Marker identification and automated clustering based on gene activities

Ivan Berest and Christian Arnold

Contents

Goals

This vignette will teach you how to quantify the activity of each gene in the genome by assessing the chromatin accessibility associated with each gene. We will create a new assay for this and add it to our `Seurat` object. Also, we will write our first R function to visualize individual genes, connect the gene activities with the UMAP we have seen before, and we will also identify the top marker genes per cluster based on the new gene activity assay!

References

This vignette contains a modified version of this vignette.

Load libraries

As before, we first load all necessary R packages that are needed for this vignette.

```
suppressPackageStartupMessages({
library(Signac)
library(Seurat)
library(tidyverse)
library(patchwork)
library(clustermole)
})
set.seed(1990)
```

Load data

We begin by loading our pre-processed `Seurat` from the introductory QC vignette object into R. This code is shared among all subsequent vignettes.

```
# Make sure to have a trailing slash here
outFolder="/datadisk/ATAC_2022/outs/"
outFolder="/g/scb2/zaugg/zaugg_shared/Courses_and_Teaching/ATAC-Seq_Course/2022/data/testRun_March/coun
seu.s = readRDS(file = paste0(outFolder,"obj.filt.rds"))

# Make sure we have the ATAC array in the object
seu.s[["ATAC"]]
```

```
## ChromatinAssay data with 158583 features for 3401 cells
## Variable features: 119325
## Genome: mm10
## Annotation present: TRUE
## Motifs present: FALSE
## Fragment files: 1
```

```
# In case the object contains multiple assays, make the ATAC assay the standard
DefaultAssay(seu.s) = "ATAC"
```

Add and analyze gene activities

The UMAP visualization we have seen before reveals the presence of multiple cell groups. How can we learn more about them?

Unfortunately, annotating and interpreting clusters is more challenging in scATAC-seq data as much less is known about the functional roles of non-coding genomic regions than is known about protein coding regions (genes).

However, we can try to quantify the activity of each gene in the genome by assessing the chromatin accessibility associated with each gene, and create a new gene activity assay derived from the scATAC-seq data. Here we will use a simple approach of summing the fragments intersecting the gene body and promoter region. However, the `Cicero` R package accomplishes a similar goal, see the other vignette for this!

Calculate gene activities

To create a gene activity matrix, we first extract gene coordinates and then extend them to include the (here) 3 kb upstream region (as promoter accessibility is often correlated with gene expression) as well as 100 bp downstream region. We then count the number of fragments for each cell that map to each of these regions, using the `FeatureMatrix()` function. These steps are automatically performed by the function `GeneActivity()` as shown below:

```
gene.activities <- GeneActivity(seu.s, extend.upstream = 3000, extend.downstream = 100)
```

```
## Extracting gene coordinates
```

```
## Extracting reads overlapping genomic regions
```

Create gene activity assay

```
# add the gene activity matrix to the Seurat object as a new assay and normalize it
newAssayName = "geneact"
seu.s[[newAssayName]] <- CreateAssayObject(counts = gene.activities)
seu.s <- NormalizeData(
  object = seu.s,
  assay = newAssayName,
  normalization.method = 'LogNormalize',
  scale.factor = median(seu.s@meta.data[,paste0("nCount_",newAssayName)])
)
```

```
# Change the default assay name to our new geneact name for this vignette
DefaultAssay(seu.s) <- newAssayName

# Scaling seems necessary for the heatmap, but does it affect something else? TODO
seu.s <- ScaleData(seu.s, verbose = F)
```

Find markers

It is time to find markers again. Here, we base the cell identities on the previously calculated embedding with resolution of 0.5. We then again invoke the `FindAllMarkers` function, but this time with the new gene activity assay!

```
topMarkerGenesPerCluster = 3

# Set identity of cells based on ATAC_snn_res.0.5
Idents(seu.s) <- seu.s$ATAC_snn_res.0.5
markersGA <- FindAllMarkers(seu.s, only.pos = TRUE, min.pct = 0.1, logfc.threshold = 0.1, assay = newAs

gr1 = markersGA %>%
    group_by(cluster) %>%
    slice_max(n = topMarkerGenesPerCluster, order_by = avg_log2FC) %>%
    ungroup()
gr1
```

```
## # A tibble: 15 x 7
##        p_val avg_log2FC pct.1 pct.2 p_val_adj cluster gene
##        <dbl>      <dbl> <dbl> <dbl>     <dbl> <fct>   <chr>
##  1 1.12e- 66      0.519 0.671 0.411 2.42e- 62 0       Ppp4r4
##  2 5.71e- 47      0.401 0.747 0.54  1.23e- 42 0       Pitx2
##  3 3.92e- 44      0.357 0.529 0.301 8.48e- 40 0       Larp7
##  4 2.22e-130      0.803 0.88  0.671 4.81e-126 1       Esrrb
##  5 4.74e- 81      0.693 0.836 0.679 1.03e- 76 1       Kctd16
##  6 2.18e- 80      0.641 0.616 0.347 4.72e- 76 1       Tfcp2l1
##  7 2.28e- 13      0.237 0.773 0.627 4.93e-  9 2       Ntrk2
##  8 2.51e-  8      0.207 0.73  0.638 5.43e-  4 2       Abcc4
##  9 5.70e-  6      0.165 0.77  0.718 1.23e-  1 2       Aff3
## 10 8.02e- 41      0.750 0.862 0.479 1.73e- 36 3       Ahnak
## 11 1.36e- 38      0.650 0.96  0.759 2.93e- 34 3       Myh9
## 12 3.04e- 43      0.639 0.693 0.282 6.56e- 39 3       Itgb6
## 13 1.04e- 12      0.369 0.605 0.393 2.24e-  8 4       Syne3
## 14 6.07e- 31      0.331 0.343 0.091 1.31e- 26 4       Fbxo40
## 15 4.55e-  9      0.299 0.424 0.25  9.83e-  5 4       Mfhas1
```

