FigureS8_Cortical excitatory neurons

Connect loom file from La Manno et al. 2020.

```
dir <- "/Users/j76630as/Documents/tTF_paper_2020/scRNAseq/"

setwd(paste0(dir, "/output/"))

## connect sc.loom file downloaded from mousebrain.org
sc.loom <- loomR::connect(filename = paste0(dir, "/input/dev_all.loom"), mode = "r+", skip.validate = TR

## Generate sc.meta file by extracting parameters from connected sc.loom file
sc.meta <- data.frame(
  sc.loom$col.attrs$Age[],
  sc.loom$col.attrs$PseudoAge[],
  sc.loom$col.attrs$Tissue[],
  sc.loom$col.attrs$PseudoTissue[],
  sc.loom$col.attrs$Class[],
  sc.loom$col.attrs$Clusters[],
  10000 / sc.loom$col.attrs$TotalUMI[],
  sc.loom$col.attrs$CellID[],
  sc.loom$col.attrs$SampleID[]
)

colnames(sc.meta) <- c("age", "pseudoage", "tissue", "pseudotissue", "class", "clusters", "normalizatio
```

Load forebrain scRNAseq data from La Manno et al. 2020

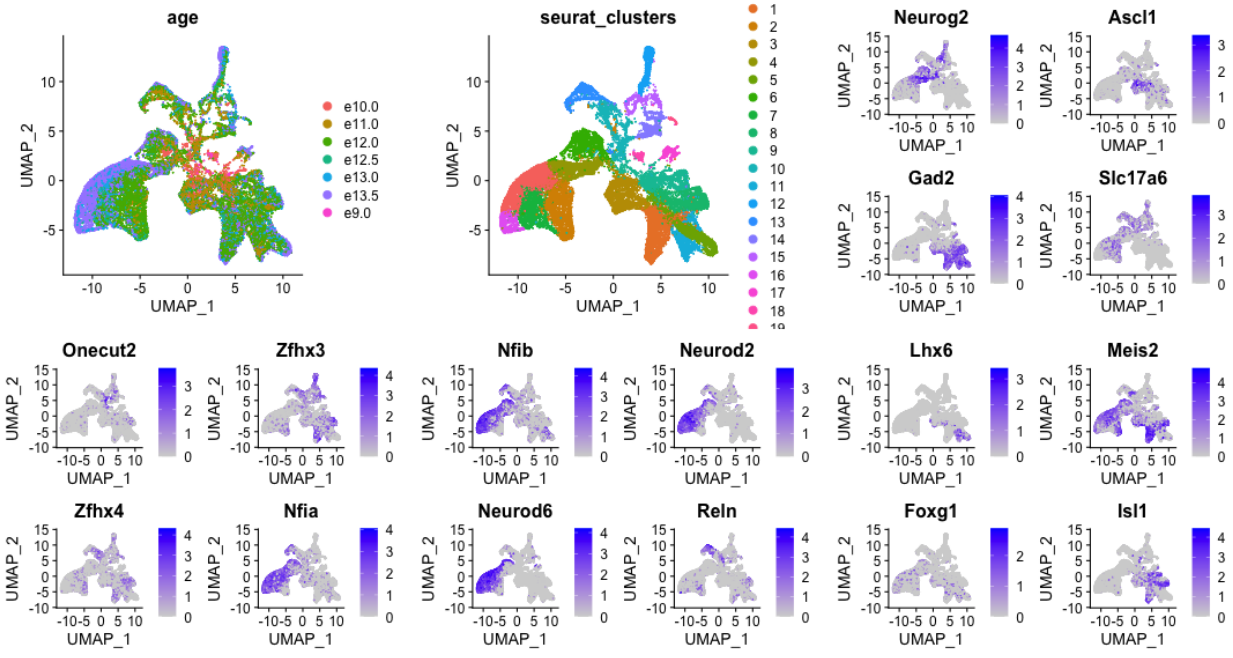
We subset the data to annotated forebrain neurons and convert the data in a Seurat object.

```
### Generate Forebrain-specific Seurat object

tissue <- "Forebrain"
celltype <- "Neuron"
timepoints <- c("e9.0", "e10.0", "e11.0", "e12.0", "e12.5", "e13.0", "e13.5", "e14.0")

tissue.id <- which(grepl(tissue, unique(sc.loom$col.attrs$Tissue[])) == TRUE)
cell.id <- intersect(
  which(sc.meta$tissue %in% unique(sc.loom$col.attrs$Tissue[])[tissue.id] & sc.meta$class == celltype),
  which(sc.meta$age %in% timepoints)
)

exp.mat <- sc.loom[["matrix"]][cell.id, ]

colnames(exp.mat) <- sc.loom$row.attrs$Gene[]
```

```
rownames(exp.mat) <- sc.meta$cellID[cell.id]

exc.seurat <- CreateSeuratObject(
  counts = t(exp.mat),
  meta.data = sc.meta[cell.id, ] %>%
    as_tibble() %>%
    tibble::column_to_rownames("cellID")
)

exc.seurat[["percent.mt"]] <- PercentageFeatureSet(exc.seurat, pattern = "^mt-")

exc.seurat <- exc.seurat %>%
  subset(subset = nFeature_RNA > 600 & nFeature_RNA < 6000 & percent.mt < 6) %>%
  NormalizeData(verbose = FALSE) %>%
  ScaleData(verbose = FALSE) %>%
  FindVariableFeatures(selection.method = "vst", verbose = FALSE) %>%
  RunPCA(npcs = 30, verbose = FALSE) %>%
  RunUMAP(reduction = "pca", dims = 1:30) %>%
  FindNeighbors(dims = 1:30) %>%
  FindClusters(resolution = 0.5)
```

