Supplementary Resource 2 - R script

The script is sectioned into the following,

- Loading requisite libraries and setting up workspaces. This will need to be updated as per requirement and setup by the end user.
- Filtered results from cell ranger are loaded and quality control is performed.
- Post-QC data is normalised and analysed to identify and segregate microglia and T cells.
- Differential gene expression is performed on sub-populations, contrasting IL2 treated samples with GFP controls in Sham samples and TBI samples separately.
- Differential expression results are used alongside total expression to perform gene set and pathway enrichment on microglia sub-populations.

–

#loading required libraries and setting up the workspace

```r
#load libraries
{
  library(Seurat)
  library(ggplot2)
  library(sctransform)
  library(DESeq2)
  library(scDblFinder)
  library(BiocParallel)
  library(SeuratWrappers)
  library(ggpubr)
  library(rstatix)
  library(stringr)
  library(ggpattern)
  library(gage)
  library(pathview)
  library(clusterProfiler)
}

#setting up workspace and location variables
## adjust accordingly
{
  old_wrkdir <- getwd()
  setwd("/home/tareens/Workspace/TBI_paper/")
  loc_workspace <- getwd()

  loc_genomes <- paste0(loc_workspace, "/00_genomes/")
  loc_raw_data <- paste0(loc_workspace, "/00_raw_data/")
  loc_metadata <- paste0(loc_workspace, "/00_metadata/")
  loc_scripts <- paste0(loc_workspace, "/00_scripts/")
  loc_cellranger_results <- paste0(loc_workspace,
                                   "/01_cellranger_results/")
  loc_seurat_results <- paste0(loc_workspace,
                               "/02_seurat_results/")
  loc_superclusters <- paste0(loc_workspace,
                              "/03_superclusters/")
  loc_diff_exprs_results <- paste0(loc_workspace,
                                "/04_diff_exprs_results/")

  loc_pathview_results <- paste0(loc_workspace,
                                "/05_pathview_results/")

}
```

#loading data and performing QC with filtering and doublet removal

```
#loading individual cellranger count results for manual normalisation
{
  #loading filtered_features results, which have been filtered by cellranger to
  # remove background barcodes, into a list
  samples_list <- dir(path = paste0(loc_workspace,
                                    "/01_cellranger_results/",
                                    "all_individual_custom_filtered_and_raw/"))

  #reading cellranger results and formatting list
  individual_count_data <- list()
  for(sample_name in samples_list){
    print(sample_name)
    individual_count_data[[length(individual_count_data)+1]] <-
      Read10X(data.dir = paste0(loc_workspace,
                                "/01_cellranger_results/",
                                "all_individual_custom_filtered_and_raw/",
                                sample_name,
                                "/filtered_feature_bc_matrix/"))
    colnames(individual_count_data[[length(individual_count_data)]]) <-
      paste0(gsub(pattern = "^1_",
                  replacement = "1.",
                  x = sample_name,
                  ignore.case = T),
             "__",
             gsub(pattern = "\\-[0-9]+$",
                  replacement = "",
                  x = colnames(
                    individual_count_data[[length(individual_count_data)]]),
                  ignore.case = T)
             )
    names(individual_count_data)[length(individual_count_data)] <- sample_name
  }

  #converting data to Seurat objects and adding metadata
  for(i in seq(1, length(individual_count_data))){
    individual_count_data[[i]] <-
      CreateSeuratObject(counts = individual_count_data[[i]],
                         project = names(individual_count_data)[i])

    individual_count_data[[i]]@meta.data$label <-
      names(individual_count_data)[i]
```

```r
    individual_count_data[[i]]@meta.data$sample <-
      gsub(pattern = "^.*_",
           replacement = "",
           x = names(individual_count_data)[i],
           ignore.case = T)

    individual_count_data[[i]]@meta.data$tcell_group <-
      ifelse(gsub(pattern = "^.*_",
                  replacement = "",
                  x = names(individual_count_data)[i],
                  ignore.case = T) %in% c("04", "08"), "Sham", "TBI")

    individual_count_data[[i]]@meta.data$micro_group <-
      ifelse(gsub(pattern = "^.*_",
                  replacement = "",
                  x = names(individual_count_data)[i],
                  ignore.case = T) %in% c("01", "08"), "Sham", "TBI")

    individual_count_data[[i]]@meta.data$treatment <-
      ifelse(gsub(pattern = "^.*_",
                  replacement = "",
                  x = names(individual_count_data)[i],
                  ignore.case = T) %in% c("01", "02", "03", "04"), "PHP.GFAP-GFP",
             "PHP.GFAP-IL2")
  }
}

#performing QC/filtering on SeuratObjects
{
  #calculating mitochondrial gene percentage
  mt_gene_list <- list()
  for(i in seq(1, length(individual_count_data))){

    mt_gene_list[[i]] <- grep(pattern = "mt-",
                         x = rownames(individual_count_data[[i]]),
                         value = T, ignore.case = T)
    names(mt_gene_list)[i] <- names(individual_count_data)[i]
    individual_count_data[[i]] <-
      PercentageFeatureSet(object = individual_count_data[[i]],
                           pattern = "^mt-",
                           col.name = "percent_mt_genes")
  }

  #filtering out cells with >10% mitochondrial expression of total
  for(i in seq(1, length(individual_count_data))){
    individual_count_data[[i]] <- subset(x = individual_count_data[[i]],
                                         subset = percent_mt_genes<=10)
  }
}

#performing SCT normalisation
{
  #merging individual samples into a single object
  SCT_norm_data <- merge(x = individual_count_data[[1]],
                         y = individual_count_data[-1])

  #SCTransform normalisation -- regressing mt-gene expression and counts