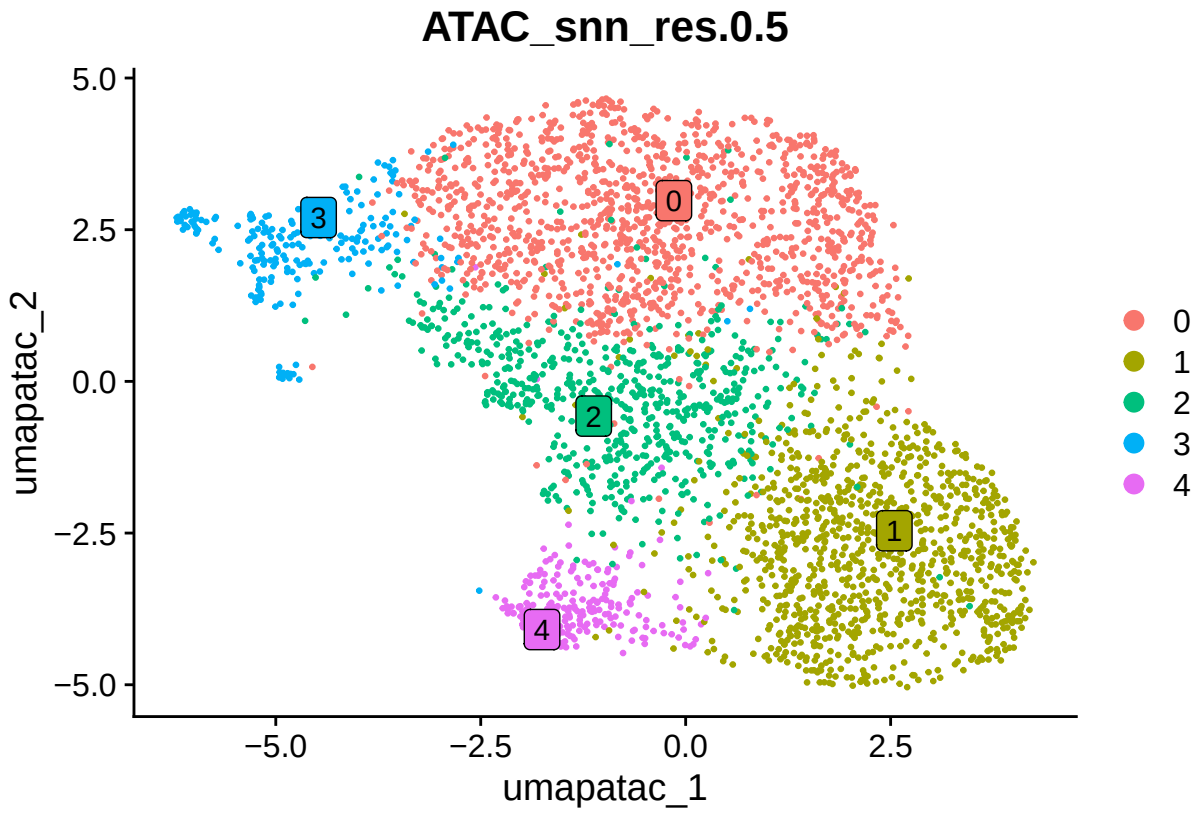
The new table that we just produced shows the new marker genes per cluster, along with raw and adjusted p-values, and the average log2 fold-change.

Marker genes UMAP

Now we can visualize the activities of canonical marker genes to help interpret our ATAC-seq clusters. Note that the activities will be much noisier than scRNA-seq measurements. As discussed before, this is because they represent measurements from sparse chromatin data, and because they assume a general correspondence between gene body/promoter accessibility and gene expression which may not always be the case. Nonetheless, we can begin to discern the various populations based on these gene activity profiles. However, further subdivision of these cell types is challenging based on supervised analysis alone.

Before we start, let's recapitulate how the clusters look like from the UMAP:

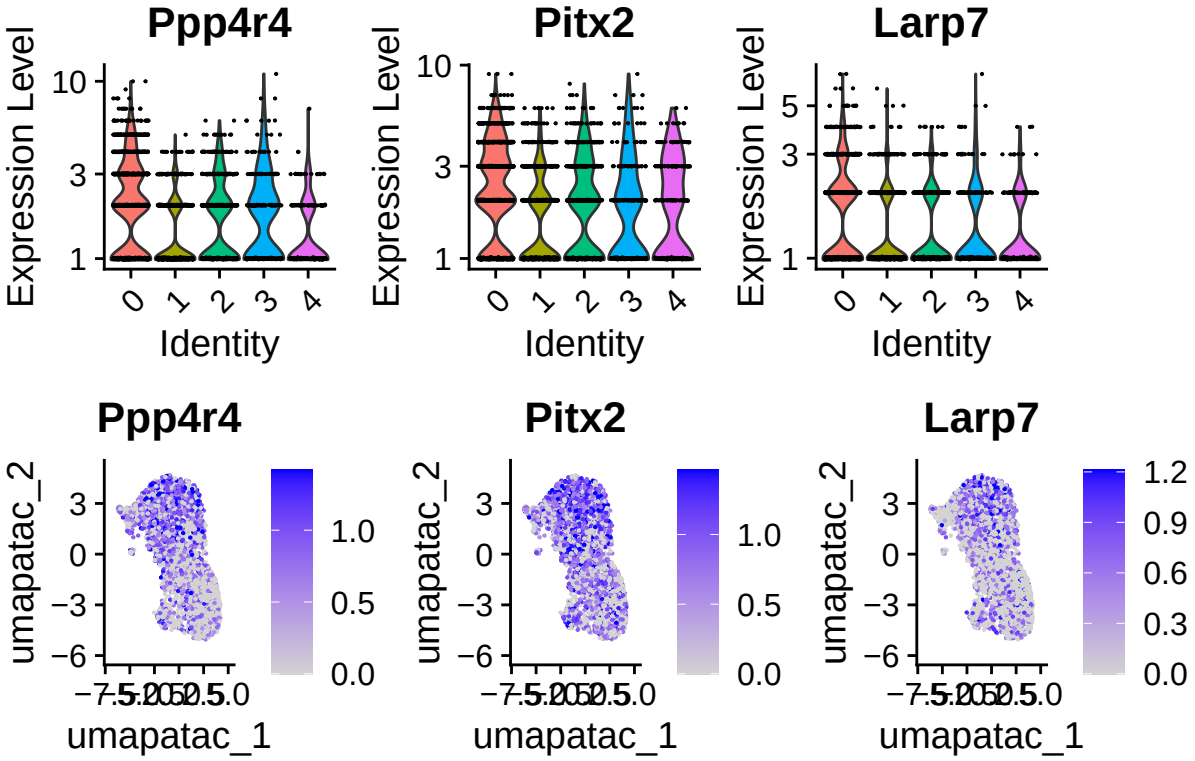
```
res = 0.5

DimPlot(object = seu.s, pt.size = 0.5,
            group.by = paste0("ATAC_snn_res.",res),
            label = TRUE, repel = TRUE, label.box = TRUE)
```



