SpaTME

Xiaojie Cheng

2024-07-29

Preprocessing

```
library(SpaTME)
library(dplyr)
library(ggplot2)
# load ST data
st <- readRDS("~/test_data/st.rds")
```

SpaTME checks if the coordinate information is available in the meta.data, images, or colnames (formatted as 1x2) of st. Specifically, for Visium data, the x and y coordinates are adjusted to follow a hexagonal arrangement.

```
st <- STcoordCheck(st = st, platform = "Visium", hexagon.correct = T,
                   hexagon.direct = "vertical",verbose = T)
```

```
## Get coordinate information from @image
```

```
## Converting coordinate position to hexagon arrangement!
```

```
## [1] "Completing coordinate check!"
```

Preprocessing includes standard steps in Seurat tool, like SCT normalization, dimensionality reduction (PCA, UMAP, t-SNE), and clustering.

```
st <- SePreprocess(se = st, assay = "Spatial", norm.SCT = TRUE, cluster.resolution =
0.8)
```

Characterization of TME with cellular and molecular features.

Creating an `STFeature` object, which serves as a container for identified ST features. The STFeature object is initialized with positional data and cell abundance annotations.

```
# Loading cell abundances deconvoluted using Cell2location.
cell.abun <- readRDS("~/test_data/cell.abun.rds")
stf <- CreateStfObj(st = st, assay = "SCT", slot = "data", norm = T,
                    cell.anno = cell.abun, min.prop = 0.05,
                    init.fea = c("Position", "CellAnno"))
```

Molecular features

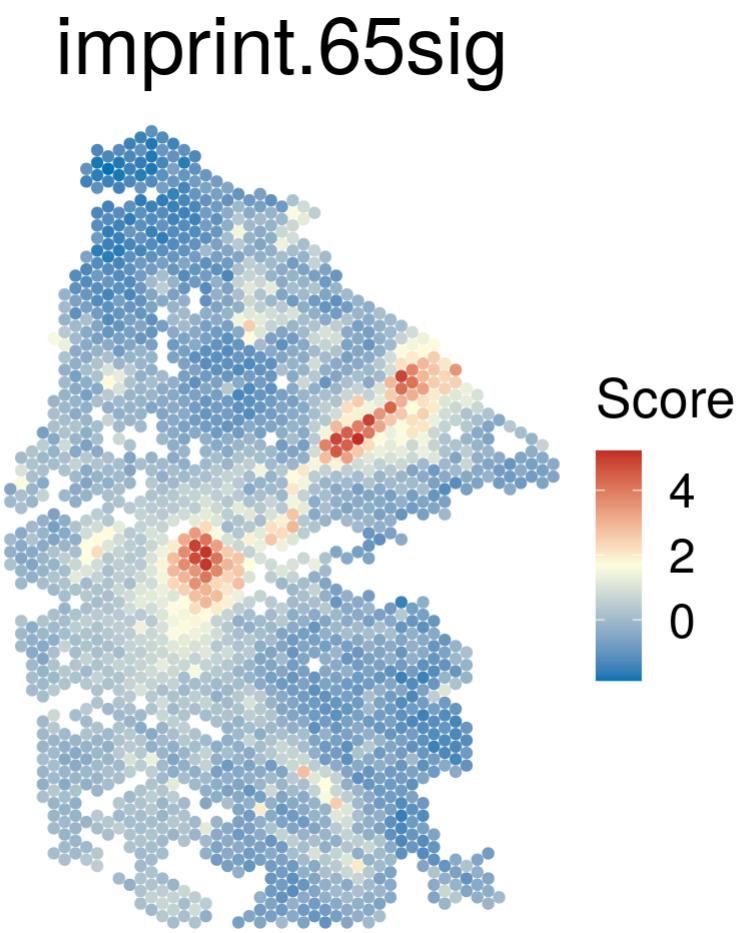
SpaTME explores the molecular features from three aspects: (1) Assessing the enrichment of annotated gene sets. (2) Determining gene co-expression modules. (3) Quantifying the ligand-receptor (LR) interactions.

Firstly, SpaTME enables assessing the enrichment of immune and tumor-related signatures collected from literatures (CuratedSig category) or various biomedical ontologies sourced from databases (MsigDB category). Here, we calculate the enrichment scores of TLS signatures using the `AddModuleScore` function in Seurat.

```
expr <- GetAssayData(object = st, assay = "SCT", slot = "data")
stf = GetGsetSigScore(expr = expr, stf = stf, category = "CuratedSig",
                      types = c("Immune"),
                      subtype = "TLS",
                      method = "AddModuleScore", scale = T)
```

```
## Calculating Immune-related signatures
```

```
# In SpaTME, the SpotVisualize() function is used to visualize features.
SpotVisualize(st, title = "imprint.65sig",
              legend.name = "Score",size = 1.5,
              meta = stf@GsetSig$CuratedSig$Immune$TLS$imprint.65sig)
```



SpaTME allows identification of gene co-expression modules utilizing `WGCNA` . Here, we use the top 2000 Spatially variable genes (SVGs) identified previously as an example.

```
topSVGs <- readRDS("~/test_data/topSVGs.rds")
data <- st@assays$SCT@data[topSVGs,]
# gene co-expression module
Topgene.net <- BulidCGnet(data = t(as.matrix(data)), outdir = NULL,
                      power.seq = c(c(1:10), seq(from = 12, to=20, by=2)),
                      verbose = 0, minModuleSize = 20, reassignThreshold = 0,
                      mergeCutHeight = 0.15, pamRespectsDendro = FALSE, detectCutHeigh
t = 0.99)
```

```
##
```

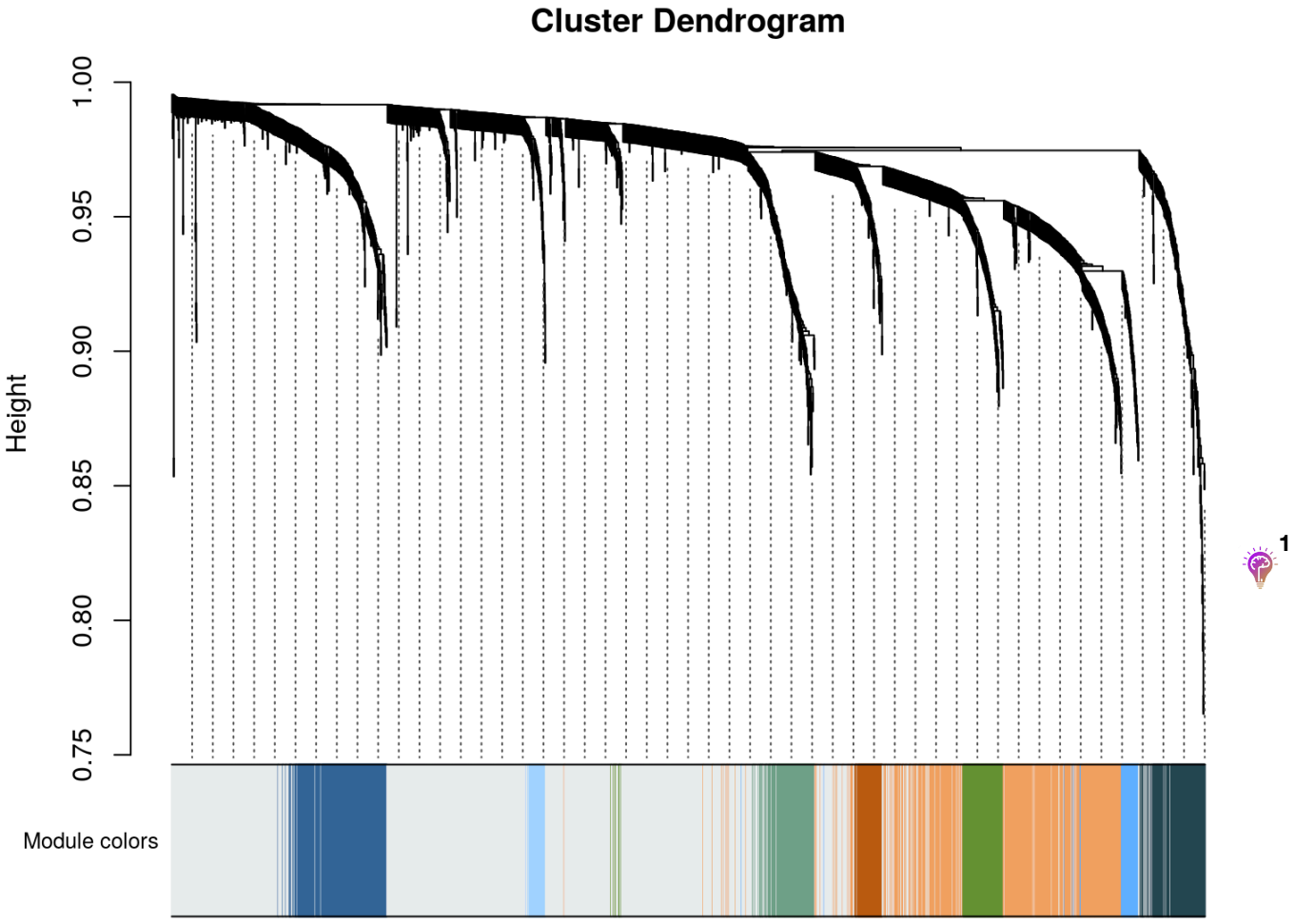
```
## The output directory is recommended to save the result files!
```

```
## Selecting soft threshold...
```


```
##    Power SFT.R.sq slope truncated.R.sq  mean.k. median.k.  max.k.
## 1      1    0.461 -2.03         0.9450 1.33e+02  1.20e+02 370.000
## 2      2    0.871 -2.55         0.9630 1.59e+01  1.12e+01 102.000
## 3      3    0.377 -3.42         0.3350 2.83e+00  1.36e+00  34.800
## 4      4    0.374 -2.94         0.3490 6.96e-01  1.98e-01  13.800
## 5      5    0.353 -2.52         0.3190 2.21e-01  3.26e-02   6.060
## 6      6    0.247 -1.94         0.0330 8.62e-02  6.06e-03   3.090
## 7      7    0.278 -1.87         0.0721 3.94e-02  1.22e-03   2.270
## 8      8    0.380 -2.03         0.2980 2.05e-02  2.65e-04   1.770
## 9      9    0.381 -1.88         0.2870 1.18e-02  6.10e-05   1.440
## 10    10    0.389 -1.76         0.2970 7.41e-03  1.34e-05   1.190
## 11    12    0.326 -1.90         0.1610 3.49e-03  7.80e-07   0.842
## 12    14    0.361 -2.26         0.2010 1.93e-03  4.68e-08   0.612
## 13    16    0.314 -1.99         0.1180 1.18e-03  3.01e-09   0.452
## 14    18    0.320 -1.92         0.1260 7.67e-04  1.96e-10   0.337
## 15    20    0.328 -1.84         0.1370 5.19e-04  1.35e-11   0.253
```

```
## soft-thresholding power is selected: 2
```

```
## Starting module detection...
```