  SCT_norm_data <- SCTransform(object = SCT_norm_data,
                               vars.to.regress = c("percent_mt_genes",
                                                   "nCount_RNA"),
                               return.only.var.genes = F,
                               verbose = T)
}

#performing doublet analysis and removal
{
  #renaming to data and calling garbage collection
  data <- SCT_norm_data
  rm(SCT_norm_data)
  gc()

  #running PCA on 100 components
  data <- RunPCA(data, npcs = 100)

  #elbow plot to asses the number of components to include
  plot(
  ElbowPlot(object = data, ndims = 100)  +
    geom_hline(yintercept = 1) +
    geom_hline(yintercept = 2) +
    geom_hline(yintercept = 3)
  )

  #running doublet analysis and removal
  metadata <- data@meta.data
  data_sce <- as.SingleCellExperiment(x = data)
  data_sce <- scDblFinder(sce = data_sce,
                          clusters = NULL,
                          samples = metadata$label,
                          trajectoryMode = F,
                          use.cxds = T,
                          nfeatures = 3000,
                          dims = 75,
                          includePCs = 1:75,
                          returnType = "full",
                          nrounds = NULL,
                          BPPARAM = MulticoreParam(6))

  metadata$doublet_status <- as.character(data_sce$scDblFinder.class)
  data@meta.data <- metadata

  #running umap to visualise doublets
  data <- RunUMAP(object = data,
                  dims = 1:75,
                  reduction = "pca")

  #finding cell neighbours
  data <- FindNeighbors(object = data,
                        reduction = "umap",
                        dims = 1:2,
                        nn.eps = 0)

  #finding clusters
  data <- FindClusters(object = data,
                       resolution = 0.013,
                       n.start = 50)

  plot(
  DimPlot(object = data,
          reduction = "umap",
          group.by = "doublet_status") +
    xlab("Dimension 1") + ylab("Dimension 2") +
    labs(title = "Detected doublets")
  )

  data <- subset(data, subset = doublet_status == "singlet")
}

#doublechecking QC
# following (https://hbctraining.github.io/In-depth-NGS-Data-Analysis-Course/
# sessionIV/lessons/SC_quality_control_analysis.html)
{
  # Visualize the number UMIs/transcripts per cell
  print(
  ggplot(data = data@meta.data, aes(color=sample, x=nCount_RNA, fill= sample)) +
        geom_density() +
        scale_x_log10() +
        ylab("log10 cell density") +
        geom_vline(xintercept = 500) + theme_classic()
  )
  print(
  ggplot(data = data@meta.data, aes(color=sample, x=nCount_SCT, fill= sample)) +
        geom_density() +
        scale_x_log10() +
        ylab("log10 cell density") +
        geom_vline(xintercept = 500) + theme_classic()
  )

  # Visualize the distribution of genes detected per cell via histogram
  print(
  ggplot(data = data@meta.data, aes(color=sample, x=nFeature_RNA, fill= sample)) +
        geom_density() +
        scale_x_log10() +
        ylab("log10 cell density") +
        geom_vline(xintercept = 500) + theme_classic()
  )
  print(
  ggplot(data = data@meta.data, aes(color=sample, x=nFeature_SCT, fill= sample)) +
        geom_density() +
        scale_x_log10() +
        ylab("log10 cell density") +
        geom_vline(xintercept = 500) + theme_classic()
  )

  # Visualize the correlation between genes detected and number of UMIs and
  # determine whether there is a strong presence of cells with low numbers of genes/UMIs
  print(
  ggplot(data = data@meta.data,
         aes(x=nCount_SCT, y=nFeature_SCT, color=label)) +
    geom_point() +
    stat_smooth(method=lm) +
    scale_x_log10() +
    scale_y_log10() +
    geom_vline(xintercept = 800) + theme_classic()
  )

  # Visualize the distribution of mitochondrial gene expression detected per cell
  print(
  ggplot(data = data@meta.data, aes(color=label, x=percent_mt_genes, fill=label)) +
        geom_density() +
        scale_x_log10() +
        geom_vline(xintercept = 0.1) + theme_classic()
  )
}
```

#processing SCT normalised data

```
#full dataset
{
  #creating combined label
  data@meta.data$group_label <-
    paste0("T-cell.", data@meta.data$tcell_group,
           "-",
           "Micro.", data@meta.data$micro_group,
           "-",
           "Astro.", data@meta.data$astro_group)


  #running PCA
  data <- RunPCA(data, npcs = 100)

  #elbow plot
  print(
  ElbowPlot(object = data, ndims = 100) +
    geom_hline(yintercept = 1) +
    geom_hline(yintercept = 2) +
    geom_hline(yintercept = 3)
  )

  #running UMAP
  data <- RunUMAP(object = data,
                  dims = 1:75,
                  reduction = "pca"
                  )
  #finding cell neighbours using t-SNE
  data <- FindNeighbors(object = data,
                        reduction = "umap",
                        dims = 1:2,
                        nn.eps = 0)

  #finding clusters
  data <- FindClusters(object = data,
                       resolution = 0.0025,
                       n.start = 50)

  print(
  DimPlot(object = data,
          label = T,
          reduction = "umap") +
    NoLegend() + xlab("Dimension 1") + ylab("Dimension 2")
  )

  #saving seurat object
  save(data,
       file = paste0(loc_seurat_results,
                     "full_data_seurat.RData"),
       compress = T)

  #checking expression of known markers
  print(
  FeaturePlot(object = data,
              features = c("Tmem119", "Cx3cr1",
                           "Aqp4", "Fgfr3", "Slc4a4", "Sox9",
                           "Cd3d", "Cd3e", "Cd3g"),
              ncol = 3, reduction = "umap", slot = "data",
              pt.size = 0.01) & coord_fixed()
  )
}

#subsetting and analysing microglia data
{
  micro_data <- subset(x = data, subset = seurat_clusters %in% c(0))

  #creating microglia specific label
  micro_data@meta.data$micro_treat_group <-
    paste0(micro_data@meta.data$micro_group, "_", micro_data@meta.data$treatment)

