scATAC-seq analysis of adult mouse brain

Manisha Barse

```
library(Signac)
library(Seurat)
library(EnsDb.Mmusculus.v79)
library(ggplot2)
library(patchwork)
set.seed(1234)
```

Pre-processing workflow

```
counts <- Read10X_h5("atac_v1_adult_brain_fresh_5k_filtered_peak_bc_matrix.h5")
```

```
#counts
#157797 x 5337 sparse Matrix of class "dgCMatrix"
```

```
metadata <- read.csv(
  file = "atac_v1_adult_brain_fresh_5k_singlecell.csv",
  header = TRUE,
  row.names = 1
)
```

```
brain_assay <- CreateChromatinAssay(
  counts = counts,
  sep = c(":", "-"),
  genome = "mm10",
  fragments = 'atac_v1_adult_brain_fresh_5k_fragments.tsv.gz',
  min.cells = 1
)
```

```
## Computing hash
```

```
brain <- CreateSeuratObject(
  counts = brain_assay,
  assay = 'peaks',
  project = 'ATAC',
  meta.data = metadata
)
```

```
brain
```

```
## An object of class Seurat
## 157203 features across 5337 samples within 1 assay
## Active assay: peaks (157203 features, 0 variable features)
##  2 layers present: counts, data
```

By printing the assay we can see some of the additional information that can be contained in the ChromatinAssay, including motif information, gene annotations, and genome information.

```
brain[['peaks']]
```

```
## ChromatinAssay data with 157203 features for 5337 cells
## Variable features: 0
## Genome: mm10
## Annotation present: FALSE
## Motifs present: FALSE
## Fragment files: 1
```

Applying granges() to SeuratObject, to see the genomic ranges associated with each feature in the object.

```
granges(brain)
```

```
## GRanges object with 157203 ranges and 0 metadata columns:
##            seqnames            ranges strand
##               <Rle>         <IRanges>  <Rle>
##        [1]     chr1   3094708-3095552      *
##        [2]     chr1   3119414-3121782      *
##        [3]     chr1   3204809-3205178      *
##        [4]     chr1   3217330-3217359      *
##        [5]     chr1   3228123-3228463      *
##        ...      ...               ...    ...
##   [157199]     chrY 90761049-90762169      *
##   [157200]     chrY 90800417-90800831      *
##   [157201]     chrY 90804515-90805440      *
##   [157202]     chrY 90808545-90809111      *
##   [157203]     chrY 90810741-90810960      *
##   -------
##   seqinfo: 21 sequences from an unspecified genome; no seqlengths
```

Next, adding gene annotations to the brain object for the mouse genome. This will allow downstream functions to pull the gene annotation information directly from the object.

```
# extract gene annotations from EnsDb
annotations <- GetGRangesFromEnsDb(ensdb = EnsDb.Mmusculus.v79)
```

```
# change to UCSC style since the data was mapped to hg19
seqlevels(annotations) <- paste0('chr', seqlevels(annotations))
genome(annotations) <- "mm10"
```

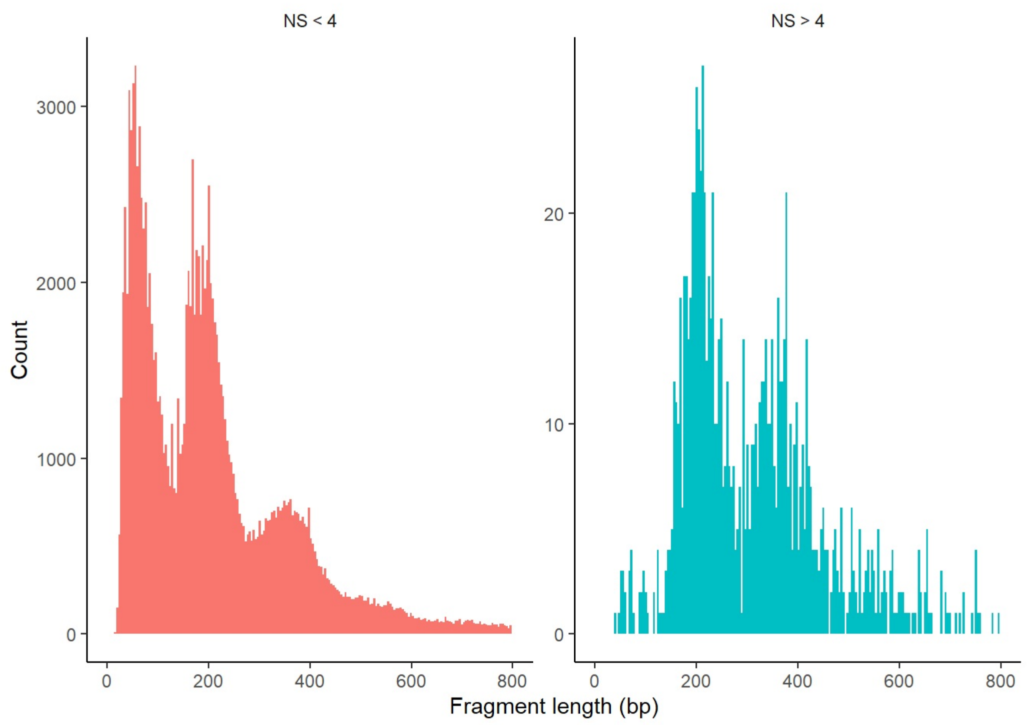
```
# add the gene information to the object
Annotation(brain) <- annotations
```

Computing QC Metrics

```
brain <- NucleosomeSignal(object = brain)
```

We can look at the fragment length periodicity for all the cells, and group by cells with high or low nucleosomal signal strength. You can see that cells which are outliers for the mononucleosomal or nucleosome-free ratio have different banding patterns. The remaining cells exhibit a pattern that is typical for a successful ATAC-seq experiment.

```
brain$nucleosome_group <- ifelse(brain$nucleosome_signal > 4, 'NS > 4', 'NS < 4')
FragmentHistogram(object = brain, group.by = 'nucleosome_group', region = 'chr1-1-10000000')
```



The enrichment of Tn5 integration events at transcriptional start sites (TSSs) can also be an important quality control metric to assess the targeting of Tn5 in ATAC-seq experiments. As per the ENCODE consortium, a TSS enrichment score is defined as the number of Tn5 integration site around the TSS normalized to the number of Tn5 integration sites in flanking regions.

```
brain <- TSSEnrichment(brain, fast = FALSE)
```

```
## Extracting TSS positions
```

