Overview

Short answer

Yes you can. Following the standard data preparation steps that are typical for single cell RNA sequencing (scRNAseq) data, tools such as FlowSOM etc can be run on scRNAseq data. This is also true of using scRNAseq analysis tools on flow/CyTOF data.

Longer answer

Both flow/CyTOF and scRNAseq measure the expression of some feature (genes/proteins etc) on single cells cells, but in different ways. Fundamentally the data structure of each is inherently similar – tabular data of features vs cells. Most clustering and dimensionality reduction (DR) tools will work on any kind of tabular data (features vs observations/cells) once the data has been prepared appropriately. One reason that tools from one field are not often used on data from another, is that many tools are developed to interact with data formats that are specific for each field. For example:

- *Flow/CyTOF*: FlowSOM is designed to work with 'FCS files/flowFrames',
- *scRNAseq*: Seurat's clustering is designed to work on 'Seurat Objects', and many other scRNAseq approaches are designed to work on in the Bioconductor scRNAseq 'SingleCellExperiment' data format.

It is not that these tools *cannot* work on different data types, but simply their implementation makes it difficult to do so. In **Spectre**, we have made a deliberate choice to use a simple '**data.table**' structures for the analysis all cytometry/scRNAseq datasets. This allows for:

1. A common workflow and manipulation processes for data from different fields, allowing for easy conversions between data types.
2. Simple interactions for filtering/plotting etc.
3. Easy integration of tools from the flow/CyTOF and scRNAseq communities, and from the broader machine learning community.

Additionally, the use of **data.table**s massively increases processing speed for reading/writing CSV files, as well as for common operations (filtering, subsetting etc).

Worked example: analysing scRNAseq data with Spectre (FlowSOM + UMAP)

Worked example: analysing scRNAseq data in R using Spectre

For this example, we'll be using a ~3000 cell dataset from Seurat's guided clustering tutorial (https://satijalab.org/seurat/v3.2/pbmc3k_tutorial.html). We will perform preparation and pre-processing steps with Seurat (in R) and then perform clustering/DR and plotting etc with Spectre (in R).

First, some preparation of scRNAseq data (in brief – more information on the Seurat page here)

A key difference in how flow/CyTOF and scRNAseq data is analysed is in how the data is prepared. In this tutorial, the sequencing data is initially processed in CellRanger (from 10X Genomics). This data is read into R to create unique molecular identified (UMI) count matrix (number of molecules per gene (row) per cell (column). Subsequently, a 'Seurat Object' is created from this count matrix, with additional cell/gene/experiment metadata included.

Comparing scRNAseq to flow/CyTOF

Both scRNAseq and flow/CyTOF are measuring the level/count of a gene/marker on numerous cells. Traditionally, tabular scRNAseq data is organised such that rows represent genes, and columns represent cells. In cytometry, this is the other way round.

- scRNAseq: genes/features = rows, cells = columns
- Flow/CyTOF: markers/features = columns, cells = rows

Once the Seurat object has been created, the following pre-processing/preparatory steps were performed.

- *Filtering* – removing low quality cells, dying cells, empty droplets, doublets etc
- *Normalising data* – normalising gene expression values for each cell to the total expression (note, this is not 'batch' normalisation)
- *Feature selection* (identifying highly variable features) – determining which genes have meaningful differences between the cells in the dataset
- *Data scaling* – gives equal weight to each gene so that highly expressed genes don't dominate the analysis
- *Linear dimensionality reduction* – normally this step would be performed with Seurat, but we have performed it in Spectre (see below)

Comparing scRNAseq to flow/CyTOF

A number of these steps are conceptually similar to what is performed in flow/CyTOF preparation:

