# HTO Analysis

Code

This is a notebook for HTO Analysis, according to Stoeckius, et al (https://genomebiology.biomedcentral.com/articles/10.1186/s13059-018-1603-1)

After we run CellRanger for the gene expression part, we run the CellRanger for the feature barcodes.

Now it is time to load the samples in R, following Satija's lab hashing vignette (https://satijalab.org/seurat/v3.1/hashing_vignette.html)

# Basic setup

Hide

```
# Load packages
library(Seurat)
```

Read in data

Hide

```
# Load the data (Change the paths according to the location of the files on your computer)
ge.data <- Read10X("filtered_ge_022_bc_matrix")
hto.data <- Read10X("filtered_hto_022_bc_matrix", gene.column = 1)
```

```
10X data contains more than one type and is being returned as a list containing matrices of each t
ype.
```

Hide

```
# Select cell barcodes detected by both RNA and HTO In the example datasets we have already
# filtered the cells for you, but perform this step for clarity.
joint.bcs <- intersect(colnames(ge.data), colnames(hto.data$`Antibody Capture`))
# Subset RNA and HTO counts by joint cell barcodes
ex.umis <- ge.data[, joint.bcs]
ex.htos <- as.matrix(hto.data$`Antibody Capture`[, joint.bcs])
# Confirm that the HTO have the correct names
rownames(ex.htos)
```

```
[1] "0251" "0252" "0253" "0254" "0255" "0256" "0257" "0258"
```

Setup Seurat object and add in the HTO data

Hide

```
# Setup Seurat object
ex.hashtag <- CreateSeuratObject(counts = ex.umis)
# Normalize RNA data with log normalization
ex.hashtag <- NormalizeData(ex.hashtag)
```

```
Performing log-normalization
0%   10   20   30   40   50   60   70   80   90   100%
[----|----|----|----|----|----|----|----|----|----|
**************************************************|
```

Hide

```
# Find and scale variable features
ex.hashtag <- FindVariableFeatures(ex.hashtag, selection.method = "mean.var.plot")
```

```
Calculating gene means
0%   10   20   30   40   50   60   70   80   90   100%
[----|----|----|----|----|----|----|----|----|----|
**************************************************|
Calculating gene variance to mean ratios
0%   10   20   30   40   50   60   70   80   90   100%
[----|----|----|----|----|----|----|----|----|----|
**************************************************|
```

Hide

```
ex.hashtag <- ScaleData(ex.hashtag, features = VariableFeatures(ex.hashtag))
```

```
Centering and scaling data matrix

  |
  |
|   0%
  |
  |==================================================
|  50%
  |
  |================================================================================================
======| 100%
```

# Adding HTO data as an independent assay

You can read more about working with multi-modal data here (https://satijalab.org/seurat/multimodal_vignette.html)

Hide

```
# Add HTO data as a new assay independent from RNA
ex.hashtag[["HTO"]] <- CreateAssayObject(counts = ex.htos)
# Normalize HTO data, here we use centered log-ratio (CLR) transformation
ex.hashtag <- NormalizeData(ex.hashtag, assay = "HTO", normalization.method = "CLR")
```

```
Normalizing across features
```

```
  |                                                    | 0 % ~calculating
  |+++++++                                             | 12% ~00s
  |+++++++++++++                                       | 25% ~00s
  |+++++++++++++++++++                                 | 38% ~00s
  |+++++++++++++++++++++++++                           | 50% ~00s
  |++++++++++++++++++++++++++++++++                    | 62% ~00s
  |++++++++++++++++++++++++++++++++++++++              | 75% ~00s
  |++++++++++++++++++++++++++++++++++++++++++++        | 88% ~00s
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=00s
```

# Demultiplex cells based on HTO enrichment

Here we use the Seurat function HTODemux() to assign single cells back to their sample origins.

Hide

```
# If you have a very large dataset we suggest using k_function = 'clara'. This is a k-medoid
# clustering function for large applications. You can also play with additional parameters (see
# documentation for HTODemux()) to adjust the threshold for classification. Here we are using the
# default settings
ex.hashtag <- HTODemux(ex.hashtag, assay = "HTO", positive.quantile = 0.99) # , verbose=T
```

```
Cutoff for 0251 : 47 reads
Cutoff for 0252 : 118 reads
Cutoff for 0253 : 34 reads
Cutoff for 0254 : 43 reads
Cutoff for 0255 : 33 reads
Cutoff for 0256 : 43 reads
Cutoff for 0257 : 109 reads
Cutoff for 0258 : 74 reads
```

# Visualize demultiplexing results

Output from running HTODemux() is saved in the object metadata. We can visualize how many cells are classified as singlets, doublets and negative/ambiguous cells.

Hide

```
# Global classification results
table(ex.hashtag$HTO_classification.global)
```

```

 Doublet Negative  Singlet
    1452      522     8890
```

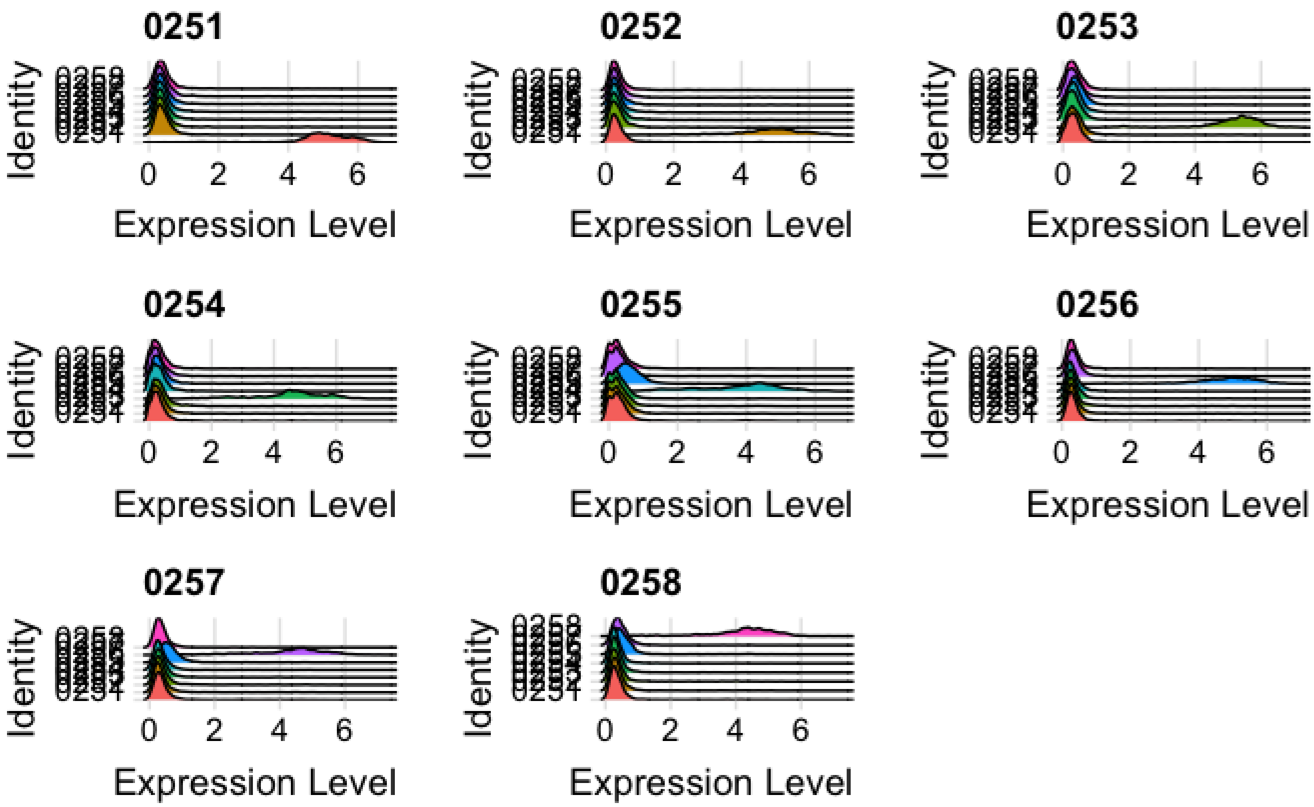
Hide

```
# Save sum
n_cells <- sum(table(ex.hashtag$HTO_classification.global))
```

Visualize enrichment for selected HTOs with ridge plots

Hide

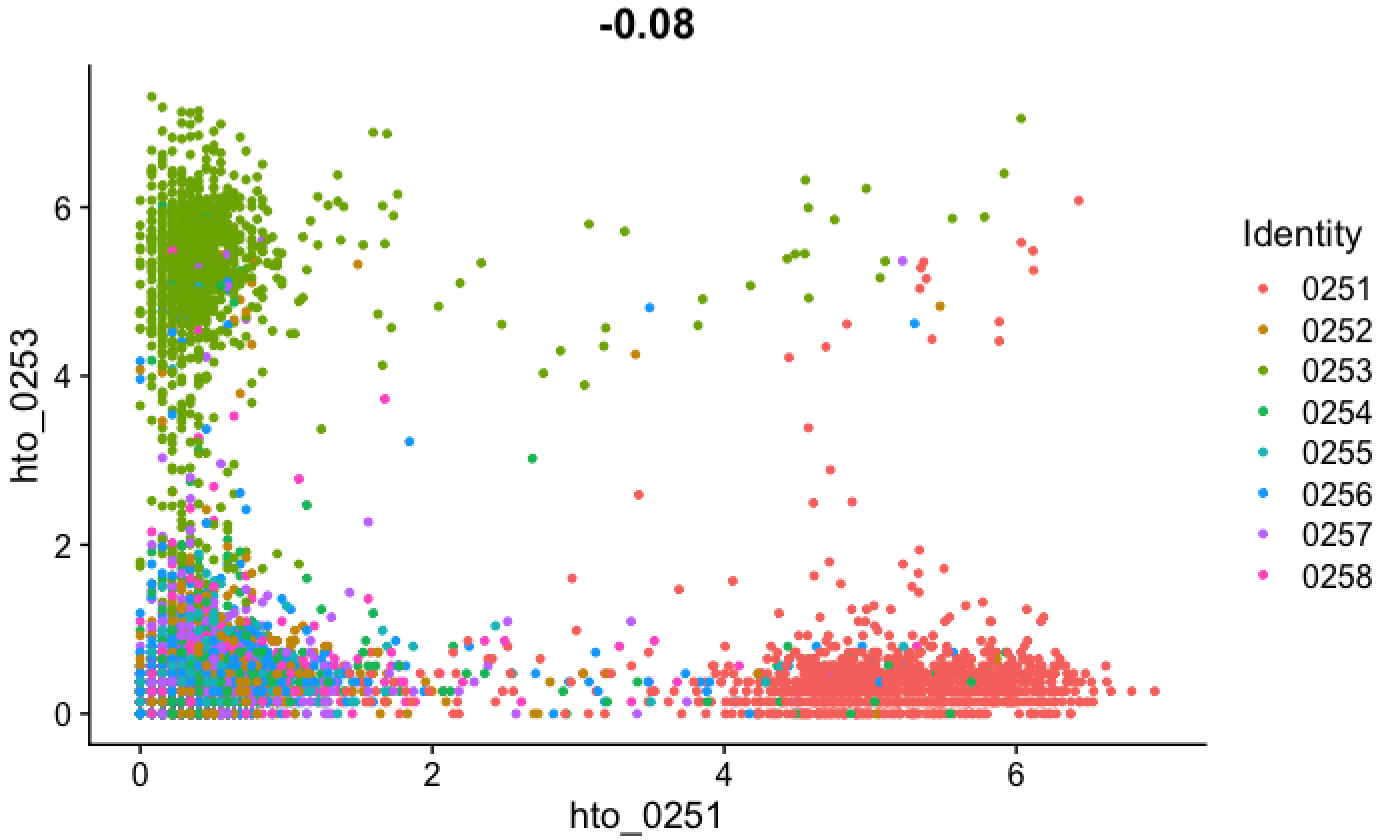
```
# Group cells based on the max HTO signal
Idents(ex.hashtag) <- "HTO_maxID"
RidgePlot(ex.hashtag, assay = "HTO", features = rownames(ex.hashtag[["HTO"]]), ncol = 3)
```




Visualize pairs of HTO signals to confirm mutual exclusivity in singlets

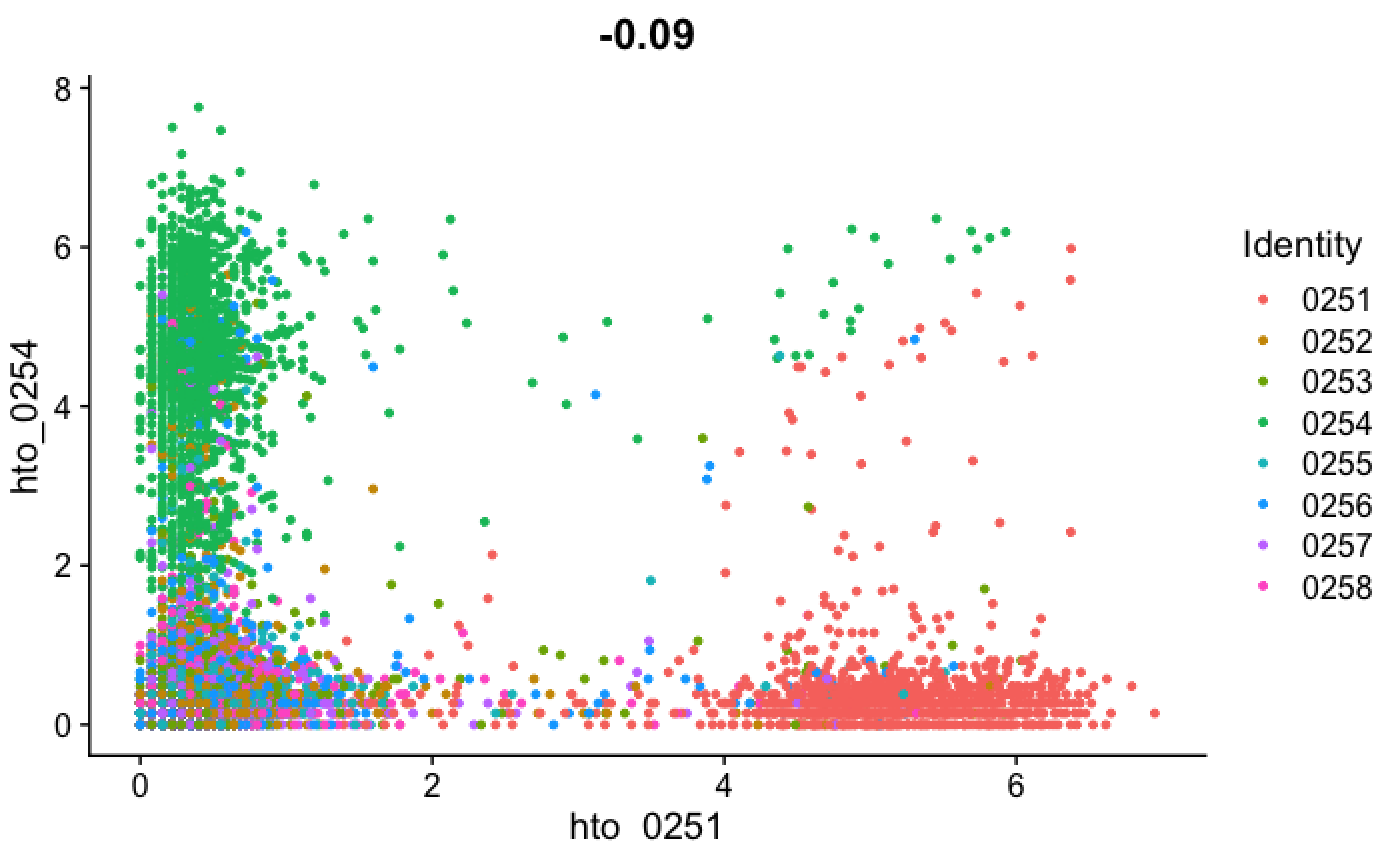
Hide

```
FeatureScatter(ex.hashtag, feature1 = "hto_0251", feature2 = "hto_0253")
```



