Querying PathCards Tutorial

Jinwoo Lee

Introduction

In this tutorial we will query the website PathCards with a ligand-receptor list to find pathways shared by each pair, score each pathway, and create a figure for the scored pathways.

For this example, we will be using the pbmc seurat object provided by Seurat's official website. The download link is (https://satijalab.org/seurat/vignettes.html).

Also, we will be using a sample list of receptor and ligand pairs. I have included this list in the vignette folder of my package.

We load up the required packages:

```
knitr::opts_chunk$set(echo = TRUE)
library(queryPathcards)
#Loading up the required packages by queryPathcards:
library(dplyr)
library(Seurat)
library(httr)
library(rvest)
library(stringr)
library(tidyr)
library(ggplot2)
library(reshape2)
```

Filtering The Ligand-Receptor List

We utilize the receptor.ligand.annotation.from.object() function to only select the ligand-receptor pairs that are significant from our sample ligand-receptor list.

The receptor.ligand.annotation.from.object() function has 5 parameters: * The Seurat object * Variables to group by * List of ligands and receptors * Percent expression threshold * Mean expression threshold

```
#Loading in the Seurat object downloaded from the website above:
sample_seurat_object <- readRDS("~/Downloads/pbmc3k_final.rds")

#Creating the sample ligand-receptor list. Be sure to name your columns as Ligand and Receptor.
sample_list <- read.table(file = "~/Downloads/receptor_ligand_sample_list.txt",
                          sep = "\t", header = TRUE)[1:500,]

sample_list_refined <- receptor.ligand.annotation.from.object(sample_seurat_object,
                                                              "seurat_clusters", sample_list, 0.1, 0.2)
```

```
## Using seurat_clusters as id variables

## Warning: The `printer` argument is deprecated as of rlang 0.3.0.
## This warning is displayed once per session.

## Using seurat_clusters as id variables
```

Querying

For this part, I'm going to create a sample ligand-receptor list and run it through the queryPathcards() function. For each ligand-receptor pair in the sample list, this function is going to query the PathCards website, find the pathways shared by the gene pair, and put it into a list.

The output will have two different data sets. The first data set, ligand_receptor_pathways, is going to be a dataframe that shows the pathways of each gene pair. The second data set, pathway_genes, is going to be a list that shows the list of genes in the pathway for each pathway.

Since queryPathcards() function only accepts matrix of column length 2, Ligand and Receptor, in that exact order, we have to change our sample_list_refined accordingly.

```
#Making sample_list_refined readable by the queryPathcards() function
sample_list_refined <- sample_list_refined[,c(2,4)]
sample_list_refined <- sample_list_refined[,c(2,1)]
colnames(sample_list_refined) <- c("Ligand", "Receptor")
sample_list_refined <- unique(sample_list_refined)

sample_result <- queryPathcards::queryPathcards(sample_list_refined)
#You can access each data set by utilizing:
#View(sample_result$ligand_receptor_pathways)
#View(sample_result$pathway_genes)
```

Scoring

Scoring the pathways created by the queryPathcards() function requires a Seurat object. The function scoreGenes() will require 3 variables: (1)the result by queryPathcards(), (2) Seurat object, and (3) variables to group by.

Running this function will output two data sets as well. The first is the raw data of the scores without any tidying up done. The second takes the average of each score within the categories set by the user (the 3rd variable in the function) and outputs the summarized data.

```
sample_scored <- scoreGenes(sample_result, sample_seurat_object, c("seurat_clusters"))
#Access these data sets by utilizing:
#View(sample_scored$full_data)
#View(sample_scored$summarized_data)
```

Creating figures

I created a function for creating a heatmap based on the scores. As of right now, this function will only accept the summarized data. To run the function, you need to put in (1) the scored data set, (2) the x-variable, and (3) the y-variable. The "fill=" is always Score_Average.

Tidying up the data will be different for each data set. For ours, we will arrange the data so that it is in descending order of the average scores of each pathway.

