Comparison between in-vivo and in-vitro scRNAseq data

Load packages and data

```r
##################################################################
## analysis of temporal genes in the in-vivo spinal cord data  ##
##################################################################

rm(list = ls()) ## clears environment

## check if all necessary packages are installed, or install them if not
dir <- dirname(rstudioapi::getSourceEditorContext()$path)

if (!dir.exists(paste0(dir, "output"))) {
  dir.create(paste(dir, "output"))
}

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

packages <- c("Biobase", "dplyr", "plyr", "scater", "heatmap.plus", "Seurat", "dendextend", "tibble", "
if (length(setdiff(packages, rownames(installed.packages()))) > 0) {
  install.packages(setdiff(packages, rownames(installed.packages())))
}

library(Biobase)
library(plyr)
library(ggplot2)
library(scater)
library(dplyr)
library(Seurat)
library(dendextend)
library(tibble)
library(pheatmap)

## load 10X in-vivo data and list of TFs

## list of mouse TFs downloaded from AnimalTFDB3.0 (http://bioinfo.life.hust.edu.cn/AnimalTFDB/#!/tf_su
TF.lst <- read.delim(paste0(dir, "/input/Mus_musculus_TF.txt"), header = TRUE)$Symbol

eset <- readRDS(paste0(dir, "/input/m_neural.rds"))
eset <- eset$expressionSet

rownames(Biobase::pData(eset)) <- gsub("-", ".", rownames(Biobase::pData(eset)))
colnames(Biobase::exprs(eset)) <- gsub("-", ".", colnames(Biobase::exprs(eset)))

## functions for converting ensemblIDs into real gene names and vice versa
```

```r
convert.to.ensemblID <- function(genes) {
  return(unlist(lapply(genes, function(x) {
    return(rownames(Biobase::fData(eset))[which(Biobase::fData(eset)[, "external_gene_name"] == x)])
  })))
}

convert.to.realname <- function(ensemblIDs, eset) {
  return(unlist(lapply(ensemblIDs, function(x) {
    return(Biobase::fData(eset)$external_gene_name[which(rownames(Biobase::fData(eset)) == x)])
  })))
}

## load data into the Seurat package
mat <- Biobase::exprs(eset)
rownames(mat) <- Biobase::fData(eset)[, "external_gene_name"]

seurat <- CreateSeuratObject(
  counts = mat,
  meta.data = Biobase::pData(eset),
  project = "MouseSpinalCordAtlas"
)

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

seurat <- 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)
```

Differential gene expression in-vivo neural progenitors

I perform differential gene expression test on progenitors from each DV domain (excluding dp6) from each embryonic day. Genes are counted as differentially expressed if they come back as differentially expressed for 8 out of the 10 analyzed progenitor domains.

```r
## analyze genes that are differentially expressed between progenitors from different days
domains.p <- c("dp1", "dp2", "dp3", "dp4", "dp5", "p0", "p1", "p2", "pMN", "p3") ## define progenitor d

### runs subclustering on progenitor domains
markers.age <- lapply(domains.p, function(x, threshold = 0.001) {
  print(paste0("Subclustering domain ", x))

  celllist <- rownames(Biobase::pData(eset))[which(Biobase::pData(eset)$Type_step2 == x)]

  seurat.sub <- subset(seurat, cells = celllist) %>%
    FindVariableFeatures(selection.method = "vst", verbose = FALSE) %>%
    ScaleData(verbose = FALSE) %>%
    RunPCA(npcs = 30, verbose = FALSE) %>%
    RunUMAP(reduction = "pca", dims = 1:30)
  Idents(seurat.sub) <- "timepoint"
  age.markers <- FindAllMarkers(seurat.sub, only.pos = TRUE, min.pct = 0.25, logfc.threshold = 0.25, ba

  return(age.markers)
})
```

```
## [1] "Subclustering domain dp1"
## [1] "Subclustering domain dp2"
## [1] "Subclustering domain dp3"
## [1] "Subclustering domain dp4"
## [1] "Subclustering domain dp5"
## [1] "Subclustering domain p0"
## [1] "Subclustering domain p1"
## [1] "Subclustering domain p2"
## [1] "Subclustering domain pMN"
## [1] "Subclustering domain p3"
```

```
### limit to to genes identified as differentially expressed in multiple domains

num.domains <- 7 ### min 8 domains

age.markers <- table(unlist(lapply(markers.age, function(x) {
  return(unique(x[, "gene"]))
}))) %>%
  as.data.frame() %>%
  dplyr::filter(Freq > num.domains) %>%
  dplyr::arrange(desc(Freq))

age.TFs <- intersect(age.markers$Var1, TF.lst)

cat(paste0(length(age.markers$Var1), " genes detected!"))