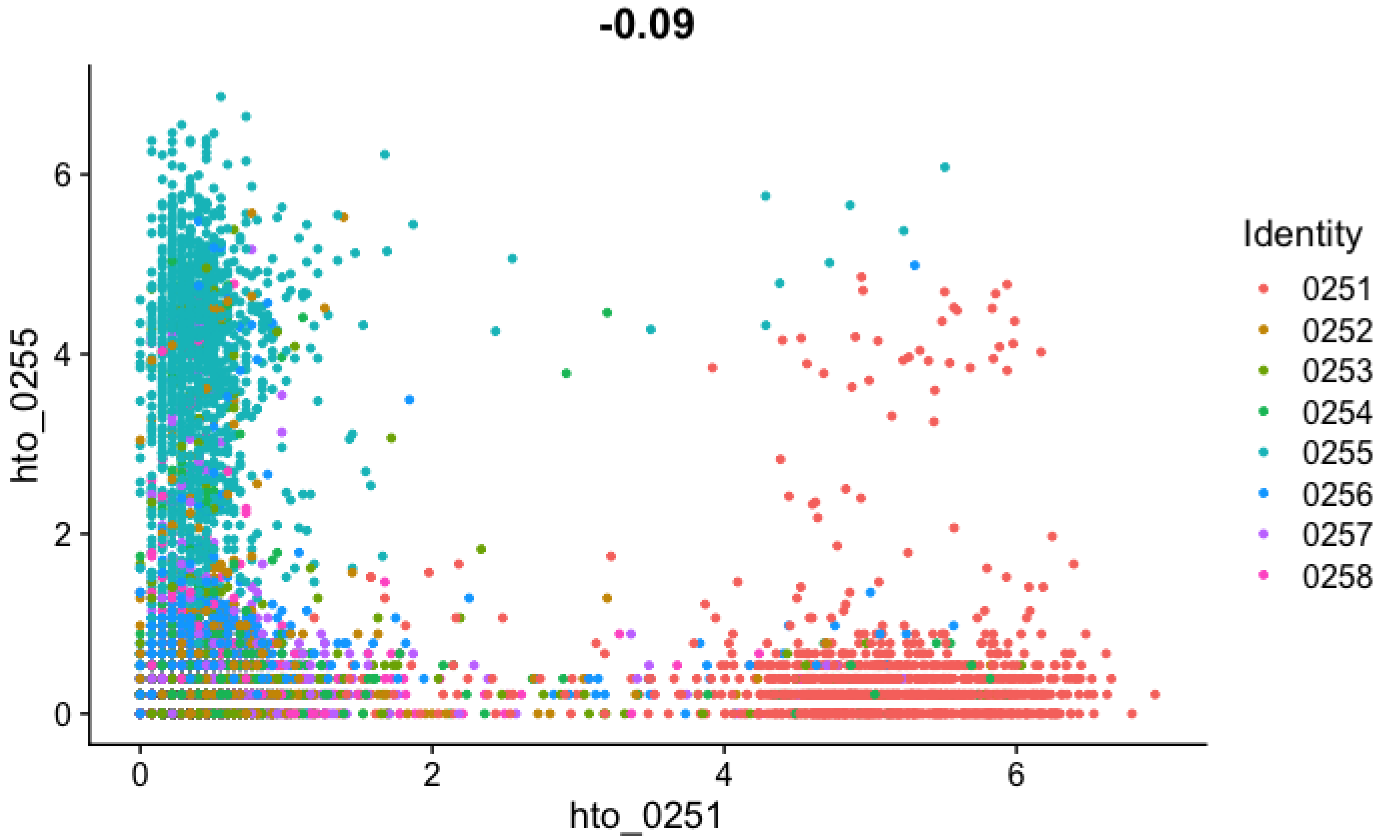

Hide

```
FeatureScatter(ex.hashtag, feature1 = "hto_0251", feature2 = "hto_0254")
```



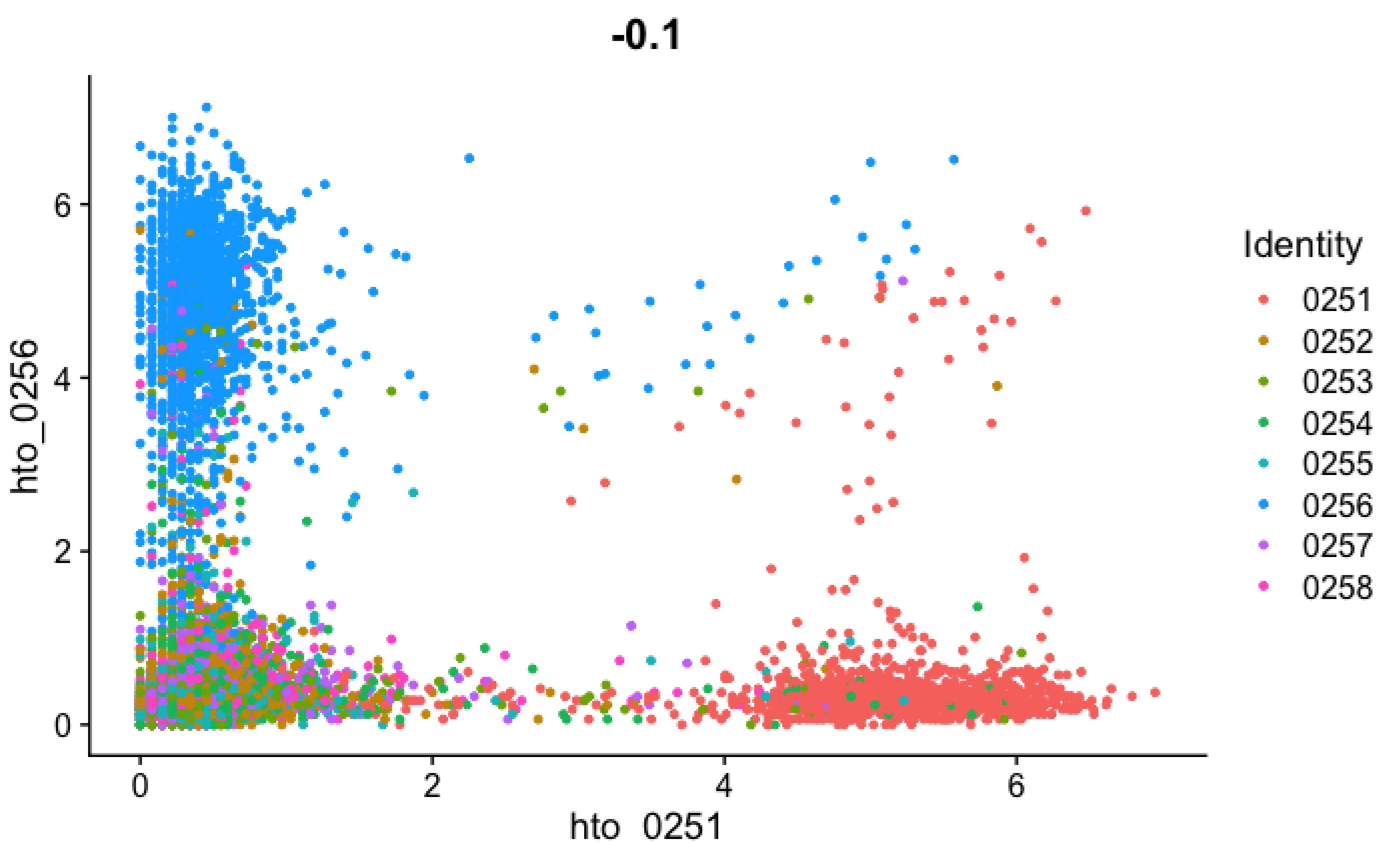

Hide

```
FeatureScatter(ex.hashtag, feature1 = "hto_0251", feature2 = "hto_0255")
```



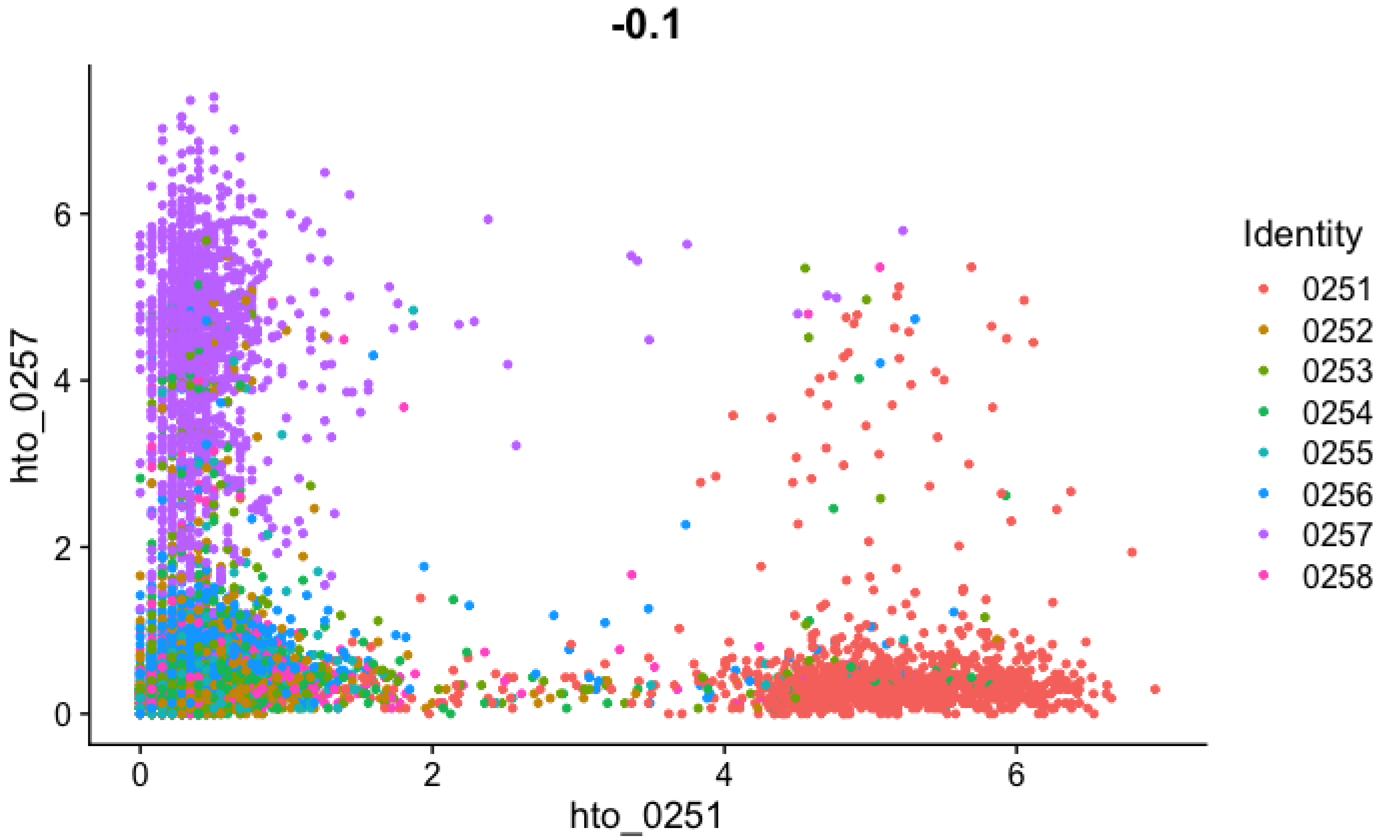

Hide

```
FeatureScatter(ex.hashtag, feature1 = "hto_0251", feature2 = "hto_0256")
```



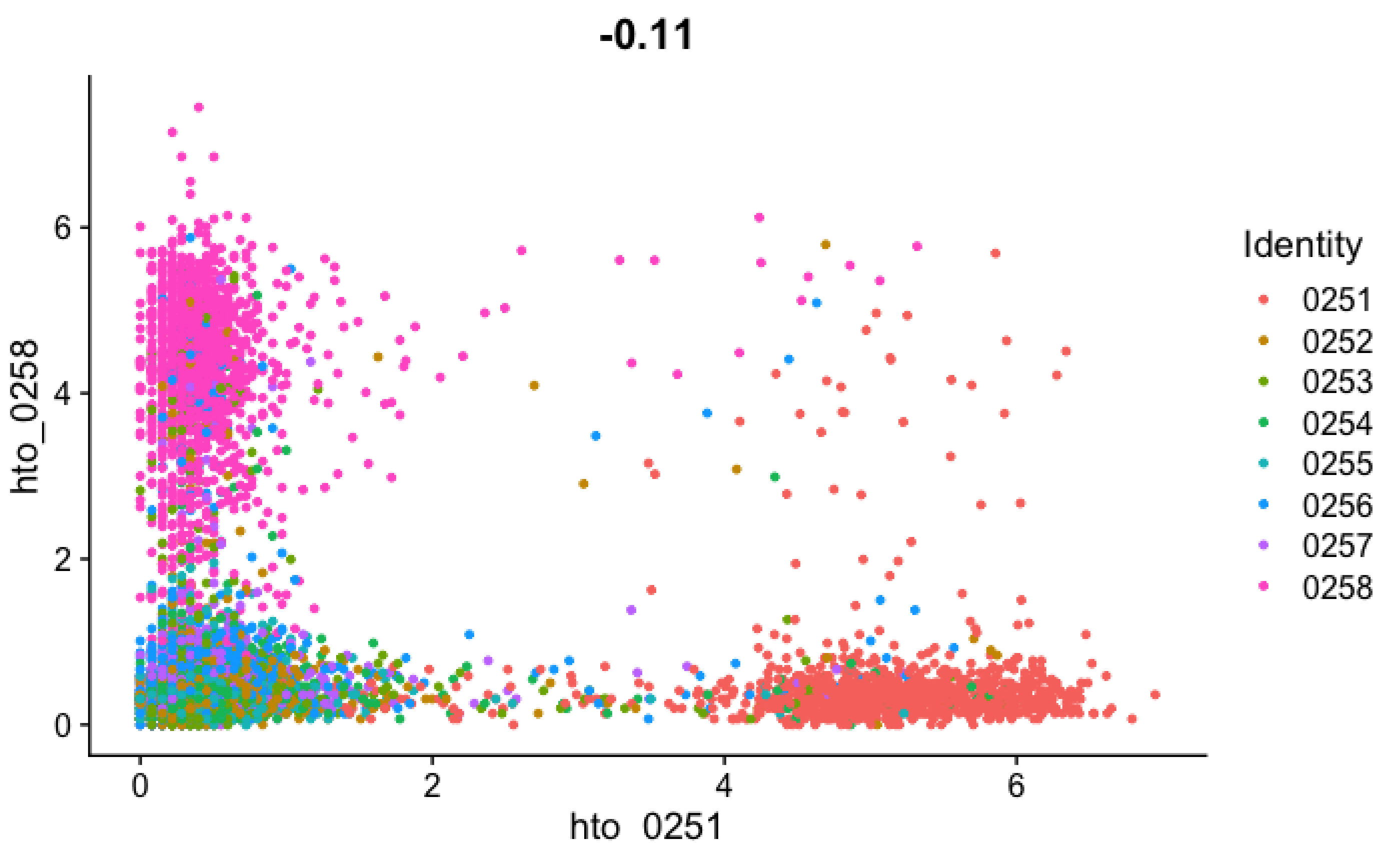

Hide

```
FeatureScatter(ex.hashtag, feature1 = "hto_0251", feature2 = "hto_0257")
```



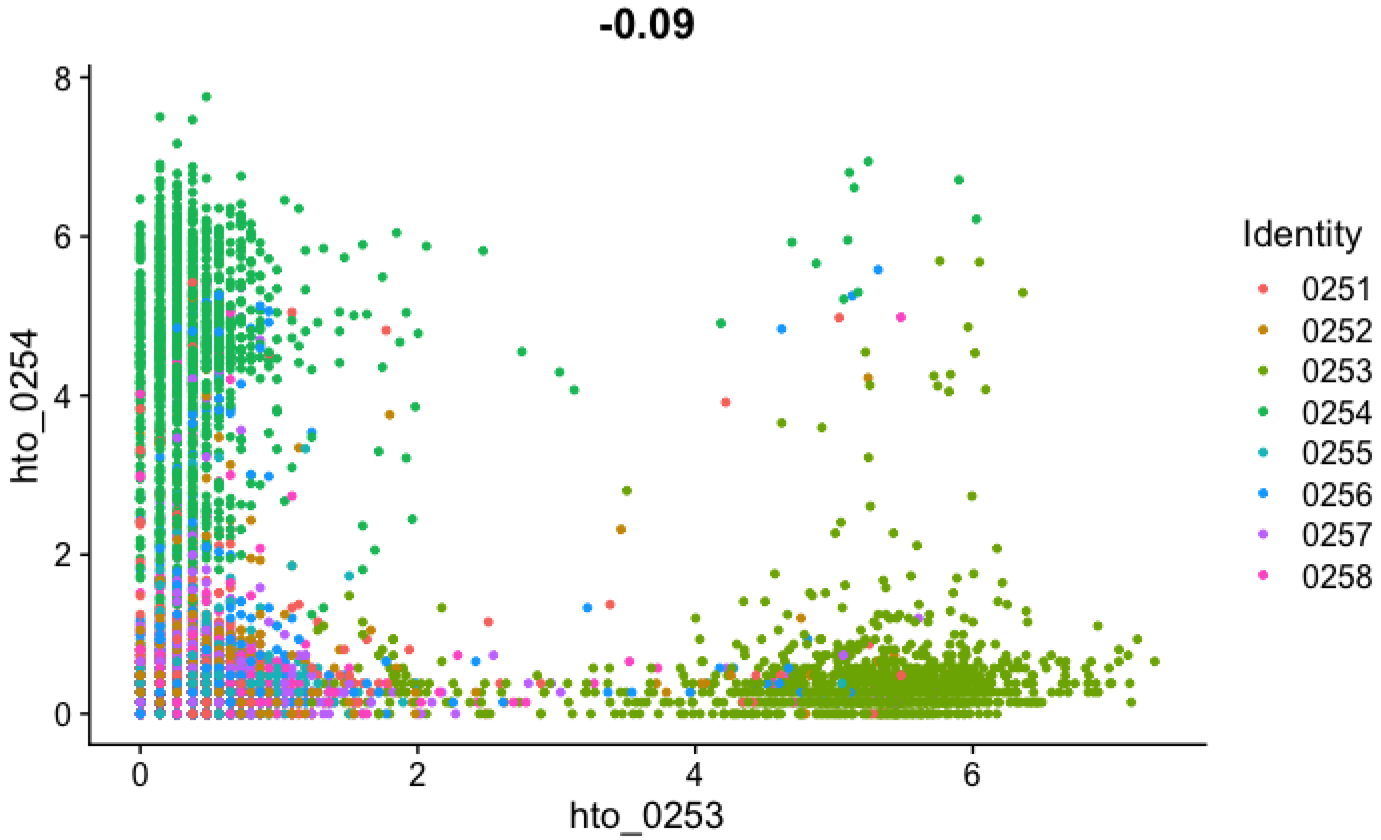

Hide

```
FeatureScatter(ex.hashtag, feature1 = "hto_0251", feature2 = "hto_0258")
```