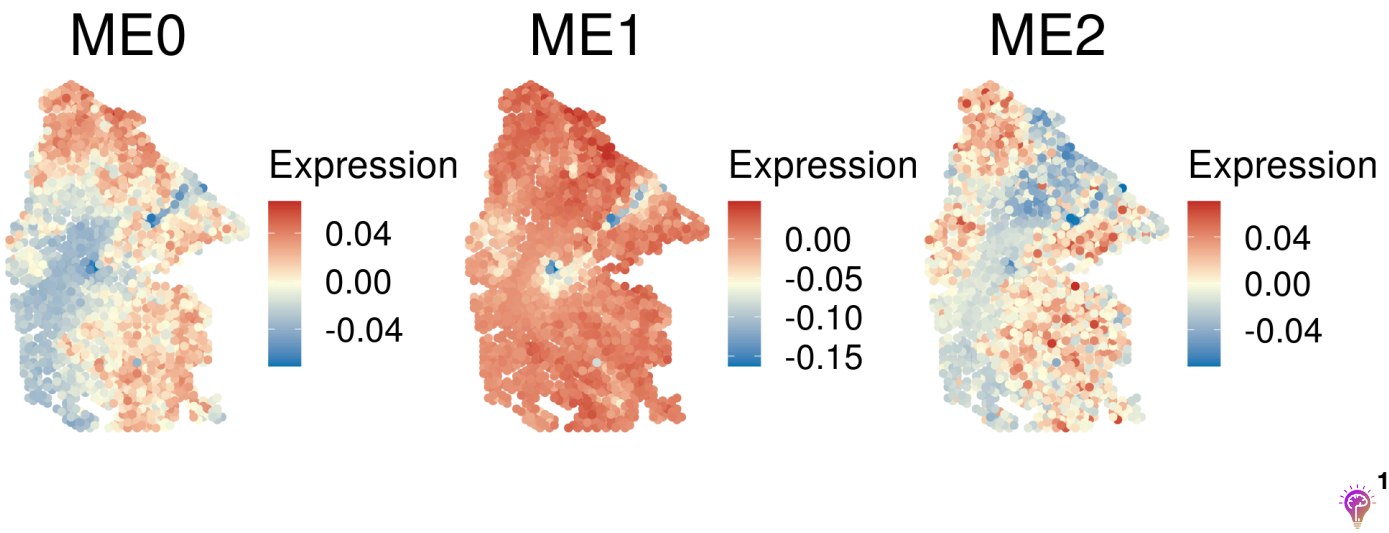

```
## $mar
## [1] 1 5 0 1
```

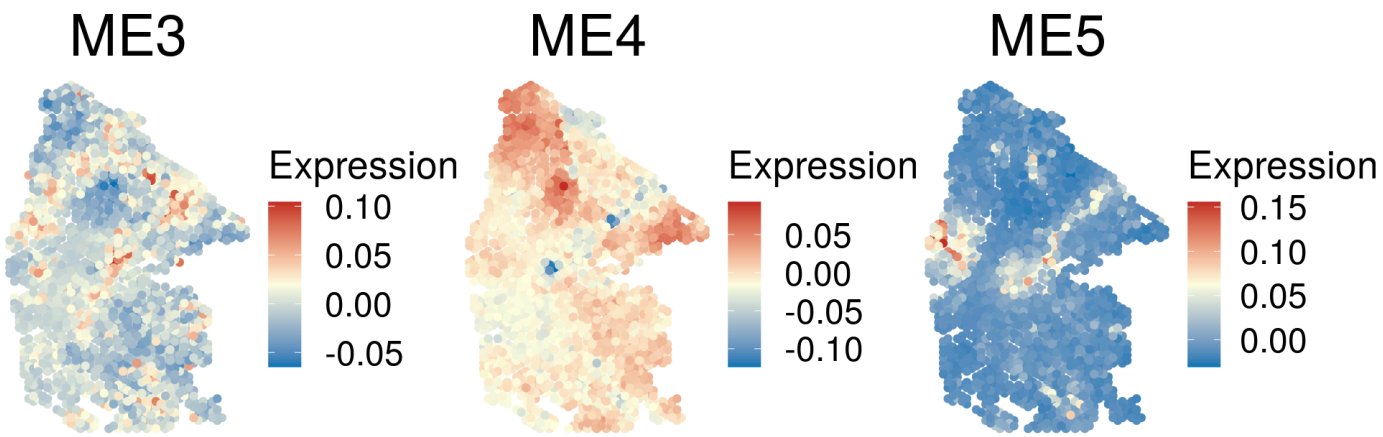
Calculating the eigengenes of modules.

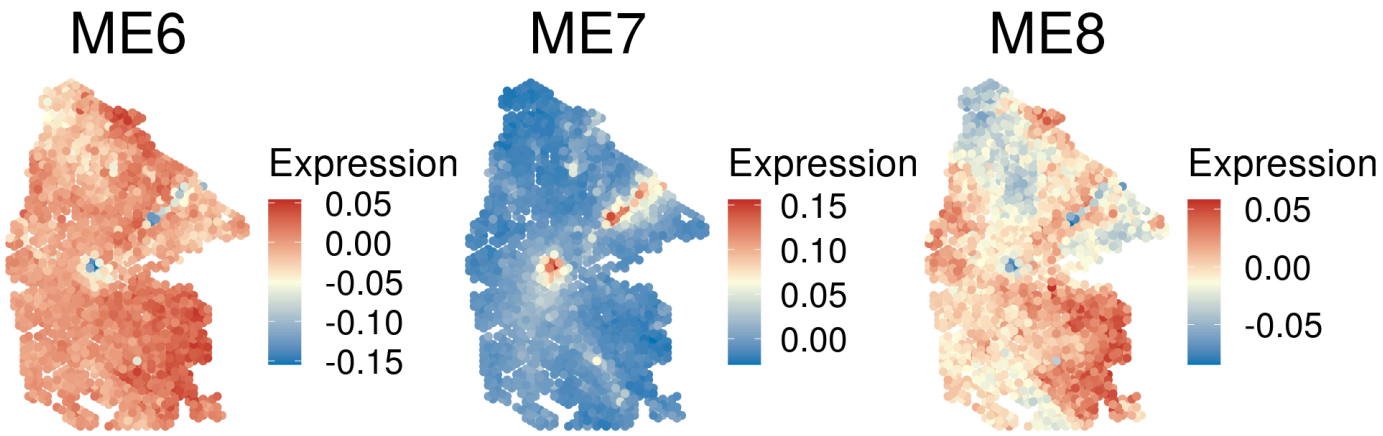
```
net_MEs <- WGCNA::moduleEigengenes(expr = t(as.matrix(data)),
                           colors = Topgene.net$colors)
net_MEs <- net_MEs$eigengenes
```

Visualizing the distribution of each ME.

```
library(cowplot)
plot.list = c()
for(i in colnames(net_MEs)){
  plot.list[[i]] <- SpotVisualize(st, meta = net_MEs[, i], size = 1.5, return = T,tit
le = i, legend.name = "Expression")
}
plot_grid(plotlist = plot.list, ncol = 3)
```


ME0
Expression
0.04
0.00
-0.04
ME1
Expression
0.00
-0.05
-0.10
-0.15
ME2
Expression
0.04
0.00
-0.04


ME3
Expression
0.10
0.05
0.00
-0.05
ME4
Expression
0.05
0.00
-0.05
-0.10
ME5
Expression
0.15
0.10
0.05
0.00


ME6
Expression
0.05
0.00
-0.05
-0.10
-0.15
ME7
Expression
0.15
0.10
0.05
0.00
ME8
Expression
0.05
0.00
-0.05

SpaTME enables quantification of 1396 LR interactions collected from the `CellPhoneDB` database, which includes multi-subunit information of protein complexes. The `BuildLRExprssion()` function recalculates the effect expression of secreted proteins and membrane proteins separately, taking into account the interaction distance.

```
data("LR_input")
expr <- GetAssayData(object = st,
                     assay = "SCT",
                     slot = "data")
LR_expression <- BuildLRExprssion(Expr = expr, st_pos = stf@Position,
                                  LR_input = LR_input, do.dist = T, r.dist = 4,
                                  long.dist.method = "weighted",
                                  adjust = 2, na.rm = T, verbose = F)
```