```

```
## 542 genes detected!
```

```
cat(paste0(length(age.TFs), " TFs detected!"))
```

```
## 33 TFs detected!
```

This analysis identified 542 genes including 33 TFs as differentially expressed. I make a heatmap and perform hierarchical clustering on these 542 genes

```
## plot heatmap

seurat.pt <- subset(seurat, subset = Type_step1 == "Progenitor")
Idents(seurat.pt) <- "timepoint"

## fit gene expression dynamics for heatmaps
pt.mat <- as.matrix(GetAssayData(object = seurat.pt, slot = "counts"))
colnames(pt.mat) <- gsub("-", ".", colnames(pt.mat))

# log and recenter dataset
```

```r
pt.mat_log <- log(0.000001 + pt.mat)
pt.mat_zscored <- t(scale(t(pt.mat_log), center = T, scale = T))

## generate dataframe of in-vivo genes for plotting with ggplot
pt.mat.TFs <- data.frame(pt.mat_zscored[as.character(age.markers$Var1), ]) %>%
  tibble::rownames_to_column(var = "gene") %>%
  as_tibble() %>%
  tidyr::gather(key = "cellname", value = "readcount", -gene) %>%
  dplyr::mutate("timepoint" = Biobase::pData(eset)[.$cellname, "timepoint"]) %>%
  dplyr::group_by(gene, timepoint) %>%
  dplyr::summarise(mean = mean(readcount)) %>%
  dplyr::ungroup() %>%
  tidyr::spread(gene, value = "mean") %>%
  tibble::column_to_rownames("timepoint") %>%
  t() %>%
  as.matrix()
```

```
## `summarise()` has grouped output by 'gene'. You can override using the `.groups` argument.
```

```r
## generate hierarchical clustering

correlation_dist <- as.dist((1 - cor(t(pt.mat.TFs), method = "pearson")))
hc <- stats::hclust(correlation_dist, method = "ward.D2")

clusters <- cutree(hc, k = 2)
table(clusters)
```

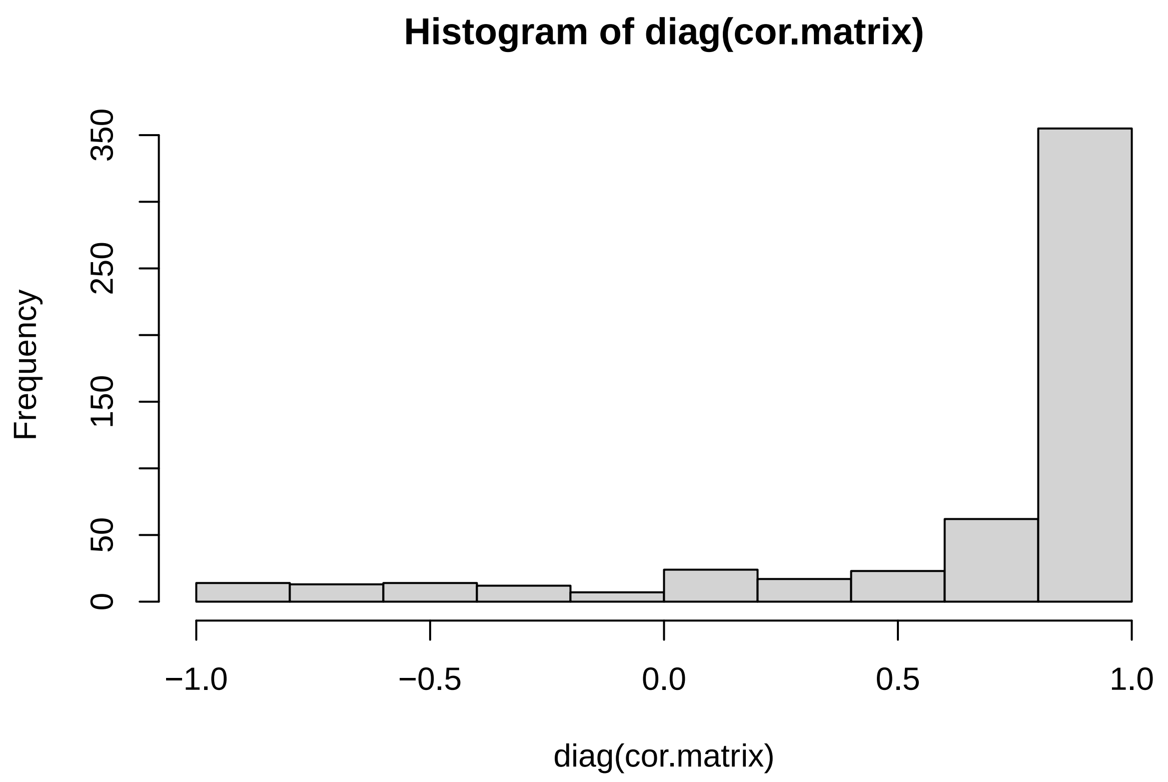
```
## clusters
##   1   2
## 182 360
```

Comparison with in-vitro RNAseq data from Rayon et al. 2020

To identify correlated, uncorrelated and anti-correlated genes we compare the expression dynamics of the 542 genes identified from the in-vivo scRNAseq data D5-D9 RNAseq data from the in-vitro differentiations by Pearson correlation.

