Statistical modeling of Epstein-Barr virus infection using scRNA-Seq host expression

by

Japan Patel

B.S. Mathematical Biology, University of Pittsburgh, 2022

B.S. Economics, University of Pittsburgh, 2022

Submitted to the Graduate Faculty of the

School of Public Health in partial fulfillment

of the requirements for the degree of

Master of Science

University of Pittsburgh

2024

UNIVERSITY OF PITTSBURGH

SCHOOL OF PUBLIC HEALTH

This thesis was presented

by

Japan Patel

It was defended on

April 16, 2024

and approved by

Jeanine M Buchanich MEd, MPH, PhD, Associate Professor, Department of Biostatistics

Jiebiao Wang PhD, Assistant Professor, Department of Biostatistics

Kathy H.Y. Shair PhD, Assistant Professor, Department of Microbiology & Molecular Genetics

Thesis Advisor: Jeanine M Buchanich, Associate Professor, Department of Biostatistics

Statistical modeling of Epstein-Barr virus infection using scRNA-Seq host expression

Japan Patel, MS

University of Pittsburgh, 2024

Epstein-Barr virus (EBV) is a ubiquitous virus that infects the majority of people worldwide. EBV replication is characterized by latent and lytic cycles where viral gene expression during latency is associated with certain cancers, such as Burkitt's Lymphoma and Nasopharyngeal Carcinoma (NPC). The field's understanding of viral-host mechanisms of EBV in epithelial cells is incomplete. To determine host gene expression profiles which influence the lifecycle of EBV we modeled Single Cell RNA Sequencing (scRNA-seq) data of EBV-infected cell lines by a Random Forest algorithm and multinomial logistic regression approaches. This methodology allowed us to refine EBV infection status into established and newly identified classifications, defining subcategories of lytic and latent cycles as well as unveiling specific host genes and biological pathways influential to EBV pathogenesis. Our analysis revealed that certain host genes, implicated in pathways related to viral mRNA translation, keratinization, neutrophil degranulation, and cytokine signaling, play a significant role in shaping the viral-host interaction landscape. Traditionally, scRNA-data is limited by the prevalence of false negatives which arise due to low abundance of EBV transcripts in most cells. These host genes served as surrogate markers of infection which enabled us to predict infection status in cells that otherwise appear as void of EBV infection. Future in-vivo and in-vitro analysis should be conducted on a variety of epithelial and B-cell lines to detect conserved host markers of EBV infection to further the field's understanding of EBV viral-host interactions possibly contributing to the development diagnostic markers or therapeutic targets for EBV associated diseases.

Table of Contents

List of Figures

1.0 Introduction

Epstein-Barr virus (EBV) is a ubiquitous γ-herpesvirus which affects more than 90% of individuals worldwide [1]. EBV replication is characterized by latent and lytic cycles where viral gene expression during latency is associated with certain B-cell lymphomas and epithelial carcinomas, such as Burkitt's Lymphoma and Nasopharyngeal Carcinoma (NPC). EBV associated NPC is endemic to Southeast Asia with incidence rates ranging from 3-25/100,000 persons [2]. Certain at-risk populations in the United States, such as Asian Americans, show elevated incidence of approximately 10/100,000 persons compared to 0.5/100,000 persons in low-risk populations. Only 10% of NPC cases are diagnosed during Stage I when five-year survival exceeds 90%, compared to 60% survival for Stage IV patients.

The nasopharynx is composed of pseudostratified epithelium and multiple cell types, establishing a unique biology where in vivo detection of EBV is infrequent and factors of NPC onset are difficult to study using traditional approaches [3]. Differentiation of infected epithelial cells can trigger the reactivation of EBV from latency to the lytic phase which influences viral-host gene interactions. We hypothesize that statistical modeling of Single Cell RNA Sequencing (scRNA-seq) data of EBV-infected cell lines will reveal insights into viral-host genome interactions adding to the incomplete model of EBV pathogenesis and NPC onset. To investigate host expression profiles which contribute to EBV infection status, we implemented a unique data analysis methodology leveraging scRNA-seq data to model EBV infection status by a Random Forest algorithm and multinomial logistic regression approaches. Our data explores well known EBV infection states as well as proposes novel infection profiles to elucidate influential host genes vital to the mechanisms of EBV pathogenesis. These influential host genes are surrogate markers of infection states, addressing the problem of misidentification of false negatives within the scRNA-seq dataset produced by a low abundance of EBV transcripts.

2.0 Methods

2.1 Single Cell Data

2.1.1 Single Cell Methods

Six HK1 cell lines, originating from a differentiated NPC tumor biopsy specimen and infected with EBV stably expressing either China 1 or a vector control LMP1 sequence variant (known as LMP1 strain), were established in Air Liquid Interface (ALI) for simultaneous collection at days 0, 2, and 4 post-infection [3]. China 1 is noted as the dominant LMP1 strain in NPC tumor biopsies and throat washing samples overexpressing primary oncogene Latent Membrane Protein 1 (LMP-1) [4]. Samples were multiplexed into two libraries (Ch1 and IRES) using TotalSeq-C0251 (Biolegend). Single-cell RNA sequencing (scRNA-seq) was conducted using the 10x Genomics 5' v2 library. Data was accessed and analyzed in Python and R environments through the University of Pittsburgh Center for Research Computing.

Cellranger (v. 7.1.0) was employed to demultiplex and align scRNA-seq data against a tailored reference, combining the human genome (GRCh38), EBV Akata reference genome (KC207813.1), and the HA-tagged LMP-1 sequence, integrated using a Python script [5]. Akata is a strain of EBV isolated from Burkitt's lymphoma cells that can be effectively induced to enter lytic cycles [6].

2.1.2 Seurat Workflow

Seurat (v. 4.3.0) was used to process Cellranger outputs [7]. Initial steps involved filtering cells to uphold quality control metrics. Low quality cells often exhibit mitochondrial contamination, thus cells with more than 20% mitochondrial counts were removed. Low quality cells or empty droplets tend to have few genes whereas cell doublets or multiplets exhibit high gene count. Cells with RNA feature counts less than 200 or greater than 9000 were not retained.

Following quality control steps, the data was normalized by the "LogNormalize" function in the Seurat package [7]. This global-scaling method normalized gene expression measurements for every cell, multiplied each by a scale factor of 10,000, and log-transformed the data.

Variable genes were selected following data normalization. Genes exhibiting high between cell variability often highlight biological effects in scRNA-seq data. The "FindVariableFeatures" function from the Seurat package was used to identify 3500 variable genes by utilizing a variance stabilizing transformation (VST) approach [7]. VST considered the relationship between the log-variance and the log-mean and fitted a line using local polynomial regression, Locally Weighted Scatter-plot Smoother (LOESS) [8]. LOESS is a nonparametric method with relaxed linearity assumptions to fit a smooth curve between two variables. LOESS fitted multiple models to localized subsets of the data. For each point, LOESS selected a subset of data points that were closest to the point of interest. The selection was based on a parameter that defined the proportion of the data to be used in each local fit. The selected data points were weighted based on their distance from the point of interest, with points closer to the target receiving higher weights. A linear model was fitted to the selected and weighted data points. These steps were repeated for each point in the dataset, resulting in a smooth curve. The following assumptions were made:

1. The mean of y around point x can be approximated through a class of parametric functions based on polynomial regression.
2. Errors in estimating y are independently and randomly distributed with mean 0.

The data was then scaled using a linear transformation by the “ScaleData” function from the Seurat package [7]. The data was transformed such that the mean gene expression across all cells was 0 and the variance was 1. Principle Component Analysis (PCA) was conducted on the scaled data. PCA begins by computing the covariance matrix of the scaled data to evaluate variation between genes. Eigen-decomposition was conducted on this covariance matrix to extract the eigenvalues and eigenvectors. The eigenvalues represent the amount of variance captured by each principal component, while the eigenvectors defined the direction of these components in the multidimensional space. The principal components were ranked according to their corresponding eigenvalues, with higher eigenvalues indicating components that capture more variance. This step reduced the dimensionality of the data, with the first principal component accounting for the largest possible variance, and each subsequent component capturing the maximum remaining variance under orthogonality to preceding components.

We utilized a K-nearest neighbor (KNN) graph-based clustering approach, where cells are represented as nodes, and edges are drawn between cells exhibiting similar gene expression patterns. Clustering analysis leveraged the PCA dimensionality-reduced data, specifically utilizing the first 12 principal components to define the cellular distance metric. The “FindNeighbors” function from the Seurat package, constructed a KNN graph based on Euclidean distances within the PCA-reduced space [7]. The graph's edge weights were refined using Jaccard similarity to reflect the shared overlap in local neighborhoods of cells. Modularity optimization techniques were applied to partition the graph into clusters. The Louvain algorithm iteratively grouped cells to optimize a standard modularity function, indicating the strength of division of the graph into clusters.

2.1.3 Single Cell Visualizations

Uniform Manifold Approximation and Projection (UMAP) was utilized for dimensionality reduction and visualization of cellular clusters. The UMAP algorithm utilized results from previous PCA to project high-dimensional data into a two-dimensional space. Violin plots were generated using the "VlnPlot" function from the Seurat package to visualize the distribution of gene expression levels across different cellular conditions or clusters [7]. The proportional composition of cell types across different conditions or clusters was illustrated using stacked bar plots, generated by the ggplot2 package [9]. Scatterplots plots were generated to compare the expression levels of multiple genes across all cells using the ggplot2 package. Heatmaps were created using the "ComplexHeatmap" package to depict the expression patterns of genes across different cellular conditions and clusters [10]. Heatmaps included hierarchical clustering to group genes and cells based on similarities in expression profiles. Visualizations were used to determine unique five cellular infection states (LF3 associated Δ-lytic, LMP-1/BNLF2a associated Δ-lytic, LMP-1/BNLF2a+LF3 associated Δ-lytic, lytic cells undergoing host shutoff, and lytic cells not undergoing host shutoff).

2.1.4 Differential Gene Testing

Differential gene testing was conducted using the "FindMarkers" function from the Seurat package [7]. This test utilized the non-parametric Wilcoxon rank sum test to discern genes exhibiting significant differences in expression levels between any two distinct cellular populations [11]. The null hypothesis posits that there is no difference in the distribution of gene expression levels between the two compared groups. The alternative hypothesis contends that there exists a difference (two-sided) in the distribution of gene expression levels between the groups. The Wilcoxon rank sum test assumes:

1. Independence between groups.
2. Ability to rank data.

The test ranked all observations across both groups simultaneously and compared the sum of ranks in one group against the sum of ranks in the other. The difference in these rank-sums served as the basis for evaluating the significance of the observed difference in gene expression between cell populations. The FindMarkers function yielded several key metrics [7]:

1. Average Log-Fold Change (AvgLog2FC), a measure of the magnitude of differential expression between two groups. Calculated as the logarithm (base 2) of the ratio of the average expression levels of a gene in two groups. This metric indicated changes in magnitude of the average gene expression in one group compared to another. Positive values indicate upregulation and negative values indicate downregulation in the first group relative to the second.
2. Proportion of cells within the first group (PCT1) and the second group (PCT2) where the gene of interest is detected above a threshold level of expression ($|\text{AvgLog2FC}|>0.25$).
3. Unadjusted p-value is calculated from the test statistics derived from the median ranks of gene expression levels between the two groups.
4. The adjusted p-value accounts for the multiple comparisons through Bonferroni correction. This was done by dividing the desired overall α level (0.05) by the number of

tests conducted. This calculation yielded a stricter threshold for statistical significance reducing the chance of false positives.

Differentially expressed genes were determined to be candidate predicters for subsequent model fitting.

2.2 Random Forest Modeling

2.2.1 Random Forest Methods

A Random Forest algorithm was employed for regression tasks using the ranger package [11]. The Random Forest algorithm utilized ensemble learning, where multiple decision trees were constructed during the training phase [13]. The model predicted outcomes based on the mean of the predictions from all trees in the forest.

Combined gene lists derived from Differential Gene Testing, infection states derived from single cell visualizations, cell identifiers, and normalized gene expression values were merged to curate a dataset tailored for subsequent analysis. A 'class' (classification) variable, denoting different cellular infection states, was transformed into a categorical factor variable. To promote analytical reproducibility and consistency, a specific seed (123) was set.

Following common machine learning practices, the dataset was randomly divided, allocating 70% to the training set and the remaining 30% to the testing set. The model was configured to generate 500 decision trees and the number of variables to fit at each split was determined by the square root of the total number of variables in the dataset.

Post-training, the model's classification ability was evaluated on the testing set. Predicted classifications were juxtaposed against actual classifications to construct a confusion matrix. From the confusion matrix generated by comparing the actual and predicted classifications on the test set, sensitivity (true positive rate) was calculated as the proportion of true positive observations to the total actual positives, defined as:

$$Sensivity = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

where TP represents true positives and FN represents false negatives. Similarly, specificity (true negative rate) was determined as the proportion of true negative observations to the total actual negatives, defined as:

$$Specificity = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

where TN represents true negatives and FP represents false positives [14]. Each candidate predictor variable was given a relative importance score based on model contribution. The top 50 predictor genes were used to determine significant biological pathways using Reactome [21].

2.2.2 Random Forest Assumptions

The Random Forest algorithm assumes [13]:

1. The errors across individual trees are uncorrelated.
2. The ensemble of trees will provide superior prediction accuracy compared to any single tree within the forest.

2.3 Multinomial Logistic Regression Modeling

2.3.1 Multinomial Logistic Regression with AIC

The top 25 genes based on Random Forest importance score were incorporated into a multinomial logistic regression model [15] using the nnet package [16].

$$\log\left(\frac{P(Y=k)}{P(Y=K)}\right) = \beta_{0k} + \beta_{1k}X_1 + \beta_{2k}X_2 + \cdots + \beta_{pk}X_p$$

where:

$P(Y = k)$ is the probability of the dependent variable being in category k,

$P(Y = K)$ is the probability of the dependent variable being in the reference category,

$X_1, X_2, \dots, X_p$ are the independent predictor variables,

$\beta_{0k}, \beta_{1k}, \dots, \beta_{pk}$ are the coefficients for category k which will be estimated by the model.

Similar to the dataset used for the previous random forest model, this model utilized a merged dataset containing combined gene lists derived from top ranking genes by Random Forest importance score, infection states derived from single cell visualizations as a factor variable, cell identifiers, and normalized gene expression values.

Gene selection was optimized by implementing an Akaike Information Criterion (AIC) iterative gene elimination strategy. A loop mechanism assessed the impact of removing each gene on the model's AIC. For each iteration, a temporary model excluding one gene was constructed to compute the AIC, enabling the identification of the gene whose exclusion reduced the AIC. This gene was then permanently excluded in subsequent iterations to minimize AIC. This process was repeated until no further reduction in AIC was observed upon gene exclusion. The predictive performance of the model was evaluated using a confusion matrix, sensitivity, and specificity.

2.3.2 Multinomial Logistic Regression Assumptions

The multinomial logistic regression model assumes [15]:

1. Independence of irrelevant alternatives.
2. Linearity of independent variables and log-odds.
3. No perfect multicollinearity. The Variance Inflation Factor (VIF) was calculated using the "vif" function from the car package to assess multicollinearity among the predictors.
4. Sufficiently large sample size.
5. Multinomial distribution of the dependent variable.

2.3.3 Multinomial Logistic Regression with Elastic Net

The same dataset was utilized to fit a multinomial logistic regression model with an Elastic Net penalty [17] using the "cv.glmnet" function from the glmnet package [18,19]. Multinomial logistic regression modeled the log-odds of the probabilities of K-1 classifications as a linear combination of the predictors compared to reference category K. The probability of observing category k for a set of predictors x is given by [15,17]:

$$P(Y = k|x) = \frac{\exp(\beta_{0k} + \beta_k^T x)}{1 + \sum_{i=1}^{K-1} \exp(\beta_{0i} + \beta_i^T x)}$$

and

$$P(Y = K|x) = \frac{1}{1 + \sum_{i=1}^{K-1} \exp(\beta_{0i} + \beta_i^T x)}$$

where:

Y is a categorical response variable with K categories, x is the vector of predictor variables,

β_{0k} is the intercept term for category k,

β_k is the vector of coefficients for category k.

The Elastic Net penalty for each non-intercept coefficient in the above model is defined as [17]:

$$\lambda\left(\frac{1-\alpha}{2}\sum_{j=1}^{P}\beta_j^2+\sum_{j=1}^{P}|\beta_j|\right)$$

where:

λ is a regularization parameter adjusting the strength of the penalty,

α is a parameter to adjust lasso and ridge penalties ($\alpha = 1\text{: lasso}, \alpha = 0\text{: ridge}$),

β_j represents non-intercept coefficients for all predictor variables across all categories k,

p is the total number of predictor variables.

Combining the multinomial logistic regression model with the Elastic Net penalty yields:

$$-\left[logL(\beta)-\lambda\left(\frac{1-\alpha}{2}\sum_{j=1}^{P}\beta_j^2+\sum_{j=1}^{P}|\beta_j|\right)\right]$$

to be minimized where:

$logL(\beta)$ is the log-likelihood of the multinomial logistic regression model.

The Elastic Net regularization technique combined the properties of Lasso and Ridge penalties to regularize regression models. The initial α parameter was set to 0.5, to choose a balanced Elastic Net approach that equally weighs the lasso and ridge penalties. The best λ value identified during cross-validation was used to finalize the model. The predictive performance of the model was evaluated using a confusion matrix, sensitivity, and specificity.

3.0 Results

3.1 Single Cell Data

3.1.1 Single Cell Visualizations

Samples from 6 samples representing two LMP1 condition states at three early time points post ALI-induced reactivation (Fig 1B) were combined into one comprehensive dataset encompassing a total of 31,619 cells, with 12 distinct cellular clusters (Fig. 1A).

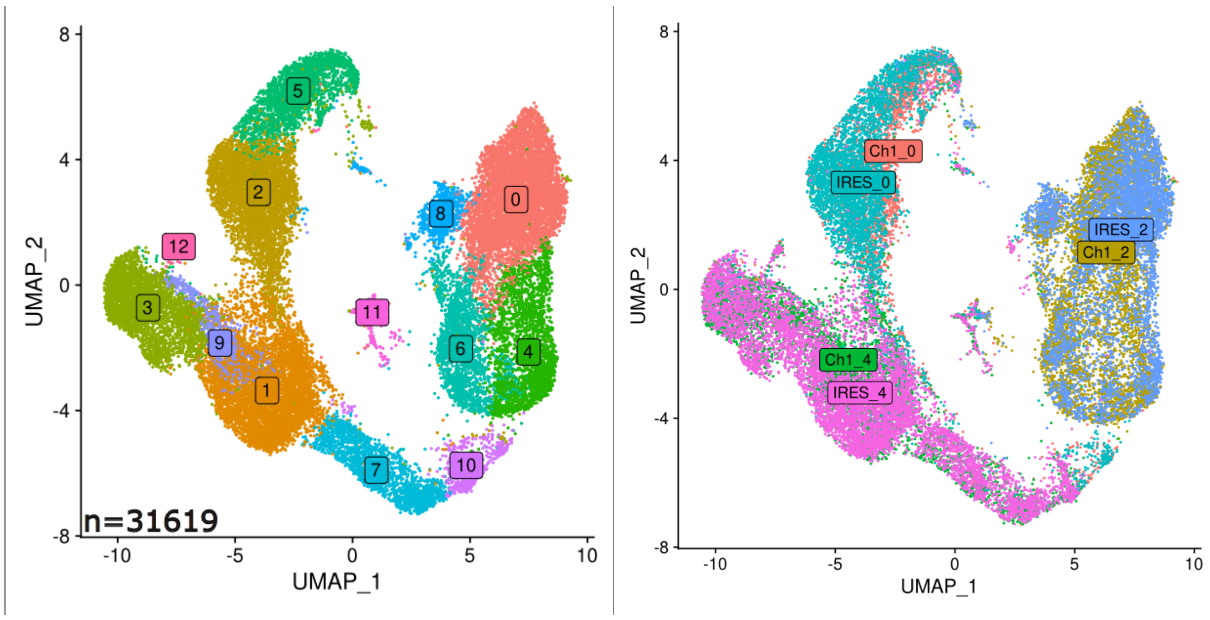

Figure 1A and 1B. UMAP of cellular clustering and sample origin

LMP-1 is the principal oncoprotein and crucial for EBV pathogenesis, reprogramming host cellular mechanisms to promote oncogenic transformation, cell proliferation, and survival, among other functions [23]. It activates various signaling pathways contributing to the development of the tumor microenvironment and influencing metabolism, immune defenses, and antioxidative responses. The dataset examines early time points to reveal early reactivation effects before host-shutoff alters host gene expression profiles.

UMAP visualizations of cellular clusters by sample origin (Fig. 1B), indicated samples predominantly clustered according to time point post-reactivation, rather than by the LMP1 condition (China 1 or IRES). This indicates that the time point is a stronger classifier of the global transcription pattern than the additional stable expression of LMP1. Each cluster exhibited a unique distribution of cells originating from the six samples (Fig. 2).

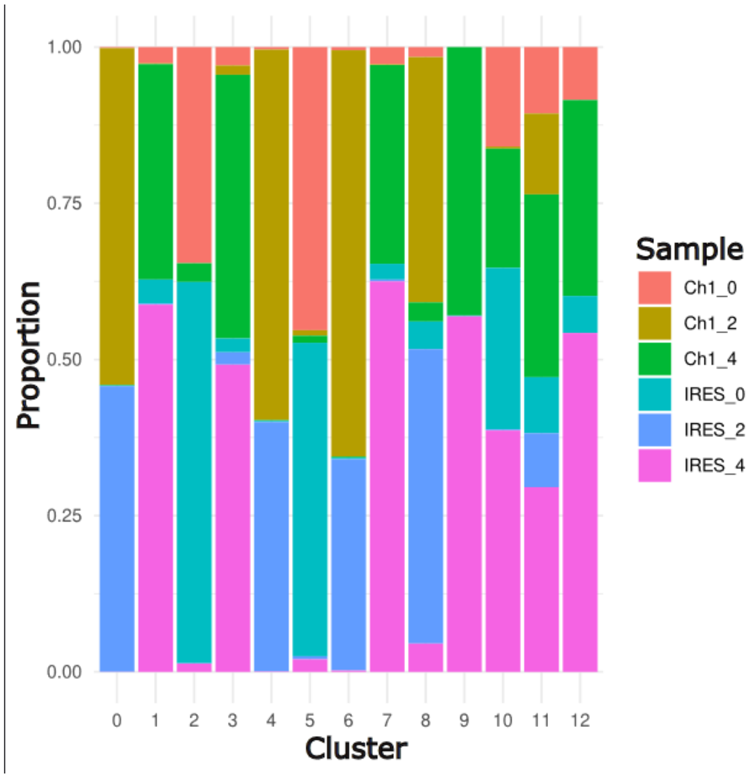

Figure 2. Proportion of sample origin by cluster

Cluster 11 displayed unique heterogeneity containing a proportional distribution of cells from all six samples. The violin plot of EBV gene expression by cluster (Fig. 3), designated cluster 11 as abundant in viral gene expression compared to other clusters, indicating that cluster 11 is the lytic cluster.

