Contents

ROI Analysis

VoltRon is capable of analyzing readouts from distinct spatial technologies including **segmentation (ROI)-based transciptomics assays** that capture large polygonic regions on tissue sections. VoltRon recognizes such readouts including ones from commercially available tools and allows users to implement a workflow similar to ones conducted on bulk RNA-Seq datasets. In this tutorial, we will analyze morphological images and gene expression profiles provided by the readouts of the Nanostring's GeoMx Digital Spatial Profiler platform, a high-plex spatial profiling technology which produces segmentation-based protein and RNA assays.

In this use case, **eight tissue sections** were fitted into the scan area of the slide loaded on the GeoMx DSP instrument. These sections were cut from **control and COVID-19 lung tissues** of donors categorized based on disease durations (acute and prolonged). See GSE190732 for more information.

We now import the GeoMx data, and start analyzing 87 user selected segments (i.e. region of interests, **ROI**) to check spatial localization of signals. The **importGeoMx** function requires:

- The path to the Digital Count Conversion file, **dcc.path**, and Probe Kit Configuration file, **pkc.file**, which are both provided as the output of the GeoMx NGS Pipeline.
- The path the to the metadata file, **summarySegment**, and the specific excel sheet that the metadata is found, **summarySegmentSheetName**, the path to the main morphology **image** and the original **ome.tiff** file, all of which are provided by the DSP Control Center. Please see GeoMx DSP Data Analysis User Manual for more information.

```r
# get data
library(dplyr)
library(VoltRon)
GeoMxR1 <- importGeoMx(dcc.path = "out/",
                       pkc.file = "out/Hs_R_NGS_WTA_v1.0.pkc",
                       summarySegment = "out/segmentSummary.xlsx",
                       summarySegmentSheetName = "Segment summary",
                       image = "out/morphology.tiff",
                       ome.tiff = "out/Lu 1A 1B 5 um, true exp.ome.tiff",
                       sample_name = "GeoMxR1")
```

Quality Control

Once the GeoMx data is imported, we can start off with examining key quality control measures and statistics on each segment to investigate each individual ROI such as sequencing saturation and the number of cells (nuclei) within each segment. VoltRon also provides the total number of unique transcripts per ROI and stores in the metadata.

```r
# visualize values
vrBarPlot(GeoMxR1_lessfeatures, features = c("Count", "Nuclei.count", "Sequencing.saturation"),
          x.label = "ROI.name", group.by = "ROI.type") +
  theme(axis.text.x = element_text(size = 3))
```

For measuring the quality of individual ROIs, we can add a new metadata column, called **CountPerNuclei**, to check the average quality of cells per ROI. It seems some number of ROIs with low counts per nuclei also have low sequencing saturation.

```r
# creating new metadata fields
Metadata(GeoMxR1)$CountPerNuclei <- Metadata(GeoMxR1)$Count/Metadata(GeoMxR1)$Nuclei.count
```

```
# visualize values again
vrBarPlot(GeoMxR1,
          features = c("Count", "Nuclei.count",
                       "Sequencing.saturation", "CountPerNuclei"),
          x.label = "ROI.name", group.by = "ROI.type", ncol = 3) +
  theme(axis.text.x = element_text(size = 5))
```

Processing

We can now filter ROIs based on our earlier observation of them having low count per nuclei where some also have low sequencing saturation.

```
# Filter for count per nuclei
GeoMxR1 <- subset(GeoMxR1, subset = CountPerNuclei > 500)
```

We then filter genes with low counts by extracting the count matrix and putting aside all genes whose maximum count across all 87 ROIs are less than 10.

```
# Filter for count per nuclei
GeoMxR1 <- subset(GeoMxR1, subset = CountPerNuclei > 500)

# filter for low count features
GeoMxR1_data <- vrData(GeoMxR1, norm = FALSE)
feature_ind <- apply(GeoMxR1_data, 1, function(x) max(x) > 10)
selected_features <- vrFeatures(GeoMxR1)[feature_ind]
GeoMxR1_lessfeatures <- subset(GeoMxR1, features = selected_features)
```

VoltRon is capable of normalizing data provided by a diverse set of spatial technologies, including the quantile normalization method suggested by the GeoMx DSP Data Analysis User Manual which scales the ROI profiles to the third quartile followed by the geometric mean of all third quartiles multipled to the scaled profile.

```
GeoMxR1 <- normalizeData(GeoMxR1, method = "Q3Norm")
```

Interactive Subsetting

Spatially informed genomic technologies heavily depend on image manipulation as images provide an additional set of information. Hence, VoltRon incorporates several interactive built-in utilities. One such functionality allows manipulating images of VoltRon assays where users can interactively choose subsets of images. However, we first resize the morphology image by providing the width of the new image (thus height will be reduced to preserve the aspect ratio).

```
# resizing the image
GeoMxR1 <- resizeImage(GeoMxR1, size = 4000)
```

VoltRon provides **a mini Shiny app** for subsetting spatial points of a VoltRon object by using the image as a reference. This app is particularly useful when multiple tissue sections were fitted to a scan area of a slide, such as the one from GeoMx DSP instrument. We use **interactive = TRUE** option in the subset function to call the mini Shiny app, and select bounding boxes of each newly created subset.

```
GeoMxR1_subset <- subset(GeoMxR1, interactive = TRUE)
```