```r
mouse <- read.table(paste0(dir, "/input/GSE140748_expression_matrix.mouse.abundance.tsv"), sep = "\t",

age.TFs.symbols.mouse <- fData(eset)$ensembl_gene_id[which(fData(eset)$current_gene_names %in% age.mark

mouse.df <- mouse[age.TFs.symbols.mouse, ] %>%
  t() %>%
  data.frame() %>%
  tibble::rownames_to_column(var = "sample_name") %>%
  tidyr::separate(sample_name, sep = "_", c("species", "day", "repeat_number")) %>%
  dplyr::filter(day %in% c(0, 1, 2, 3, 4, 5, 6, 7)) %>%
  dplyr::select(-c(species, repeat_number)) %>%
  dplyr::group_by(day) %>%
  dplyr::summarise_each(funs(mean)) %>%
  dplyr::mutate(day = as.character(day)) %>%
  mutate_if(is.numeric, function(x) x + 1) %>%
  mutate_if(is.numeric, function(x) log(x)) %>%
  tibble::remove_rownames() %>%
  tibble::column_to_rownames(var = "day") %>%
  scale(., center = T, scale = T) %>%
  as.data.frame() %>%
  tibble::rownames_to_column(var = "day") %>%
  tidyr::gather(key = "genename", value = "readcounts", -day) %>%
  dplyr::mutate(genename = fData(eset)[genename, "current_gene_names"]) %>%
  tidyr::spread(genename, readcounts) %>%
  tibble::column_to_rownames(var = "day") %>%
  as.matrix() %>%
  t()

mouse.df.sub <- mouse[age.TFs.symbols.mouse, ] %>%
  t() %>%
  data.frame() %>%
  tibble::rownames_to_column(var = "sample_name") %>%
  tidyr::separate(sample_name, sep = "_", c("species", "day", "repeat_number")) %>%
  dplyr::filter(day %in% c(2, 3, 4, 5, 6)) %>%
  dplyr::select(-c(species, repeat_number)) %>%
  dplyr::group_by(day) %>%
  dplyr::summarise_each(funs(mean)) %>%
  dplyr::mutate(day = as.character(day)) %>%
  mutate_if(is.numeric, function(x) x + 1) %>%
  mutate_if(is.numeric, function(x) log(x)) %>%
  tibble::remove_rownames() %>%
  tibble::column_to_rownames(var = "day") %>%
  scale(., center = T, scale = T) %>%
  as.data.frame() %>%
  tibble::rownames_to_column(var = "day") %>%
  tidyr::gather(key = "genename", value = "readcounts", -day) %>%
  dplyr::mutate(genename = fData(eset)[genename, "current_gene_names"]) %>%
  tidyr::spread(genename, readcounts) %>%
  tibble::column_to_rownames(var = "day") %>%
  as.matrix() %>%
  t()

mouse.df <- mouse.df[, as.character(sort(as.numeric(colnames(mouse.df))))]

order.mouse.genes <- intersect(rownames(pt.mat.TFs)[dendextend::order.hclust(hc)], rownames(mouse.df))

### for correlation analysis
common.names <- intersect(rownames(mouse.df.sub), rownames(pt.mat.TFs))

matrix1 <- mouse.df.sub[common.names, ] ## in-vitro data D5-D9
matrix2 <- pt.mat.TFs[common.names, ] ## in-vivo e9.5 - e13.5 progenitors

### remove genes which are NA due to z-score scaling
na.genes <- append(
  rownames(matrix1)[which(is.na(matrix1[, 1] == TRUE))],
  rownames(matrix2)[which(is.na(matrix2[, 1] == TRUE))]
)
```

```r
## calculate Pearson correlation matrix between in-vitro and in-vivo
cor.matrix <- cor(t(matrix1[!(rownames(matrix1) %in% na.genes), ]),
  t(matrix2[!(rownames(matrix2) %in% na.genes), ]),
  method = "pearson"
)

hist(diag(cor.matrix))
```



```r
correlated.genes <- names(diag(cor.matrix)[diag(cor.matrix) > .5])
uncorrelated.genes <- names(diag(cor.matrix)[diag(cor.matrix) < .5 & diag(cor.matrix) > -.5])
anticorrelated.genes <- names(diag(cor.matrix)[diag(cor.matrix) < -.5])


cat(paste0(length(correlated.genes), " correlated genes!"))