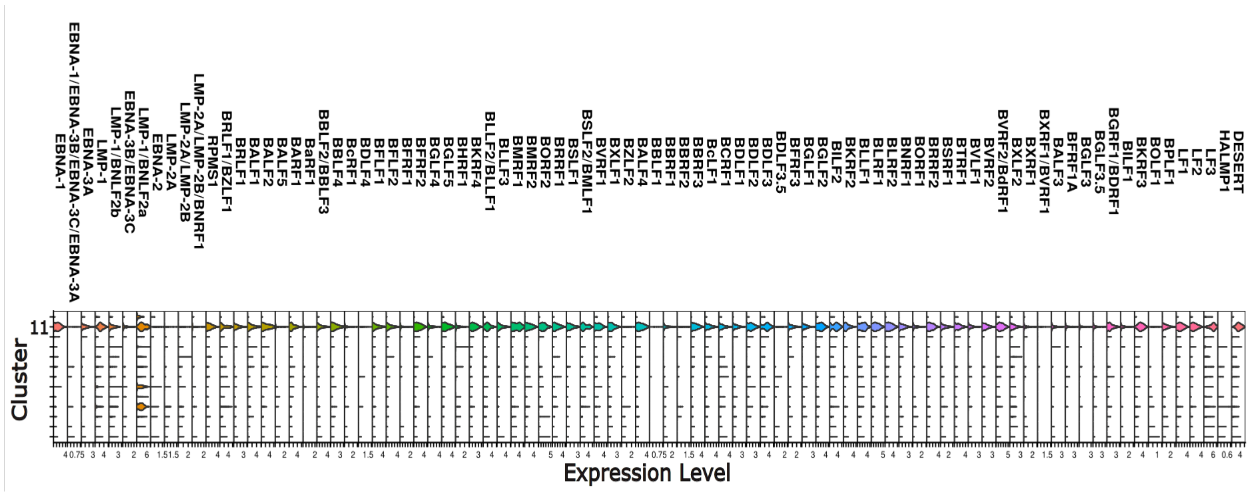

Figure 3. Violin plot of EBV gene expression by cluster

The heatmap of EBV gene expression by cluster (Fig. 4) revealed unique transcriptional profiles across different clusters. Similarly to Fig. 3, Cluster 11 expressed complete activation of the lytic cascade, showcasing the presence of gene expression across latent, immediate early, early, late, and unassigned EBV genes.

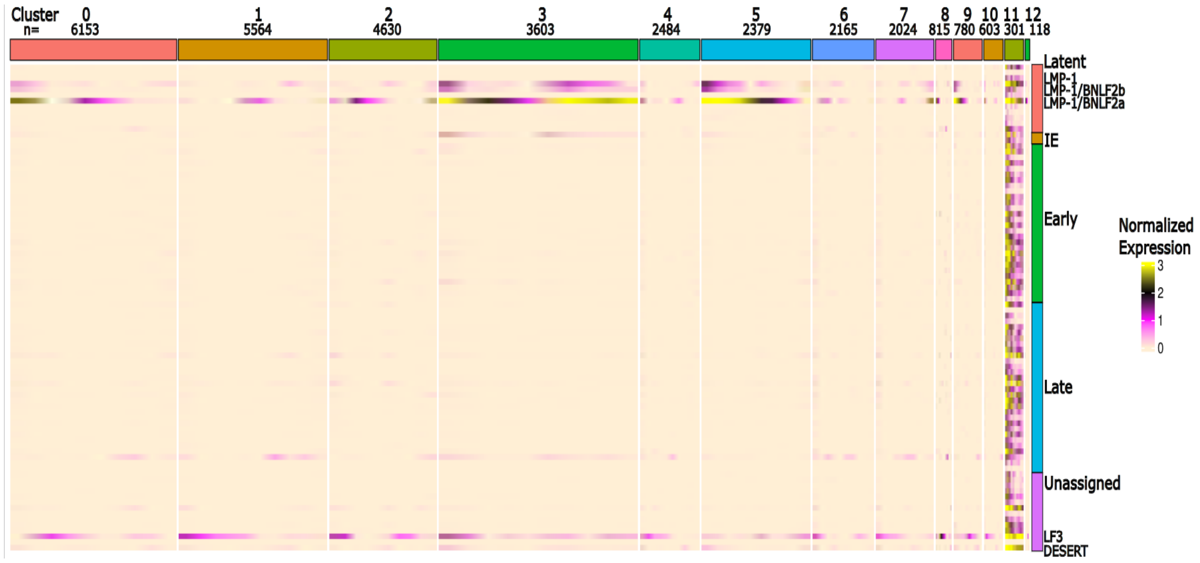

Figure 4. Heatmap of EBV gene expression by cluster

In contrast, other clusters only displayed notable expression patterns in three LMP-1 annotations (LMP-1, LMP-1/BNLF2b, and LMP-1/BNLF2a) and two unassigned annotations, LF3 and Desert. Desert represented regions within the EBV genome that are not annotated. Cluster 11 was categorized as a lytic group of cells whereas all other clusters were categorized as Δ-lytic. The tables presented in Fig. 5 show the distribution of cells by cellular cluster and infection classification. The notable differences between total cells and those exhibiting EBV transcripts within Δ-lytic clusters (1-10; 12) revealed the primary limitation of scRNA-seq data; most cells appear as uninfected due to a low abundance of EBV transcripts.

		Total	**EBV**
	0	6153	1788
	1	5564	2555
	2	4630	1438
	3	3603	1254
	4	2484	883
	5	2379	474
Cluster	**6**	2165	779
	7	2024	927
	8	815	250
	9	780	375
	10	603	308
	11	301	301
	12	118	41

Lytic	**LMP1**	6262
	LF3	4492
	LMP1+LF3	2088
Δ-Lytic	**HS**	44
	nHS	257

Figure 5. Table of cells by classification

Among 31,318 Δ-lytic cells, the heatmap of LMP1/BNLF2a and LF3 (Fig. 6) encompasses 10,754 cells. Notably, 4,492 cells expressed LF3 alone, 6,262 cells expressed LMP-1/BNLF2a alone, and 2088 cells exhibited co-expression of both LF3 and LMP1/BNLF2a. Therefore, LMP1/BNLF2a and LF3 expression was used to sub-categorize Δ-lytic cells.

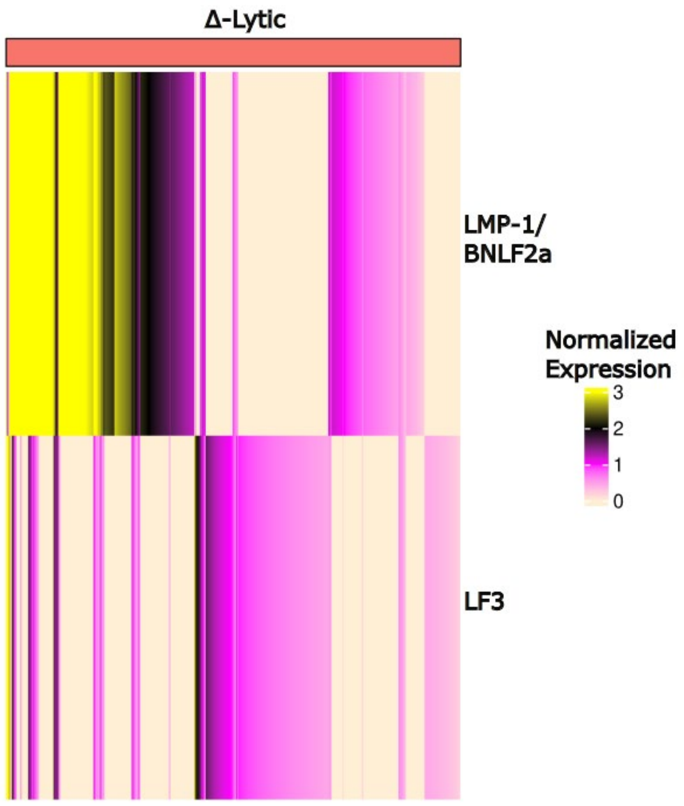

Figure 6. Heatmap of LMP1/BNLF2a and LF3 expression in Δ-lytic cells

The transcript for the primary EBV oncoprotein (LMP1) overlaps with the BNLF2a transcript and is therefore represented as LMP1/BNLF2a. LF3 can initiate from latent and lytic promoters and is contained within the 12kb deletion in the B958 EBV genome which is the original strain of EBV capable of showing immortalization of B cells [24]. LF3 is a paralog of BHLF1 and are some of the most abundant transcripts expressed during lytic infection. Both LF3 and BHLF1 are also expressed during latency and deletion of the BHLF1 gene attenuates EBV immortalization [25], although LF3 is naturally deleted in the B95-8 EBV strain that is used in immortalizing B-cells [26].

The violin plot illustrating the expression of BGLF5, a marker of host shutoff, across all clusters (Fig. 7) identified distinctively concentrated expression in the lytic cluster [20].

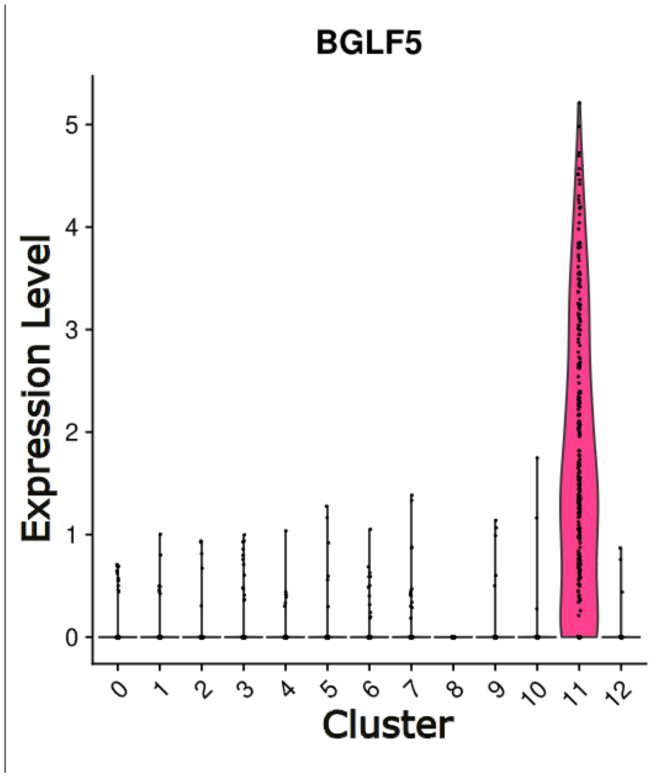

Figure 7. Violin plot of BGLF5 expression by cluster

The RNase activity of BGLF5 mediates host shutoff by mRNA degradation of both poly-adenylated and non-poly-adenylated transcripts [27]. Current configuration of the 10X Genomics libraries selectively enrich for poly-adenylated transcripts. Thus, shutoff of the most abundantly expressed viral transcript (EBER) which is non-poly-adenylated is not observable. However, downregulation of host genes is observable. BGLF5 expression was negligible in Δ-lytic clusters. The exclusive expression pattern in the lytic cluster suggested a population of cells undergoing host shutoff, a process where the virus inhibits host protein synthesis to favor viral gene expression. The scatterplot comparing the unique molecular identifiers (UMIs) associated with EBV genes (red) to the UMIs for all genes (blue) within each cell (Fig. 8) revealed a distinct population of cells characterized by a higher proportion of EBV gene UMIs compared to all gene UMIs.

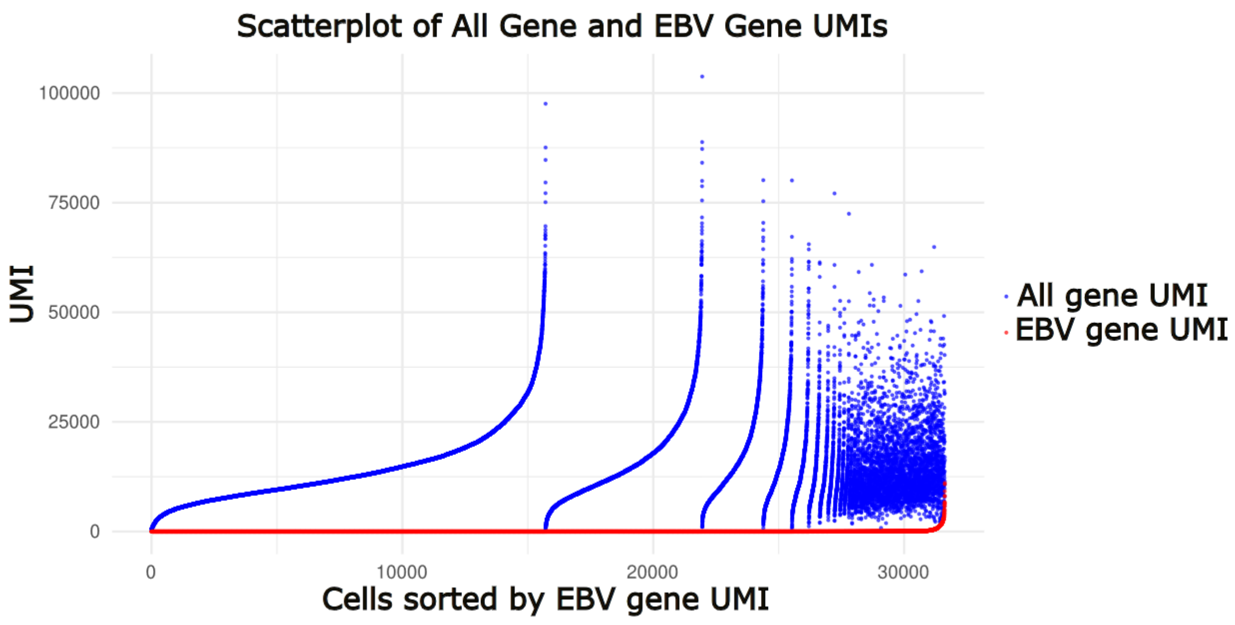

Figure 8. Scatter plot comparing EBV gene and all gene UMIs

This observation indicated a subset of cells where EBV transcription was disproportionately high, suggesting active viral gene expression patterns that may be overshadowing the host cell's transcriptome. EBV gene UMIs were divided by all gene UMIs per cell, representing the proportion of EBV gene to total gene transcripts within individual cells. The scatterplot which mapped the ratio of EBV gene UMIs to all gene UMIs in each cell, with an overlay of BGLF5 expression level (Fig. 9), revealed a population of cells characterized by both a high proportion of EBV gene UMIs and elevated expression of BGLF5. The expression of BGLF5 identified these cells as part of the lytic cluster (Fig. 7), undergoing host shutoff.

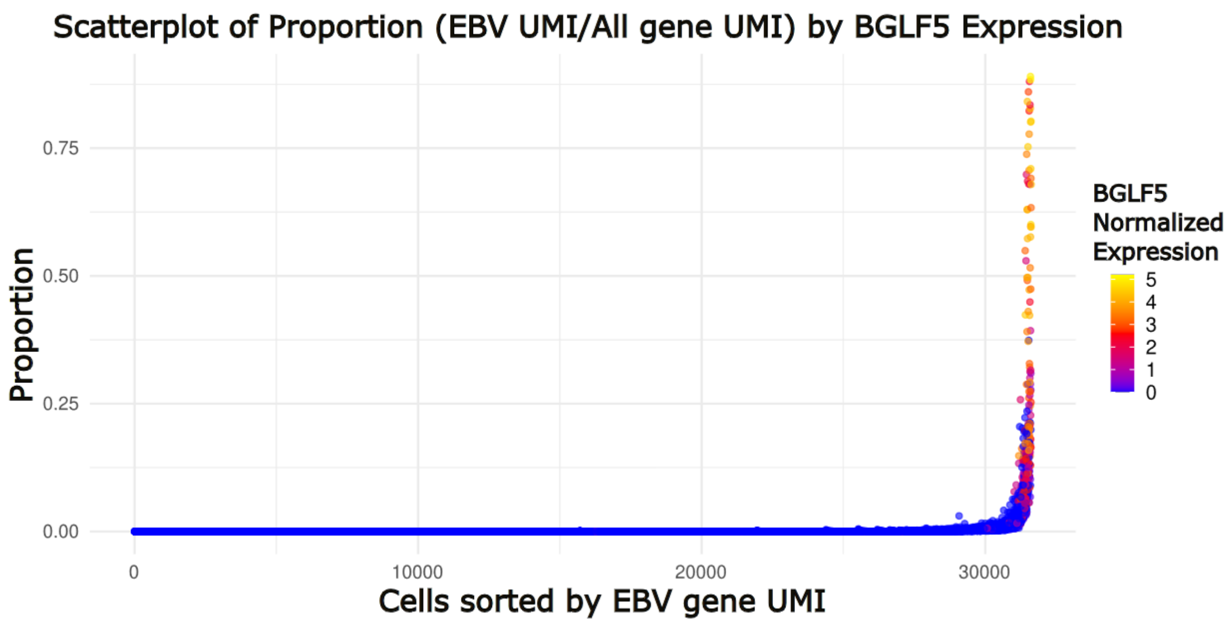

Figure 9. Scatterplot comparing proportion of EBV gene UMIs to all gene UMIs by BGLF5 expression

The delineation between the Δ-lytic and lytic groups was marked by their respective EBV to all gene UMI proportions; the Δ-lytic group exhibited proportions ranging from 0 to 0.37, while the lytic group's proportions spanned from 0.003 to 0.87. The ability to distinguish cells engaging in host shutoff is an advantage of scRNA-seq over conventional bulk RNA-seq, the parameters of which have not yet been defined by prior literature or exploited in differential gene analysis.

3.2 Random Forest Modeling

3.2.1 Random Forest Results

The tables presented in Fig. 10 are confusion matrices derived from the Random Forest model, which categorize the predictions of the five different EBV infection classifications (LF3 associated Δ-lytic, LMP-1/BNLF2a associated Δ-lytic, LMP-1/BNLF2a+LF3 associated Δ-lytic, lytic cells undergoing host shutoff, and lytic cells not undergoing host shutoff).

Random Forest

Proportion EBV UMI = 0.37		**Predicted**				
		Δ-Lytic			**Lytic**	
		LMP1	**LF3**	**LMP1+LF3**	**HS**	**nHS**
Actual	**LMP1**	**1548**	315	1	0	0
	LF3	287	**1064**	0	0	0
	LMP1+LF3	516	112	**1**	0	0
	HS	3	0	0	**14**	0
	nHS	46	36	0	0	**0**

Proportion EBV UMI = 0.20		**Predicted**				
		Δ-Lytic			**Lytic**	
		LMP1	**LF3**	**LMP1+LF3**	**HS**	**nHS**
Actual	**LMP1**	**1543**	319	2	0	0
	LF3	292	**1059**	0	0	0
	LMP1+LF3	517	110	**2**	0	0
	HS	10	2	0	**12**	0
	nHS	47	28	0	0	**0**

Proportion EBV UMI = 0.12		**Predicted**				
		Δ-Lytic			**Lytic**	
		LMP1	**LF3**	**LMP1+LF3**	**HS**	**nHS**
Actual	**LMP1**	**1535**	328	1	0	0
	LF3	270	**1080**	1	0	0
	LMP1+LF3	510	116	**3**	0	0
	HS	21	12	0	**14**	0
	nHS	26	26	0	0	**0**

Proportion EBV UMI = 0.08		**Predicted**				
		Δ-Lytic			**Lytic**	
		LMP1	**LF3**	**LMP1+LF3**	**HS**	**nHS**
Actual	**LMP1**	**1545**	316	3	0	0
	LF3	268	**1083**	0	0	0
	LMP1+LF3	510	113	**6**	0	0
	HS	36	17	0	**14**	0
	nHS	15	17	0	0	**0**

Figure 10. Random Forest summary

The matrices were separated by the proportion of EBV UMIs to all gene UMIs, with each table reflecting a different proportion threshold (greater than or less than 0.37, 0.08, 0.12, and 0.20), for example HS (host shut off) denotes a proportion greater than 0.37 and nHS (no host shut off) denotes a proportion less than 0.37. The rows represent the actual classifications of EBV infection, while the columns correspond to the predicted classifications by the model. Diagonal values indicate correct predictions, where the model's classification aligns with the actual classification, and off-diagonal values represent misclassifications. The tables presented in Fig. 11 display sensitivity and specificity metrics from the Random Forest model, which quantify model accuracy of the five different EBV infection classifications for each of the four EBV UMI proportions.

Random Forest

Proportion EBV UMI = 0.37	**Δ-Lytic**			**Lytic**	
	LMP1	**LF3**	**LMP1+LF3**	**HS**	**nHS**
Sensitivity	0.830472	0.787565	0.00159	0.823529	0
Specificity	0.590188	0.821373	0.999698	1	1

Proportion EBV UMI = 0.20	**Δ-Lytic**			**Lytic**	
	LMP1	**LF3**	**LMP1+LF3**	**HS**	**nHS**
Sensitivity	0.82779	0.783864	0.00318	0.5	0
Specificity	0.583454	0.822917	0.999396	1	1

Proportion EBV UMI = 0.12	**Δ-Lytic**			**Lytic**	
	LMP1	**LF3**	**LMP1+LF3**	**HS**	**nHS**
Sensitivity	0.823498	0.799408	0.004769	0.297872	0
Specificity	0.602213	0.814043	0.999396	1	1

Proportion EBV UMI = 0.08	**Δ-Lytic**			**Lytic**	
	LMP1	**LF3**	**LMP1+LF3**	**HS**	**nHS**
Sensitivity	0.828863	0.801628	0.009539	0.208955	0
Specificity	0.601251	0.821373	0.999095	1	1

Figure 11. Random Forest sensitivity and specificity

The tables presented in Fig. 12 display Reactome pathway analysis of the top 50 genes from the Forest model with importance scores [21].