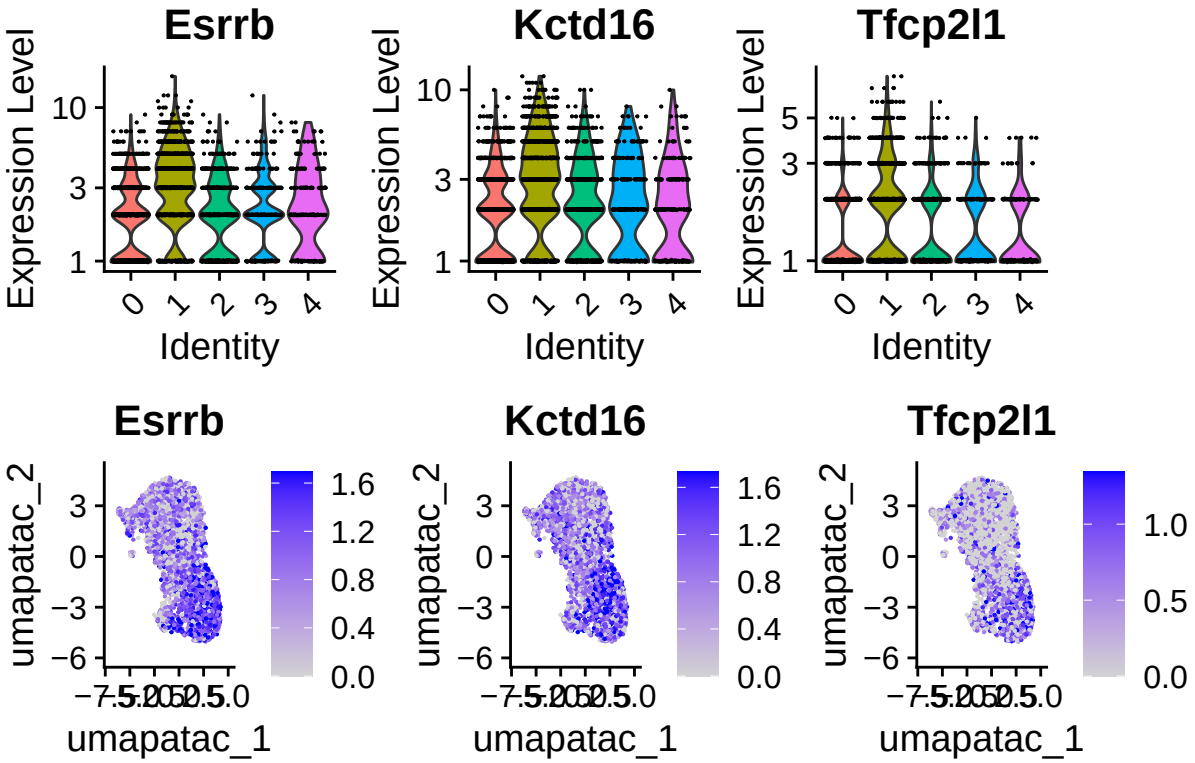
Now, let's visualize the top marker genes per cluster:

```
for (cl in sort(unique(gr1$cluster))) {

    # get a list of all cluster-specific genes
    fts = gr1 %>% filter(cluster == cl) %>% pull(gene)

    p1 = VlnPlot(seu.s, features = fts, slot = "counts", log = TRUE)
    p2 = FeaturePlot(seu.s, features = fts, max.cutoff = "q90", pt.size = 0.1, ncol = 3, reduction = "u

    # Construct the patchwork plot from both individual ones
    pall = p1 / p2 + plot_annotation(title = paste0("Cluster ",cl))
    print(pall)


}
```