```

```
## 431 correlated genes!
```

```r
cat(paste0(length(uncorrelated.genes), " uncorrelated genes!"))
```

```
## 74 uncorrelated genes!
```

```r
cat(paste0(length(anticorrelated.genes), " anticorrelated genes!"))
```

```
## 36 anticorrelated genes!
```

Plot data

The data used for generating the figure in the manuscript has been saved as individual PDFs in the output folder. The image below is shown for aestethical reasons only.

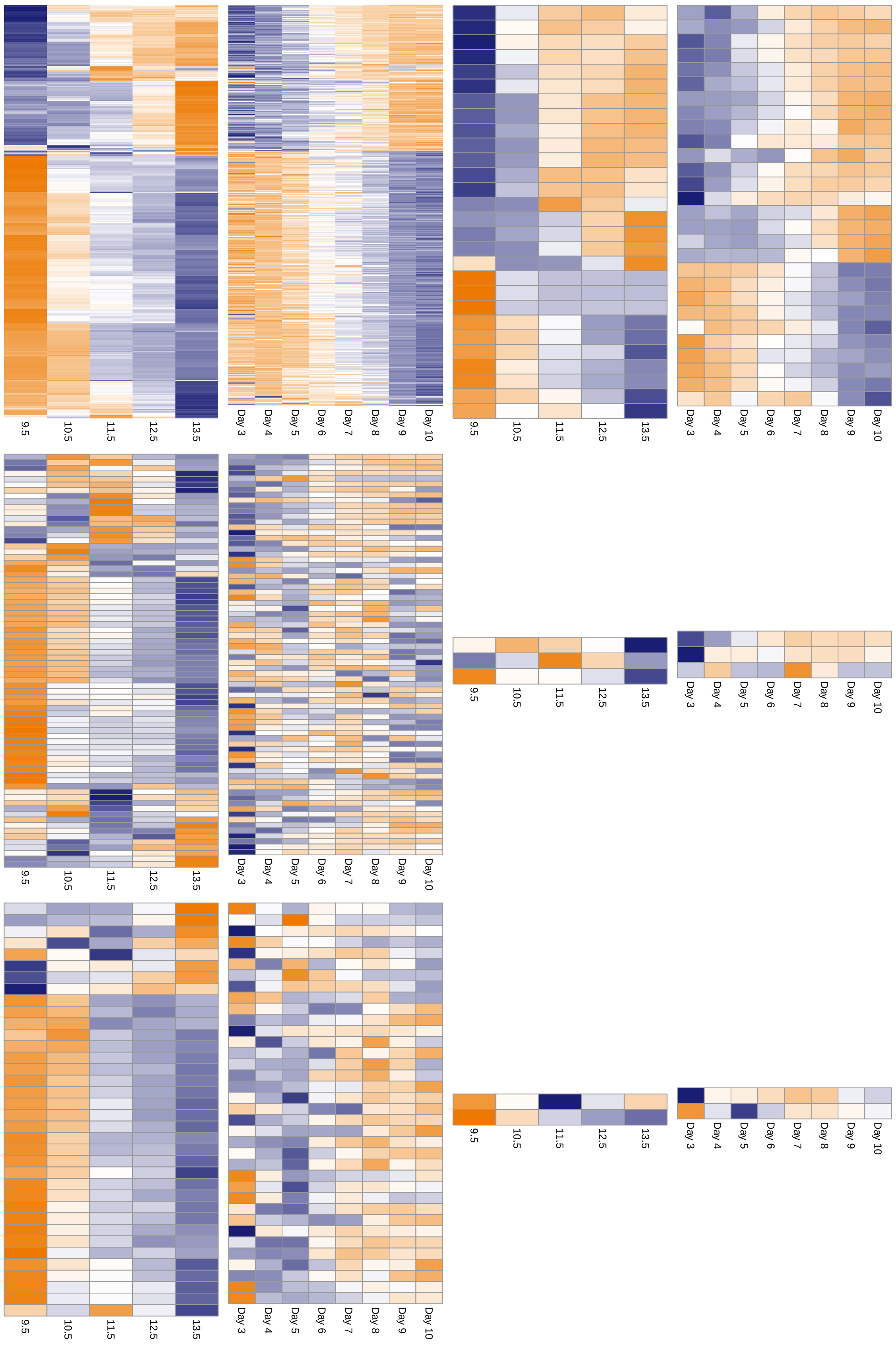
```r
colnames(mouse.df) <- c("Day 3", "Day 4", "Day 5", "Day 6", "Day 7", "Day 8", "Day 9", "Day 10")

plot.heatmaps.invivo.invitro <- function(genes,
                                         invivo.matrix = pt.mat.TFs,
                                         invitro.matrix = mouse.df,
                                         title = "Correlated.genes") {
  invivo.heatmap <- invivo.matrix[genes, ]
  invitro.heatmap <- invitro.matrix[genes, ]

  correlation_dist <- as.dist((1 - cor(t(invivo.heatmap), method = "pearson")))
  hc <- stats::hclust(correlation_dist, method = "ward.D2")

  clusters <- cutree(hc, k = 2)
  table(clusters)

  order.genes <- rownames(invivo.heatmap)[dendextend::order.hclust(hc)]

  lab.row <- unlist(lapply(order.genes, function(x) {
    if (x %in% TF.lst) {
      print(x)
      return(x)
    } else {
      return("")
    }
  }))

  if (length(which(order.genes %in% TF.lst == TRUE)) < 10) {
    cellheight <- 15
  } else {
    cellheight <- NA
  }

  hm1 <- pheatmap(invivo.heatmap[rev(order.genes), ],
    color = colorRampPalette(c("#191d73", "white", "#ed7901"))(n = 1000),
    cluster_rows = FALSE,
    cluster_cols = FALSE,
    show_rownames = F,
    scale = "row",
    legend = F
  )

  hm2 <- pheatmap(invivo.heatmap[rev(order.genes[order.genes %in% TF.lst]), ],
    color = colorRampPalette(c("#191d73", "white", "#ed7901"))(n = 1000),
    cluster_rows = FALSE,
    cluster_cols = FALSE,
    show_rownames = F,
    scale = "row",
    cellheight = cellheight,
    legend = F
  )

  hm3 <- pheatmap(invitro.heatmap[rev(order.genes), ],
    color = colorRampPalette(c("#191d73", "white", "#ed7901"))(n = 1000),
    cluster_rows = FALSE,
    cluster_cols = FALSE,
    show_rownames = F,
    scale = "row",
    legend = F
  )

  hm4 <- pheatmap(invitro.heatmap[rev(order.genes[order.genes %in% TF.lst]), ],