Pathway Analysis

Viral mRNA Translation

RPL13	11.44078
RPS2	10.2332
RPL18A	11.04989
RPL32	12.60838
RPS18	9.742981
RPLP1	11.18593
RPL12	15.87549
RPS24	10.15674

Keratinization

KRT15	23.33335
KRT16	24.12137
JUP	16.35692
PERP	9.7588
KRT13	24.62746
KRT6A	14.33602
SPRR1A	21.44978
DSC2	17.62421
SPRR1B	43.61339

Neutrophil Degranulation

S100A7	12.64868
GSTP1	10.97858
S100A8	33.35091
S100A9	34.19331
SLPI	15.28682
HSP90AB1	10.0238
S100A11	13.14203
LGALS3	20.62328
CSTB	24.40935

Cytokine Signaling

MT2A	31.75472
ANXA1	14.81194
IL1RN	17.07174
YWHAZ	11.20575

Pathways with less than 3 Entities

CLDN4	53.56539
CD24	50.35485
ELF3	34.92743
NECTIN4	26.6372
CAV1	23.71347
CDKN2B	18.31926
KLK6	15.77327
AQP3	12.56929
CAST	12.50293
PFN1	11.28674
MT1E	11.05495
MDK	9.743815

Not Found

SCEL	10.96989
RAB11FIP1	13.91565
CYSRT1	13.28588
S100A16	13.22095
TACSTD2	50.9853
MAL2	57.41673
PITX1	27.18976
KLK10	16.85305

Figure 12. Pathway analysis of influential genes with importance scores

The dot plot presented in Fig. 13 displays the expression profile of identified influential pathways (Fig. 12). Lytic cells undergoing host shutoff were characterized by both a downregulation and decreased abundance of host genes belonging to keratinization, neutrophil degranulation, and cytokine signaling pathways in addition to a downregulation but comparable abundance of the viral mRNA translation pathway.

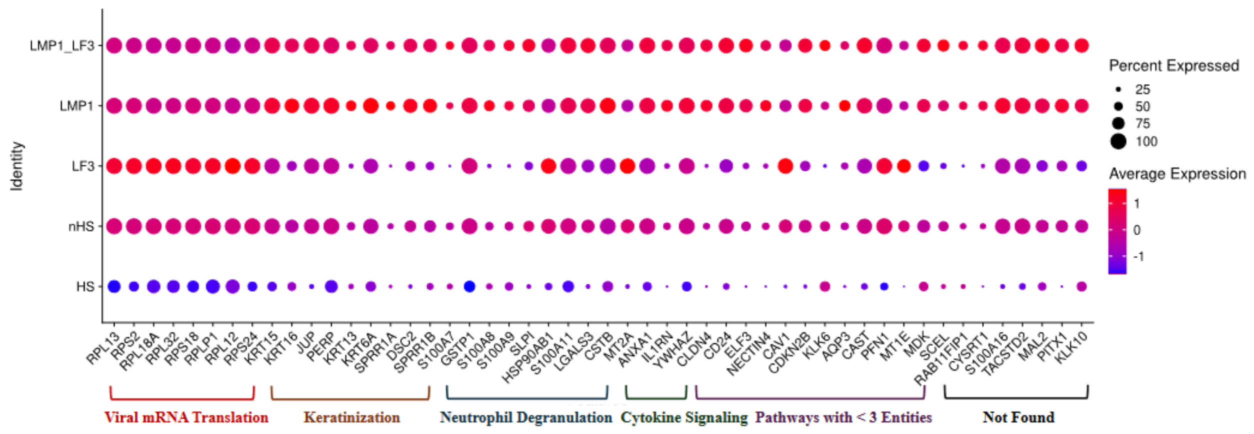

Figure 13. Dot plot of influential pathways of all infection states

3.3 Multinomial Logistic Regression Modeling

3.3.1 Multinomial Logistic Regression Assumptions

The tables presented in Fig. 14 display VIF metrics from the multinomial logistic regression model with iterative AIC reduction based variable selection. The VIFs measure multicollinearity among predictor variables for each of the five EBV infection classifications.

Variance Inflation Factor - Multinomial Logistic Regression AIC

Proportion EBV UMI							
0.37		0.08		0.12		0.2	
Gene	VIF	Gene	VIF	Genes	VIF	Genes	VIF
MAL2	178.8008	CD24	12.30758	MAL2	14.25656	CD24	44.30813
CLDN4	121.5907	MAL2	9.532307	CLDN4	7.018979	MAL2	33.64846
TACSTD2	308.2492	CLDN4	4.43511	CD24	19.07352	CLDN4	17.7081
CD24	313.658	TACSTD2	22.0784	TACSTD2	32.85541	TACSTD2	63.81294
ELF3	115.552	ELF3	5.802699	ELF3	8.542363	ELF3	18.80182
S100A9	113.4903	MT2A	17.05919	S100A9	6.666901	S100A9	17.99887
MT2A	494.7283	KRT15	22.60199	KRT16	12.32568	MT2A	49.4106
PITX1	140.8052	S100A9	4.782769	KRT15	32.9648	KRT16	24.19165
KRT13	75.50219	KRT16	8.427786	MT2A	22.00865	CSTB	62.97505
CSTB	259.5024	PITX1	4.823701	PITX1	7.177241	CAV1	45.81189
KRT16	154.8858	CSTB	24.93538	KRT13	4.875467	KRT15	62.1836
CAV1	262.4517	CAV1	17.2875	IL1RN	5.830894	IL1RN	12.1491
KRT15	299.3498	KRT13	3.377745	CSTB	31.73975	KRT13	14.91353
LGALS3	175.0134	JUP	17.06359	KLK10	11.07507	KLK10	25.71368
CDKN2B	109.7493	IL1RN	3.891209	KRT6A	19.41581	PITX1	20.04855
IL1RN	85.26432	LGALS3	7.212126	ANXA1	47.20069	SLPI	19.9254
KLK10	137.852	KRT6A	13.67992	JUP	22.74084	RPL12	99.55335
JUP	231.7729	KLK10	7.160552	CAV1	21.98149	JUP	42.60141
RPL12	289.5157	SLPI	5.785015	LGALS3	10.65643	SPRR1A	11.02541
KLK6	166.1318			SLPI	9.059008	SCEL	14.9553

Figure 14. VIF summary

3.3.2 Multinomial Logistic Regression with AIC

The tables presented in Fig. 15 are confusion matrices derived from the multinomial logistic regression model with iterative AIC reduction based variable elimination (see Fig. 10 for more details).

Multinomial Logistic Regression - AIC

Proportion EBV UMI = 0.37		Predicted				
		Δ-Lytic			Lytic	
		LMP1	LF3	LMP1+LF3	HS	nHS
Actual	LMP1	5034	787	79	2	2
	LF3	787	3695	8	2	0
	LMP1+LF3	1578	388	121	1	0
	HS	2	3	0	39	0
	nHS	127	120	6	2	2

Proportion EBV UMI = 0.20		Predicted				
		Δ-Lytic			Lytic	
		LMP1	LF3	LMP1+LF3	HS	nHS
Actual	LMP1	5069	1121	70	2	0
	LF3	809	3677	3	3	0
	LMP1+LF3	1578	378	132	0	0
	HS	15	23	2	39	0
	nHS	106	108	5	3	0

Proportion EBV UMI = 0.12		Predicted				
		Δ-Lytic			Lytic	
		LMP1	LF3	LMP1+LF3	HS	nHS
Actual	LMP1	5058	1124	76	4	0
	LF3	801	3684	5	2	0
	LMP1+LF3	1586	374	127	1	0
	HS	4	2	1	41	2
	nHS	65	83	6	2	0

Proportion EBV UMI = 0.08		Predicted				
		Δ-Lytic			Lytic	
		LMP1	LF3	LMP1+LF3	HS	nHS
Actual	LMP1	5070	1118	69	5	0
	LF3	806	3681	3	2	0
	LMP1+LF3	1575	381	130	2	0
	HS	82	70	5	42	0
	nHS	39	60	3	1	0

Figure 15. Multinomial logistic regression with AIC summary

The tables presented in Fig. 16 display sensitivity and specificity metrics from the multinomial logistic regression model with iterative AIC reduction based variable elimination (see Fig. 11 for more details).

Multinomial Logistic Regression - AIC

Proportion EBV UMI = 0.37	**Δ-Lytic**			**Lytic**	
	LMP1	**LF3**	**LMP1+LF3**	**HS**	**nHS**
Sensitivity	5.786207	0.822573	0.061515	0.886364	0.007782
Specificity	0.637553	0.843482	0.991306	0.999451	0.99984

Proportion EBV UMI = 0.20	**Δ-Lytic**			**Lytic**	
	LMP1	**LF3**	**LMP1+LF3**	**HS**	**nHS**
Sensitivity	0.809486	0.818566	0.063218	0.493671	0
Specificity	0.635518	0.811582	0.992763	0.999388	1

Proportion EBV UMI = 0.12	**Δ-Lytic**			**Lytic**	
	LMP1	**LF3**	**LMP1+LF3**	**HS**	**nHS**
Sensitivity	0.807729	0.820125	0.060824	0.82	0
Specificity	0.638078	0.814984	0.991971	0.999308	0.999845

Proportion EBV UMI = 0.08	**Δ-Lytic**			**Lytic**	
	LMP1	**LF3**	**LMP1+LF3**	**HS**	**nHS**
Sensitivity	0.809645	0.819457	0.062261	0.211055	0
Specificity	0.636443	0.81172	0.992764	0.999228	1

Figure 16. Multinomial logistic regression with AIC sensitivity and specificity

3.3.3 Multinomial Logistic Regression with Elastic Net

The tables presented in Fig. 17 are confusion matrices derived from the multinomial logistic regression model with an Elastic Net penalty (see Fig. 10 for more details).

Multinomial Logistic Regression - Elastic Net

Proportion EBV UMI = 0.37		**Predicted**				
		Δ-Lytic			**Lytic**	
		LMP1	**LF3**	**LMP1+LF3**	**HS**	**nHS**
Actual	**LMP1**	**5047**	1137	75	3	0
	LF3	799	**3686**	5	2	0
	LMP1+LF3	1567	388	**133**	0	0
	HS	4	5	0	**35**	0
	nHS	124	119	9	2	**3**

Proportion EBV UMI = 0.20		**Predicted**				
		Δ-Lytic			**Lytic**	
		LMP1	**LF3**	**LMP1+LF3**	**HS**	**nHS**
Actual	**LMP1**	**5061**	1125	1585	2	0
	LF3	795	**3691**	373	2	0
	LMP1+LF3	1585	373	**130**	0	0
	HS	16	22	2	**39**	0
	nHS	103	109	7	3	**0**

Proportion EBV UMI = 0.12		**Predicted**				
		Δ-Lytic			**Lytic**	
		LMP1	**LF3**	**LMP1+LF3**	**HS**	**nHS**
Actual	**LMP1**	**5058**	1123	78	3	0
	LF3	781	**3704**	5	48	0
	LMP1+LF3	1597	376	**115**	0	0
	HS	3	2	0	**40**	2
	nHS	64	84	6	2	**0**

Proportion EBV UMI = 0.08		**Predicted**				
		Δ-Lytic			**Lytic**	
		LMP1	**LF3**	**LMP1+LF3**	**HS**	**nHS**
Actual	**LMP1**	**5057**	1127	75	3	0
	LF3	790	**3694**	6	2	0
	LMP1+LF3	1586	378	**123**	1	0
	HS	81	73	6	**39**	0
	nHS	38	60	3	1	**0**

Figure 17. Multinomial logistic regression with Elastic Net summary

The tables presented in Fig. 18 display sensitivity and specificity metrics from the multinomial logistic regression model with an Elastic Net penalty (see Fig. 11 for more details).

Multinomial Logistic Regression - Elastic Net

Proportion EBV UMI = 0.37	Δ-Lytic			Lytic	
	LMP1	**LF3**	**LMP1+LF3**	**HS**	**nHS**
Sensitivity	0.805973	0.82057	0.063697	0.795455	0.011673
Specificity	0.637553	0.809386	0.991949	0.999466	1

Proportion EBV UMI = 0.20	Δ-Lytic			Lytic	
	LMP1	**LF3**	**LMP1+LF3**	**HS**	**nHS**
Sensitivity	0.6511	0.759309	0.062261	0.493671	0
Specificity	0.65531	0.839697	0.847932	0.999532	1

Proportion EBV UMI = 0.12	Δ-Lytic			Lytic	
	LMP1	**LF3**	**LMP1+LF3**	**HS**	**nHS**
Sensitivity	0.807729	0.816219	0.055077	0.851064	0
Specificity	0.641968	0.814685	0.991911	0.995937	0.999845

Proportion EBV UMI = 0.08	Δ-Lytic			Lytic	
	LMP1	**LF3**	**LMP1+LF3**	**HS**	**nHS**
Sensitivity	0.807569	0.822351	0.058908	0.19598	0
Specificity	0.637407	0.810658	0.991859	0.999459	1

Figure 18. Multinomial logistic regression with Elastic Net sensitivity and specificity

3.4 Iterative Analysis

Random Forest model development of all infection classifications is largely driven by changes in host expression profile pertaining to lytic cells undergoing host shutoff (Fig. 13). As such, a random forest model was fitted removing cells undergoing host shutoff. The tables presented in Fig. 19 are confusion matrices and sensitivity and specificity metrics derived from the random forest model after removing host shutoff cells (see Fig. 10 and Fig. 11 for more details).

Random Forest

Proportion EBV UMI = 0.37		Predicted			
		Δ-Lytic			**Lytic**
		LMP1	**LF3**	**LMP1+LF3**	**nHS**
Actual	**LMP1**	**1548**	315	1	0
	LF3	287	**1064**	0	0
	LMP1+LF3	516	112	**1**	0
	nHS	46	36	0	**0**
Proportion EBV UMI = 0.37		**Δ-Lytic**			**Lytic**
		LMP1	**LF3**	**LMP1+LF3**	**nHS**
Sensitivity		0.830472	0.787565	0.00159	0
Specificity		0.588264	0.820194	0.999697	1

Figure 19. Random Forest model removing host shut off cells

The dot plot presented in Fig. 20 displays the expression profile of identified influential pathways (Fig. 12), once lytic cells undergoing host shutoff were removed from the random forest model (Fig. 19).

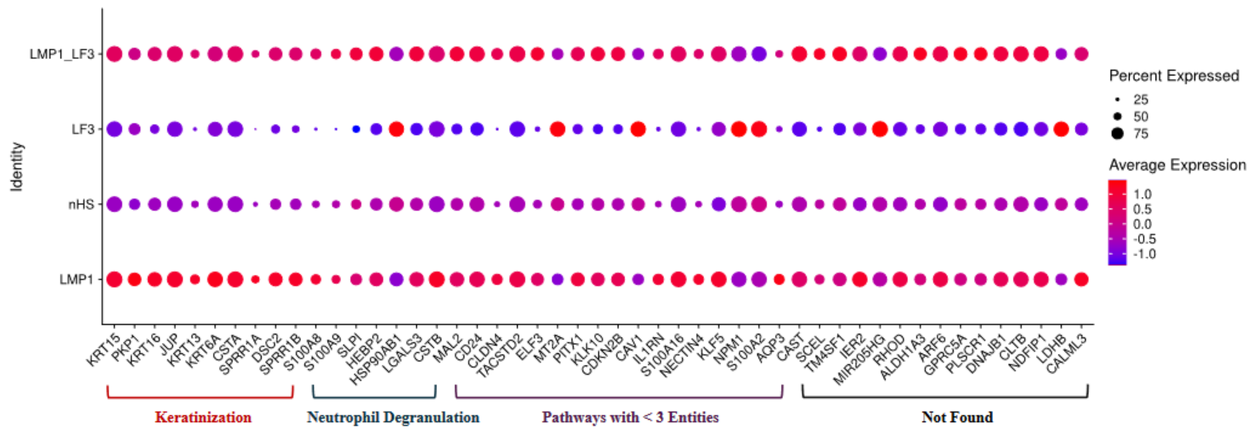

Figure 20. Dot plot of influential pathways of all non-host shutoff infection states

Lytic cells not undergoing host shutoff were characterized by a downregulation but comparable abundance of host genes belonging to keratinization and neutrophil degranulation pathways when compared to the LMP-1/BNLF2a associated Δ-lytic classification. Cells belonging to the LF3 dependent Δ-lytic classification show a similar keratinization host gene expression profile to lytic cells undergoing host shutoff in addition to a downregulation and decreased abundance of neutrophil degranulation host genes when compared to the LMP-1/BNLF2a associated Δ-lytic classification.

Random Forest model development after removing host shutoff cells is largely influenced by changes in host expression profile pertaining to lytic cells not undergoing host shutoff (Fig. 20). As such, a random forest model was fit removing lytic cells to better understand differences between Δ-lytic infection states. The tables presented in Fig. 19 are confusion matrices and sensitivity and specificity metrics derived from the random forest model after removing host shutoff cells (see Fig. 10 and Fig. 11 for more details).

Random Forest

Proportion EBV UMI = 0.37		**Predicted**		
		Δ-Lytic		
		LMP1	**LF3**	**LMP1+LF3**
Actual	**LMP1**	**1524**	335	23
	LF3	259	**1092**	0
	LMP1+LF3	483	107	**30**
Proportion EBV UMI = 0.37		**Δ-Lytic**		
		LMP1	**LF3**	**LMP1+LF3**
Sensitivity		0.809777	0.80829	0.048387
Specificity		0.623541	0.823341	0.992886

Figure 21. Random Forest model removing lytic cells

The dot plot presented in Fig. 22 displays the expression profile of identified influential pathways (Fig. 12), once lytic were removed from the random forest model (Fig. 21).

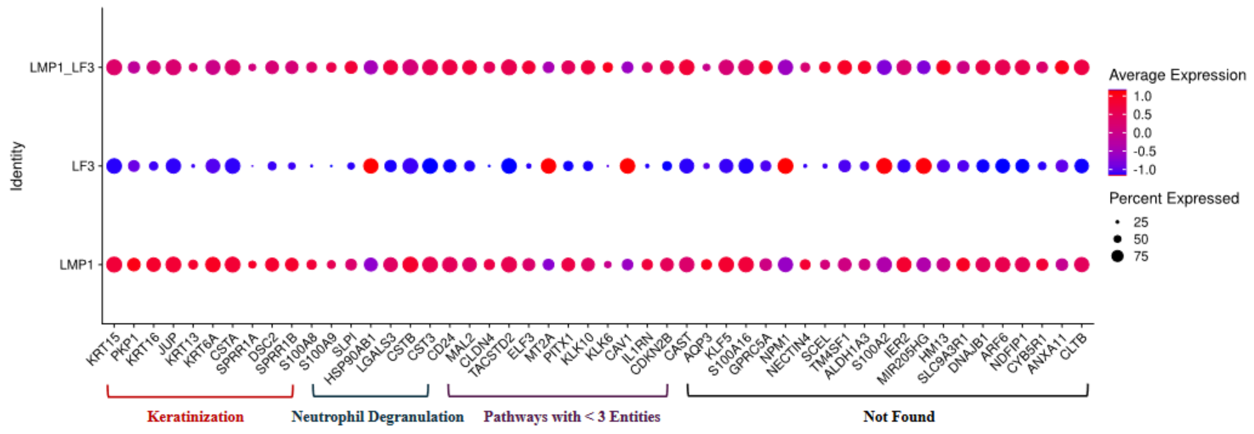

Figure 22. Dot plot of influential pathways of all Δ-lytic infection states

LF3 dependent Δ-lytic cells were characterized by a downregulation of host genes belonging to keratinization and neutrophil degranulation pathways when compared to LMP-1/BNLF2a and LMP1+LF3 associated Δ-lytic cells. Clear differentiation between LMP-1/BNLF2a and LMP1+LF3 associated Δ-lytic cells was unsuccessful.

Traditionally, scRNA-data is limited by the prevalence of false negatives which arise due to low abundance of EBV transcripts in most cells. The random forest model (Fig. 10) was used to predict the infection states of false negatives. 5046 of 18,476 false negative cells were classified as LMP-1/BNLF2a associated Δ-lytic cells, 13,428 were classified as LF3 associated Δ-lytic, and 2 were classified as lytic undergoing host shutoff with a EBV UMI ratio cutoff of 0.37 (Fig. 23).

Random Forest

	Predicted				
	Δ-Lytic			Lytic	
Proportion EBV UMI = 0.37	LMP1	LF3	LMP1+LF3	HS	nHS
	5046	13428	0	2	0

Figure 23. Predicting false negatives from the Random Forest model

UMAPs displaying both predicted (Fig. 23) and known infection states (Fig. 24A) and cell cycle phases (Fig. 24B) show unique characteristics of LMP-1/BNLF2a and LF3 associated Δ-lytic cell classifications.