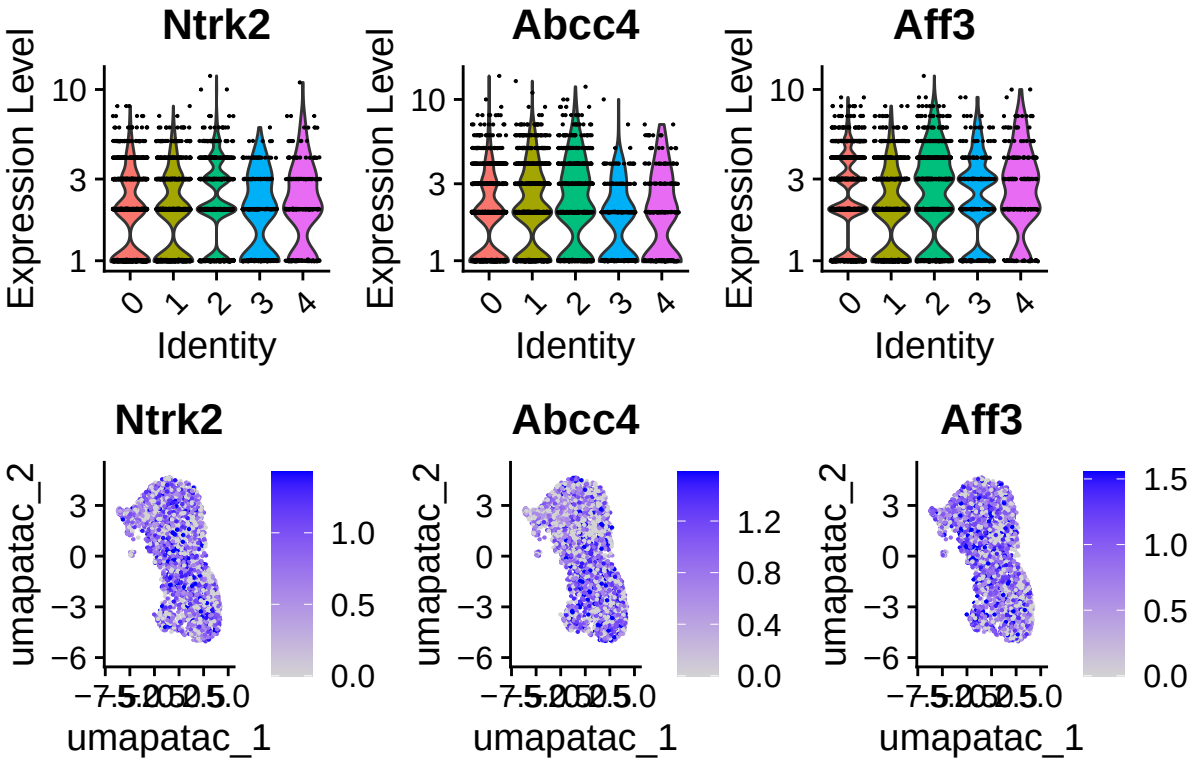
Cluster 0



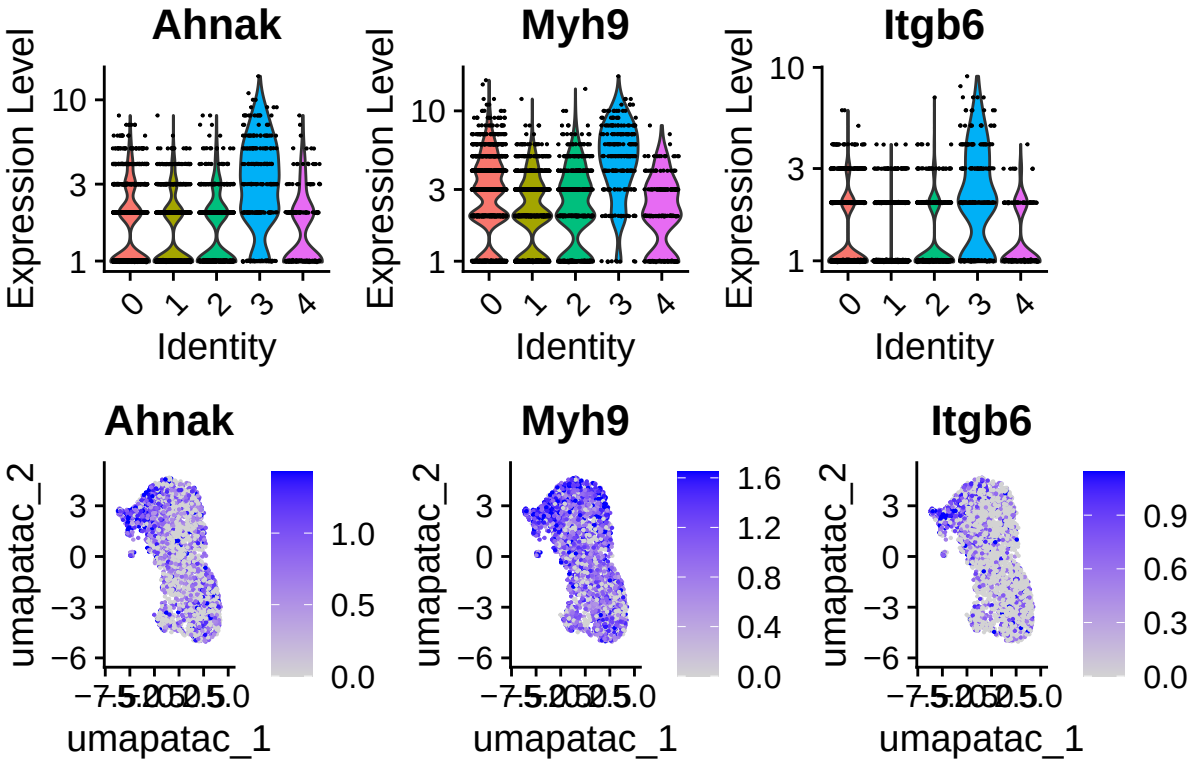
Cluster 1



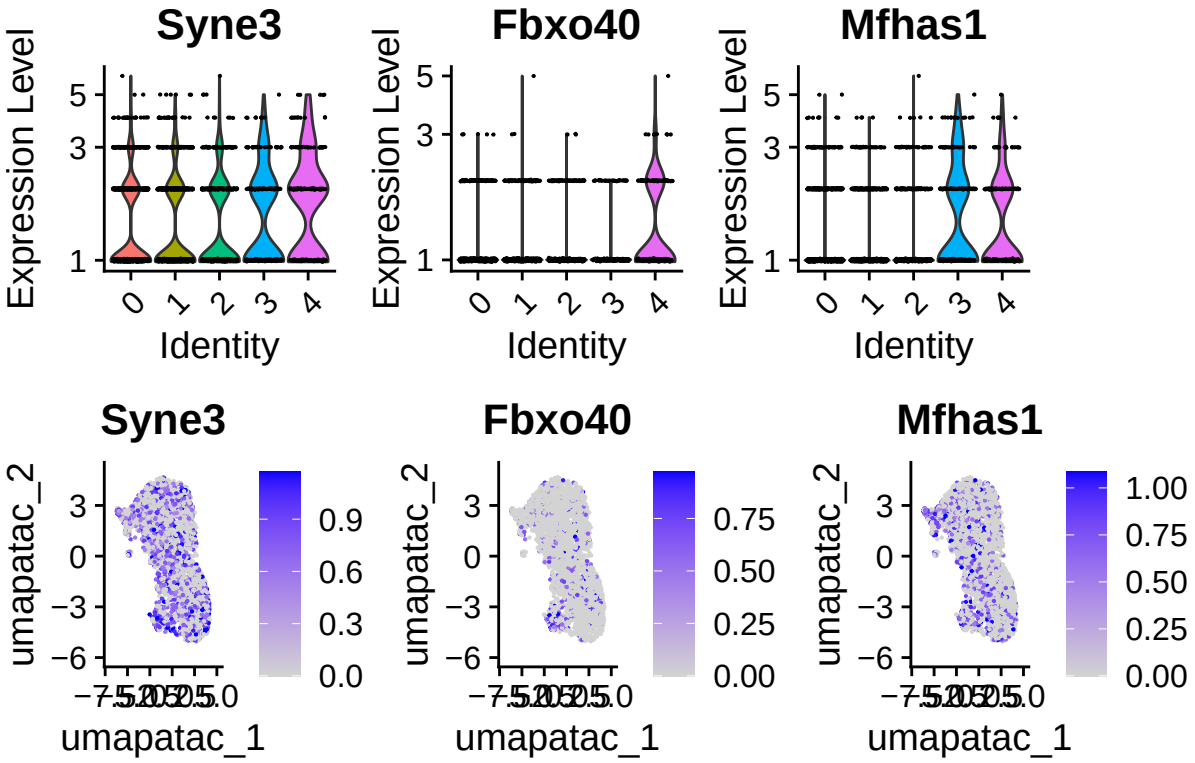
Cluster 2



Cluster 3



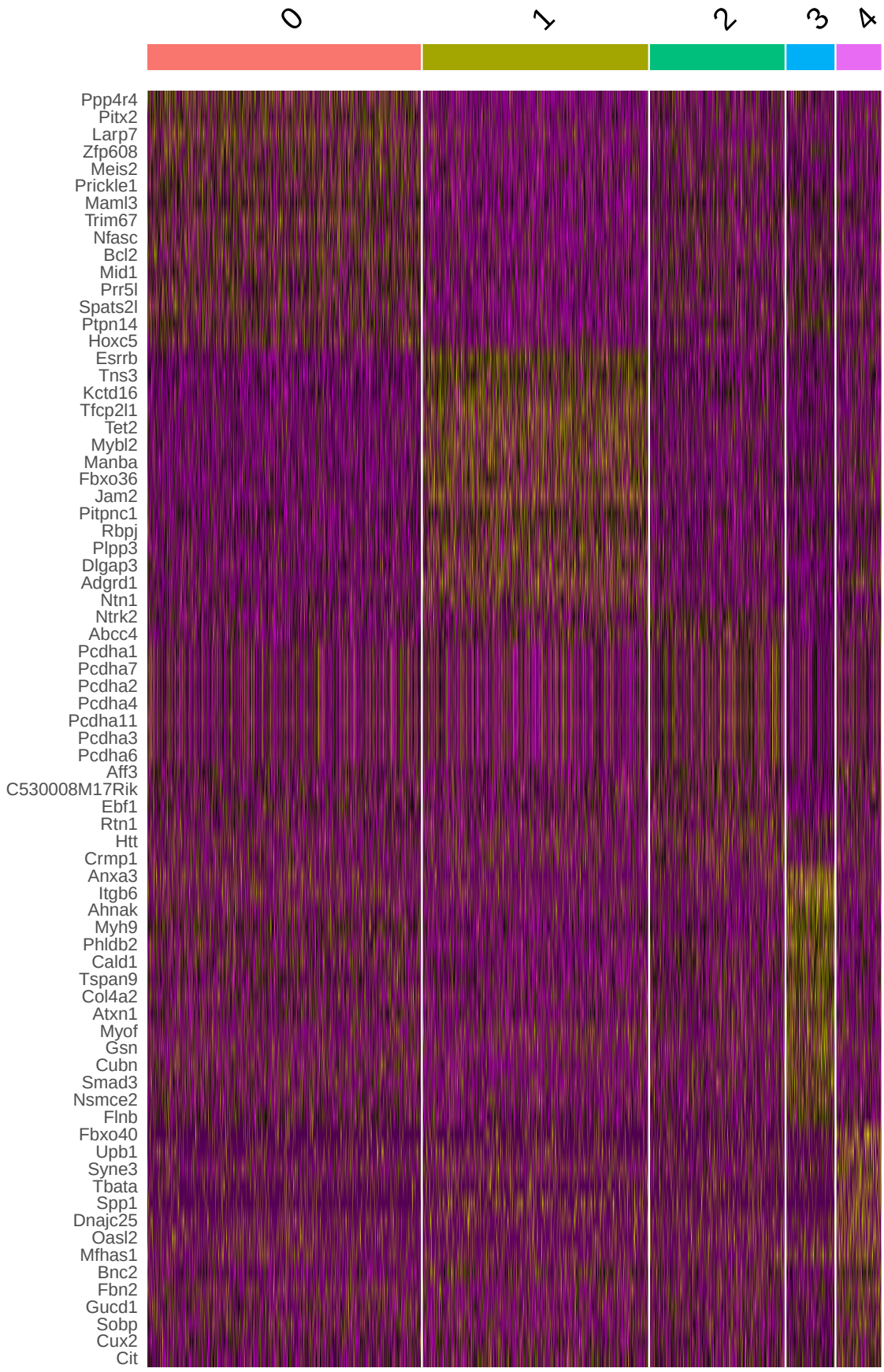
Cluster 4



Heatmap

Let's create a heatmap based on the top 15 genes per cluster, based on their average log2 fold-change.

```
top <- markersGA %>%
    group_by(cluster) %>%
    top_n(n = 15, wt = avg_log2FC)
DoHeatmap(seu.s, features = top$gene, assay = newAssayName) + NoLegend()
```