    color = colorRampPalette(c("#191d73", "white", "#ed7901"))(n = 1000),
    cluster_rows = FALSE,
    cluster_cols = FALSE,
    show_rownames = F,
    scale = "row",
    cellheight = cellheight,
    legend = F
  )

  return(cowplot::plot_grid(hm1$gtable, hm3$gtable, hm2$gtable, hm4$gtable,
    nrow = 1,
    align = "h", axis = "bt"
  ))

  pdf(paste0(title, "_invivo.pdf"), width = 4)
  heatmap.plus::heatmap.plus(pt.mat.TFs[order.genes, ],
    scale = "row",
    Colv = NA,
    Rowv = NA,
    col = colorRampPalette(c("#191d73", "white", "#ed7901"))(n = 1000),
    labCol = colnames(pt.mat.TFs),
    labRow = lab.row,
    ylab = paste0(title, " (", length(order.genes), " genes)"),
    cexRow = .8
  )
  graphics.off()

  pdf(paste0(title, "_invivo_TFs.pdf"), width = 4)
  heatmap.plus::heatmap.plus(pt.mat.TFs[order.genes[order.genes %in% TF.lst], ],
    scale = "row",
    Colv = NA,
    Rowv = NA,
    col = colorRampPalette(c("#191d73", "white", "#ed7901"))(n = 1000),
    labCol = colnames(pt.mat.TFs),
    labRow = order.genes[order.genes %in% TF.lst],
    ylab = paste0(title, " (", length(order.genes[order.genes %in% TF.lst]), " TFs)"),
    cexRow = .8
  )
  graphics.off()

  pdf(paste0(title, "_invitro.pdf"), width = 4)
```

```
  heatmap.plus::heatmap.plus(mouse.df[order.genes, ],
    scale = "row",
    Colv = NA,
    Rowv = NA,
    col = colorRampPalette(c("#191d73", "white", "#ed7901"))(n = 1000),
    labCol = colnames(mouse.df),
    labRow = lab.row,
    cexRow = .8
  )
  graphics.off()

  pdf(paste0(title, "_invitro_TFs.pdf"), width = 4)
  heatmap.plus::heatmap.plus(mouse.df[order.genes[order.genes %in% TF.lst], ],
    scale = "row",
    Colv = NA,
    Rowv = NA,
    col = colorRampPalette(c("#191d73", "white", "#ed7901"))(n = 1000),
    labCol = colnames(mouse.df),
    labRow = order.genes[order.genes %in% TF.lst],
    cexRow = .8
  )
  graphics.off()
}

cowplot::plot_grid(
  plot.heatmaps.invivo.invitro(genes = correlated.genes, title = "Correlated.genes"),
  plot.heatmaps.invivo.invitro(genes = uncorrelated.genes, title = "Uncorrelated.genes"),
  plot.heatmaps.invivo.invitro(genes = anticorrelated.genes, title = "Anticorrelated.genes"),
  ncol = 1
)
```

```