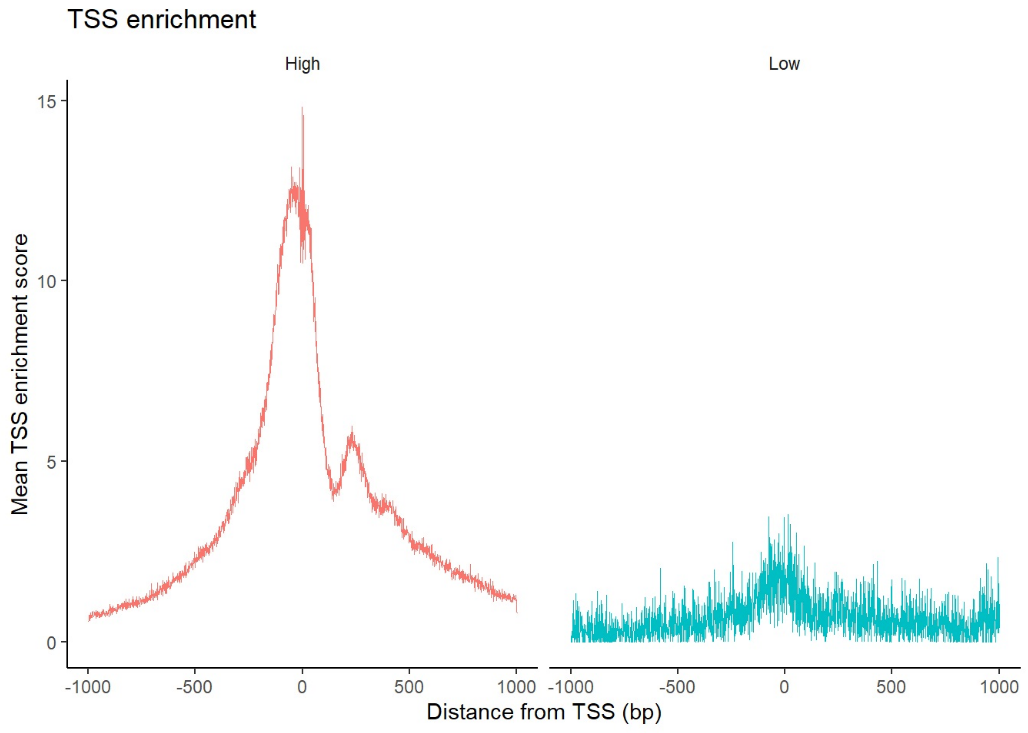
```
## Finding + strand cut sites
```

```
## Finding - strand cut sites
```

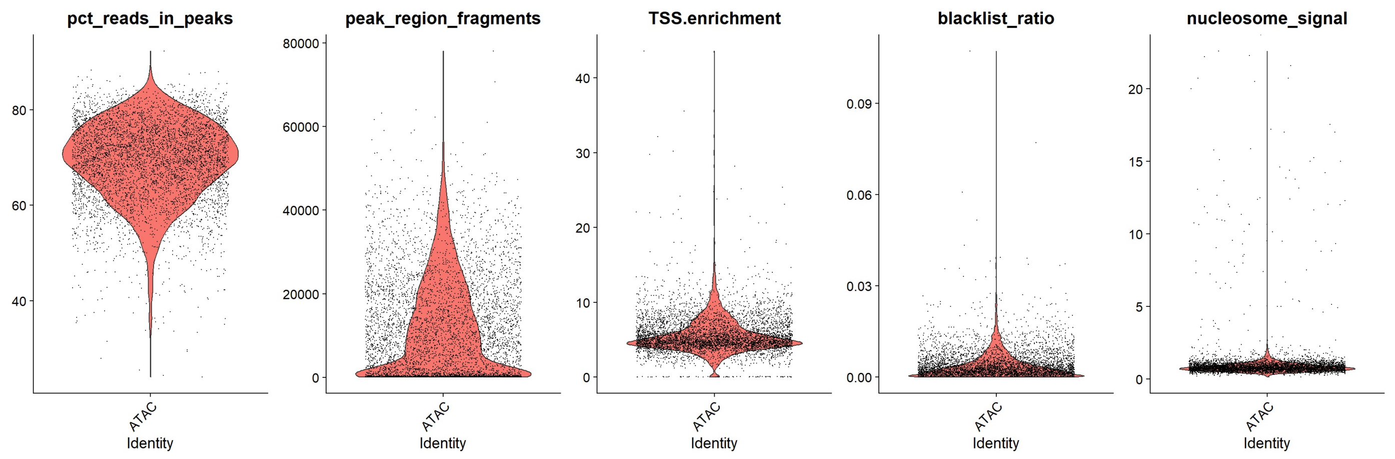
```
## Computing mean insertion frequency in flanking regions
```

```
## Normalizing TSS score
```

```
brain$high.tss <- ifelse(brain$TSS.enrichment > 2, 'High', 'Low')
TSSPlot(brain, group.by = 'high.tss') + NoLegend()
```



```
brain$pct_reads_in_peaks <- brain$peak_region_fragments / brain$passed_filters * 100
brain$blacklist_ratio <- brain$blacklist_region_fragments / brain$peak_region_fragments

VlnPlot(
  object = brain,
  features = c('pct_reads_in_peaks', 'peak_region_fragments',
               'TSS.enrichment', 'blacklist_ratio', 'nucleosome_signal'),
  pt.size = 0.1,
  ncol = 5
)
```



Let us remove cells that are outliers for these QC metrics.

```
brain <- subset(
  x = brain,
  subset = peak_region_fragments > 3000 &
    peak_region_fragments < 100000 &
    pct_reads_in_peaks > 40 &
    blacklist_ratio < 0.025 &
    nucleosome_signal < 4 &
    TSS.enrichment > 2
)
```

```
brain
```

```
## An object of class Seurat 
## 157203 features across 3512 samples within 1 assay 
## Active assay: peaks (157203 features, 0 variable features)
##  2 layers present: counts, data
```