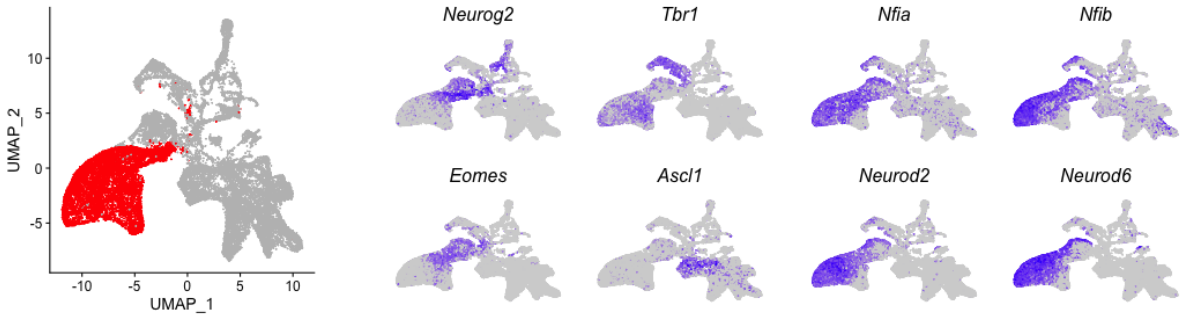
To get a general feel for the data and to identify which clusters define which neuronal populations, we plot the data on a UMAP

```
cowplot::plot_grid(DimPlot(exc.seurat, reduction = "umap", group.by = "age") + theme(aspect.ratio = 1),
  DimPlot(exc.seurat, reduction = "umap", group.by = "seurat_clusters") + theme(aspect.ratio = 1),
  FeaturePlot(exc.seurat, features = c("Neurog2", "Ascl1", "Gad2", "Slc17a6")),
  FeaturePlot(exc.seurat, features = c("Onecut2", "Zfhx3", "Zfhx4", "Nfia")),
  FeaturePlot(exc.seurat, features = c("Nfib", "Neurod2", "Neurod6", "Reln")),
  FeaturePlot(exc.seurat, features = c("Lhx6", "Meis2", "Foxg1", "Isl1")),
  nrow = 2
)
```

