide binding.
- mhc - This is the specific MHC allele.
- peptide length - MHC class I molecules bind peptides that are predominantly 8-10 amino acid in length. Traditionally, there has been a focus on 9mer peptides when mapping HLA-I restricted T cell epitopes.
- sequence - This is the sequence of the peptide.
- inequality - This reflects the uncertainty for some of the peptide MHC binding data, where there some reported affinities are either an upper-bound or lower-bound to the true binding affinity.
- meas - The predicted output is given in units of IC50 nM. Therefore a lower number indicates higher affinity. As a rough guideline, peptides with IC50 values <50 nM are considered high affinity.

Why is this interesting? Several T-cell based cancer immunotherapies are being engineered to drive anti-tumor immune responses for specific antigens presented by the human MHC allele HLA-A*02:01. In cancer, somatic mutations altering the amino acid sequence of endogenous protein coding genes can result in the generation of tumor-specific HLA-presented antigenic peptide epitopes (or neo-antigens). These neo-antigens have the potential to activate cytotoxic T lymphocytes (CD8+ T cells) of the host immune system through HLA class I molecules, thereby provoking an anti-tumor immune response. It stands to reason that if we knew the binding specificity of a given MHC, we could evaluate different somatic mutations in a cancer sample and determine if it could be presented by the cancer.

Given the data available, can we model the repertoire of high affinity peptides that are presented by the human HLA-A*02:01 allele? Let's try to model the distribution in a position specific manner:

```
# let's select human, 'HLA-A*02:01', peptides of length 9 and binding affinity <
# 50
filtered_iedb = subset(iedb, species == "human" & mhc == "HLA-A*02:01" & meas < 50 &
    peptide_length == 9)

# let's grab the peptide sequences
peptide_sequences = filtered_iedb[, 4]
```

Exercise 2.2 — Data Visualization One way to visualize the repertoire of high affinity peptides that can bind to HLA-A*02:01 is to use a sequence logo plot. First, the relative frequency of each amino acid at each position is calculated. This can be referred to as a positional weight matrix (PWM). A PWM is a type of scoring matrix in which amino acid substitution scores are inferred separately for each position from a collection of aligned protein sequences. Second, the logo plot depicts the relative frequency of each character by stacking characters on top of each other, with the height of each character proportional to its relative frequency. The total height of the letters depicts the information content of the position, in bits. Here, we will use an R package called ggseqlogo to calculate the position specific frequencies for all high affinity 9mer peptides and visualize the sequence logo. Notably, the Matthew Stephens Lab at UChicago has also developed a sequence logo tool called Logolas.

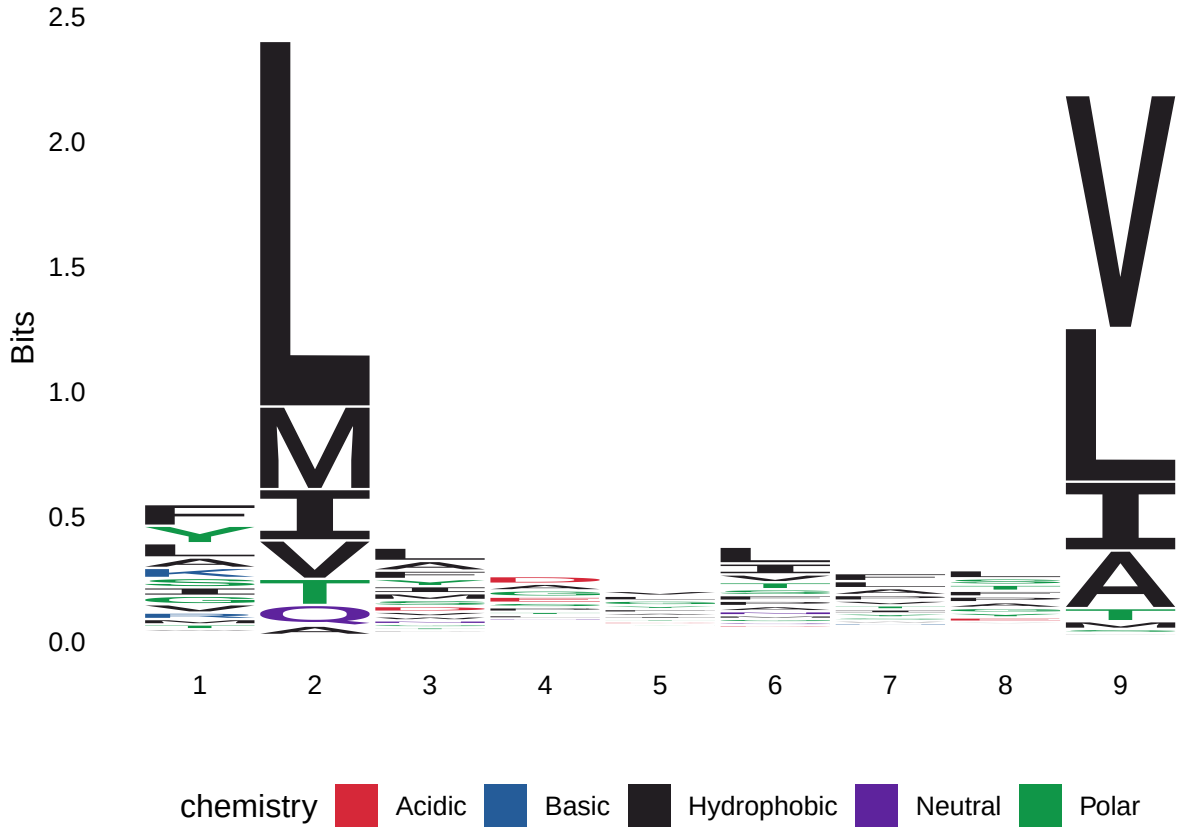
```
library(ggseqlogo)

# First let's install a seqlogo tool if not present install.packages('ggseqlogo')
# library(ggseqlogo)
```

```
# You can also install Logolas BiocManager::install('Logolas') library(Logolas)
```

Let's pass the list of 9mers to ggseqlogo to visualize sequence logo:

```
ggseqlogo(peptide_sequences, seq_type = "aa")
```



What positions of the high affinity peptides to seem to be highly specific for binding HLA-A*02:01? How does this compare with high affinity binding specificities of other HLA, for example, HLA-C*06:02?

Discussion Although we have made tremendous progress in modelling peptide-MHC interactions over the past several decades, connecting which T cells interact with which MHC-bound antigens remains a challenge. An efficent solution to this problem would have broad applications in improving our understanding of T cells in health, autoimmunity, and cancer (and potentially a free trip to Sweden). This remains a challenging task, in part, due to the enormous number of potential T cell receptors, the diversity of the MHC and bound antigen peptides. However, new computational methods may help resolve TCR-pMHC interactions by integrating and learning complex patterns from diverse high-throughput experimental approaches.