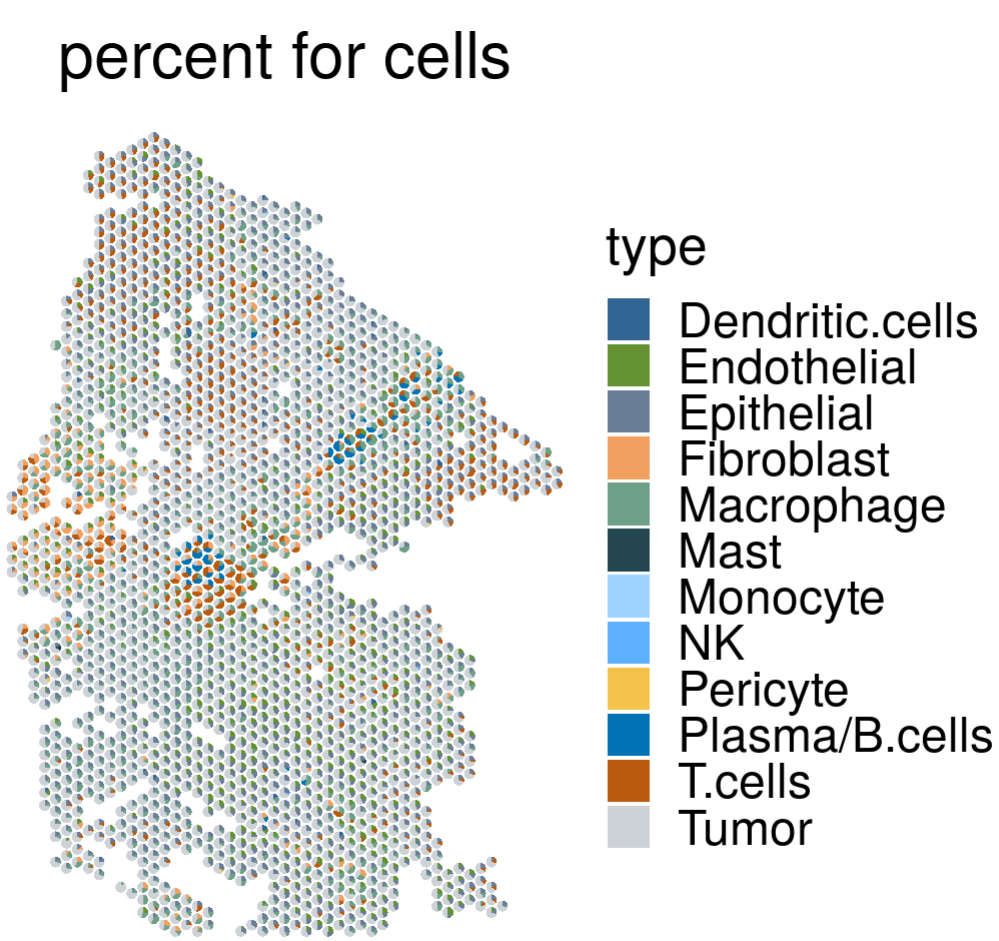
The `CalLRIScore` function quantifies the interaction scores of LR pairs.

```
data(interaction_input)
LRscore <- CalLRIScore(interaction_input = interaction_input[1:100,],
                       LR_expression = LR_expression, na.rm = T,
                       p = 3, verbose = F)
```

Cellular features

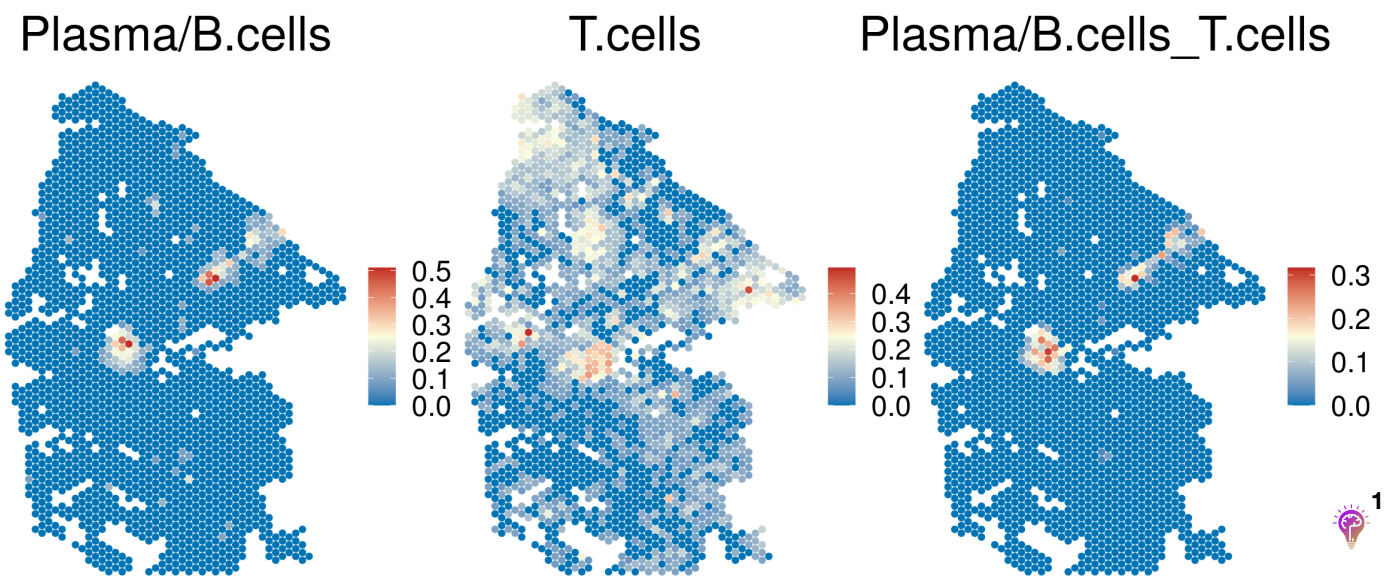
SpaTME describes the spatial distribution of cell populations by calculating co-distribution scores of cell types and measuring the degree of immune infiltration.

```
# Visualizaition of top three cell types for each spot
library("scatterpie")
library("cowplot")
data <- stf@CellAnno
data <- apply(data, 1, function(x){
  th <- sort(x, decreasing = T)[3]
  x[x < th] = 0
  return(x/sum(x))
})
data <- t(data)
colors <- LabelMapcolor(labels = unique(colnames(data)),
                        assgin.col = c("Tumor" = "#ced4da", "Plasma/B.cells" = "#0077
b6", "T.cells" = "#b95b13", "Epithelial" = "#6c809a"))
PlotCellSpot(decon_mtrx = data, st_pos = stf@Position,
             pie_scale = 0.5, pie_color = colors,
             separate = F, e.color = NA)
```



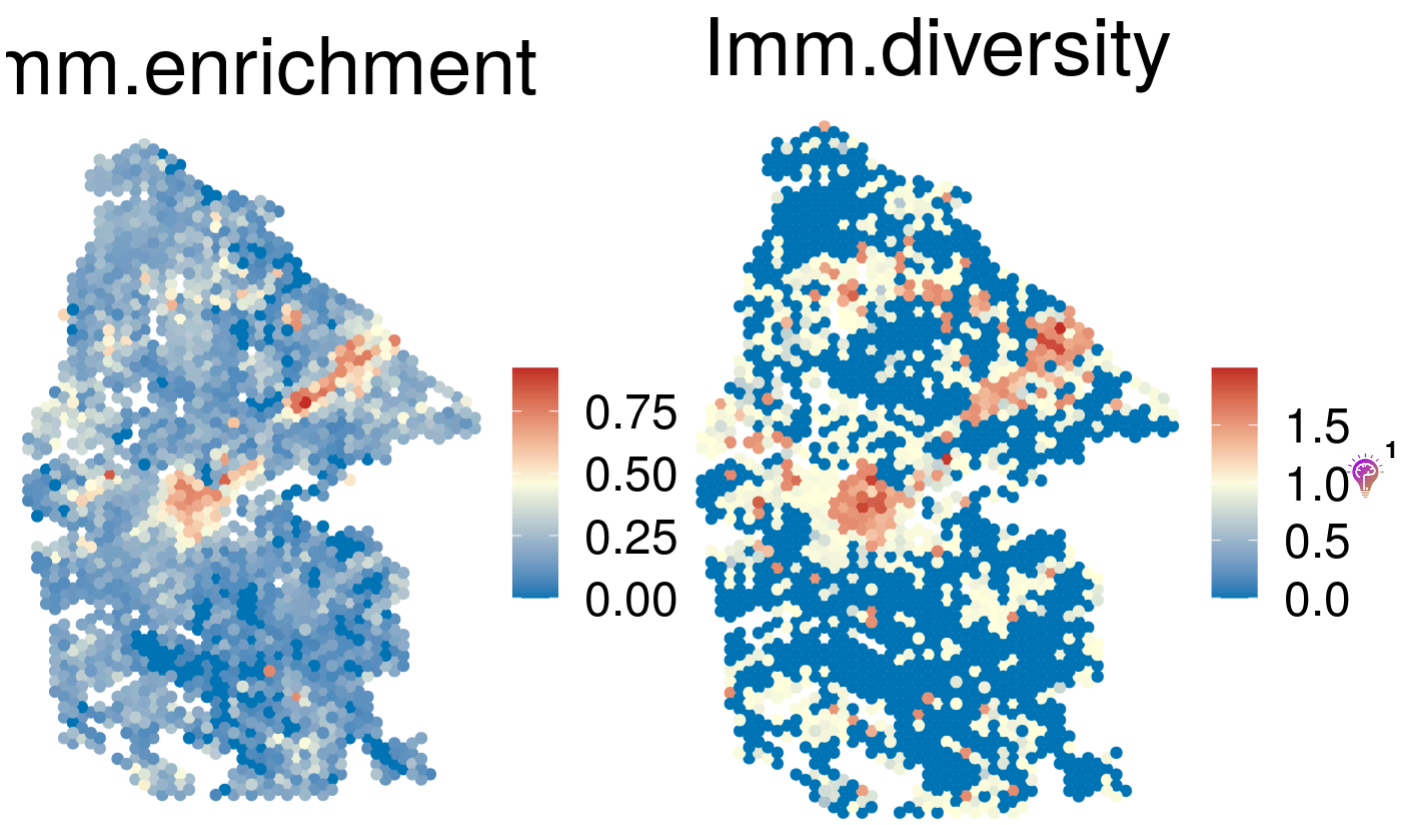
SpaTME calculates the cell co-distribution scores with the `CalCellCodis()` function.

```
stf@CellCodis <- CalCellCodis(stf@CellAnno, sort = T)
# Distribution of T cells and Plasma/B.cells, and their co-diatribution scores.
p1 <- SpotVisualize(pos = stf@Position, meta = stf@CellAnno$`Plasma/B.cells`,
              size = 1.5, return = T, title = "Plasma/B.cells")
p2 <- SpotVisualize(pos = stf@Position, meta = stf@CellAnno$T.cells,
              size = 1.5, return = T, title = "T.cells")
p3 <- SpotVisualize(pos = stf@Position,
              meta = stf@CellCodis$`Plasma/B.cells_T.cells`,
              size = 1.5, return = T, title = "Plasma/B.cells_T.cells")
plot_grid(plotlist = list(p1, p2, p3), ncol = 3)
```



