Received: 25 October 2020 | Revised: 1 April 2021 | Accepted: 6 April 2021

DOI: 10.1002/cyto.a.24350

ORIGINAL ARTICLE

ISAC CYTOMETRY Journal of Quantitative Cell Science PART A

Integration, exploration, and analysis of high-dimensional single-cell cytometry data using Spectre

Thomas Myles Ashhurst[1,2,3] | Felix Marsh-Wakefield[3,4,5] | Givanna Haryono Putri[3,6] | Alanna Gabrielle Spiteri[3,7] | Diana Shinko[1,3] | Mark Norman Read[3,6,8] | Adrian Lloyd Smith[1,3] | Nicholas Jonathan Cole King[1,2,3,7,9]

[1]Sydney Cytometry Core Research Facility, Charles Perkins Centre, Centenary Institute and The University of Sydney, Sydney, New South Wales, Australia

[2]Marie Bashir Institute for Infectious Diseases and Biosecurity, The University of Sydney, Sydney, New South Wales, Australia

[3]Charles Perkins Centre, The University of Sydney, Sydney, New South Wales, Australia

[4]School of Medical Sciences, Faculty of Medicine and Health, The University of Sydney, Sydney, New South Wales, Australia

[5]Vascular Immunology Unit, Department of Pathology, The University of Sydney, Sydney, New South Wales, Australia

[6]School of Computer Science, The University of Sydney, Sydney, New South Wales, Australia

[7]Viral Immunopathology Laboratory, Discipline of Pathology, School of Medical Sciences, Faculty of Medicine and Health, The University of Sydney, Sydney, New South Wales, Australia

[8]The Westmead Initiative, The University of Sydney, Sydney, New South Wales, Australia

[9]Sydney Nano, The University of Sydney, Sydney, New South Wales, Australia

Correspondence
Thomas Myles Ashhurst, Charles Perkins Centre, The University of Sydney, Camperdown, NSW 2050.
Email: thomas.ashhurst@sydney.edu.au

Funding information
Marie Bashir Institute for Infectious Disease and Biosecurity; Merridew Foundation; National Health and Medical Research Council (NH&MRC), Grant/Award Number: 1088242; Marie Bashir Institute, University of Sydney

Abstract

As the size and complexity of high-dimensional (HD) cytometry data continue to expand, comprehensive, scalable, and methodical computational analysis approaches are essential. Yet, contemporary clustering and dimensionality reduction tools alone are insufficient to analyze or reproduce analyses across large numbers of samples, batches, or experiments. Moreover, approaches that allow for the integration of data across batches or experiments are not well incorporated into computational toolkits to allow for streamlined workflows. Here we present Spectre, an R package that enables comprehensive end-to-end integration and analysis of HD cytometry data from different batches or experiments. Spectre streamlines the analytical stages of raw data pre-processing, batch alignment, data integration, clustering, dimensionality reduction, visualization, and population labelling, as well as quantitative and statistical analysis. Critically, the fundamental data structures used within Spectre, along with the implementation of machine learning classifiers, allow for the scalable analysis of very large HD datasets, generated by flow cytometry, mass cytometry, or spectral cytometry. Using open and flexible data structures, Spectre can also be used to analyze data generated by single-cell RNA sequencing or HD imaging technologies, such as Imaging Mass Cytometry. The simple, clear, and modular design of analysis workflows allow these tools to be used by bioinformaticians and laboratory scientists alike. Spectre is available as an R package or Docker container. R code is available on Github (https://github.com/immunedynamics/spectre).

KEYWORDS
clustering, computational analysis, dimensionality reduction, FlowSOM, high-dimensional cytometry, mass cytometry, spectral cytometry, t-SNE, UMAP

Thomas Myles Ashhurst, Felix Marsh-Wakefield, and Givanna Haryono Putri contributed equally to this work

1 | INTRODUCTION

1.1 | High-dimensional analysis tools

High-dimensional (HD) cytometry plays an important role in the study of immunology, infectious diseases, autoimmunity, hematology, and cancer biology, elucidating critical mediators of immunity and disease at a single-cell level. Advances in single-cell cytometry systems (including flow, spectral, and mass cytometry) have enabled the simultaneous analysis of over 40 proteins [1–4] in a single panel, resulting in vast, and complex datasets. A large portion of the analysis still utilizes manual gating, which is the sequential and often arduous process of identifying cells, based on the expression of one or two cellular markers at a time. While this allows for user-guided examination of the dataset, it is intractable when mapping out the vast variety of cell phenotypes that may be present in HD datasets, and may introduce selective bias through a subjective and sequential bifurcating inclusion/exclusion of markers [5]. This is in part due to the number of gates required to fully parse the dataset. As such, a variety of computational approaches have been adopted by the cytometry community to help analyze HD datasets, including automated gating [6], clustering (such as PhenoGraph [7], FlowSOM [8], X-Shift [9]), dimensionality reduction (such as t-SNE [10, 11], UMAP [12]), trajectory inference (such as Wanderlust [1], Wishbone [13]), and automated cell classification [14–16]. Many of these tools have been brought together into 'toolboxes', providing either code- or GUI-based analysis workflows, such as Cytofkit [17], CITRUS [18], CATALYST [19], Cytofworkflow [20], or diffcyt [21]. These toolboxes add to an environment of computational tools that drive computational cytometry analysis, such as the Cytoverse environment, including ggCyto [22] and OpenCyto [23].

In parallel, similar tools have also been developed in other fields, such as a wide variety of analysis approaches for single-cell RNA sequencing (scRNAseq; Seurat [24, 25], SingleCellExperiment (SCE) [26], Monocle [27]). These have been developed to address similar difficulties in analysis, with developments in one field sometimes assisting another, such as the graph-based clustering from Phenograph [7] informing the design of clustering in Seurat [24]. Overall, these tools allow for automated and data-driven data processing that may be performed in an unsupervised (requiring no human supervision) or semi-supervised manner, requiring some human decision-making.

1.2 | Limitations of existing algorithms and toolkits

Despite the advantages offered by these computationally-driven approaches, a number of challenges persist, including slow operation speeds, difficulty in scaling to large datasets, and insufficient reproducibility of analysis results across independent experiments. As such, the use of such tools is often limited to relatively small datasets from single experiments. In particular, many scRNAseq specific tools, while suited to processing datasets with a large number of features (markers), do not necessarily scale well to dataset volumes that include tens to hundreds of millions of cells, as the volume of data may exceed the memory capacity of the computer being used for analysis, and/or the processing may carry excessively long run times [28]. In scRNAseq, independent datasets of this volume are not common outside of cell atlas-type projects [29], but cytometry frequently generates datasets of this size. In addition, these data-driven approaches are highly sensitive to the influence of batch effects, a phenomenon in which cells of the same phenotype differ in their signal intensity across multiple experimental batches. Batch effects are an artifact of time and context, in which experimental conditions and/or instrumental performance inadvertently influence the measured signal intensity. Left uncorrected, computational algorithms may falsely identify differences (or similarities) based on processing batch, rather than biologically relevant (and correct) differences between samples or experimental groups. Hence, having the ability to control for and differentiate batch effects from biological effect is critical. While a variety of approaches exist for scRNAseq data [24, 25, 30–32], approaches designed for cytometry data are less well developed. While there exist several approaches to align data from different batches, many of these align samples individually against each other in the context of standardized clinical profiling [33], potentially removing important biological variations between samples. In contrast, recent approaches calculate batch adjustments using reference controls: aliquots of a control sample run with each batch. The resultant conversions can then be applied to all samples in the batch, preserving biologically relevant differences [34]. Because some batch effects can differentially impact select cell lineages [35], further techniques (such as CytoNorm) extend this reference-based approach to apply adjustments in a cluster-specific manner [35]. This is a developing area and a number of other approaches have also been proposed, using a variety of methods to remove technical variance [14, 36], some inspired by approaches in scRNAseq.

While a number of clustering, dimensionality reduction, and batch alignment approaches exist, many of these are not directly integrated with each other – and many implementations require specific dataset formats, limiting the flexibility and interoperability of these tools. Specifically, tools that are developed in different fields (e.g., cytometry or scRNAseq) are often provided as stand-alone packages that operate on custom data formats (e.g., flowFrames [37], SCE objects, Seurat objects, etc.). The dependency of individual tools on custom data formats also makes it non-trivial to apply computational tools from a broader data science and machine learning field that are not specifically designed for cytometry (and thus have no knowledge of the custom data format) but are of significant advantage in existing analysis pipelines. A streamlined solution that can operate on large datasets at speed and integrate tools across fields in a modular fashion would be of significant advantage to the cytometry community.

1.3 | Spectre for analysis of large and complex HD cytometry datasets

To address these challenges, we developed Spectre, a computational toolkit in R that facilitates rapid and flexible analysis of large and

complex cytometry datasets. This toolkit expands on the "Cytometry Analysis Pipeline for large and compleX datasets" (CAPX) workflow that we have published previously for deep profiling of hematopoietic datasets [38]. Through specific function and workflow design, we demonstrate intuitive, modular, and high-speed workflows for data pre-processing (such as scaling/transformation), clustering, dimensionality reduction, plotting, as well as quantitative statistical analyses. Additionally, we incorporate an effective method for integrating data across multiple batches or experiments by extending the functionality of CytoNorm [35] batch alignment algorithm. Furthermore, we provide a means to transfer labels from reference datasets onto new datasets using machine learning classification algorithms, allowing for reproducible analytical workflows across multiple experiments. An extension of this process allows for the analysis of very large datasets, where the size of the dataset may exceed the memory capacity of the computer being used. Analysis with Spectre has been applied to flow, spectral, and mass cytometry datasets, consisting of tens to hundreds of millions of cells, in areas such as immune profiling in COVID-19 [39]. Additionally, Spectre has been applied to scRNAseq data and HD imaging data (such as Imaging Mass Cytometry, IMC). Spectre is available as an R package or Docker container from Github (https://github.com/immunedynamics/spectre).

Here we demonstrate the utility of the package by analyzing cells isolated from murine bone marrow (BM), spleen, and central nervous system (CNS) following inoculation with West Nile virus (WNV), measured by HD flow, spectral, or mass cytometry (CyTOF). Because of the significant cellular dynamics that occur in multiple tissues in response to infection with WNV, this provides an ideal model for demonstrating analysis workflows using Spectre.

2 | METHODS

Figure 1 illustrates an overview of analysis tools available in Spectre, and Supplementary Table 1 provides a summary comparison between Spectre and other existing packages (including CATALYST and Cytofkit).

2.1 | Sample preparation and acquisition

Ethical approval for the experimental use of mice was obtained from the Animal Ethics Committee at the University of Sydney. Briefly, mice were anesthetized with 250–300 μL of Avertin anesthetic via an intraperitoneal injection, and then inoculated intranasally (*i.n.*) with 10 μL of PBS or a lethal dose of WNV in sterile PBS (6×10^4 plaque-forming units [PFU]). After 1–7 days post infection (*p.i.*), mice received 350 μL of Avertin anesthetic, followed by vena cava section and left ventricular cardiac perfusion with 10–30 mL of ice-cold PBS. Spleen, BM, and CNS tissue were then extracted and prepared for flow/spectral [40] or mass cytometry [38] as previously described. Labeled samples were acquired on a 5-laser BD LSR-II, a 10-laser custom BD LSR-II, a 5-laser Cytek Aurora, or a Fluidigm Helios. Initial cleanup gating and compensation was performed using FlowJo v10.7, where single live leukocytes were then exported as CSV (scale value) or FCS files. We recommend cleaning data (e.g., compensation and live cell gating) with other software such as DIVA or FlowJo, and/or performing automated quality control processes (such as with flowAI [41]) prior to using Spectre.