  #running SCT on a copy of micro_data to get VariableFeatures specific to this cluster
  VariableFeatures(micro_data) <-
    VariableFeatures(SCTransform(object = micro_data,
                                 vars.to.regress = c("percent_mt_genes",
                                                     "nCount_RNA"),
                                 return.only.var.genes = F,
                                 verbose = T))

  #running PCA with the new variable features
  micro_data <- RunPCA(object = micro_data,
```

```r
                       features = VariableFeatures(object = micro_data),
                       npcs = 100)

#elbow plot
print(
ElbowPlot(object = micro_data, ndims = 100) +
  geom_hline(yintercept = 1) +
  geom_hline(yintercept = 2) +
  geom_hline(yintercept = 3)
)

#running UMAP
micro_data <- RunUMAP(object = micro_data,
                      dims = 1:75)

#finding cell neighbours using t-SNE
micro_data <- FindNeighbors(object = micro_data,
                            reduction = "umap",
                            dims = 1:2,
                            nn.eps = 0)

#finding clusters
micro_data <- FindClusters(object = micro_data,
                           resolution = 0.05,
                           n.start = 50)

#visualising umap
print(
DimPlot(object = micro_data,
        label = F,
        reduction = "umap") +
  NoLegend() + xlab("Dimension 1") + ylab("Dimension 2") +
  scale_color_brewer(palette = "Set2")
)

#assigning new names to clusters for downstream analyses
## cluster 5 and 7 are monocytes and macrophages, hence removed
{
  micro_data <- subset(micro_data,
                       seurat_clusters %in% c(0, 2, 3, 4, 6, 1))

  micro_data$seurat_clusters_renamed <-
    as.character(micro_data$seurat_clusters)
  micro_data$seurat_clusters_renamed[
    micro_data$seurat_clusters_renamed %in% c("0", "2", "3", "6")] <-
    "Homeostatic microglia"
  micro_data$seurat_clusters_renamed[
    micro_data$seurat_clusters_renamed %in% c("1", "4")] <- "Activated microglia"
  micro_data$seurat_clusters_renamed <-
    as.factor(micro_data$seurat_clusters_renamed)

  print(
  DimPlot(object = micro_data,
          group.by = "seurat_clusters_renamed",
          split.by = "micro_treat_group",
          reduction = "umap",
          raster = T) +
      xlab("Dimension 1") + ylab("Dimension 2") +
      labs(colour = "Type", title = "") +
      scale_color_brewer(palette = "Set2") +
      coord_fixed() &
      theme(legend.position = "bottom")
    )
  }

  #checking expression of specific genes
  print(
  FeaturePlot(object = micro_data,
              features = c("Apoe", "H2-Eb1"),
              slot = "data", pt.size = 3, raster = T, ncol = 1) &
    coord_fixed() &
    xlab("Dimension 1") & ylab("Dimension 2") &
    theme(text = element_text(size = 20))
  )

  print(
  FeaturePlot(object = micro_data,
              features = c("Lpl", "Cst7", "Axl", "Itgax",
                           "Spp1", "Ccl6", "Csf1", "H2-Aa"),
              slot = "data", pt.size = 3, raster = T, ncol = 4) &
    coord_fixed() &
    xlab("Dimension 1") & ylab("Dimension 2") &
    theme(text = element_text(size = 20))
  )

  #saving the microglia object
  save(micro_data,
       file = paste0(loc_superclusters,
                     "micro/microglia_seurat.RData"),
       compress = T)
}

#subsetting and analysing T cell data
{
  tcell_data <- subset(x = data, subset = seurat_clusters %in% c(1,4,6))

  tcell_data@meta.data$tcell_treat_group <-
    paste0(tcell_data@meta.data$tcell_group, "_", tcell_data@meta.data$treatment)

  #running SCT on a copy of tcell_data to get VariableFeatures specific to this cluster
  VariableFeatures(tcell_data) <-
    VariableFeatures(SCTransform(object = tcell_data,
                                 #batch_var = "batch",
                                 vars.to.regress = c("percent_mt_genes",
                                                     "nCount_RNA"),
                                 return.only.var.genes = F,
                                 verbose = T))

#running PCA with the new variable features
tcell_data <- RunPCA(object = tcell_data,
                     features = VariableFeatures(object = tcell_data),
                     npcs = 100)

#elbow plot
print(
ElbowPlot(object = tcell_data, ndims = 100) +
  geom_hline(yintercept = 1) +
  geom_hline(yintercept = 2) +
  geom_hline(yintercept = 3)
)

#running UMAP
tcell_data <- RunUMAP(object = tcell_data,
                      dims = 1:75)

#finding cell neighbours using t-SNE
tcell_data <- FindNeighbors(object = tcell_data,
                            reduction = "umap",
                            dims = 1:2,
                            nn.eps = 0)

#finding clusters
tcell_data <- FindClusters(object = tcell_data,
                           resolution = 0.025,
                           n.start = 50)

#visualising umap
print(
DimPlot(object = tcell_data,
        label = F,
        reduction = "umap") +
  NoLegend() + xlab("Dimension 1") + ylab("Dimension 2")
)

#some cells are exhibiting Hbb- haemoglobin genes and are contaminants using
# umap coordinates to remove them from the data
# check umap and update coordinate thresholds accordingly
{
  umapCoord <- as.data.frame(Embeddings(object = tcell_data[["umap"]]))
  exclusion_cells <-
    rownames(subset(umapCoord, umapCoord$UMAP_1 < -10 & umapCoord$UMAP_2 > 4))

  tcell_data@meta.data$cell_id <- rownames(tcell_data@meta.data)

  tcell_data <- subset(tcell_data,
                       cell_id %in% exclusion_cells,
                       invert = T)
  print(
  DimPlot(object = tcell_data,
          label = TRUE,
          reduction = "umap", label.size = 5, repel = T) +
    NoLegend() + xlab("Dimension 1") + ylab("Dimension 2")
  )
}

#assigning new names to clusters for visualisations
{
  tcell_data$seurat_clusters_renamed <-
    as.character(tcell_data$seurat_clusters)
  tcell_data$seurat_clusters_renamed[
    tcell_data$seurat_clusters_renamed %in% c("0", "4", "6")] <- "CD8+ T cells"