The degree of immune infiltration in ST sample.

```
immune.cells <- sort(c("Plasma/B.cells", "T.cells", "Dendritic.cells", "Macrophage",
"Monocyte", "NK"))
data <- stf@CellAnno[, immune.cells]
## Immune cells with a proportion less than 0.05 will not be considered for immune in
filtration
Imm.infil <- CalImmInfiltration(data, min.prop = 0.05)
p1 <- SpotVisualize(pos = stf@Position, meta = Imm.infil$Imm.enrichment, return = T,
               size = 1.5, title = "Imm.enrichment")
p2 <- SpotVisualize(pos = stf@Position, meta = Imm.infil$Imm.diversity, return = T,
              size = 1.5, title = "Imm.diversity")
plot_grid(plotlist = list(p1, p2), ncol = 2)
```



Recognition of spatial architectures

Domain annotation

Any domain labels annotated with others tools can be used for SpaTME. In addition to clusters obtained from `Seurat` , SpaTME also implements the `BayesSpace` tool to annotate the spatial domains of ST data. If the number of clusters (`num_clu`) is not specified, the `BayesCluster()` function will automatically select it by running the BayesSpace algorithm for multiple values specified by the `qs` parameter. Alternatively, users could select the point around the elbow of plot if the `show_qplot` is set to TRUE. For example, we run the `BayesCluster()` with `qs = 5:12` and select the point 9, which around the first elbow.

```
sce <- BayesCluster(st = st, assay = "Spatial",platfrom = "Visium",
                    qs = 6:12, show_qplot = T)
```

```
## [1] "Starting clustering with BayesSpace"
```

```
## Number of cluster is not provided, qTune will used for selecting best q value
```

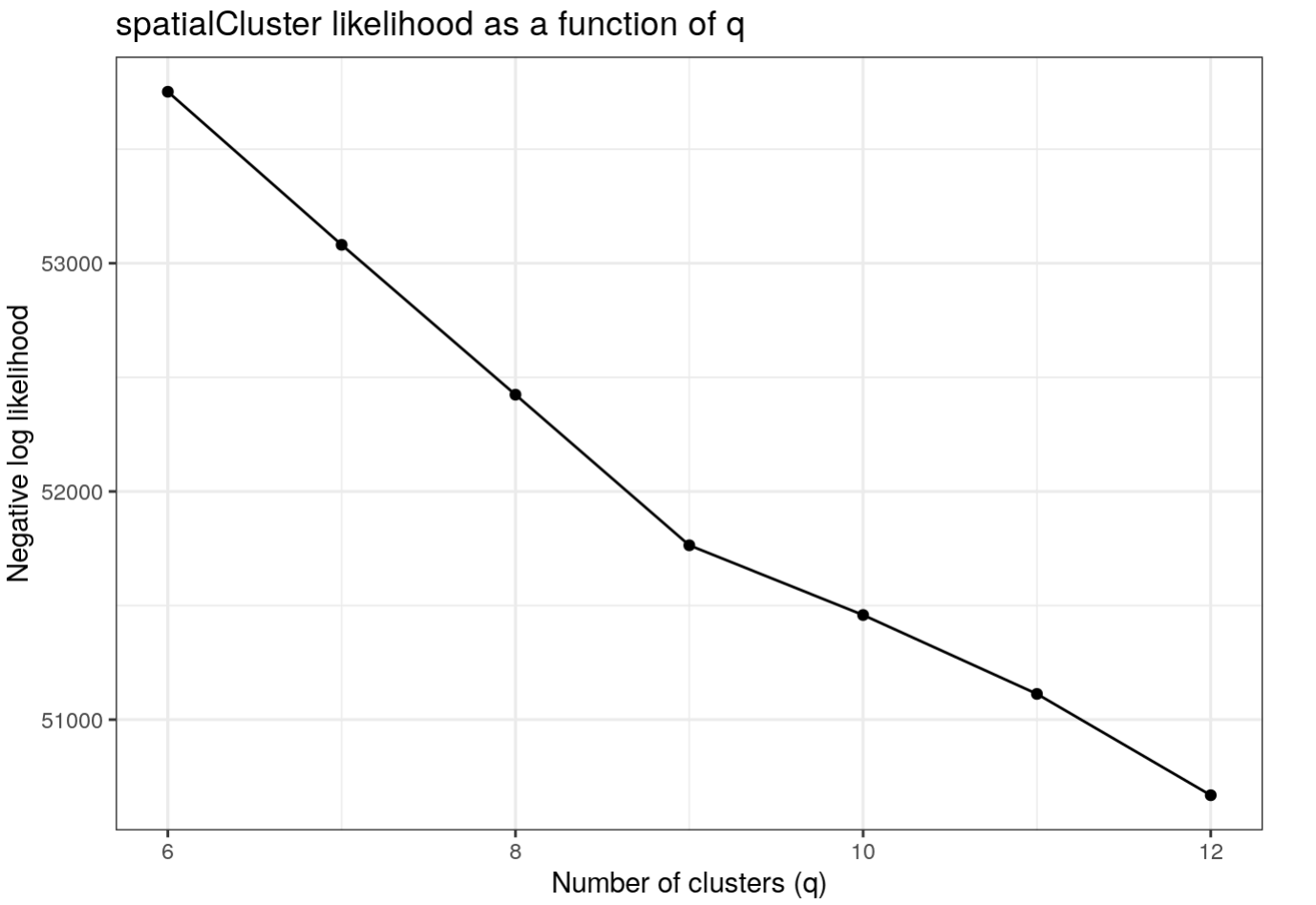
```
## Neighbors were identified for 1921 out of 1946 spots.
```

```
## Fitting model...
## Fitting model...
## Fitting model...
## Fitting model...
## Fitting model...
## Fitting model...
## Fitting model...
```

```
## [1] "The recommended cluster number is 9"
```

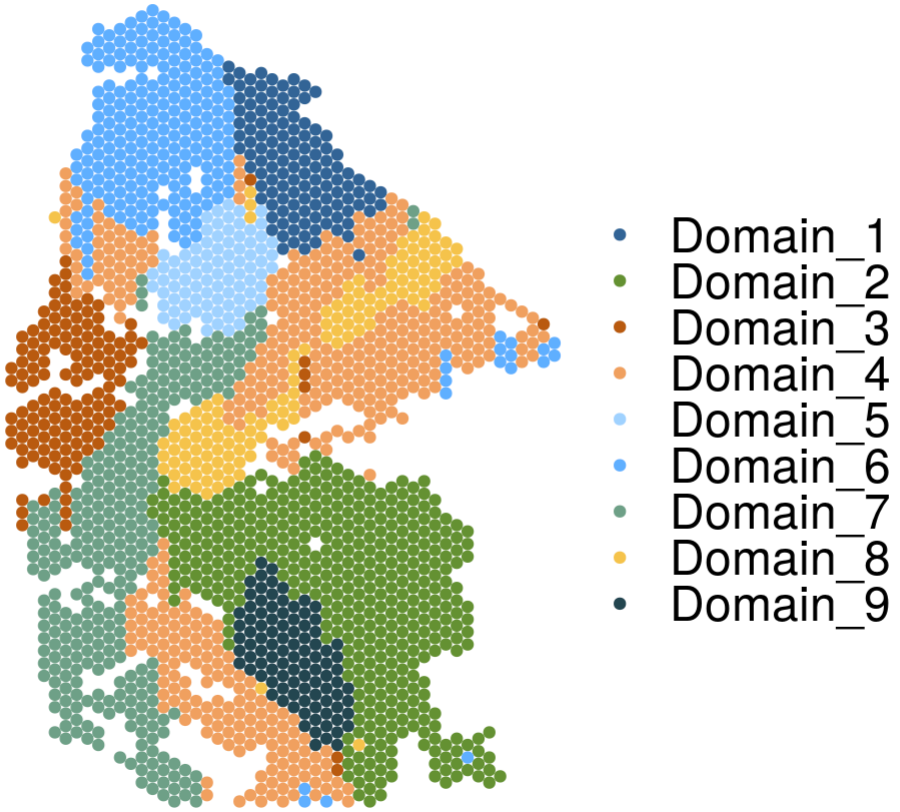
```
## Neighbors were identified for 1921 out of 1946 spots.
## Fitting model...
```

```
## Calculating labels using iterations 1000 through 50000.
```




```
domains <- paste0("Domain_", sce@colData@listData$spatial.cluster)
stf@Annotation <- data.frame(row.names = rownames(stf@Position))
stf@Annotation[, "bayes_cluster"] <- domains
SpotVisualize(pos = stf@Position, size = 1.5,
                  meta = domains, title = "Domain annotation",
                  cha.col = LabelMapcolor(unique(domains)))
```

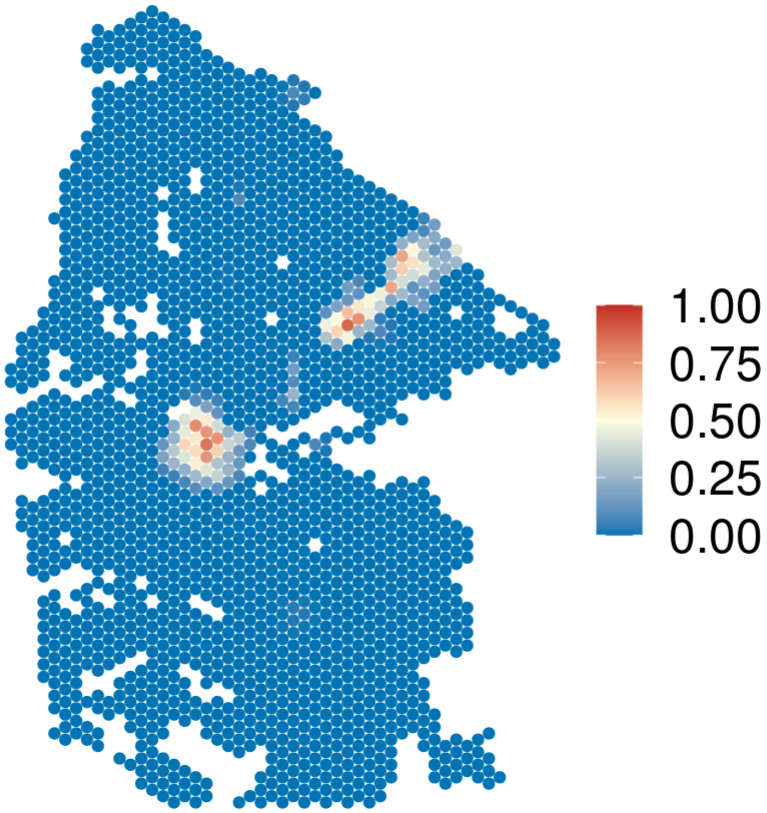
Domain annotation



TLS prediction

SpaTME employs two TLS signatures, as well as the co-localization of Plasma/B cells and T cells, to detect TLS. In particular, SpaTME improves the accuracy of TLS prediction by integrating the neighborhood information of spots within the same domain. The `CalTLSfea` function returns a list containing distinct forms of TLS features and the final TLS scores.

```
tls.sig <- stf@GsetSig$CuratedSig$Immune$TLS[, c("LC.50sig", "imprint.65sig")]
cell.fea <- stf@CellCodis$`Plasma/B.cells_T.cells`
comb.fea <- cbind(tls.sig, cell.fea)
rownames(comb.fea) <- rownames(stf@Position)
TLS.fea <- CalTLSfea(data = comb.fea, st_pos = stf@Position,
                     cluster = stf@Annotation$bayes_cluster,
                     r.dist = 2, method = "weighted", verbose = F)
SpotVisualize(pos = stf@Position, size = 1.5, limits = c(0, 1),
                  meta = TLS.fea$TLS.score, title = "TLS score")
```



Tumor-normal interface (TNI) identification

SpaTME identifies the TNI regions by inputted with tumor cell and domain annotations. Here, we used a RCC slice with clear tumor boundaries.