0
1
2
3
4
Ppp4r4
Pitx2
Larp7
Zfp608
Meis2
Prickle1
Maml3
Trim67
Nfasc
Bcl2
Mid1
Prr5l
Spats2l
Ptpn14
Hoxc5
Esrrb
Tns3
Kctd16
Tfcp2l1
Tet2
Mybl2
Manba
Fbxo36
Jam2
Pitpnc1
Rbpj
Plpp3
Dlgap3
Adgrd1
Ntn1
Ntrk2
Abcc4
Pcdha1
Pcdha7
Pcdha2
Pcdha4
Pcdha11
Pcdha3
Pcdha6
Aff3
C530008M17Rik
Ebf1
Rtn1
Htt
Crmp1
Anxa3
Itgb6
Ahnak
Myh9
Phldb2
Cald1
Tspan9
Col4a2
Atxn1
Myof
Gsn
Cubn
Smad3
Nsmce2
Flnb
Fbxo40
Upb1
Syne3
Tbata
Spp1
Dnajc25
Oasl2
Mfhas1
Bnc2
Fbn2
Gucd1
Sobp
Cux2
Cit

As we can see in the heatmap, each cluster has a specific set of marker genes!

Automated annotation with clustermole based on marker genes

As seen and calculated before, a typical pipeline to process single-cell data (such as scRNA-seq and scATAC-seq) includes clustering of cells. Assigning cell type labels to clusters is often a time-consuming and involves manual inspection of the cluster marker genes complemented with a detailed literature search. This task becomes even more daunting in the presence of unexpected or poorly described populations. The `clustermole` R package provides methods to query thousands of human and mouse cell identity markers sourced from a variety of databases. Here, we will give it a try!

```
# Let's first create a data frame with the top 30 markers according to the average log2 fold-change
topMarkers <- markersGA %>%
    group_by(cluster) %>%
    top_n(n = 30, wt = avg_log2FC)

# We now create a new list in which we store the cluster-specific annotation of marker genes
celltypes.l = list()
for (cluster.id in unique(top$cluster)){
  celltypes.l[[cluster.id]] = clustermole_overlaps(genes = topMarkers %>% filter(cluster == cluster.id)

}
# How does our list look like? let's check for clusters 0 and 1
celltypes.l[["0"]]
```

```
## # A tibble: 13 x 9
##    celltype_full    db     species organ celltype  n_genes overlap p_value   fdr
##    <chr>            <chr>  <chr>   <chr> <chr>       <int>   <dbl>   <dbl> <dbl>
##  1 Type IB spiral ~ CellM~ Mouse   "Bra~ Type IB ~      43       2 0.00295 0.427
##  2 Neuroendocrine ~ CellM~ Mouse   "Tra~ Neuroend~     370       4 0.00490 0.617
##  3 Parathyroid chi~ Pangl~ Mouse   "Par~ Parathyr~       7       1 0.0128  0.894
##  4 Rheaume et al.N~ CellM~ Mouse   "Ret~ Rheaume ~       7       1 0.0128  0.894
##  5 Mast cells | Im~ Pangl~ Mouse   "Imm~ Mast cel~      99       2 0.0142  0.894
##  6 Enterochromaffi~ Pangl~ Mouse   "GI ~ Enteroch~       9       1 0.0165  0.894
##  7 Glutaminergic n~ Pangl~ Mouse   "Bra~ Glutamin~      10       1 0.0183  0.926
##  8 Embryonic carci~ TISSU~ Mouse   ""    Embryoni~      10       1 0.0201  0.926
##  9 Embryonic cell ~ TISSU~ Mouse   ""    Embryoni~      10       1 0.0201  0.926
## 10 Merkel cells | ~ Pangl~ Mouse   "Ski~ Merkel c~      12       1 0.0219  0.952
## 11 Rheaume et al.N~ CellM~ Mouse   "Ret~ Rheaume ~      14       1 0.0255  1
## 12 Neuroblast | Br~ CellM~ Mouse   "Bra~ Neurobla~      19       1 0.0381  1
## 13 Anterior pituit~ Pangl~ Mouse   "Bra~ Anterior~      23       1 0.0416  1
```

```
celltypes.l[["1"]]
```