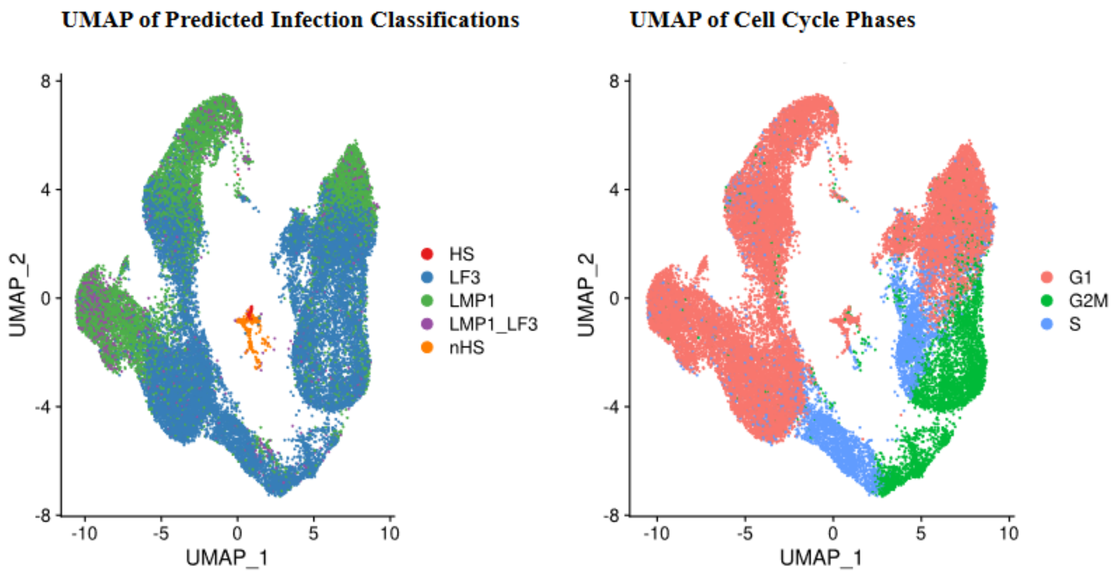

Figure 24A and 24B. UMAP of predicted infection states and cell cycle phases

LMP-1/BNLF2a associated Δ-lytic cells cluster together on the UMAP space and belong exclusively to the G1 phase, LF3 associated Δ-lytic cells cluster together and belong to both the G1 and G2M phase, and LMP-1/BNLF2a + LF3 associated Δ-lytic cells are scattered throughout the UMAP space with greater overlap with the LMP-1/BNLF2a associated Δ-lytic classification when compared to the LF3 associated Δ-lytic classification.

4.0 Discussion and Conclusions

4.1 Discussion

Early stages of EBV infection can reveal the viral impact of EBV on host cell environments. During this phase, EBV undergoes either lytic replication or latent integration into the host genome dictating the course of EBV pathogenesis [1]. Understanding the host effects contributing to this mechanism may inform potential translational targets for EBV-associated NPC in the form of diagnostic and therapeutic strategies.

Fig. 1A and 1B show cells cluster according to timepoint post-infection rather than infection type, suggesting that cell cycle effects are primary drivers of gene expression profiles. To highlight transcriptional effects rather than cell cycle effects all samples were compiled into one comprehensive dataset. Cluster 11 represents a distinct profile, not overlapping with the standard day 0, 2, or 4 classifications as the other clusters do, suggesting that cluster 11 may be indicative of a cellular state unique to EBV infection or response. Fig. 2 further showcases the unique heterogeneity exclusive to cluster 11, containing proportional quantities of cells from all six samples when compared to other clusters.

The pronounced EBV gene expression in cluster 11, as demonstrated in Fig. 3 and 4, strongly suggests that this cluster is driven by EBV lytic gene effects. The comprehensive expression of the lytic cascade in cluster 11 defines this cluster as a lytic group of cells in a state of active viral replication or reactivation. This contrasts with other clusters where the EBV gene expression is limited to specific LMP-1 annotations and two unassigned gene annotations, indicative of a Δ-lytic state defined as either latent or some transitory infection state. LMP-1 annotations include LMP-1, LMP-1/BNLF2b, and LMP-1/BNLF2a where BNLF2b and BNLF2a cannot be entirely separated from LMP-1 due to overlapping regions within the EBV genome, in the same orientation. Unassigned annotations include LF3 and DESERT, where DESERT represents regions of the EBV genome that were not successfully annotated.

In Δ-lytic clusters, cells exhibiting expression of LMP-1 and LMP-1/BNLF2b were found to be a subset of those expressing LMP-1/BNLF2a. The Δ-lytic expression pattern outlined in Fig. 6 allowed the classification of EBV-infected Δ-lytic cells into cells expressing LF3, LMP-1/BNLF2a, or both LMP-1/BNLF2a and LF3. This analysis establishes three categories of EBV Δ-lytic infection status, either LMP-1/BNLF2a associated Δ-lytic, LF3 associated Δ-lytic, or LMP-1/BNLF2a+LF3 associated Δ-lytic infection.

BGLF5 is an early lytic categorized EBV gene which plays a role in host shutoff during the lytic phase by degrading mRNA, thus reducing host protein synthesis and enhancing viral replication [20]. The distinct expression of BGLF5 within cluster 11, as highlighted by Fig. 7, provides evidence of host shutoff events in some of these cells. The lack of BGLF5 expression in Δ-lytic clusters underscores the unique state of cluster 11 and its potential role in categorizing EBV pathogenesis. This is further supported by Fig. 8 which reveals a distinct population of cells characterized by a higher proportion of EBV gene UMIs compared to all gene UMIs.

This group of cells is confirmed by Fig. 9. The distinct population of cells within cluster 11, characterized by both a high proportion of EBV gene UMIs and elevated BGLF5 expression, substantiates the hypothesis that these cells are in an advanced stage of the lytic cycle where host shutoff occurs. The Δ-lytic group exhibited proportions ranging from 0 to 0.37, while the lytic group's proportions spanned from 0.003 to 0.87. This analysis allowed for two categorizations within the lytic cluster, lytic cells with a proportion greater than 0.37 that were undergoing host shutoff (HS) and lytic cells with a proportion less than or equal to 0.37 which were expressing viral genes, but not undergoing host shutoff (nHS). As the proportion of EBV UMIs to all gene UMIs overlaps from 0.03 to 0.37 between lytic and Δ-lytic clusters, Q1, Q2, and Q3 (0.08, 0.12, 0.20) were also tested. Ultimately, five novel classifications of EBV infection status were established: LF3 associated Δ-lytic, LMP-1/BNLF2a associated Δ-lytic, LMP-1/BNLF2a+LF3 associated Δ-lytic, lytic cells undergoing host shutoff, and lytic cells not undergoing host shutoff.

To determine predictors for model fitting among the five EBV infection statuses, a non-parametric Wilcoxon rank sum test identified genes with significant differences in gene expression, helping to highlight potential markers indicative of each infection status and guide the development of predictive models for EBV infection classifications.

A Random Forest algorithm was utilized to classify cells based on their EBV infection status, leveraging the significant genes identified as predictors. The confusion matrices in Fig. 10 and sensitivity and specificity calculations in Fig. 11 demonstrate the model's performance across various EBV UMI proportion thresholds, illustrating the predictive accuracy for each EBV infection classification. Based on Fig. 11 Sensitivity did not change for LF3 associated Δ-lytic, LMP-1/BNLF2a associated Δ-lytic, LMP-1/BNLF2a+LF3 associated Δ-lytic, and lytic cells not undergoing host shutoff classifications. Sensitivity for the lytic cells undergoing host shutoff classification increased from 0.21 to 0.82 from an increase in EBV UMI proportion of 0.08 to 0.37. Low sensitivities for LMP-1/BNLF2a+LF3 associated Δ-lytic and lytic cells undergoing host shutoff, and incorrect classification of these categories to LMP-1/BNLF2a and LF3 associated Δ-lytic classifications across all EBV UMI proportions indicates the model is unable to accurately predict LMP-1/BNLF2a+LF3 associated Δ-lytic and lytic cells undergoing host shutoff classifications. Specificity remained consistent across all classifications for all proportions.

The random forest model ranked the predictors by an importance score and the top 50 ranked genes were analyzed using Reactome pathway analysis [21]. From Fig. 12, the most influential genes from the Random Forest model belong to viral mRNA translation, keratinization, neutrophil degranulation, and cytokine signaling pathways. Lytic cells undergoing host shutoff were characterized by both a downregulation and decreased abundance of host genes belonging to keratinization, neutrophil degranulation, and cytokine signaling pathways in addition to a downregulation but comparable abundance of the viral mRNA translation pathway (Fig. 13).

The top 25 genes from the Random Forest model by importance score were utilized as candidate predictors for a multinomial logistic regression model. The impact of removing any of the 25 genes on the model's AIC was observed and variables were iteratively removed until no further reduction in AIC was observed upon variable exclusion. Fig. 14 indicates that an increase in EBV UMI proportion corresponds to an increase in the degree of multicollinearity between predicter variables. However, based on Fig. 16 while specificity remains consistent across varying EBV UMI proportions, sensitivity for lytic cells undergoing host shutoff increased from 0.21 to 0.82 from an increase in proportion of 0.08 to 0.37. Similar to the Random Forest model, this method failed to successfully identify lytic cells not undergoing host shutoff and LMP1/BNLF2a+LF3 associated Δ-Lytic cells for any EBV UMI proportion cutoff.

To account for multicollinearity between predictor variables a multinomial logistic regression model with an Elastic Net penalty was utilized to combine effects of Lasso and Ridge penalties [17]. Lasso encourages the sum of the absolute values of the regression coefficients to be sufficiently small, effectively shrinking some coefficients to zero, thus aiding in variable selection. Ridge encourages the sum of the squares of the coefficients to be sufficiently small, which does not set coefficients to zero but rather shrinks them towards zero. This aids in addressing multicollinearity by ensuring that the model coefficients are not overly sensitive to changes in the model. Elastic Net aims to combine these two properties to improve both variable selection and address multicollinearity concerns, resulting in a model that is robust against overfitting. When comparing this model (Fig. 15 and Fig. 16) to the Random Forest model, the sensitivity for an EBV UMI proportion of 0.2 decreased from 0.83 to 0.65. Similar to the previous models, the multinomial logistic regression model with an Elastic Net penalty was unable to accurately predict lytic cells not undergoing host shutoff and LMP1/BNLF2a+LF3 associated Δ-Lytic cells for any EBV UMI proportion cutoff.

No one model seems to substantially outperform the other models in predicting infection status. Fig. 11, 16, and 18 indicate regardless of chosen model, lytic cells not undergoing host shutoff and LMP1/BNLF2+LF3 associated Δ-Lytic cells were unable to be accurately distinguished from the LMP-1/BNLF2a and LF3 associated Δ-lytic classifications. This is likely due to significant similarities in gene expression profiles between these classifications. Fig. 13 indicates that lytic cells undergoing host shutoff were characterized by dramatic changes to host expression profiles and downregulation of identified host gene pathways.

To elucidate differences between other categories, a random forest model was fit after omitting lytic cells undergoing host shutoff (Fig. 19). While sensitivity and specificity remained consistent with previous models containing all five classifications, a different expression profile of influential genes was observed (Fig. 20). Lytic cells not undergoing host shutoff were characterized by a downregulation but comparable abundance of host genes belonging to keratinization and neutrophil degranulation pathways when compared to the LMP-1/BNLF2a associated Δ-lytic classification. Cells belonging to the LF3 dependent Δ-lytic classification show a similar keratinization host gene expression profile to lytic cells undergoing host shutoff in addition to a downregulation and decreased abundance of neutrophil degranulation host genes when compared to the LMP-1/BNLF2a associated Δ-lytic classification. However, cells belonging to the LMP-1/BNLF2a+LF3 associated Δ-lytic classification showed a nearly identical expression profile to the LMP-1/BNLF2a associated Δ-lytic classification. This analysis successfully differentiated the lytic cells not undergoing host shutoff classification from the other three Δ-lytic classifications.

To differentiate between the remaining three Δ-lytic classifications, a random forest model was fit after omitting all lytic cells (Fig. 21). While sensitivity and specificity remained consistent with all previous models and LF3 associated Δ-lytic cells were generally characterized by a downregulation of host genes belonging to keratinization and neutrophil degranulation pathways, some unique host genes emerged. LF3 associated Δ-lytic cells in contrast to LMP-1/BNLF2a and LMP-1/BNLF2a+LF3 associated Δ-lytic cells observed specific upregulation of HSP90AB1, MT2A, CAV1, NPM1, S100A2, and MIR205HG genes. Expression profiles of LMP-1/BNLF2a and LMP-1/BNLF2a+LF3 associated Δ-lytic cells remained nearly identical, suggesting that the LMP-1 effects overshadow the effects of LF3 in cells which contain both transcripts.

4.2 Limitations

4.2.1 Single Cell Limitations

Limitations of the single-cell methods used may include potential bias introduced during cell collection, sensitivity to detect lowly expressed genes, and the challenge of fully capturing transient states of EBV infection. Additionally, the analysis relies on existing annotations, which might not fully represent all EBV strains or capture certain genes due to overlapping regions. The annotation used was customized for a 5’ single cell library and is the Akata reference EBV strain used in the experimented cell lines. These results may not be conserved under similar methods applied to other epithelial or B-cell cell lines. The reliance on computational algorithms for clustering and differential expression analysis might also introduce variability, dependent on parameter settings and the statistical models applied.

The normalization process assumes that cells have roughly equal total RNA content, which justifies scaling gene expression measurements by a constant factor across all cells. However, this might not account for natural variations in RNA content between different cell types or states, potentially leading to overestimation or underestimation of gene expression levels in cells with unusually high or low RNA content that make it past the filtering steps. This simplification can affect the interpretation of gene expression differences across conditions or cell types.

Variable feature selection in the Seurat workflow utilizes LOESS which introduces two assumptions [8]. The first being the mean of y around point x can be approximated through a class of parametric functions based on polynomial regression. The approximation of the mean response as a parametric function might not capture complex biological interactions accurately, potentially oversimplifying underlying patterns. The second assumption posits that the errors in estimating y are independently and randomly distributed with mean 0. The assumption of independent and identically distributed errors assumes homoscedasticity, which may not hold for biological data where technical and biological variances can introduce heteroscedasticity, impacting the reliability of predictions.

4.2.2 Random Forest Limitations

The random forest model assumes the errors across individual trees are uncorrelated and the ensemble of trees will provide superior prediction accuracy compared to any single tree within the forest [13]. While averaging the results across a multitude of trees reduces the risk of overfitting, the assumption that errors across trees are uncorrelated may not hold if the data has inherent correlations unable to be addressed by random sampling. While the ensemble generally outperforms individual trees, the improvement in prediction accuracy isn't guaranteed if the model overfits the training data.

4.2.3 Multinomial Logistic Regression Limitations

The multinomial logistic regression model introduces five assumptions, three of which introduce potential limitations to this methodology [15]. The first is independence of irrelevant alternatives. This assumption implies that the choice between outcomes is unaffected by the presence or absence of additional choices. In biological contexts, this may not hold due to complex interactions where the presence of one outcome could influence the likelihood of another. The second is linearity of independent variables and log-odds. Biological processes often exhibit non-linear dynamics, making this assumption too simplistic, potentially leading to inaccurate estimations of effect sizes. The third is no perfect multicollinearity. While the VIF metrics help identify multicollinearity and the Elastic Net penalty helps to mitigate multicollinearity among predictors, high-dimensional data such as gene expression profiles often exhibit multicollinearity due to biological pathways involving multiple co-expressed genes. This can make it difficult to discern the individual contribution of predictors.

4.3 Future Directions

Future directions should focus on applying this methodology to publicly available NPC tumor datasets and sequenced B-cell lines to validate and extend these findings. Other methodologies such as SLIDE: Significant Latent Factor Interaction Discover and Exploration may also be considered to identify latent factors contributing to the underlying pathology of EBV infection [28]. Such tools are optimized for high-dimensional omics datasets without assumptions pertaining to mechanisms of data generation. A combination of approaches is necessary to identify conserved host markers and influential pathways in the landscape of modern multi-omic datasets.

Ultimately, both in vivo and in vitro studies would be needed to extend these findings and further elucidate the mechanisms behind EBV's impact on host cell environments and its role in NPC pathogenesis. Identification of conserved host markers of EBV infection between a combination of computational approaches and experimental validation would enhance the field's ability to better understand EBV viral-host interactions and contribute to improving EBV associated disease outcomes. In the short term such markers may play a role in the development of EBV associated NPC screening protocols in at risk populations. If further validated, these pathways and markers may assist in creating the groundwork for the development of targeted therapeutic strategies, decreasing the disease burden of EBV associated disease.