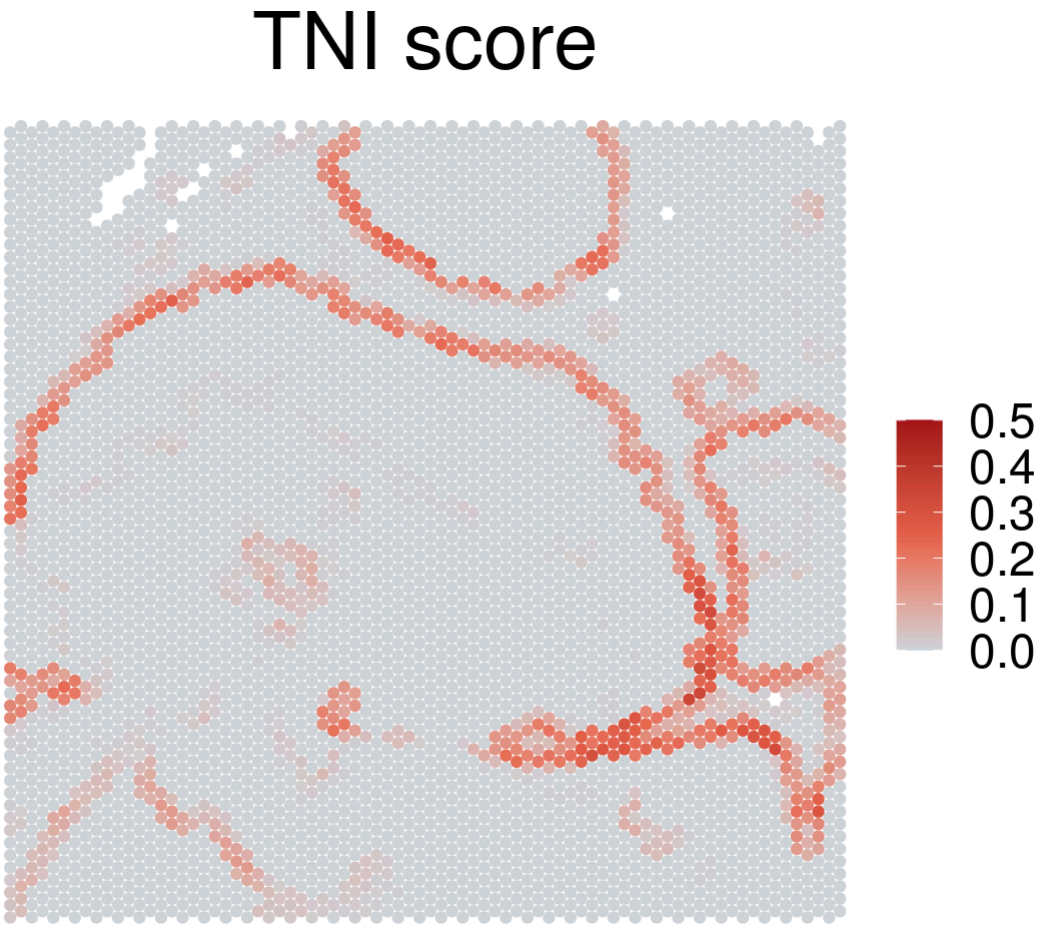
```
st2 <- readRDS("~/test_data/st2.rds")
st2 <- STcoordCheck(st = st2, platform = "Visium", hexagon.correct = T, reset = T,
                    hexagon.direct = "vertical",verbose = T)
```

```
## [1] "Completing coordinate check!"
```

```
st2 <- SePreprocess(se = st2, assay = "Spatial", norm.SCT = TRUE, cluster.resolution
= 0.8)
stf2 <- readRDS("~/test_data/stf2.rds")
```

In SpaTME, the `TNIscore` and `DefineTNIregion` functions are developed to calculate the TNI scores and determine the final TNI regions respectively. Firstly, we adjust the proportions of tumor cells using the total abundances of each spot (This step can be ignored if only proportions are provided).

```
cell.abun <- readRDS("~/test_data/cell.abun_2.rds")
abun.all <- log2(rowSums(cell.abun)+1)
abun.all <- abun.all/max(abun.all)
cell.abun <- cell.abun/rowSums(cell.abun)
ES <- cell.abun$Tumor*abun.all
ES <- ES/max(ES)
score <- TNIscore(ES, st_pos = stf2@Position,
                  cluster = stf2@Annotation$bayes_cluster,
                  r.dist = 2)
SpotVisualize(pos = stf2@Position,
                meta = score, size = 1.5,
                title = "TNI score", limits = c(0,0.5),
              f.color = c("#ced4da", "#e5614b", "#a4161a"))
```

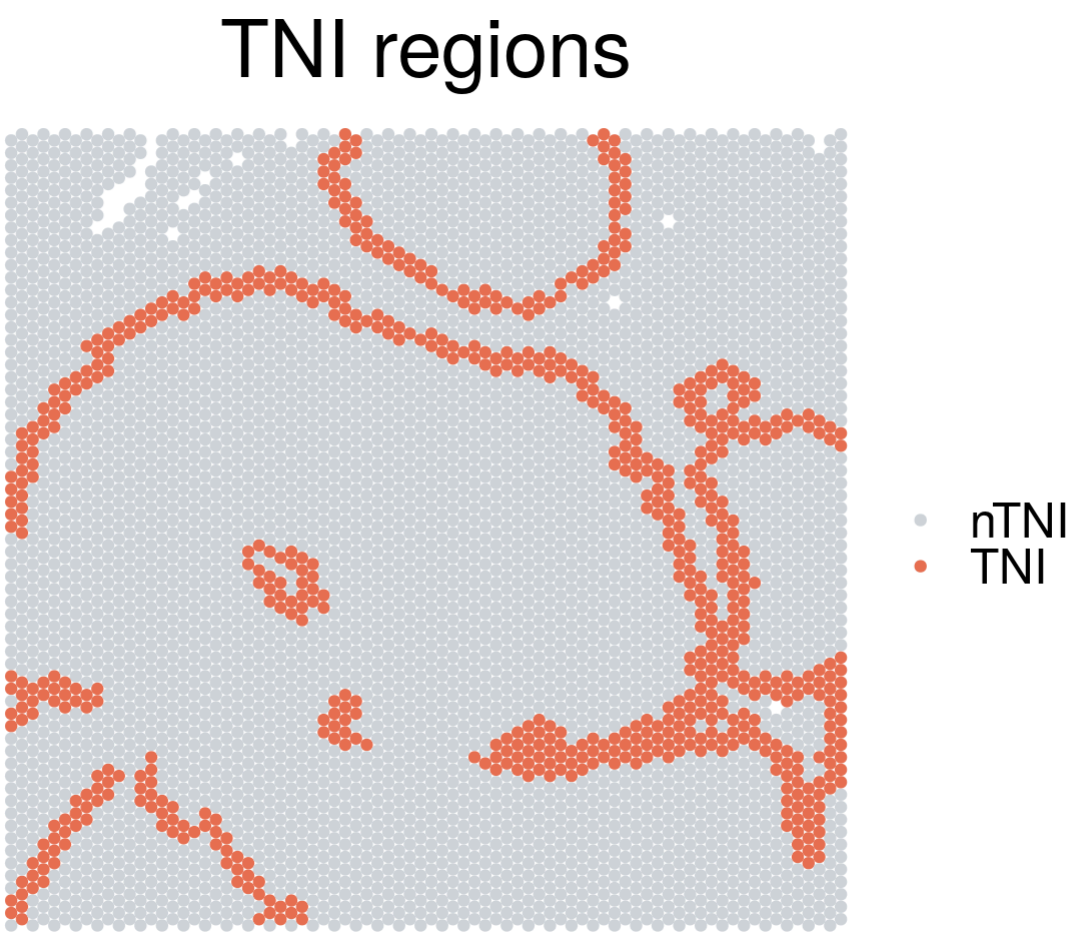


In `DefineTNIregion`, the `maxval` and `minval` directly control the selection of TNI spots.

```
TNIreg <- DefineTNIregion(score, st_pos = stf2@Position,
                          maxval = 0.08, minval = 0.03,
                          r.dist = 2, verbose = T)
```

