Geneset over-representation analysis of single-cell RNA-seq clusters with gsfisher

Stephen N. Sansom

08 April, 2020

This vignette describes how the gsfisher package can be used to test for over-representation of Gene Ontology (GO) categories, KEGG pathways and arbitrary genesets (including xCell cell type annotations) amongst the markers of single cell clusters.

Identifying cluster marker genes using Seurat

We demonstrate the approach using data from the Seurat 3K PBMC tutorial. First we load the required R libraries.

```
library(dplyr)
library(kableExtra)
library(knitr)
library(Seurat)
library(patchwork)
library(gsfisher)
```

Here we run a minimal set of commands to obtain the cluster markers for this dataset:

```
if(!file.exists("filtered_gene_bc_matrices/hg19/matrix.mtx"))
{
  download.file("https://s3-us-west-2.amazonaws.com/10x.files/samples/cell/pbmc3k/pbmc3k_filtered_gene_
 untar("pbmc3k_filtered_gene_bc_matrices.tar.gz")
}
if(!file.exists("pbmc.markers.rds") | !file.exists("pbmc.rds"))
{
  pbmc.data <- Read10X(data.dir = "filtered_gene_bc_matrices/hg19/")

  pbmc <- CreateSeuratObject(counts = pbmc.data, project = "pbmc3k",
                             min.cells = 3, min.features = 200)

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

  pbmc <- subset(pbmc, subset = nFeature_RNA > 200
                 & nFeature_RNA < 2500 & percent.mt < 5)

  pbmc <- NormalizeData(pbmc)

  pbmc <- FindVariableFeatures(pbmc, selection.method = "vst",
                               nfeatures = 2000)

  all.genes <- rownames(pbmc)
```

```
  pbmc <- ScaleData(pbmc, features = all.genes)

  pbmc <- RunPCA(pbmc, features = VariableFeatures(object = pbmc))

  pbmc <- FindNeighbors(pbmc, dims = 1:10)

  pbmc <- FindClusters(pbmc, resolution = 0.5)

  pbmc.markers <- FindAllMarkers(pbmc, only.pos = TRUE,
                                 min.pct = 0.25, logfc.threshold = 0.25)

  saveRDS(pbmc, "pbmc.rds")
  saveRDS(pbmc.markers, "pbmc.markers.rds")
}
pbmc <- readRDS("pbmc.rds")
pbmc.markers <- readRDS("pbmc.markers.rds")
```

We now have a table containing the positive marker genes for each cluster.

Defining the gene universe

Before we can test the lists of cluster markers for geneset over-representation we must define a suitable background list of expressed genes. This is also sometimes referred to as the gene universe.

Here, to construct the background list we select genes that are expressed in the same fraction of cells as was required for inclusion in the testing with the Seurat FindAllMarkers function (i.e. the min.pct parameter).

In this case, we need to identify the set of genes which are expressed in 25% of the cluster cells, or 25% of the other cells (for any the clusters).

```
# Helper function for extracting the expressed genes from a seurat object
getExpressedGenesFromSeuratObject <- function(seurat_object,
                                      clusters,
                                      min.pct=0.1)
{
  expressed <- c()
  for(cluster in clusters)
  {
    # get genes detected in the cluster
    cluster_cells <- names(pbmc$seurat_clusters[pbmc$seurat_clusters==cluster])
    clust_pcts <- apply(pbmc@assays$RNA@counts[,cluster_cells],
           1, function(x) sum(x>0)/length(x))

    detected_in_clust <- names(clust_pcts[clust_pcts>min.pct])

    # get genes detected in the other cells
    other_cells <- names(pbmc$seurat_clusters[pbmc$seurat_clusters!=cluster])
    other_pcts <- apply(pbmc@assays$RNA@counts[,other_cells],
                        1, function(x) sum(x>0)/length(x))

    detected_in_other_cells <- names(other_pcts[other_pcts>min.pct])

    expressed <- c(expressed, detected_in_clust, detected_in_other_cells)
  }
```

```
  expressed <- unique(expressed)
  expressed
}

expressed_genes <- getExpressedGenesFromSeuratObject(
  pbmc,unique(pbmc$seurat_clusters), min.pct=0.25)
```