Appendix A Appendices and Supplemental Content

Appendix A.1 R Script

```
#0: Environment:
{
 ##0.1: Working Directory:
 {
  getwd()
  setwd("/bgfs/kshair/shared/kshair_jap282/Single-Cell_NPC/Seurat/")

 }

 ##0.2: Packages:
 {
  library(Seurat)
  library(SeuratWrappers)
  library(SeuratObject)
  library(DoubletFinder)
  library(infercnv)
  library(tidyverse)
  library(ggplot2)
  library(ggpubr)
  library(reshape2)
  library(SingleR)
  library(celldex)
  library(pROC)
  library(EnhancedVolcano)
  library(monocle3)
  library(harmony)
  library(patchwork)
  library(pheatmap)
  library(ComplexHeatmap)
  library(presto)
  library(tictoc)
  library(corrplot)
  library(openxlsx)
  library(UpSetR)
  library(writexl)
  library(Matrix)
  library(irlba)
```

```
            library(RColorBrewer)
            library(circlize)
            library(randomForest)
            library(ranger)
            library(nnet)
            library(caret)
            library(MASS)
            library(glmnet)
            library(car)
          }

          ##0.3: Markers:
          {

              EBV <- c("EBNA-1", "EBNA-1/EBNA-3B/EBNA-3C", "EBNA-1/EBNA-
3B/EBNA-3C/EBNA-3A",
                       "EBNA-1/EBNA-LP/EBNA-3B/EBNA-3C/EBNA-3A", "EBNA-3A",
"LMP-1", "LMP-1/BNLF2b",
                       "EBNA-3B/EBNA-3C", "LMP-1/BNLF2a", "EBNA-2", "EBNA-
2/EBNA-LP",
                       "EBNA-LP", "LMP-2A", "LMP-2B", "LMP-2A/LMP-2B", "LMP-
2A/LMP-2B/BNRF1",
                       "RPMS1","BRLF1/BZLF1", "BRLF1","BALF1", "BALF2", "BALF5",
"BARF1", "BaRF1",
                       "BBLF2/BBLF3", "BBLF4", "BcRF1", "BDLF4", "BDLF4/BDLF3.5",
"BFLF1", "BFLF2",
                       "BFRF1", "BFRF1/BFRF1A", "BFRF2", "BFRF2/BFRF1",
"BFRF2/BFRF3", "BGLF4",
                       "BGLF4/BGLF5", "BGLF4/BGLF3.5", "BGLF5", "BHRF1", "BKRF4",
"BLLF2/BLLF1",
                       "BLLF3", "BMRF1", "BMRF2", "BORF2", "BRRF1", "BSLF1",
"BSLF2/BMLF1",
                       "BSLF2/BMLF1/BSLF1", "BVRF1", "BXLF1", "BZLF2","BALF4",
"BALF4/BALF3", "BBLF1",
                       "BBLF1/BGLF5", "BBRF1", "BBRF2", "BBRF2/BBRF1", "BBRF3",
"BcLF1", "BCRF1",
                       "BDLF1", "BDLF2", "BDLF3", "BDLF3.5", "BFRF3", "BGLF1",
"BGLF1/BGLF2",
                       "BGLF1/BDLF4", "BGLF2", "BILF2", "BKRF2", "BLLF1", "BLRF1",
"BLRF2", "BNRF1",
                       "BORF1", "BRRF2", "BSRF1", "BTRF1", "BTRF1/BcRF1", "BVLF1",
"BVRF2", "BVRF2/BdRF1",
                       "BXLF2", "BXRF1", "BXRF1/BVRF1","BALF3", "BFRF1A",
"BGLF3", "BGLF3.5",
                       "BGLF3.5/BGLF3", "BGRF1/BDRF1", "BILF1", "BKRF3",
"BKRF3/BKRF2", "BOLF1",
                              "BPLF1", "BPLF1/BOLF1", "LF1", "LF2", "LF2/LF1", "LF3",
"HALMP1", "DESERT", "LMP-2A", "LMP-2B")


          }

          ##0.4: Cell-cycle Markers:
          {
            s.genes <- cc.genes$s.genes
            g2m.genes <- cc.genes$g2m.genes

          }
        }

        #1: Data
        {
          #Shair_Ch1
          {
            Shair_05_0_df <- Read10X(data.dir = "/bgfs/kshair/shared/kshair_alb635/Single-
Cell_NPC/Seurat/Shair/Shair_05_Akata-Collapse/Ch1-ALI-
4/sample_filtered_feature_bc_matrix") #There was confusion in labeling
            Shair_05_0_df <- CreateSeuratObject(counts = Shair_05_0_df$`Gene Expression`,
project = "Ch1_0", min.cells = 3, min.features = 200)

            Shair_05_2_df <- Read10X(data.dir = "/bgfs/kshair/shared/kshair_alb635/Single-
Cell_NPC/Seurat/Shair/Shair_05_Akata-Collapse/Ch1-ALI-
0/sample_filtered_feature_bc_matrix") #There was confusion in labeling
            Shair_05_2_df <- CreateSeuratObject(counts = Shair_05_2_df$`Gene Expression`,
project = "Ch1_2", min.cells = 3, min.features = 200)

            Shair_05_4_df <- Read10X(data.dir = "/bgfs/kshair/shared/kshair_alb635/Single-
Cell_NPC/Seurat/Shair/Shair_05_Akata-Collapse/Ch1-ALI-
2/sample_filtered_feature_bc_matrix") #There was confusion in labeling
            Shair_05_4_df <- CreateSeuratObject(counts = Shair_05_4_df$`Gene Expression`,
project = "Ch1_4", min.cells = 3, min.features = 200)
          }
          #Shair_IRES
          {
            Shair_06_0_df <- Read10X(data.dir = "/bgfs/kshair/shared/kshair_alb635/Single-
Cell_NPC/Seurat/Shair/Shair_06_Akata-Collapse/IRES-ALI-
0/sample_filtered_feature_bc_matrix") #There was confusion in labeling
            Shair_06_0_df <- CreateSeuratObject(counts = Shair_06_0_df$`Gene Expression`,
project = "IRES_0", min.cells = 3, min.features = 200)
```

```
              Shair_06_2_df <- Read10X(data.dir = "/bgfs/kshair/shared/kshair_alb635/Single-
Cell_NPC/Seurat/Shair/Shair_06_Akata-Collapse/IRES-ALI-
2/sample_filtered_feature_bc_matrix") #There was confusion in labeling
              Shair_06_2_df <- CreateSeuratObject(counts = Shair_06_2_df$`Gene Expression`,
project = "IRES_2", min.cells = 3, min.features = 200)

              Shair_06_4_df <- Read10X(data.dir = "/bgfs/kshair/shared/kshair_alb635/Single-
Cell_NPC/Seurat/Shair/Shair_06_Akata-Collapse/IRES-ALI-
4/sample_filtered_feature_bc_matrix") #There was confusion in labeling
              Shair_06_4_df <- CreateSeuratObject(counts = Shair_06_4_df$`Gene Expression`,
project = "IRES_4", min.cells = 3, min.features = 200)
            }
          }

        #2: Merge Data
        {
          #Shair_CL:
          {
            Shair_CL_df <-  merge(x = Shair_05_0_df,
                            y = c(Shair_05_2_df, Shair_05_4_df,
                                 Shair_06_0_df, Shair_06_2_df, Shair_06_4_df),
                            add.cell.ids = c("Ch1_0", "Ch1_2", "Ch1_4",
                                         "IRES_0", "IRES_2", "IRES_4"),
                            project = "Shair")
            gc()
          }
        }

        #3: Standard Processing
        {
          #Shair CL:
          {
          Shair_CL_df[["percent.mt"]] <- PercentageFeatureSet(Shair_CL_df, pattern = "^MT-
")
          Shair_CL_df <- subset(Shair_CL_df, subset = nFeature_RNA > 200 & nFeature_RNA
< 9000 & percent.mt < 20)
          Shair_CL_df <- NormalizeData(Shair_CL_df)
          Shair_CL_df <- FindVariableFeatures(Shair_CL_df, selection.method = "vst",
nfeatures = 3500)
          Shair_CL_df <- ScaleData(Shair_CL_df, features = rownames(Shair_CL_df))
          Shair_CL_df <- RunPCA(Shair_CL_df, features = VariableFeatures(object =
Shair_CL_df))
          Shair_CL_df <- RunUMAP(Shair_CL_df, dims = 1:30)
          Shair_CL_df <- FindNeighbors(Shair_CL_df, dims = 1:30)
          Shair_CL_df <- FindClusters(Shair_CL_df, resolution = 0.5)
```

```
    Shair_CL_df = CellCycleScoring(Shair_CL_df,
                                s.features = s.genes,
                                g2m.features = g2m.genes,
                                set.ident = F)

    Shair_CL_dfPlot <- DimPlot(Shair_CL_df,
                            reduction = "umap",
                            pt.size = 0.1,
                            raster = F,
                            label = T,
                            label.box = T) +
      NoLegend()

    Shair_CL_dfPlot
    }
  }

#4: Figures
{
  #UMAP
  {
    #Cluster
    {
      Idents(Shair_CL_df) <- Shair_CL_df$seurat_clusters
      Shair_CL_dfPlot <- DimPlot(Shair_CL_df,
                              reduction = "umap",
                              pt.size = 0.25,
                              raster = F,
                              label = T,
                              label.box = T) +
        NoLegend()

      Shair_CL_dfPlot

    }
    #Origin
    {
      Idents(Shair_CL_df) <- Shair_CL_df$orig.ident
      Shair_CL_dfPlot <- DimPlot(Shair_CL_df,
                              reduction = "umap",
                              pt.size = 0.25,
                              raster = F,
                              label = T,
                              label.box = T) +
        NoLegend()
      Idents(Shair_CL_df) <- Shair_CL_df$seurat_clusters
```

```
          Shair_CL_dfPlot
        }
        #Cell Cylce
        {
          DimPlot(Shair_CL_df, reduction = "umap", group.by = "Phase", label = FALSE,
pt.size = 0.05) +
            ggtitle("UMAP of Cell Cycle Phases")
        }
      }

      #Violin
      {
        Idents(Shair_CL_df) <- Shair_CL_df$seurat_clusters
        VlnPlot(Shair_CL_df, features = EBV, stack = T) + NoLegend()
        VlnPlot(object = Shair_CL_df, features = 'BGLF5', split.by = 'seurat_clusters')
      }

      #Barplot
      {
        data_to_plot <- Shair_CL_df@meta.data %>%
          group_by(seurat_clusters, orig.ident) %>%
          summarise(count = n()) %>%
          mutate(proportion = count / sum(count, na.rm = TRUE)) %>%
          ungroup() %>%
          arrange(seurat_clusters, orig.ident)
        data_to_plot_wide <- pivot_wider(data_to_plot, names_from = orig.ident, values_from
= proportion, values_fill = list(proportion = 0))
        ggplot(data_to_plot, aes(y = factor(seurat_clusters), x = proportion, fill = orig.ident)) +
          geom_bar(stat = "identity") +
          labs(y = "Cluster", x = "Proportion", fill = "orig Identity") +
          theme_minimal() +
          coord_flip()
      }

    #Heatmap
      {
      #EBV Heatmap
        {
        Shair_CL_df_EBV <- Shair_CL_df
        Shair_CL_df_EBV <- subset(Shair_CL_df, features = EBV)
        genes <- Shair_CL_df_EBV[EBV, ]
        keep.cells <- colnames(genes[, colSums(genes) != 0])
        Shair_CL_df_EBV <- Shair_CL_df_EBV[, keep.cells]
        mat <- Shair_CL_df_EBV[["RNA"]]@data %>% as.matrix()
        cluster_anno <- Shair_CL_df_EBV@meta.data$seurat_clusters
        quantile(mat, c(.01, .99))
```

```
            quantile(mat, c(.01, .999))
            quantile(mat, c(.01, .9999))
            col_fun = circlize::colorRamp2(c(0, 1, 2, 3),
c("papayawhip","#FF00FF","black","#FFFF00"))
            Heatmap(mat, name = "Normalized Expression",
                  column_split = factor(cluster_anno),
                  cluster_columns = TRUE,
                  show_column_dend = FALSE,
                  cluster_column_slices = FALSE,
                  column_title_gp = gpar(fontsize = 8),
                  column_gap = unit(0.5, "mm"),
                  cluster_rows = FALSE,
                  show_row_dend = FALSE,
                  col = col_fun,
                  row_names_gp = gpar(fontsize = 4),
                  column_title_rot = 45,
                  top_annotation = HeatmapAnnotation(foo = anno_block(gp = gpar(fill =
scales::hue_pal()(9)))),
                  show_column_names = FALSE,
                  use_raster = TRUE,
                  raster_quality = 10)


          }

          #LMP1 LF3 Heatmap
            {
            Shair_CL_df_n11 <- subset(Shair_CL_df, idents = "11", invert = TRUE)
            levels(Shair_CL_df_n11)
            latent <- c("LMP-1/BNLF2a","LF3")
            Shair_CL_df_n11_filter <- subset(Shair_CL_df_n11, features = latent)
            Shair_CL_df_n11_filter@meta.data$infect <- "delta_lytic"
            Idents(Shair_CL_df_n11_filter) <- Shair_CL_df_n11_filter$infect
            levels(Shair_CL_df_n11_filter)
            genes <- Shair_CL_df_n11_filter[latent, ]
            keep.cells <- colnames(genes[, colSums(genes) != 0])
            Shair_CL_df_n11_filter <- Shair_CL_df_n11_filter[, keep.cells]
            mat <- Shair_CL_df_n11_filter[["RNA"]]@data %>% as.matrix()
            cluster_anno <- Shair_CL_df_n11_filter@meta.data$infect
            quantile(mat, c(.01, .99))
            quantile(mat, c(.01, .999))
            quantile(mat, c(.01, .9999))
            col_fun = circlize::colorRamp2(c(0, 1, 2, 3),
c("papayawhip","#FF00FF","black","#FFFF00"))
            Heatmap(mat, name = "Normalized Expression",
                  column_split = factor(cluster_anno),
                  cluster_columns = TRUE,
                  show_column_dend = FALSE,
                  cluster_column_slices = FALSE,
                  column_title_gp = gpar(fontsize = 8),
                  column_gap = unit(0.5, "mm"),
                  cluster_rows = TRUE,
                  show_row_dend = FALSE,
                  col = col_fun,
                  row_names_gp = gpar(fontsize = 4),
                  column_title_rot = 45,
                  top_annotation = HeatmapAnnotation(foo = anno_block(gp = gpar(fill =
scales::hue_pal()(9)))),
                  show_column_names = FALSE,
                  use_raster = TRUE,
                  raster_quality = 5)
            }



          }

        #Dotplot
          {
          #Dotplot nCountRNA EBV and All
          {
            metadata_1 <- Shair_CL_df@meta.data
            data_1 <- data.frame(
              CellID = rownames(metadata_1),
              nCount_RNA = metadata_1$nCount_RNA
            )
            Shair_CL_df_filter <- subset(Shair_CL_df, features = EBV)
            metadata_2 <- Shair_CL_df_filter@meta.data
            data_2 <- data.frame(
              CellID = rownames(metadata_2),
              nCount_RNA = metadata_2$nCount_RNA
            )
            combined_data <- merge(data_1, data_2, by = "CellID")
            colnames(combined_data)
            colnames(combined_data)[colnames(combined_data) == "nCount_RNA.x"] <-
"nCount_RNA_All"
            colnames(combined_data)[colnames(combined_data) == "nCount_RNA.y"] <-
"nCount_RNA_EBV"
            colnames(combined_data)
            sorted_data <- arrange(combined_data, nCount_RNA_EBV, nCount_RNA_All)
            sorted_data <- sorted_data %>%
              mutate(Number_ID = row_number())
            long_data <- tidyr::pivot_longer(sorted_data, cols = c("nCount_RNA_All",
"nCount_RNA_EBV"), names_to = "ID", values_to = "nCount_RNA")

            ggplot(long_data, aes(x = Number_ID, y = nCount_RNA, color = ID)) +
              geom_point(alpha = 0.6, size = 0.5) +
              scale_color_manual(values = c("nCount_RNA_All" = "blue", "nCount_RNA_EBV"
= "red")) +
              theme_minimal(base_size = 14) +
              theme(axis.ticks.x = element_blank(),
                    legend.position = "right",
                    legend.key.size = unit(0.5, "cm"),
                    legend.title.align = 0.5,
                    legend.text = element_text(size = 12)) +
              labs(x = "Cells sorted by EBV gene nCountRNA", y = "nCount_RNA",
                   title = "nCount_RNA All Cells")

          }
        #Ratio Dotplot
        {
            sorted_data$Ratio <- sorted_data$nCount_RNA_EBV /
sorted_data$nCount_RNA_All
            ggplot(sorted_data, aes(x = Number_ID, y = Ratio)) +
              geom_point() +
              theme_minimal() +
              theme(axis.title.x = element_blank(),
                    axis.text.x = element_blank(),
                    axis.ticks.x = element_blank()) +
              labs(y = "nCount_RNA_EBV / nCount_RNA_All", title = "Dot Plot of Ratio
(EBV/All RNA Counts)")
            BGLF5_expression <- GetAssayData(Shair_CL_df, assay = "RNA", layer =
"data")["BGLF5", ]
            BGLF5_expression_df <- as.data.frame(BGLF5_expression)
            BGLF5_expression_df$CellID <- rownames(BGLF5_expression_df)
            sorted_data <- merge(sorted_data, BGLF5_expression_df, by = "CellID")
            ggplot(sorted_data, aes(x = Number_ID, y = Ratio, color = BGLF5_expression)) +
              geom_point(alpha = 0.6, size = 1.5) +
              scale_color_gradientn(colors = c("blue", "red", "yellow"), name =
"BGLF5\nExpression") +
              theme_minimal(base_size = 14) +
              theme(axis.ticks.x = element_blank(),
                    legend.position = "right",
                    legend.key.size = unit(0.5, "cm"),
                    legend.title.align = 0.5,
                    legend.text = element_text(size = 12)) +
              labs(x = "Cells sorted by EBV gene nCountRNA", y = "nCountRNA EBV/ALL",
                   title = "Dot Plot of Proportion (EBV/All RNA Counts) by BGLF5")
            }
             # Reactome Dotplot Iteration 1
             {
              top_50_1 <-
c("RPL13","RPS2","RPL18A","RPL32","RPS18","RPLP1","RPL12","RPS24",
"KRT15","KRT16", "JUP",

"PERP","KRT13","KRT6A","SPRR1A","DSC2","SPRR1B","S100A7","GSTP1","S100A8","S1
00A9","SLPI",

"HSP90AB1","S100A11","LGALS3","CSTB","MT2A","ANXA1","IL1RN","YWHAZ",
"CLDN4","CD24","ELF3",
                      "NECTIN4","CAV1","CDKN2B", "KLK6",
"AQP3","CAST","PFN1","MT1E","MDK","SCEL","RAB11FIP1",
                      "CYSRT1","S100A16","TACSTD2","MAL2","PITX1","KLK10"
              )

              DotPlot(Shair_CL_df_data, features = top_50_1, cols = c("blue", "red")) +
                theme(axis.text.x = element_text(angle = 45, hjust = 1))
             }
             # Reactome Dotplot Iteration 2
             {
              top_50_2 <-
c("KRT15","PKP1","KRT16","JUP","KRT13","KRT6A","CSTA","SPRR1A","DSC2","SPRR1
B","S100A8",

"S100A9","SLPI","HEBP2","HSP90AB1","LGALS3","CSTB","MAL2","CD24","CLDN4","T
ACSTD2","ELF3",
                      "MT2A","PITX1","KLK10","CDKN2B","KLK6
","CAV1","IL1RN","S100A16","NECTIN4","KLF5","NPM1",

"S100A2","AQP3","CAST","SCEL","TM4SF1","IER2","MIR205HG","RHOD","ALDH1A3","
ARF6","GPRC5A",
                      "PLSCR1","DNAJB1","CLTB","NDFIP1","LDHB","CALML3")

              Idents(Shair_CL_df_data) <- Shair_CL_df_data$class
              DotPlot(Shair_CL_df_data, features = top_50_2, cols = c("blue", "red")) +
                theme(axis.text.x = element_text(angle = 45, hjust = 1))
             }
             # Reactome Dotplot Iteration 3
             {
              top_50_3 <-
c("KRT15","PKP1","KRT16","JUP","KRT13","KRT6A","CSTA","SPRR1A","DSC2","SPRR1
B","S100A8",
"S100A9","SLPI","HSP90AB1","LGALS3","CSTB","CST3","CD24","MAL2","CLDN4","TAC
STD2","ELF3",

"MT2A","PITX1","KLK10","KLK6","CAV1","IL1RN","CDKN2B","CAST","AQP3","KLF5",
"S100A16",

"GPRC5A","NPM1","NECTIN4","SCEL","TM4SF1","ALDH1A3","S100A2","IER2","MIR205
HG","HM13",

"SLC9A3R1","DNAJB1","ARF6","NDFIP1","CYB5R1","ANXA11","CLTB")
            Idents(Shair_CL_df_data) <- Shair_CL_df_data$class
            DotPlot(Shair_CL_df_data, features = top_50_3, cols = c("blue", "red")) +
              theme(axis.text.x = element_text(angle = 45, hjust = 1))
          }
        }
      }

      #5: Analysis
      {
        #Calculate Quantiles
        {
        filtered_data <- sorted_data[sorted_data$Ratio >= 0.05, ]
        quartiles <- quantile(filtered_data$Ratio, probs = c(0.25, 0.5, 0.75, 1))
        print(quartiles)
      }

        #037
        {
      #Prepare Dataset
      {
      Shair_CL_df_LF3<-Shair_CL_df
      Shair_CL_df_LF3 <- subset(Shair_CL_df_LF3, idents = "11", invert = TRUE)
      cells_to_remove <- WhichCells(Shair_CL_df, expression = `LMP-1/BNLF2a` > 0)
      Shair_CL_df_LF3 <- subset(Shair_CL_df_LF3, cells = cells_to_remove, invert = TRUE)
      Shair_CL_df_LF3 <- subset(Shair_CL_df_LF3, subset = LF3 > 0)
      final_cell_ids <- colnames(Shair_CL_df_LF3)
      Shair_CL_df_LF3 <- subset(Shair_CL_df, cells = final_cell_ids)
      Shair_CL_df_LMP1 <- Shair_CL_df
      Shair_CL_df_LMP1 <- subset(Shair_CL_df_LMP1, idents = "11", invert = TRUE)
      cells_to_remove_LF3 <- WhichCells(Shair_CL_df_LMP1, expression = LF3 > 0)
      Shair_CL_df_LMP1 <- subset(Shair_CL_df_LMP1, cells = cells_to_remove_LF3, invert
= TRUE)
      Shair_CL_df_LMP1 <- subset(Shair_CL_df_LMP1, subset = `LMP-1/BNLF2a` > 0)
      final_cell_ids <- colnames(Shair_CL_df_LMP1)
      Shair_CL_df_LMP1 <- subset(Shair_CL_df, cells = final_cell_ids)
```

```
        Shair_CL_df_cluster11 <- subset(Shair_CL_df, idents = "11")
        metadata_cluster11 <- Shair_CL_df_cluster11@meta.data
        data_cluster11 <- data.frame(
          CellID = rownames(metadata_cluster11),
          nCount_RNA = metadata_cluster11$nCount_RNA)
        Shair_CL_df_cluster11_filter <- subset(Shair_CL_df_cluster11, features = EBV)
        metadata_cluster11_filter <- Shair_CL_df_cluster11_filter@meta.data
        data_cluster11_filter <- data.frame(
          CellID = rownames(metadata_cluster11_filter),
          nCount_RNA = metadata_cluster11_filter$nCount_RNA)
        combined_data_cluster11 <- merge(data_cluster11, data_cluster11_filter, by = "CellID")
        colnames(combined_data_cluster11)[colnames(combined_data_cluster11) ==
"nCount_RNA.x"] <- "nCount_RNA_All"
        colnames(combined_data_cluster11)[colnames(combined_data_cluster11) ==
"nCount_RNA.y"] <- "nCount_RNA_EBV"
        sorted_data_cluster11 <- combined_data_cluster11 %>%
          arrange(nCount_RNA_EBV, nCount_RNA_All) %>%
          mutate(Number_ID = row_number())
        sorted_data_cluster11 <- combined_data_cluster11 %>%
          arrange(nCount_RNA_EBV, nCount_RNA_All) %>%
          mutate(Number_ID = row_number(),
                 Ratio = nCount_RNA_EBV / nCount_RNA_All)
        sorted_data_cluster11_filtered <- sorted_data_cluster11 %>%
          filter(Ratio <= 0.37)
        cell_ids_to_keep <- sorted_data_cluster11_filtered$CellID
        Shair_CL_df_nHS <- subset(Shair_CL_df, cells = cell_ids_to_keep)
        sorted_data_cluster11_filtered <- sorted_data_cluster11 %>%
          filter(Ratio > 0.37)
        cell_ids_to_keep <- sorted_data_cluster11_filtered$CellID
        Shair_CL_df_HS <- subset(Shair_CL_df, cells = cell_ids_to_keep)
        Shair_CL_df_LMP1_LF3 <- Shair_CL_df
        Shair_CL_df_LMP1_LF3 <- subset(Shair_CL_df_LMP1_LF3, idents = "11", invert =
TRUE)
        Shair_CL_df_LMP1_LF3 <- subset(Shair_CL_df_LMP1_LF3, subset = `LMP-
1/BNLF2a` > 0 & LF3 > 0)
        final_cell_ids <- colnames(Shair_CL_df_LMP1_LF3)
        Shair_CL_df_LMP1_LF3 <- subset(Shair_CL_df, cells = final_cell_ids)
        Shair_CL_df_HS@meta.data$class <- "HS"
        Idents(Shair_CL_df_HS) <- Shair_CL_df_HS$class
        levels(Shair_CL_df_HS)
        Shair_CL_df_nHS@meta.data$class <- "nHS"
        Idents(Shair_CL_df_nHS) <- Shair_CL_df_nHS$class
        levels(Shair_CL_df_nHS)
        Shair_CL_df_LF3@meta.data$class <- "LF3"
        Idents(Shair_CL_df_LF3) <- Shair_CL_df_LF3$class
        levels(Shair_CL_df_LF3)
```

```
	Shair_CL_df_LMP1@meta.data$class <- "LMP1"
	Idents(Shair_CL_df_LMP1) <- Shair_CL_df_LMP1$class
	levels(Shair_CL_df_LMP1)
	Shair_CL_df_LMP1_LF3@meta.data$class <- "LMP1_LF3"

	levels(Shair_CL_df_LMP1_LF3)
	Shair_CL_df_data <- merge(x = Shair_CL_df_HS, y = c(Shair_CL_df_nHS,
Shair_CL_df_LF3, Shair_CL_df_LMP1, Shair_CL_df_LMP1_LF3), add.cell.ids = c("HS",
"nHS", "LF3", "LMP1", "LMP1_LF3"), project = "Combined")
	Idents(Shair_CL_df_data) <- Shair_CL_df_data$class
	levels(Shair_CL_df_data)
	cell_ids <- colnames(Shair_CL_df_data)
	class_labels <- Idents(Shair_CL_df_data)
	gene_expression_data <- GetAssayData(Shair_CL_df_data, layer = "data")
	gene_expression_df <- as.data.frame(t(gene_expression_data))
	cell_class_df <- data.frame(CellID = cell_ids, Class = as.character(class_labels))
	final_dataset <- cbind(cell_class_df, gene_expression_df)
	}

	#DGT
	{
	HS.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "HS", ident.2 = NULL)
	HS.markers_filtered <- subset(HS.markers, p_val_adj < 0.05)
	nHS.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "nHS", ident.2 = NULL)
	nHS.markers_filtered <- subset(nHS.markers, p_val_adj < 0.05)
	nHS.HS.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "nHS", ident.2 = "HS")
	nHS.HS.markers_filtered <- subset(nHS.HS.markers, p_val_adj < 0.05)
	LMP1.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "LMP1", ident.2 = NULL)
	LMP1.markers_filtered <- subset(LMP1.markers, p_val_adj < 0.05)
	LF3.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "LF3", ident.2 = NULL)
	LF3.markers_filtered <- subset(LF3.markers, p_val_adj < 0.05)
	LMP1.LF3.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "LMP1", ident.2 =
"LF3")
	LMP1.LF3.markers_filtered <- subset(LMP1.LF3.markers, p_val_adj < 0.05)
	LMP1_LF3.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "LMP1_LF3", ident.2
= NULL)
	LMP1_LF3.markers_filtered <- subset(LMP1_LF3.markers, p_val_adj < 0.05)
	combined_gene_list <- unique(c(rownames(HS.markers_filtered),
	                       rownames(nHS.markers_filtered),
	                       rownames(nHS.HS.markers_filtered),
	                       rownames(LMP1.markers_filtered),
	                       rownames(LF3.markers_filtered),
	                       rownames(LMP1_LF3.markers_filtered),
	                       rownames(LMP1.LF3.markers_filtered)))
	combined_gene_list_filtered <- setdiff(combined_gene_list, EBV)