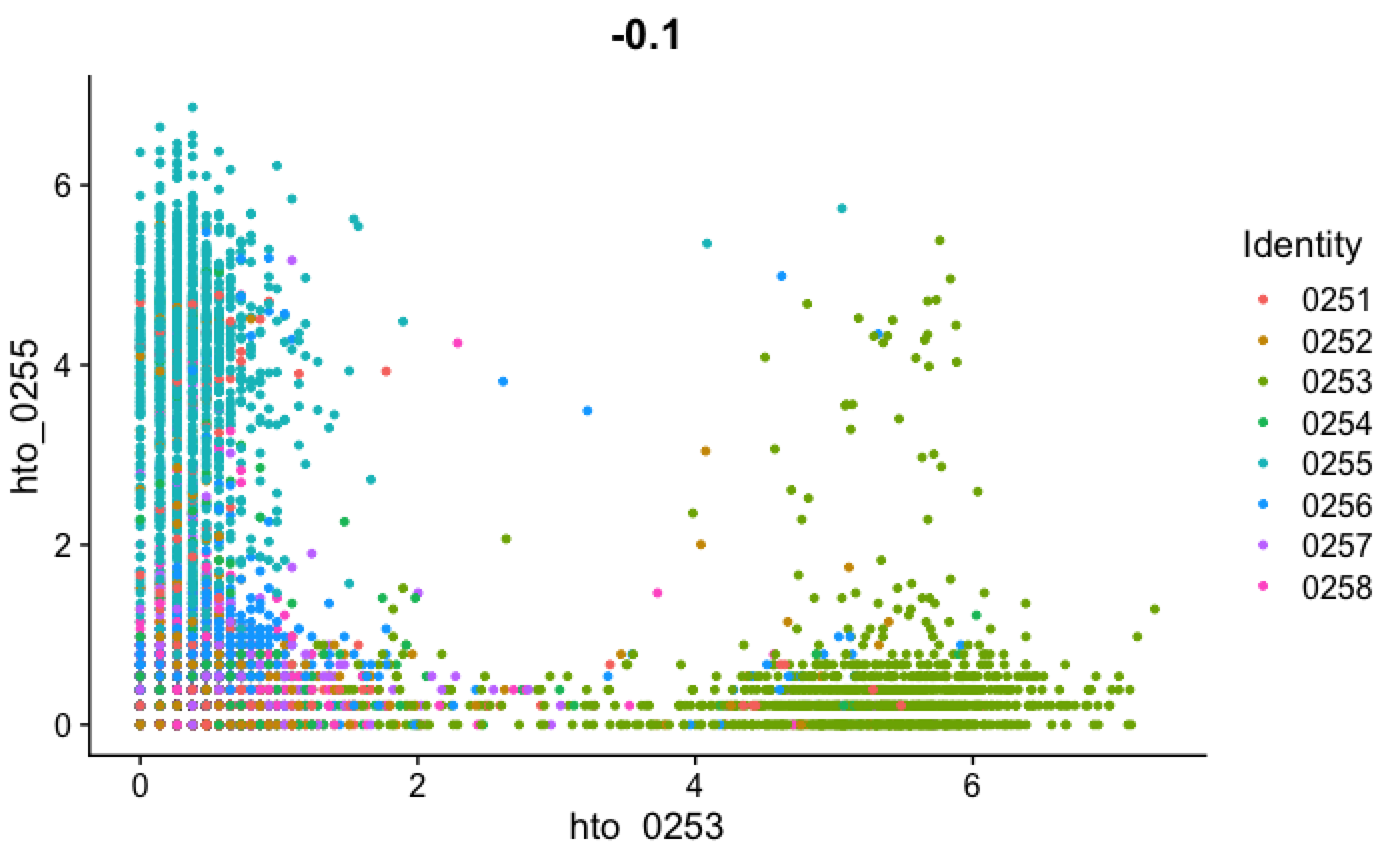

Hide

```
FeatureScatter(ex.hashtag, feature1 = "hto_0253", feature2 = "hto_0254")
```



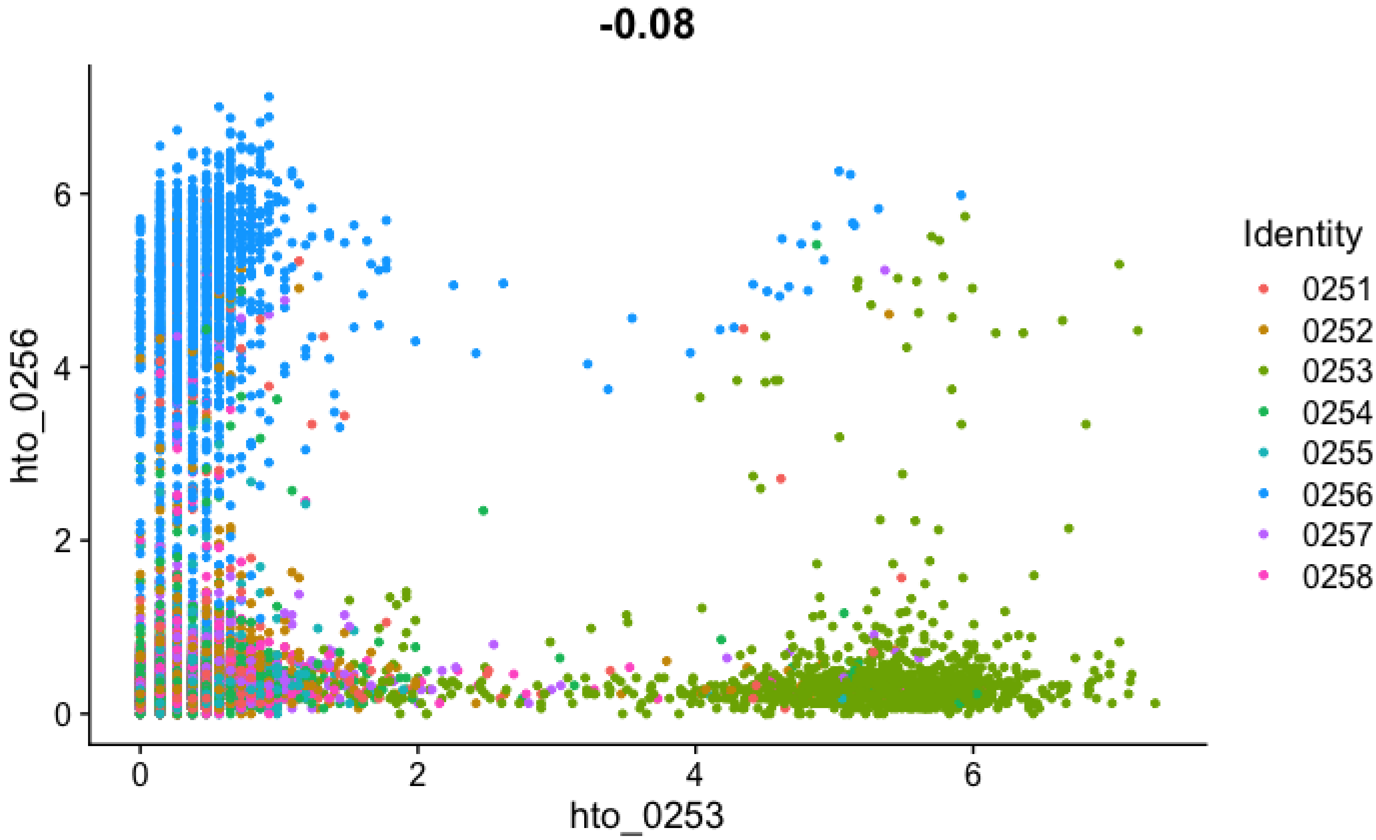

Hide

```
FeatureScatter(ex.hashtag, feature1 = "hto_0253", feature2 = "hto_0255")
```



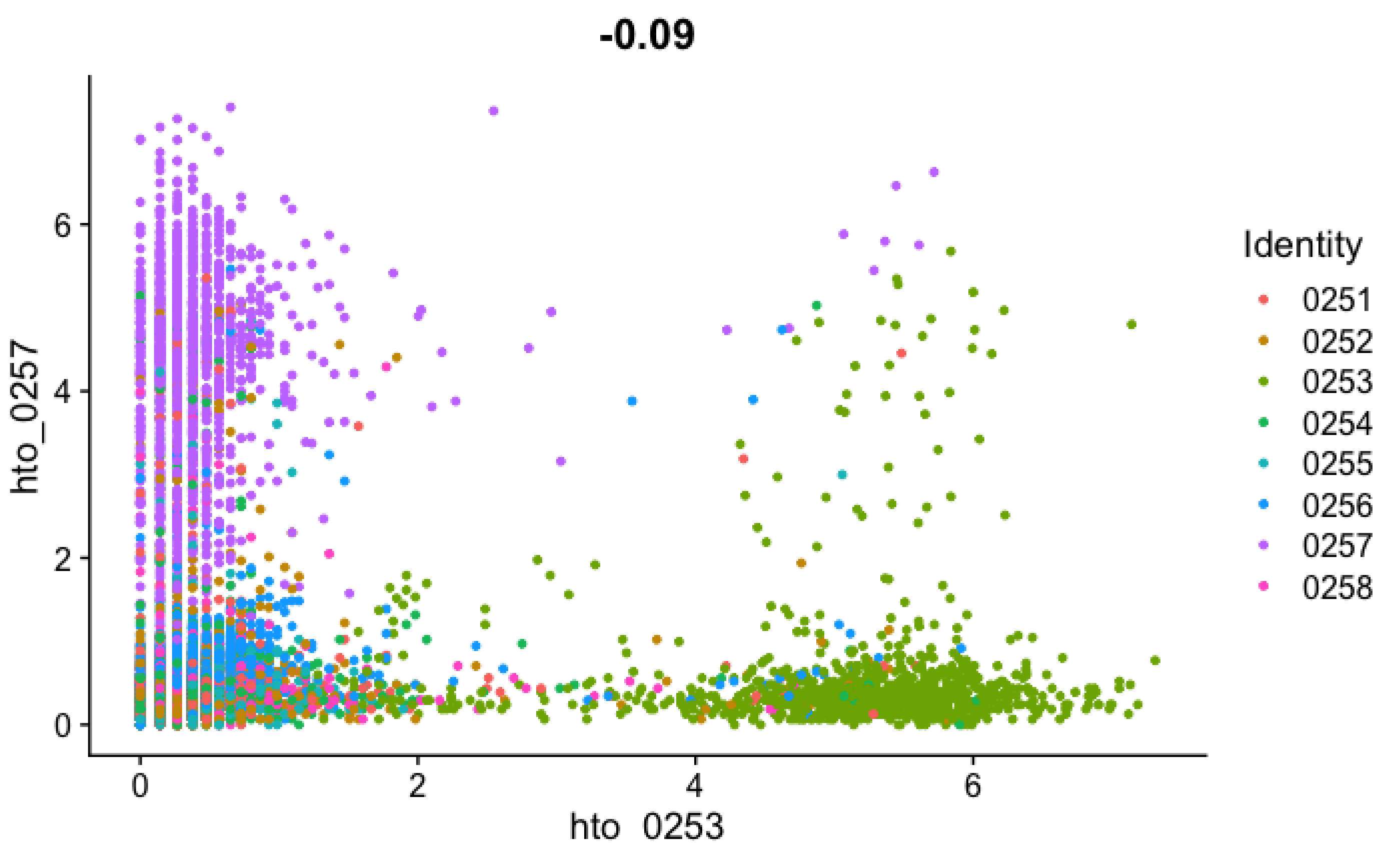

Hide

```
FeatureScatter(ex.hashtag, feature1 = "hto_0253", feature2 = "hto_0256")
```



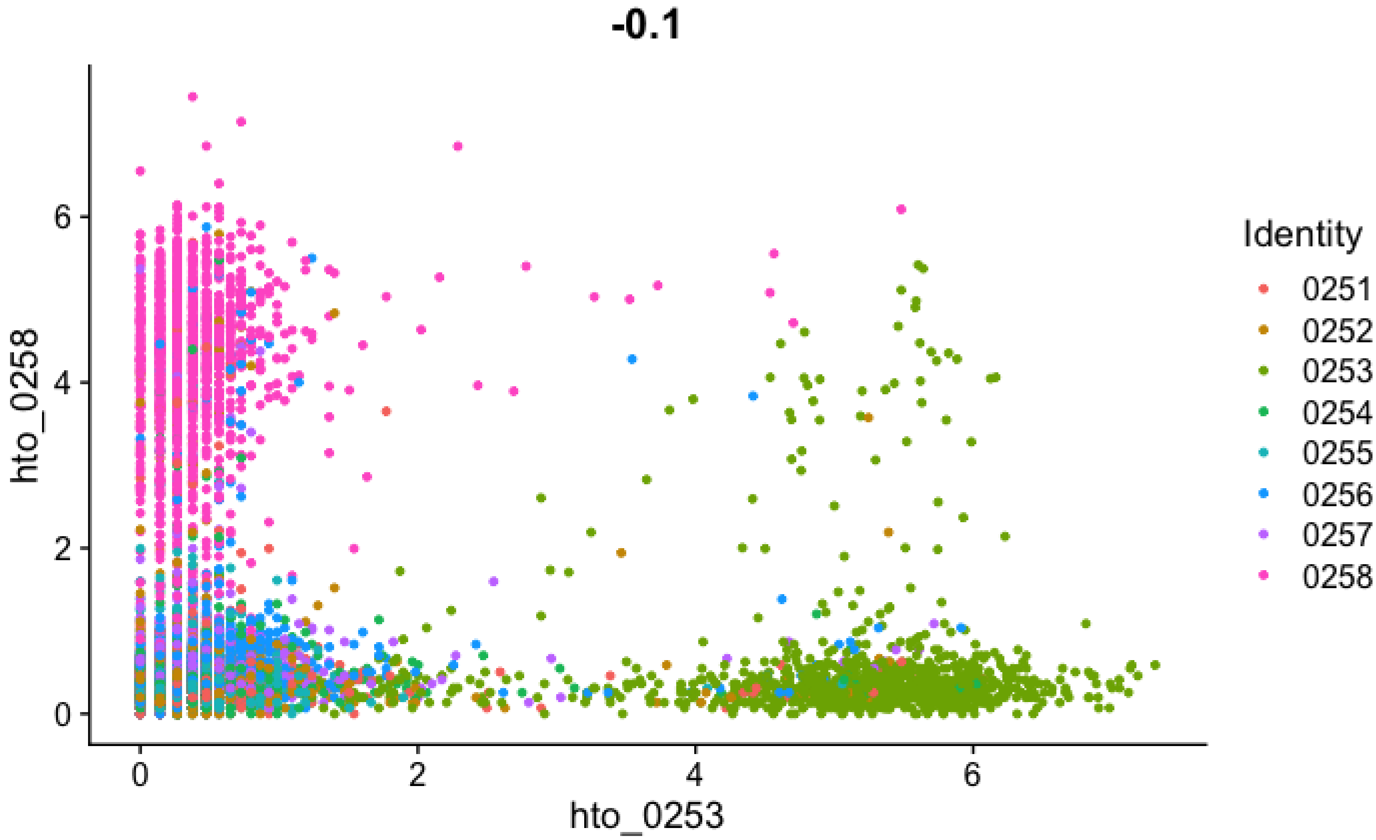

Hide

```
FeatureScatter(ex.hashtag, feature1 = "hto_0253", feature2 = "hto_0257")
```



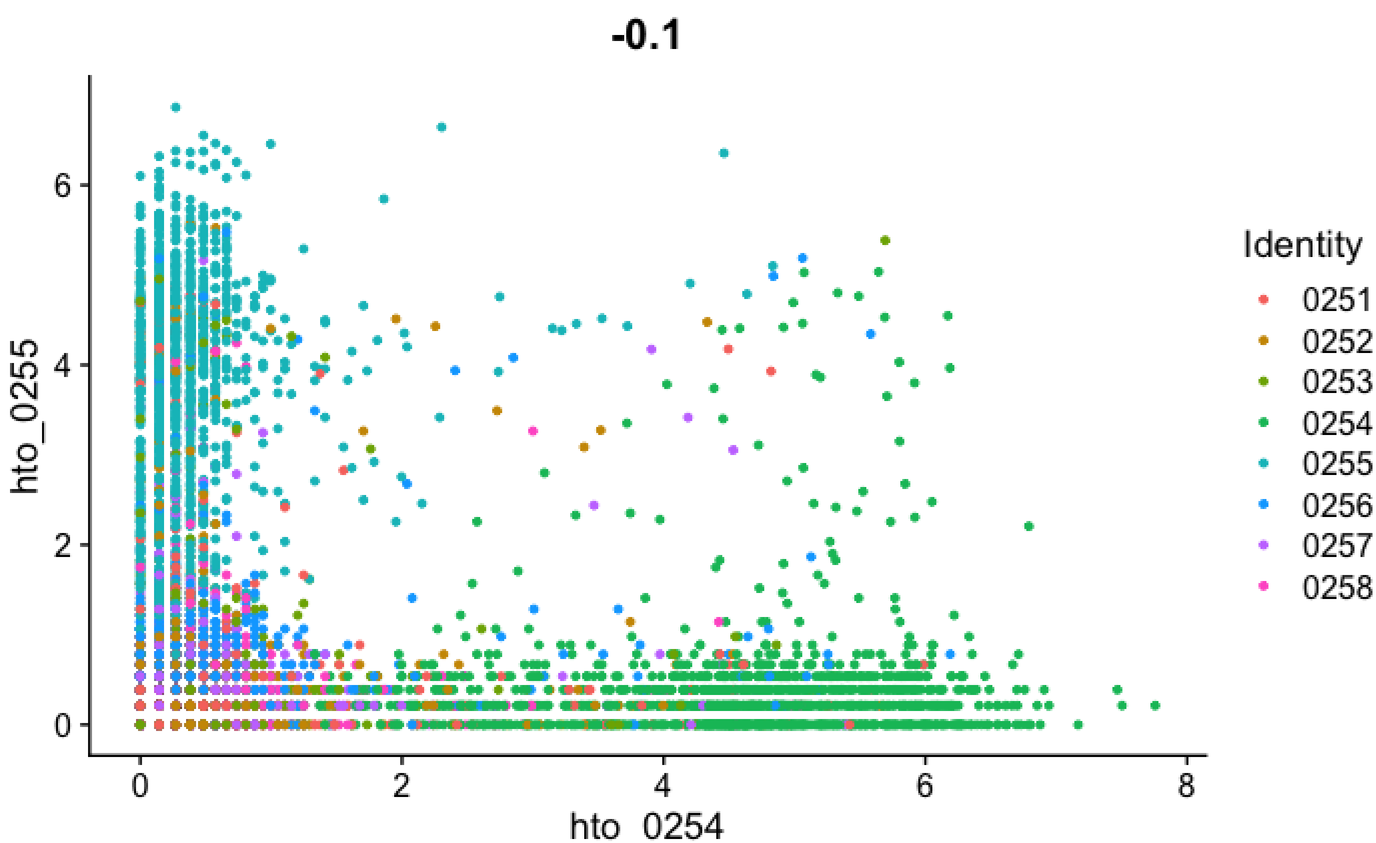

Hide

```
FeatureScatter(ex.hashtag, feature1 = "hto_0253", feature2 = "hto_0258")
```