  tcell_data$seurat_clusters_renamed[
    tcell_data$seurat_clusters_renamed %in% c("1")] <- "CD4+ Tconv"
  tcell_data$seurat_clusters_renamed[
    tcell_data$seurat_clusters_renamed %in% c("3")] <- "Tregs"
  tcell_data$seurat_clusters_renamed[
    tcell_data$seurat_clusters_renamed %in% c("2")] <- "Invariant-like T cells"
  tcell_data$seurat_clusters_renamed[
    tcell_data$seurat_clusters_renamed %in% c("5")] <- "Proliferating T cells"
  tcell_data$seurat_clusters_renamed[
    tcell_data$seurat_clusters_renamed %in% c("7")] <- "TH17 T cells"
  tcell_data$seurat_clusters_renamed <-
    as.factor(tcell_data$seurat_clusters_renamed)

  print(
  DimPlot(object = tcell_data,
        group.by = "seurat_clusters_renamed",
        split.by = "tcell_treat_group",
        pt.size = 3,
        reduction = "umap",
        raster = T) +
    xlab("Dimension 1") + ylab("Dimension 2") +
    labs(colour = "Type", title = "") +
    scale_color_brewer(palette = "Set2") +
    coord_fixed() &
    theme(legend.position = "bottom")
  )
}

#saving tcell seurat object
save(tcell_data,
     file = paste0(loc_superclusters,
                   "tcell/tcell_data.rds"),
     compress = T)

#checking expression of specific genes
print(
FeaturePlot(object = tcell_data,
            features = c("Cd3d", "Cd4", "Cd8a", "Cd8b1",
                         "Foxp3", "Il2ra", "Sell"),
            slot = "data", pt.size = 3, raster = T, ncol = 1) &
  coord_fixed() &
```

```r
    xlab("Dimension 1") & ylab("Dimension 2") &
    theme(text = element_text(size = 20))
  )
}
```

#differential expression analysis

```r
#finding markers for homeostatic and disease-associated microglia clusters separately
{
  #for all microglia
  {
    #Sham - IL2 vs GFP
    seurat_object <- subset(micro_data,
                            micro_treat_group %in% c("Sham_PHP.GFAP-GFP",
                                                     "Sham_PHP.GFAP-IL2"))
    markers_list <- FindMarkers(object = seurat_object,
                                ident.1 = "Sham_PHP.GFAP-IL2",
                                group.by = "micro_treat_group",
                                logfc.threshold = 0,
                                min.pct = 0,
                                test.use = "negbinom")
    saveRDS(object = markers_list,
            file = paste0(loc_diff_exprs_results,
                          "micro/micro_all_markers_list_Sham.rds"),
            compress = T)

    #TBI - IL2 vs GFP
    seurat_object <- subset(micro_data,
                            micro_treat_group %in% c("TBI_PHP.GFAP-GFP",
                                                     "TBI_PHP.GFAP-IL2"))
    markers_list <- FindMarkers(object = seurat_object,
                                ident.1 = "TBI_PHP.GFAP-IL2",
                                group.by = "micro_treat_group",
                                logfc.threshold = 0,
                                min.pct = 0,
                                test.use = "negbinom")
    saveRDS(object = markers_list,
            file = paste0(loc_diff_exprs_results,
                          "micro/micro_all_markers_list_TBI.rds"),
            compress = T)
  }

  #for homeostatic microglia
  {
    #Sham - IL2 vs GFP
    seurat_object <- subset(micro_data,
                            micro_treat_group %in% c("Sham_PHP.GFAP-GFP",
                                                     "Sham_PHP.GFAP-IL2") &
                              seurat_clusters %in% c(0, 2, 3, 6))
    markers_list <- FindMarkers(object = seurat_object,
                                ident.1 = "Sham_PHP.GFAP-IL2",
                                group.by = "micro_treat_group",
                                logfc.threshold = 0,
                                min.pct = 0,
                               test.use = "negbinom")
  saveRDS(object = markers_list,
          file = paste0(loc_diff_exprs_results,
                        "micro/micro_homeo_markers_list_Sham.rds"),
          compress = T)

  #TBI - IL2 vs GFP
  seurat_object <- subset(micro_data,
                          micro_treat_group %in% c("TBI_PHP.GFAP-GFP",
                                                   "TBI_PHP.GFAP-IL2") &
                            seurat_clusters %in% c(0, 2, 3, 6))
  markers_list <- FindMarkers(object = seurat_object,
                              ident.1 = "TBI_PHP.GFAP-IL2",
                              group.by = "micro_treat_group",
                              logfc.threshold = 0,
                              min.pct = 0,
                              test.use = "negbinom")
  saveRDS(object = markers_list,
          file = paste0(loc_diff_exprs_results,
                        "micro/micro_homeo_markers_list_TBI.rds"),
          compress = T)
}

#for activated cluster microglia
{
  #Sham - IL2 vs GFP
  seurat_object <- subset(micro_data,
                          micro_treat_group %in% c("Sham_PHP.GFAP-GFP",
                                                   "Sham_PHP.GFAP-IL2") &
                            seurat_clusters %in% c(1, 4))
  markers_list <- FindMarkers(object = seurat_object,
                              ident.1 = "Sham_PHP.GFAP-IL2",
                              group.by = "micro_treat_group",
                              logfc.threshold = 0,
                              min.pct = 0,
                              test.use = "negbinom")
  saveRDS(object = markers_list,
          file = paste0(loc_diff_exprs_results,
                        "micro/micro_activated_markers_list_Sham.rds"),
          compress = T)

  #TBI - IL2 vs GFP
  seurat_object <- subset(micro_data,
                          micro_treat_group %in% c("TBI_PHP.GFAP-GFP",
                                                   "TBI_PHP.GFAP-IL2") &
                            seurat_clusters %in% c(1, 4))
  markers_list <- FindMarkers(object = seurat_object,
                              ident.1 = "TBI_PHP.GFAP-IL2",
                              group.by = "micro_treat_group",
                              logfc.threshold = 0,