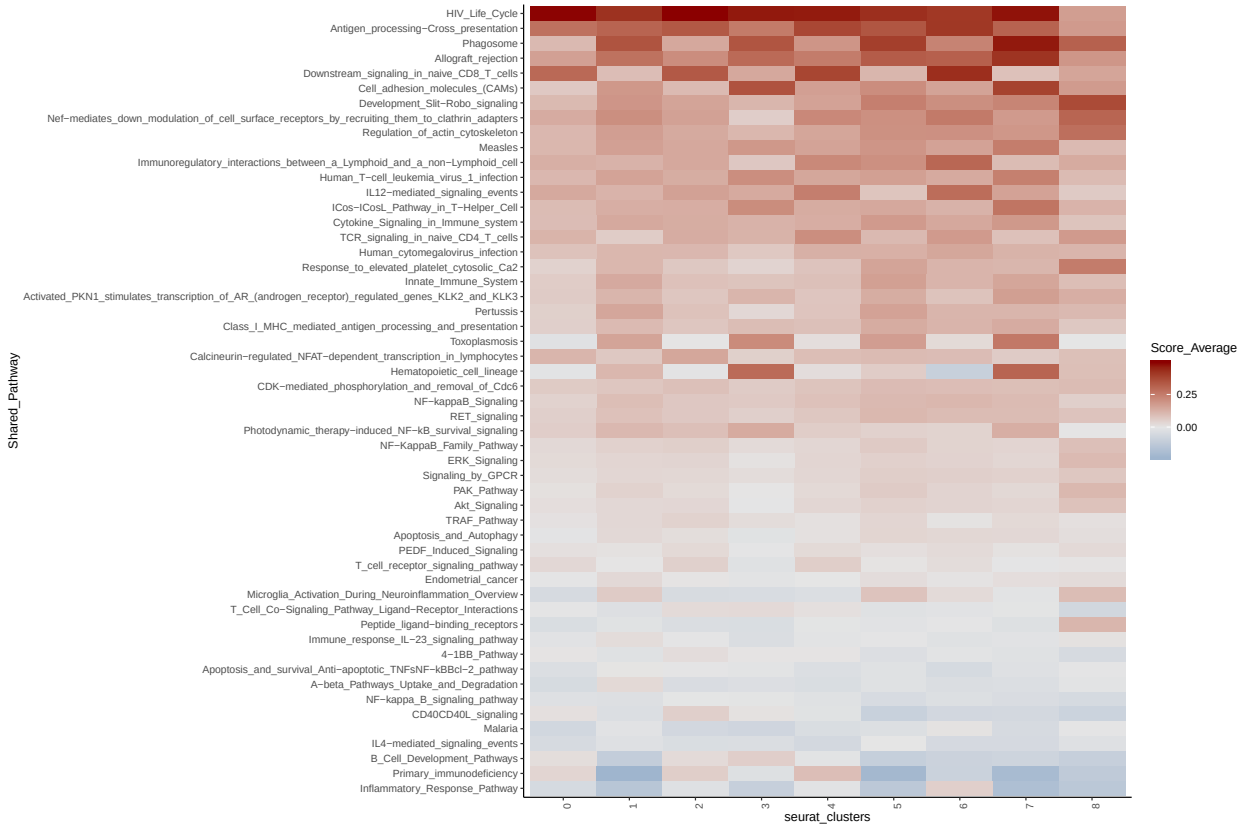
```
score_for_heatmap <- sample_scored$summarized_data %>%
  select(Shared_Pathway, Score_Average) %>%
  group_by(Shared_Pathway) %>%
  summarise_all(funs(mean)) %>%
  ungroup() %>%
```

```
  arrange(desc(Score_Average))

score_for_heatmap <- score_for_heatmap %>%
  mutate(id = as.numeric(rownames(score_for_heatmap))) %>%
  select(Shared_Pathway, id) %>%
  merge(sample_scored$summarized_data) %>%
  arrange(desc(id))

score_for_heatmap$Shared_Pathway <- factor(score_for_heatmap$Shared_Pathway,
                                           levels = unique(score_for_heatmap$Shared_Pathway))

#After organizing the data set, utilize this function to create the heatmap:
createHeatMap(score_for_heatmap, "Shared_Pathway", "seurat_clusters")
```




```
sessionInfo()
```

```
## R version 3.5.0 (2018-04-23)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS  10.14.6
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
```

```
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] bindrcpp_0.2.2       reshape2_1.4.3       ggplot2_3.2.0
##  [4] tidyr_0.8.1          stringr_1.3.1        rvest_0.3.4
##  [7] xml2_1.2.0           httr_1.3.1           Seurat_3.0.2
## [10] dplyr_0.7.5          queryPathcards_0.1.0
##
## loaded via a namespace (and not attached):
##  [1] tsne_0.1-3          nlme_3.1-137        bitops_1.0-6
##  [4] RColorBrewer_1.1-2  rprojroot_1.3-2     sctransform_0.2.0
##  [7] tools_3.5.0         backports_1.1.2     R6_2.2.2
## [10] irlba_2.3.3         KernSmooth_2.23-15  lazyeval_0.2.1
## [13] colorspace_1.3-2    withr_2.1.2         npsurv_0.4-0
## [16] tidyselect_0.2.4    gridExtra_2.3       curl_3.2
## [19] compiler_3.5.0      plotly_4.7.1        labeling_0.3
## [22] caTools_1.17.1      scales_0.5.0        lmtest_0.9-37
## [25] ggridges_0.5.1      pbapply_1.4-0       digest_0.6.15
## [28] rmarkdown_1.10      R.utils_2.9.0       pkgconfig_2.0.1
## [31] htmltools_0.3.6     bibtex_0.4.2        htmlwidgets_1.2
## [34] rlang_0.4.0         bindr_0.1.1         zoo_1.8-6
## [37] jsonlite_1.5        ica_1.0-2           gtools_3.8.1
## [40] R.oo_1.22.0         magrittr_1.5        Matrix_1.2-14
## [43] Rcpp_0.12.17        munsell_0.5.0       ape_5.3
## [46] reticulate_1.12     R.methodsS3_1.7.1   stringi_1.2.3
## [49] yaml_2.1.19         gbRd_0.4-11         MASS_7.3-49
## [52] gplots_3.0.1        Rtsne_0.15          plyr_1.8.4
## [55] grid_3.5.0          parallel_3.5.0      gdata_2.18.0
## [58] listenv_0.7.0       ggrepel_0.8.1       crayon_1.3.4
## [61] lattice_0.20-35     cowplot_0.9.4       splines_3.5.0
## [64] SDMTools_1.1-221.1  knitr_1.20          pillar_1.2.3
## [67] igraph_1.2.4.1      future.apply_1.3.0  codetools_0.2-15
## [70] glue_1.3.1          evaluate_0.10.1     lsei_1.2-0
## [73] metap_1.1           data.table_1.11.4   selectr_0.4-1
## [76] png_0.1-7           Rdpack_0.11-0       gtable_0.2.0
## [79] RANN_2.6.1          purrr_0.2.5         future_1.13.0
## [82] assertthat_0.2.0    rsvd_1.0.1          survival_2.41-3
## [85] viridisLite_0.3.0   tibble_1.4.2        cluster_2.0.7-1
## [88] globals_0.12.4      fitdistrplus_1.0-14 ROCR_1.0-7
```