Getting gene annotations

Most of the genesets and pathways we are interested in are mapped to Entrez gene identifiers. In order to map the gene symbols (which are used by Seurat) to Entrez identifiers we use the "fetchAnnotation()" function from gsfisher to retrieve a table of gene annotations.

```
if(!file.exists("annotation.rds"))
{
  annotation <- fetchAnnotation(species="hs",
                                ensembl_version=NULL,
                                ensembl_host = NULL)
 saveRDS(annotation, "annotation.rds")
}

annotation <- readRDS("annotation.rds")
```

Next we add the Entrez ids to the FindAllMarkers() results table and also translate the background gene symbols to entrez ids.

```
pbmc.markers$entrez_id <- as.character(annotation$entrez_id[
  match(pbmc.markers$gene, annotation$gene_name)])

pbmc.markers <- pbmc.markers[!is.na(pbmc.markers$entrez_id),]

background_entrez <- as.character(annotation$entrez_id[
  match(expressed_genes, annotation$gene_name)])

background_entrez <- background_entrez[!is.na(background_entrez)]
```

Testing for Gene Ontology (GO) category enrichments

We can now use the gsfisher "runGO.all()" function to test for GO category enrichments. Note that before doing so we filter the markers table to the level of significance (after multiple testing correction) that we are interested in.

```
pbmc.markers.filtered <- pbmc.markers[pbmc.markers$p_val_adj < 0.05,]

if(!file.exists("go.results.rds"))
{
go.results <- runGO.all(results=pbmc.markers.filtered,
                        species = "hs",
                        background_ids = background_entrez,
                        gene_id_col="entrez_id",
                        gene_id_type="entrez",
                        sample_col="cluster",
                        p_col="p_val_adj",
```

```
                          p_threshold=0.05)
saveRDS(go.results, "go.results.rds")
}
go.results <- readRDS("go.results.rds")
```

We now have a results table that contains odds ratios and p-values for all GO categories from the biological process (BP), cellular component (CC) and molecular function (MF) sub-ontologies. Here the first 3 rows of the table are shown:

	geneset_id	description	ontology	fg_freq	bg_freq	n_fg	n_bg	n_set	p.val	odds.ratio	gene_names	entrez_ids	cluster
2282	GO:0022626	cytosolic ribosome	CC	0.5045872	0.0377724	55	78	158	0	84.67752	RPSA, RPL10A, RPL3, RPL4, RPL5, RPL7, RPL9, RPL11, RPL13, RPL17, RPL18, RPL19, RPL21, RPL22, RPL23A, RPL24, RPL30, RPL27A, RPL31, RPL32, RPL34, RPL35A, RPL37, RPL38, RPL36A, RPLP0, RPLP2, RPS3, RPS3A, RPS4X, RPS4Y1, RPS5, RPS6, RPS8, RPS10, RPS12, RPS13, RPS14, RPS15, RPS15A, RPS16, RPS18, RPS20, RPS21, RPS23, RPS25, RPS26, RPS27, RPS27A, RPS28, RPS29, RPL14, RPL35, RPL13A, RPL36	3921, 4736, 6122, 6124, 6125, 6129, 6133, 6135, 6137, 6139, 6141, 6143, 6144, 6146, 6147, 6152, 6156, 6157, 6160, 6161, 6164, 6165, 6167, 6169, 6173, 6175, 6181, 6188, 6189, 6191, 6192, 6193, 6194, 6202, 6204, 6206, 6207, 6208, 6209, 6210, 6217, 6222, 6224, 6227, 6228, 6230, 6231, 6232, 6233, 6234, 6235, 9045, 11224, 23521, 25873	0
297	GO:0006614	SRP-dependent cotranslational protein targeting to membrane	BP	0.5045872	0.0406780	55	84	105	0	67.13248	RPSA, RPL10A, RPL3, RPL4, RPL5, RPL7, RPL9, RPL11, RPL13, RPL17, RPL18, RPL19, RPL21, RPL22, RPL23A, RPL24, RPL30, RPL27A, RPL31, RPL32, RPL34, RPL35A, RPL37, RPL38, RPL36A, RPLP0, RPLP2, RPS3, RPS3A, RPS4X, RPS4Y1, RPS5, RPS6, RPS8, RPS10, RPS12, RPS13, RPS14, RPS15, RPS15A, RPS16, RPS18, RPS20, RPS21, RPS23, RPS25, RPS26, RPS27, RPS27A, RPS28, RPS29, RPL14, RPL35, RPL13A, RPL36	3921, 4736, 6122, 6124, 6125, 6129, 6133, 6135, 6137, 6139, 6141, 6143, 6144, 6146, 6147, 6152, 6156, 6157, 6160, 6161, 6164, 6165, 6167, 6169, 6173, 6175, 6181, 6188, 6189, 6191, 6192, 6193, 6194, 6202, 6204, 6206, 6207, 6208, 6209, 6210, 6217, 6222, 6224, 6227, 6228, 6230, 6231, 6232, 6233, 6234, 6235, 9045, 11224, 23521, 25873	0
13	GO:0000184	nuclear-transcribed mRNA catabolic process, nonsense-mediated decay	BP	0.5045872	0.0421308	55	87	144	0	60.71800	RPSA, RPL10A, RPL3, RPL4, RPL5, RPL7, RPL9, RPL11, RPL13, RPL17, RPL18, RPL19, RPL21, RPL22, RPL23A, RPL24, RPL30, RPL27A, RPL31, RPL32, RPL34, RPL35A, RPL37, RPL38, RPL36A, RPLP0, RPLP2, RPS3, RPS3A, RPS4X, RPS4Y1, RPS5, RPS6, RPS8, RPS10, RPS12, RPS13, RPS14, RPS15, RPS15A, RPS16, RPS18, RPS20, RPS21, RPS23, RPS25, RPS26, RPS27, RPS27A, RPS28, RPS29, RPL14, RPL35, RPL13A, RPL36	3921, 4736, 6122, 6124, 6125, 6129, 6133, 6135, 6137, 6139, 6141, 6143, 6144, 6146, 6147, 6152, 6156, 6157, 6160, 6161, 6164, 6165, 6167, 6169, 6173, 6175, 6181, 6188, 6189, 6191, 6192, 6193, 6194, 6202, 6204, 6206, 6207, 6208, 6209, 6210, 6217, 6222, 6224, 6227, 6228, 6230, 6231, 6232, 6233, 6234, 6235, 9045, 11224, 23521, 25873	0