                              min.pct = 0,
                              test.use = "negbinom")
  saveRDS(object = markers_list,
```

```r
            file = paste0(loc_diff_exprs_results,
                          "micro/micro_activated_markers_list_TBI.rds"),
            compress = T)
  }
}

#finding markers for CD4, CD8 and Treg clusters separately
{
  #for CD8+
  {
    #Sham - IL2 vs GFP
    seurat_object <- subset(tcell_data,
                            tcell_treat_group %in% c("Sham_PHP.GFAP-GFP",
                                                     "Sham_PHP.GFAP-IL2") &
                              seurat_clusters %in% c(0, 4, 6))
    markers_list <- FindMarkers(object = seurat_object,
                                ident.1 = "Sham_PHP.GFAP-IL2",
                                group.by = "tcell_treat_group",
                                logfc.threshold = 0,
                                min.pct = 0,
                                test.use = "negbinom")
    saveRDS(object = markers_list,
            file = paste0(loc_diff_exprs_results,
                          "tcell/tcell_CD8_markers_list_Sham.rds"),
            compress = T)

    #TBI - IL2 vs GFP
    seurat_object <- subset(tcell_data,
                            tcell_treat_group %in% c("TBI_PHP.GFAP-GFP",
                                                     "TBI_PHP.GFAP-IL2") &
                              seurat_clusters %in% c(0, 4, 6))
    markers_list <- FindMarkers(object = seurat_object,
                                ident.1 = "TBI_PHP.GFAP-IL2",
                                group.by = "tcell_treat_group",
                                logfc.threshold = 0,
                                min.pct = 0,
                                test.use = "negbinom")
    saveRDS(object = markers_list,
            file = paste0(loc_diff_exprs_results,
                          "tcell/tcell_CD8_markers_list_TBI.rds"),
            compress = T)
  }

  #for CD4+
  {
    #Sham - IL2 vs GFP
    seurat_object <- subset(tcell_data,
                            tcell_treat_group %in% c("Sham_PHP.GFAP-GFP",
                                                     "Sham_PHP.GFAP-IL2") &
                              seurat_clusters %in% c(1))
    markers_list <- FindMarkers(object = seurat_object,
                                ident.1 = "Sham_PHP.GFAP-IL2",
                                group.by = "tcell_treat_group",
                                logfc.threshold = 0,
                                min.pct = 0,
                                test.use = "negbinom")
  saveRDS(object = markers_list,
          file = paste0(loc_diff_exprs_results,
                        "tcell/tcell_CD4_markers_list_Sham.rds"),
          compress = T)

  #TBI - IL2 vs GFP
  seurat_object <- subset(tcell_data,
                          tcell_treat_group %in% c("TBI_PHP.GFAP-GFP",
                                                   "TBI_PHP.GFAP-IL2") &
                            seurat_clusters %in% c(1))
  markers_list <- FindMarkers(object = seurat_object,
                                ident.1 = "TBI_PHP.GFAP-IL2",
                                group.by = "tcell_treat_group",
                                logfc.threshold = 0,
                                min.pct = 0,
                                test.use = "negbinom")
  saveRDS(object = markers_list,
          file = paste0(loc_diff_exprs_results,
                        "tcell/tcell_CD4_markers_list_TBI.rds"),
          compress = T)
}

#for Tregs
{
  #Sham - IL2 vs GFP
  seurat_object <- subset(tcell_data,
                          tcell_treat_group %in% c("Sham_PHP.GFAP-GFP",
                                                   "Sham_PHP.GFAP-IL2") &
                            seurat_clusters %in% c(3))
  markers_list <- FindMarkers(object = seurat_object,
                                ident.1 = "Sham_PHP.GFAP-IL2",
                                group.by = "tcell_treat_group",
                                logfc.threshold = 0,
                                min.pct = 0,
                                test.use = "negbinom")
  saveRDS(object = markers_list,
          file = paste0(loc_diff_exprs_results,
                        "tcell/tcell_treg_markers_list_Sham.rds"),
          compress = T)

  #TBI - IL2 vs GFP
  seurat_object <- subset(tcell_data,
                          tcell_treat_group %in% c("TBI_PHP.GFAP-GFP",
                                                   "TBI_PHP.GFAP-IL2") &
                            seurat_clusters %in% c(3))
  markers_list <- FindMarkers(object = seurat_object,
                                ident.1 = "TBI_PHP.GFAP-IL2",
                                group.by = "tcell_treat_group",
                                logfc.threshold = 0,
                                min.pct = 0,
                                test.use = "negbinom")
    saveRDS(object = markers_list,
            file = paste0(loc_diff_exprs_results,
                          "tcell/tcell_treg_markers_list_TBI.rds"),
            compress = T)
  }
}
```

#GSEA analysis using GAGE and PathView

```
#reading in the seurat objects
{
  load(file = paste0(loc_superclusters,
                                   "micro/microglia_seurat.RData"))
  load(file = paste0(loc_superclusters,
                                   "tcell/tcell_seurat.RData"))

  #mapping gene symbols to entrez/ncbi ids (gage only uses these)
  genes <- data.frame(genes_symbol = rownames(micro_data[["SCT"]]@data))
  genes <- as.data.frame(id2eg(ids = genes$genes_symbol,
                               category = "SYMBOL",
                               org = "mouse",
                               na.rm = F))
  genes$ENTREZID[genes$ENTREZID == "NA"] <- NA

  #loading/downloading kegg genesets
  kegg_genesets <- kegg.gsets(species = "mouse",
                              id.type = "entrez",
                              check.new = T)
  saveRDS(object = kegg_genesets,