## [1] "Hmgb1"
## [1] "Lyar"
## [1] "Ybx3"
## [1] "Cebpz"
## [1] "Dnajc2"
## [1] "Hmga2"
## [1] "Hmga1"
## [1] "Nr6a1"
## [1] "Lin28a"
## [1] "Lin28b"
## [1] "Tsc22d4"
## [1] "Hopx"
## [1] "Thra"
## [1] "Nfix"
## [1] "Zfp422"
## [1] "Sox4"
## [1] "Sox6"
## [1] "Sall3"
## [1] "Tcf4"
## [1] "Zbtb20"
## [1] "Nfia"
## [1] "Nfib"
## [1] "Tsc22d1"
## [1] "Id4"
## [1] "Npas3"
## [1] "Sox9"
## [1] "Bcl11a"
## [1] "Jund"
```

```
## [1] "Bclaf1"
## [1] "Sox2"
## [1] "Sox11"
```

```
## [1] "Sub1"
## [1] "Id3"
```



9.5
10.5
11.5
12.5
13.5
Day 3
Day 4
Day 5
Day 6
Day 7
Day 8
Day 9
Day 10

Plot sessionInfo

```
sessionInfo()
```

```
## R version 4.0.4 (2021-02-15)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Catalina 10.15.7
##
## 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] stats4    parallel  stats     graphics  grDevices utils
## [7] datasets  methods   base
##
## other attached packages:
##  [1] pheatmap_1.0.12             tibble_3.1.1
##  [3] dendextend_1.14.0           SeuratObject_4.0.0
##  [5] Seurat_4.0.1                dplyr_1.0.5
##  [7] scater_1.18.6               SingleCellExperiment_1.12.0
##  [9] SummarizedExperiment_1.20.0 GenomicRanges_1.42.0
## [11] GenomeInfoDb_1.26.7         IRanges_2.24.1
## [13] S4Vectors_0.28.1            MatrixGenerics_1.2.1
## [15] matrixStats_0.58.0          ggplot2_3.3.3
## [17] plyr_1.8.6                  Biobase_2.50.0
## [19] BiocGenerics_0.36.1
##
## loaded via a namespace (and not attached):
##   [1] backports_1.2.1          igraph_1.2.6
##   [3] lazyeval_0.2.2           splines_4.0.4
##   [5] BiocParallel_1.24.1      listenv_0.8.0
##   [7] scattermore_0.7          digest_0.6.27
##   [9] htmltools_0.5.1.1        viridis_0.6.0
##  [11] fansi_0.4.2              memoise_2.0.0
##  [13] magrittr_2.0.1           tensor_1.5
##  [15] cluster_2.1.2            ROCR_1.0-11
##  [17] limma_3.46.0             globals_0.14.0
##  [19] R.utils_2.10.1           spatstat.sparse_2.0-0
##  [21] colorspace_2.0-0         blob_1.2.1
##  [23] ggrepel_0.9.1            xfun_0.22
##  [25] crayon_1.4.1             RCurl_1.98-1.3
##  [27] jsonlite_1.7.2           spatstat.data_2.1-0
##  [29] survival_3.2-10          zoo_1.8-9
##  [31] glue_1.4.2               polyclip_1.10-0
##  [33] gtable_0.3.0             zlibbioc_1.36.0
##  [35] XVector_0.30.0           leiden_0.3.7
##  [37] DelayedArray_0.16.3      R.cache_0.14.0
##  [39] BiocSingular_1.6.0       future.apply_1.7.0
##  [41] abind_1.4-5              scales_1.1.1
##  [43] DBI_1.1.1                   miniUI_0.1.1.1
##  [45] Rcpp_1.0.6                  