Normalization and linear dimensional reduction

```
# Running TF-IDF normalization on the brain dataset to transform the count matrix into a term frequency-inverse d
ocument frequency (TF-IDF) matrix, which weights the features by their importance.
brain <- RunTFIDF(brain)
```

```
## Performing TF-IDF normalization
```

```
## Warning in RunTFIDF.default(object = GetAssayData(object = object, slot =
## "counts"), : Some features contain 0 total counts
```

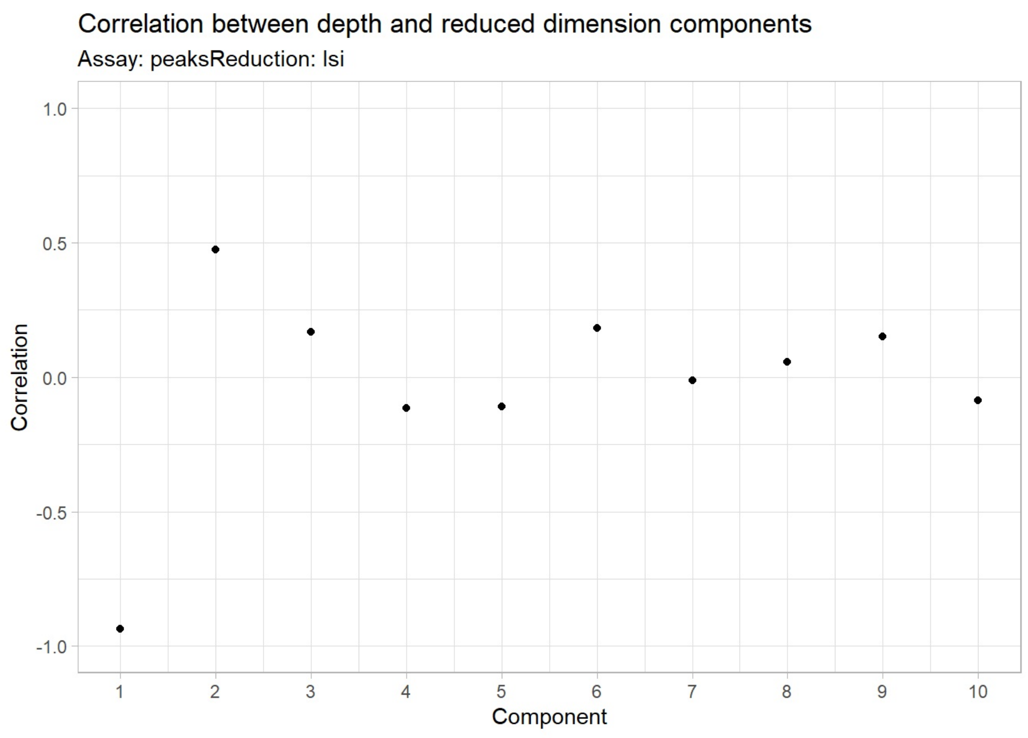
```
# Finding top features based on TF-IDF scores above the minimum cutoff 'q0', ensuring that only informative featu
res are retained for downstream analysis.
brain <- FindTopFeatures(brain, min.cutoff = 'q0')

# Running singular value decomposition (SVD) on the TF-IDF normalized data to perform linear dimensional reductio
n, capturing the main patterns of variation in the dataset.
brain <- RunSVD(object = brain)
```

```
## Running SVD
```

```
## Scaling cell embeddings
```

To assess if the first LSI component captures sequencing depth (technical variation) instead of biological variation, we can utilize the DepthCor() function to evaluate the correlation between each LSI component and sequencing depth. If a strong correlation is observed with sequencing depth, it suggests that the component should be removed from downstream analysis.

```
DepthCor(brain)
```



Here we observe a strong correlation between the first Latent Semantic Indexing (LSI) component and the total number of counts for the cell. Hence, we will proceed with downstream analysis excluding this component.

Non-linear dimension reduction and clustering

Now that the cells are embedded in a low-dimensional space, we can use methods commonly applied for the analysis of scRNA-seq data to perform graph-based clustering and non-linear dimension reduction for visualization. These functions, `RunUMAP()` , `FindNeighbors()` , and `FindClusters()` , are all provided by the Seurat package.

- RunUMAP function performs uniform manifold approximation and projection (UMAP) on the specified Seurat object using the latent semantic indexing (LSI) reduction method, retaining dimensions.
- FindNeighbors function identifies the nearest neighbors for each cell in the Seurat object based on the specified reduction ('lsi') and dimensions (2 to 30) obtained from the LSI reduction, computes a shared nearest neighbor (SNN) graph to capture local cell similarities.
- Finding clusters of cells based on their similarities in the lower-dimensional UMAP space.

```
# Perform dimensionality reduction using UMAP on the 'lsi' space with dimensions 2 to 30.
brain <- RunUMAP(
  object = brain,
  reduction = 'lsi',
  dims = 2:30
)
```

```
## Warning: The default method for RunUMAP has changed from calling Python UMAP via reticulate to the R-native UW
OT using the cosine metric
## To use Python UMAP via reticulate, set umap.method to 'umap-learn' and metric to 'correlation'
## This message will be shown once per session
```

```
## 11:34:53 UMAP embedding parameters a = 0.9922 b = 1.112
```

```
## Found more than one class "dist" in cache; using the first, from namespace 'BiocGenerics'
```

```
## Also defined by 'spam'
```

```
## 11:34:53 Read 3512 rows and found 29 numeric columns
```

```
## 11:34:53 Using Annoy for neighbor search, n_neighbors = 30
```

```
## Found more than one class "dist" in cache; using the first, from namespace 'BiocGenerics'
```

```
## Also defined by 'spam'
```

```
## 11:34:53 Building Annoy index with metric = cosine, n_trees = 50
## 0%   10   20   30   40   50   60   70   80   90   100%
```