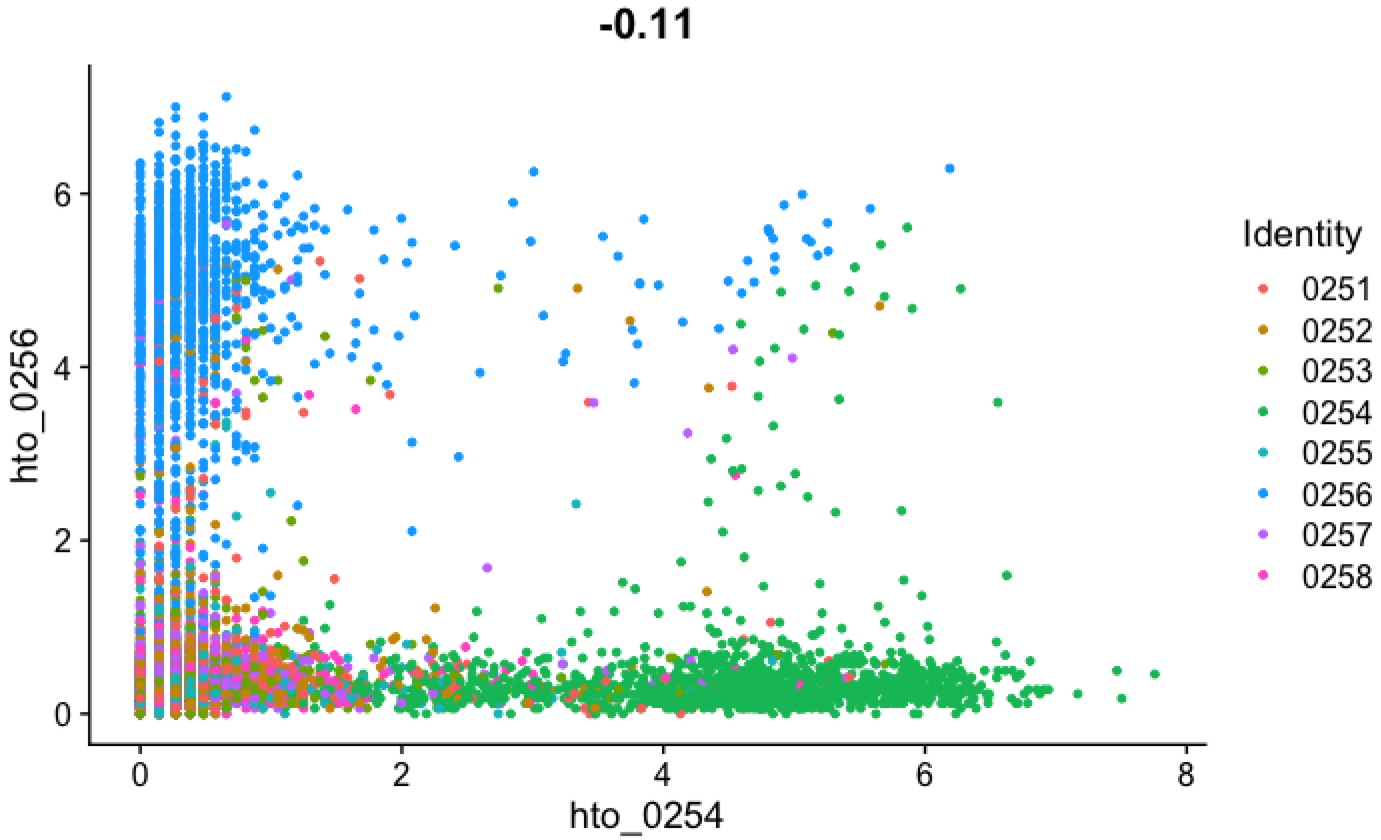

Hide

```
FeatureScatter(ex.hashtag, feature1 = "hto_0254", feature2 = "hto_0255")
```



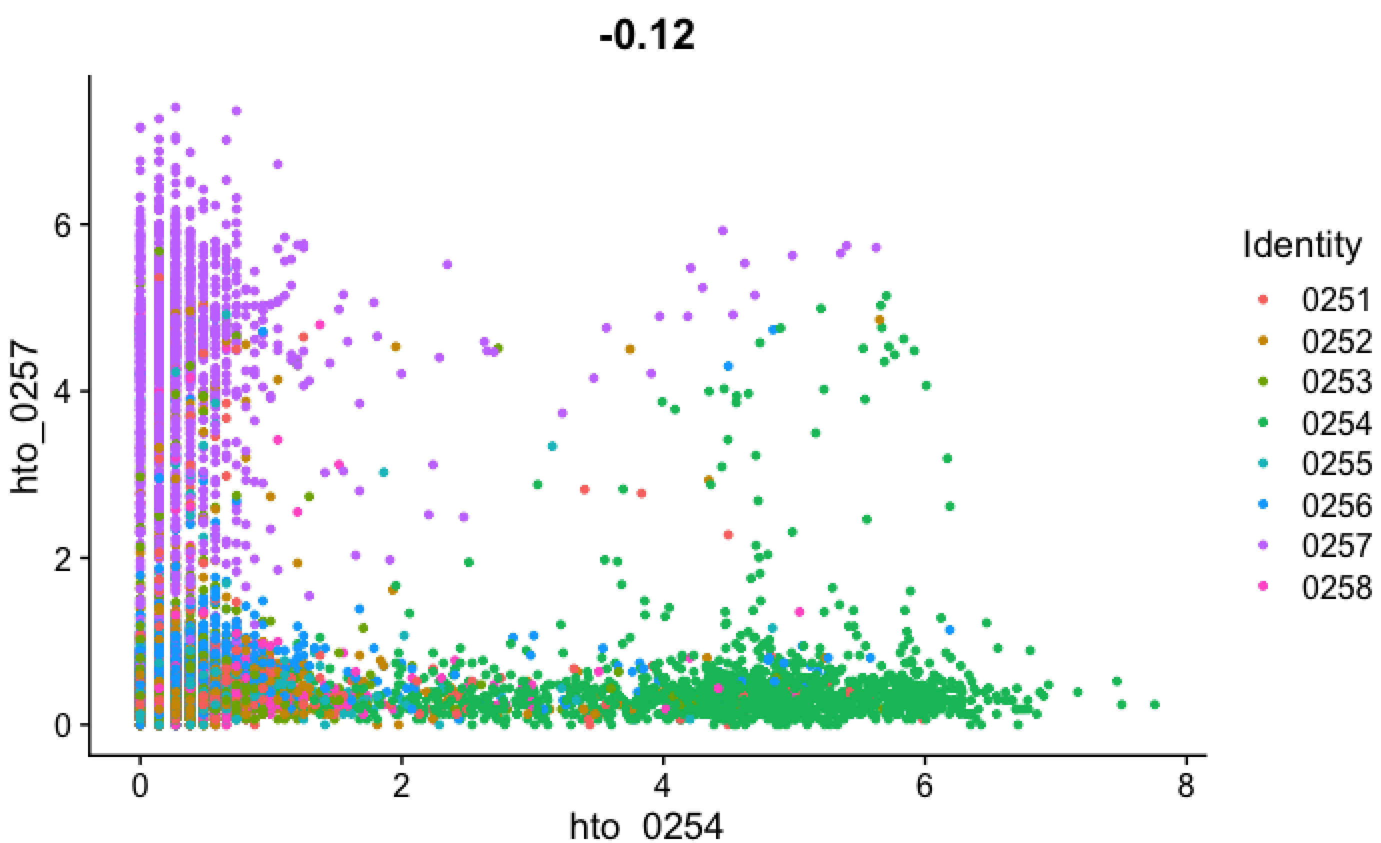

Hide

```
FeatureScatter(ex.hashtag, feature1 = "hto_0254", feature2 = "hto_0256")
```



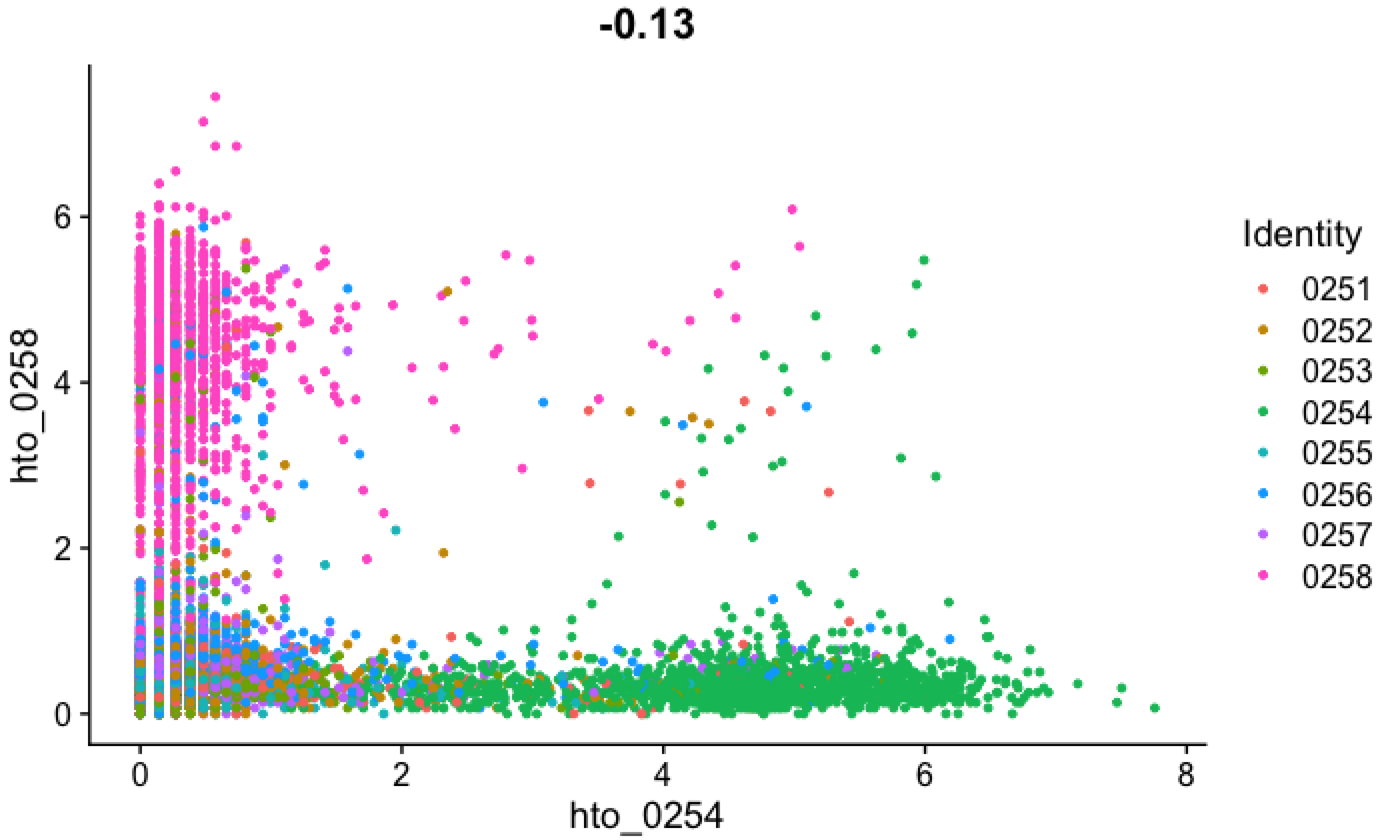

Hide

```
FeatureScatter(ex.hashtag, feature1 = "hto_0254", feature2 = "hto_0257")
```



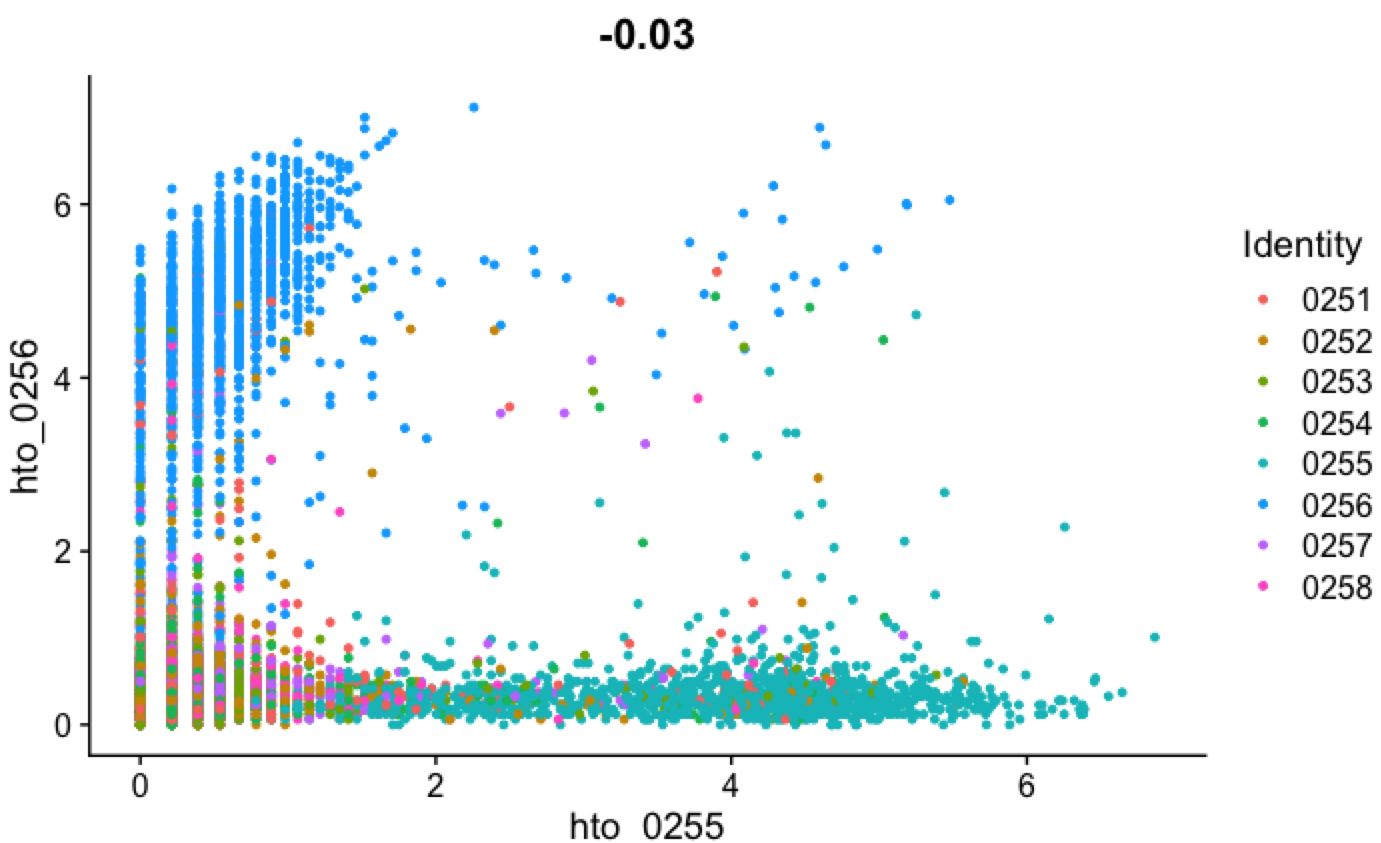

Hide

```
FeatureScatter(ex.hashtag, feature1 = "hto_0254", feature2 = "hto_0258")
```