          file = paste0(loc_pathview_results, "kegg_genesets.rds"),
          compress = T)
  kegg_genesets <- readRDS(file = paste0(loc_pathview_results,
                                         "kegg_genesets.rds"))
}

#analysing microglia
{
  #homeostatic microglia
  {
    #loading metadata
    metadata <- subset(micro_data,
                       seurat_clusters %in% c(0, 2, 3, 6))@meta.data

    #running gage analysis for Sham - IL2 vs GFP
    {
      #getting cell names and finding column indices
      sham_cells <- rownames(subset(metadata,
                                metadata$micro_treat_group == "Sham_PHP.GFAP-GFP"))
      tbi_cells <- rownames(subset(metadata,
                                metadata$micro_treat_group == "Sham_PHP.GFAP-IL2"))
      sham_cells <- which(colnames(micro_data[["SCT"]]@data) %in% sham_cells)
      tbi_cells <- which(colnames(micro_data[["SCT"]]@data) %in% tbi_cells)
    #extracting and formatting log counts matrix to minimise ram footprint
    counts_matrix <- micro_data[["SCT"]]@data[,c(sham_cells, tbi_cells)]
    counts_matrix <- as.data.frame(counts_matrix)
    gc()

    #mapping ids
    # do check that the order is the same
    print(all(unique(rownames(counts_matrix) == genes$SYMBOL)))
    counts_matrix$ENTREZID <- genes$ENTREZID
    counts_matrix <- counts_matrix[complete.cases(counts_matrix),]
    rownames(counts_matrix) <- counts_matrix$ENTREZID
    counts_matrix$ENTREZID <- NULL

    #running gage
    gage_results <- gage(exprs = counts_matrix,
                         #exprs = data.matrix(gse16873),
                         gsets = kegg_genesets$kg.sets,
                         #gsets = kegg.gs,
                         ref = (1:length(sham_cells)),
                         samp = ((length(sham_cells)+1) : ncol(counts_matrix)),
                         same.dir = F,
                         compare = "as.group",
                         set.size = c(15, 1000)
                         )
    gc()

    saveRDS(object = gage_results,
            file = paste0(loc_pathview_results, "micro/homeo/gage_results_Sham.rds"),
            compress = T)

    #creating the essential group for dotplot later
    essential_group <-
      esset.grp(setp = gage_results$greater,
                exprs = counts_matrix,
                gsets = kegg_genesets$kg.sets,
                ref = (1:length(sham_cells)),
                samp = ((length(sham_cells)+1) : ncol(counts_matrix)),
                test4up = F,
                same.dir = F,
                make.plot = T,
                compare = "by.group",
                cutoff = 0.01,
                use.q = T)
    gc()

    saveRDS(object = essential_group,
            file = paste0(loc_pathview_results, "micro/homeo/esset_grp_Sham.rds"),
            compress = T)

  }
  #running gage analysis for TBI - IL2 vs GFP
  {
    #getting cell names and finding column indices
```

```r
sham_cells <- rownames(subset(metadata,
                             metadata$micro_treat_group == "TBI_PHP.GFAP-GFP"))
tbi_cells <- rownames(subset(metadata,
                             metadata$micro_treat_group == "TBI_PHP.GFAP-IL2"))
sham_cells <- which(colnames(micro_data[["SCT"]]@data) %in% sham_cells)
tbi_cells <- which(colnames(micro_data[["SCT"]]@data) %in% tbi_cells)

#extracting and formatting log counts matrix to minimise ram footprint
counts_matrix <- micro_data[["SCT"]]@data[,c(sham_cells, tbi_cells)]
counts_matrix <- as.data.frame(counts_matrix)
gc()

#mapping ids
# do check that the order is the same
print(all(unique(rownames(counts_matrix) == genes$SYMBOL)))
counts_matrix$ENTREZID <- genes$ENTREZID
counts_matrix <- counts_matrix[complete.cases(counts_matrix),]
rownames(counts_matrix) <- counts_matrix$ENTREZID
counts_matrix$ENTREZID <- NULL

#running gage
gage_results <- gage(exprs = counts_matrix,
                     #exprs = data.matrix(gse16873),
                     gsets = kegg_genesets$kg.sets,
                     #gsets = kegg.gs,
                     ref = (1:length(sham_cells)),
                     samp = ((length(sham_cells)+1) : ncol(counts_matrix)),
                     same.dir = F,
                     compare = "as.group",
                     set.size = c(15, 1000)
                     )
gc()

saveRDS(object = gage_results,
        file = paste0(loc_pathview_results, "micro/homeo/gage_results_TBI.rds"),
        compress = T)

#creating the essential group for dotplot later
essential_group <-
  esset.grp(setp = gage_results$greater,
            exprs = counts_matrix,
            gsets = kegg_genesets$kg.sets,
            ref = (1:length(sham_cells)),
            samp = ((length(sham_cells)+1) : ncol(counts_matrix)),
            test4up = F,
            same.dir = F,
            make.plot = T,
            compare = "by.group",
            cutoff = 0.01,
            use.q = T)
gc()

saveRDS(object = essential_group,
            file = paste0(loc_pathview_results, "micro/homeo/esset_grp_TBI.rds"),
            compress = T)

  }
}

#activated microglia
{
  #loading metadata
  metadata <- subset(micro_data,
                     seurat_clusters %in% c(1, 4))@meta.data

  #running gage analysis for Sham - IL2 vs GFP
  {
    #getting cell names and finding column indices
    sham_cells <- rownames(subset(metadata,
                                  metadata$micro_treat_group == "Sham_PHP.GFAP-GFP"))
    tbi_cells <- rownames(subset(metadata,
                                 metadata$micro_treat_group == "Sham_PHP.GFAP-IL2"))
    sham_cells <- which(colnames(micro_data[["SCT"]]@data) %in% sham_cells)
    tbi_cells <- which(colnames(micro_data[["SCT"]]@data) %in% tbi_cells)

    #extracting and formatting log counts matrix to minimise ram footprint
    counts_matrix <- micro_data[["SCT"]]@data[,c(sham_cells, tbi_cells)]
    counts_matrix <- as.data.frame(counts_matrix)
    gc()

    #mapping ids
    # do check that the order is the same
    print(all(unique(rownames(counts_matrix) == genes$SYMBOL)))
    counts_matrix$ENTREZID <- genes$ENTREZID