2.2 | Data management in R via data.table

Data management and operations within Spectre were performed using the *data.table* format, an extension of R's base *data.frame*, provided by the *data.table* package [42]. This table-like structure organizes cells (rows) against cellular features or metadata (columns). This simple The advantage of the *data.table* is in high-speed processing, manipulation (subsetting, filtering, etc.), and plotting of large datasets, as well as fast reading/writing of large CSV files. While CSV files are the preferred file format in this context, Spectre also supports reading and writing FCS files through the use of *flowCore* [37], where the data are then converted from an FCS file into a *flowFrame*, and then into a *data.table*.

2.3 | Data pre-processing and transformation

In Spectre, the entire dataset to be analyzed is condensed into a single *data.table* (Figure 1(A)). Each row (cell) contains cellular expression data and metadata pertaining to the file, sample, group, and batch, stored in separate columns. Metadata for each of the columns (e.g., alternative cellular marker names, etc.,) in the *data.table* can be stored separately, if required. In order to reduce the contribution of background to measured signal, and to convert cytometry data into a linear space, we facilitated ArcSinh transformation of data using the *do.asinh* function in Spectre, as well as Logicle transformation ([43]) using the *do.logicle* function. This function allows a transformation cofactor to be specified for all columns (typical in mass cytometry) or for individual columns (typical in flow or spectral cytometry). The resulting transformed values are added to the dataset as new columns. This addition of new values, rather than replacement of original values, enables greater data retention and transparency. Additionally, an optional pre-processing step for re-scaling values to a new numerical range (such as a 0 and 1) is provided in the *do.rescale* function, similarly adding new columns containing the re-scaled values.

2.4 | Batch alignment and integration

To facilitate batch alignment in Spectre, we developed a wrapper for CytoNorm [35] using the *prep.cytonorm*, *train.cytonorm*, and *run.cytonorm* functions (Figure 1(B)). CytoNorm utilizes reference samples (aliquots of a control sample run with each batch) to determine batch-derived technical differences and calculates a quantile conversion model to align data from each of these batches. This

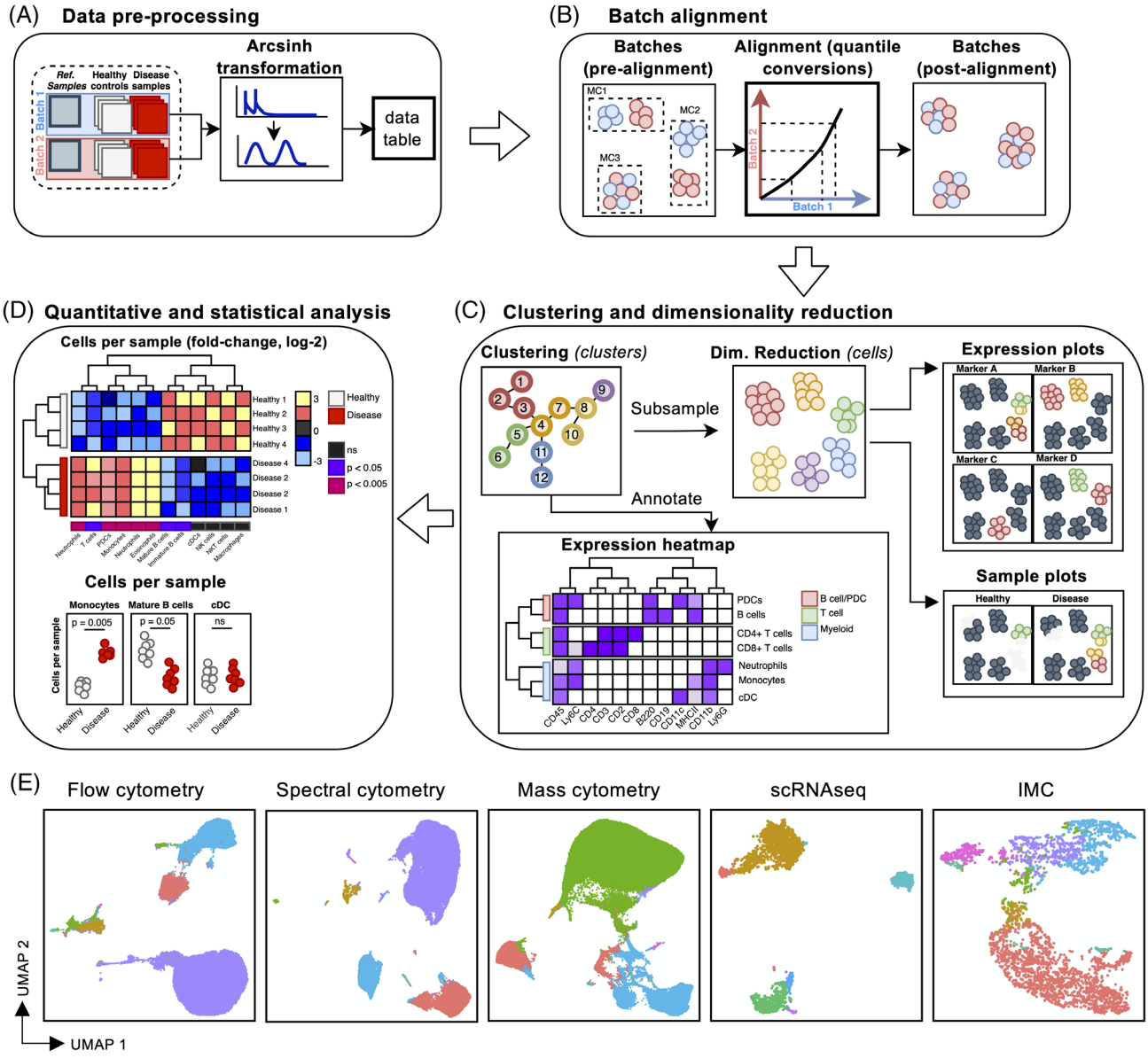


FIGURE 1 Spectre analysis overview. An overview of Spectre's analysis workflow. (A) Data preparation steps including sample, group, and batch annotation, in addition to ArcSinh transformation. (B) Batch alignment using CytoNorm. (C) Clustering and dimensionality reduction, along with marker expression plotting and expression heatmaps. (D) Quantification and statistical analysis through *z*-score/fold-change heatmaps and grouped dot plots. (E) Application of Spectre's analysis workflow to data generated by different technologies, including analyzing a split murine spleen sample by flow cytometry, spectral cytometry, and mass cytometry. Also shown are unrelated PBMC data analyzed by single-cell RNA sequencing (scRNAseq) acquired from 10X genomics, via the Seurat webpage (https://satijalab.org/seurat/v3.2/pbmc3k_tutorial.html) and imaging data generated by Imaging Mass Cytometry (IMC)

conversion model is then applied to samples in each batch, removing technical variation while preserving biologically relevant differences. The CytoNorm functions operate on a *data.table*, and allow for reference and target data to be easily specified in a dataset containing a mixture of batches. Quantile conversions for the entire dataset, or conversions for individual clusters, can then be calculated, resulting in a new set of "aligned" features that are compatible across the batches. Since raw, ArcSinh transformed, and aligned data are preserved in the *data.table*, transparency of these analytical processes is maintained throughout the analytical process. We also implemented alignment processes from CytofBatchAdjust [34] in the *run.align* function, but this has not been explored in this manuscript.

2.5 | Clustering of cells/events

Clustering was implemented using the *run.flowsom* function, which acts as a wrapper around the FlowSOM function (available as "FlowSOM" from Bioconductor [44]), which organizes acquired events (i.e., cells) into clusters using a self-organizing map (SOM), and

then groups these clusters together into "metaclusters" (Figure 1(C)). The function accepts a *data.table* and set of column names for generating clusters as input. Once clustering is performed, *run.flowsom* returns the *data.table* with new columns containing the cluster and metacluster IDs for each cell. Spectre's *run.flowsom* provides two options for specifying a target number of metaclusters. By default, FlowSOM will determine the number of metaclusters to generate by performing consensus hierarchical clustering on the clusters using a range of different metacluster numbers. It then computes the variance in each metacluster number and determines the point in which the variances suddenly decrease at a gentler rate (the elbow point), which is regarded as the optimal number of metaclusters for the dataset. We refer the reader to the FlowSOM publication for more details on how the optimum number is chosen [8]. Alternatively, users can manually specify a target number of metaclusters (which is recommended over automatic determination) or choose not to generate metaclusters at all. In addition to the ability to choose the number of metaclusters to create, the user is also able to change the SOM grid size (*xdim* and *ydim* parameters), the seed used to generate clusters and metaclusters for reproducibility.

2.6 | Dimensionality reduction

Non-linear dimensionality reduction (DR) was implemented in Spectre using the *run.tsne*, *run.fitsne*, or *run.umap* functions, which are wrappers around t-SNE (available as "rtsne" from CRAN [45]), FIt-SNE [46] (available from https://github.com/KlugerLab/FIt-SNE), and UMAP (available as umap from CRAN [47]), respectively (Figure 1(C)). All functions accept a *data.table* and set of column names for DR as input, and return the *data.table* with new columns containing tSNE, FIt-SNE, or UMAP coordinates for each cell. We also provide a function for running Principal Component Analysis (PCA), a linear DR approach, using the *run.pca* function, acting as a wrapper around *prcomp* (available in the "stats" package from CRAN [48]). The input data for *run.pca* can either be individual cells (to find cell markers that contribute to the variance), or individual samples/patients with summary data (such as median fluorescence intensity [MFI], cell counts or proportions). The output of *run.pca* is similar to *run.tsne*, *run.fitsne*, and *run.umap*, though additional outputs can be generated, including scree and contribution plots for detailed assessment of the source of variance in the dataset.

2.7 | Plotting and visualization

Plotting of cellular data was implemented in the *make.color.plot* function, serving as a wrapper around ggplot2 [49] (Figure 1(C)). The dataset, and desired columns to use as *X* and *Y* axis are taken as input. By default, each plotted cell is colored by relative plot density. Where desired, each cell can be colored by the level of expression of a specified marker (column), or color by some factor (e.g., cluster, sample, or group etc). To reduce the effect of outlier datapoints on the color scale, minimum and maximum thresholds are applied by default as 0.01 (1st percentile) and 0.995 (99.5th percentile), respectively, so that datapoints with values below or above these thresholds are colored at the minimum or maximum, respectively. These can be modified where required. In cases where a subset of the dataset is being plotted (e.g., separate samples or groups, etc.) the limits of the *X*, *Y*, and color parameters can be set by the complete dataset, allowing for the plots to be compared directly. By default, plots will automatically be saved to the current working directory, allowing for the rapid generation of plots. This plotting functionality is extended in the *make.multi.plot* function, which arranges multiple plots consisting of set of colored plots (e.g., for examining marker expression) or set of sample group divisions (e.g., for comparing changes between groups).

2.8 | Aggregation and summary data

A number of analysis steps require the aggregation or summary of data, either by sample, by cluster, or by both. To facilitate this, we implemented two functions – *do.aggregate* and *create.sumtable*. The *do.aggregate* function takes advantage of the fast aggregation function of *data.table*, providing the mean or median level of expression of each marker per cluster/population. This is helpful for plotting expression heatmaps, allowing for the interpretation of cluster/population identities. For every population in each sample, the *create.sumtable* function will calculate the percentage of each population as a percentage of cells in each sample (as well as cell counts for each population per sample if a total count of cells per sample is provided). Additionally, for all specified cellular markers, the median expression level, as well as the percentage of each population that expresses the marker are calculated for each population in every sample. This summarized *data.table* (in the format of samples (rows) versus features (columns)) can then be used for quantitative and statistical analysis, either in the same workflow, or can be saved to disk as a CSV file for analysis in a separate pipeline.