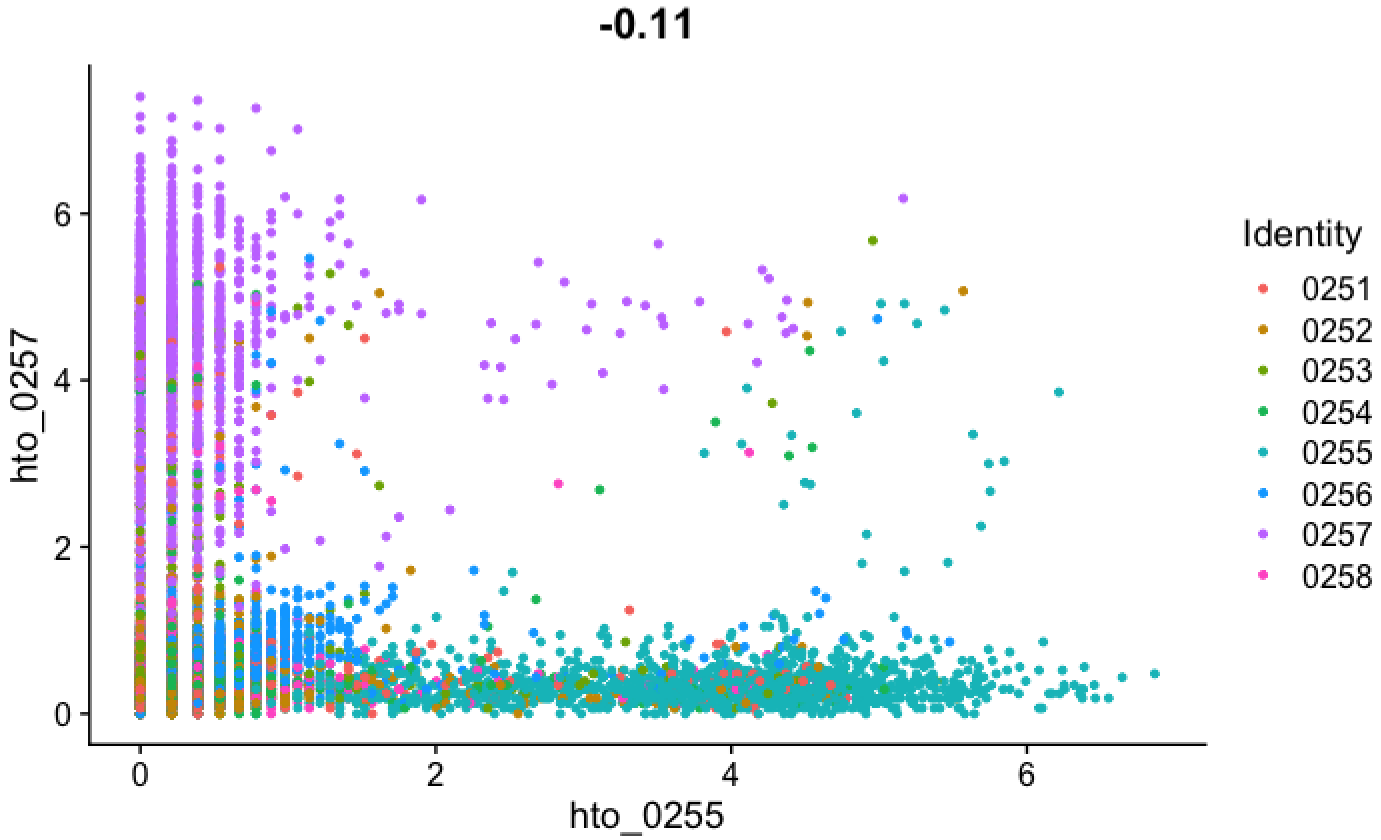

Hide

```
FeatureScatter(ex.hashtag, feature1 = "hto_0255", feature2 = "hto_0256")
```



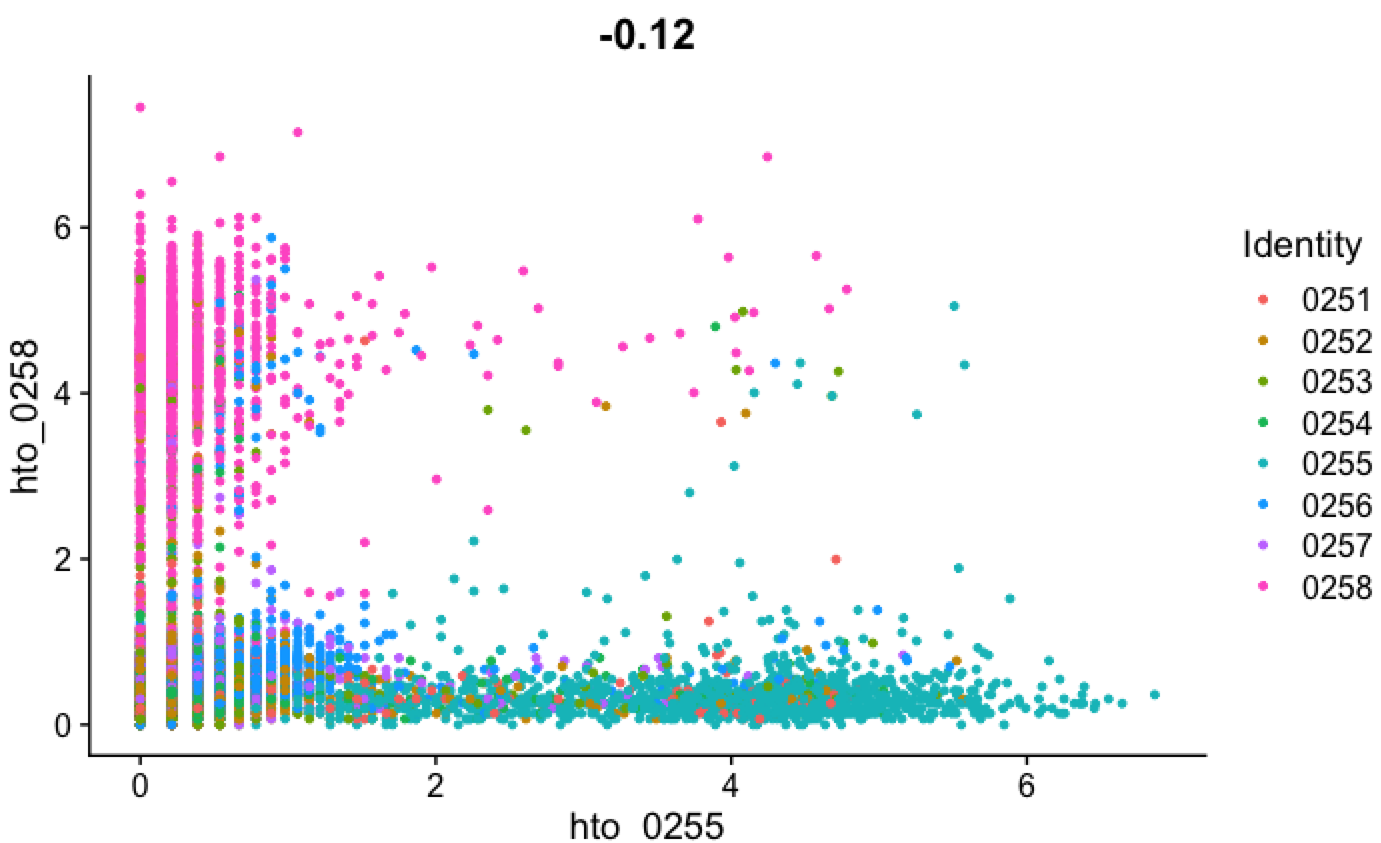

Hide

```
FeatureScatter(ex.hashtag, feature1 = "hto_0255", feature2 = "hto_0257")
```



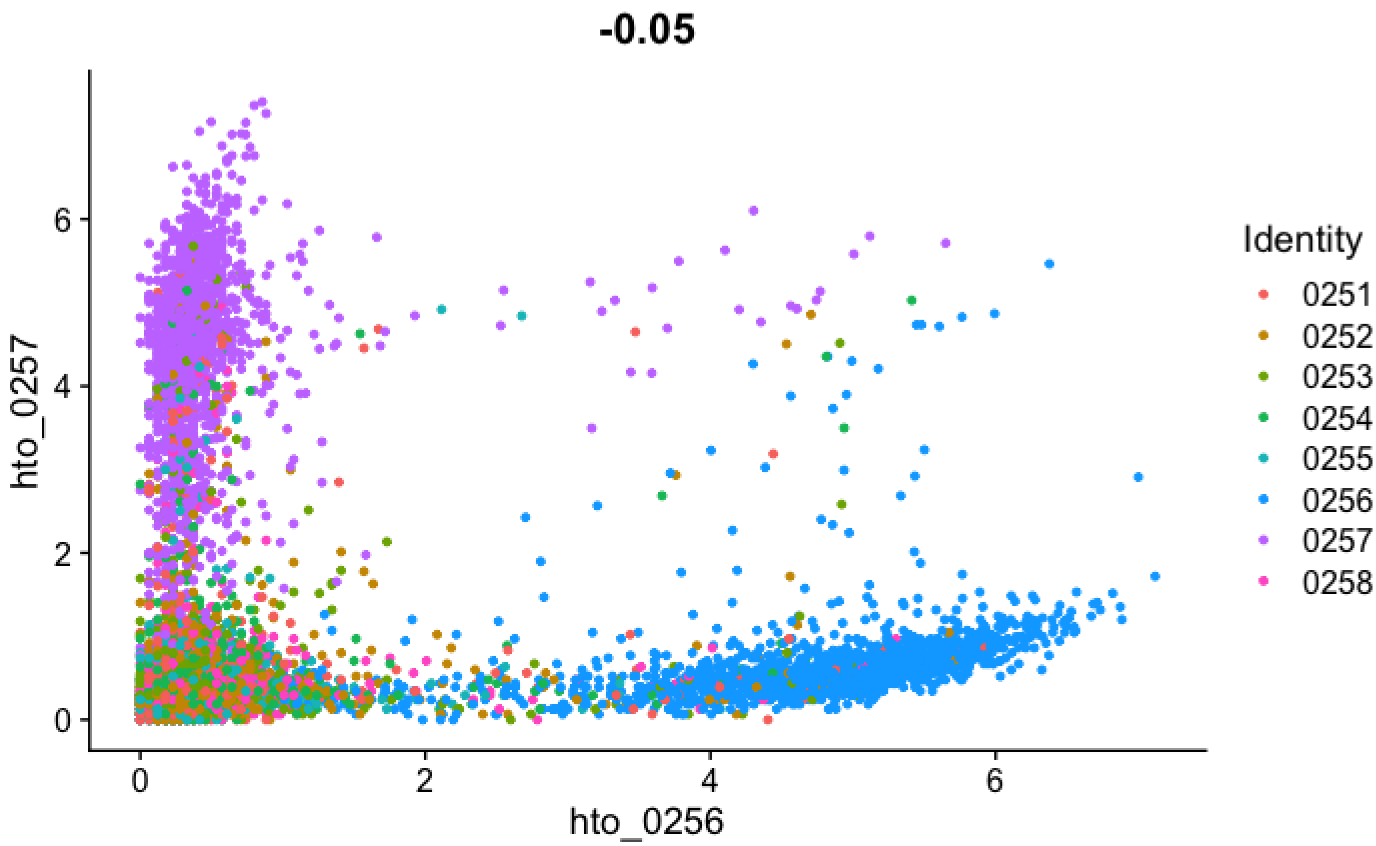

Hide

```
FeatureScatter(ex.hashtag, feature1 = "hto_0255", feature2 = "hto_0258")
```



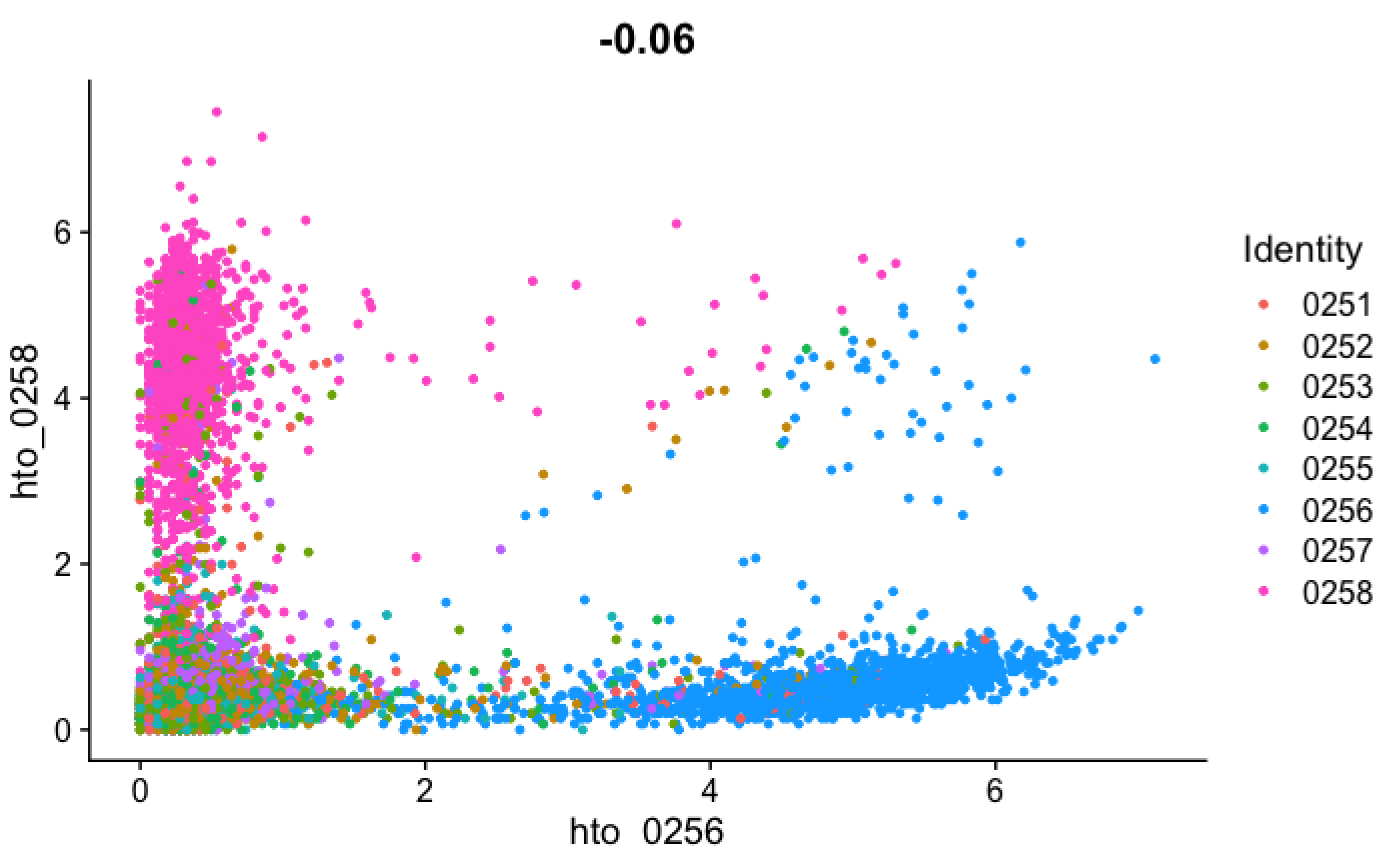

Hide

```
FeatureScatter(ex.hashtag, feature1 = "hto_0256", feature2 = "hto_0257")
```



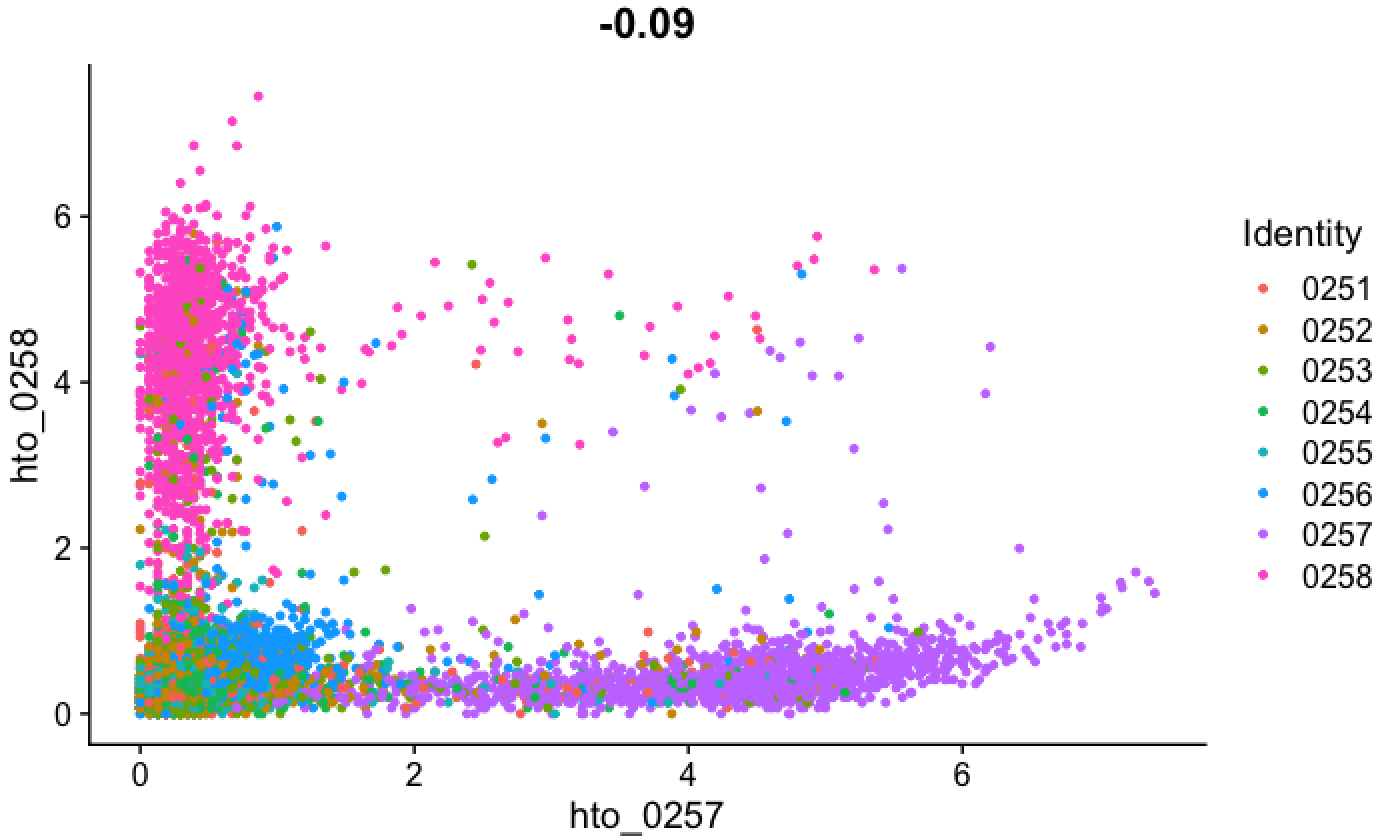

Hide

```
FeatureScatter(ex.hashtag, feature1 = "hto_0256", feature2 = "hto_0258")
```