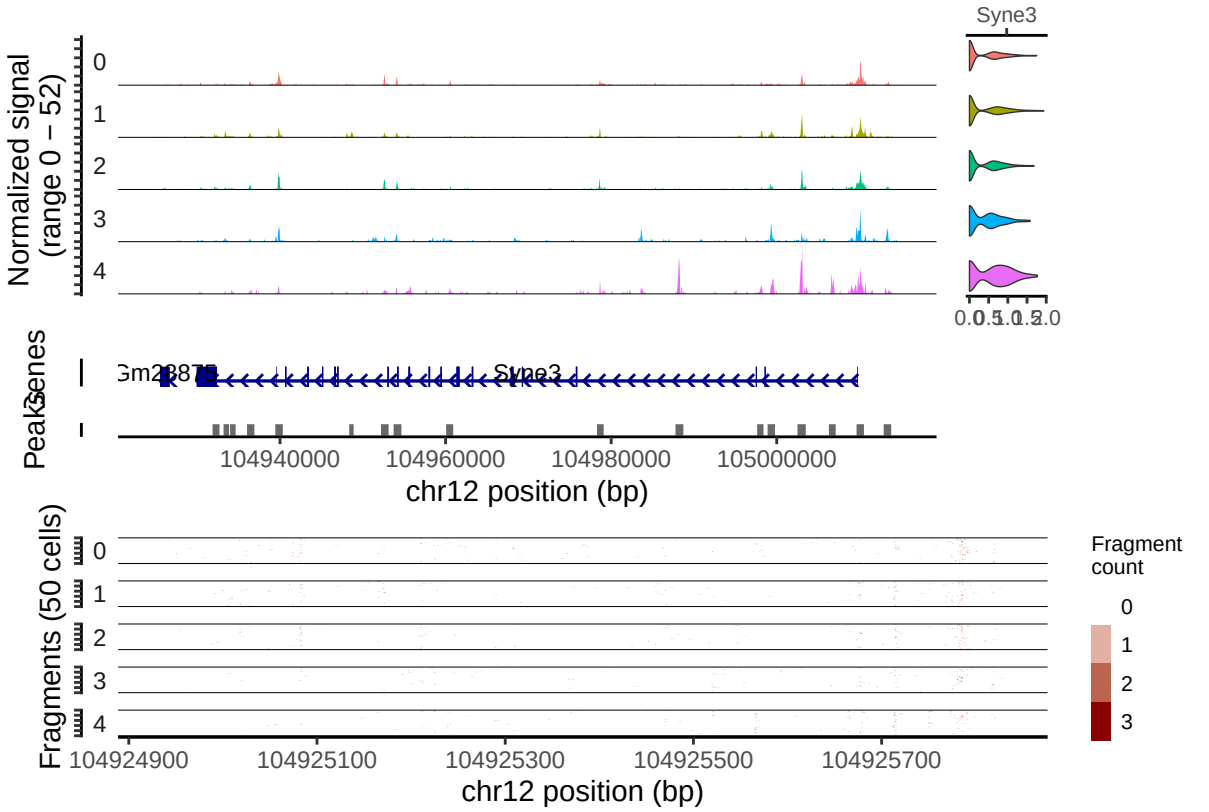
```
## # A tibble: 15 x 9
##    celltype_full    db     species organ celltype  n_genes overlap p_value   fdr
##    <chr>            <chr>  <chr>   <chr> <chr>       <int>   <dbl>   <dbl> <dbl>
##  1 Macrophage | Ki~ CellM~ Mouse   "Kid~ Macropha~      71       2 0.00898     1
##  2 Collecting duct~ CellM~ Mouse   "Kid~ Collecti~      87       2 0.0119      1
##  3 Multipotent pro~ CellM~ Mouse   "Bon~ Multipot~       7       1 0.0124      1
##  4 Ionocytes | Lun~ Pangl~ Mouse   "Lun~ Ionocytes       8       1 0.0141      1
##  5 Rheaume et al.N~ CellM~ Mouse   "Ret~ Rheaume ~       8       1 0.0141      1
##  6 Rheaume et al.N~ CellM~ Mouse   "Ret~ Rheaume ~       9       1 0.0176      1
##  7 Epithelium | Mo~ TISSU~ Mouse   ""    Epitheli~      14       1 0.0246      1
##  8 Dopaminergic ne~ Pangl~ Mouse   "Bra~ Dopamine~      17       1 0.0297      1
##  9 Interneuron-sel~ CellM~ Mouse   "Bra~ Interneu~      17       1 0.0297      1
```

```
## 10 Mesangial cell ~ CellM~ Mouse   "Kid~ Mesangia~     149       2 0.0302      1
## 11 Schwalie et al.~ CellM~ Mouse   "Adi~ Schwalie~     148       2 0.0324      1
## 12 Schwalie et al.~ CellM~ Mouse   "Adi~ Schwalie~     157       2 0.0335      1
## 13 Vascular endoth~ CellM~ Mouse   "Kid~ Vascular~     153       2 0.0339      1
## 14 Schwalie et al.~ CellM~ Mouse   "Adi~ Schwalie~     154       2 0.0373      1
## 15 CCK basket cell~ CellM~ Mouse   "Bra~ CCK bask~      24       1 0.0417      1
```

Visualise genomic tracks for selected genes

Let's visualize a particular gene. As we will re-use this functionality in a more general context (for a whole cluster) in the next chunk as well, we will here write our first own function to minimize code repetition!

```
visualizeSingleGene <- function(obj, geneName, rangeUp = 5000, rangeDown = 5000, dimReductionName) {

  cov_plot <- CoveragePlot(object = obj, region = geneName,
                           extend.upstream = rangeUp,extend.downstream = rangeDown,
                           features = geneName, expression.assay = "geneact",
                           annotation = TRUE, peaks = TRUE, links = FALSE,  assay = "ATAC")

  tile_plot <- TilePlot(object = obj, region = geneName,
                        idents = unique(FetchData(obj, vars = dimReductionName)[,dimReductionName]),
                        extend.upstream = rangeUp, extend.downstream = rangeDown,
                        tile.cells = 50, assay = "ATAC")

  p = CombineTracks(
    plotlist = list(cov_plot, tile_plot),
    heights = c(10,5)
  )
  plot(p)
}

visualizeSingleGene(obj = seu.s, geneName = "Syne3", dimReductionName = "ATAC_snn_res.0.5")
```

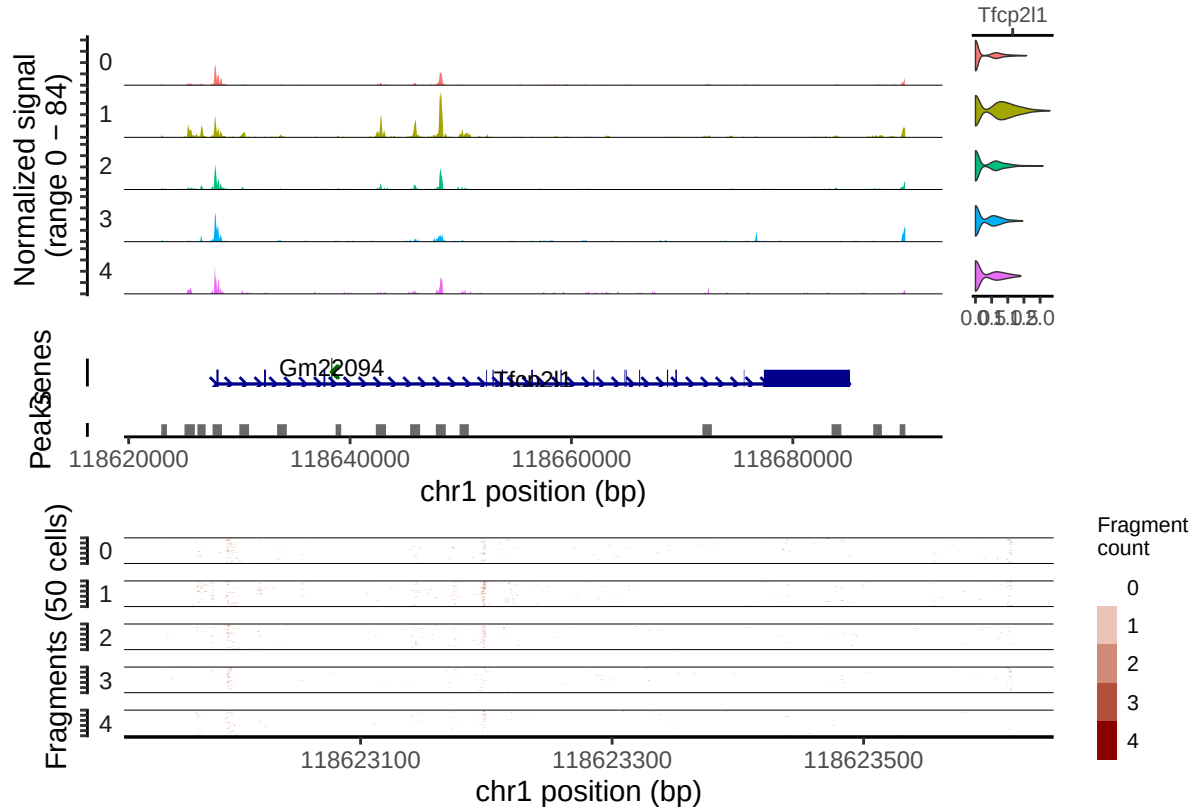


Now, let's make use of our new function and run this for all genes from a particular cluster, here cluster `0` as an example:

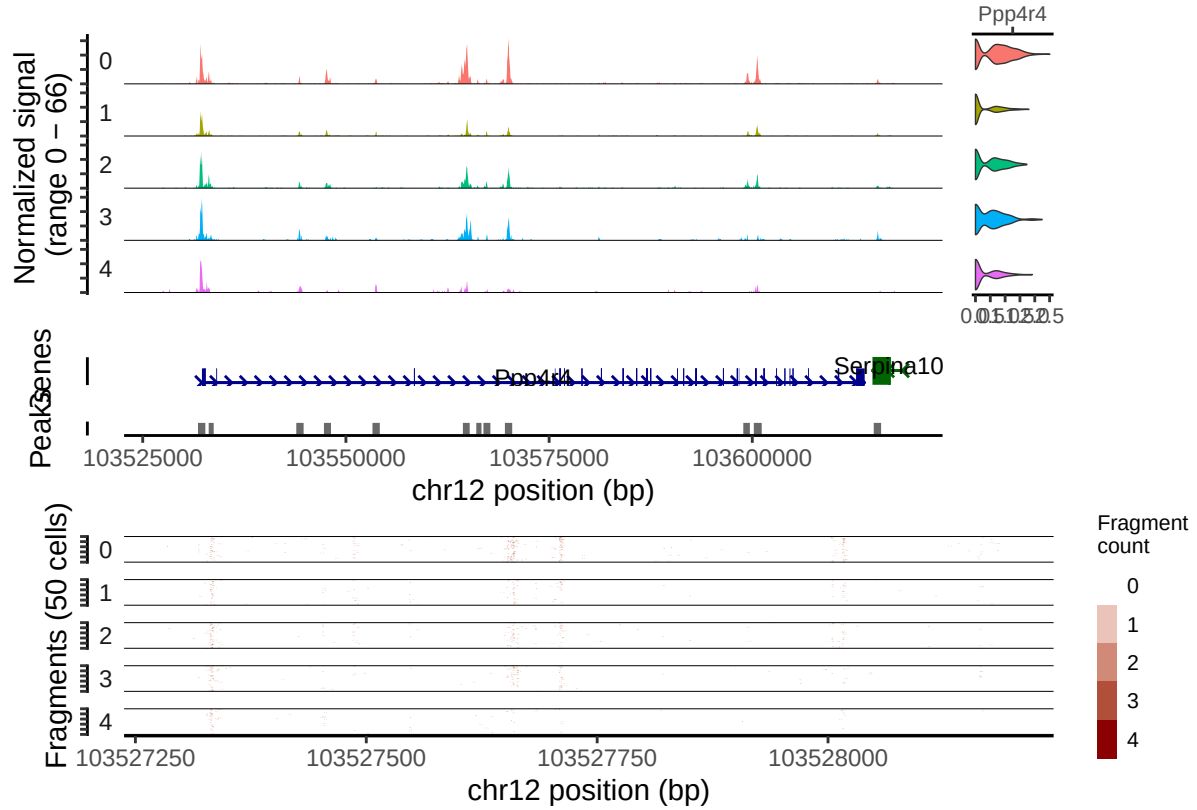
```
# Use one cluster as example
cluster = "1"

# Get the top 5 genes w.r.t average log2 fold-change
genes_top = gr1 %>%
    dplyr::filter(cluster == cluster) %>%
    dplyr::arrange(desc(avg_log2FC)) %>%
    dplyr::top_n(n = 3) %>%
    dplyr::pull(gene)
```

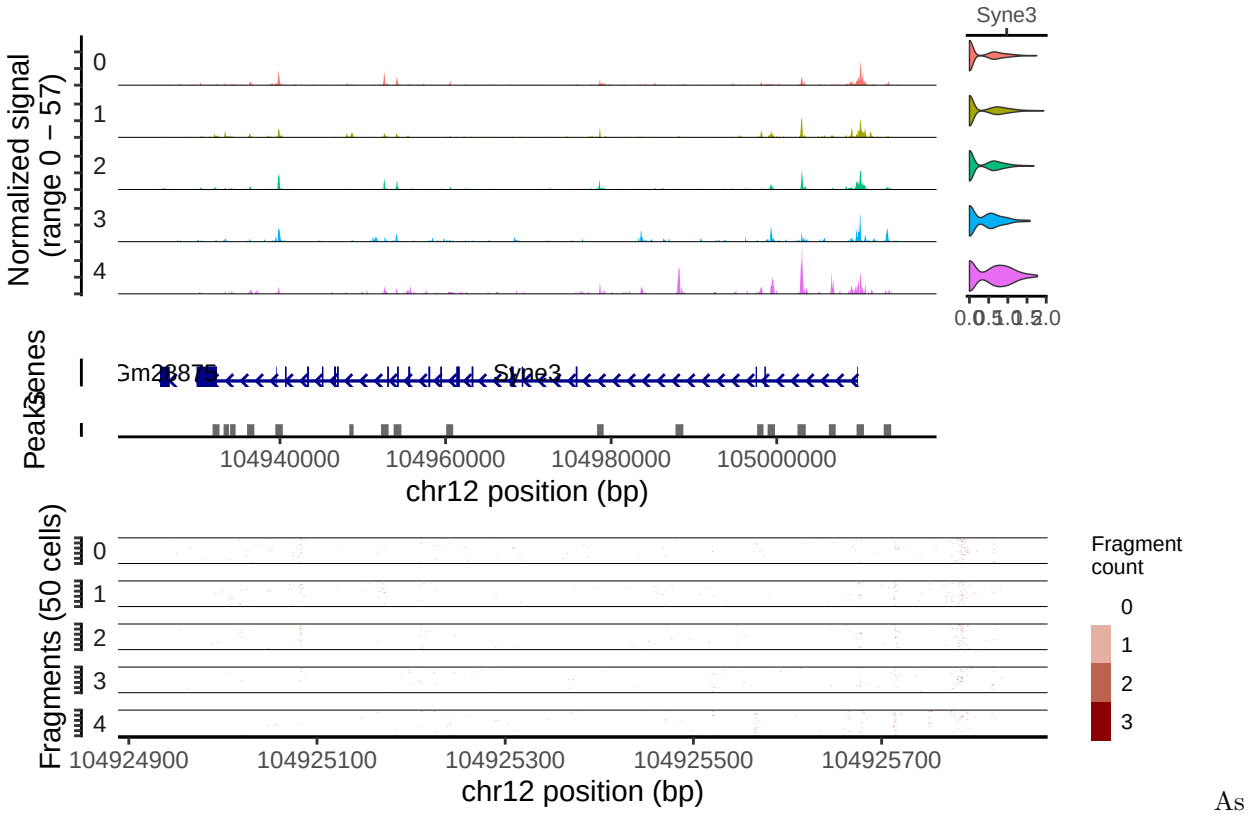
```
## Selecting by gene
```

```
for (gene in genes_top) {

  visualizeSingleGene(obj = seu.s, geneName = gene, dimReductionName = "ATAC_snn_res.0.5")

}
```




```
## Warning: Removed 8 rows containing missing values (geom_segment).
```





As you can see from the output, we here plotted the top 3 genes for a particular cluster, feel free to use our new function to plot any gene you want, or to plot the top genes from a different cluster!