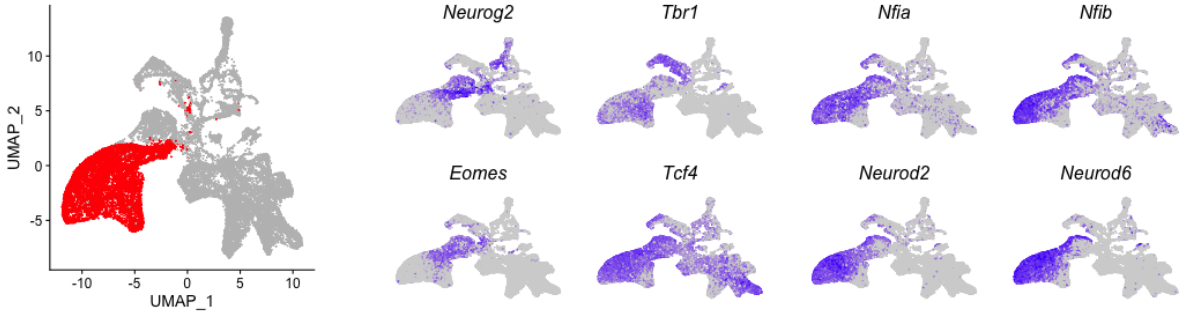


We subset the data to cortical excitatory neurons.

```
exc.neurons.clusters <- c(0, 2, 4, 7, 16)

exc.seurat$excitatory_neurons <- 0
exc.seurat$excitatory_neurons[which(exc.seurat$seurat_clusters %in% exc.neurons.clusters)] <- 1

overview <- cowplot::plot_grid(DimPlot(exc.seurat, reduction = "umap", group.by = "excitatory_neurons",
  NoLegend() +
  theme(aspect.ratio = 1, plot.title = element_blank()),
FeaturePlot(exc.seurat, features = c("Neurog2", "Tbr1", "Eomes", "Ascl1")) & NoLegend() & NoAxes() & the
FeaturePlot(exc.seurat, features = c("Nfia", "Nfib", "Neurod2", "Neurod6")) & NoLegend() & NoAxes() & tl
nrow = 1
)

overview
```



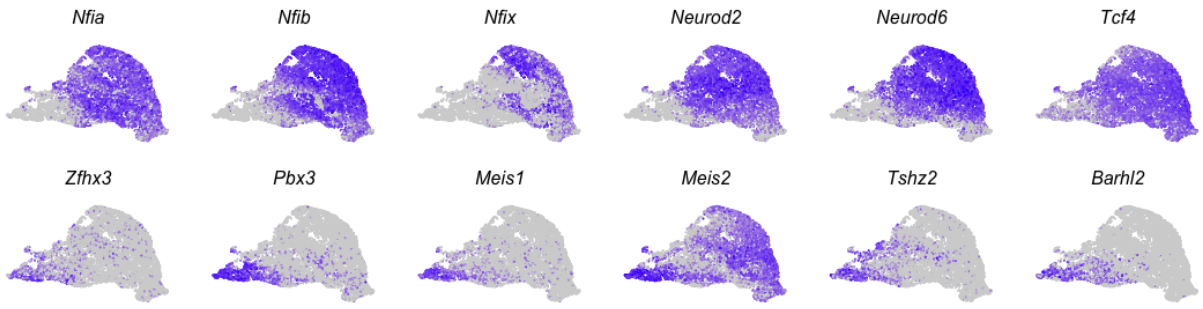

```
exc.neurons.clusters <- c(0, 2, 4, 7, 16)

exc.seurat$excitatory_neurons <- 0
exc.seurat$excitatory_neurons[which(exc.seurat$seurat_clusters %in% exc.neurons.clusters)] <- 1

overview <- cowplot::plot_grid(DimPlot(exc.seurat, reduction = "umap", group.by = "excitatory_neurons",
  NoLegend() +
  theme(aspect.ratio = 1, plot.title = element_blank()),
FeaturePlot(exc.seurat, features = c("Neurog2", "Tbr1", "Eomes", "Tcf4")) & NoLegend() & NoAxes() & ther
FeaturePlot(exc.seurat, features = c("Nfia", "Nfib", "Neurod2", "Neurod6")) & NoLegend() & NoAxes() & tl
nrow = 1
)

overview
```



```r
sub.seurat <- exc.seurat %>%
  subset(subset = seurat_clusters %in% exc.neurons.clusters) %>%
  ScaleData(verbose = FALSE) %>%
  FindVariableFeatures(selection.method = "vst", verbose = FALSE) %>%
  RunPCA(npcs = 30, verbose = FALSE) %>%
  RunUMAP(reduction = "pca", dims = 1:30) %>%
  FindNeighbors(dims = 1:30) %>%
  FindClusters(resolution = 0.5)
```

```
## 17:22:43 UMAP embedding parameters a = 0.9922 b = 1.112

## 17:22:43 Read 7615 rows and found 30 numeric columns

## 17:22:43 Using Annoy for neighbor search, n_neighbors = 30

## 17:22:43 Building Annoy index with metric = cosine, n_trees = 50

## 0%   10   20   30   40   50   60   70   80   90   100%

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

## **************************************************|
## 17:22:44 Writing NN index file to temp file /var/folders/tg/10nw964n3q129p3cb9wr0d240000gp/T//Rtmpu1
## 17:22:44 Searching Annoy index using 1 thread, search_k = 3000
## 17:22:45 Annoy recall = 100%
## 17:22:46 Commencing smooth kNN distance calibration using 1 thread
## 17:22:46 Initializing from normalized Laplacian + noise
## 17:22:46 Commencing optimization for 500 epochs, with 293908 positive edges
## 17:22:55 Optimization finished
## Computing nearest neighbor graph
## Computing SNN

## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 7615
## Number of edges: 254385
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8578
## Number of communities: 11
## Elapsed time: 0 seconds
```