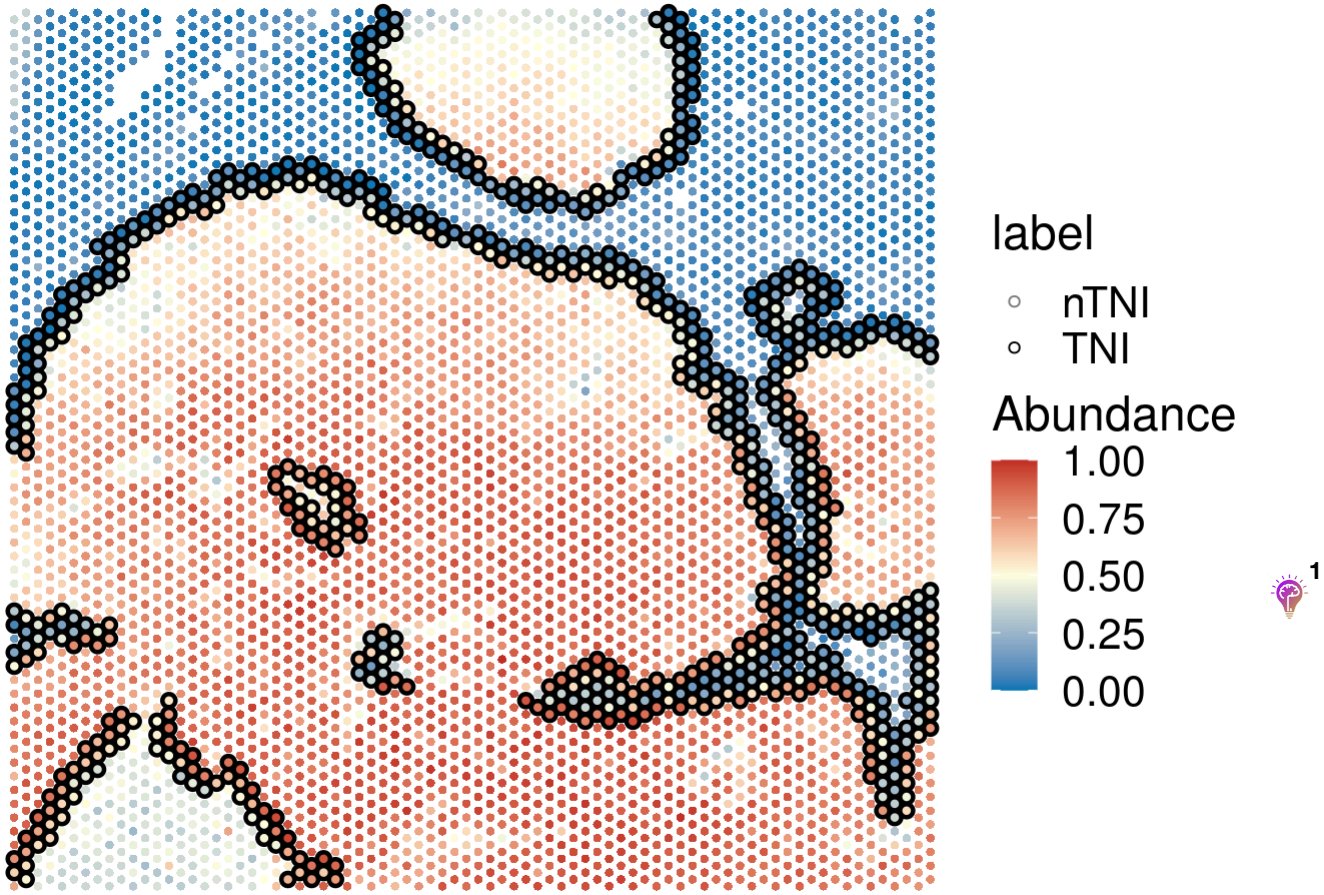
```
## [1] "Identifing TNI spots from edge scores ..."
## [1] "removing unconnected TNI regions ..."
## [1] "Completing the identificaton of TNI spots !"
```

```
SpotVisualize(pos = stf2@Position,
              meta = TNIreg, size = 1.5,
              cha.col = c(TNI = "#e76f51", nTNI = "#ced4da"),
              title = "TNI regions")
```



Visualization of TNI region and tumor cell distribution.

```
AbunTNIPlot(abun = ES, label = TNIreg, pos = stf2@Position,
            l_nshow = "nTNI", size = 1.5, legend.name = "Abundance",
            line_col = c("TNI" = "black", "nTNI" = NA))
```



SpaTME categorizes the TNI regions into distinct types based on their spatial domain composition.

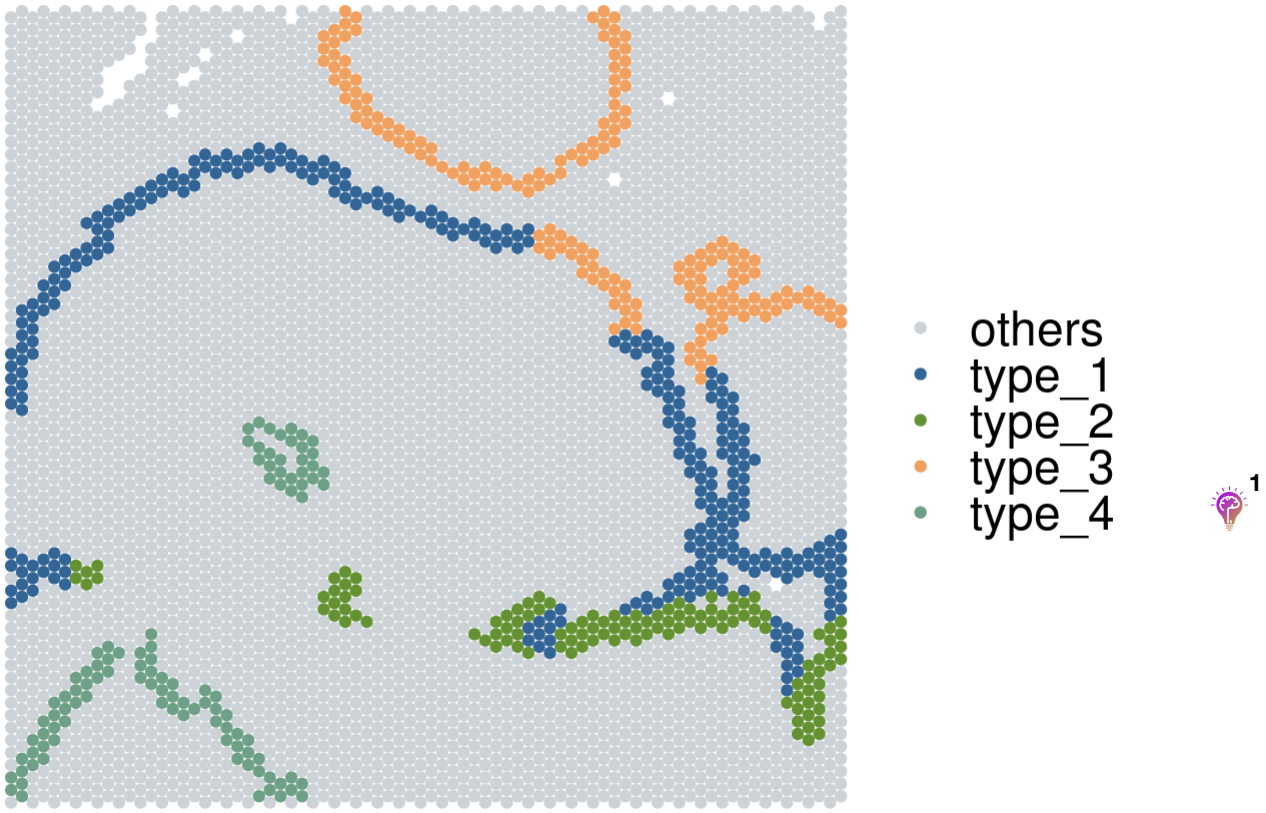
```
TNI_types <- GroupTNItypes(TNI_pos = stf2@Position[TNIreg == "TNI",],
                           cluster = stf2@Annotation$bayes_cluster[TNIreg == "TNI"])
```

```
## Separating tumor boundary to different type according it's label composition of sp
ot-neighbors
```

```
## pamk: best number of clusters is 4
```

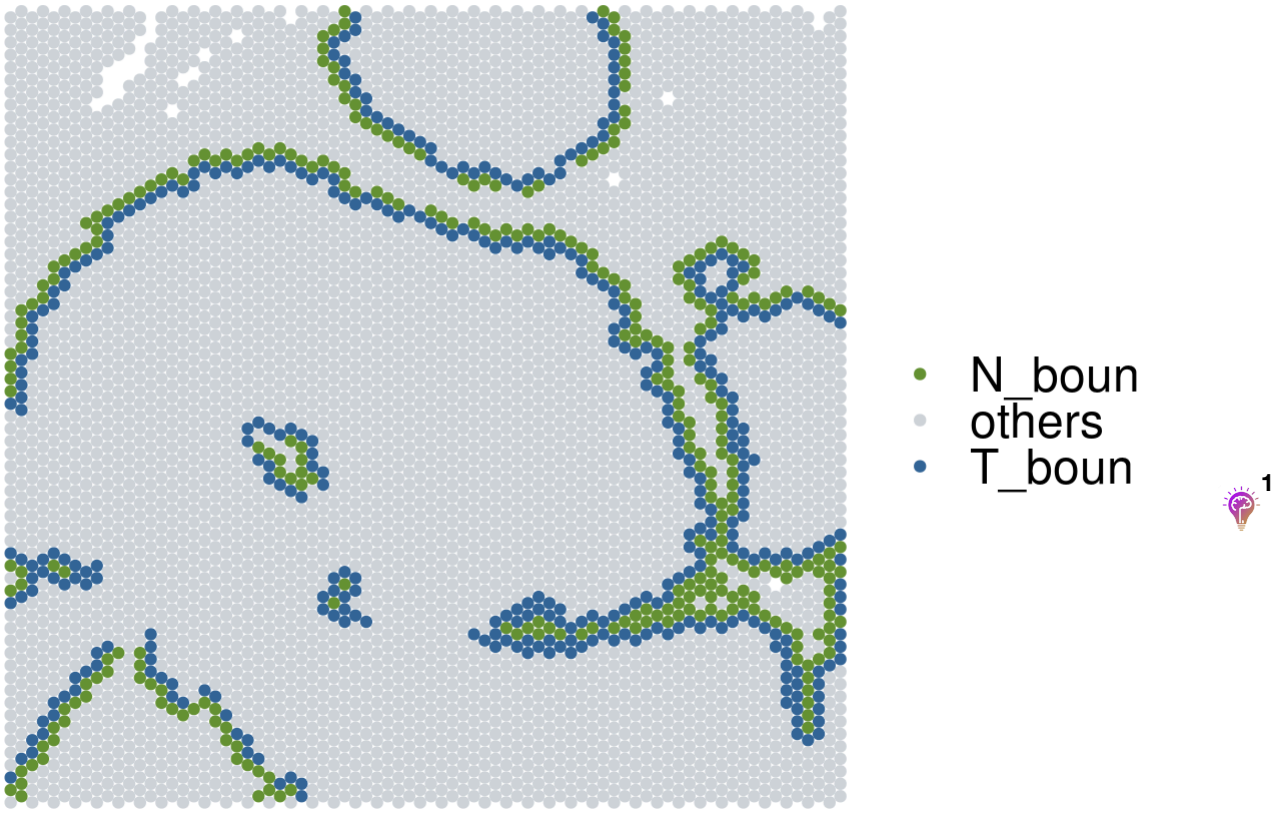
```
## Filtering short boundary type and remove discontinuous spots
```

```
allspots <- rep("others", length(TNIreg))
names(allspots) <- names(TNIreg)
allspots[names(TNI_types)] <- TNI_types
SpotVisualize(pos = stf2@Position,
              meta = allspots, size = 1.5,
              cha.col = LabelMapcolor(allspots, assgin.col = c(others = "#ced4da")),
              title = "TNI types")
```