2.9 | Heatmaps

To facilitate the generation of heatmaps for examining marker expression on populations, or for comparing populations across samples, we implemented "*make.pheatmap*" as a wrapper around the pheatmap function from pheatmap [50] (Figure 1(C)). In *make.pheatmap*, the user must specify the dataset, the column that contains values to be used as the heatmap rows (e.g., sample or cluster identity), and the columns to be plotted as heatmap columns. This can be used to examine expression of marker (columns) on each cluster (rows), or can be used on z-score transformed data plotting immune features (columns) versus samples (rows), to reveal data variance for each cellular feature being measured.

2.10 | Group and volcano plots

To facilitate quantitative and statistical analysis, we implemented functions to create graphs (*make.autograph*) (Figure 1(D)) and volcano

plots (*make.volcano.plot*). Graphs are constructed using the *ggplot2* R package, and integrate pairwise statistical comparisons between groups using functionality from the *ggpubr* R package [51]. Pairwise comparisons between pairs of groups are performed using either Wilcox test (equivalent to a Mann–Whitney test), or a *t*-test. Overall group variance is assessed using a Kruskal–Wallis test, or a one-way ANOVA. Volcano plots were generated using the *EnchancedVolcano* package from Bioconductor [52], and customized to work seamlessly with *data.table* data structures. Initially the relative-fold change of values in each immune parameter between two groups are calculated, along with the *p*-value of that change. Each immune parameter can then be plotted as fold change versus *p*-value, allowing for a rapid assessment of changes with strong and significant effects between groups.

2.11 | Classification

Classification provides the means to transfer the annotation of a dataset to another dataset. In instances where cells have been annotated (e.g.,, with the cellular population they belong to) in one dataset, these labels can be automatically applied to cells in separate dataset, without having to repeat the entire analysis process. These instances might occur when seeking to replicate an analysis approach in a new dataset, if a dataset is acquired over time (such as time-course experiments), or if limited computational resources restrict the size of the dataset that can be analyzed. Spectre facilitates classification via the *run.knn.classifier* function. It runs a k-nearest neighbor (kNN) classification algorithm provided by the FNN package [53]. For ease of explanation, from here on, the term "labeled dataset" and "unlabeled dataset" will be used to refer to datasets in which the cells have and have not been assigned annotations, respectively. The phrase "labeled cells" and "unlabeled cells" will refer to cells in labeled and unlabeled datasets, respectively.

For each unlabeled cell, kNN identifies the k labeled cell(s) which lie the closest to it. The value of *k* can be any positive value, dictating the number of neighbors. We call these cells "nearest neighbors." The classifier then determines the most common annotation among the nearest neighbors, and assign that annotation to the unlabeled cell. kNN classifier resolves an unlabeled cell's nearest neighbors by computing the distance (euclidean) between it and all the cells in the labeled dataset. To ensure that all markers contribute equally to the distance calculation, and that both labeled and unlabeled dataset assume the same range of values, both the *run.knn.classifier* and *train.knn.classifier* (discussed below) functions re-scale both datasets to value range of between 0 and 1 at the beginning of the function.

The performance of a kNN algorithm is sensitive to the value *k*, the number of nearest neighbors considered when determining the annotation to assign to an unlabeled cell. A *k* value which classifies one dataset well may not do so on another dataset. To determining suitable *k* value, Spectre provides the *train.knn.classifier* function to assist in evaluation. *The train.knn.classifier* function assess the performance of a range of *k* values by evaluating each of it on the labeled dataset. To do this, the labeled dataset will need to be split into two portions. 1 portion (testing data) will be annotated by the kNN classifier (actual annotation for this portion will be hidden from the classifier), while the other portion will be used as the reference data to do the annotation (training data). Upon splitting the labeled dataset, the function will then choose a *k* value (from a range given by user), and instruct the kNN classifier to annotate the testing data using that *k* value. It will then compute the proportion of testing data which annotations assigned by kNN match the actual annotations (accuracy score). After testing all the *k* values, the function will collate the accuracy scores and return them to the user.

When splitting the labeled dataset into training and testing data, *train.knn.classifier* uses either train-test split or n-fold cross-validation (via caret R package [54]) approach. The first randomly split the labeled dataset into two equal halves while the latter split it into *n* portions, allocating one portion as testing data and the remaining as training data. Unlike the train-test split which uses the same training and testing data for all *k* values, n-fold cross-validation technique trains each *k* value *n* times, each using different portion(s) for training and testing data. While n-fold cross-validation technique more comprehensively tests the performance of the kNN classifier, it requires significantly more time to run. For this reason, we recommend the use n-fold cross-validation only for when there is very little labeled data available. The parameter *method* in the *train.knn.classifier* function determines the approach used for splitting the data, and is where the train-test split approach is default. We note that for n-fold cross-validation, the accuracy score for each *k* value is the average accuracy score across all combinations of training and testing data.

3 | RESULTS

3.1 | Spectre facilitates comprehensive end-to-end integration and analysis of large HD cytometry datasets

Spectre was developed to facilitate rapid and flexible analysis of large and complex cytometry datasets across multiple batches or experiments. Specifically, Spectre facilitates data pre-processing (Figure 1(A)), alignment of data from multiple batches/experiments (Figure 1(B)), clustering (Figure 1(C)), dimensionality reduction and visualization (Figure 1(C)), manual or automated population classification and labelling (Figure 1(C)), as well as extensive plotting and graphing options for qualitative and quantitative statistical analyses (Figure 1(D)). Key to this process is strategic selection, implementation, and customization of high-performance computational tools; and the development of wrapper functions around these tools, enabling them to operate on and produce the same data format for input and output, respectively. This allows for multiple analysis and plotting/graphing tools to be seamlessly woven together into single analysis workflows, where functions can be used throughout the stages of the workflow, drastically increasing ease of use. Moreover, this modularity and flexibility allows these workflows to be adapted to meet different experimental

requirements or analytical approaches. Spectre can be used on data generated by a variety of single-cell technologies, including flow, spectral, and mass cytometry (Figure 1(E)). In addition, following some additional pre-processing, Spectre can also be used for the analysis of data generated by scRNAseq or HD imaging technologies such as IMC (Figure 1(E)). In the case of scRNAseq, users may wish to convert data from formats like Seurat or SCE in a simple table format using data.table, allowing them to take advantage of specific clustering, processing, plotting, or other analysis processes, using Spectre as well as other generic analysis of plotting tools (some examples provided in Supplementary Figure 1). To facilitate this, we created a function to extract all information from Seurat, SCE, or flowFrame object and structure it as a *data.table*, using *create.dt*.

3.2 | Data management and pre-processing

Critical to our approach is the use of the *data.table* data structure and operations, in place of prevalent FCS/flowFrame objects or other data structures. The *data.table* format is an enhancement of base R's *data.frame*, a table-like structure commonly used to store data. This enhancement allows for fast and efficient aggregation and manipulation of data through the use of concise flexible syntax and low-level parallelism [42] (Supplementary Figure 2). Using Spectre, all samples in the analysis are merged into a single *data.table*, with relevant sample, group, and batch information stored in separate columns. As a result, each row contains all the information relevant for a particular cell, making data manipulation and filtering with *data.table* simple and fast.

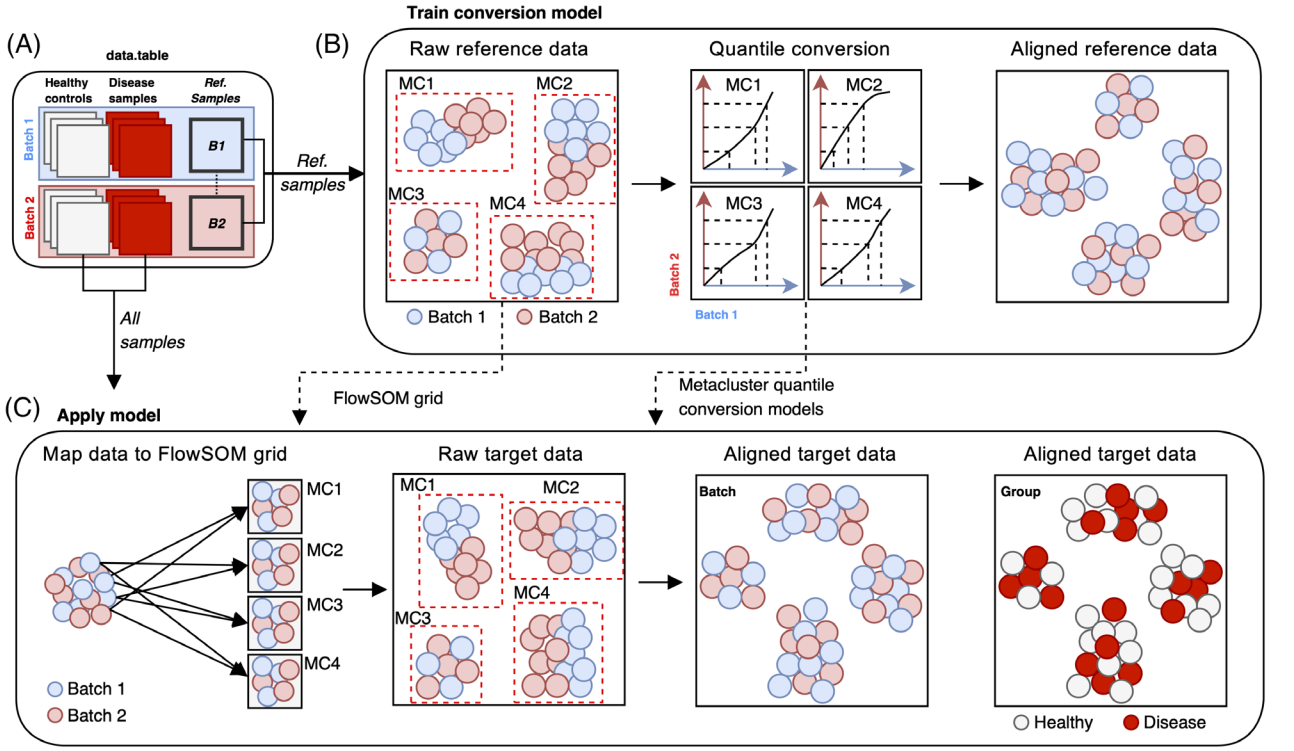
A key initial step in computational analysis is the transformation of cellular expression data. Biologically meaningful results are most easily interpreted through plotting cellular expression on a logarithmic scale. Because of potentially misleading visual artifacts for signals at the low end of the logarithmic scale, the logicle/bi-exponential scale was developed, where the high end of the scale is logarithmic and the low end of the scale is converted into a linear scale, and the scale then returns to logarithmic at values below the linear component (Supplementary Figure 3(A)) [43, 55]. Critically, this allows for the compression of low-end data points with high spreading error, autofluorescence, or noise into a linear space around zero, which can be tailored for the requirements of each channel (Supplementary Figure 3(B)). For computational analysis to meaningfully manage biological data, a similar compression of low-end data needs to be performed. In cytometry, this is commonly performed using the ArcSinh (http://mathworld.wolfram.com/InverseHyperbolicSine.html) transformation [56]. The data values are transformed into a format that can be viewed on a linear scale, where compression of low-end values is determined using a specified co-factor to determine the extent of compression around zero (Supplementary Figures 3(B) and 4).