viridisLite_0.4.0
##  [47] xtable_1.8-4                reticulate_1.18
##  [49] spatstat.core_2.1-2         bit_4.0.4
##  [51] rsvd_1.0.5                  htmlwidgets_1.5.3
##  [53] httr_1.4.2                  RColorBrewer_1.1-2
##  [55] ellipsis_0.3.1              ica_1.0-2
##  [57] pkgconfig_2.0.3             R.methodsS3_1.8.1
##  [59] scuttle_1.0.4               uwot_0.1.10
##  [61] deldir_0.2-10               utf8_1.2.1
##  [63] AnnotationDbi_1.52.0        tidyselect_1.1.1
##  [65] rlang_0.4.10                reshape2_1.4.4
##  [67] later_1.1.0.1               cachem_1.0.4
##  [69] munsell_0.5.0               tools_4.0.4
##  [71] RSQLite_2.2.6               generics_0.1.0
##  [73] ggridges_0.5.3              evaluate_0.14
##  [75] stringr_1.4.0               fastmap_1.1.0
##  [77] goftest_1.2-2               yaml_2.2.1
##  [79] bit64_4.0.5                 knitr_1.32
##  [81] fitdistrplus_1.1-3          purrr_0.3.4
##  [83] RANN_2.6.1                  nlme_3.1-152
##  [85] pbapply_1.4-3               future_1.21.0
##  [87] sparseMatrixStats_1.2.1     mime_0.10
##  [89] R.oo_1.24.0                 compiler_4.0.4
##  [91] rstudioapi_0.13             beeswarm_0.3.1
##  [93] plotly_4.9.3                png_0.1-7
##  [95] spatstat.utils_2.1-0        stringi_1.5.3
##  [97] highr_0.9                   RSpectra_0.16-0
##  [99] lattice_0.20-41             Matrix_1.3-2
## [101] styler_1.4.1                vctrs_0.3.7
## [103] pillar_1.6.0                lifecycle_1.0.0
## [105] spatstat.geom_2.1-0         lmtest_0.9-38
## [107] RcppAnnoy_0.0.18            BiocNeighbors_1.8.2
## [109] data.table_1.14.0           cowplot_1.1.1
## [111] bitops_1.0-6                irlba_2.3.3
## [113] httpuv_1.5.5                patchwork_1.1.1
## [115] R6_2.5.0                    promises_1.2.0.1
## [117] KernSmooth_2.23-18          gridExtra_2.3
## [119] vipor_0.4.5                 parallelly_1.25.0
## [121] codetools_0.2-18            MASS_7.3-53.1
## [123] assertthat_0.2.1            withr_2.4.2
## [125] sctransform_0.3.2           GenomeInfoDbData_1.2.4
## [127] Antler_0.9.0                mgcv_1.8-35
## [129] rpart_4.1-15                grid_4.0.4
## [131] beachmat_2.6.4              tidyr_1.1.3
## [133] rmarkdown_2.7               DelayedMatrixStats_1.12.3
## [135] Rtsne_0.15                  shiny_1.6.0
## [137] ggbeeswarm_0.6.0
```