SpaTME further classifies the TNI spots into either tumor or normal boundaries based on the tumor cell density.

```
TNI_class <- TNIClass(type = allspots, cluster = stf2@Annotation$bayes_cluster,
                      ES = ES)
SpotVisualize(pos = stf2@Position,
              meta = TNI_class, size = 1.5,
              cha.col = LabelMapcolor(labels = TNI_class,
                        assgin.col = c(others = "#ced4da")),
             title = "TNI classes")
```



Linking with phentoypes

The bulk RNA-seq profiles and paired clinical annotations of KIRC from TCGA were downloaded and preprocessed. The phenotype-related genes are identified using `PhenoAssoFeatures` function with `cox` method. As an illustration, we conducted NMF analysis using survival-negative genes.

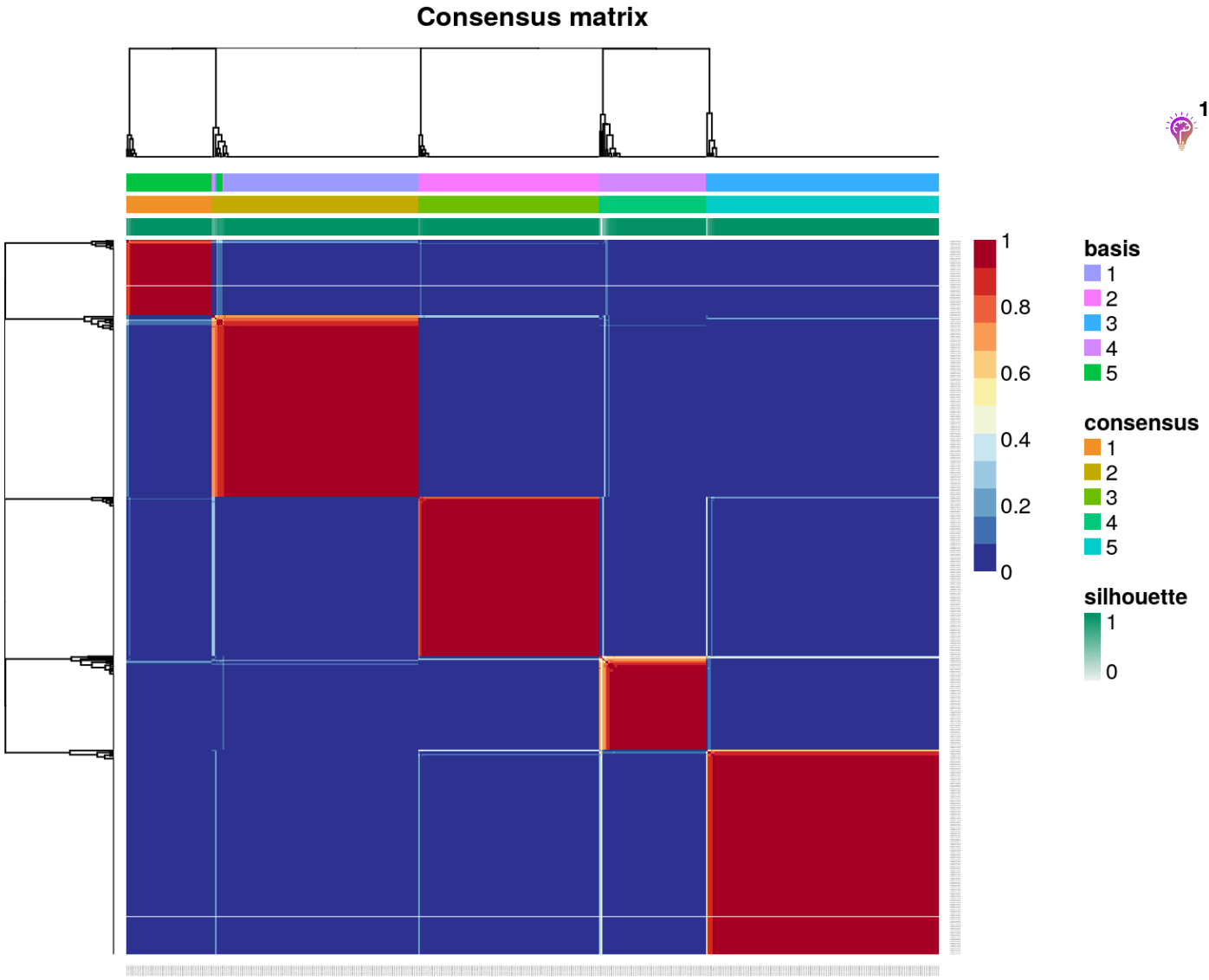
```
Bulk_data <- readRDS("~/test_data/TCGA_KIRC.rds")
bulk_input <- NMF_bulk_input(data = Bulk_data$Expr_TPM, pt_gene_exp = 0.2, totpm = F,
base_gene_exp = 1, dolog = F)
asso.genes <- PhenoAssoFeatures(data = bulk_input, phenotype = Bulk_data$phenotype,
                                method = "cox", p.cut = 0.01)
used.genes <- asso.genes[asso.genes$coef > 0, "features"]
```

In SpaTME, the NMF analysis is completed with the `RunNMFtest` function. The number of factors (parameter `rank`) can be specified with a single numeric value or selected from a numeric vector.

```
library(NMF)
nmf <- RunNMFtest(expr = bulk_input[used.genes,], rank = seq(2, 8),
                      min_cophenetic = 0.95, return.all = F)
```

```
## [1] "2024-07-29 15:58:14 CST"
## Compute NMF rank= 2  ... + measures ... OK
## Compute NMF rank= 3  ... + measures ... OK
## Compute NMF rank= 4  ... + measures ... OK
## Compute NMF rank= 5  ... + measures ... OK
## Compute NMF rank= 6  ... + measures ... OK
## Compute NMF rank= 7  ... + measures ... OK
## Compute NMF rank= 8  ... + measures ... OK
```

```
consensusmap(nmf)
```

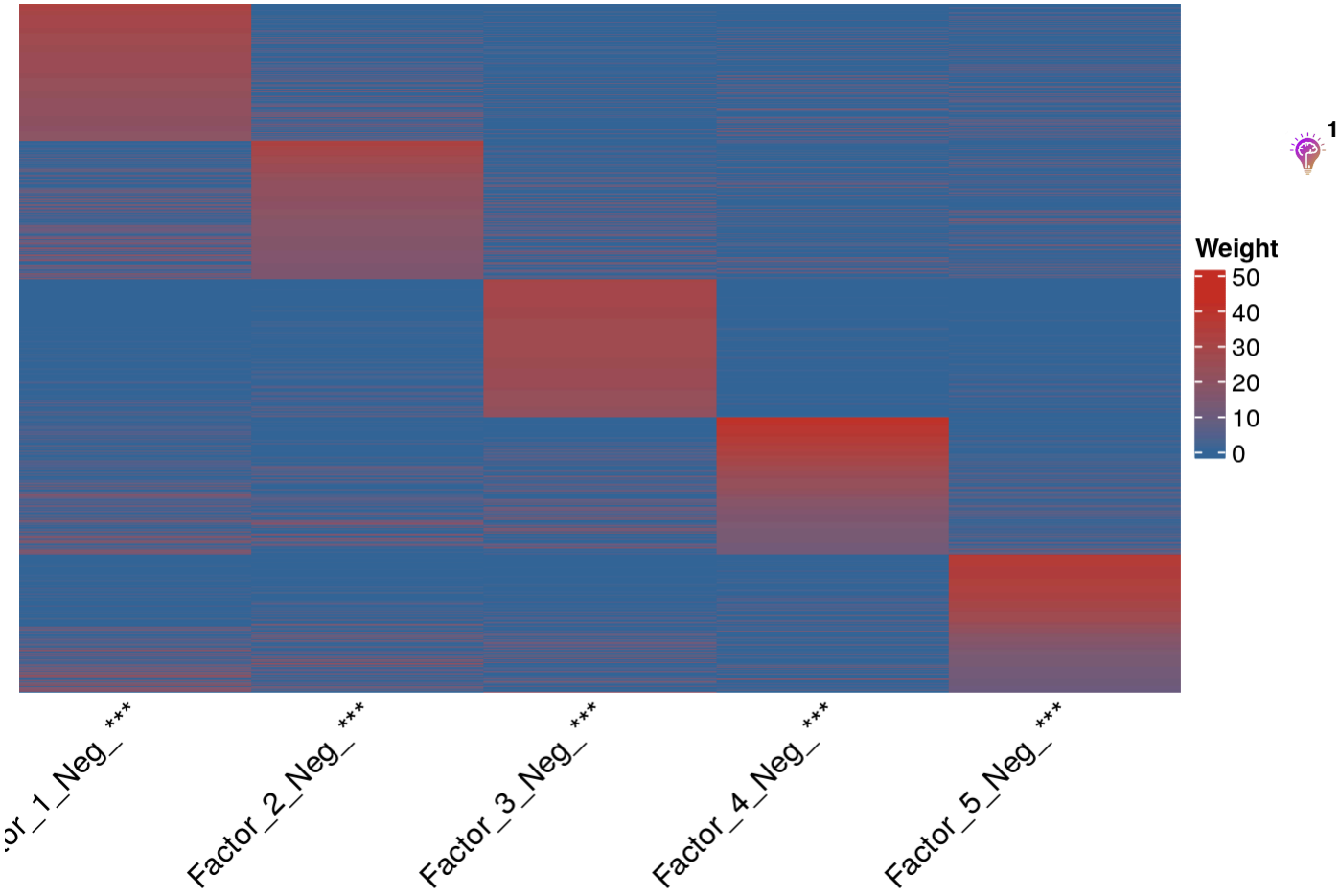



```
W_type <- HPhenoAsso(nmfobj = nmf, phenotype = Bulk_data$phenotype, method = "cox",
                     p.cut = 1, cox.test.cut = 0, p.adj = F,verbose = F)
print(W_type)
```

```
##  Factor_1  Factor_2  Factor_3  Factor_4  Factor_5
## "Neg_***" "Neg_***" "Neg_***" "Neg_***" "Neg_***"
```