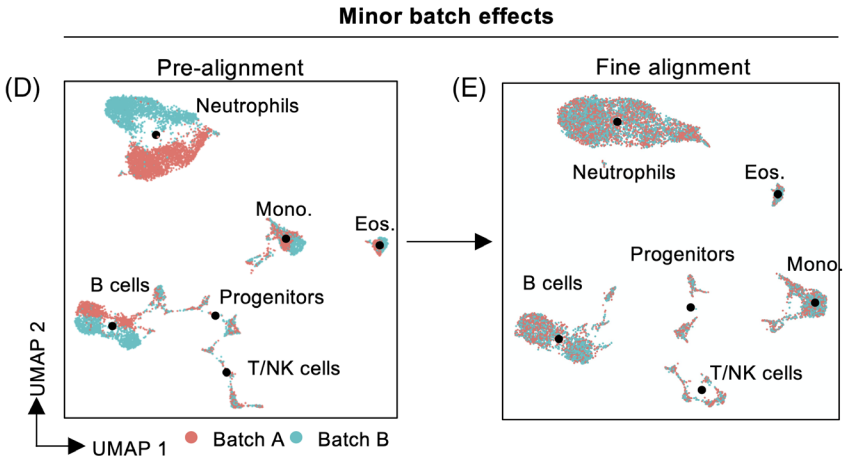
Using Spectre, ArcSinh transformation was applied to data from each channel/marker individually, using different co-factors, allowing for highly customizable data transformations. For flow, spectral, and mass cytometry data, we tested various co-factor values across multiple channels (Supplementary Figures 5 and 6). Overall, we found that a co-factor of 5–15, was suitable for all mass cytometry channels. However, in our experience, we found a range between 100 and 10,000 to be suitable for different channels in conventional or spectral flow cytometry data.

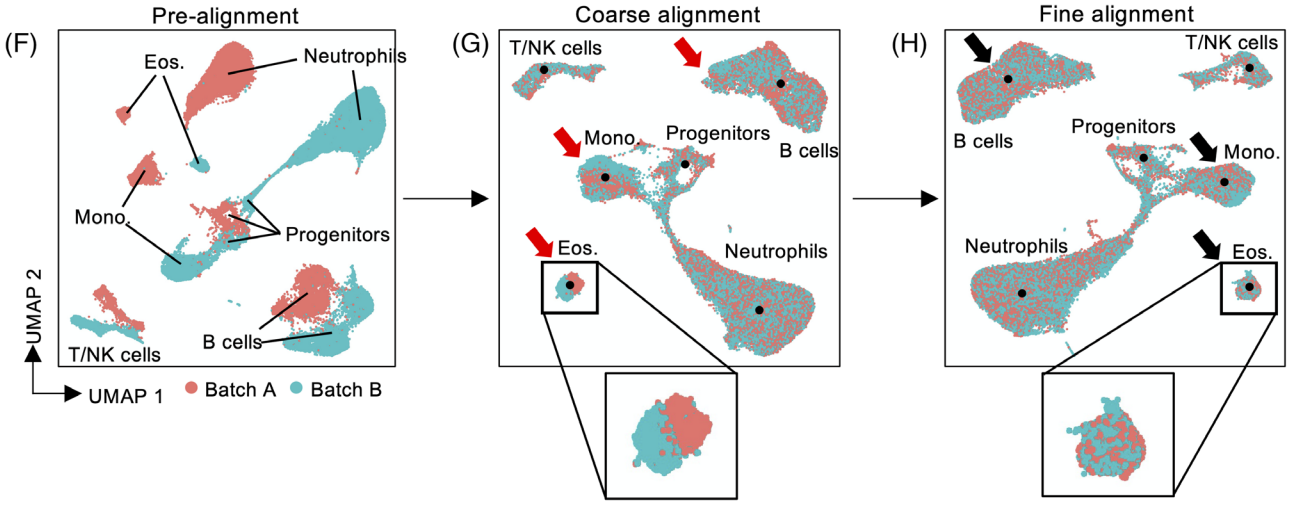
We developed a function for re-scaling each marker in the ArcSinh-transformed data to range between two new values, usually 0 and 1 (Supplementary Figure 7). This may prevent markers with extremely high expression levels from exerting greater influence over clustering and DR results, when compared to other markers (Supplementary Figure 8). This is implemented in the *do.rescale* function. Although we have demonstrated the utility of the *do.rescale* function in Supplementary Figure 7, this was not used for the rest of the data presented in this paper. Additionally, we emphasize that it is not mandatory to use this function when pre-processing data, and data scaled externally (such as with FlowJo) can still be used with any of Spectre's functions. If using do.rescale, we recommend applying it prior to data alignment, clustering, and DR for maximum benefit. We also provided a clipping function *do.clip* that will convert all values above or below a specified value to that value, which maybe helpful in the management of outliers. For example, changing all values below 0, to 0 (Supplementary Figure 7).

3.3 | Integrating data from multiple batches or experiments into a single feature space

When samples are prepared, stained or run in multiple batches, technical batch-effects can occur, usually consisting of shifts in signal intensity in one or more markers. Because of this, data clustered together with uncorrected batch effects may separate samples based on the batch they belong to, a confounding factor that substantially hinders aggregated analysis of datasets from multiple batches or experiments. To provide a comprehensive and adaptable batch alignment and data integration approach, we expanded on the functionality of CytoNorm within Spectre. Users specify reference control samples that Spectre uses to determine the alignment conversions. Typically, these are aliquots of a single "healthy" patient sample that are run with each batch of samples. These reference samples should span the full range of the data seeking to be aligned. Where marker expression data is absent on the reference controls (e.g., absence of activation markers on cells from healthy donors), alignment is not feasible, but the original expression data for those markers can still be analyzed. In cases where multiple aliquots of a single control sample are not available, multiple control samples of the same type (e.g., healthy mouse BM) that are run with each batch can be used as the next best. Finally, it is also possible to use all samples from each batch as 'reference' controls, taking the entire range of marker expression from each batch into account. However, these later approaches should be used with caution, as any biological differences between the samples used for reference will be interpreted as technical variation due to batch effects. To execute this alignment process in Spectre, user-indicated reference samples are extracted from the combined *data.table* (Figure 2(A)) and clustered using FlowSOM. For


Train conversion model
(A)
data.table
Healthy controls
Disease samples
Ref. Samples
Batch 1
Batch 2
B1
B2
Ref. samples
All samples
(B)
Raw reference data
Quantile conversion
Aligned reference data
MC1
MC2
MC3
MC4
Batch 1
Batch 2
FlowSOM grid
Metacluster quantile conversion models
(C)
Apply model
Map data to FlowSOM grid
Raw target data
Aligned target data
Aligned target data
Batch
Group
Healthy
Disease


Minor batch effects
(D)
Pre-alignment
(E)
Fine alignment
Neutrophils
Eos.
Mono.
B cells
Progenitors
T/NK cells
UMAP 1
UMAP 2
Batch A
Batch B


Major batch effects
(F)
Pre-alignment
(G)
Coarse alignment
(H)
Fine alignment
Eos.
Neutrophils
Mono.
Progenitors
B cells
T/NK cells
UMAP 1
UMAP 2
Batch A
Batch B

each resulting metacluster, quantile conversion coefficients between each batch for each marker are calculated (Figure 2(B)). Cells from the full dataset are then mapped to the FlowSOM grid, and each cell is assigned to its nearest metacluster (Figure 2(C)). Quantile conversion is then applied to the cells in each metacluster, unifying cells from each batch into a single feature space, while preserving biological differences between experimental groups (Figure 2(C)).

To verify the robustness of this process, we applied batch alignment to a set of mock- or WNV-infected BM samples, where synthetic data manipulations were introduced in a population-specific manner to half the samples, to mimic population-specific batch effects. One healthy BM sample from each "batch" were selected as a reference controls, aggregated together, and plotted using UMAP, revealing subsets of neutrophils, eosinophils, monocytes, B cells, T/NK cells, and progenitors. Batch-specific differences in their distributions were evident in the UMAP plots (Figure 2(D)). When FlowSOM was run, consistent populations from each batch were captured within the same metacluster, despite the presence of batch effects. As such, metacluster-specific alignment with CytoNorm was able to adequately integrate the cells from each population into a unified dataset (Figure 2(E)).

To test this process on datasets with more substantial batch-effects, we applied batch alignment to two sets of mock- or WNV-infected BM samples, prepared and acquired months apart, using slightly different panels. After matching column names between the two datasets, the disparity of these two datasets was apparent upon plotting with UMAP (Figure 2(F)). In this scenario, the batch effects were so significant that cells from the same population in each batch did not map to the same metacluster, thus preventing alignment. To address this, we initially performed a "coarse" alignment by mapping all cells from each batch into a single metacluster and performing quantile alignment on the entire dataset. This corrected the majority of batch effects, though some metacluster-specific effects were still evident (Figure 2(G), inset). Nevertheless, the resulting data was sufficiently aligned so that populations from each batch could now be mapped into consistent metaclusters (Figure 2(G)), allowing for a more fine-tuned alignment (Figure 2(H)) that corrected residual metacluster-specific batch effects (Figure 2(H), inset) and unified the dataset into a single feature space.

3.4 | Clustering and dimensionality reduction strategies

A critical analytical step in HD cytometry is the comprehensive identification of cellular populations, including those that are well-established and those that are yet to be characterized. This is particularly relevant when comparing between experimental groups (e.g., diseased patients compared to healthy controls). Clustering and dimensionality reduction are powerful techniques for identifying cell populations and contrasting them between groups. Clustering tools collect phenotypically similar cells into groups (clusters) in a data-driven fashion (i.e., cells are grouped together based on the similarity of their marker expression). The output of many clustering approaches are plotted as a collection of nodes, where each node represents a cluster that contains a number of cells. These nodes are typically connected by a form of minimum spanning tree (MST) [8] and colored by mean or median marker expression of the cells within each cluster. However, verifying the phenotypic heterogeneity (or homogeneity) of cells captured within a cluster is difficult when looking at the data at the cluster level. An alternative approach is to compress cellular data onto two dimensions using non-linear dimensionality reduction tools such as t-SNE [10, 11], FIt-SNE [46], and UMAP [12], and to visualize these on a scatter plot, coloring each cell by marker expression level or cluster/metacluster ID.

Spectre supports clustering and DR by providing a wrapper function for FlowSOM, Phenograph, t-SNE, Fit-SNE, and UMAP. These functions accept and return data in the *data.table* format. Whilst clustering tools such as FlowSOM scale well to large datasets [57], some DR approaches can incur lengthy computing times, excessive memory usage, and significant crowding effects that inhibit their utility (Supplementary Figure 9(A)). Whilst some improvements to runtime (FIt-SNE [46], Supplementary Figure 9(B)) and plot crowding (opt-SNE [58]) have been made, scalability and plot crowding limitations persist. As DR tools are primarily used to visualize cellular data and clustering results, we propose plotting only a subset of the clustered data, which addresses scalability and retains legibility. By using proportional subsampling from each sample, the relative number of cells from each cluster in each sample can be preserved in a smaller dataset, allowing for interpretable analysis via DR. Putative cellular populations among the clusters can then be identified, and annotated in both the subsampled DR dataset, as well as the whole clustered dataset. The whole annotated dataset can subsequently be used in downstream quantification and statistical analysis. However, we note that it is possible for rare cell populations to be obscured when visualizing data in this manner, and that it is important to specify sample size which maintain some balance between rare and common cell populations. Additionally, Specre's *do.subsample* function permits users to specify different sample size for every discrete category in the data (cell populations, group, batch, sample, clusters), thus allowing disproportionate subsampling.

FIGURE 2 Batch alignment using CytoNorm. Batch alignment process using CytoNorm. (A) Reference samples acquired with each batch are extracted from the *data.table*, and (B) clustered using FlowSOM. Metacluster-specific quantile conversion models are then calculated. (C) Cells from all samples/batches are mapped to the FlowSOM grid, and assigned to their nearest metacluster. Cells are then aligned using the metacluster-specific quantile conversion models (D) Two sets of BM samples with synthetic batch effects introduced in a population-specific manner. (E) CytoNorm alignment was performed using a metacluster-specific alignment process (fine alignment). (F) Two sets of BM samples generated with slightly different panels, but targeting the same cellular markers, resulting in significant batch effects. (G) CytoNorm is initially performed on the whole dataset (coarse alignment) by mapping the entire dataset into a single metacluster, where H) subsequent FlowSOM clustering allowed for further metacluster-specific alignment (fine alignment)

To demonstrate this strategy, we applied clustering and DR to a dataset of brain cells from mock- or WNV-infected mice (Figure 3(A)). The whole dataset was clustered using FlowSOM (Figure 3(B)), and the clustered data were proportionally subsampled for plotting with UMAP (Figure 3(C)). In this case, the number of cells extracted from each sample were in proportion to the total cells recovered from each brain, ensuring that DR plots accurately reflected sample composition (Figure 3(D)). By examining expression heatmaps (Figure 3(E)) and colored DR plots (Figure 3(F)) we manually determined cluster population identities, and these annotations were applied to both the subsampled and full datasets.