Hide

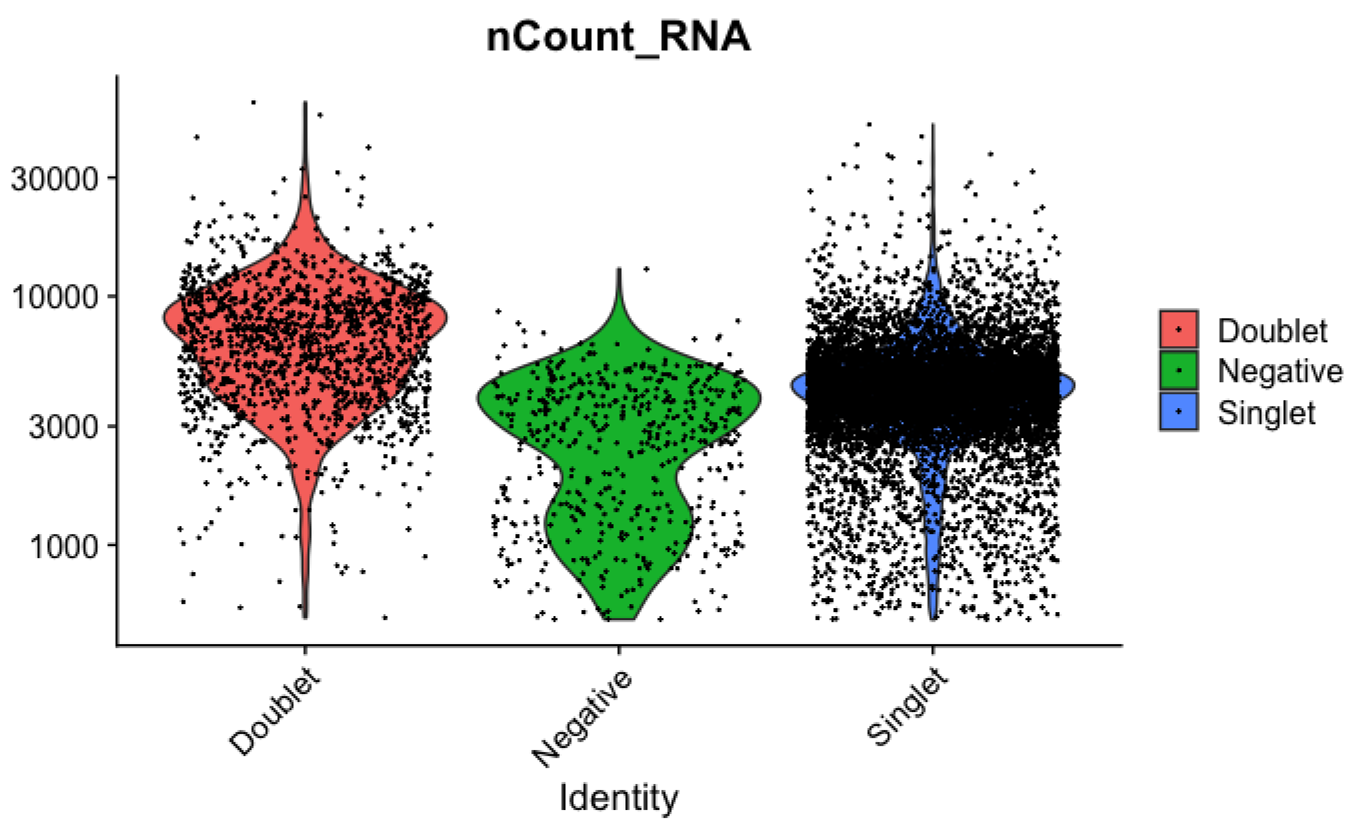
```
FeatureScatter(ex.hashtag, feature1 = "hto_0257", feature2 = "hto_0258")
```




Compare number of UMIs for singlets, doublets and negative cells

Hide

```
Idents(ex.hashtag) <- "HTO_classification.global"
VlnPlot(ex.hashtag, features = "nCount_RNA", pt.size = 0.1, log = TRUE)
```




Generate a two dimensional tSNE embedding for HTOs. Here we are grouping cells by singlets and doublets for simplicity.

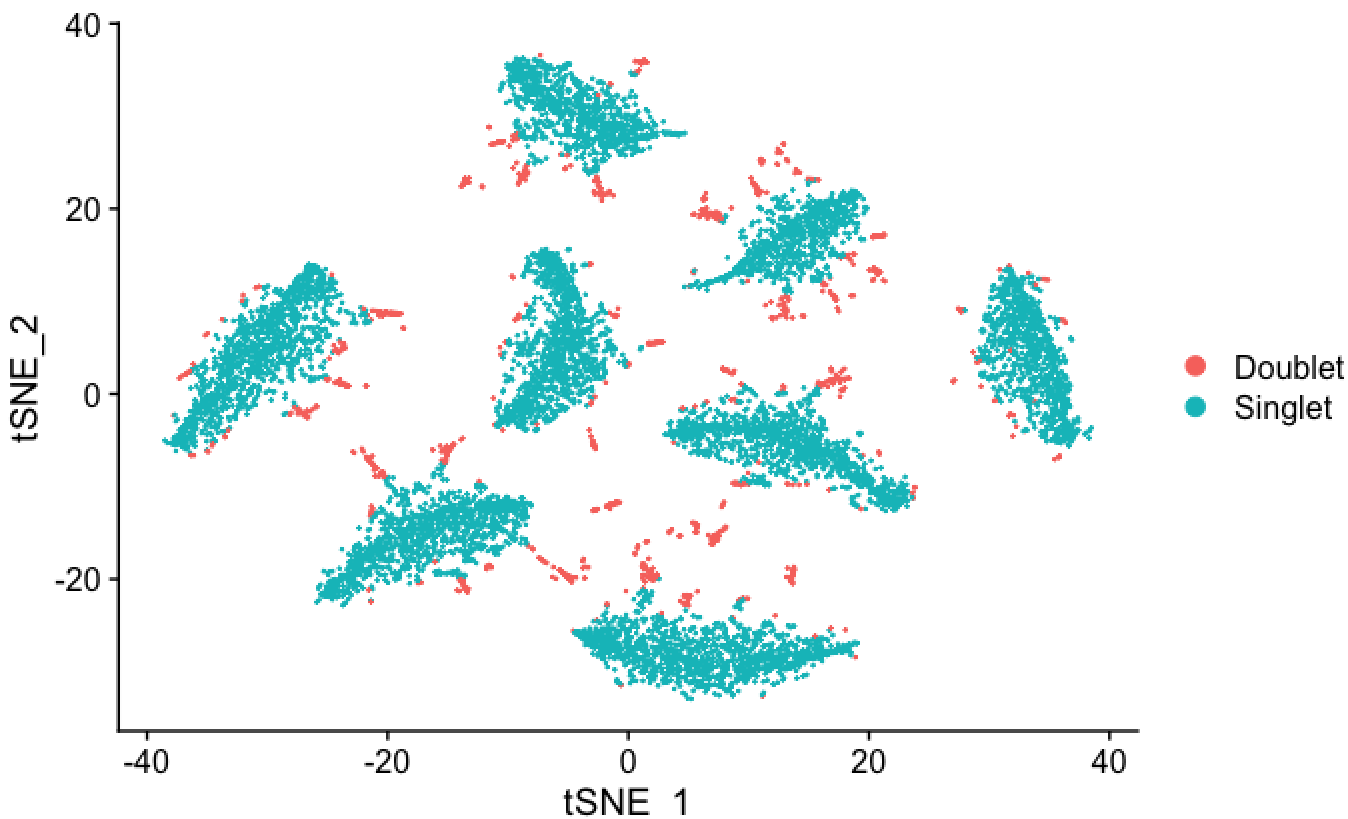
Hide

```
# First, we will remove negative cells from the object (if any)
ex.hashtag.subset <- subset(ex.hashtag, idents = "Negative", invert = TRUE)
# Calculate a distance matrix using HTO
# (use the subset if you just created in case you had negative cells removed)
hto.dist.mtx <- as.matrix(dist(t(GetAssayData(object = ex.hashtag.subset, assay = "HTO"))))
# Calculate tSNE embeddings with a distance matrix
# (use the subset if you just created in case you had negative cells removed)
ex.hashtag.subset <- RunTSNE(ex.hashtag.subset, distance.matrix = hto.dist.mtx, perplexity = 100)
```

```
Adding a command log without an assay associated with it
```

Hide

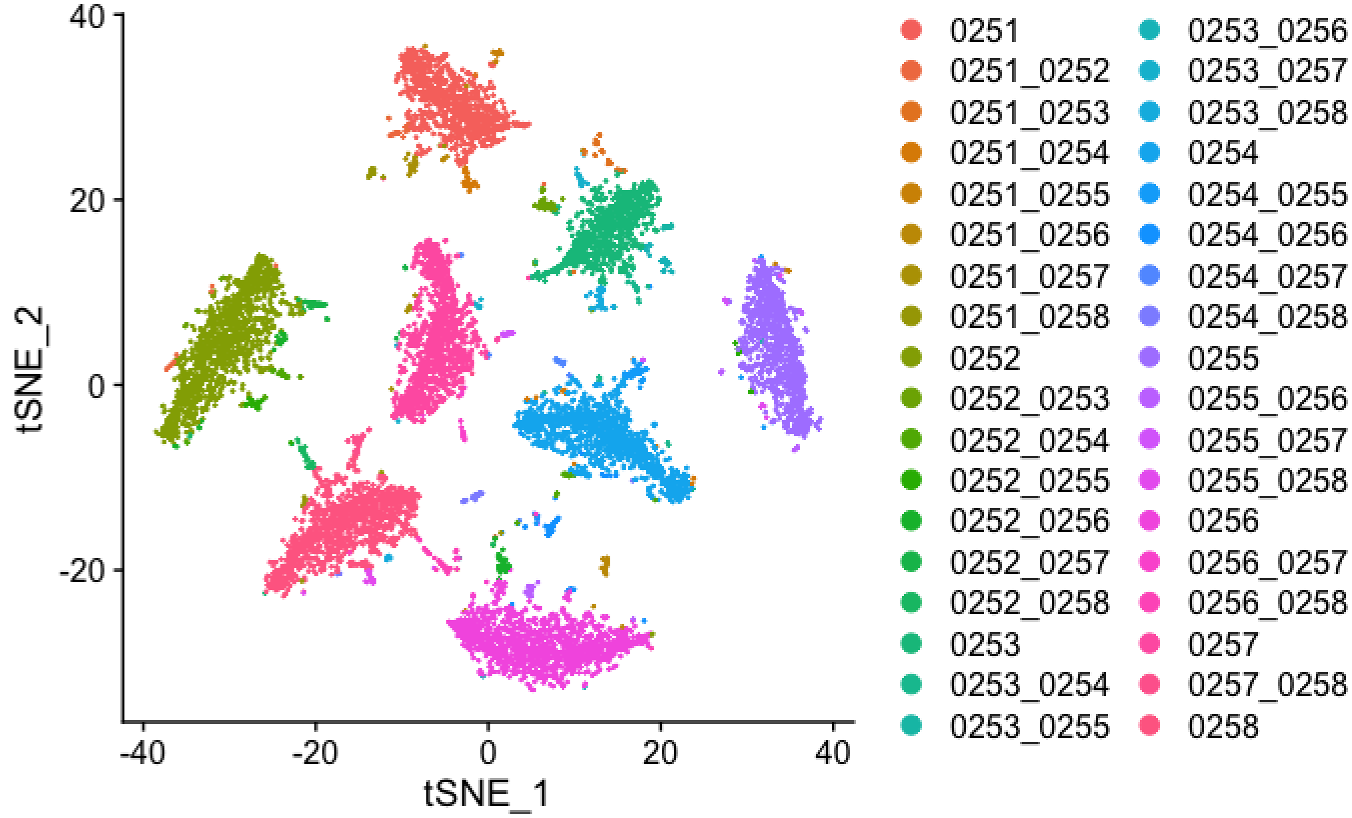
```
DimPlot(ex.hashtag.subset)
```




Visualize the more detailed classification result. Here, you should be able to see that each of the small clouds on the tSNE plot corresponds to one of the possible doublet combinations.

Hide

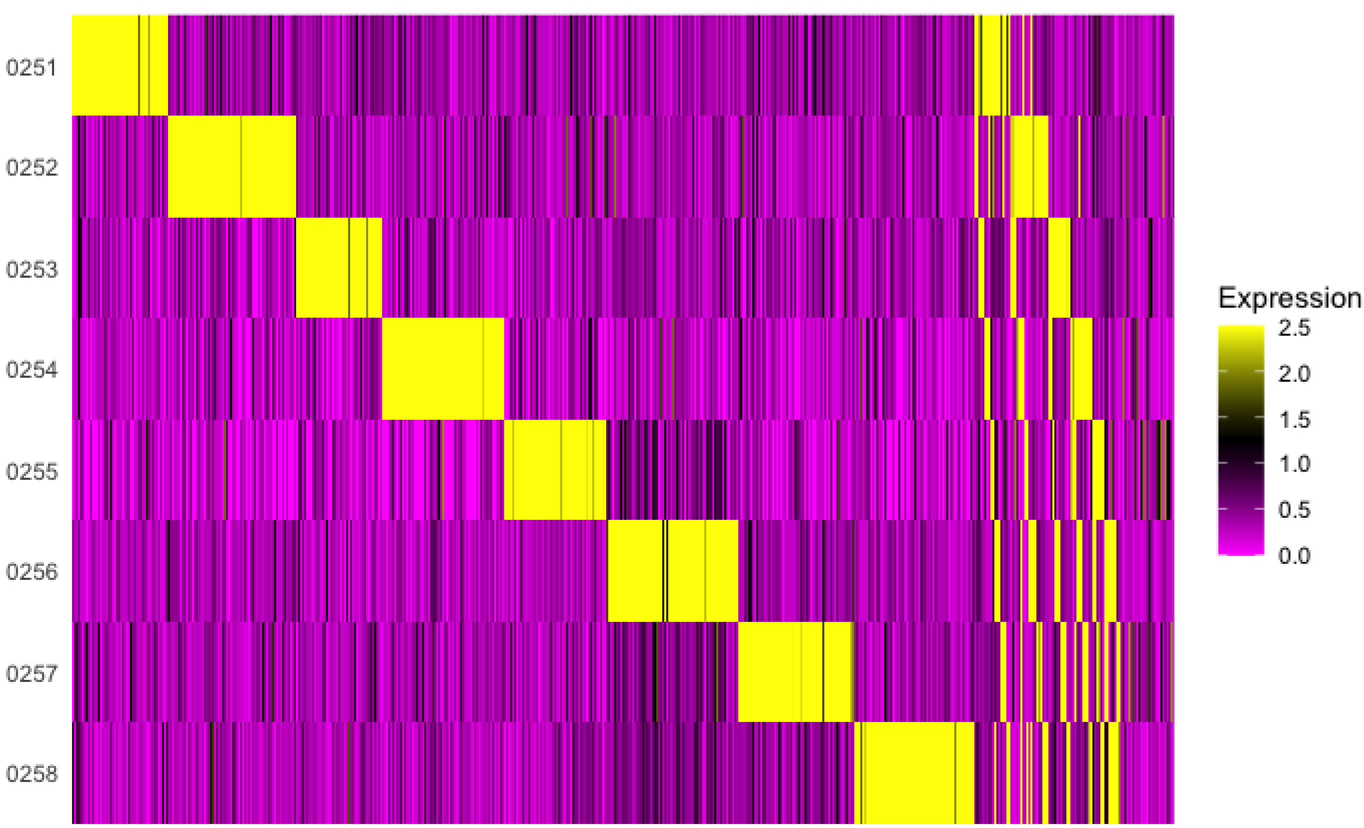
```
Idents(ex.hashtag.subset) <- 'HTO_classification'
DimPlot(ex.hashtag.subset)
```




Create an HTO heatmap, based on Figure 1C in the Cell Hashing paper.

Hide

```
# To increase the efficiency of plotting, you can subsample cells using the num.cells argument
HTOHeatmap(ex.hashtag, assay = "HTO", ncells = n_cells)
```




Cluster and visualize cells using the usual scRNA-seq workflow, and examine for the potential presence of batch effects.