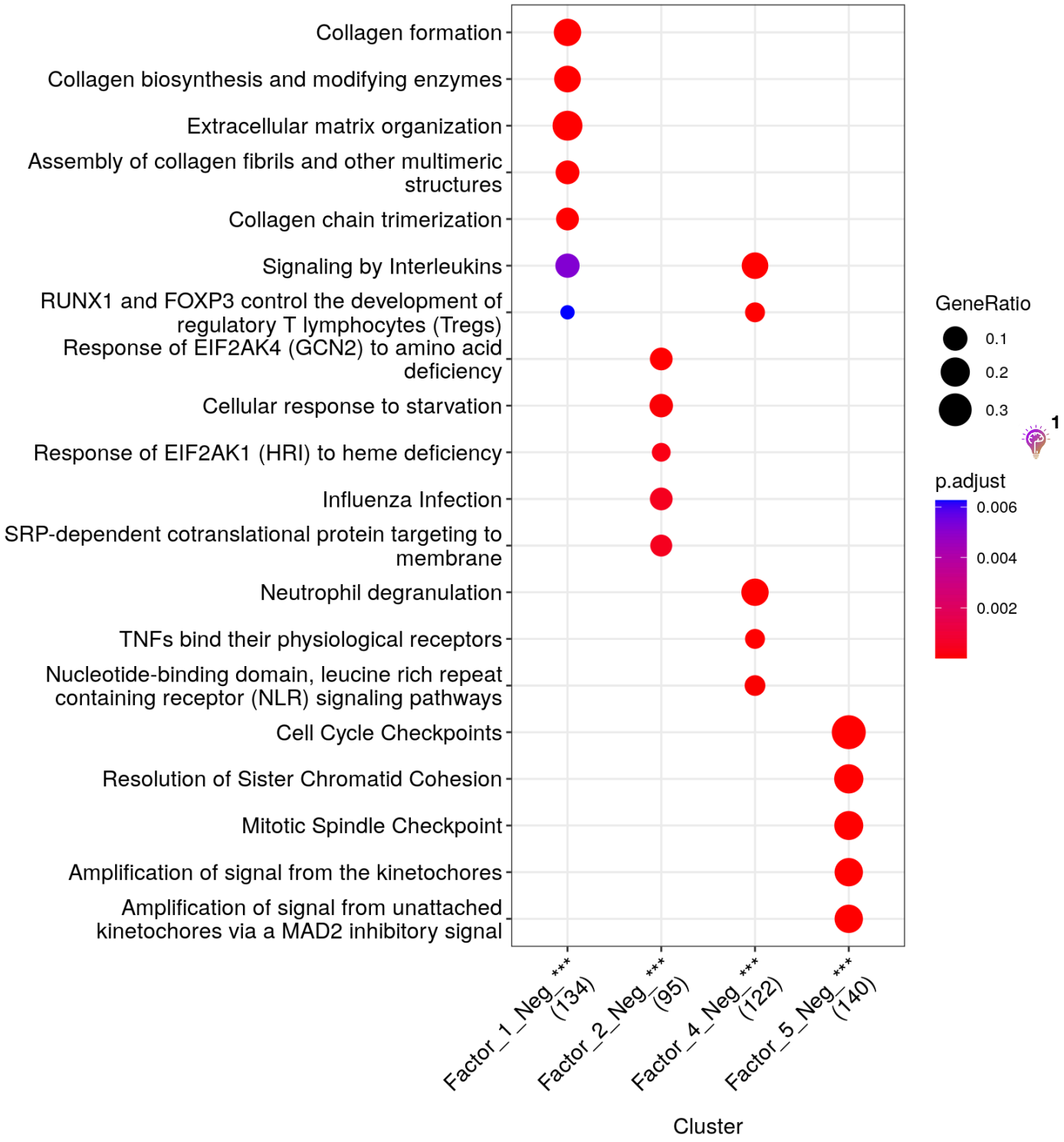
Showing the top metagenes of factors.

```
W <- basis(nmf)
colnames(W) <- paste0(names(W_type), "_", W_type)
mg_vt <- FactorMetagenes(ref_W = W, top_num = 200)
library(ComplexHeatmap)
library(circlize)
Heatmap(W[names(mg_vt),], show_row_names = F, row_title = NULL, cluster_rows = F,
                name = "Weight", split = mg_vt, cluster_columns = F, column_names_rot
= 45, row_gap = unit(0, "mm"),
                col = colorRamp2(c(min(W), max(W)), colors = c("#336699", "#c32f27")))
```



Enrichment analysis of these metagenes.

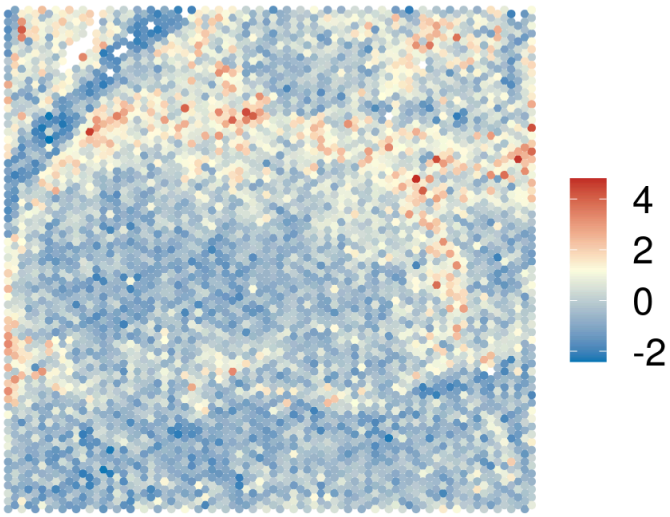
```
mg_enrich <- FactorEnrichAnalysis(mg_vt = mg_vt, wrap_width = 50,
                                  fun = "enrichPathway",
                                  pAdjustMethod = "none")
```



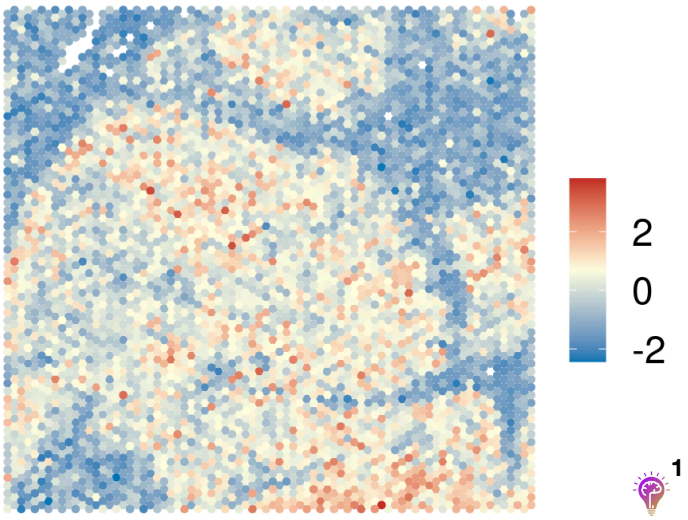
SpaTME recovers the relative expression of these factors in ST data.

```
NMFpred <- PredNMFinST(st2, W = nmf@fit@W,
                       assay = "SCT", slot = "data",
                       size = 1.2, numCol = 2)
```

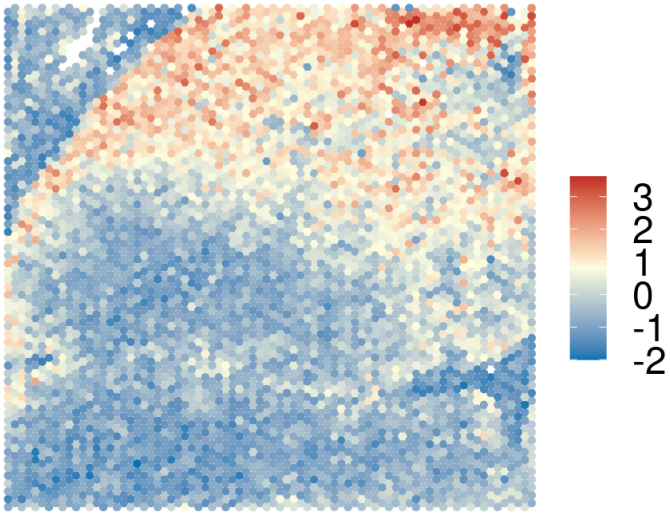
Factor_1



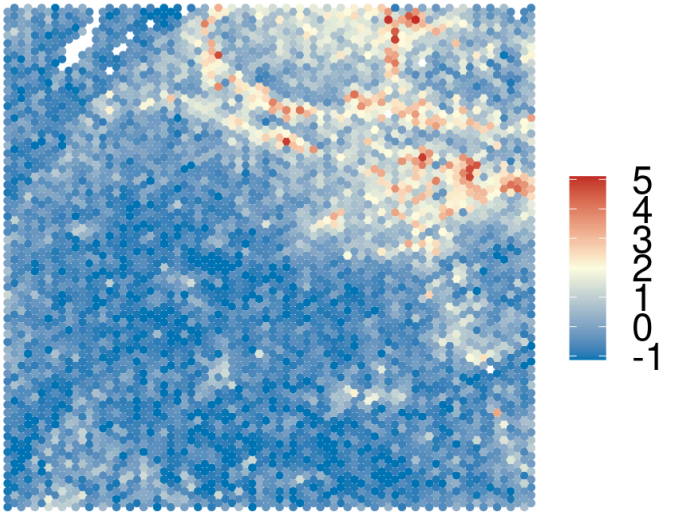
Factor_2



Factor_3



Factor_4



Factor_5