The choice of markers used to inform the generation of clusters/DR results is dependent on the overall goal of analysis. Cellular markers may be broadly categorized into two groups: static (stably expressed) or dynamic (changing expression). Typically, statically expressed markers are helpful for identifying consistent cell *types*

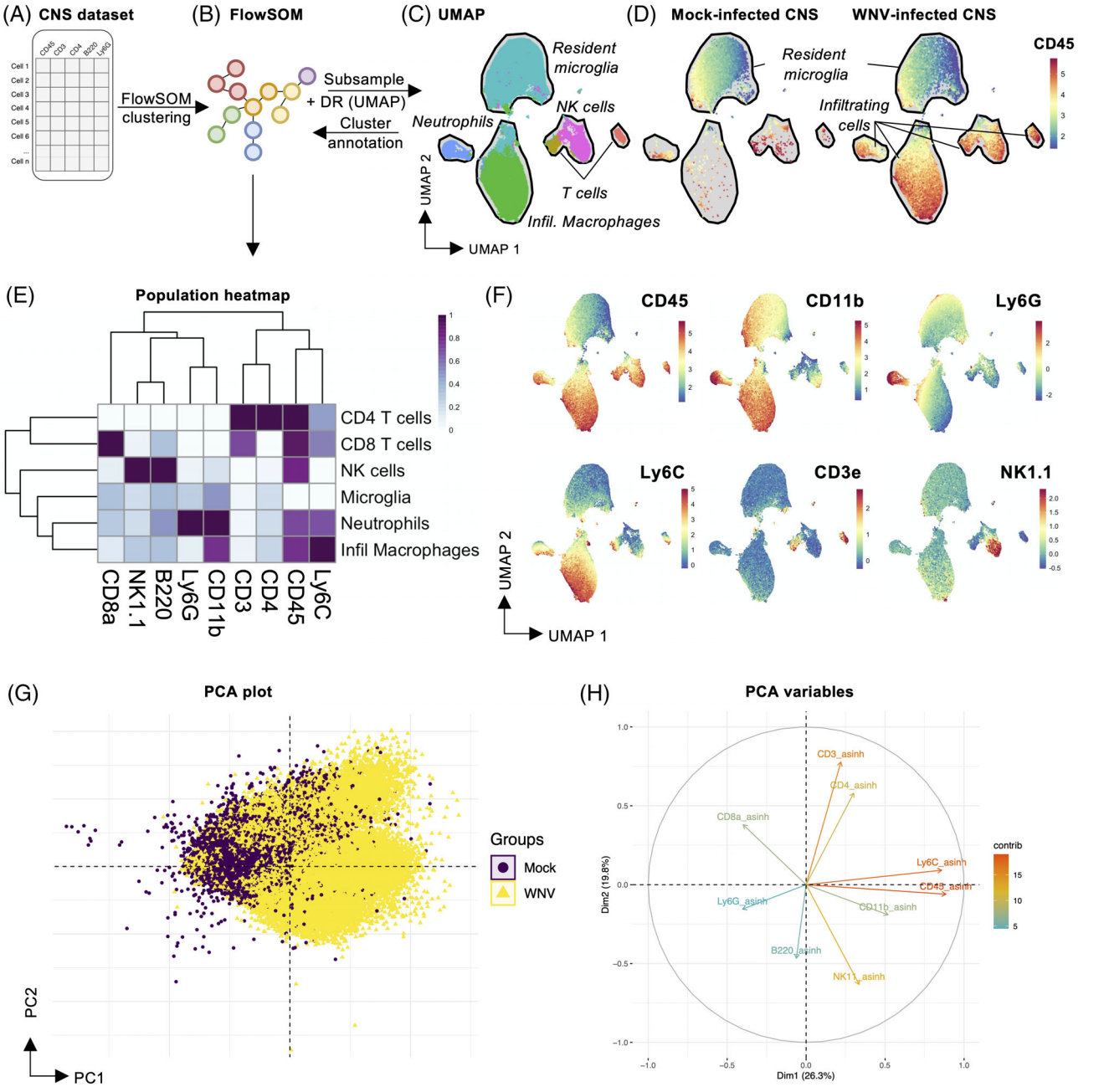


FIGURE 3 Clustering and dimensionality reduction using Spectre. (A) A dataset of cells isolated from mock- or WNV-infected CNS were used to demonstrate clustering and DR in Spectre. (B) FlowSOM clustering was performed on the full dataset, which was C) then subsampled and plotted using UMAP. "Infil. Macrophages" refers to infiltrating macrophages, as distinct from resident macrophages (microglia). (D) Parsing the dataset by each experimental group reveals substantial changes to immune populations. (E–F) An examination of marker expression on each cluster allows for a user-determined annotation into biological relevant cell types. (G) Analysis using PCA allows for visualization of the data variance, and (H) the relative contribution of markers to the first two principal components

(e.g., T cells, B cells, etc.), whereas dynamic markers are helpful for identifying reactive cellular *states* (e.g., activated, resting, etc.). When seeking to discover novel cellular populations or states, incorporating all cellular markers may be of benefit, as both stable cell types and dynamic cellular states will be captured in separate clusters. However, a more selective approach may also be desired, such as using statically expressed markers to capture known populations within clusters, and then examining those stable populations for dynamic changes in activation status. Along with domain-specific knowledge about the expression patterns of various markers, it is possible to identify markers that contribute most to the level of variance across a given dataset. To do this we can using PCA, (Figure 3(G)–(H)) to determine the relative contribution of each cellular marker to the overall variance of the dataset.

3.5 | Multi-level immune profiling

Many populations of interest, such as hematopoietic stem cells (HSC) in the BM, are of very low frequency within individual samples. As such, representation of these rare subsets may be extremely sparse on DR plots, relative to more abundant populations. Moreover, more nuanced clustering and analyses of such populations are often desired, but the global data structures provided by more abundant subsets of cells may dominate the analysis. To address this, we extended our analysis approach to enable the exploration of data at multiple levels. Upon clustering the complete dataset, clusters representing rare or novel populations can be isolated and re-clustered independent of other populations and thereafter annotated in greater detail. Expanding on this approach, multiple lineages can be split and profiled independently, then re-merged, retaining detailed cluster annotations for combined plotting and quantitative analysis. This process is conceptually similar to hierarchical approaches such as hierarchical-SNE (h-SNE, [59]), but accommodates a more bespoke tailoring of the analytical process.

To demonstrate this, we examined a dataset of BM HSCs generated by mass cytometry. Clusters containing HSC and progenitors (denoted by CD117 expression) (Figure 4(A)) were extracted from the full dataset, and subjected to independent clustering and DR (Figure 4 (B)). This independent analysis allowed for a more detailed assessment of low frequency subsets (Figure 4(C)) that were not easily assessed in the plots from the full dataset.

3.6 | Automated cellular classification and label transfer between aligned datasets

A crucial application of computational analysis is *discovery* – defining novel subsets and/or investigating experimental changes in novel subsets or states in new diseases, tissues, or experimental conditions. Analytical approaches to this often depend heavily on unsupervised techniques, such as clustering and DR. Such analyses culminate in an annotated datasets in which each cell is manually assigned a

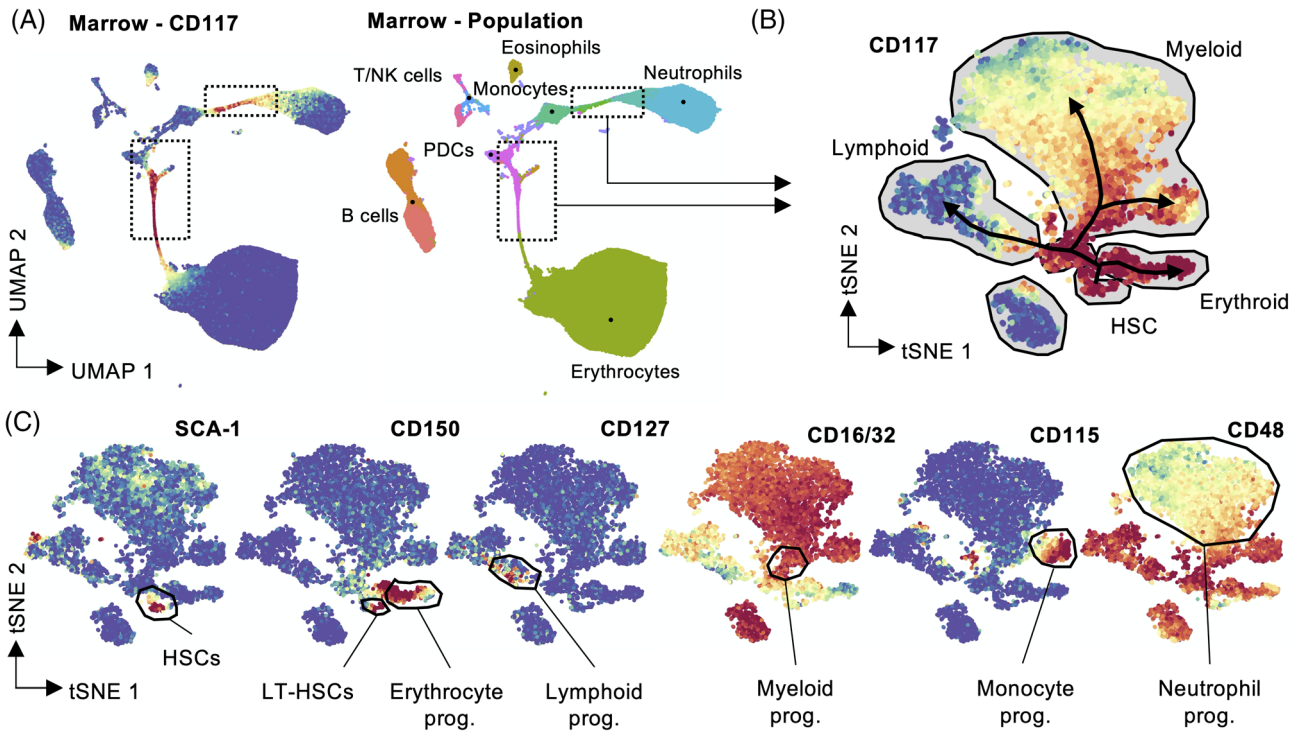


FIGURE 4 Multi-level analysis for profiling of rare populations. (A) A UMAP plot (left) where clusters representing stem cell and progenitor subsets were identified via CD117 expression. Through cross-referencing against FlowSOM clusters (right), these cells were (B) subjected to new clustering, subsampling, and plotting using UMAP. (C) Expression color plots reveal low frequency cellular subsets that were difficult to otherwise detect on the full UMAP plot

population label by the user. Alternatively, other studies seek to apply a more *repetitive* analytical process, using semi-supervised tools, such as automatic gating, to replicate a method of population identification over a large number of samples. While effective, many of these tools rely on some form of gating strategy and forgo the use of any unsupervised techniques to expedite gating, thereby limiting the possible identification of new or complex/overlapping populations. In contrast, machine learning-based approaches provide the opportunity for automated transfer of cellular labels from an annotated dataset to a novel dataset, following alignment of the datasets into a single feature space (as demonstrated in Figure 2). To facilitate this, we provided functions within Spectre to train and run classifiers, a type of machine learning approach designed to predict the class of given data points. As opposed to clustering, which groups cells together based on marker expression, classifiers assign cells a label based on "training" data. This training data could be previously gated, clustered, or annotated and is used by the classifier to determine how given features (marker expression) relates to a class (cluster or cell phenotype).