	}
```

```
#Random Forest
{
valid_genes <- intersect(combined_gene_list_filtered, colnames(final_dataset))
trimmed_dataset <- final_dataset[, c("CellID", "Class", valid_genes)]
trimmed_dataset$Class <- as.factor(trimmed_dataset$Class)
set.seed(123)
train_indices <- sample(nrow(trimmed_dataset), size = 0.7 * nrow(trimmed_dataset))
train_data <- trimmed_dataset[train_indices, -1]
test_data <- trimmed_dataset[-train_indices, -1]
x_train <- train_data[, !(names(train_data) %in% c("CellID", "Class"))]
y_train <- train_data$Class
x_test <- test_data[, !(names(test_data) %in% c("CellID", "Class"))]
y_test <- test_data$Class
rf_model <- ranger(
  dependent.variable.name = "y_train",
  data = data.frame(x_train, y_train = y_train),
  num.trees = 500,
  mtry = sqrt(ncol(x_train)),
  importance = 'impurity'
)
pred_class <- predict(rf_model, data.frame(x_test), type="response")
predictions <- pred_class$predictions
confusionMatrix <- table(y_test, Predictions = predictions)
print(confusionMatrix)
importance_scores <- rf_model$variable.importance
ordered_importance <- sort(importance_scores, decreasing = TRUE)
print(head(ordered_importance, n=50))
print(ordered_importance)
gene_names <- names(ordered_importance)
print(gene_names)
gene_names<-data.frame(gene_names)
gene_names
}

#Mult Log Reg
{
top_25_genes <- names(ordered_importance)[1:25]
final_dataset_filtered <- final_dataset[, c("CellID", "Class", top_25_genes)]
final_dataset_filtered$Class <- as.factor(final_dataset_filtered$Class)
current_dataset <- final_dataset_filtered
min_aic <- Inf
optimal_dataset <- current_dataset
removed_genes <- c()
repeat {
      aic_values <- setNames(numeric(ncol(current_dataset) - 2), colnames(current_dataset)[-
(1:2)])

      for (gene in names(aic_values)) {
        dataset_minus_gene <- current_dataset[, !colnames(current_dataset) %in% c(gene,
"CellID")]
        model <- multinom(Class ~ ., data = dataset_minus_gene, trace = FALSE)
        aic_values[gene] <- AIC(model)
      }

      gene_to_remove <- names(which.min(aic_values))
      new_aic <- min(aic_values)

      if (new_aic < min_aic) {
        min_aic <- new_aic
        current_dataset <- current_dataset[, !colnames(current_dataset) %in%
c(gene_to_remove, "CellID")]
        optimal_dataset <- current_dataset
        removed_genes <- c(removed_genes, gene_to_remove)
        cat("Removed gene:", gene_to_remove, "New AIC:", new_aic, "\n")
      } else {
        break
      }
    }

    final_model <- multinom(Class ~ ., data = optimal_dataset, trace = FALSE)
    summary(final_model)
    predicted_probs <- predict(final_model, newdata = optimal_dataset, type = "probs")
    predicted_class <- apply(predicted_probs, 1, which.max)
    predicted_class <- levels(optimal_dataset$Class)[predicted_class]
    actual_class <- optimal_dataset$Class
    confusion_matrix <- table(Predicted = predicted_class, Actual = actual_class)
    print(confusion_matrix)
    vif_values <- vif(final_model)
    print(vif_values)
    }

    #Elastic Net
    {
      top_25_genes <- names(ordered_importance)[1:25]
      predictors <- as.matrix(final_dataset[, top_25_genes])
      response <- as.factor(final_dataset$Class)
      set.seed(123)
      cv_model <- cv.glmnet(predictors, response, family = "multinomial", type.multinomial
= "grouped",
                    alpha = 0.5)
      best_lambda <- cv_model$lambda.min
      plot(cv_model)
      predicted_probs <- predict(cv_model, newx = predictors, s = "lambda.min", type =
"response")
      predicted_class <- apply(predicted_probs, 1, which.max)
      predicted_class <- colnames(predicted_probs)[predicted_class]
      actual_class <- response
      confusion_matrix <- table(Predicted = predicted_class, Actual = actual_class)
      print(confusion_matrix)
      summary(cv_model)
    }
      }

      #Predict EBV Null
      {

        cell_ids_HS <- colnames(Shair_CL_df_HS)
        cell_ids_nHS <- colnames(Shair_CL_df_nHS)
        cell_ids_LF3 <- colnames(Shair_CL_df_LF3)
        cell_ids_LMP1 <- colnames(Shair_CL_df_LMP1)
        cell_ids_LMP1_LF3 <- colnames(Shair_CL_df_LMP1_LF3)
        Shair_CL_df$cell_type <- "unknown"
        Shair_CL_df$cell_type[cell_ids_HS] <- "HS"
        Shair_CL_df$cell_type[cell_ids_nHS] <- "nHS"
        Shair_CL_df$cell_type[cell_ids_LF3] <- "LF3"
        Shair_CL_df$cell_type[cell_ids_LMP1] <- "LMP1"
        Shair_CL_df$cell_type[cell_ids_LMP1_LF3] <- "LMP1_LF3"
        Shair_CL_df_unknown <- subset(Shair_CL_df, subset = cell_type == "unknown" &
`LMP-1/BNLF2a` == 0 & LF3 == 0)
        cell_ids_unknown <- colnames(Shair_CL_df_unknown)
        Shair_CL_df_unknown@meta.data$class <- "unknown"
        gene_expression_data_unknown <- GetAssayData(Shair_CL_df_unknown, slot =
"data")
        gene_expression_df_unknown <- as.data.frame(t(gene_expression_data_unknown))
        cell_class_df <- data.frame(CellID = cell_ids_unknown, Class = rep("unknown",
length(cell_ids_unknown)))
        unknown_dataset <- cbind(cell_class_df, gene_expression_df_unknown)
        valid_genes_unknown <- intersect(valid_genes, colnames(unknown_dataset))
        unknown_dataset_filtered <- unknown_dataset[, c("CellID", "Class",
valid_genes_unknown)]
        x_unknown <- unknown_dataset_filtered[, !(names(unknown_dataset_filtered) %in%
c("CellID", "Class"))]
        unknown_predictions <- predict(rf_model, data = data.frame(x_unknown), type =
"response")
        predicted_classes <- unknown_predictions$predictions
        unknown_dataset_filtered$Predicted_Class <- predicted_classes
```

```
            levels_mapping <- levels(trimmed_dataset$Class)
            unknown_dataset_filtered$Predicted_Class <- levels_mapping[predicted_classes]
            Shair_CL_df_unknown@meta.data$Predicted_Class <-
unknown_dataset_filtered$Predicted_Class[match(colnames(Shair_CL_df_unknown),
unknown_dataset_filtered$CellID)]
            Shair_CL_df$cell_type <- factor(Shair_CL_df$cell_type)
            levels(Shair_CL_df$cell_type) <- unique(c(levels(Shair_CL_df$cell_type),
levels(unknown_dataset_filtered$Predicted_Class)))
            Shair_CL_df$cell_type[cell_ids_unknown] <-
Shair_CL_df_unknown@meta.data$Predicted_Class
            DimPlot(Shair_CL_df, reduction = "umap", group.by = "cell_type", label = FALSE,
pt.size = 0.05,
                  cols = colorRampPalette(brewer.pal(5,
"Set1"))(length(unique(Shair_CL_df$cell_type)))) +
             ggtitle("UMAP of Predicted Infection Classifications")
          }

          #008
          {
            #Prepare Dataset
            {
             Shair_CL_df_LF3<-Shair_CL_df
             Shair_CL_df_LF3 <- subset(Shair_CL_df_LF3, idents = "11", invert = TRUE)
             cells_to_remove <- WhichCells(Shair_CL_df, expression = `LMP-1/BNLF2a` > 0)
             Shair_CL_df_LF3 <- subset(Shair_CL_df_LF3, cells = cells_to_remove, invert =
TRUE)
             Shair_CL_df_LF3 <- subset(Shair_CL_df_LF3, subset = LF3 > 0)
             final_cell_ids <- colnames(Shair_CL_df_LF3)
             Shair_CL_df_LF3 <- subset(Shair_CL_df, cells = final_cell_ids)
             Shair_CL_df_LMP1 <- Shair_CL_df
             Shair_CL_df_LMP1 <- subset(Shair_CL_df_LMP1, idents = "11", invert = TRUE)
             cells_to_remove_LF3 <- WhichCells(Shair_CL_df_LMP1, expression = LF3 > 0)
             Shair_CL_df_LMP1 <- subset(Shair_CL_df_LMP1, cells = cells_to_remove_LF3,
invert = TRUE)
             Shair_CL_df_LMP1 <- subset(Shair_CL_df_LMP1, subset = `LMP-1/BNLF2a` > 0)
             final_cell_ids <- colnames(Shair_CL_df_LMP1)
             Shair_CL_df_LMP1 <- subset(Shair_CL_df, cells = final_cell_ids)
             Shair_CL_df_cluster11 <- subset(Shair_CL_df, idents = "11")
             metadata_cluster11 <- Shair_CL_df_cluster11@meta.data
             data_cluster11 <- data.frame(
               CellID = rownames(metadata_cluster11),
               nCount_RNA = metadata_cluster11$nCount_RNA)
             Shair_CL_df_cluster11_filter <- subset(Shair_CL_df_cluster11, features = EBV)
             metadata_cluster11_filter <- Shair_CL_df_cluster11_filter@meta.data
             data_cluster11_filter <- data.frame(
               CellID = rownames(metadata_cluster11_filter),
               nCount_RNA = metadata_cluster11_filter$nCount_RNA)
             combined_data_cluster11 <- merge(data_cluster11, data_cluster11_filter, by =
"CellID")
             colnames(combined_data_cluster11)[colnames(combined_data_cluster11) ==
"nCount_RNA.x"] <- "nCount_RNA_All"
             colnames(combined_data_cluster11)[colnames(combined_data_cluster11) ==
"nCount_RNA.y"] <- "nCount_RNA_EBV"
             sorted_data_cluster11 <- combined_data_cluster11 %>%
               arrange(nCount_RNA_EBV, nCount_RNA_All) %>%
               mutate(Number_ID = row_number())
             sorted_data_cluster11 <- combined_data_cluster11 %>%
               arrange(nCount_RNA_EBV, nCount_RNA_All) %>%
               mutate(Number_ID = row_number(),
                    Ratio = nCount_RNA_EBV / nCount_RNA_All)
             sorted_data_cluster11_filtered <- sorted_data_cluster11 %>%
               filter(Ratio <= 0.08)
             cell_ids_to_keep <- sorted_data_cluster11_filtered$CellID
             Shair_CL_df_nHS <- subset(Shair_CL_df, cells = cell_ids_to_keep)
             sorted_data_cluster11_filtered <- sorted_data_cluster11 %>%
               filter(Ratio > 0.08)
             cell_ids_to_keep <- sorted_data_cluster11_filtered$CellID
             Shair_CL_df_HS <- subset(Shair_CL_df, cells = cell_ids_to_keep)
             Shair_CL_df_LMP1_LF3 <- Shair_CL_df
             Shair_CL_df_LMP1_LF3 <- subset(Shair_CL_df_LMP1_LF3, idents = "11", invert
= TRUE)
             Shair_CL_df_LMP1_LF3 <- subset(Shair_CL_df_LMP1_LF3, subset = `LMP-
1/BNLF2a` > 0 & LF3 > 0)
             final_cell_ids <- colnames(Shair_CL_df_LMP1_LF3)
             Shair_CL_df_LMP1_LF3 <- subset(Shair_CL_df, cells = final_cell_ids)
             Shair_CL_df_HS@meta.data$class <- "HS"
             Idents(Shair_CL_df_HS) <- Shair_CL_df_HS$class
             levels(Shair_CL_df_HS)
             Shair_CL_df_nHS@meta.data$class <- "nHS"
             Idents(Shair_CL_df_nHS) <- Shair_CL_df_nHS$class
             levels(Shair_CL_df_nHS)
             Shair_CL_df_LF3@meta.data$class <- "LF3"
             Idents(Shair_CL_df_LF3) <- Shair_CL_df_LF3$class
             levels(Shair_CL_df_LF3)
             Shair_CL_df_LMP1@meta.data$class <- "LMP1"
             Idents(Shair_CL_df_LMP1) <- Shair_CL_df_LMP1$class
             levels(Shair_CL_df_LMP1)
             Shair_CL_df_LMP1_LF3@meta.data$class <- "LMP1_LF3"
             Idents(Shair_CL_df_LMP1_LF3) <- Shair_CL_df_LMP1_LF3$class
             levels(Shair_CL_df_LMP1_LF3)
             Shair_CL_df_data <- merge(x = Shair_CL_df_HS, y = c(Shair_CL_df_nHS,
Shair_CL_df_LF3, Shair_CL_df_LMP1, Shair_CL_df_LMP1_LF3), add.cell.ids = c("HS",
"nHS", "LF3", "LMP1", "LMP1_LF3"), project = "Combined")
             levels(Shair_CL_df_data)
             cell_ids <- colnames(Shair_CL_df_data)
             class_labels <- Idents(Shair_CL_df_data)
             gene_expression_data <- GetAssayData(Shair_CL_df_data, layer = "data")
             gene_expression_df <- as.data.frame(t(gene_expression_data))
             cell_class_df <- data.frame(CellID = cell_ids, Class = as.character(class_labels))
             final_dataset <- cbind(cell_class_df, gene_expression_df)
            }

            #DGT
            {
             HS.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "HS", ident.2 = NULL)
             HS.markers_filtered <- subset(HS.markers, p_val_adj < 0.05)
             nHS.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "nHS", ident.2 = NULL)
             nHS.markers_filtered <- subset(nHS.markers, p_val_adj < 0.05)
             nHS.HS.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "nHS", ident.2 =
"HS")
             nHS.HS.markers_filtered <- subset(nHS.HS.markers, p_val_adj < 0.05)
             LMP1.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "LMP1", ident.2 =
NULL)
             LMP1.markers_filtered <- subset(LMP1.markers, p_val_adj < 0.05)
             LF3.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "LF3", ident.2 = NULL)
             LF3.markers_filtered <- subset(LF3.markers, p_val_adj < 0.05)
             LMP1.LF3.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "LMP1", ident.2 =
"LF3")
             LMP1.LF3.markers_filtered <- subset(LMP1.LF3.markers, p_val_adj < 0.05)
             LMP1_LF3.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "LMP1_LF3",
ident.2 = NULL)
             LMP1_LF3.markers_filtered <- subset(LMP1_LF3.markers, p_val_adj < 0.05)
             combined_gene_list <- unique(c(rownames(HS.markers_filtered),
                                 rownames(nHS.markers_filtered),
                                 rownames(nHS.HS.markers_filtered),
                                 rownames(LMP1.markers_filtered),
                                 rownames(LF3.markers_filtered),
                                 rownames(LMP1_LF3.markers_filtered),
                                 rownames(LMP1.LF3.markers_filtered)))
             combined_gene_list_filtered <- setdiff(combined_gene_list, EBV)
            }

            #Random Forest
            {
             valid_genes <- intersect(combined_gene_list_filtered, colnames(final_dataset))
             trimmed_dataset <- final_dataset[, c("CellID", "Class", valid_genes)]
```

```
  trimmed_dataset$Class <- as.factor(trimmed_dataset$Class)
  set.seed(123)
  train_indices <- sample(nrow(trimmed_dataset), size = 0.7 * nrow(trimmed_dataset))
  train_data <- trimmed_dataset[train_indices, -1]
  test_data <- trimmed_dataset[-train_indices, -1]
  x_train <- train_data[, !(names(train_data) %in% c("CellID", "Class"))]
  y_train <- train_data$Class
  x_test <- test_data[, !(names(test_data) %in% c("CellID", "Class"))]
  y_test <- test_data$Class
  rf_model <- ranger(
    dependent.variable.name = "y_train",
    data = data.frame(x_train, y_train = y_train),
    num.trees = 500,
    mtry = sqrt(ncol(x_train)),
    importance = 'impurity'
  )
  pred_class <- predict(rf_model, data.frame(x_test), type="response")
  predictions <- pred_class$predictions
  confusionMatrix <- table(y_test, Predictions = predictions)
  print(confusionMatrix)
  importance_scores <- rf_model$variable.importance
  ordered_importance <- sort(importance_scores, decreasing = TRUE)
  print(head(ordered_importance, n=50))
  print(ordered_importance)
  gene_names <- names(ordered_importance)
  print(gene_names)
  gene_names<-data.frame(gene_names)
  gene_names
}

#Mult Log Reg
{
  top_25_genes <- names(ordered_importance)[1:25]
  final_dataset_filtered <- final_dataset[, c("CellID", "Class", top_25_genes)]
  final_dataset_filtered$Class <- as.factor(final_dataset_filtered$Class)
  current_dataset <- final_dataset_filtered
  min_aic <- Inf
  optimal_dataset <- current_dataset
  removed_genes <- c()
  repeat {
    aic_values <- setNames(numeric(ncol(current_dataset) - 2),
colnames(current_dataset)[-(1:2)])

    for (gene in names(aic_values)) {
      dataset_minus_gene <- current_dataset[, !colnames(current_dataset) %in% c(gene,
"CellID")]
```

```
          model <- multinom(Class ~ ., data = dataset_minus_gene, trace = FALSE)
          aic_values[gene] <- AIC(model)
        }

        gene_to_remove <- names(which.min(aic_values))
        new_aic <- min(aic_values)

        if (new_aic < min_aic) {
          min_aic <- new_aic
          current_dataset <- current_dataset[, !colnames(current_dataset) %in%
c(gene_to_remove, "CellID")]
          optimal_dataset <- current_dataset
          removed_genes <- c(removed_genes, gene_to_remove)
          cat("Removed gene:", gene_to_remove, "New AIC:", new_aic, "\n")
        } else {
          break
        }
      }

      final_model <- multinom(Class ~ ., data = optimal_dataset, trace = FALSE)
      summary(final_model)
      predicted_probs <- predict(final_model, newdata = optimal_dataset, type = "probs")
      predicted_class <- apply(predicted_probs, 1, which.max)
      predicted_class <- levels(optimal_dataset$Class)[predicted_class]
      actual_class <- optimal_dataset$Class
      confusion_matrix <- table(Predicted = predicted_class, Actual = actual_class)
      print(confusion_matrix)
      vif_values <- vif(final_model)
      print(vif_values)
    }

    #Elastic Net
    {
      top_25_genes <- names(ordered_importance)[1:25]
      predictors <- as.matrix(final_dataset[, top_25_genes])
      response <- as.factor(final_dataset$Class)
      set.seed(123)
      cv_model <- cv.glmnet(predictors, response, family = "multinomial",
type.multinomial = "grouped",
                            alpha = 0.5)
      best_lambda <- cv_model$lambda.min
      plot(cv_model)
      predicted_probs <- predict(cv_model, newx = predictors, s = "lambda.min", type =
"response")
      predicted_class <- apply(predicted_probs, 1, which.max)
      predicted_class <- colnames(predicted_probs)[predicted_class]
```

```
        actual_class <- response
        confusion_matrix <- table(Predicted = predicted_class, Actual = actual_class)
        print(confusion_matrix)
        summary(cv_model)
      }
    }

    #012
    {
      #Prepare Dataset
      {
        Shair_CL_df_LF3<-Shair_CL_df
        Shair_CL_df_LF3 <- subset(Shair_CL_df_LF3, idents = "11", invert = TRUE)
        cells_to_remove <- WhichCells(Shair_CL_df, expression = `LMP-1/BNLF2a` > 0)
        Shair_CL_df_LF3 <- subset(Shair_CL_df_LF3, cells = cells_to_remove, invert =
TRUE)
        Shair_CL_df_LF3 <- subset(Shair_CL_df_LF3, subset = LF3 > 0)
        final_cell_ids <- colnames(Shair_CL_df_LF3)
        Shair_CL_df_LF3 <- subset(Shair_CL_df, cells = final_cell_ids)
        Shair_CL_df_LMP1 <- Shair_CL_df
        Shair_CL_df_LMP1 <- subset(Shair_CL_df_LMP1, idents = "11", invert = TRUE)
        cells_to_remove_LF3 <- WhichCells(Shair_CL_df_LMP1, expression = LF3 > 0)
        Shair_CL_df_LMP1 <- subset(Shair_CL_df_LMP1, cells = cells_to_remove_LF3,
invert = TRUE)
        Shair_CL_df_LMP1 <- subset(Shair_CL_df_LMP1, subset = `LMP-1/BNLF2a` > 0)
        final_cell_ids <- colnames(Shair_CL_df_LMP1)
        Shair_CL_df_LMP1 <- subset(Shair_CL_df, cells = final_cell_ids)
        Shair_CL_df_cluster11 <- subset(Shair_CL_df, idents = "11")
        metadata_cluster11 <- Shair_CL_df_cluster11@meta.data
        data_cluster11 <- data.frame(
          CellID = rownames(metadata_cluster11),
          nCount_RNA = metadata_cluster11$nCount_RNA)
        Shair_CL_df_cluster11_filter <- subset(Shair_CL_df_cluster11, features = EBV)
        metadata_cluster11_filter <- Shair_CL_df_cluster11_filter@meta.data
        data_cluster11_filter <- data.frame(
          CellID = rownames(metadata_cluster11_filter),
          nCount_RNA = metadata_cluster11_filter$nCount_RNA)
        combined_data_cluster11 <- merge(data_cluster11, data_cluster11_filter, by =
"CellID")
        colnames(combined_data_cluster11)[colnames(combined_data_cluster11) ==
"nCount_RNA.x"] <- "nCount_RNA_All"
        colnames(combined_data_cluster11)[colnames(combined_data_cluster11) ==
"nCount_RNA.y"] <- "nCount_RNA_EBV"
        sorted_data_cluster11 <- combined_data_cluster11 %>%
          arrange(nCount_RNA_EBV, nCount_RNA_All) %>%
          mutate(Number_ID = row_number())
        sorted_data_cluster11 <- combined_data_cluster11 %>%
          arrange(nCount_RNA_EBV, nCount_RNA_All) %>%
          mutate(Number_ID = row_number(),
                 Ratio = nCount_RNA_EBV / nCount_RNA_All)
        sorted_data_cluster11_filtered <- sorted_data_cluster11 %>%
          filter(Ratio <= 0.12)
        cell_ids_to_keep <- sorted_data_cluster11_filtered$CellID
        Shair_CL_df_nHS <- subset(Shair_CL_df, cells = cell_ids_to_keep)
        sorted_data_cluster11_filtered <- sorted_data_cluster11 %>%
          filter(Ratio > 0.12)
        cell_ids_to_keep <- sorted_data_cluster11_filtered$CellID
        Shair_CL_df_HS <- subset(Shair_CL_df, cells = cell_ids_to_keep)
        Shair_CL_df_LMP1_LF3 <- Shair_CL_df
        Shair_CL_df_LMP1_LF3 <- subset(Shair_CL_df_LMP1_LF3, idents = "11", invert
= TRUE)
        Shair_CL_df_LMP1_LF3 <- subset(Shair_CL_df_LMP1_LF3, subset = `LMP-
1/BNLF2a` > 0 & LF3 > 0)
        final_cell_ids <- colnames(Shair_CL_df_LMP1_LF3)
        Shair_CL_df_LMP1_LF3 <- subset(Shair_CL_df, cells = final_cell_ids)
        Shair_CL_df_HS@meta.data$class <- "HS"
        Idents(Shair_CL_df_HS) <- Shair_CL_df_HS$class
        levels(Shair_CL_df_HS)
        Shair_CL_df_nHS@meta.data$class <- "nHS"
        Idents(Shair_CL_df_nHS) <- Shair_CL_df_nHS$class
        levels(Shair_CL_df_nHS)
        Shair_CL_df_LF3@meta.data$class <- "LF3"
        Idents(Shair_CL_df_LF3) <- Shair_CL_df_LF3$class
        levels(Shair_CL_df_LF3)
        Shair_CL_df_LMP1@meta.data$class <- "LMP1"
        Idents(Shair_CL_df_LMP1) <- Shair_CL_df_LMP1$class
        levels(Shair_CL_df_LMP1)
        Shair_CL_df_LMP1_LF3@meta.data$class <- "LMP1_LF3"
        Idents(Shair_CL_df_LMP1_LF3) <- Shair_CL_df_LMP1_LF3$class
        levels(Shair_CL_df_LMP1_LF3)
        Shair_CL_df_data <- merge(x = Shair_CL_df_HS, y = c(Shair_CL_df_nHS,
Shair_CL_df_LF3, Shair_CL_df_LMP1, Shair_CL_df_LMP1_LF3), add.cell.ids = c("HS",
"nHS", "LF3", "LMP1", "LMP1_LF3"), project = "Combined")
        levels(Shair_CL_df_data)
        cell_ids <- colnames(Shair_CL_df_data)
        class_labels <- Idents(Shair_CL_df_data)
        gene_expression_data <- GetAssayData(Shair_CL_df_data, layer = "data")
        gene_expression_df <- as.data.frame(t(gene_expression_data))
        cell_class_df <- data.frame(CellID = cell_ids, Class = as.character(class_labels))
        final_dataset <- cbind(cell_class_df, gene_expression_df)
       }
```

```
#DGT
{
 HS.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "HS", ident.2 = NULL)
 HS.markers_filtered <- subset(HS.markers, p_val_adj < 0.05)
 nHS.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "nHS", ident.2 = NULL)
 nHS.markers_filtered <- subset(nHS.markers, p_val_adj < 0.05)
 nHS.HS.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "nHS", ident.2 =
"HS")
 nHS.HS.markers_filtered <- subset(nHS.HS.markers, p_val_adj < 0.05)
 LMP1.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "LMP1", ident.2 =
NULL)
 LMP1.markers_filtered <- subset(LMP1.markers, p_val_adj < 0.05)
 LF3.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "LF3", ident.2 = NULL)
 LF3.markers_filtered <- subset(LF3.markers, p_val_adj < 0.05)
 LMP1.LF3.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "LMP1", ident.2 =
"LF3")
 LMP1.LF3.markers_filtered <- subset(LMP1.LF3.markers, p_val_adj < 0.05)
 LMP1_LF3.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "LMP1_LF3",
ident.2 = NULL)
 LMP1_LF3.markers_filtered <- subset(LMP1_LF3.markers, p_val_adj < 0.05)
 combined_gene_list <- unique(c(rownames(HS.markers_filtered),
                    rownames(nHS.markers_filtered),
                    rownames(nHS.HS.markers_filtered),
                    rownames(LMP1.markers_filtered),
                    rownames(LF3.markers_filtered),
                    rownames(LMP1_LF3.markers_filtered),
                    rownames(LMP1.LF3.markers_filtered)))
 combined_gene_list_filtered <- setdiff(combined_gene_list, EBV)
}

#Random Forest
{
 valid_genes <- intersect(combined_gene_list_filtered, colnames(final_dataset))
 trimmed_dataset <- final_dataset[, c("CellID", "Class", valid_genes)]
 trimmed_dataset$Class <- as.factor(trimmed_dataset$Class)
 set.seed(123)
 train_indices <- sample(nrow(trimmed_dataset), size = 0.7 * nrow(trimmed_dataset))
 train_data <- trimmed_dataset[train_indices, -1]
 test_data <- trimmed_dataset[-train_indices, -1]
 x_train <- train_data[, !(names(train_data) %in% c("CellID", "Class"))]
 y_train <- train_data$Class
 x_test <- test_data[, !(names(test_data) %in% c("CellID", "Class"))]
 y_test <- test_data$Class
 rf_model <- ranger(
  dependent.variable.name = "y_train",
  data = data.frame(x_train, y_train = y_train),
  num.trees = 500,
  mtry = sqrt(ncol(x_train)),
  importance = 'impurity'
 )
 pred_class <- predict(rf_model, data.frame(x_test), type="response")
 predictions <- pred_class$predictions
 confusionMatrix <- table(y_test, Predictions = predictions)
 print(confusionMatrix)
 importance_scores <- rf_model$variable.importance