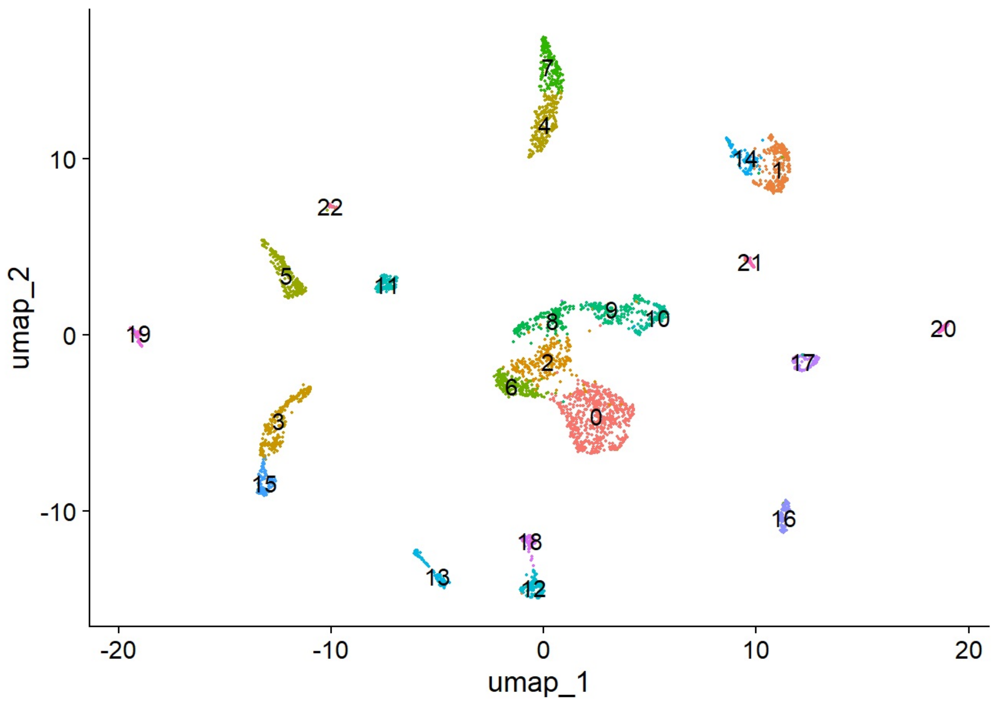
```
## [----|----|----|----|----|----|----|----|----|----|
```

```
## **************************************************|
## 11:34:55 Writing NN index file to temp file C:\Users\manis\AppData\Local\Temp\RtmpW04i0z\file35083ca9335b
## 11:34:55 Searching Annoy index using 1 thread, search_k = 3000
## 11:34:59 Annoy recall = 100%
## 11:35:19 Commencing smooth kNN distance calibration using 1 thread with target n_neighbors = 30
## 11:35:24 Initializing from normalized Laplacian + noise (using RSpectra)
## 11:35:25 Commencing optimization for 500 epochs, with 130092 positive edges
## 11:35:53 Optimization finished
```

```
# Find nearest neighbors in the 'lsi' space with dimensions 2 to 30.
brain <- FindNeighbors(
  object = brain,
  reduction = 'lsi',
  dims = 2:30
)
```

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

```
# Find clusters in the UMAP space. Adjusting the resolution parameter to control the granularity of the identifie
d cell clusters.
brain <- FindClusters(
  object = brain,
  algorithm = 3,
  resolution = 1.2,
  verbose = FALSE
)

# Generate a UMAP-based visualization of the data with cluster labels.
DimPlot(object = brain, label = TRUE) + NoLegend()
```



## Create a gene activity matrix

Gene activity refers to the level of expression or activity of genes across different cells in the dataset.

```
# Compute gene activities from the Seurat object 'brain'
gene.activities <- GeneActivity(brain)
```

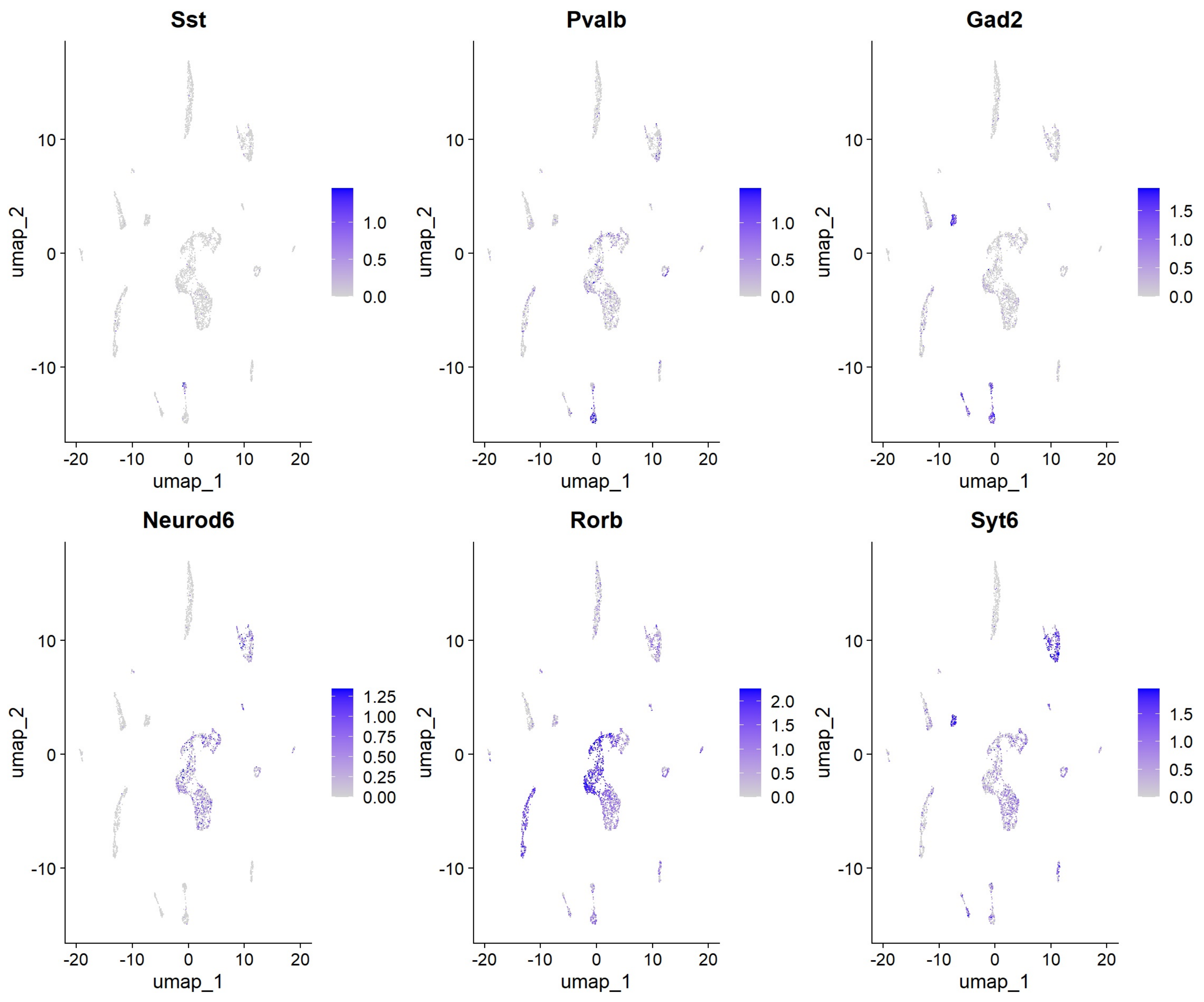
Add the gene activity matrix to the Seurat object as a new assay named ‘RNA’. This allows for easy access and manipulation of the gene activity data within the Seurat framework.

```
brain[['RNA']] <- CreateAssayObject(counts = gene.activities)
```

```
# Normalize the gene activity data using the 'LogNormalize' method
brain <- NormalizeData(
  object = brain,
  assay = 'RNA',
  normalization.method = 'LogNormalize',
  scale.factor = median(brain$nCount_RNA)
)
```