Filtering for categories of interest

We next filter the results table to discard genesets that we not interested in. In order to maintain statistical rigour these criteria should be decided in advance.

To filter the results table we use the gsfisher "filterGenesets()" convenience function which will also correct the p-values for multiple testing after the filtering.

Here we are only interested in the biological processes, GO categories that have at least 3 genes in the foreground list, GO categories that have no more than 500 genes (to avoid generic enrichments) and GO categories that have an over-representation odds ratio of at least 2 (we are not interested in small effect sizes).

```
go.results.filtered <- go.results[go.results$ontology=="BP",]

go.results.filtered <- filterGenesets(go.results.filtered,
                                      min_foreground_genes = 3,
                                      max_genes_geneset = 500,
                                      min_odds_ratio = 2,
                                      p_col = "p.val",
                                      padjust_method = "BH",
                                      use_adjusted_pvalues = TRUE,
                                      pvalue_threshold = 0.05)
```

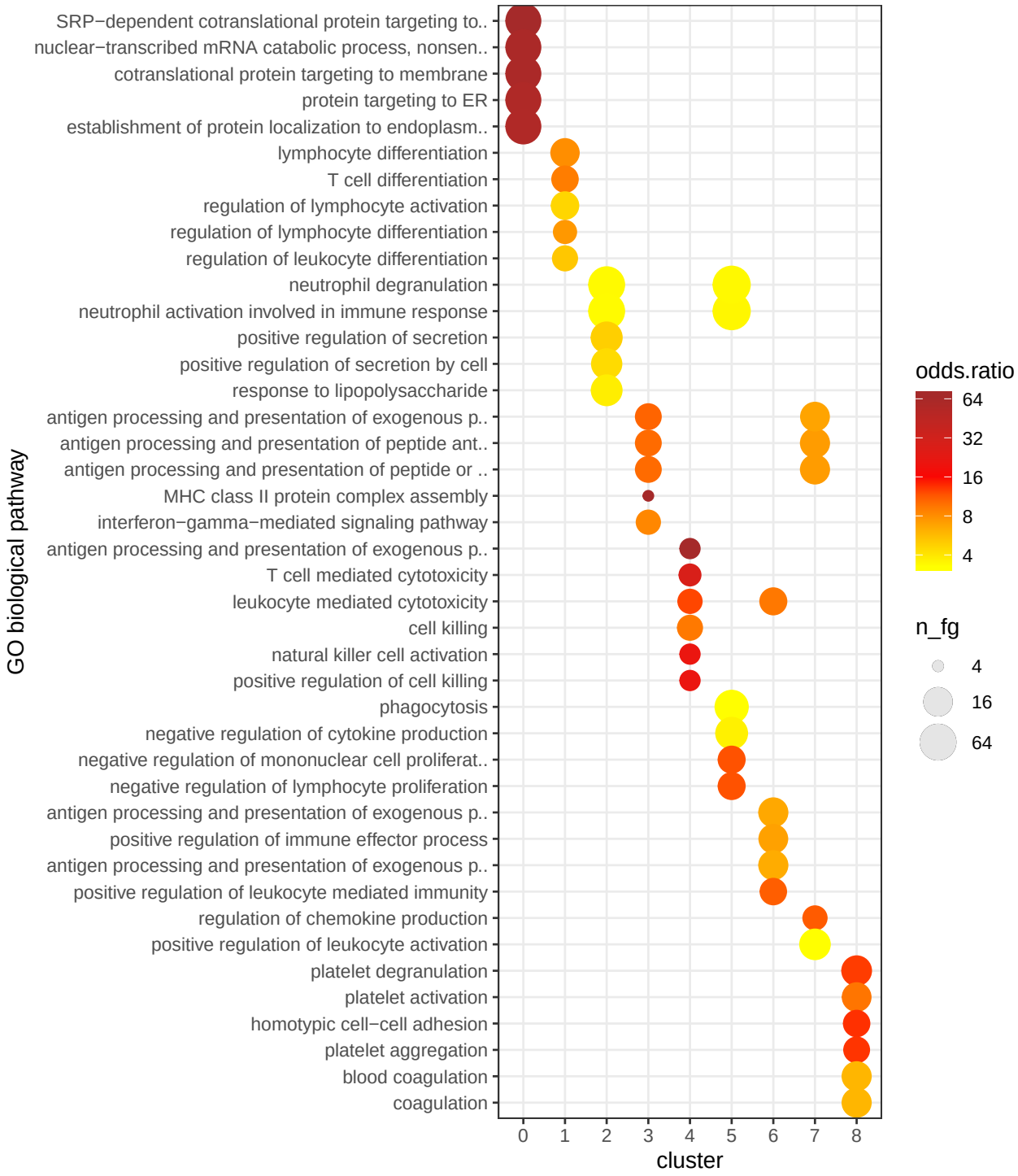
```
## [1] 810  14
```

Visualising the results

We can use the gsfisher "sampleEnrichmentDotplot()" function to make a dotplot showing the enrichments of the top 5 most significant categories by cluster. Here dot size is proportional to the number of genes in the category that were found as cluster markers (n_fg). The dot color is used to communicate the effect size (odds ratio).

```
go.results.top <- go.results.filtered %>%
                    group_by(cluster) %>%
                    top_n(n=5, -p.val)

sampleEnrichmentDotplot(go.results.top,
                        selection_col = "description",
                        selected_genesets = unique(go.results.top$description),
                        sample_id_col = "cluster",
                        fill_var = "odds.ratio",
                        maxl=50,
                        title="GO biological pathway")
```