Hide

```
# Extract the singlets
ex.singlets <- subset(ex.hashtag, idents = "Singlet")
# Select the top 1000 most variable features
ex.singlets <- FindVariableFeatures(ex.singlets, selection.method = "mean.var.plot")
```

```
Calculating gene means
0%   10   20   30   40   50   60   70   80   90   100%
[----|----|----|----|----|----|----|----|----|----|
**************************************************|
Calculating gene variance to mean ratios
0%   10   20   30   40   50   60   70   80   90   100%
[----|----|----|----|----|----|----|----|----|----|
**************************************************|
```

Hide

```
# Scaling RNA data, we only scale the variable features here for efficiency
ex.singlets <- ScaleData(ex.singlets, features = VariableFeatures(ex.singlets))
```

```
Centering and scaling data matrix


  |
  |
|   0%
  |
  |==================================================
|  50%
  |
  |===============================================================================================
======| 100%
```

Hide

```
# Run PCA
ex.singlets <- RunPCA(ex.singlets, features = VariableFeatures(ex.singlets))
```

```
PC_ 1
Positive:  IFITM1, CD247, RPS12, CTSW, GZMA, CST7, CCL5, PRF1, MT-CYB, KLRB1
	   NKG7, MT-ATP8, KLRD1, HOPX, CD8A, GNLY, KLRG1, CD8B, GZMB, IL2RB
	   GZMH, FGFBP2, SPON2, TBX21, STMN1, GRAP2, CLIC3, NCR3, KLRF1, S1PR5
Negative:  LYZ, FCN1, CST3, S100A9, IFI30, S100A8, VCAN, MNDA, SERPINA1, SPI1
	   CTSS, AIF1, LST1, TYMP, CSTA, CD14, CEBPD, CD68, TNFAIP2, MS4A6A
	   CYBB, SAT1, TMEM176B, CSF3R, CFD, FGL2, S100A12, PSAP, KLF4, TYROBP
PC_ 2
Positive:  NKG7, PRF1, GNLY, GZMB, CST7, KLRD1, FGFBP2, GZMA, SPON2, CTSW
	   FCGR3A, GZMH, KLRF1, CLIC3, HOPX, CCL5, ADGRG1, S1PR5, EFHD2, CCL4
	   CX3CR1, TBX21, PRSS23, IL2RB, SH2D1B, IGFBP7, FCRL6, PTGDR, C1orf21, CD247
Negative:  CD79A, MS4A1, BANK1, LINC00926, FCRLA, RALGPS2, BLK, SPIB, RPS12, AFF3
	   IGHM, CD19, POU2AF1, P2RX5, NIBAN3, TNFRSF13C, CD24, TNFRSF13B, HLA-DQA1, CD22
	   COBLL1, CD79B, SWAP70, IGKC, TCF4, JCHAIN, FCER2, ITM2C, HLA-DRA, IGHA1
PC_ 3
Positive:  MS4A1, CD79A, BANK1, FCRLA, HLA-DQA1, LINC00926, SPIB, BLK, CD19, CD79B
	   RALGPS2, IGHM, NIBAN3, TNFRSF13B, HLA-DPA1, POU2AF1, CD24, TNFRSF13C, SWAP70, CD22
	   HLA-DPB1, P2RX5, TCF4, CD74, HLA-DRA, PDLIM1, COBLL1, AFF3, HLA-DQB1, IGKC
Negative:  TNFAIP3, IFITM1, FOS, CHRM3-AS2, SLC40A1, RPS12, TSHZ2, TNFRSF25, ANKRD55, S100A12
	   ADTRP, TSPO, LRRN3, S100A8, JUN, S100A9, VCAN, AIF1, S100A11, TRBV20-1
	   AL138963.4, NFKBIZ, CSF3R, S100A4, LINC02446, FAAH2, TRAV8-3, THBS1, HNRNPLL, CD14
PC_ 4
Positive:  MT-CO3, S100A12, VCAN, MT-ND3, CSF3R, AC020916.1, MT-ATP6, AC007952.4, MT-CO2, MTRNR2L1
2
	   FOSB, S100A8, MT-CO1, NCF1, LINC00937, JUN, MT-CYB, CRISPLD2, CXCL8, AC253572.2
	   AC245014.3, NAIP, PADI4, THBS1, CLEC4E, DYSF, CD14, STAB1, PLBD1, NLRP12
Negative:  CDKN1C, TCF7L2, HMOX1, SIGLEC10, FCGR3A, SMIM25, MS4A7, IFITM3, CXCL16, RRAS
	   CAMK1, ACTB, HLA-DPA1, SECTM1, MAFB, FAM110A, RPS12, FTH1, TBC1D8, EPB41L3
	   LMO2, CD68, LST1, HLA-DRB5, FTL, HLA-DPB1, LILRA1, SERPINA1, AC090559.1, SLC2A6
PC_ 5
Positive:  RPS12, ACTB, S100A12, FTL, FTH1, THBS1, JUN, PLBD1, TSPO, RBP7
	   SAP30, MCEMP1, S100A9, PADI4, IL1R2, GSTP1, FOLR3, DUSP1, FOS, S100A8
	   CD163, CSTA, MGST1, NCF1, AREG, TKT, S100A4, AC020656.1, FPR1, NRGN
Negative:  MT-CO1, MT-CO2, MT-CYB, MT-CO3, MT-ATP6, MTRNR2L12, MT-ND3, MT-ATP8, MT-ND6, XIST
	   CDKN1C, TCF7L2, MS4A7, MTRNR2L8, IL1B, NR4A1, NFKBIZ, TNF, NEAT1, CCL3L1
	   SIGLEC10, ADGRE2, ZEB2, CD83, LINC00342, CCL3, SLC2A6, BCL2A1, CD300E, FCGR3A
```

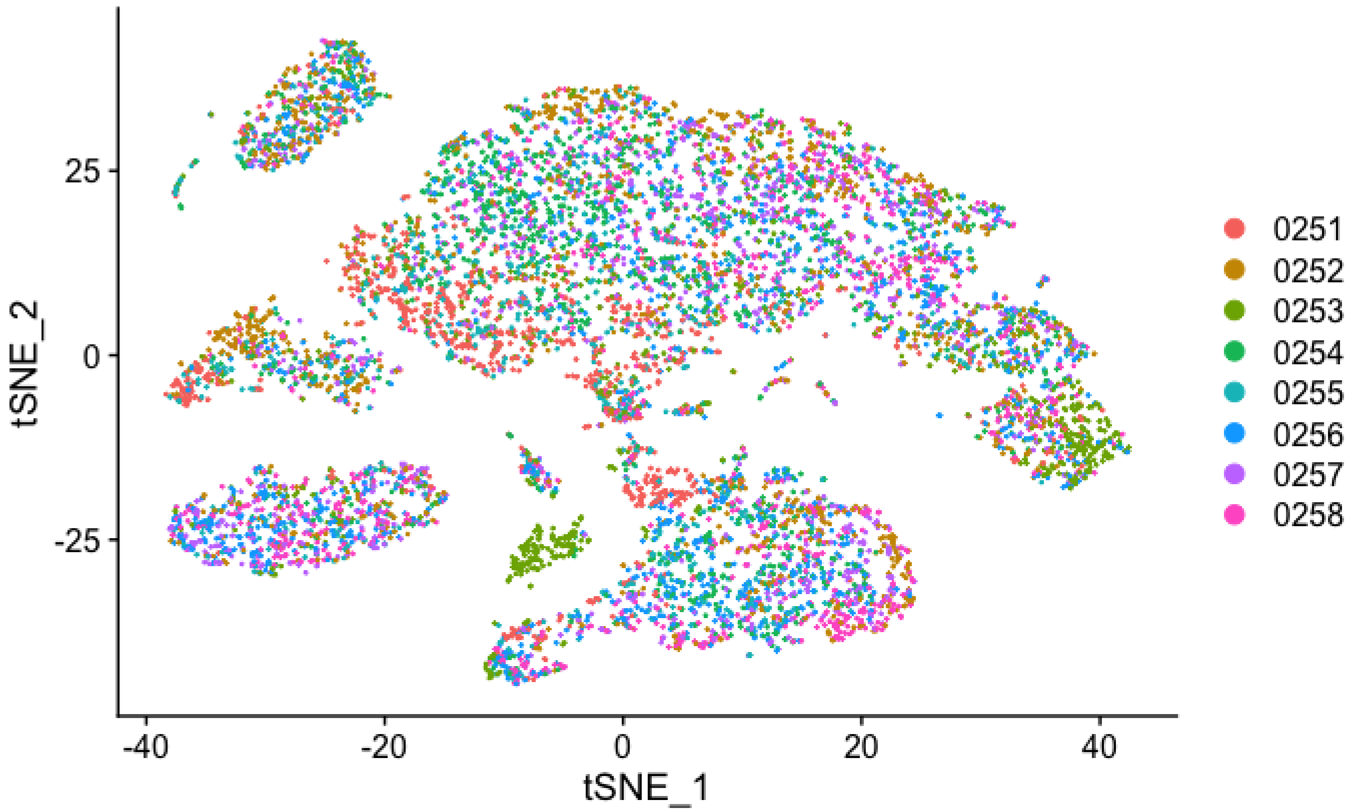
Hide

```
# We select the top 10 PCs for clustering and tSNE based on PCElbowPlot
ex.singlets <- FindNeighbors(ex.singlets, reduction = "pca", dims = 1:10)
```

```
Computing nearest neighbor graph
Computing SNN
```

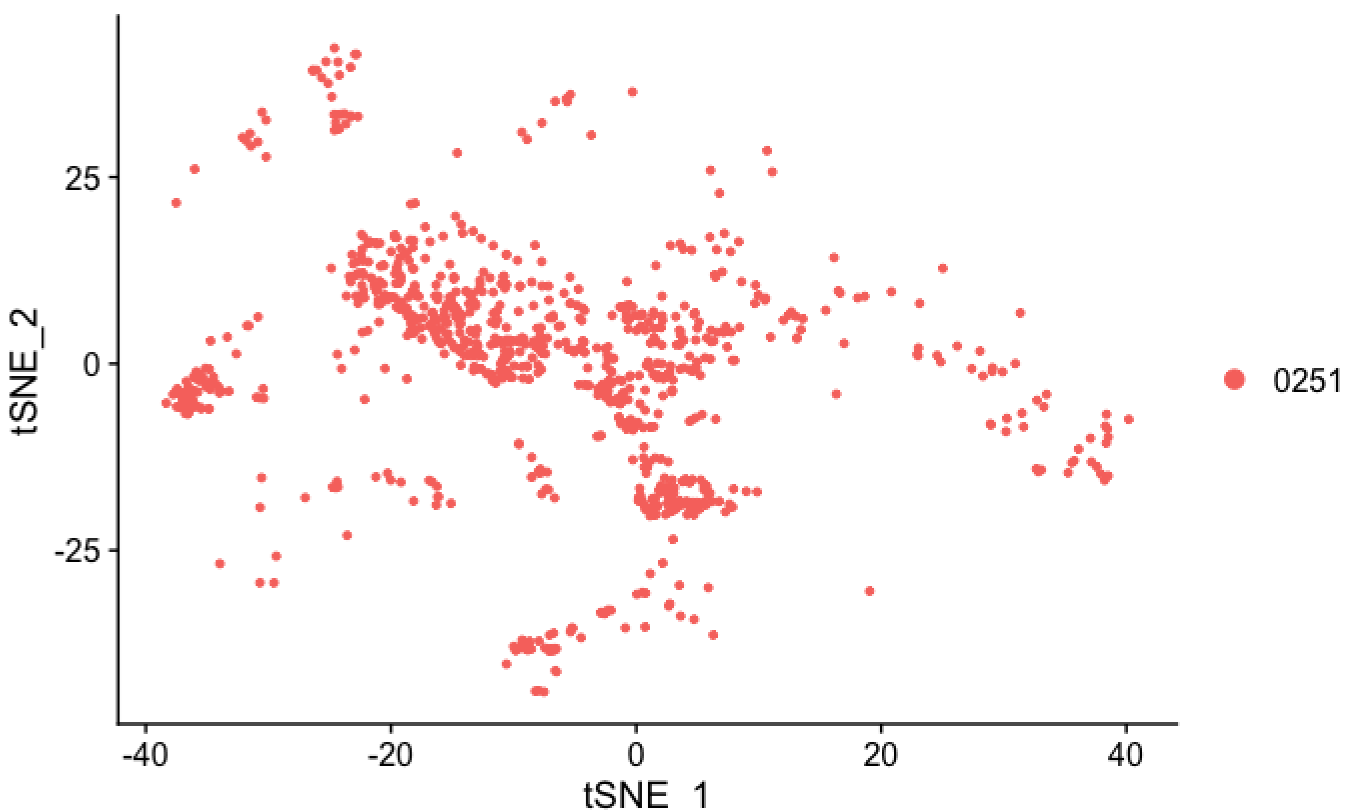
Hide

```
ex.singlets <- FindClusters(ex.singlets, resolution = 0.6, verbose = FALSE)
ex.singlets <- RunTSNE(ex.singlets, reduction = "pca", dims = 1:10)
# Projecting singlet identities on TSNE visualization
DimPlot(ex.singlets, group.by = "HTO_classification")
```



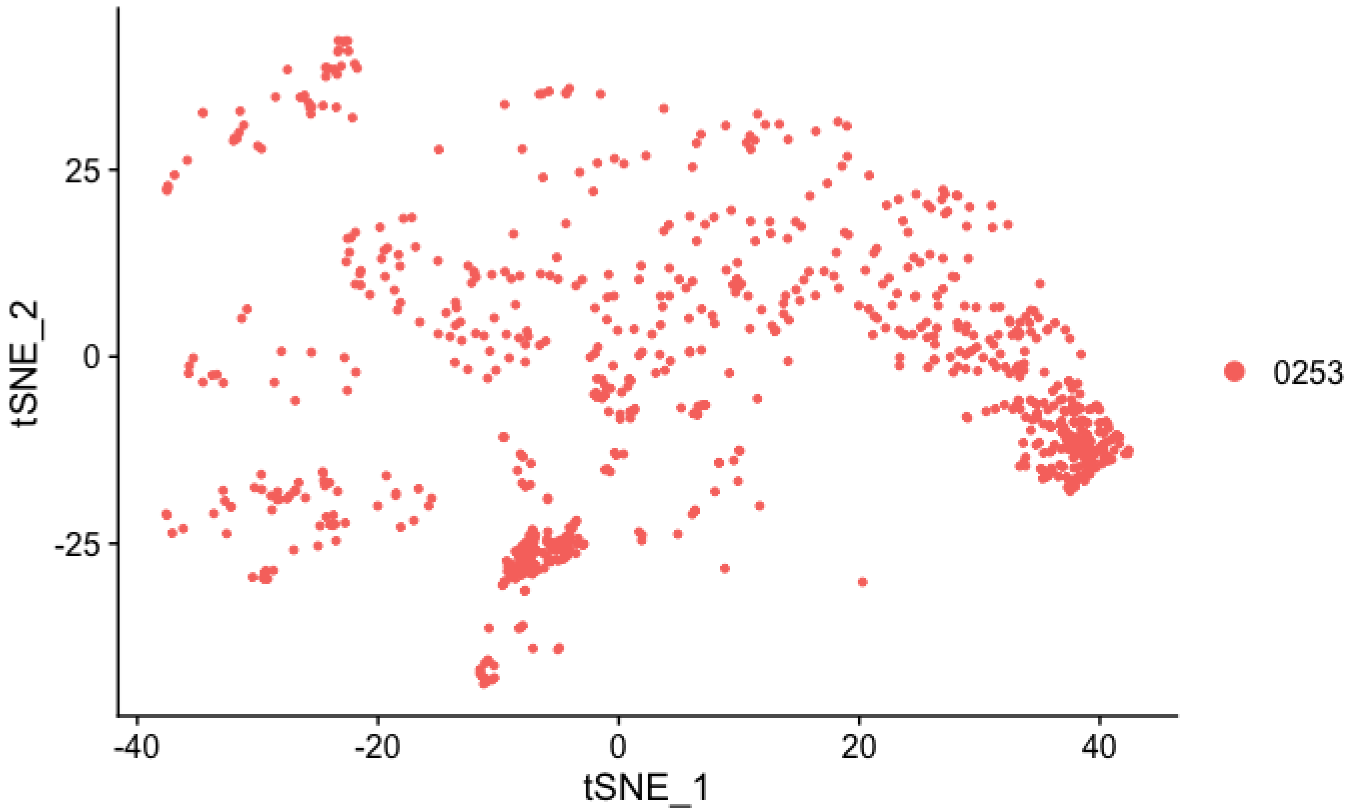

Hide

```
# Projecting singlets for each hash ID separately
DimPlot(ex.singlets[, ex.singlets$hash.ID == "0251"], group.by = "HTO_classification")
```



