

# Application of RESET to Seurat pbmc\_small scRNA-seq data using Seurat log normalization.

H. Robert Frost

## 1 Load the RESET package

```
> library(RESET)
```

## 2 Summary statistics for the pbmc\_small scRNA-seq data

This example uses the pbmc\_small data set included in the SeuratObject package and two contrived gene sets. Please see the other vignettes for more realistic examples using larger scRNA-seq data sets and gene set collections based on MSigDB.

```
> if (requireNamespace("Seurat", quietly=TRUE) && requireNamespace("SeuratObject", quietly=TRUE)) {
+   SeuratObject::pbmc_small
+   gene.names = rownames(SeuratObject::pbmc_small)
+   gene.names[1:5]
+   Seurat::VariableFeatures(SeuratObject::pbmc_small)[1:5]
+ } else {
+   message("Seurat package not available! Not executing associated vignette content.")
+ }

[1] "PPBP" "IGLL5" "VDAC3" "CD1C" "AKR1C3"
```

## 3 Define gene set collection

Create a gene set collection containing two contrived sets: one with the top 5 variable genes and one with randomly selected genes.

```
> if (requireNamespace("Seurat", quietly=TRUE)) {
+   gene.set.id.list = list()
+   # Create set with top 5 variable genes
+   gene.set.id.list[[1]] = c("PPBP", "IGLL5", "VDAC3", "CD1C", "AKR1C3")
+   names(gene.set.id.list)[1] = "VarGenes"
+   # Create set with 5 random genes
+   gene.set.id.list[[2]] = c("TREML1", "CD79B", "LRRRC25", "GPX1", "CFD")
+   names(gene.set.id.list)[2] = "RandomGenes"
+   print(gene.set.id.list)
+   # Create the list of gene indices required by resetForSeurat()
+   gene.set.collection = createVarSetCollection(var.names=gene.names,
+   var.sets=gene.set.id.list)
+ } else {
+   message("Seurat package not available! Not executing associated vignette content.")
+ }
```

```
$VarGenes
[1] "PPBP" "IGLL5" "VDAC3" "CD1C" "AKR1C3"

$RandomGenes
[1] "TREML1" "CD79B" "LRRC25" "GPX1" "CFD"
```

## 4 Execute RESET method

Since the scRNA-seq data has been processed using Seurat, we execute RESET using the `resetForSeurat()` function with scores based on the reconstruction of the top 5 PCs. Setting `k=5`, which is the size of the gene sets, will generate the reconstruction using a non-randomized algorithm.

```
> if (requireNamespace("Seurat", quietly=TRUE)) {
+   pbmc.reset = resetForSeurat(seurat.data=SeuratObject::pbmc_small,
+                               num.pcs=5,
+                               gene.set.collection=gene.set.collection,
+                               k=5)
+ } else {
+   message("Seurat package not available! Not executing associated vignette content.")
+ }
```

Look at the sample level and overall scores in "RESET" Assay

```
> if (requireNamespace("Seurat", quietly=TRUE)) {
+   # Display RESET scores for first 10 cells
+   print(pbmc.reset@assays$RESET[,1:10])
+   # Display overall RESET scores
+   pbmc.reset@assays$RESET@meta.features
+ } else {
+   message("Seurat package not available! Not executing associated vignette content.")
+ }
```

```
[1] "counts" "data"
      RESET
VarGenes  1.0402079
RandomGenes 0.5397981
```

## 5 Visualize RESET scores for the variable gene set

Visualize RESET scores using Seurat `FeaturePlot()`. The default Assay must first be changed to "RESET".

```
> if (requireNamespace("Seurat", quietly=TRUE)) {
+   Seurat::DefaultAssay(object = pbmc.reset) = "RESET"
+   Seurat::FeaturePlot(pbmc.reset, reduction="tsne", features="VarGenes")
+ } else {
+   message("Seurat package not available! Not executing associated vignette content.")
+   oldpar = par(mar = c(0,0,0,0))
+   plot(c(0, 1), c(0, 1), ann = F, bty = 'n', type = 'n', xaxt = 'n', yaxt = 'n')
+   text(x = 0.5, y = 0.5, paste("Seurat package not available!\n",
+                                "FeaturePlot not generated."),
+        cex = 1.6, col = "black")
+ }
```

```
+   par(oldpar)
+ }
```