Inspired in part by Seurat's mutual nearest neighbors approach for data integration [24, 25], we reasoned that a simple nearest neighbors approach could facilitate rapid label transfer between datasets. In Spectre, we implemented a kNN classifier, which classifies unlabeled cells based on the label of their k nearest neighbor within the training dataset. To determine the accuracy of the kNN classifier, we evaluated the classifier on a labeled dataset containing 169,004 cells using train-test split – one half retained cellular labels and served as the training dataset, and the other half had their cellular labels hidden, and served as the testing dataset (Supplementary Figure 10(A–C)). We investigated k value between 1 and 30, and found kNN's accuracy in general to be excellent (>98.5%), peaking at 98.97% when $k = 9$, and slowly decreasing thereafter (Supplementary Figure 10(D)).

To demonstrate the kNN classification process in an experimental context, we aligned two datasets of BM cells using CytoNorm (shown in Figure 2(D)–(E)), where the first dataset (batch A) contained labeled cells, and the second dataset (batch B) did not (Figure 5(A)). Following alignment, the data was split into the labeled and unlabeled datasets

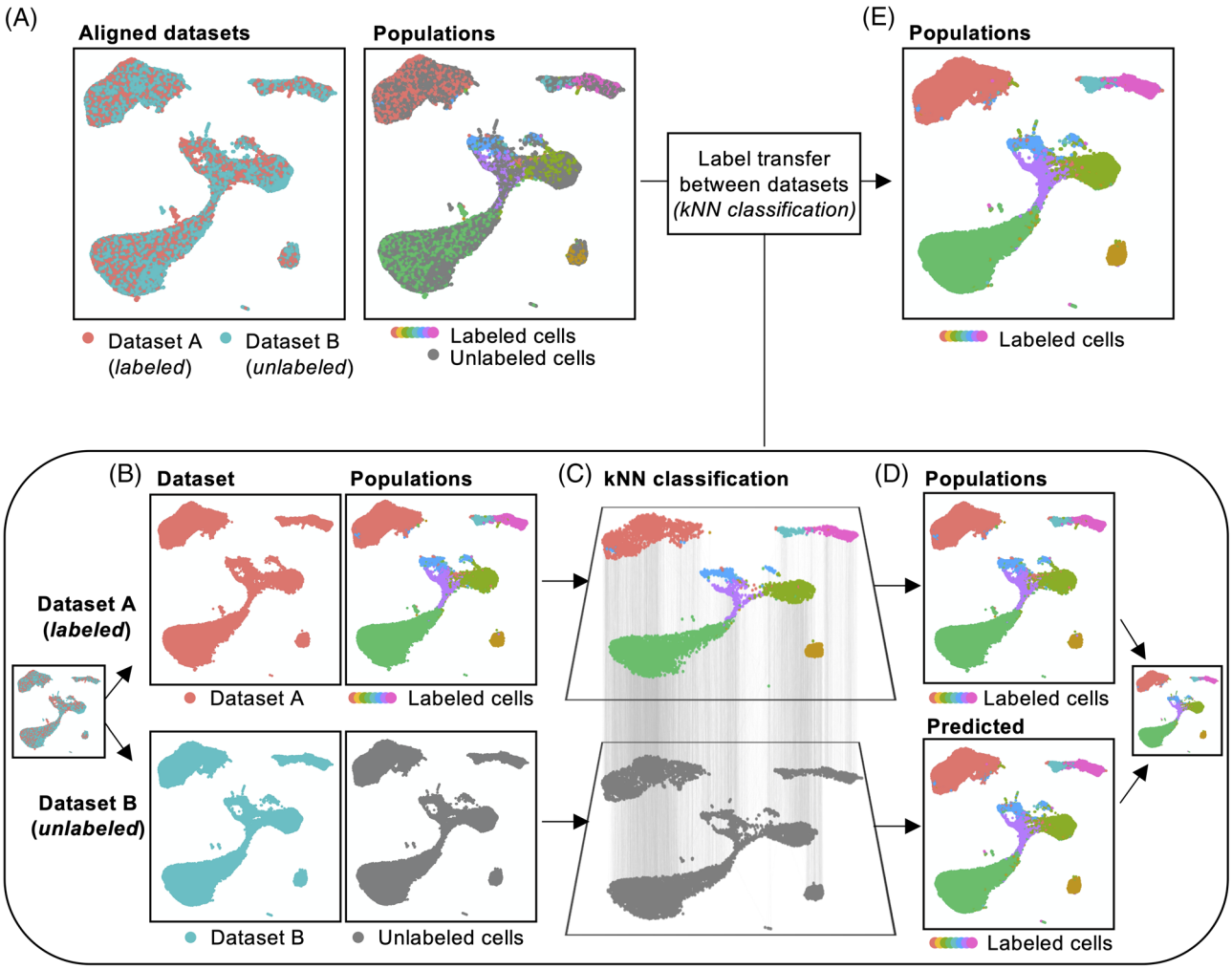


FIGURE 5 Cellular classification and label transfer. (A) A dataset of two batches of BM cells, following alignment with CytoNorm. One batch contains annotated clusters, and the other does not. (B) The dataset is split into each batch (labeled and unlabeled), and a kNN classifier trained on the labeled dataset with $k = 1$. (C) The kNN classifier was then applied to the unlabeled dataset, (D) resulting in an accurate transfer of cellular labels between datasets

(Figure 5(B)), where the kNN classifier was trained on the labeled cells (batch A) and applied to unlabeled cells (batch B) with $k = 1$ (Figure 5(C)). When we compared the predicted cellular labels to manually annotated cellular labels, we found the classifier was able to accurately transfer cellular labels between the two datasets (Figure 5(D)). Merging of the two datasets resulted in a fully annotated dataset (Figure 5(E)).

3.7 | Quantitative and statistical analysis

The endpoint of most analytical workflows is to make quantitative and statistical comparisons between experimental groups. Many components in this workflow can be automated to simplify the processing steps and reduce the time taken to generate relevant statistics and plots. To facilitate this, we have developed a series of functions to summarize a dataset rapidly at either the cluster or cellular population level, resulting in a series of summary tables. For each population in each sample, these tables summarize the proportion of cells, total cell counts, marker expression levels, and proportion of cells that are 'positive' for each cellular marker, which can then be used to generate quantitative plots. The generation of grouped scatter or violin plots (Figure 6(A)) provide a simple method to assess changes of a single feature (e.g., number of infiltrating macrophages per brain) between experimental groups, including grouped or pairwise statistical comparisons. However, more global statistical analyses are often desired. The generation of *z*-score heatmaps (Figure 6(B)) provide an overview of relative changes between samples, with optional clustering on samples (rows) or features (columns) based on similarity. Additionally, the result of pairwise comparisons between groups can be indicated for each heatmap column, revealing statistical significance for uncorrected or false discovery rate (FDR)-corrected *p*-values. Furthermore, PCA plots (Figure 6(C)) and volcano plots (Figure 6(D)) provide a further global view of how these immune features differentiate samples within the experimental context.

4 | DISCUSSION

4.1 | Challenges in cytometric analysis

The rapid increase in the complexity and size of cytometry data has made traditional analysis by manual gating untenable. This has led to a myriad of computationally-driven analysis approaches (automated gating, clustering, DR, and classification) being developed. However, these approaches come with limitations, in particular the lack of interoperability between them. Computational-based analysis tools are often developed as standalone packages which operate on specific data formats, making it difficult to combine them in a single analysis pipeline. As a solution, we developed Spectre: an adaptable and easy-to-use package for analyzing HD cytometry data. Spectre enhances existing computational tools through strategic implementation and customization of high-performance tools, and provision of wrapper functions to simplify and improve their flexibility and interoperability. This package expands on many of the key aspects of the CAPX workflow [38] that has previously been utilized in a number of studies [60–64].

4.2 | As the foundation for data in Spectre, data.table allows for easy handling of large datasets

Many cytometry or single-cell analysis tools operate on custom data formats, such as the flowFrame [37], (SCE) [26], or Seurat objects [24]. Each custom data format may contain numerous elements, made up of both base R and custom data formats. Typically, primary cellular expression data is stored in a table or matrix, and separate elements contain metadata for each cell (cell number, sample, group, etc.,) or feature (marker names, parameters names, voltages, gene numbers, etc). Importantly, manipulated data (such as ArcSinh and logicle transformations or normalization) may be contained within the primary data table/matrix, or structured separately. The complexity of these custom formats, and the significant differences in structure between them, results in difficulties in converting one data format to another. While custom data formats may be convertible into other formats, including conversion to one of R's base data formats, these conversions are non-trivial, and often result in the loss of important metadata. Of note, cytometry data is typically structured with rows representing cells, and columns representing cellular features, though this is transposed in scRNAseq data.

The foundation of Spectre is built upon the *data.table* package, which enhances R's base *data.frame* format. The efficiency of *data.table* in performing basic data manipulation operations, such as filtering, ordering, importing, and exporting, makes it suitable for processing large HD cytometry datasets. While there are R packages which operate on *data.frame* formats (e.g., dplyr [65]), our simple benchmarking measurements show *data.table* to be faster when handling large datasets (Supplementary Figure 1). The simple tabular format allows for interoperability between Spectre, basic R functions, and other functions from cytometry or single-cell-specific packages, by storing all the relevant information for each cell in a single, high-performance table. Here, columns desired for use with each function can be easily specified, and new columns that are added as a result of analysis are easily identifiable through the use of regular expression patterns (such as "CD4_asinh," "CD4_aligned," etc.). Column metadata in this context is less relevant for processing and analysis, but can still be imported and managed with the main data should the user choose to do so. By designing all our functions to operate on this simple *data.table* structure, we remove the requirements for users to convert their data into specific formats for different functions, greatly improving usability.

Functions in Spectre that perform data transformation, batch alignment, clustering, and DR append results as separate columns to the main dataset. In the rare occasion where results require more memory than what is physically available (e.g., when processing a very