Normalization is performed to remove technical variations and ensure that the gene activity values are comparable across cells. The scale factor used for normalization is set to the median of the total counts of RNA in the dataset.

```
# Set the default assay of the Seurat object 'brain' to 'RNA'
DefaultAssay(brain) <- 'RNA'

# Generate a feature plot to visualize gene expression of specific genes
FeaturePlot(
  object = brain,
  features = c('Sst','Pvalb',"Gad2","Neurod6","Rorb","Syt6"),
  pt.size = 0.1,
  max.cutoff = 'q95',
  ncol = 3
)
```



Integrating scRNA-seq data with scATAC-seq data

We can classify cells based on an scRNA-seq experiment from the adult mouse brain, to help interpret the scATAC-seq data. We utilize methods for cross-modality integration and label transfer, as described in the Seurat vignette. * Original methods described in the publication: DOI: 10.1016/j.cell.2019.05.031 (https://doi.org/10.1016/j.cell.2019.05.031) * More in-depth tutorial available here (https://satijalab.org/seurat/v3.0/atacseq_integration_vignette.html)

- Raw data for the scRNA-seq experiment can be downloaded from the Allen Institute website (http://celltypes.brain-map.org/api/v2/well_known_file_download/694413985)

- Code used to construct the Seurat object can be found on GitHub (https://github.com/satijalab/Integration2019/blob/master/preprocessing_scripts/allen_brain.R)
- Alternatively, you can download the pre-processed Seurat object from here (https://signac-objects.s3.amazonaws.com/allen_brain.rds)

```
# Load the pre-processed scRNA-seq data from the file 'allen_brain.rds' and update the Seurat object.
allen_rna <- readRDS("allen_brain.rds")
allen_rna <- UpdateSeuratObject(allen_rna)
```

```
## Validating object structure
```

```
## Updating object slots
```

```
## Ensuring keys are in the proper structure
```

```
## Updating matrix keys for DimReduc 'pca'
```

```
## Updating matrix keys for DimReduc 'umap'
```

```
## Warning: Assay RNA changing from Assay to Assay
```

```
## Warning: DimReduc pca changing from DimReduc to DimReduc
```

```
## Warning: DimReduc umap changing from DimReduc to DimReduc
```

```
## Ensuring keys are in the proper structure
```

```
## Ensuring feature names don't have underscores or pipes
```

```
## Updating slots in RNA
```

```
## Updating slots in pca
```

```
## Updating slots in umap
```

```
## Setting umap DimReduc to global
```

```
## Setting assay used for NormalizeData.RNA to RNA
```

```
## Setting assay used for FindVariableFeatures.RNA to RNA
```

```
## Setting assay used for ScaleData.RNA to RNA
```

```
## Setting assay used for RunPCA.RNA to RNA
```

```
## Setting assay used for RunUMAP.RNA.pca to RNA
```

```
## Validating object structure for Assay 'RNA'
```

```
## Validating object structure for DimReduc 'pca'
```

```
## Validating object structure for DimReduc 'umap'
```

```
## Object representation is consistent with the most current Seurat version
```

```
allen_rna <- FindVariableFeatures(
  object = allen_rna,
  nfeatures = 5000
)
```

Identify the top 5000 variable features in the scRNA-seq data.

```
# Find anchors between the scRNA-seq and scATAC-seq datasets using canonical correlation analysis (CCA) on dimens
ions 1 to 30.
transfer.anchors <- FindTransferAnchors(
  reference = allen_rna,
  query = brain,
  reduction = 'cca',
  dims = 1:30
)
```

```
## Running CCA
```

```
## Merging objects
```

```
## Finding neighborhoods
```

```
## Finding anchors
```

```
##  Found 6862 anchors
```

```
# Transfer labels from the scRNA-seq data to the scATAC-seq data using the identified anchors and dimensionality
reduction results from the scATAC-seq data.
predicted.labels <- TransferData(
  anchorset = transfer.anchors,
  refdata = allen_rna$subclass,
  weight.reduction = brain[['lsi']],
  dims = 2:30
)
```

```
## Finding integration vectors
```