    counts_matrix <- counts_matrix[complete.cases(counts_matrix),]
    rownames(counts_matrix) <- counts_matrix$ENTREZID
    counts_matrix$ENTREZID <- NULL

    #running gage
    gage_results <- gage(exprs = counts_matrix,
                         gsets = kegg_genesets$kg.sets,
                         ref = (1:length(sham_cells)),
                         samp = ((length(sham_cells)+1) : ncol(counts_matrix)),
                         same.dir = F,
                         compare = "as.group",
                         set.size = c(15, 1000)
                         )
    gc()

    saveRDS(object = gage_results,
            file = paste0(loc_pathview_results,
                          "micro/activated/gage_results_Sham.rds"),
            compress = T)

    #creating the essential group for dotplot later
    essential_group <-
```

```
        esset.grp(setp = gage_results$greater,
                  exprs = counts_matrix,
                  gsets = kegg_genesets$kg.sets,
                  ref = (1:length(sham_cells)),
                  samp = ((length(sham_cells)+1) : ncol(counts_matrix)),
                  test4up = F,
                  same.dir = F,
                  make.plot = T,
                  compare = "by.group",
                  cutoff = 0.01,
                  use.q = T)
      gc()

      saveRDS(object = essential_group,
              file = paste0(loc_pathview_results, "micro/activated/esset_grp_Sham.rds"),
              compress = T)

    }
    #running gage analysis for TBI - IL2 vs GFP
    {
      #getting cell names and finding column indices
      sham_cells <- rownames(subset(metadata,
                                    metadata$micro_treat_group == "TBI_PHP.GFAP-GFP"))
      tbi_cells <- rownames(subset(metadata,
                                    metadata$micro_treat_group == "TBI_PHP.GFAP-IL2"))
      sham_cells <- which(colnames(micro_data[["SCT"]]@data) %in% sham_cells)
      tbi_cells <- which(colnames(micro_data[["SCT"]]@data) %in% tbi_cells)

      #extracting and formatting log counts matrix to minimise ram footprint
      counts_matrix <- micro_data[["SCT"]]@data[,c(sham_cells, tbi_cells)]
      counts_matrix <- as.data.frame(counts_matrix)
      gc()

      #mapping ids
      # do check that the order is the same
      print(all(unique(rownames(counts_matrix) == genes$SYMBOL)))
      counts_matrix$ENTREZID <- genes$ENTREZID
      counts_matrix <- counts_matrix[complete.cases(counts_matrix),]
      rownames(counts_matrix) <- counts_matrix$ENTREZID
      counts_matrix$ENTREZID <- NULL

      #running gage
      gage_results <- gage(exprs = counts_matrix,
                           gsets = kegg_genesets$kg.sets,
                           ref = (1:length(sham_cells)),
                           samp = ((length(sham_cells)+1) : ncol(counts_matrix)),
                           same.dir = F,
                           compare = "as.group",
                           set.size = c(15, 1000)
                           )
      gc()

      saveRDS(object = gage_results,
            file = paste0(loc_pathview_results, "micro/activated/gage_results_TBI.rds"),
            compress = T)

    #creating the essential group for dotplot later
    essential_group <-
      esset.grp(setp = gage_results$greater,
                exprs = counts_matrix,
                gsets = kegg_genesets$kg.sets,
                ref = (1:length(sham_cells)),
                samp = ((length(sham_cells)+1) : ncol(counts_matrix)),
                test4up = F,
                same.dir = F,
                make.plot = T,
                compare = "by.group",
                cutoff = 0.01,
                use.q = T)
    gc()

    saveRDS(object = essential_group,
            file = paste0(loc_pathview_results, "micro/activated/esset_grp_TBI.rds"),
            compress = T)

  }
}

#both microglia
{
  #running pathview analysis on treatment and age results
  {
    #loading homeo Sham results
    homeo_gage_results_sham <-
      readRDS(file = paste0(loc_pathview_results,
                            "micro/homeo/gage_results_Sham.rds"))
    homeo_diff_exprs_results_sham <-
      readRDS(file = paste0(loc_diff_exprs_results,
                            "micro/micro_homeo_markers_list_Sham.rds"))[[2]]
    homeo_diff_exprs_results_sham <-
      homeo_diff_exprs_results_sham[
        homeo_diff_exprs_results_sham$p_val_adj < 0.1, "avg_log2FC", drop = F]

    #loading homeo TBI results
    homeo_gage_results_tbi <- readRDS(file = paste0(loc_pathview_results,
                                                    "micro/homeo/gage_results_TBI.rds"))
    homeo_diff_exprs_results_tbi <-
      readRDS(file = paste0(loc_diff_exprs_results,
                            "micro/micro_homeo_markers_list_TBI.rds"))[[2]]
    homeo_diff_exprs_results_tbi <-
      homeo_diff_exprs_results_tbi[
        homeo_diff_exprs_results_tbi$p_val_adj < 0.1, "avg_log2FC", drop = F]

    #loading activated Sham results
    activ_gage_results_sham <- readRDS(file = paste0(loc_pathview_results,
                                                     "micro/activated/gage_results_Sham.rds"))
```

```r
activ_diff_exprs_results_sham <-
  readRDS(file = paste0(loc_diff_exprs_results,
                        "micro/micro_activated_markers_list_Sham.rds"))[[2]]
activ_diff_exprs_results_sham <-
  activ_diff_exprs_results_sham[
    activ_diff_exprs_results_sham$p_val_adj < 0.1, "avg_log2FC", drop = F]

#loading activated TBI results
activ_gage_results_tbi <- readRDS(file = paste0(loc_pathview_results,
                                          "micro/activated/gage_results_TBI.rds"))
activ_diff_exprs_results_tbi <-
  readRDS(file = paste0(loc_diff_exprs_results,
                        "micro/micro_activated_markers_list_TBI.rds"))[[2]]
activ_diff_exprs_results_tbi <-
  activ_diff_exprs_results_tbi[
    activ_diff_exprs_results_tbi$p_val_adj < 0.1, "avg_log2FC", drop = F]

#merging diff exprs data
diff_exprs_results <- merge(x = homeo_diff_exprs_results_sham,