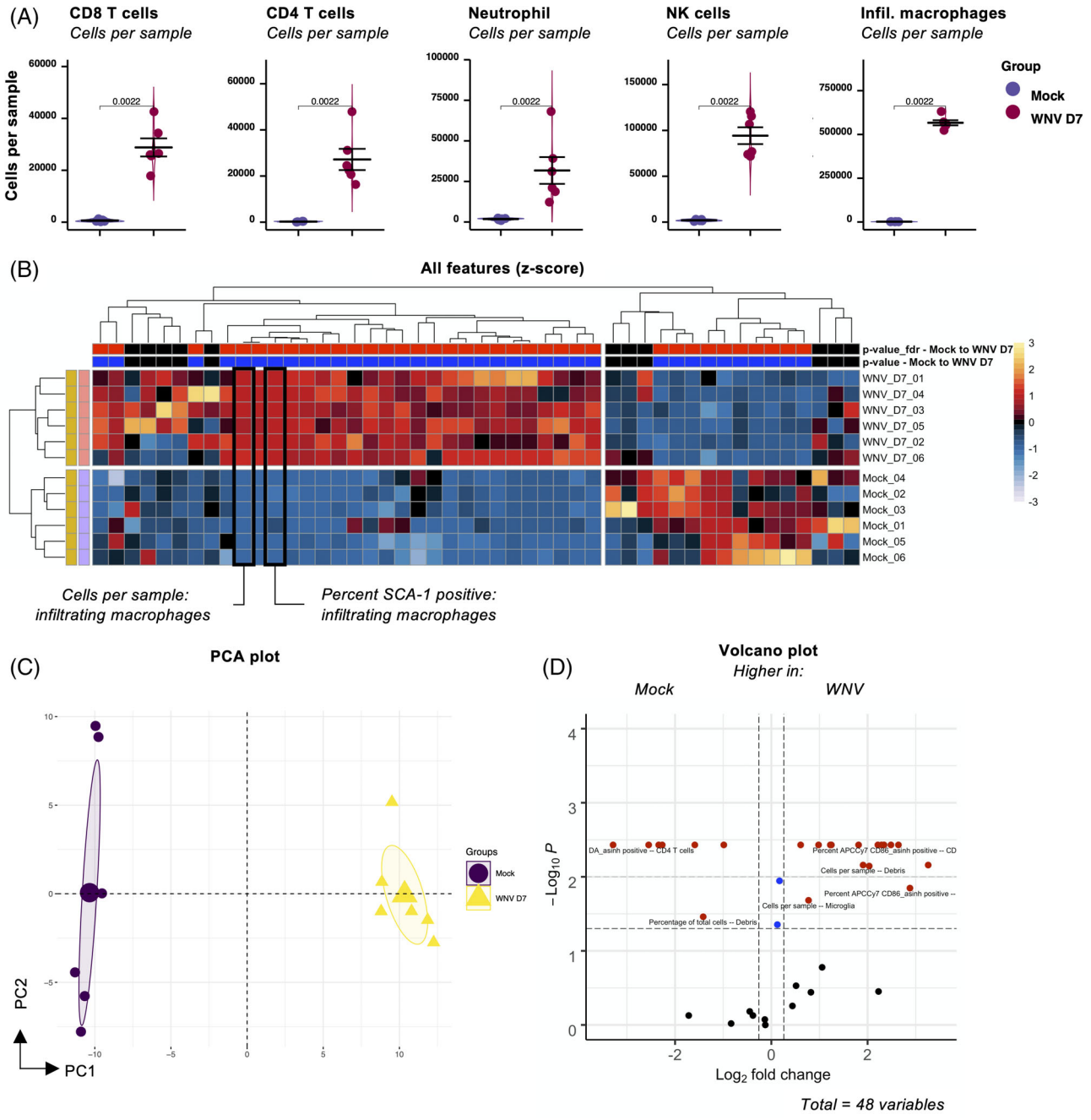


FIGURE 6 Quantification and statistical analysis. Quantitative and statistical analysis of CNS samples. (A) Grouped dot plots with underlying violin plots indicating the relative cells per sample for each population in the dataset. Pairwise statistical analysis for non-parametric data was performed using a Mann–Whitney/Wilcox test, where $p < 0.05$ was considered significant. Importantly, data for each plot were not adjusted for multiple comparisons. (B) The *z*-score for each column of data was calculated, and plotted using the make.pheatmap function. Each row represents a sample, and each column represents a measured immune feature. Rows and columns were clustered using Euclidean distance. Pairwise comparisons between experimental groups were calculated using a Mann Whitney/Wilcox test, and the significance results plotted as significant ($p < 0.05$) or non-significant ($p > 0.05$) for each column. To adjust for multiple comparisons, p-value results were corrected using a FDR correction, and results plotted as significant ($p < 0.05$) or non-significant ($p > 0.05$) for each column. (C) PCA results show the distribution of each sample, and (D) volcano plots show the relative fold-change increase or decrease of each immune feature (*X* axis), and the inverse *p*-value of each change (*Y* axis)

large dataset), users may periodically export the existing results to a CSV file using either Spectre's *write.files* function or *data.table*'s *fwrite* function, discard columns which are not immediately required, and import them back in using *data.table*'s *fread* function when required. This will free up more memory resource for subsequent experiment without incurring any loss in flexibility and transparency.

4.3 | Usage flexibility

We have demonstrated the use of Spectre through various workflows. While they each illustrate an approach to analysis, they are flexible in such a way that functions can be run in any order, and can be replaced with others (including those that are not built into Spectre). For instance, it is possible to run a DR tool on the dataset before clustering, or replace Spectre's run kNN classifier function with another function (like a Decision Tree classifier [66]). For the latter, an additional step may be required to convert the *data.table* to another format that is accepted by the function. This is trivial for functions which operate on R basic data structure-like matrix, and involve only calling a transform function (e.g., *as.matrix*) built-in to *data.table*. For functions which work on custom data structure such as SCE, flowFrame, or Seurat object, each package contains helpful functions for creating these objects from input data, usually a table with additional metadata entries. Some tools may also require users to include only rows/columns which are relevant. This can easily be done using functions built-into the *data.table* package. We refer readers to *data.table*'s extensive documentation on their Github page (https://github.com/Rdatatable/data.table) on how to perform those operations.

Additionally, most of Spectre's functions are designed such that they do not rely on each other to operate. A function may simply be executed on its own without having to run others beforehand. Some functions require data to be in a particular layout with appropriate calculations (e.g., calculating MFI across samples for heatmap creation with make.pheatmap), but this can be achieved either with (using write.sumtables) or without (using other functions or manual creation) Spectre. In addition, Spectre's functions are equipped with parameters that allow users to customize how they operate. The majority of these parameters come with a set of default values that are based on the original implementation of the tools. The capacity to run Spectre's functions with either default or custom parameter values make Spectre both simple and customizable .

4.4 | Docker for accessibility

Spectre provides wrapper functions to many existing computational tools, and thus require users to install the corresponding R packages before using them. Inadvertently, this creates a cycle of package dependencies as these packages often rely on other existing R packages. Managing such dependencies has proven to be a challenge even to advanced analysts, as there exists a myriad of ways to install, remove, and update packages (e.g., CRAN, Bioconductor [67], devtools [68]). Moreover, some packages rely on users to have other softwares (e.g., Xcode in Mac, Rtools in Windows) or compilers available on their computer. This may pose as a major hurdle for those who are not familiar with R. To address this, we made Spectre available as a Docker image. The image is a prepared environment loaded with RStudio for users to interact with R code (write and run) and all the libraries required by Spectre. By downloading the Docker image and launching it as a self-contained computing environment, users will be able to run their analysis without going through the complicated setup process. While Docker introduces an additional layer between user's physical computing resources and the analytical tools, previous work by IBM indicates the performance degradation to be negligible [69].

In addition to making it easier for potential users to use Spectre, Spectre's Docker image can also improve flexibility and transparency. Instead of exporting and uploading every intermediate step in their analyses, users can now simply perform their analysis in the Docker container running Spectre's Docker image, and thereafter export them out as Docker image and upload them in a public repository for future investigators to download and either replicate or extend those analyses. The Docker image will store all data, variables, functions, and R packages used in the analyses unless explicitly removed.

4.5 | Versatility of application

The structure of all forms of cytometry data, including flow, spectral or mass, are inherently similar: each row is an individual cell, whilst each column is an individual marker or feature measured on those cells. The values in such a table indicate the signal intensity for each individual marker within the panel on each cell. Thus, Spectre can be used on datasets generated by both flow (including spectral) and mass cytometry, following compensation (or spectral unmixing), and initial cleanup gating. As a result, Spectre can also be used on other forms of cellular data, such single-cell RNA seq data, following some additional pre-processing steps. Additionally, Spectre can be used to analyze HD imaging data, such as that generated by IMC, once cellular segmentation has been performed.

ACKNOWLEDGMENTS

This work was supported by a grant from the Marie Bashir Institute for Infectious Disease and Biosecurity, The University of Sydney. This work was supported in part by NH&MRC Project grant 1088242 and grant from the Merridew Foundation to N.J.C.K. Authors T.M.A. and F.M-W. are supported by the International Society for the Advancement of Cytometry (ISAC) Marylou Ingram Scholars program. G.H.P. was supported by the Australian Government Research Training Program (RTP) Scholarship. A.G.S. is supported by the Australian Government RTP Scholarship and The University of Sydney Postgraduate Merit Award. We would like to acknowledge the contribution of our colleagues who inspired the design and implementation of Spectre.

AUTHOR CONTRIBUTIONS

Thomas Ashhurst: Conceptualization; data curation; formal analysis; funding acquisition; investigation; methodology; project administration; resources; software; supervision; validation; visualization; writing-original draft; writing-review & editing. **Felix Marsh-Wakefield:** Conceptualization; data curation; formal analysis; investigation; methodology; software; validation; visualization; writing-original draft; writing-review & editing. **Givanna Putri:** Conceptualization; formal analysis; investigation; methodology; software; validation;

visualization; writing-original draft; writing-review & editing. **Alanna Spiteri:** Investigation; methodology; visualization. **Diana Shinko:** Data curation; investigation; methodology; resources; visualization. **Mark Read:** Conceptualization; methodology; supervision; validation; writing-review & editing. **Adrian Smith:** Conceptualization; methodology; resources; supervision. **Nicholas King:** Conceptualization; funding acquisition; resources; supervision; writing-review & editing.

DATA AVAILABILITY

Spectre code and demonstration data used in this paper are available at https://github.com/immunedynamics/spectre.

ORCID

Thomas Myles Ashhurst https://orcid.org/0000-0001-7269-7773
Felix Marsh-Wakefield https://orcid.org/0000-0002-6839-7628
Mark Norman Read https://orcid.org/0000-0002-1481-4780
Adrian Lloyd Smith https://orcid.org/0000-0002-0505-0344
Nicholas Jonathan Cole King https://orcid.org/0000-0002-3877-9772