- *Filtering* – this is usually performed by 'gating' out dead cells, doublets etc
- *Normalisation* – this is not typically done with flow/CyTOF data, as measurements in cytometry are made from the entire intact cell
- *Data scaling* – this is not typically done with flow/CyTOF data, but it is a reasonable practice to ensure that highly expressed proteins (such as HLA-DR) do not dominate the analysis.
- *Linear dimensionality reduction* – see below.

```
### Code derived from Seurat's excellent analysis tutorial

library(dplyr)
library(Seurat)
library(patchwork)

library(Spectre)
package.load()

pbmc.data <- Read10X(data.dir = "/Users/thomasa/Downloads/filtered_gene_bc_matrices/hg19/")
pbmc.data

pbmc <- CreateSeuratObject(counts = pbmc.data, project = "pbmc3k", min.cells = 3, min.features = 200)
pbmc

### Exploration
    pbmc$orig.ident
    pbmc$nCount_RNA
    pbmc$nFeature_RNA

    nGenes <- pbmc@assays$RNA@counts@Dim[[1]]
    nCells <- pbmc@assays$RNA@counts@Dim[[2]]

    nObs <- length(pbmc@assays$RNA@counts@x)

    nGenes * nCells

    nObs / nGenes


### QC etc

    # The [[ operator can add columns to object metadata. This is a great place to stash QC stats
    pbmc[["percent.mt"]] <- PercentageFeatureSet(pbmc, pattern = "^MT-")


    # Visualize QC metrics as a violin plot
    VlnPlot(pbmc, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)

    # FeatureScatter is typically used to visualize feature-feature relationships, but can be used
    # for anything calculated by the object, i.e. columns in object metadata, PC scores etc.

    plot1 <- FeatureScatter(pbmc, feature1 = "nCount_RNA", feature2 = "percent.mt")
    plot2 <- FeatureScatter(pbmc, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
    plot1 + plot2

    pbmc <- subset(pbmc, subset = nFeature_RNA > 200 & nFeature_RNA < 2500 & percent.mt < 5)

    pbmc <- NormalizeData(pbmc, normalization.method = "LogNormalize", scale.factor = 10000)
    pbmc <- NormalizeData(pbmc)
    #pbmc[["RNA"]]@data

    pbmc <- FindVariableFeatures(pbmc, selection.method = "vst", nfeatures = 2000)

    # Identify the 10 most highly variable genes
    top10 <- head(VariableFeatures(pbmc), 10)

    # plot variable features with and without labels
    plot1 <- VariableFeaturePlot(pbmc)
    plot2 <- LabelPoints(plot = plot1, points = top10, repel = TRUE)
    plot1 + plot2

    all.genes <- rownames(pbmc)
    pbmc <- ScaleData(pbmc, features = all.genes)

### Variable genes
        var.genes <- pbmc[["RNA"]]@var.features
        var.genes
```

Converting the filtered/normalised/scaled data to a data.table

To interact with scRNAseq data in Spectre, we will extract the filtered/normalised/scaled RNA data and convert it into a **data.table**. To facilitate this, we have create a conversion function which performs a few key steps:

1. The specific data subset (scaled RNA data) is extracted from the Seurat object
2. The feature names (rows) and cell IDs (columns) are extracted from the Seurat object
3. A data.table is constructed (and transposed) so that the scaled RNA is tabulated with rows = cells and columns = genes.

We can then use a saved vector of 'column names' to select specific genes, for example, the highly variable genes.

First, create the function to extract the relevant data from within the Seurat object, and convert it into a data.table.

```
 do.convert <- function(dat, # whatever dataset required (seurat object, flowFrame, etc)
                        from, # "Seurat", "flowFrame"
                        seurat.assay = "RNA",
                        seurat.slot = "scale.data",
                        to = "data.table"){ # data.table

  ### Setup

     library(Spectre)
     package.load()

  ### Seurat Objects

     if(from == "Seurat"){
       library(dplyr)
       library(Seurat)
```

```
        library(patchwork)

        a <- GetAssayData(object = dat)
        x <- GetAssayData(object = dat, assay = seurat.assay, slot = seurat.slot)

        geneNames <- a@Dimnames[[1]]
        cellNames <- a@Dimnames[[2]]

  ### Turn into data.table

        if(to == "data.table"){
          x.dt <- as.data.table(x)
          rownames(x.dt) <- geneNames
          colnames(x.dt)

          x.dt.t <- data.table::transpose(x.dt)
          rownames(x.dt.t) <- colnames(x.dt)
          colnames(x.dt.t) <- rownames(x.dt)

          res <- x.dt.t
        }
      }

  ### Return

      return(res)
}
```

Next, apply it to the 'pbmc' dataset.

```
    cell.dat <- do.convert(dat = pbmc, # The seurat object 'pbmc'
                           from = "Seurat", # Converting 'from' a Seurat object
                           seurat.assay = "RNA", # Exporting the RNA data
                           seurat.slot = "scale.data") # Specifically exporting the scaled RNA data
```

```
    dim(cell.dat) # check the dimensionality of the resulting data.table to ensure it's been generated
correctly
```

Running PCA

A key step in preparing scRNAseq data for analysis is running principle component analysis (PCA) to create linear combinations of gene expression patterns. Because scRNAseq data contains a huge number of measured genes, and is inherently sparse, meaningful linear combinations of gene expression patterns can be summarised in principle components that define the structure of the data. In this example, the top 30 principle components (PCs) are used. *This serves a different role to the non-linear DR approaches (e.g. tSNE/UMAP) that are discussed below.*

Comparing scRNAseq to flow/CyTOF

- *Linear dimensionality reduction* – this is sometimes done with flow/CyTOF data, but is not as common. Essentially, because each antibody used in a panel is chosen for it's staining patterns across the cells of interest, the antibody signals themselves have specific and meaningful expression patterns in and of themselves, whereas the variable and sheer number of genes that are measured in scRNAseq require some kind of summation/reduction to create meaningful features