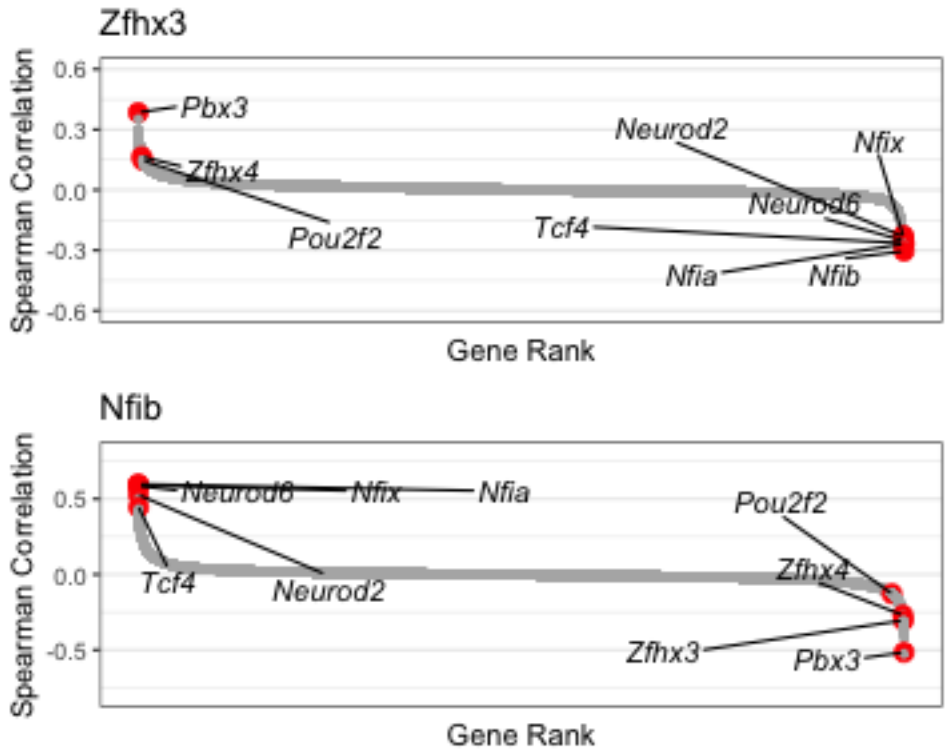
```r
umaps <- FeaturePlot(sub.seurat, features = c(
  "Nfia", "Nfib", "Nfix", "Neurod2", "Neurod6", "Tcf4",
  "Zfhx3", "Pbx3", "Meis1", "Meis2", "Tshz2", "Barhl2"
), ncol = 6) &
  NoLegend() & NoAxes() & theme(plot.title = element_text(face = "italic"))

umaps
```




```
fb.corr.rank <- cowplot::plot_grid(
  plot.correlation.ranks(
    input = sub.seurat,
    input.type = c("Seurat"),
    plot.gene = "Zfhx3",
    correlation.genes = c(
      "Pou2f2", "Zfhx3", "Zfhx4", "Pbx3", "Nfia", "Nfib", "Nfix", "Neurod2", "Neurod6", "Tcf4"
    ),
    min = -0.6,
    max = 0.6
  ),
  plot.correlation.ranks(
    input = sub.seurat,
    input.type = c("Seurat"),
    plot.gene = "Nfib",
    correlation.genes = c(
      "Pou2f2", "Zfhx3", "Zfhx4", "Pbx3", "Nfia", "Nfib", "Nfix", "Neurod2", "Neurod6", "Tcf4"
    ),
    min = -0.8,
    max = 0.8
  ),
  ncol = 1
)
```

```
## [1] "Calculate correlation ranks!"
## [1] "Calculate correlation ranks!"
```