REFERENCES

1. Bendall SC, Davis KL, Amir EAD, Tadmor MD, Simonds EF, Chen TJ, et al. Single-cell trajectory detection uncovers progression and regulatory coordination in human B cell development. Cell. 2014;157(3):714–25.
2. Park LM, Lannigan J, Jaimes MC. OMIP-069: forty-color full spectrum flow cytometry panel for deep Immunophenotyping of major cell subsets in human peripheral blood. Cytometry A. 2020;97(10):1044–51.
3. Mair F, Prlic M. OMIP-044: 28-color immunophenotyping of the human dendritic cell compartment. Cytometry A. 2018;93(4):402–5.
4. Nettey L, Giles AJ, Chattopadhyay PK. OMIP-050: a 28-color/30-parameter fluorescence flow cytometry panel to enumerate and characterize cells expressing a wide Array of immune checkpoint molecules. Cytometry A. 2018;93(11):1094–6.
5. Mair F, Hartmann FJ, Mrdjen D, Tosevski V, Krieg C, Becher B. The end of gating? An introduction to automated analysis of high dimensional cytometry data. Eur J Immunol. 2016;46(1):34–43.
6. Brinkman RR. Improving the rigor and reproducibility of flow cytometry-based clinical research and trials through automated data analysis. Cytometry A. 2020;97(2):107–12.
7. Levine JH, Simonds EF, Bendall SC, Davis KL, Amir EAD, Tadmor MD, et al. Data-driven phenotypic dissection of AML reveals progenitor-like cells that correlate with prognosis. Cell. 2015;162(1):184–97.
8. Van Gassen S, Callebaut B, Van Helden MJ, Lambrecht BN, Demeester P, Dhaene T, et al. FlowSOM: using self-organizing maps for visualization and interpretation of cytometry data. Cytometry A. 2015;87(7):636–45.
9. Samusik N, Good Z, Spitzer MH, Davis KL, Nolan GP. Automated mapping of phenotype space with single-cell data. Nat Methods. 2016;13(6):493–6.
10. van der Maaten L, Hinton G. Visualizing data using t-SNE. J Mac Learn Res. 2008;9:2579–605.
11. van der Maaten L. Accelerating t-SNE using tree-based algorithms. J Mac Learn Res. 2014;15:3221–45.
12. McInnes L, Healy J, James Melville J. UMAP: uniform manifold approximation and projection for dimension reduction; 2018. arXiv, 2018: p. 1802.03426.
13. Setty M, Tadmor MD, Reich-Zeliger S, Angel O, Salame TM, Kathail P, et al. Wishbone identifies bifurcating developmental trajectories from single-cell data. Nat Biotechnol. 2016;34(6):637–45.
14. Li H, Shaham U, Stanton KP, Yao Y, Montgomery RR, Kluger Y. Gating mass cytometry data by deep learning. Bioinformatics. 2017;33(21):3423–30.
15. Chen Y, Lakshmikanth T, Mikes J, Brodin P. Single-cell classification using learned cell phenotypes. 2020. bioRxiv 2020.07.22.216002.
16. Kaushik A, Dunham D, He Z, Manohar M, Desai M, Nadeau K, Less SA. CyAnno: a semi-automated approach for cell type annotation of mass cytometry datasets; 2020. bioRxiv. 2020.08.28.272559.
17. Chen H, Lau MC, Wong MT, Newell EW, Poidinger M, Chen J. Cytofkit: a bioconductor package for an integrated mass Cytometry data analysis pipeline. PLoS Comput Biol. 2016;12(9):e1005112.
18. Bruggner RV, Bodenmiller B, Dill DL, Tibshirani RJ, Nolan GP. Automated identification of stratifying signatures in cellular subpopulations. Proc Natl Acad Sci USA. 2014;111(26):E2770–7.
19. Chevrier S, Crowell HL, Zanotelli VRT, Engler S, Robinson MD, Bodenmiller B. Compensation of signal spillover in suspension and imaging mass Cytometry. Cell Syst. 2018;6(5):612–20.e5.
20. Nowicka M, Krieg C, Crowell HL, Weber LM, Hartmann FJ, Guglietta S, et al. CyTOF workflow: differential discovery in high-throughput high-dimensional cytometry datasets. F1000Res. 2017;6:748.
21. Weber LM, Nowicka M, Soneson C, Robinson MD. Diffcyt: differential discovery in high-dimensional cytometry via high-resolution clustering. Commun Biol. 2019;2:183.
22. Van P, Jiang W, Gottardo R, Finak G. ggCyto: next generation open-source visualization software for cytometry. Bioinformatics. 2018;34(22):3951–3.
23. Finak G, Frelinger J, Jiang W, Newell EW, Ramey J, Davis MM, et al. OpenCyto: an open source infrastructure for scalable, robust, reproducible, and automated, end-to-end flow cytometry data analysis. PLoS Comput Biol. 2014;10(8):e1003806.
24. Butler A, Hoffman P, Smibert P, Papalexi E, Satija R. Integrating single-cell transcriptomic data across different conditions, technologies, and species. Nat Biotechnol. 2018;36(5):411–20.
25. Stuart T, Butler A, Hoffman P, Hafemeister C, Papalexi E, Mauck WM III, et al. Comprehensive integration of single-cell data. Cell. 2019;177(7):1888–902.e21.
26. Amezquita RA, Lun ATL, Becht E, Carey VJ, Carpp LN, Geistlinger L, et al. Orchestrating single-cell analysis with bioconductor. Nat Methods. 2020;17(2):137–45.
27. Zeng C, Mulas F, Sui Y, Guan T, Miller N, Tan Y, et al. Pseudotemporal ordering of single cells reveals metabolic control of postnatal beta cell proliferation. Cell Metab. 2017;25(5):1160–75.e11.
28. Tran HTN, Ang KS, Chevrier M, Zhang X, Lee NYS, Goh M, et al. A benchmark of batch-effect correction methods for single-cell RNA sequencing data. Genome Biol. 2020;21(1):12.
29. Regev A, Teichmann SA, Lander ES, Amit I, Benoist C, Birney E, et al. The human cell atlas. Elife. 2017;6:e27041.
30. Korsunsky I, Millard N, Fan J, Slowikowski K, Zhang F, Wei K, et al. Fast, sensitive and accurate integration of single-cell data with harmony. Nat Methods. 2019;16(12):1289–96.
31. Hao Y, Hao S, Andersen-Nissen E, Mauck III WM, Zheng S, et al. Integrated analysis of multimodal single-cell data. 2020. bioRxiv.2020.10.12.335331.
32. Lin Y, Ghazanfar S, Wang KYX, Gagnon-Bartsch JA, Lo KK, Su X, et al. scMerge leverages factor analysis, stable expression, and pseudoreplication to merge multiple single-cell RNA-seq datasets. Proc Natl Acad Sci USA. 2019;116(20):9775–84.
33. Hahne F, Khodabakhshi AH, Bashashati A, Wong CJ, Gascoyne RD, Weng AP, et al. Per-channel basis normalization methods for flow cytometry data. Cytometry A. 2010;77(2):121–31.
34. Schuyler RP, Jackson C, Garcia-Perez JE, Baxter RM, Ogolla S, Rochford R, et al. Minimizing batch effects in mass Cytometry data. Front Immunol. 2019;10:2367.

35. Van Gassen S, Gaudilliere B, Angst MS, Saeys Y, Aghaeepour N. CytoNorm: a normalization algorithm for Cytometry data. Cytometry A. 2020;97(3):268–78.
36. Trussart M, Teh CE, Tan T, Leong L, Gray DHD, Speed TP. Removing unwanted variation with CytofRUV to integrate multiple CyTOF datasets. Elife. 2020;9:e59630.
37. Hahne F, LeMeur N, Brinkman RR, Ellis B, Haaland P, Sarkar D, et al. flowCore: a bioconductor package for high throughput flow cytometry. BMC Bioinformatics. 2009;10:106.
38. Ashhurst TM, Cox DA, Smith AL, King NJC. Analysis of the murine bone marrow hematopoietic system using mass and flow Cytometry. Methods Mol Biol. 2019;1989:159–92.
39. Koutsakos M, Rowntree LC, Hensen L, Chua BY, van de Sandt CE, Habel JR, et al. Integrated immune dynamics define correlates of COVID-19 severity and antibody responses. Cell Rep Med. 2021;2:100208.
40. Niewold P, Ashhurst TM, Smith AL, King NJC. Evaluating spectral cytometry for immune profiling in viral disease. Cytometry A. 2020;97:1165–79.
41. Monaco G, Chen H, Poidinger M, Chen J, de Magalhães JP, Larbi A. flowAI: automatic and interactive anomaly discerning tools for flow cytometry data. Bioinformatics. 2016;32(16):2473–80.
42. Dowle M, Srinivasan A. data.table: extension of data.frame. R package version 1.13.0; 2020. https://CRAN.R-project.org/package=data.table.
43. Parks DR, Roederer M, Moore WA. A new "Logicle" display method avoids deceptive effects of logarithmic scaling for low signals and compensated data. Cytometry A. 2006;69(6):541–51.
44. Van Gassen S, Callebaut B, Saeys Y. FlowSOM: using self-organizing maps for visualization and interpretation of cytometry data; 2020. http://bioconductor.org/packages/release/bioc/html/FlowSOM.html.
45. Krijthe JH. Rtsne: T-distributed stochastic neighbor embedding using a barnes-hut implementation; 2015. https://github.com/jkrijthe/Rtsne.
46. Linderman GC, Rachh M, Hoskins JG, Steinerberger S, Kluger Y. Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data. Nat Methods. 2019;16(3):243–5.
47. Konopka T. umap: uniform manifold approximation and projection. R package version 0.2.5.0; 2020. https://CRAN.R-project.org/package=umap.
48. Team RC. R: a language and environment for statistical computing; 2020. https://www.R-project.org/.
49. Wickham H. ggplot2: elegant graphics for data analysis. New York: Springer-Verlag; 2016. https://ggplot2.tidyverse.org.
50. Kolde R. pheatmap: pretty heatmaps. R package version 1.0.12; 2019. https://CRAN.R-project.org/package=pheatmap.
51. Kassambara A. ggpubr: 'ggplot2' based publication ready plots. R package version 0.4.0; 2020. https://CRAN.R-project.org/package=ggpubr.
52. Blighe K, Rana S, Lewis M. EnhancedVolcano: publication-ready volcano plots with enhanced colouring and labeling. R package version 1.6.0; 2020. https://github.com/kevinblighe/EnhancedVolcano.
53. Beygelzimer A, Kakadet S, Langford J, Arya S, Mount D, Li S. FNN: fast nearest neighbor search algorithms and applications. R package version 1.1.3; 2019. https://CRAN.R-project.org/package=FNN.
54. Kuhn M. caret: classification and regression training. R package version 6.0–86; 2020. https://CRAN.R-project.org/package=caret.
55. Moore WA, Parks DR. Update for the logicle data scale including operational code implementations. Cytometry A. 2012;81(4):273–7.
56. Bendall SC, Simonds EF, Qiu P, Amir EAD, Krutzik PO, Finck R, et al. Single-cell mass cytometry of differential immune and drug responses across a human hematopoietic continuum. Science. 2011;332(6030):687–96.
57. Weber LM, Robinson MD. Comparison of clustering methods for high-dimensional single-cell flow and mass cytometry data. Cytometry A. 2016;89(12):1084–96.
58. Belkina AC, Ciccolella CO, Anno R, Halpert R, Spidlen J, Snyder-Cappione JE. Automated optimized parameters for T-distributed stochastic neighbor embedding improve visualization and analysis of large datasets. Nat Commun. 2019;10(1):5415.
59. van Unen V, Höllt T, Pezzotti N, Li N, Reinders MJT, Eisemann E, et al. Visual analysis of mass cytometry data by hierarchical stochastic neighbour embedding reveals rare cell types. Nat Commun. 2017;8(1):1740.
60. Vuckovic S, Bryant CE, Lau KHA, Yang S, Favaloro J, McGuire HM, et al. Inverse relationship between oligoclonal expanded CD69- TTE and CD69+ TTE cells in bone marrow of multiple myeloma patients. Blood Adv. 2020;4(19):4593–604.
61. Marsh-Wakefield F, Ashhurst T, Trend S, McGuire H, Juillard P, Zinger A, et al. IgG3 (+) B cells are associated with the development of multiple sclerosis. Clin Transl Immunology. 2020;9(5):e01133.
62. Marsh-Wakefield F, Kruzins A, McGuire HM, Yang S, Bryant C, Fazekas de St. Groth B, et al. Mass Cytometry discovers two discrete subsets of CD39(−)Treg which discriminate MGUS from multiple myeloma. Front Immunol. 2019;10:1596.
63. Shinko D, McGuire HM, Diakos CI, Pavlakis N, Clarke SJ, Byrne SN, et al. Mass Cytometry reveals a sustained reduction in CD16(+) natural killer cells following chemotherapy in colorectal cancer patients. Front Immunol. 2019;10:2584.
64. Hayashida E, Ling ZL, Ashhurst TM, Viengkhou B, Jung SR, Songkhunawej P, et al. Zika virus encephalitis in immunocompetent mice is dominated by innate immune cells and does not require T or B cells. J Neuroinflammation. 2019;16(1):177.
65. Wickham H, François R, Henry L, Müller K. dplyr: a grammar of data manipulation. R package version 0.8.5; 2020. https://CRAN.R-project.org/package=dplyr.
66. Belson WA. Matching and prediction on the principle of biological classification. J R Stat Soc Ser C Appl Stat. 1959;8(2):65–75.
67. Morgan M. BiocManager: access the bioconductor project package repository. R package version 1.30.10; 2019. https://CRAN.R-project.org/package=BiocManager.
68. Wickham H, Hester J, Chang W. devtools: tools to make developing R packages easier. R package version 2.3.0; 2020. https://CRAN.R-project.org/package=devtools.
69. Felter W,Ferreira A, Rajamony R, Rubio J. An updated performance comparison of virtual machines and Linux containers. 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS); 2015. p. 171–172.
70. Mersmann O. Microbenchmark: accurate timing functions. R package version 1.4–7. CRAN; 2019.

SUPPORTING INFORMATION

Additional supporting information may be found online in the Supporting Information section at the end of this article.

How to cite this article: Ashhurst TM, Marsh-Wakefield F, Putri GH, et al. Integration, exploration, and analysis of high-dimensional single-cell cytometry data using Spectre. *Cytometry*. 2021;1–17. https://doi.org/10.1002/cyto.a.24350