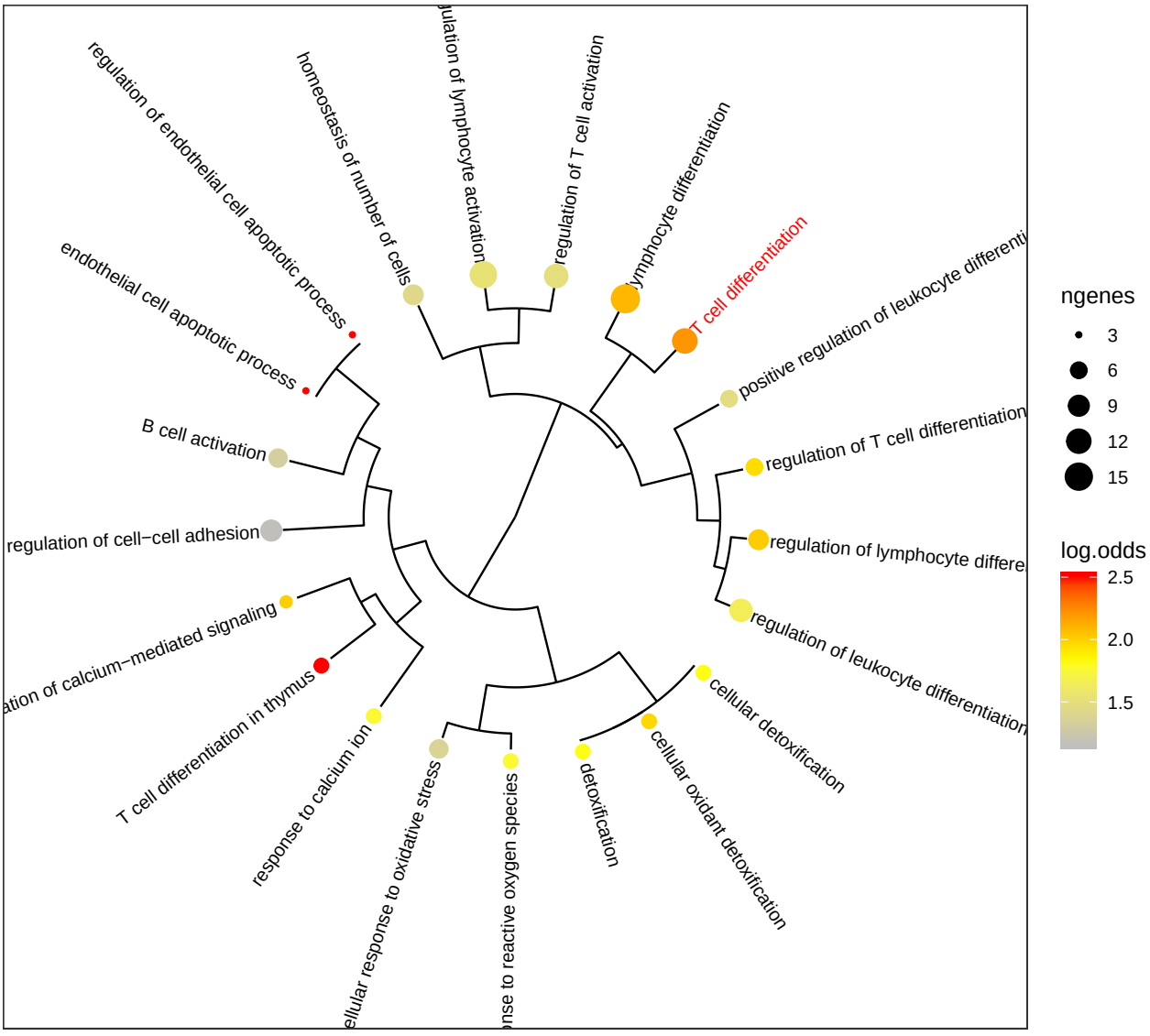


Exploring the over-represented categories for a cluster

Sometimes many semi-redundant categories will be enriched for a cluster. To help understand the results we can cluster the enriched categories according to the similarity of their gene membership and visualise the resulting dendrogram using the gsfisher "visualiseClusteredGenesets()" function.

```
go.results.filtered.1 <- go.results.filtered[go.results.filtered$cluster==1,]

visualiseClusteredGenesets(go.results.filtered.1,
                           geneset_clust = NULL,
                           odds_ratio_max_quantile = 0.9,
                           highlight = c("T cell differentiation"),
                           id_col = "geneset_id",
                           name_col = "description",
                           ngenes_col = "n_fg",
                           odds_ratio_col = "odds.ratio")
```