We can now merge the list of subsets, or samples, each associated with one of eight sections. You can provide a list of names for the newly subsetted samples.

```
GeoMxR1_subset_list <- GeoMxR1_subset$subsets
samples <- c("prolonged case 4", "acute case 3", "control case 2",
             "acute case 1", "acute case 2", "prolonged case 5",
             "prolonged case 3", "control case 1")
GeoMxR1 <- merge(GeoMxR1_subset_list[[1]], GeoMxR1_subset_list[-1], samples = samples)
```

You may also save the selected image subsets and reproduce the interactive subsetting operation for later use.

```
samples <- c("prolonged case 4", "acute case 3", "control case 2",
             "acute case 1", "acute case 2", "prolonged case 5",
             "prolonged case 3", "control case 1")
subset_info_list <- GeoMxR1_subset$subset_info_list[[1]]
GeoMxR1_subset_list <- list()
for(i in 1:length(subset_info_list)){
  GeoMxR1_subset_list[[i]]  <- subset(GeoMxR1, image = subset_info_list[i])
  GeoMxR1_subset_list[[i]] <- samples[i]
}
GeoMxR1 <- merge(GeoMxR1_subset_list[[1]], GeoMxR1_subset_list[-1])
```

Visualization

We will now select sections of interests from the VoltRon object, and visualize features for each of them. The function **vrSpatialFeaturePlot** detects the number of assays within each VoltRon object and visualizes each feature per each spatial image. A grid of images are visualized either the number of assays or the number of features are larger than 1.

```
GeoMxR1_subset <- subset(GeoMxR1, sample = c("prolonged case 4","acute case 3"))
vrSpatialFeaturePlot(GeoMxR1_subset, features = c("CXCL11", "COL1A1"), group.by = "ROI.name",
                     label = TRUE, keep.scale = "all", title.size = 15)
```

Dimensionality Reduction

We can now process the normalized and demultiplexed samples to map ROIs across all sections onto lower dimensional spaces. The functions **getFeatures** and **getPCA** select features (i.e. genes) of interest from the data matrix across all samples and reduce it to a selected number of principal components.

```
GeoMxR1 <- getFeatures(GeoMxR1)
GeoMxR1 <- getPCA(GeoMxR1, dims = 30)
```

VoltRon provides additional dimensionality reduction techniques such as **UMAP**.

```
GeoMxR1 <- getUMAP(GeoMxR1)
```

The function **vrEmbeddingPlot** can be used to visualize embedding spaces (pca, umap, etc.) for any spatial point supported by VoltRon, hence cells, spots and ROI are all visualized using the same set of functions. Here we generate a new metadata column that represents the **disease durations (control, acute and prolonged case)**, then map gene profiles to the first two principal components.

```
Metadata(GeoMxR1)$Condition <- gsub(" [0-9]+$", "", Metadata(GeoMxR1)$Sample)
vrEmbeddingPlot(GeoMxR1, group.by = c("Condition"), embedding = "pca", pt.size = 3)
```

Gene expression profiles of ROIs associated with prolonged case sections seem to show some heterogeneity. We now color segments by section (or replicate, **Sample**) to observe the sources of variability. Three replicates of prolonged cases exhibit three different clusters of ROIs.

```
vrEmbeddingPlot(GeoMxR1, group.by = c("Condition"), embedding = "pca", pt.size = 3)
vrEmbeddingPlot(GeoMxR1, group.by = c("Sample"), embedding = "pca", pt.size = 3)
vrEmbeddingPlot(GeoMxR1, group.by = c("Sample"), embedding = "umap", pt.size = 3)
```

Differential Expression Analysis

VoltRon provides wrapping functions for calling tools and methods from popular differential expression analysis package such as DESeq2. We utilize **DESeq2** to find differentially expressed genes across disease conditions and select the **control case** as the base group (the condition that is of the baseline expression).

```
# get DE for all conditions
library(DESeq2)
DEresults <- getDiffExp(GeoMxR1, group.by = "Condition",
                        group.base = "control case", method = "DESeq2")
DEresults_sig <- DEresults %>% filter(!is.na(padj)) %>%
  filter(padj < 0.05, abs(log2FoldChange) > 1)
head(DEresults_sig)
```

The **vrHeatmapPlot** takes a set of features for any type of spatial point (cells, spots and ROIs) and visualizes scaled data per each feature. The select **highlight.some = TRUE** to annotate features which could be large in size. The heterogeneity across prolonged case segments are highlighted by different set of markers.

```
# get DE for all conditions
vrHeatmapPlot(GeoMxR1, features = unique(DEresults_sig$gene),
              group.by = "Condition", highlight.some = TRUE)
```

Markers of each individual tissue section for each disease duration is shown on the Heatmap.

```
# get DE for all conditions
vrHeatmapPlot(GeoMxR1, features = unique(DEresults_sig$gene),
              group.by = "Sample", highlight.some = TRUE)
```