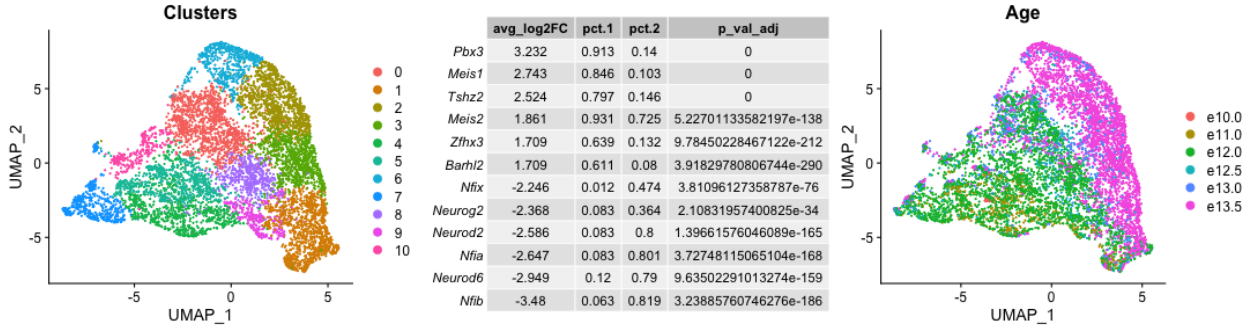
```
fb.corr.rank & theme(plot.title = element_text(face = "italic"))
```



Differential gene expression analysis to identify other TFs enriched in the Zfhx3-positive cluster 7.

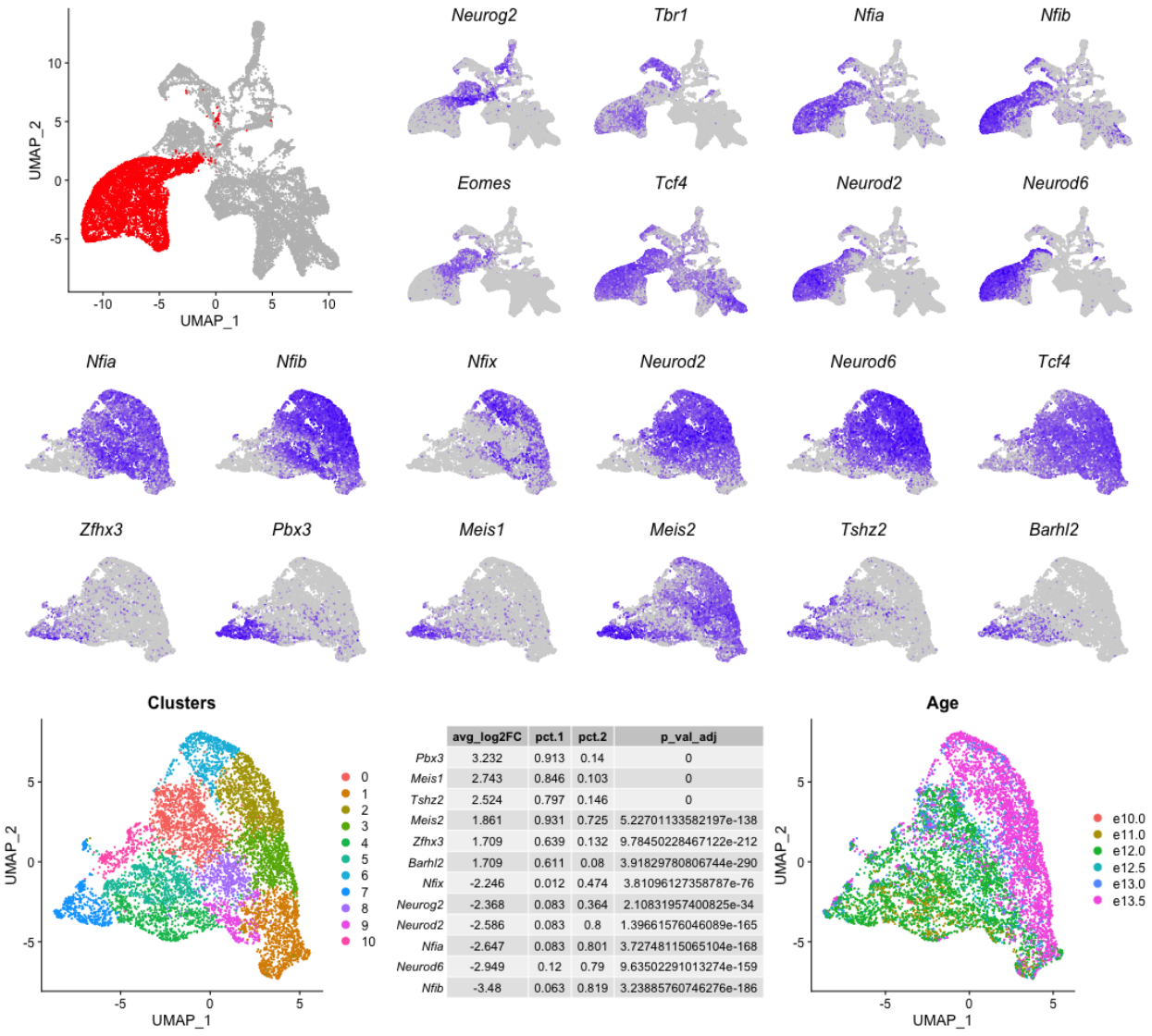
```
markers <- FindMarkers(sub.seurat, ident.1 = 7) %>%
  tibble::rownames_to_column("gene") %>%
  dplyr::filter(gene %in% TF.lst) %>%
  dplyr::arrange(desc(avg_log2FC)) %>%
  tibble::column_to_rownames("gene") %>%
  dplyr::mutate(avg_log2FC = round(avg_log2FC, digits = 3)) %>%
  dplyr::select(-p_val)
```

```
grid <- grid.arrange(DimPlot(sub.seurat, group.by = "seurat_clusters") + ggtitle("Clusters"),
  tableGrob(rbind(head(markers, n = 6), tail(markers, n = 6)), theme = ttheme_default(base_size = 11.5)),
  DimPlot(sub.seurat, group.by = "age") + ggtitle("Age"),
  ncol = 3
)
```