Testing for over-represented Kegg pathways

To test for overenriched KEGG pathways we can use the gsfisher "runKEGG.all()" function.

```
if(!file.exists("kegg.results.rds"))
{
```

```
kegg.results <- runKEGG.all(results=pbmc.markers.filtered,
                species = "hs",
                background_ids = background_entrez,
                gene_id_col="entrez_id",
                gene_id_type="entrez",
                sample_col="cluster",
                p_col="p_val_adj",
                p_threshold=0.05)
saveRDS(kegg.results,"kegg.results.rds")
}
kegg.results <- readRDS("kegg.results.rds")

kegg.results.filtered <- filterGenesets(kegg.results)
```

```
## [1] 231  13
```

```
kegg.results.top  <- kegg.results.filtered  %>%
                         group_by(cluster) %>%
                         top_n(n=5, -p.val)

sampleEnrichmentDotplot(kegg.results.top,
                        selected_genesets = unique(kegg.results.top$description),
                        maxl=50,
                        title="KEGG pathway")
```

Testing for over-represented xCell cell type markers

We can use the genesets from the xCell package to help identify the cell types. First we need to extract the genesets from the xCell R package and translate them to entrez ids.

```
library(devtools)
library(GSEABase)
```

```
# Install the xCell R package
devtools::install_github('dviraran/xCell')

library(xCell)

# Export the xCell genesets as a gmt file
xx <- as.list(xCell.data)
toGmt(xx$signatures,"xcell.gmt")

# Clean up the xCell geneset names
xcell <- readGMT("xcell.gmt")
names(xcell) <- gsub("%","_",names(xcell))

# Translate the lists to entrez ids
xcell_entrez <- list()
for(gs in names(xcell))
{
  symbols <- xcell[[gs]]
  entrez_ids <- unique(annotation$entrez_id[match(symbols,
                                                  annotation$gene_name)])
  entrez_ids <- entrez_ids[!is.na(entrez_ids)]
  xcell_entrez[[gs]] <- as.character(entrez_ids)
}

writeGMT(xcell_entrez, "xcell_entrez.gmt")
```

```
## [1] 0
```

Now we can test for enrichment of the xCell genesets in the cluster markers. Note that the gsfisher "runGMT.all()" function can be used to test for enrichments of any genesets in the gene matrix transposed (GMT) format. More details on this format can be found here.