                            y = homeo_diff_exprs_results_tbi,
                            by = "row.names",
                            all = T)
diff_exprs_results_2 <- merge(x = activ_diff_exprs_results_sham,
                            y = activ_diff_exprs_results_tbi,
                            by = "row.names",
                            all = T)
diff_exprs_results <- merge(x = diff_exprs_results,
                            y = diff_exprs_results_2,
                            by = "Row.names",
                            all = T)
rownames(diff_exprs_results) <- diff_exprs_results$Row.names
diff_exprs_results$Row.names <- NULL
diff_exprs_results_2 <- NULL
colnames(diff_exprs_results) <- paste0("Contrast ",
                                       1:ncol(diff_exprs_results))

#setting up pathview
data(gene.idtype.bods)
work.dir <- getwd()
path <- paste0(loc_pathview_results, "micro/both/pathways")
dir.create(path = path, recursive = T)
setwd(path)

#writing out the gage results
write.table(x = homeo_gage_results_sham$greater,
            file = "homeo_gage_results_sham.tab",
            quote = F,
            sep = "\t",
            row.names = T)
write.table(x = homeo_gage_results_tbi$greater,
            file = "homeo_gage_results_tbi.tab",
            quote = F,
            sep = "\t",
                row.names = T)
write.table(x = activ_gage_results_sham$greater,
                file = "activ_gage_results_sham.tab",
                quote = F,
                sep = "\t",
                row.names = T)
write.table(x = activ_gage_results_tbi$greater,
                file = "activ_gage_results_tbi.tab",
                quote = F,
                sep = "\t",
                row.names = T)

#getting pathway ids from the gage results
homeo_pathway_ids_sham <-
  homeo_gage_results_sham$greater[, "p.val"] < 0.01 &
  !is.na(homeo_gage_results_sham$greater[, "p.val"])
homeo_pathway_ids_tbi <-
  homeo_gage_results_tbi$greater[, "p.val"] < 0.01 &
  !is.na(homeo_gage_results_tbi$greater[, "p.val"])
activ_pathway_ids_sham <-
  activ_gage_results_sham$greater[, "p.val"] < 0.01 &
  !is.na(activ_gage_results_sham$greater[, "p.val"])
activ_pathway_ids_tbi <-
  activ_gage_results_tbi$greater[, "p.val"] < 0.01 &
  !is.na(activ_gage_results_tbi$greater[, "p.val"])
pathway_ids <-
  c(rownames(homeo_gage_results_sham$greater)[homeo_pathway_ids_sham],
    rownames(homeo_gage_results_tbi$greater)[homeo_pathway_ids_tbi],
    rownames(activ_gage_results_sham$greater)[activ_pathway_ids_sham],
    rownames(activ_gage_results_tbi$greater)[activ_pathway_ids_tbi])
pathway_ids <- gsub(pattern = " .*$",
                    replacement = "",
                    x = pathway_ids,
                    ignore.case = T)
pathway_ids <- unique(pathway_ids)

#manually adding missing id
pathway_ids <- c(pathway_ids, "mmu05022")

#running pathview
pathview(gene.data = diff_exprs_results,
         cpd.data = NULL,
         pathway.id = pathway_ids,
         species = "mouse",
         kegg.dir = path,
         gene.idtype = "SYMBOL",
         low = list(gene = "#2a7fff", cpd = "blue"),
         mid = list(gene = "white", cpd = "gray"),
         high = list(gene = "#ff5555", cpd = "yellow"),
         plot.col.key= F, #hide plot colour key
         limit = list(gene = 1.5, cpd = 1),
         bins = list(gene = 20, cpd = 10),
         kegg.native = T,
```

```r
            na.col = "white",
            same.layer = F)

    #resetting work dir
    setwd(work.dir)

  }
  #creating dotplot visualisation -- micro treatment and age
  {
    #loading homeo essential groups
    homeo_esset_sham <- readRDS(file = paste0(loc_pathview_results,
                                              "micro/homeo/esset_grp_Sham.rds"))
    homeo_esset_tbi <- readRDS(file = paste0(loc_pathview_results,
                                             "micro/homeo/esset_grp_TBI.rds"))

    #loading activated essential groups
    activ_esset_sham <- readRDS(file = paste0(loc_pathview_results,
                                              "micro/activated/esset_grp_Sham.rds"))
    activ_esset_tbi <- readRDS(file = paste0(loc_pathview_results,
                                             "micro/activated/esset_grp_TBI.rds"))

    #making a list for the essential groups
    gene_clusters <- list(homeo_esset_sham,
                          homeo_esset_tbi,
                          activ_esset_sham,
                          activ_esset_tbi)
    names(gene_clusters) <- c("Homeostatic\nSham",
                              "Homeostatic\nTBI",
                              "Activated\nSham",
                              "Activated\nTBI")

    #reducing all the essential results into respective core gene sets
    for(i in seq(1, length(gene_clusters))){
      gene_clusters[[i]] <- Reduce(f = c,
                                   x = gene_clusters[[i]]$coreGeneSets)
    }

    #performing the compare cluster comparison
    ck <-
      compareCluster(geneClusters = gene_clusters,
                     fun = "enrichKEGG",
                     organism = "mouse",
                     use_internal_data = F)

    ck_dataframe <- as.data.frame(ck)
    ck_dataframe$Cluster <- as.character(ck_dataframe$Cluster)
    for(i in seq(1, nrow(ck_dataframe))){
      ck_genes <- unlist(strsplit(x = ck_dataframe$geneID[i], split = "/"))
      ck_genes <- genes$SYMBOL[which(genes$ENTREZID %in% ck_genes)]
      ck_genes <- paste(ck_genes, collapse = " / ")

      ck_dataframe$geneID[i] <- ck_genes
```

```
        ck_dataframe$Cluster[i] <- gsub(pattern = "\n",
                                        replacement = " ",
                                        x = as.character(ck_dataframe$Cluster[i]),
                                        ignore.case = T)
      }

      write.table(x = ck_dataframe,
                  file = paste0(loc_pathview_results,
                                "micro/both/compare_cluster_results.tab"),
                  quote = F, sep = "\t", row.names = F, col.names = T)

    }
  }
}
```