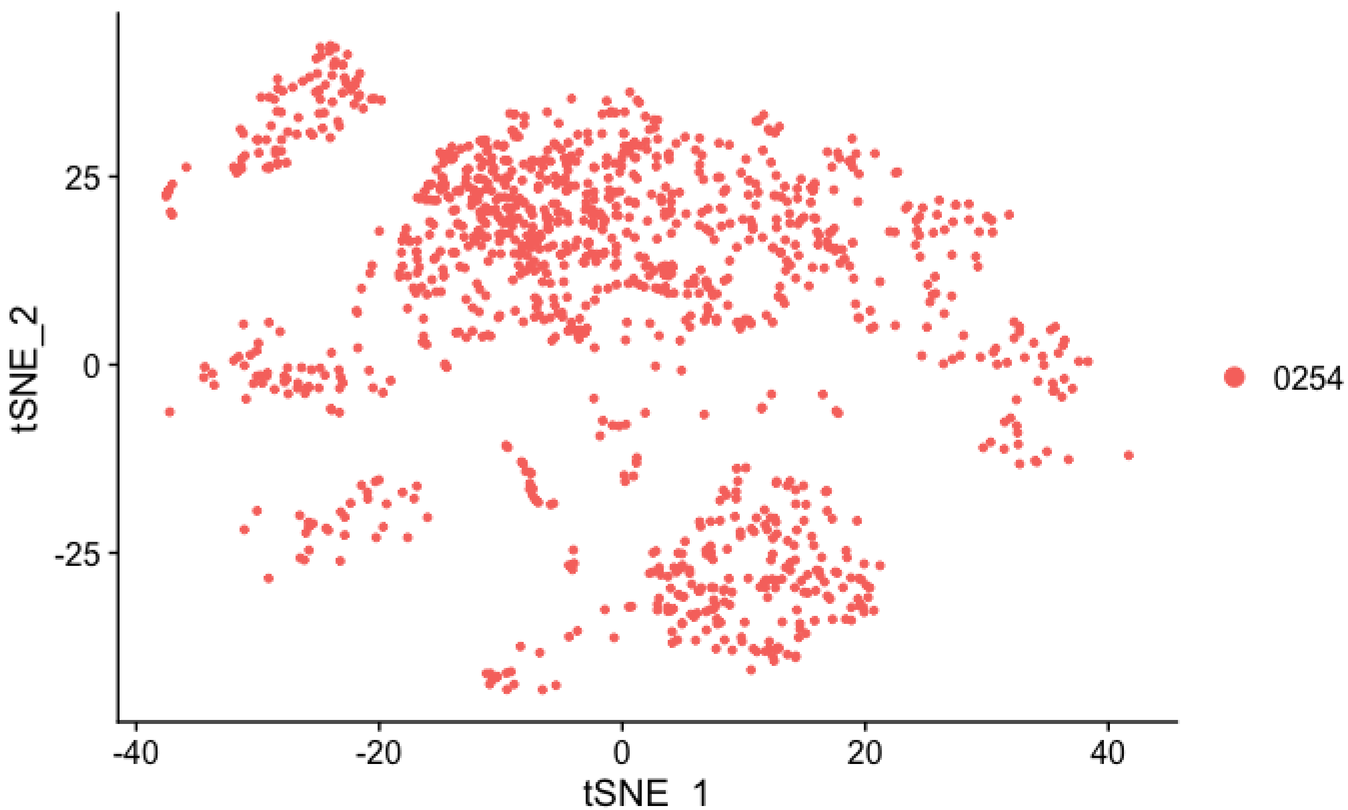

Hide

```
DimPlot(ex.singlets[, ex.singlets$hash.ID == "0253"], group.by = "HTO_classification")
```



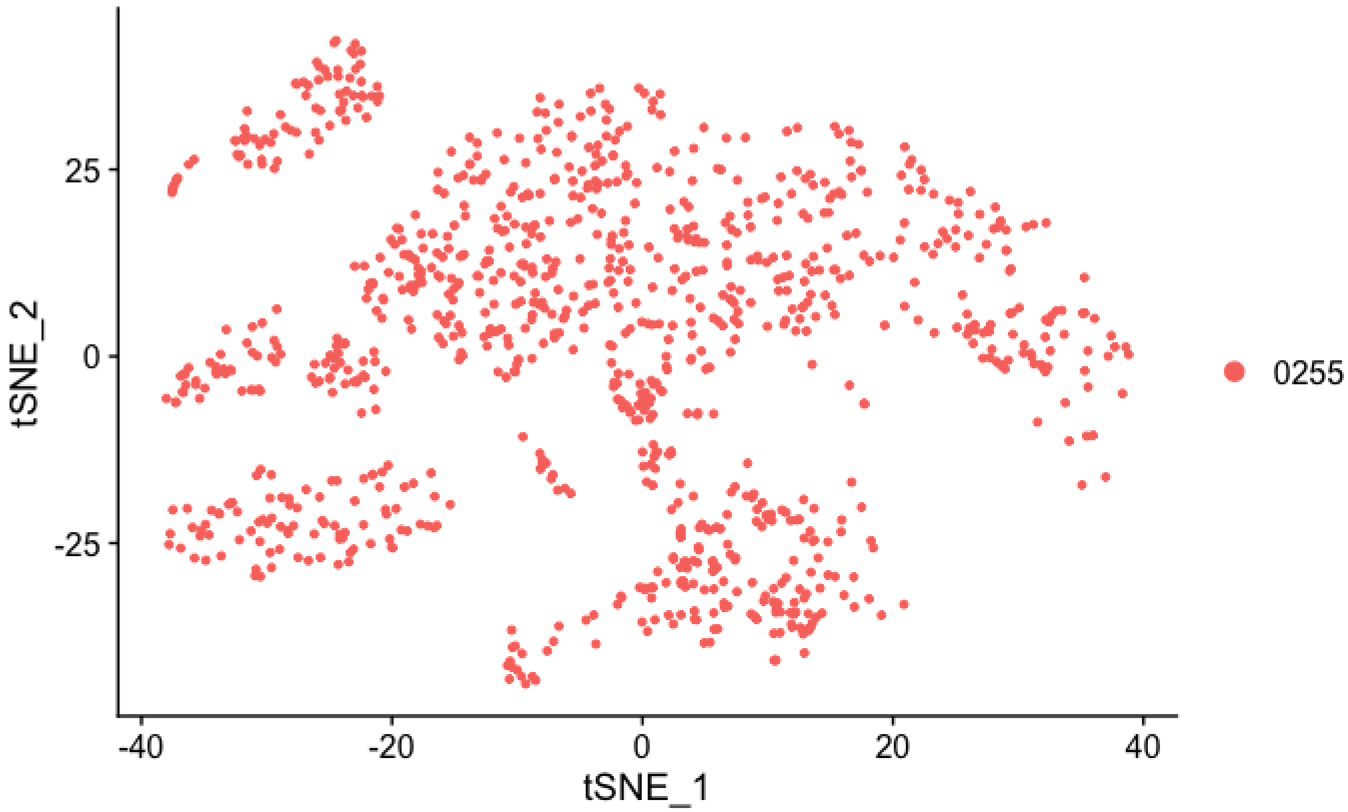

Hide

```
DimPlot(ex.singlets[, ex.singlets$hash.ID == "0254"], group.by = "HTO_classification")
```



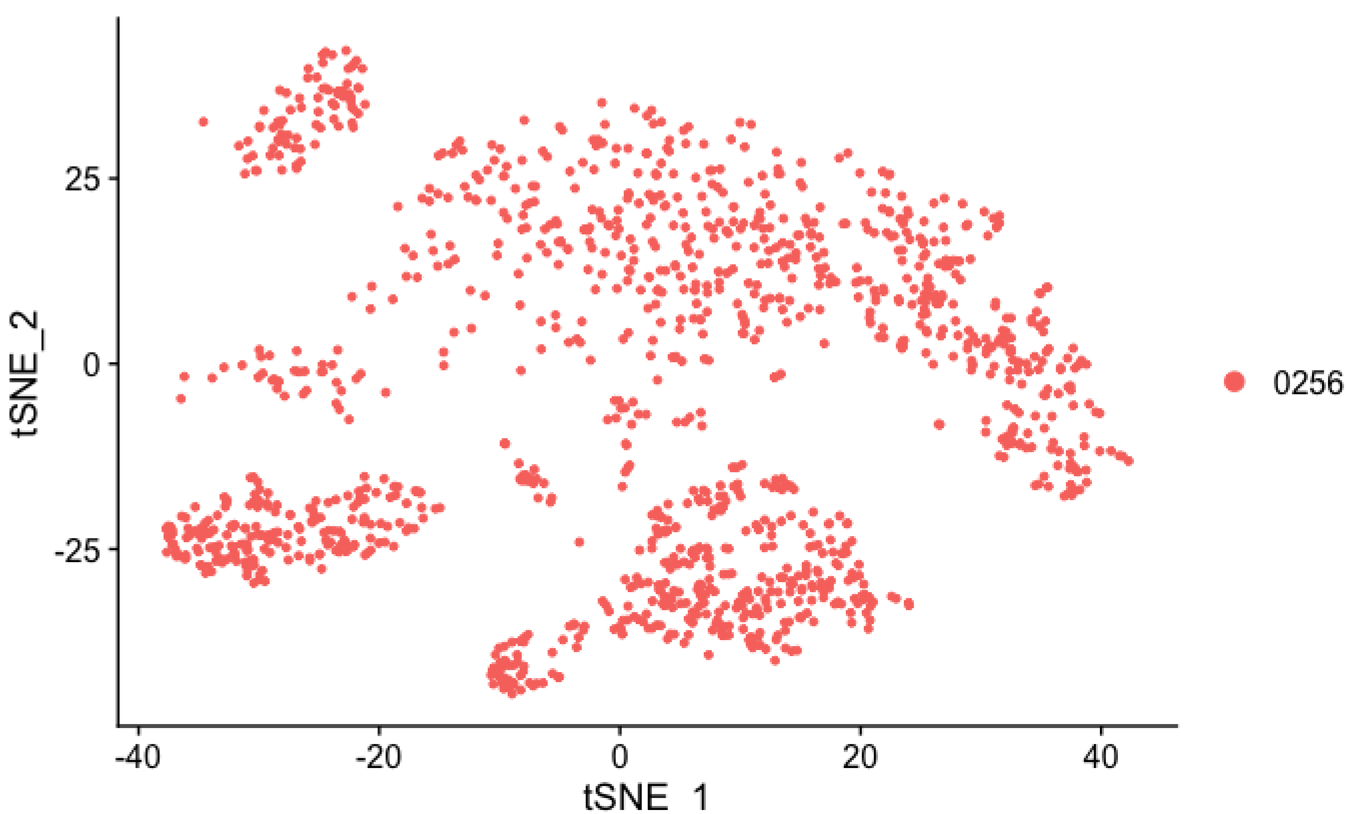

Hide

```
DimPlot(ex.singlets[, ex.singlets$hash.ID == "0255"], group.by = "HTO_classification")
```



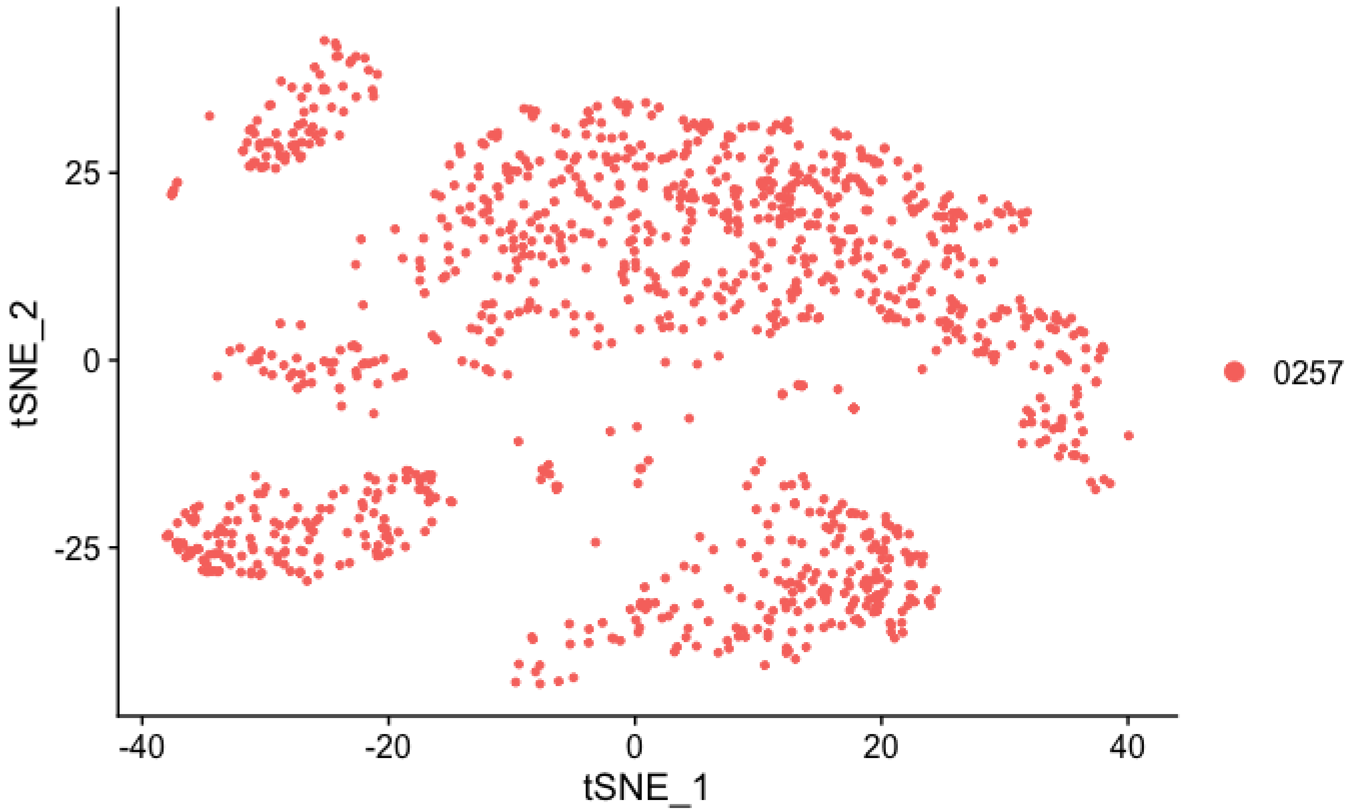

Hide

```
DimPlot(ex.singlets[, ex.singlets$hash.ID == "0256"], group.by = "HTO_classification")
```



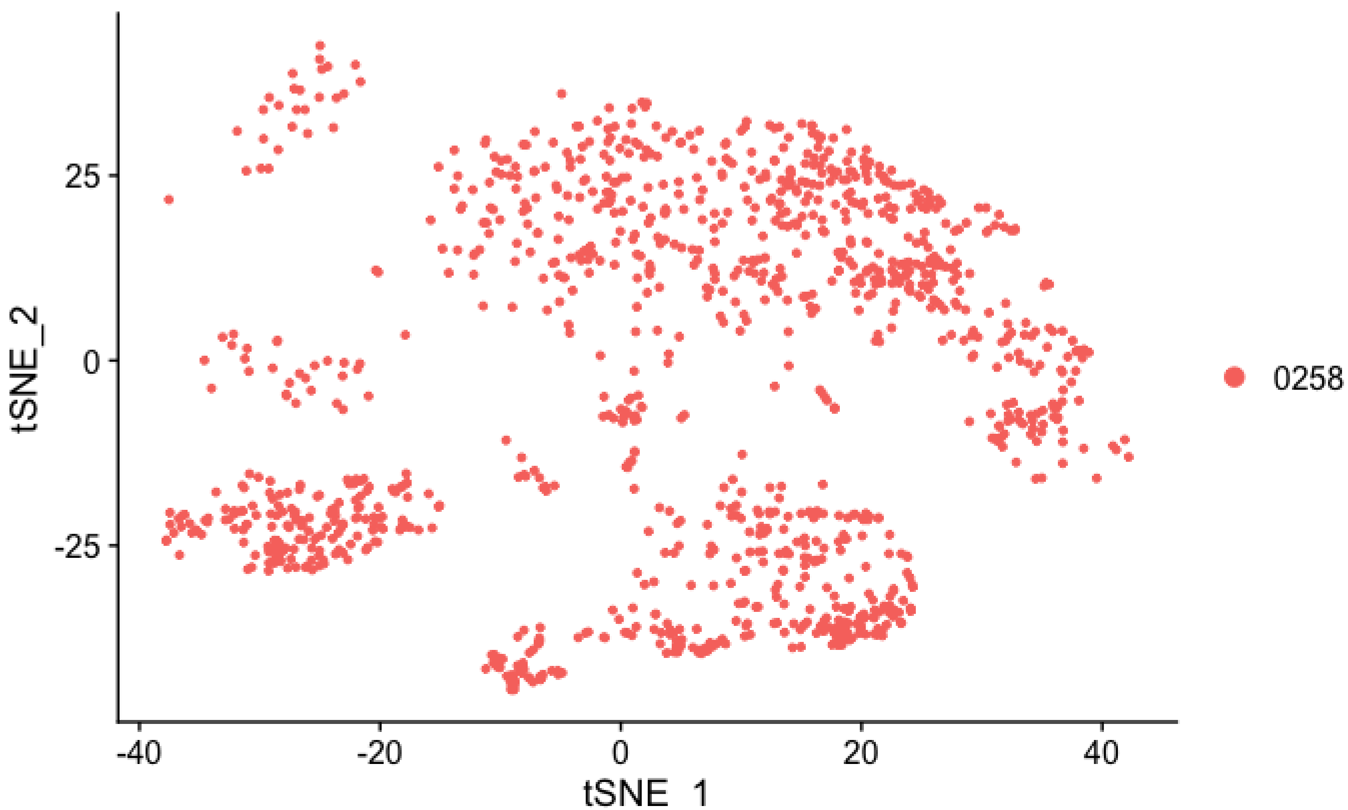

Hide

```
DimPlot(ex.singlets[, ex.singlets$hash.ID == "0257"], group.by = "HTO_classification")
```




Hide

```
DimPlot(ex.singlets[, ex.singlets$hash.ID == "0258"], group.by = "HTO_classification")
```




Hide

```
# Visualize HTOs on RNA clusters
FeaturePlot(ex.singlets, features = rownames(ex.hashtag[["HTO"]]), ncol = 3)
```

```
Could not find 0251 in the default search locations, found in HTO assay insteadCould not find 0252
in the default search locations, found in HTO assay insteadCould not find 0253 in the default sear
ch locations, found in HTO assay insteadCould not find 0254 in the default search locations, found
in HTO assay insteadCould not find 0255 in the default search locations, found in HTO assay instea
dCould not find 0256 in the default search locations, found in HTO assay insteadCould not find 025
7 in the default search locations, found in HTO assay insteadCould not find 0258 in the default se
arch locations, found in HTO assay instead
```