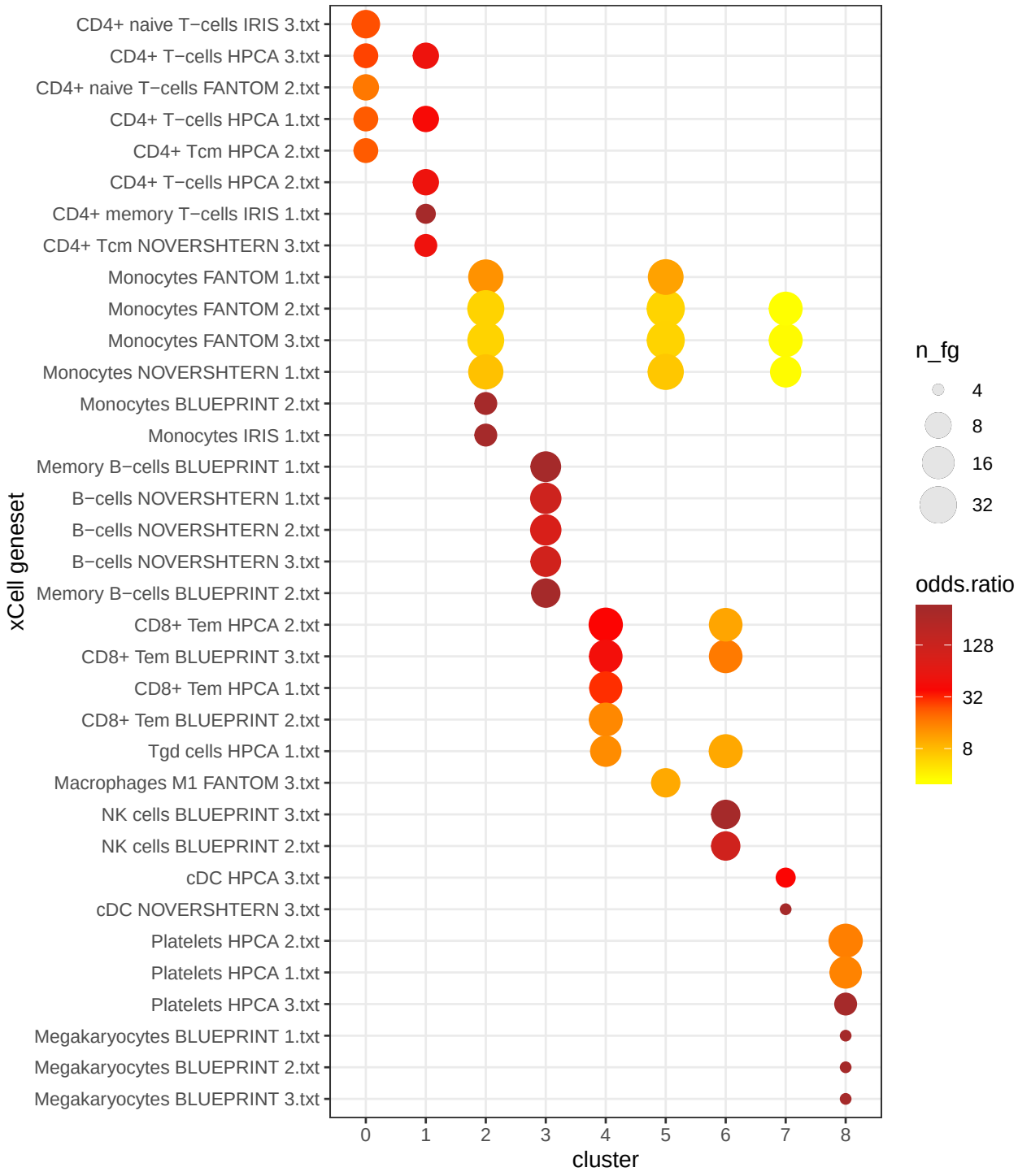
```
# run the enricment tests. This function can be used to test arbitrary gmt files
xcell.results <- runGMT.all(results=pbmc.markers.filtered,
                            species = "hs",
                            background_ids = background_entrez,
                            gene_id_col="entrez_id",
                            gene_id_type="entrez",
                            gmt_file="xcell_entrez.gmt",
                            sample_col="cluster",
                            p_col="p_val_adj",
                            p_threshold=0.05)

xcell.results.filtered <- filterGenesets(xcell.results)
```

```
## [1] 233  12
```

```
xcell.results.top <- xcell.results.filtered %>%
                          group_by(cluster) %>%
                          top_n(n=5, -p.val)

sampleEnrichmentDotplot(xcell.results.top,
                        selected_genesets = unique(xcell.results.top$geneset_id),
                        selection_col = "geneset_id",