 ordered_importance <- sort(importance_scores, decreasing = TRUE)
 print(head(ordered_importance, n=50))
 print(ordered_importance)
 gene_names <- names(ordered_importance)
 print(gene_names)
 gene_names<-data.frame(gene_names)
 gene_names
}

#Mult Log Reg
{
 top_25_genes <- names(ordered_importance)[1:25]
 final_dataset_filtered <- final_dataset[, c("CellID", "Class", top_25_genes)]
 final_dataset_filtered$Class <- as.factor(final_dataset_filtered$Class)
 current_dataset <- final_dataset_filtered
 min_aic <- Inf
 optimal_dataset <- current_dataset
 removed_genes <- c()
 repeat {
  aic_values <- setNames(numeric(ncol(current_dataset) - 2),
colnames(current_dataset)[-(1:2)])

  for (gene in names(aic_values)) {
    dataset_minus_gene <- current_dataset[, !colnames(current_dataset) %in% c(gene,
"CellID")]
    model <- multinom(Class ~ ., data = dataset_minus_gene, trace = FALSE)
    aic_values[gene] <- AIC(model)
  }

  gene_to_remove <- names(which.min(aic_values))
  new_aic <- min(aic_values)

  if (new_aic < min_aic) {
    min_aic <- new_aic
    current_dataset <- current_dataset[, !colnames(current_dataset) %in%
c(gene_to_remove, "CellID")]
    optimal_dataset <- current_dataset
    removed_genes <- c(removed_genes, gene_to_remove)
    cat("Removed gene:", gene_to_remove, "New AIC:", new_aic, "\n")
  } else {
    break
  }
 }

 final_model <- multinom(Class ~ ., data = optimal_dataset, trace = FALSE)
 summary(final_model)
 predicted_probs <- predict(final_model, newdata = optimal_dataset, type = "probs")
 predicted_class <- apply(predicted_probs, 1, which.max)
 predicted_class <- levels(optimal_dataset$Class)[predicted_class]
 actual_class <- optimal_dataset$Class
 confusion_matrix <- table(Predicted = predicted_class, Actual = actual_class)
 print(confusion_matrix)
 vif_values <- vif(final_model)
 print(vif_values)
}

#Elastic Net
{
 top_25_genes <- names(ordered_importance)[1:25]
 predictors <- as.matrix(final_dataset[, top_25_genes])
 response <- as.factor(final_dataset$Class)
 set.seed(123)
 cv_model <- cv.glmnet(predictors, response, family = "multinomial",
type.multinomial = "grouped",
                            alpha = 0.5)
 best_lambda <- cv_model$lambda.min
 plot(cv_model)
 predicted_probs <- predict(cv_model, newx = predictors, s = "lambda.min", type =
"response")
 predicted_class <- apply(predicted_probs, 1, which.max)
 predicted_class <- colnames(predicted_probs)[predicted_class]
 actual_class <- response
 confusion_matrix <- table(Predicted = predicted_class, Actual = actual_class)
 print(confusion_matrix)
 summary(cv_model)
}
}

#020
{
#Prepare Dataset
{
 Shair_CL_df_LF3<-Shair_CL_df
```

```
        Shair_CL_df_LF3 <- subset(Shair_CL_df_LF3, idents = "11", invert = TRUE)
        cells_to_remove <- WhichCells(Shair_CL_df, expression = `LMP-1/BNLF2a` > 0)
        Shair_CL_df_LF3 <- subset(Shair_CL_df_LF3, cells = cells_to_remove, invert =
TRUE)
        Shair_CL_df_LF3 <- subset(Shair_CL_df_LF3, subset = LF3 > 0)
        final_cell_ids <- colnames(Shair_CL_df_LF3)
        Shair_CL_df_LF3 <- subset(Shair_CL_df, cells = final_cell_ids)
        Shair_CL_df_LMP1 <- Shair_CL_df
        Shair_CL_df_LMP1 <- subset(Shair_CL_df_LMP1, idents = "11", invert = TRUE)
        cells_to_remove_LF3 <- WhichCells(Shair_CL_df_LMP1, expression = LF3 > 0)
        Shair_CL_df_LMP1 <- subset(Shair_CL_df_LMP1, cells = cells_to_remove_LF3,
invert = TRUE)
        Shair_CL_df_LMP1 <- subset(Shair_CL_df_LMP1, subset = `LMP-1/BNLF2a` > 0)
        final_cell_ids <- colnames(Shair_CL_df_LMP1)
        Shair_CL_df_LMP1 <- subset(Shair_CL_df, cells = final_cell_ids)
        Shair_CL_df_cluster11 <- subset(Shair_CL_df, idents = "11")
        metadata_cluster11 <- Shair_CL_df_cluster11@meta.data
        data_cluster11 <- data.frame(
          CellID = rownames(metadata_cluster11),
          nCount_RNA = metadata_cluster11$nCount_RNA)
        Shair_CL_df_cluster11_filter <- subset(Shair_CL_df_cluster11, features = EBV)
        metadata_cluster11_filter <- Shair_CL_df_cluster11_filter@meta.data
        data_cluster11_filter <- data.frame(
          CellID = rownames(metadata_cluster11_filter),
          nCount_RNA = metadata_cluster11_filter$nCount_RNA)
        combined_data_cluster11 <- merge(data_cluster11, data_cluster11_filter, by =
"CellID")
        colnames(combined_data_cluster11)[colnames(combined_data_cluster11) ==
"nCount_RNA.x"] <- "nCount_RNA_All"
        colnames(combined_data_cluster11)[colnames(combined_data_cluster11) ==
"nCount_RNA.y"] <- "nCount_RNA_EBV"
        sorted_data_cluster11 <- combined_data_cluster11 %>%
          arrange(nCount_RNA_EBV, nCount_RNA_All) %>%
          mutate(Number_ID = row_number())
        sorted_data_cluster11 <- combined_data_cluster11 %>%
          arrange(nCount_RNA_EBV, nCount_RNA_All) %>%
          mutate(Number_ID = row_number(),
                 Ratio = nCount_RNA_EBV / nCount_RNA_All)
        sorted_data_cluster11_filtered <- sorted_data_cluster11 %>%
          filter(Ratio <= 0.20)
        cell_ids_to_keep <- sorted_data_cluster11_filtered$CellID
        Shair_CL_df_nHS <- subset(Shair_CL_df, cells = cell_ids_to_keep)
        sorted_data_cluster11_filtered <- sorted_data_cluster11 %>%
          filter(Ratio > 0.20)
        cell_ids_to_keep <- sorted_data_cluster11_filtered$CellID
        Shair_CL_df_HS <- subset(Shair_CL_df, cells = cell_ids_to_keep)
```

```
      Shair_CL_df_LMP1_LF3 <- Shair_CL_df
      Shair_CL_df_LMP1_LF3 <- subset(Shair_CL_df_LMP1_LF3, idents = "11", invert
= TRUE)
      Shair_CL_df_LMP1_LF3 <- subset(Shair_CL_df_LMP1_LF3, subset = `LMP-
1/BNLF2a` > 0 & LF3 > 0)
      final_cell_ids <- colnames(Shair_CL_df_LMP1_LF3)
      Shair_CL_df_LMP1_LF3 <- subset(Shair_CL_df, cells = final_cell_ids)
      Shair_CL_df_HS@meta.data$class <- "HS"
      Idents(Shair_CL_df_HS) <- Shair_CL_df_HS$class
      levels(Shair_CL_df_HS)
      Shair_CL_df_nHS@meta.data$class <- "nHS"
      Idents(Shair_CL_df_nHS) <- Shair_CL_df_nHS$class
      levels(Shair_CL_df_nHS)
      Shair_CL_df_LF3@meta.data$class <- "LF3"
      Idents(Shair_CL_df_LF3) <- Shair_CL_df_LF3$class
      levels(Shair_CL_df_LF3)
      Shair_CL_df_LMP1@meta.data$class <- "LMP1"
      Idents(Shair_CL_df_LMP1) <- Shair_CL_df_LMP1$class
      levels(Shair_CL_df_LMP1)
      Shair_CL_df_LMP1_LF3@meta.data$class <- "LMP1_LF3"
      Idents(Shair_CL_df_LMP1_LF3) <- Shair_CL_df_LMP1_LF3$class
      levels(Shair_CL_df_LMP1_LF3)
      Shair_CL_df_data <- merge(x = Shair_CL_df_HS, y = c(Shair_CL_df_nHS,
Shair_CL_df_LF3, Shair_CL_df_LMP1, Shair_CL_df_LMP1_LF3), add.cell.ids = c("HS",
"nHS", "LF3", "LMP1", "LMP1_LF3"), project = "Combined")
      levels(Shair_CL_df_data)
      cell_ids <- colnames(Shair_CL_df_data)
      class_labels <- Idents(Shair_CL_df_data)
      gene_expression_data <- GetAssayData(Shair_CL_df_data, layer = "data")
      gene_expression_df <- as.data.frame(t(gene_expression_data))
      cell_class_df <- data.frame(CellID = cell_ids, Class = as.character(class_labels))
      final_dataset <- cbind(cell_class_df, gene_expression_df)
     }

     #DGT
     {
      HS.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "HS", ident.2 = NULL)
      HS.markers_filtered <- subset(HS.markers, p_val_adj < 0.05)
      nHS.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "nHS", ident.2 = NULL)
      nHS.markers_filtered <- subset(nHS.markers, p_val_adj < 0.05)
      nHS.HS.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "nHS", ident.2 =
"HS")
      nHS.HS.markers_filtered <- subset(nHS.HS.markers, p_val_adj < 0.05)
      LMP1.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "LMP1", ident.2 =
NULL)
      LMP1.markers_filtered <- subset(LMP1.markers, p_val_adj < 0.05)
```

```
        LF3.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "LF3", ident.2 = NULL)
        LF3.markers_filtered <- subset(LF3.markers, p_val_adj < 0.05)
        LMP1.LF3.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "LMP1", ident.2 =
"LF3")
        LMP1.LF3.markers_filtered <- subset(LMP1.LF3.markers, p_val_adj < 0.05)
        LMP1_LF3.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "LMP1_LF3",
ident.2 = NULL)
        LMP1_LF3.markers_filtered <- subset(LMP1_LF3.markers, p_val_adj < 0.05)
        combined_gene_list <- unique(c(rownames(HS.markers_filtered),
                                rownames(nHS.markers_filtered),
                                rownames(nHS.HS.markers_filtered),
                                rownames(LMP1.markers_filtered),
                                rownames(LF3.markers_filtered),
                                rownames(LMP1_LF3.markers_filtered),
                                rownames(LMP1.LF3.markers_filtered)))
        combined_gene_list_filtered <- setdiff(combined_gene_list, EBV)
       }

       #Random Forest
       {
        valid_genes <- intersect(combined_gene_list_filtered, colnames(final_dataset))
        trimmed_dataset <- final_dataset[, c("CellID", "Class", valid_genes)]
        trimmed_dataset$Class <- as.factor(trimmed_dataset$Class)
        set.seed(123)
        train_indices <- sample(nrow(trimmed_dataset), size = 0.7 * nrow(trimmed_dataset))
        train_data <- trimmed_dataset[train_indices, -1]
        test_data <- trimmed_dataset[-train_indices, -1]
        x_train <- train_data[, !(names(train_data) %in% c("CellID", "Class"))]
        y_train <- train_data$Class
        x_test <- test_data[, !(names(test_data) %in% c("CellID", "Class"))]
        y_test <- test_data$Class
        rf_model <- ranger(
          dependent.variable.name = "y_train",
          data = data.frame(x_train, y_train = y_train),
          num.trees = 500,
          mtry = sqrt(ncol(x_train)),
          importance = 'impurity'
        )
        pred_class <- predict(rf_model, data.frame(x_test), type="response")
        predictions <- pred_class$predictions
        confusionMatrix <- table(y_test, Predictions = predictions)
        print(confusionMatrix)
        importance_scores <- rf_model$variable.importance
        ordered_importance <- sort(importance_scores, decreasing = TRUE)
        print(head(ordered_importance, n=50))
        print(ordered_importance)
```

```
      gene_names <- names(ordered_importance)
      print(gene_names)
      gene_names<-data.frame(gene_names)
      gene_names
    }

    #Mult Log Reg
    {
      top_25_genes <- names(ordered_importance)[1:25]
      final_dataset_filtered <- final_dataset[, c("CellID", "Class", top_25_genes)]
      final_dataset_filtered$Class <- as.factor(final_dataset_filtered$Class)
      current_dataset <- final_dataset_filtered
      min_aic <- Inf
      optimal_dataset <- current_dataset
      removed_genes <- c()
      repeat {
        aic_values <- setNames(numeric(ncol(current_dataset) - 2),
colnames(current_dataset)[-(1:2)])

        for (gene in names(aic_values)) {
          dataset_minus_gene <- current_dataset[, !colnames(current_dataset) %in% c(gene,
"CellID")]
          model <- multinom(Class ~ ., data = dataset_minus_gene, trace = FALSE)
          aic_values[gene] <- AIC(model)
        }

        gene_to_remove <- names(which.min(aic_values))
        new_aic <- min(aic_values)

        if (new_aic < min_aic) {
          min_aic <- new_aic
          current_dataset <- current_dataset[, !colnames(current_dataset) %in%
c(gene_to_remove, "CellID")]
          optimal_dataset <- current_dataset
          removed_genes <- c(removed_genes, gene_to_remove)
          cat("Removed gene:", gene_to_remove, "New AIC:", new_aic, "\n")
        } else {
          break
        }
      }

      final_model <- multinom(Class ~ ., data = optimal_dataset, trace = FALSE)
      summary(final_model)
      predicted_probs <- predict(final_model, newdata = optimal_dataset, type = "probs")
      predicted_class <- apply(predicted_probs, 1, which.max)
      predicted_class <- levels(optimal_dataset$Class)[predicted_class]
      actual_class <- optimal_dataset$Class
      confusion_matrix <- table(Predicted = predicted_class, Actual = actual_class)
      print(confusion_matrix)
      vif_values <- vif(final_model)
      print(vif_values)
    }

    #Elastic Net
    {
      top_25_genes <- names(ordered_importance)[1:25]
      predictors <- as.matrix(final_dataset[, top_25_genes])
      response <- as.factor(final_dataset$Class)
      set.seed(123)
      cv_model <- cv.glmnet(predictors, response, family = "multinomial",
type.multinomial = "grouped",
                            alpha = 0.5)
      best_lambda <- cv_model$lambda.min
      plot(cv_model)
      predicted_probs <- predict(cv_model, newx = predictors, s = "lambda.min", type =
"response")
      predicted_class <- apply(predicted_probs, 1, which.max)
      predicted_class <- colnames(predicted_probs)[predicted_class]
      actual_class <- response
      confusion_matrix <- table(Predicted = predicted_class, Actual = actual_class)
      print(confusion_matrix)
      summary(cv_model)
    }
   }

   #Iteration 2
   {
   #Prepare Dataset
   {
    Shair_CL_df_LF3<-Shair_CL_df
    Shair_CL_df_LF3 <- subset(Shair_CL_df_LF3, idents = "11", invert = TRUE)
    cells_to_remove <- WhichCells(Shair_CL_df, expression = `LMP-1/BNLF2a` > 0)
    Shair_CL_df_LF3 <- subset(Shair_CL_df_LF3, cells = cells_to_remove, invert =
TRUE)
    Shair_CL_df_LF3 <- subset(Shair_CL_df_LF3, subset = LF3 > 0)
    final_cell_ids <- colnames(Shair_CL_df_LF3)
    Shair_CL_df_LF3 <- subset(Shair_CL_df, cells = final_cell_ids)
    Shair_CL_df_LMP1 <- Shair_CL_df
    Shair_CL_df_LMP1 <- subset(Shair_CL_df_LMP1, idents = "11", invert = TRUE)
    cells_to_remove_LF3 <- WhichCells(Shair_CL_df_LMP1, expression = LF3 > 0)
    Shair_CL_df_LMP1 <- subset(Shair_CL_df_LMP1, cells = cells_to_remove_LF3,
invert = TRUE)
```

```
            Shair_CL_df_LMP1 <- subset(Shair_CL_df_LMP1, subset = `LMP-1/BNLF2a` > 0)
            final_cell_ids <- colnames(Shair_CL_df_LMP1)
            Shair_CL_df_LMP1 <- subset(Shair_CL_df, cells = final_cell_ids)
            Shair_CL_df_cluster11 <- subset(Shair_CL_df, idents = "11")
            metadata_cluster11 <- Shair_CL_df_cluster11@meta.data
            data_cluster11 <- data.frame(
              CellID = rownames(metadata_cluster11),
              nCount_RNA = metadata_cluster11$nCount_RNA)
            Shair_CL_df_cluster11_filter <- subset(Shair_CL_df_cluster11, features = EBV)
            metadata_cluster11_filter <- Shair_CL_df_cluster11_filter@meta.data
            data_cluster11_filter <- data.frame(
              CellID = rownames(metadata_cluster11_filter),
              nCount_RNA = metadata_cluster11_filter$nCount_RNA)
            combined_data_cluster11 <- merge(data_cluster11, data_cluster11_filter, by =
"CellID")
            colnames(combined_data_cluster11)[colnames(combined_data_cluster11) ==
"nCount_RNA.x"] <- "nCount_RNA_All"
            colnames(combined_data_cluster11)[colnames(combined_data_cluster11) ==
"nCount_RNA.y"] <- "nCount_RNA_EBV"
            sorted_data_cluster11 <- combined_data_cluster11 %>%
              arrange(nCount_RNA_EBV, nCount_RNA_All) %>%
              mutate(Number_ID = row_number())
            sorted_data_cluster11 <- combined_data_cluster11 %>%
              arrange(nCount_RNA_EBV, nCount_RNA_All) %>%
              mutate(Number_ID = row_number(),
                     Ratio = nCount_RNA_EBV / nCount_RNA_All)
            sorted_data_cluster11_filtered <- sorted_data_cluster11 %>%
              filter(Ratio <= 0.37)
            cell_ids_to_keep <- sorted_data_cluster11_filtered$CellID
            Shair_CL_df_nHS <- subset(Shair_CL_df, cells = cell_ids_to_keep)
            Shair_CL_df_LMP1_LF3 <- Shair_CL_df
            Shair_CL_df_LMP1_LF3 <- subset(Shair_CL_df_LMP1_LF3, idents = "11", invert =
TRUE)
            Shair_CL_df_LMP1_LF3 <- subset(Shair_CL_df_LMP1_LF3, subset = `LMP-
1/BNLF2a` > 0 & LF3 > 0)
            final_cell_ids <- colnames(Shair_CL_df_LMP1_LF3)
            Shair_CL_df_LMP1_LF3 <- subset(Shair_CL_df, cells = final_cell_ids)
            Shair_CL_df_nHS@meta.data$class <- "nHS"
            Idents(Shair_CL_df_nHS) <- Shair_CL_df_nHS$class
            levels(Shair_CL_df_nHS)
            Shair_CL_df_LF3@meta.data$class <- "LF3"
            Idents(Shair_CL_df_LF3) <- Shair_CL_df_LF3$class
            levels(Shair_CL_df_LF3)
            Shair_CL_df_LMP1@meta.data$class <- "LMP1"
            Idents(Shair_CL_df_LMP1) <- Shair_CL_df_LMP1$class
            levels(Shair_CL_df_LMP1)
            Shair_CL_df_LMP1_LF3@meta.data$class <- "LMP1_LF3"
            Idents(Shair_CL_df_LMP1_LF3) <- Shair_CL_df_LMP1_LF3$class
            levels(Shair_CL_df_LMP1_LF3)
            Shair_CL_df_data <- merge(x = Shair_CL_df_LMP1, y = c(Shair_CL_df_nHS,
Shair_CL_df_LF3, Shair_CL_df_LMP1_LF3), add.cell.ids = c("LMP1","nHS", "LF3",
"LMP1_LF3"), project = "Combined")
            levels(Shair_CL_df_data)
            cell_ids <- colnames(Shair_CL_df_data)
            class_labels <- Idents(Shair_CL_df_data)
            gene_expression_data <- GetAssayData(Shair_CL_df_data, layer = "data")
            gene_expression_df <- as.data.frame(t(gene_expression_data))
            cell_class_df <- data.frame(CellID = cell_ids, Class = as.character(class_labels))
            final_dataset <- cbind(cell_class_df, gene_expression_df)
          }

          #DGT
          {
            nHS.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "nHS", ident.2 = NULL)
            nHS.markers_filtered <- subset(nHS.markers, p_val_adj < 0.05)
            LMP1.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "LMP1", ident.2 =
NULL)
            LMP1.markers_filtered <- subset(LMP1.markers, p_val_adj < 0.05)
            LF3.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "LF3", ident.2 = NULL)
            LF3.markers_filtered <- subset(LF3.markers, p_val_adj < 0.05)
            LMP1_LF3.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "LMP1_LF3",
ident.2 = NULL)
            LMP1_LF3.markers_filtered <- subset(LMP1_LF3.markers, p_val_adj < 0.05)
            LMP1.LF3.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "LMP1", ident.2 =
"LF3")
            LMP1.LF3.markers_filtered <- subset(LMP1.LF3.markers, p_val_adj < 0.05)
            combined_gene_list <- unique(c(rownames(nHS.markers_filtered),
                                  rownames(LMP1.markers_filtered),
                                  rownames(LF3.markers_filtered),
                                  rownames(LMP1_LF3.markers_filtered),
                                  rownames(LMP1.LF3.markers_filtered)))
            combined_gene_list_filtered <- setdiff(combined_gene_list, EBV)
          }

          #Random Forest
          {
            valid_genes <- intersect(combined_gene_list_filtered, colnames(final_dataset))
            trimmed_dataset <- final_dataset[, c("CellID", "Class", valid_genes)]
            trimmed_dataset$Class <- as.factor(trimmed_dataset$Class)
            set.seed(123)
            train_indices <- sample(nrow(trimmed_dataset), size = 0.7 * nrow(trimmed_dataset))
            train_data <- trimmed_dataset[train_indices, -1]
```

```
    test_data <- trimmed_dataset[-train_indices, -1]
    x_train <- train_data[, !(names(train_data) %in% c("CellID", "Class"))]
    y_train <- train_data$Class
    x_test <- test_data[, !(names(test_data) %in% c("CellID", "Class"))]
    y_test <- test_data$Class
    rf_model <- ranger(
      dependent.variable.name = "y_train",
      data = data.frame(x_train, y_train = y_train),
      num.trees = 500,
      mtry = sqrt(ncol(x_train)),
      importance = 'impurity'
    )
    pred_class <- predict(rf_model, data.frame(x_test), type="response")
    predictions <- pred_class$predictions
    confusionMatrix <- table(y_test, Predictions = predictions)
    print(confusionMatrix)
    importance_scores <- rf_model$variable.importance
    ordered_importance <- sort(importance_scores, decreasing = TRUE)
    print(head(ordered_importance, n=50))
  }
  }

  #Iteration 3
  {
    #Prepare Dataset
    {
      Shair_CL_df_LF3<-Shair_CL_df
      Shair_CL_df_LF3 <- subset(Shair_CL_df_LF3, idents = "11", invert = TRUE)
      cells_to_remove <- WhichCells(Shair_CL_df, expression = `LMP-1/BNLF2a` > 0)
      Shair_CL_df_LF3 <- subset(Shair_CL_df_LF3, cells = cells_to_remove, invert =
TRUE)
      Shair_CL_df_LF3 <- subset(Shair_CL_df_LF3, subset = LF3 > 0)
      final_cell_ids <- colnames(Shair_CL_df_LF3)
      Shair_CL_df_LF3 <- subset(Shair_CL_df, cells = final_cell_ids)
      Shair_CL_df_LMP1 <- Shair_CL_df
      Shair_CL_df_LMP1 <- subset(Shair_CL_df_LMP1, idents = "11", invert = TRUE)
      cells_to_remove_LF3 <- WhichCells(Shair_CL_df_LMP1, expression = LF3 > 0)
      Shair_CL_df_LMP1 <- subset(Shair_CL_df_LMP1, cells = cells_to_remove_LF3,
invert = TRUE)
      Shair_CL_df_LMP1 <- subset(Shair_CL_df_LMP1, subset = `LMP-1/BNLF2a` > 0)
      final_cell_ids <- colnames(Shair_CL_df_LMP1)
      Shair_CL_df_LMP1 <- subset(Shair_CL_df, cells = final_cell_ids)
      Shair_CL_df_LMP1_LF3 <- Shair_CL_df
      Shair_CL_df_LMP1_LF3 <- subset(Shair_CL_df_LMP1_LF3, idents = "11", invert
= TRUE)
```

```
		Shair_CL_df_LMP1_LF3 <- subset(Shair_CL_df_LMP1_LF3, subset = `LMP-
1/BNLF2a` > 0 & LF3 > 0)
		final_cell_ids <- colnames(Shair_CL_df_LMP1_LF3)
		Shair_CL_df_LMP1_LF3 <- subset(Shair_CL_df, cells = final_cell_ids)
		Shair_CL_df_LF3@meta.data$class <- "LF3"
		Idents(Shair_CL_df_LF3) <- Shair_CL_df_LF3$class
		levels(Shair_CL_df_LF3)
		Shair_CL_df_LMP1@meta.data$class <- "LMP1"
		Idents(Shair_CL_df_LMP1) <- Shair_CL_df_LMP1$class
		levels(Shair_CL_df_LMP1)
		Shair_CL_df_LMP1_LF3@meta.data$class <- "LMP1_LF3"
		Idents(Shair_CL_df_LMP1_LF3) <- Shair_CL_df_LMP1_LF3$class
		levels(Shair_CL_df_LMP1_LF3)
		Shair_CL_df_data <- merge(x = Shair_CL_df_LMP1, y = c(Shair_CL_df_LF3,
Shair_CL_df_LMP1_LF3), add.cell.ids = c("LF3", "LMP1", "LMP1_LF3"), project =
"Combined")
		levels(Shair_CL_df_data)
		cell_ids <- colnames(Shair_CL_df_data)
		class_labels <- Idents(Shair_CL_df_data)
		gene_expression_data <- GetAssayData(Shair_CL_df_data, layer = "data")
		gene_expression_df <- as.data.frame(t(gene_expression_data))
		cell_class_df <- data.frame(CellID = cell_ids, Class = as.character(class_labels))
		final_dataset <- cbind(cell_class_df, gene_expression_df)
	}

	#DGT
	{
		LMP1.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "LMP1", ident.2 =
NULL)
		LMP1.markers_filtered <- subset(LMP1.markers, p_val_adj < 0.05)
		LF3.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "LF3", ident.2 = NULL)
		LF3.markers_filtered <- subset(LF3.markers, p_val_adj < 0.05)
		LMP1_LF3.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "LMP1_LF3",
ident.2 = NULL)
		LMP1_LF3.markers_filtered <- subset(LMP1_LF3.markers, p_val_adj < 0.05)
		LMP1.LF3.markers <- FindMarkers(Shair_CL_df_data, ident.1 = "LMP1", ident.2 =
"LF3")
		LMP1.LF3.markers_filtered <- subset(LMP1.LF3.markers, p_val_adj < 0.05)
		combined_gene_list <- unique(c(
rownames(LMP1.markers_filtered),
					rownames(LF3.markers_filtered),
					rownames(LMP1_LF3.markers_filtered),
					rownames(LMP1.LF3.markers_filtered)))
		combined_gene_list_filtered <- setdiff(combined_gene_list, EBV)
	}
```

```
  #Random Forest
  {
   valid_genes <- intersect(combined_gene_list_filtered, colnames(final_dataset))
   trimmed_dataset <- final_dataset[, c("CellID", "Class", valid_genes)]
   trimmed_dataset$Class <- as.factor(trimmed_dataset$Class)
   set.seed(123)
   train_indices <- sample(nrow(trimmed_dataset), size = 0.7 * nrow(trimmed_dataset))
   train_data <- trimmed_dataset[train_indices, -1]
   test_data <- trimmed_dataset[-train_indices, -1]
   x_train <- train_data[, !(names(train_data) %in% c("CellID", "Class"))]
   y_train <- train_data$Class
   x_test <- test_data[, !(names(test_data) %in% c("CellID", "Class"))]
   y_test <- test_data$Class
   rf_model <- ranger(
     dependent.variable.name = "y_train",
     data = data.frame(x_train, y_train = y_train),
     num.trees = 500,
     mtry = sqrt(ncol(x_train)),
     importance = 'impurity'
   )
   pred_class <- predict(rf_model, data.frame(x_test), type="response")
   predictions <- pred_class$predictions
   confusionMatrix <- table(y_test, Predictions = predictions)
   print(confusionMatrix)
   importance_scores <- rf_model$variable.importance
   ordered_importance <- sort(importance_scores, decreasing = TRUE)
   print(head(ordered_importance, n=50))
  }
 }

}
```