	avg_log2FC	pct.1	pct.2	p_val_adj
Pbx3	3.232	0.913	0.14	0
Meis1	2.743	0.846	0.103	0
Tshz2	2.524	0.797	0.146	0
Meis2	1.861	0.931	0.725	5.22701133582197e-138
Zfhx3	1.709	0.639	0.132	9.78450228467122e-212
Barhl2	1.709	0.611	0.08	3.91829780806744e-290
Nfix	-2.246	0.012	0.474	3.81096127358787e-76
Neurog2	-2.368	0.083	0.364	2.10831957400825e-34
Neurod2	-2.586	0.083	0.8	1.39661576046089e-165
Nfia	-2.647	0.083	0.801	3.72748115065104e-168
Neurod6	-2.949	0.12	0.79	9.63502291013274e-159
Nfib	-3.48	0.063	0.819	3.23885760746276e-186

```
complete <- cowplot::plot_grid(overview, umaps, grid, ncol = 1)

complete
```



	avg_log2FC	pct.1	pct.2	p_val_adj
Pbx3	3.232	0.913	0.14	0
Meis1	2.743	0.846	0.103	0
Tshz2	2.524	0.797	0.146	0
Meis2	1.861	0.931	0.725	5.22701133582197e-138
Zfhx3	1.709	0.639	0.132	9.78450228467122e-212
Barhl2	1.709	0.611	0.08	3.91829780806744e-290
Nfix	-2.246	0.012	0.474	3.81096127358787e-76
Neurog2	-2.368	0.083	0.364	2.10831957400825e-34
Neurod2	-2.586	0.083	0.8	1.39661576046089e-165
Nfia	-2.647	0.083	0.801	3.72748115065104e-168
Neurod6	-2.949	0.12	0.79	9.63502291013274e-159
Nfib	-3.48	0.063	0.819	3.23885760746276e-186

```
cowplot::ggsave2("Cortical_neurons.png", dpi = 300)
```

```
## Saving 15 x 13.5 in image
```

```
sessionInfo()
```

```
## R version 4.0.4 (2021-02-15)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur 10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
```

```
## LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_GB.UTF-8/en_GB.UTF-8/en_GB.UTF-8/C/en_GB.UTF-8/en_GB.UTF-8
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets
## [7] methods   base
##
## other attached packages:
## [1] ggrepel_0.9.1      scales_1.1.1       cowplot_1.1.1
## [4] gridExtra_2.3      pbapply_1.4-3      ggplot2_3.3.3
## [7] dplyr_1.0.5        SeuratObject_4.0.0 Seurat_4.0.1
##
## loaded via a namespace (and not attached):
##   [1] Rtsne_0.15            colorspace_2.0-0
##   [3] deldir_0.2-10         ellipsis_0.3.1
##   [5] ggridges_0.5.3        spatstat.data_2.1-0
##   [7] farver_2.1.0          leiden_0.3.7
##   [9] listenv_0.8.0         bit64_4.0.5
##  [11] RSpectra_0.16-0       fansi_0.4.2
##  [13] codetools_0.2-18      splines_4.0.4
##  [15] R.methodsS3_1.8.1     