                        maxl=50,
                        title="xCell geneset")
```


CD4+ naive T−cells IRIS 3.txt
CD4+ T−cells HPCA 3.txt
CD4+ naive T−cells FANTOM 2.txt
CD4+ T−cells HPCA 1.txt
CD4+ Tcm HPCA 2.txt
CD4+ T−cells HPCA 2.txt
CD4+ memory T−cells IRIS 1.txt
CD4+ Tcm NOVERSHTERN 3.txt
Monocytes FANTOM 1.txt
Monocytes FANTOM 2.txt
Monocytes FANTOM 3.txt
Monocytes NOVERSHTERN 1.txt
Monocytes BLUEPRINT 2.txt
Monocytes IRIS 1.txt
Memory B−cells BLUEPRINT 1.txt
B−cells NOVERSHTERN 1.txt
B−cells NOVERSHTERN 2.txt
B−cells NOVERSHTERN 3.txt
Memory B−cells BLUEPRINT 2.txt
CD8+ Tem HPCA 2.txt
CD8+ Tem BLUEPRINT 3.txt
CD8+ Tem HPCA 1.txt
CD8+ Tem BLUEPRINT 2.txt
Tgd cells HPCA 1.txt
Macrophages M1 FANTOM 3.txt
NK cells BLUEPRINT 3.txt
NK cells BLUEPRINT 2.txt
cDC HPCA 3.txt
cDC NOVERSHTERN 3.txt
Platelets HPCA 2.txt
Platelets HPCA 1.txt
Platelets HPCA 3.txt
Megakaryocytes BLUEPRINT 1.txt
Megakaryocytes BLUEPRINT 2.txt
Megakaryocytes BLUEPRINT 3.txt
xCell geneset
0
1
2
3
4
5
6
7
8
cluster
n_fg
4
8
16
32
odds.ratio
128
32
8