Save object to disk

We reached the end of the vignette. We again save our updated `Seurat` object to disk with a new name, in analogy to what we did in the other vignettes.

```
saveRDS(seu.s, file = paste0(outFolder,"obj.filt.geneActivity.rds"))
```

Further reading

We will add additional references related to the content of this vignette soon, stay tuned!

Session info

It is good practice to print the so-called session info at the end of an R script, which prints all loaded libraries, their versions etc. This can be helpful for reproducibility and recapitulating which package versions have been used to produce the results obtained above.

```
sessionInfo()
```

```
## R version 4.1.2 (2021-11-01)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.4 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.9.0
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0
##
```

```
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_GB.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_GB.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_GB.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_GB.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] clustermole_1.1.0  patchwork_1.1.1    forcats_0.5.1      stringr_1.4.0
##  [5] dplyr_1.0.7        purrr_0.3.4        readr_2.1.1        tidyr_1.1.4
##  [9] tibble_3.1.6       ggplot2_3.3.5      tidyverse_1.3.1    sp_1.4-7
## [13] SeuratObject_4.1.0 Seurat_4.1.1       Signac_1.6.0
##
## loaded via a namespace (and not attached):
##   [1] utf8_1.2.2                  reticulate_1.24
##   [3] tidyselect_1.1.1            RSQLite_2.2.9
##   [5] AnnotationDbi_1.56.2        htmlwidgets_1.5.4
##   [7] grid_4.1.2                  docopt_0.7.1
##   [9] BiocParallel_1.28.3         Rtsne_0.16
##  [11] ScaledMatrix_1.2.0          munsell_0.5.0
##  [13] codetools_0.2-18            ica_1.0-2
##  [15] future_1.25.0               miniUI_0.1.1.1
##  [17] withr_2.4.3                 spatstat.random_2.2-0
##  [19] colorspace_2.0-2            progressr_0.10.0
##  [21] Biobase_2.54.0              highr_0.9
##  [23] knitr_1.37                  rstudioapi_0.13
##  [25] SingleCellExperiment_1.16.0 stats4_4.1.2
##  [27] ROCR_1.0-11                 tensor_1.5
##  [29] listenv_0.8.0               labeling_0.4.2
##  [31] MatrixGenerics_1.6.0        slam_0.1-50
##  [33] GenomeInfoDbData_1.2.7      polyclip_1.10-0
##  [35] bit64_4.0.5                 farver_2.1.0
##  [37] rhdf5_2.38.1                parallelly_1.31.1
##  [39] vctrs_0.3.8                 generics_0.1.1
##  [41] xfun_0.29                   lsa_0.73.2
##  [43] ggseqlogo_0.1               R6_2.5.1
##  [45] GenomeInfoDb_1.30.0         rsvd_1.0.5
##  [47] locfit_1.5-9.4              rhdf5filters_1.6.0
##  [49] DelayedArray_0.20.0         cachem_1.0.6
##  [51] bitops_1.0-7                spatstat.utils_2.3-0
##  [53] assertthat_0.2.1            promises_1.2.0.1
##  [55] scales_1.1.1                rgeos_0.5-9
##  [57] gtable_0.3.0                beachmat_2.10.0
##  [59] globals_0.14.0              goftest_1.2-3
##  [61] rlang_0.4.12                RcppRoll_0.3.0
##  [63] splines_4.1.2               lazyeval_0.2.2
##  [65] spatstat.geom_2.4-0         broom_0.7.11
##  [67] yaml_2.2.1                  reshape2_1.4.4
##  [69] abind_1.4-5                 modelr_0.1.8
##  [71] backports_1.4.1             httpuv_1.6.5
```

```
##  [73] tools_4.1.2                ellipsis_0.3.2
##  [75] spatstat.core_2.4-2        RColorBrewer_1.1-2
##  [77] BiocGenerics_0.40.0        ggridges_0.5.3
##  [79] Rcpp_1.0.8                 plyr_1.8.6
##  [81] sparseMatrixStats_1.6.0    zlibbioc_1.40.0
##  [83] RCurl_1.98-1.5             rpart_4.1-15
##  [85] deldir_1.0-6               pbapply_1.5-0
##  [87] cowplot_1.1.1              S4Vectors_0.32.3
##  [89] zoo_1.8-10                 SummarizedExperiment_1.24.0
##  [91] haven_2.4.3                ggrepel_0.9.1
##  [93] cluster_2.1.2              fs_1.5.2
##  [95] magrittr_2.0.1             data.table_1.14.2
##  [97] scattermore_0.8            lmtest_0.9-40
##  [99] reprex_2.0.1               RANN_2.6.1
## [101] SnowballC_0.7.0            fitdistrplus_1.1-8
## [103] matrixStats_0.61.0         GSVA_1.42.0
## [105] hms_1.1.1                  mime_0.12
## [107] evaluate_0.14              xtable_1.8-4
## [109] XML_3.99-0.8               sparsesvd_0.2
## [111] readxl_1.3.1               IRanges_2.28.0
## [113] gridExtra_2.3              compiler_4.1.2
## [115] KernSmooth_2.23-20         crayon_1.4.2
## [117] htmltools_0.5.2            mgcv_1.8-38
## [119] later_1.3.0                tzdb_0.2.0
## [121] lubridate_1.8.0            DBI_1.1.2
## [123] tweenr_1.0.2               dbplyr_2.1.1
## [125] MASS_7.3-54                Matrix_1.4-0
## [127] cli_3.1.0                  parallel_4.1.2
## [129] igraph_1.2.11              GenomicRanges_1.46.1
## [131] pkgconfig_2.0.3            plotly_4.10.0
## [133] spatstat.sparse_2.1-1      xml2_1.3.3
## [135] annotate_1.72.0            XVector_0.34.0
## [137] rvest_1.0.2                digest_0.6.29
## [139] sctransform_0.3.3          RcppAnnoy_0.0.19
## [141] graph_1.72.0               spatstat.data_2.2-0
## [143] Biostrings_2.62.0          rmarkdown_2.11
## [145] cellranger_1.1.0           leiden_0.3.10
## [147] fastmatch_1.1-3            edgeR_3.36.0
## [149] uwot_0.1.11                DelayedMatrixStats_1.16.0
## [151] GSEABase_1.56.0            shiny_1.7.1
## [153] Rsamtools_2.10.0           lifecycle_1.0.1
## [155] nlme_3.1-153               jsonlite_1.7.2
## [157] Rhdf5lib_1.16.0            limma_3.50.0
## [159] viridisLite_0.4.0          fansi_1.0.0
## [161] pillar_1.6.4               lattice_0.20-45
## [163] KEGGREST_1.34.0            fastmap_1.1.0
## [165] httr_1.4.2                 survival_3.2-13
## [167] glue_1.6.0                 qlcMatrix_0.9.7
## [169] png_0.1-7                  bit_4.0.4
## [171] HDF5Array_1.22.1           ggforce_0.3.3
## [173] stringi_1.7.6              blob_1.2.2
## [175] singscore_1.14.0           BiocSingular_1.10.0
## [177] memoise_2.0.1              irlba_2.3.5
## [179] future.apply_1.9.0
```