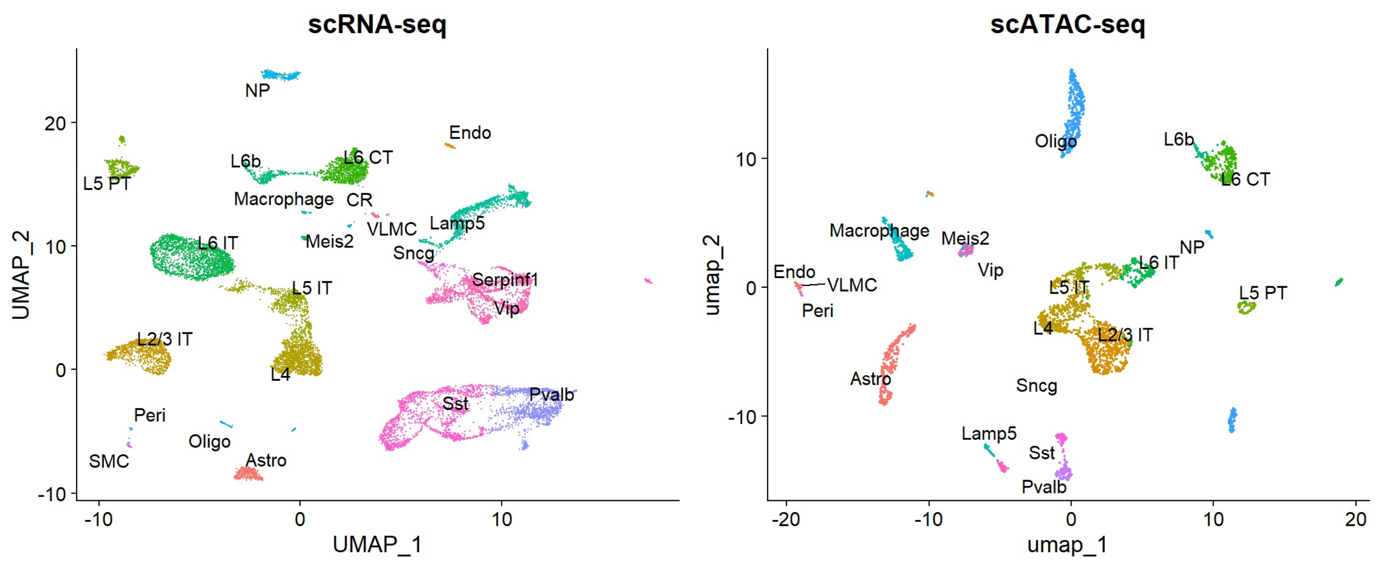
```
## Finding integration vector weights
```

```
## Predicting cell labels
```

```
# Add the predicted cell labels from the scRNA-seq data as metadata to the scATAC-seq object.
brain <- AddMetaData(object = brain, metadata = predicted.labels)
```

Create UMAP visualizations for both scRNA-seq and scATAC-seq data, with cells colored by subclass (scRNA-seq) and predicted cell labels (scATAC-seq).

```
plot1 <- DimPlot(allen_rna, group.by = 'subclass', label = TRUE, repel = TRUE) + NoLegend() + ggtitle('scRNA-seq'
)
plot2 <- DimPlot(brain, group.by = 'predicted.id', label = TRUE, repel = TRUE) + NoLegend() + ggtitle('scATAC-seq
')
plot1 + plot2
```



Integrating scRNA-seq with scATAC-seq, we validate scATAC-seq clustering using scRNA-seq cell type labels. Anchors are identified between datasets, transferring scRNA-seq labels to scATAC-seq cells. Using canonical correlation analysis (CCA), we establish cross-modality correspondences. Cell type labels from scRNA-seq are transferred to scATAC-seq. Comparing RNA-based classifications to scATAC-seq UMAP clusters ensures consistency. This validation boosts confidence in scATAC-seq cluster annotations and facilitates comprehensive biological interpretation of single-cell chromatin accessibility data.

We see consistent alignment between RNA-based classifications and the UMAP visualization, derived solely from ATAC-seq data

Find differentially accessible peaks between clusters

In this section, we identify differentially accessible regions among excitatory neurons across various cortical layers.

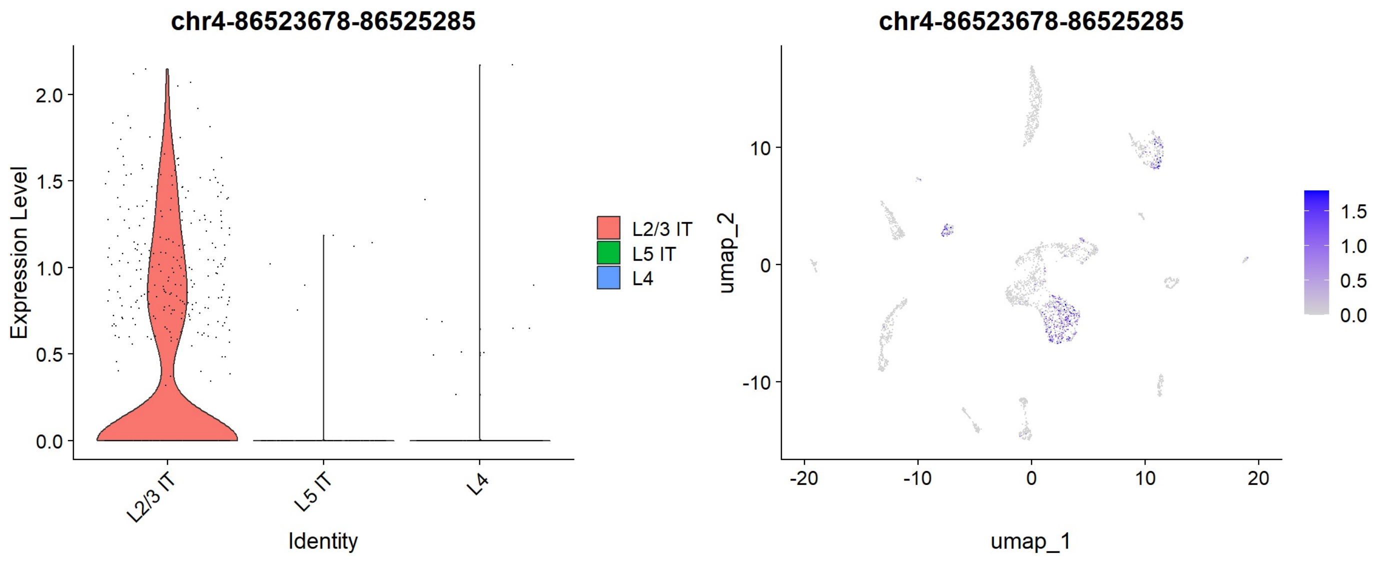
```
#Switching back to working with peaks instead of gene activities and setting cluster identities
DefaultAssay(brain) <- 'peaks'
Idents(brain) <- "predicted.id"
```

```
# Finding differentially accessible peaks between clusters of excitatory neurons in different layers of the corte
x
da_peaks <- FindMarkers(
  object = brain,
  ident.1 = c("L2/3 IT"),
  ident.2 = c("L4", "L5 IT", "L6 IT"),
  test.use = 'LR',
  latent.vars = 'nCount_peaks'
)
```

```
# Displaying the first few rows of the differentially accessible peaks table
head(da_peaks)
```

```
##                                p_val avg_log2FC pct.1 pct.2    p_val_adj
## chr4-86523678-86525285    3.266647e-69   3.691294 0.426 0.037 5.135267e-64
## chr2-118700082-118704897  8.553383e-61   2.092487 0.648 0.182 1.344617e-55
## chr15-87605281-87607659   3.864918e-55   2.450827 0.499 0.097 6.075767e-50
## chr10-107751762-107753240 1.534485e-52   1.801355 0.632 0.192 2.412257e-47
## chr4-101303935-101305131  5.949521e-51   3.427059 0.356 0.031 9.352825e-46
## chr13-69329933-69331707   1.604991e-49  -2.254722 0.140 0.435 2.523094e-44
```

```
# Creating violin and feature plots to visualize differentially accessible peaks
plot1 <- VlnPlot(
  object = brain,
  features = rownames(da_peaks)[1],
  pt.size = 0.1,
  idents = c("L4","L5 IT","L2/3 IT")
)
plot2 <- FeaturePlot(
  object = brain,
  features = rownames(da_peaks)[1],
  pt.size = 0.1,
  max.cutoff = 'q95'
)
plot1 | plot2
```



```
# Identifying the closest features to significantly differentially accessible peaks in L2/3 IT cluster
open_l23 <- rownames(da_peaks[da_peaks$avg_log2FC > 3, ])
closest_l23 <- ClosestFeature(brain, open_l23)

head(closest_l23)
```

```
##                                 tx_id gene_name            gene_id
## ENSMUST00000151481 ENSMUST00000151481   Fam154a ENSMUSG00000028492
## ENSMUST00000131864 ENSMUST00000131864   Gm12796 ENSMUSG00000085721
## ENSMUST00000139527 ENSMUST00000139527     Yipf1 ENSMUSG00000057375
## ENSMUSE00001329193 ENSMUST00000185379   Gm29414 ENSMUSG00000099392
## ENSMUSE00000514286 ENSMUST00000077353      Hmbs ENSMUSG00000032126
## ENSMUST00000161356 ENSMUST00000161356      Reln ENSMUSG00000042453
##                      gene_biotype type          closest_region
## ENSMUST00000151481 protein_coding  gap   chr4-86487920-86538964
## ENSMUST00000131864        lincRNA  gap chr4-101292521-101318425
## ENSMUST00000139527 protein_coding  cds chr4-107345009-107345191
## ENSMUSE00001329193        lincRNA exon   chr1-25026581-25026779
## ENSMUSE00000514286 protein_coding exon   chr9-44344010-44344228
## ENSMUST00000161356 protein_coding  gap   chr5-21891568-21895988
##                                query_region distance
## ENSMUST00000151481   chr4-86523678-86525285        0
## ENSMUST00000131864 chr4-101303935-101305131        0
## ENSMUST00000139527 chr4-107344435-107345145        0
## ENSMUSE00001329193   chr1-25008426-25009334    17246
## ENSMUSE00000514286   chr9-44345250-44346015     1021
## ENSMUST00000161356   chr5-21894051-21894682        0
```

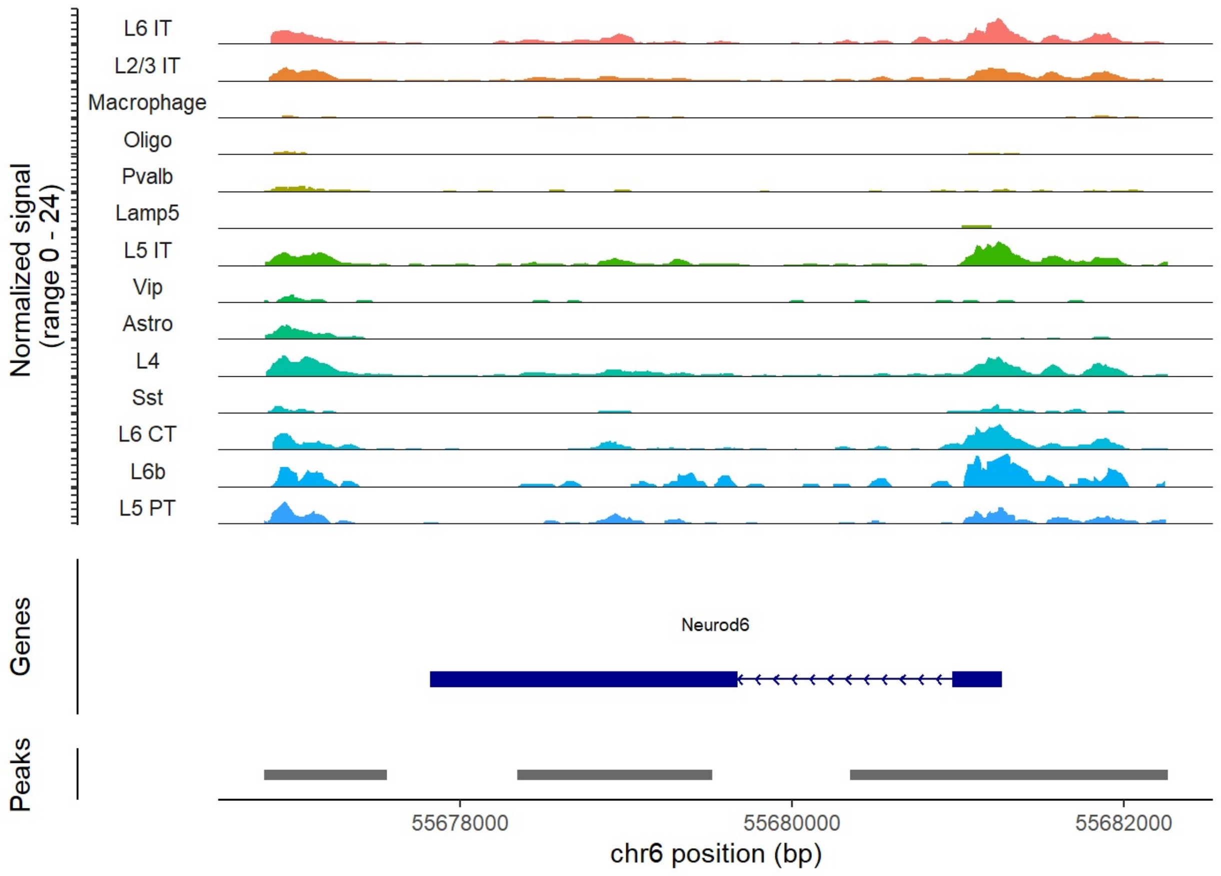
```
# Identifying the closest features to significantly differentially accessible peaks in L4, L5 IT, and L6 IT clust
ers
open_l456 <- rownames(da_peaks[da_peaks$avg_log2FC < 3, ])
closest_l456 <- ClosestFeature(brain, open_l456)

head(closest_l456)
```

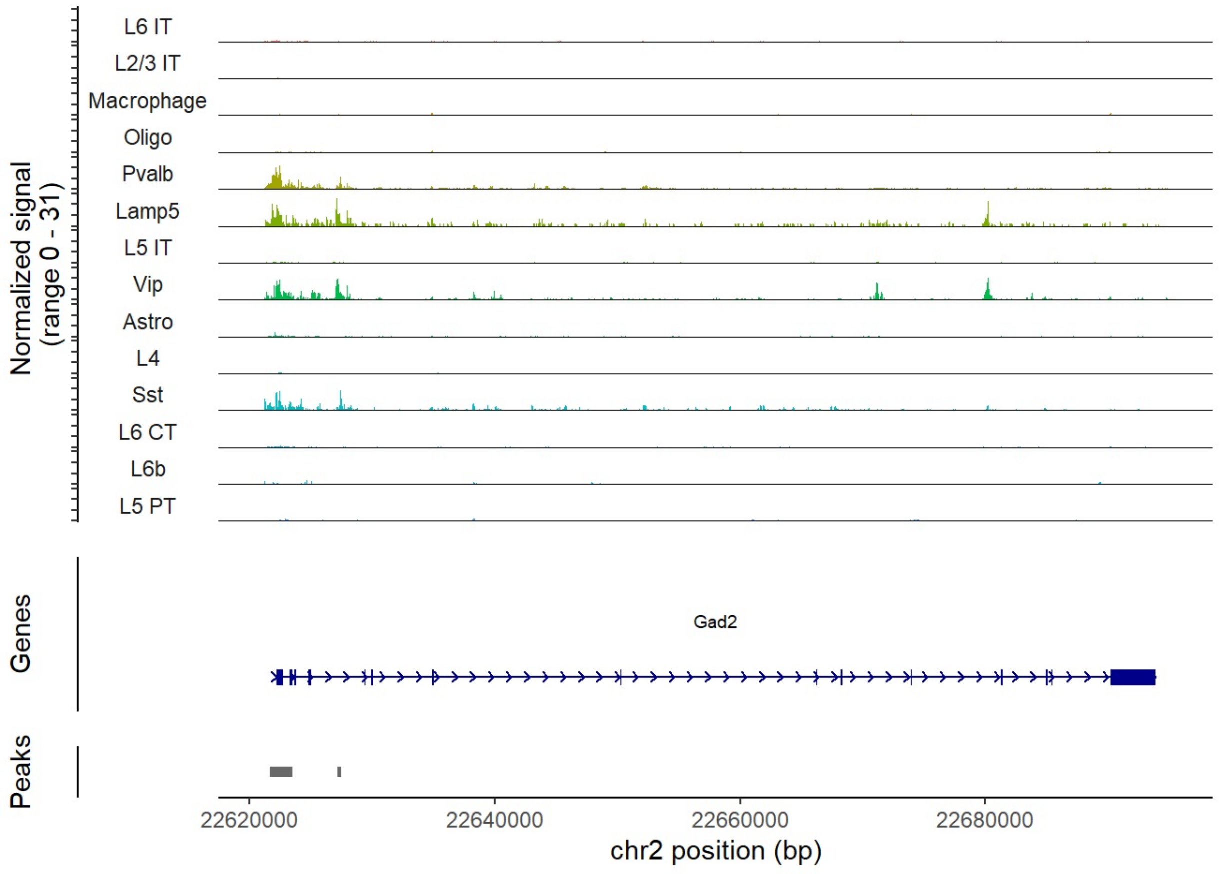
```
##                                 tx_id gene_name            gene_id
## ENSMUST00000104937 ENSMUST00000104937   Ankrd63 ENSMUSG00000078137
## ENSMUSE00000647021 ENSMUST00000068088   Fam19a5 ENSMUSG00000054863
## ENSMUST00000165341 ENSMUST00000165341     Otogl ENSMUSG00000091455
## ENSMUST00000044081 ENSMUST00000044081     Papd7 ENSMUSG00000034575
## ENSMUST00000070198 ENSMUST00000070198    Ppp3ca ENSMUSG00000028161
## ENSMUST00000084628 ENSMUST00000084628    Hs3st2 ENSMUSG00000046321
##                      gene_biotype type           closest_region
## ENSMUST00000104937 protein_coding  cds  chr2-118702266-118703438
## ENSMUSE00000647021 protein_coding exon   chr15-87625230-87625486
## ENSMUST00000165341 protein_coding  utr chr10-107762223-107762309
## ENSMUST00000044081 protein_coding  utr   chr13-69497959-69499915
## ENSMUST00000070198 protein_coding  utr  chr3-136935226-136937727
## ENSMUST00000084628 protein_coding  cds  chr7-121392730-121393214
##                                 query_region distance
## ENSMUST00000104937  chr2-118700082-118704897        0
## ENSMUSE00000647021   chr15-87605281-87607659    17570
## ENSMUST00000165341 chr10-107751762-107753240     8982
## ENSMUST00000044081   chr13-69329933-69331707   166251
## ENSMUST00000070198  chr3-137056475-137058371   118747
## ENSMUST00000084628  chr7-121391215-121395519        0
```

Plotting genomic regions

We can create coverage plots for genomic regions grouped by cluster, cell type, or any other metadata stored in the object using the CoveragePlot() function. These plots visualize the pseudo-bulk accessibility tracks, averaging signal from all cells within a group to represent DNA accessibility in a region.

```r
# Plotting coverage plots for genomic regions

# show cell types with at least 50 cells
idents.plot <- names(which(table(Idents(brain)) > 50)) # Filtering cell types with at least 50 cells

CoveragePlot(
  object = brain,
  region = c("Neurod6", "Gad2"),
  idents = idents.plot,
  extend.upstream = 1000,
  extend.downstream = 1000,
  ncol = 1
)
```





```
# Saving the Seurat object as an RDS file
saveRDS(object = brain, file = "adult_mouse_brain.rds")
```