Bibliography

[1] Wong Y, Meehan MT, Burrows SR, Doolan DL, Miles JJ. Estimating the global burden of Epstein-Barr virus-related cancers. J Cancer Res Clin Oncol. 2022 Jan;148(1):31-46. doi: 10.1007/s00432-021-03824-y. Epub 2021 Oct 27. PMID: 34705104; PMCID: PMC8752571.

[2] Wang Q, Xie H, Li Y, Theodoropoulos N, Zhang Y, Jiang C, Wen C, Rozek LS, Boffetta P. Racial and ethnic disparities in nasopharyngeal cancer with an emphasis among Asian Americans. Int J Cancer. 2022 Oct 15;151(8):1291-1303. doi: 10.1002/ijc.34154. Epub 2022 Jun 22. Erratum in: Int J Cancer. 2023 Feb 15;152(4):E3. PMID: 35666524.

[3] Ziegler P, Tian Y, Bai Y, Abrahamsson S, Bäckerholm A, Reznik AS, Green A, Moore JA, Lee SE, Myerburg MM, Park HJ, Tang KW, Shair KHY. A primary nasopharyngeal three-dimensional air-liquid interface cell culture model of the pseudostratified epithelium reveals differential donor- and cell type-specific susceptibility to Epstein-Barr virus infection. PLoS Pathog. 2021 Apr 29;17(4):e1009041. doi: 10.1371/journal.ppat.1009041. PMID: 33914843; PMCID: PMC8112674.

[4] Banko A, Miljanovic D, Lazarevic I, Cirkovic A. A Systematic Review of Epstein-Barr Virus Latent Membrane Protein 1 (LMP1) Gene Variants in Nasopharyngeal Carcinoma. Pathogens. 2021 Aug 20;10(8):1057. doi: 10.3390/pathogens10081057. PMID: 34451521; PMCID: PMC8401687.

[5] Zheng, Grace X.Y., Terry, Jessica M., [...] Bielas, Jason H. (2017). Massively parallel digital transcriptional profiling of single cells. Nature Communications. 8: 1-12, doi:10.1038/ncomms14049

[6] Lee J, Kosowicz JG, Hayward SD, Desai P, Stone J, Lee JM, Liu JO, Ambinder RF. Pharmacologic Activation of Lytic Epstein-Barr Virus Gene Expression without Virion Production. J Virol. 2019 Sep 30;93(20):e00998-19. doi: 10.1128/JVI.00998-19. PMID: 31341058; PMCID: PMC6798122.

[7] Yuhan Hao, Stephanie Hao, Erica Andersen-Nissen, William M. Mauck III, Shiwei Zheng, Andrew Butler, Maddie J. Lee, Aaron J. Wilk, Charlotte Darby, Michael Zagar, Paul Hoffman, Marlon Stoeckius, Efthymia Papalexi, Eleni P. Mimitou, Jaison Jain, Avi Srivastava, Tim Stuart, Lamar B. Fleming, Bertrand Yeung, Angela J. Rogers, Juliana M. McElrath, Catherine A. Blish, Raphael Gottardo, Peter Smibert, & Rahul Satija (2021). Integrated analysis of multimodal single-cell data. *Cell*.

[8] Cleveland, W.S.1979.Robust Locally Weighted Regression and Smoothing Scatterplots. Journal of the American Statistical Association 74:829-836

[9] Hadley Wickham (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.

[10] Gu Z, Eils R, Schlesner M (2016). "Complex heatmaps reveal patterns and correlations in multidimensional genomic data." Bioinformatics. doi:10.1093/bioinformatics/btw313.

[11] Haynes, W. (2013). Wilcoxon Rank Sum Test. In: Dubitzky, W., Wolkenhauer, O., Cho, KH., Yokota, H. (eds) Encyclopedia of Systems Biology. Springer, New York, NY. https://doi.org/10.1007/978-1-4419-9863-7_1185

[12] Marvin N. Wright, & Andreas Ziegler (2017). ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. *Journal of Statistical Software, 77(1), 1–17.*

[13] Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001). https://doi.org/10.1023/A:1010933404324

[14] Trevethan R. Sensitivity, Specificity, and Predictive Values: Foundations, Pliabilities, and Pitfalls in Research and Practice. Front Public Health. 2017 Nov 20;5:307. doi: 10.3389/fpubh.2017.00307. PMID: 29209603; PMCID: PMC5701930.

[15] Hirokazu Yanagihara, Ken-ichi Kamo, Shinpei Imori, Kenichi Satoh, Bias-corrected AIC for selecting variables in multinomial logistic regression models, Linear Algebra and its Applications, Volume 436, Issue 11, 2012, Pages 4329-4341, ISSN 0024-3795, https://doi.org/10.1016/j.laa.2012.01.018.

[16] W. N. Venables, & B. D. Ripley (2002). *Modern Applied Statistics with S*. Springer.

[17] Liuyuan Chen, Jie Yang, Juntao Li, Xiaoyu Wang, "Multinomial Regression with Elastic Net Penalty and Its Grouping Effect in Gene Selection", Abstract and Applied Analysis, vol. 2014, Article ID 569501, 7 pages, 2014. https://doi.org/10.1155/2014/569501

[18] Friedman J, Tibshirani R, Hastie T (2010). "Regularization Paths for Generalized Linear Models via Coordinate Descent." Journal of Statistical Software, 33(1), 1–22. doi:10.18637/jss.v033.i01.

[19] Tay JK, Narasimhan B, Hastie T (2023). "Elastic Net Regularization Paths for All Generalized Linear Models." Journal of Statistical Software, 106(1), 1–31. doi:10.18637/jss.v106.i01.

[20] Rowe M, Glaunsinger B, van Leeuwen D, Zuo J, Sweetman D, Ganem D, Middeldorp J, Wiertz EJ, Ressing ME. Host shutoff during productive Epstein-Barr virus infection is mediated by BGLF5 and may contribute to immune evasion. Proc Natl Acad Sci U S A. 2007 Feb 27;104(9):3366-71. doi: 10.1073/pnas.0611128104. Epub 2007 Feb 21. PMID: 17360652; PMCID: PMC1805610.

[21] Milacic M, Beavers D, Conley P, Gong C, Gillespie M, Griss J, Haw R, Jassal B, Matthews L, May B, Petryszak R, Ragueneau E, Rothfels K, Sevilla C, Shamovsky V, Stephan R, Tiwari K, Varusai T, Weiser J, Wright A, Wu G, Stein L, Hermjakob H, D'Eustachio P. The Reactome Pathway Knowledgebase 2024. Nucleic Acids Research. 2024. doi: 10.1093/nar/gkad1025

[22] Ratnasiri K, Wilk AJ, Lee MJ, Khatri P, Blish CA. Single-cell RNA-seq methods to interrogate virus-host interactions. Semin Immunopathol. 2023 Jan;45(1):71-89. doi: 10.1007/s00281-022-00972-2. Epub 2022 Nov 21. PMID: 36414692; PMCID: PMC9684776.

[23] Wang L, Ning S. New Look of EBV LMP1 Signaling Landscape. Cancers (Basel). 2021 Oct 29;13(21):5451. doi: 10.3390/cancers13215451. PMID: 34771613; PMCID: PMC8582580.

[24] Yetming KD, Lupey-Green LN, Biryukov S, Hughes DJ, Marendy EM, Miranda JL, Sample JT. The BHLF1 Locus of Epstein-Barr Virus Contributes to Viral Latency and B-Cell Immortalization. J Virol. 2020 Aug 17;94(17):e01215-20. doi: 10.1128/JVI.01215-20. PMID: 32581094; PMCID: PMC7431786.

[25] Yetming KD, Lupey-Green LN, Biryukov S, Hughes DJ, Marendy EM, Miranda JL, Sample JT. The BHLF1 Locus of Epstein-Barr Virus Contributes to Viral Latency and B-Cell Immortalization. J Virol. 2020 Aug 17;94(17):e01215-20. doi: 10.1128/JVI.01215-20. PMID: 32581094; PMCID: PMC7431786.

[26] Xue SA, Griffin BE. Complexities associated with expression of Epstein-Barr virus (EBV) lytic origins of DNA replication. Nucleic Acids Res. 2007;35(10):3391-406. doi: 10.1093/nar/gkm170. Epub 2007 May 3. PMID: 17478522; PMCID: PMC1904260.

[27] Rowe M, Glaunsinger B, van Leeuwen D, Zuo J, Sweetman D, Ganem D, Middeldorp J, Wiertz EJ, Ressing ME. Host shutoff during productive Epstein-Barr virus infection is mediated by BGLF5 and may contribute to immune evasion. Proc Natl Acad Sci U S A. 2007 Feb 27;104(9):3366-71. doi: 10.1073/pnas.0611128104. Epub 2007 Feb 21. PMID: 17360652; PMCID: PMC1805610.

[28] Javad Rahimikollu, Hanxi Xiao, Anna E. Rosengart, Tracy Tabib, Paul Zdinak, Kun He, Xin Bing, Florentina Bunea, Marten Wegkamp, Amanda C. Poholek, Alok V Joglekar, Robert A Lafyatis, Jishnu Das bioRxiv 2022.11.25.518001; doi: https://doi.org/10.1101/2022.11.25.518001