knitr_1.32
##  [17] polyclip_1.10-0       jsonlite_1.7.2
##  [19] ica_1.0-2             cluster_2.1.2
##  [21] png_0.1-7             R.oo_1.24.0
##  [23] uwot_0.1.10           shiny_1.6.0
##  [25] sctransform_0.3.2     spatstat.sparse_2.0-0
##  [27] compiler_4.0.4        httr_1.4.2
##  [29] backports_1.2.1       assertthat_0.2.1
##  [31] Matrix_1.3-2          fastmap_1.1.0
##  [33] lazyeval_0.2.2        limma_3.46.0
##  [35] later_1.1.0.1         htmltools_0.5.1.1
##  [37] tools_4.0.4           igraph_1.2.6
##  [39] gtable_0.3.0          glue_1.4.2
##  [41] RANN_2.6.1            reshape2_1.4.4
##  [43] Rcpp_1.0.6            scattermore_0.7
##  [45] styler_1.4.1          vctrs_0.3.7
##  [47] nlme_3.1-152          lmtest_0.9-38
##  [49] loomR_0.2.1.9000      xfun_0.22
##  [51] stringr_1.4.0         globals_0.14.0
##  [53] mime_0.10             miniUI_0.1.1.1
##  [55] lifecycle_1.0.0       irlba_2.3.3
##  [57] goftest_1.2-2         future_1.21.0
##  [59] MASS_7.3-53.1         zoo_1.8-9
##  [61] spatstat.core_2.1-2   promises_1.2.0.1
##  [63] spatstat.utils_2.1-0  parallel_4.0.4
##  [65] RColorBrewer_1.1-2    yaml_2.2.1
##  [67] reticulate_1.18       rpart_4.1-15
##  [69] stringi_1.5.3         highr_0.9
##  [71] rlang_0.4.10          pkgconfig_2.0.3
##  [73] matrixStats_0.58.0    evaluate_0.14
##  [75] lattice_0.20-41       ROCR_1.0-11
##  [77] purrr_0.3.4           tensor_1.5
```

```
##  [79] labeling_0.4.2        patchwork_1.1.1
##  [81] htmlwidgets_1.5.3     bit_4.0.4
##  [83] tidyselect_1.1.1      parallelly_1.25.0
##  [85] RcppAnnoy_0.0.18      plyr_1.8.6
##  [87] magrittr_2.0.1        R6_2.5.0
##  [89] generics_0.1.0        DBI_1.1.1
##  [91] pillar_1.6.0          withr_2.4.2
##  [93] mgcv_1.8-35           fitdistrplus_1.1-3
##  [95] survival_3.2-10       abind_1.4-5
##  [97] tibble_3.1.1          future.apply_1.7.0
##  [99] hdf5r_1.3.2           crayon_1.4.1
## [101] KernSmooth_2.23-18    utf8_1.2.1
## [103] spatstat.geom_2.1-0   plotly_4.9.3
## [105] rmarkdown_2.7         data.table_1.14.0
## [107] digest_0.6.27         xtable_1.8-4
## [109] R.cache_0.14.0        tidyr_1.1.3
## [111] httpuv_1.5.5          R.utils_2.10.1
## [113] munsell_0.5.0         viridisLite_0.4.0
```