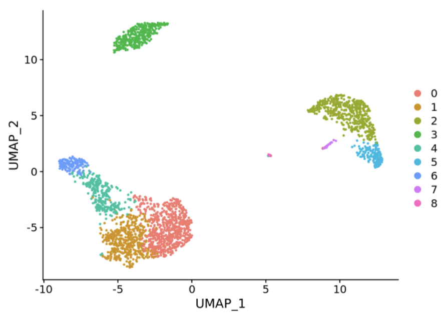
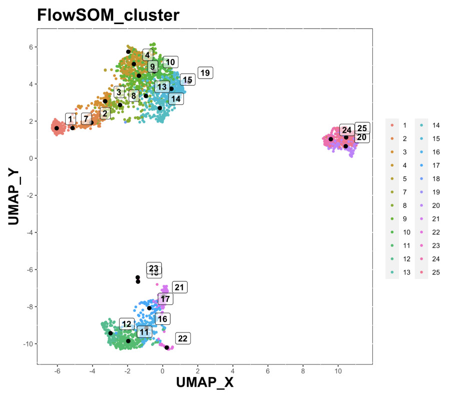
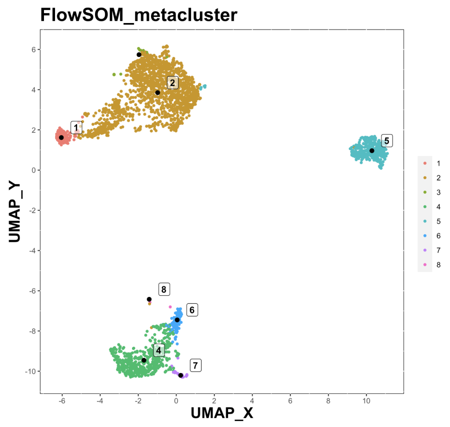
```
### Prep

    cellular.cols <- colnames(cell.dat)
    length(cellular.cols)
    cellular.cols[c(13710:length(cellular.cols))]

    test.nms <- names(cell.dat)[c(1:100)]

### PCA

    pca_out <- stats::prcomp(cell.dat[,var.genes,with = FALSE],
                             scale = TRUE)

    pcs <- colnames(pca_out$x)[c(1:30)]
    pcs

    cell.dat <- cbind(cell.dat,pca_out$x[,c(1:30)]) # Add values for the first 30 principle components
    str(cell.dat)

    names(cell.dat)[c(13710:length(names(cell.dat)))]
```

Running clustering and non-linear DR

In scRNAseq data, clustering and non-linear DR (tSNE/UMAP etc) is performed on the PCs, rather than the genes themselves. In this tutorial, the top 30 PCs were used, so clustering/DR is performed on 30 features – a similar number to what is used with CyTOF data. Below we show the clusters resulting from Seurat's clustering approaches, alongside clustering results from FlowSOM. Both approaches work on the scRNAseq data, but with typical performance differences between the approaches (for example, FlowSOM's primary clusters will capture minute changes in cellular populations, but often these are summarised together in the FlowSOM metacluster.

```
### FlowSOM + UMAP
    cell.dat <- run.flowsom(dat = cell.dat, xdim = 5, ydim = 5, meta.k = 8, use.cols = pcs)
    cell.dat <- run.umap(dat = cell.dat, use.cols = pcs)
```

```
### Plotting

    setwd("/Users/ThomasA/Desktop/")

    make.colour.plot(cell.dat, "UMAP_X", "UMAP_Y", "FlowSOM_metacluster", col.type = 'factor',
add.label = TRUE)
    make.colour.plot(cell.dat, "UMAP_X", "UMAP_Y", "FlowSOM_cluster", col.type = 'factor', add.label =
 TRUE)
    make.colour.plot(cell.dat, "UMAP_X", "UMAP_Y")

    cell.dat$CD3E

    make.colour.plot(cell.dat, "UMAP_X", "UMAP_Y", "CD3E")
    make.colour.plot(cell.dat, "UMAP_X", "UMAP_Y", "CD19")
    make.colour.plot(cell.dat, "UMAP_X", "UMAP_Y", "CD14")

    names(cell.dat)[c((length(names(cell.dat))-50):length(names(cell.dat)))]

    to.plot <- c("MS4A1", "GNLY", "CD3E", "CD14", "FCER1A", "FCGR3A", "LYZ", "PPBP",
    "CD8A")

    make.multi.plot(cell.dat, "UMAP_X", "UMAP_Y", to.plot, add.density = TRUE)
```

Seurat results (from webpage) **Phenograph-like clustering + UMAP**	**Spectre** **FlowSOM clusters (grid size = 5x5) + UMAP**	**Spectre** **FlowSOM metaclusters (n = 8) + UMAP**
Clusters  From: https://satijalab.org/seurat/v3.2/pbmc3k_tutorial.html		

Gene expression for reference: