AcneGraphicalAnalyses_Part2

TamTo

2023-11-27

Introduction

Continuing on with more analysis of single cell data, here I explain how to use and interpret monocle3 (for pseudotime analysis) and CellChat (for ligand-receptor or cell-cell exploration).

```
knitr::opts_chunk$set(echo = TRUE)

library(tidyverse) # load the tidyverse
library(Seurat) # we need this to work with Seurat objects

setwd("/Users/tamto/Documents/GitHub/MolecularBiologyProjects/Projects/Project_AcneGraphicalAnalyses_Par

load("/Users/tamto/Desktop/NewLymphocyteClusters.Rdata")
```

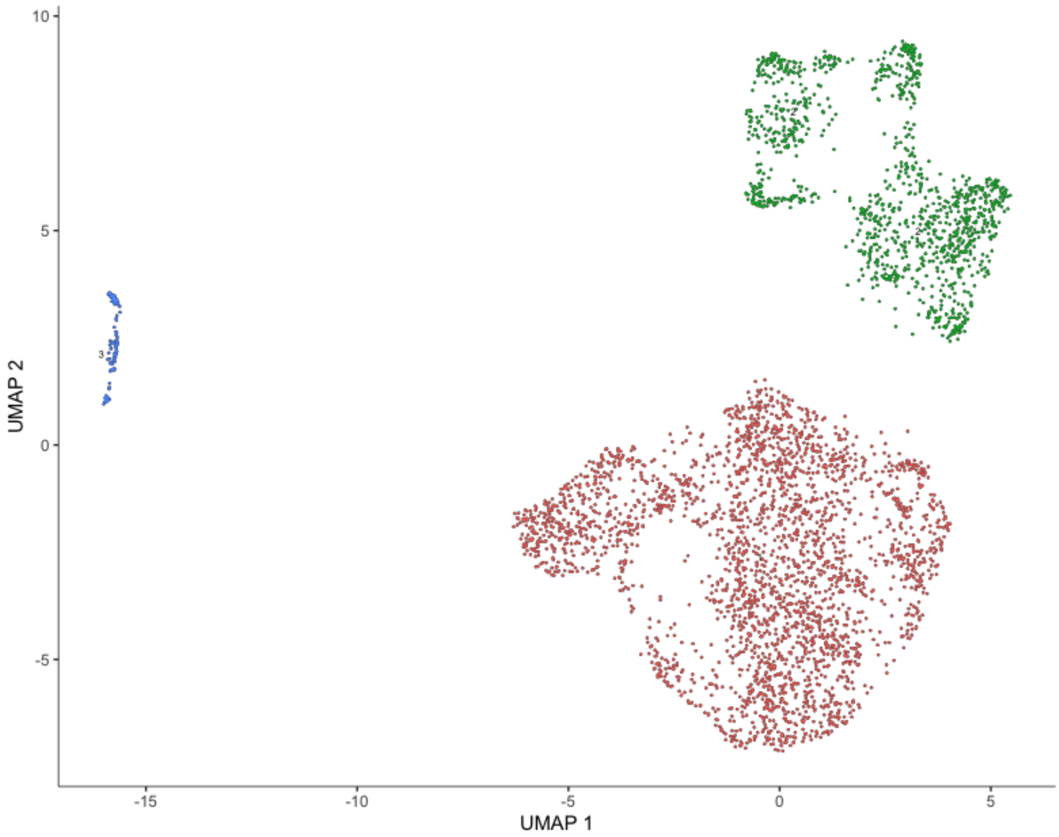
Pseudotime Analysis

Cells can transition from one functional state to another based on various stimuli or the microenvironment. (For example, a naive T cell can become a Th1, Th2, Th17, or Treg based on different stimuli.) Because we can analyze cell clusters from our single cell data, we can investigate cell trajectories too. Pseudotime analysis helps us determine which state each cell might be in based on their gene expression or the cells' progress through each transition state. Monocle is a powerful way to analyze trajectories because it learns the sequence of gene expression changes for each cell to place it within a trajectory.

Because this dataset doesn't really focus on a cell's transition over time (i.e. a stem cell data set or sequencing of cells over time), trajectory analysis is not as useful for our analysis, but it's good to know to apply for any future dataset that might need it.

Here is the site to help with installing the monocle3 package.

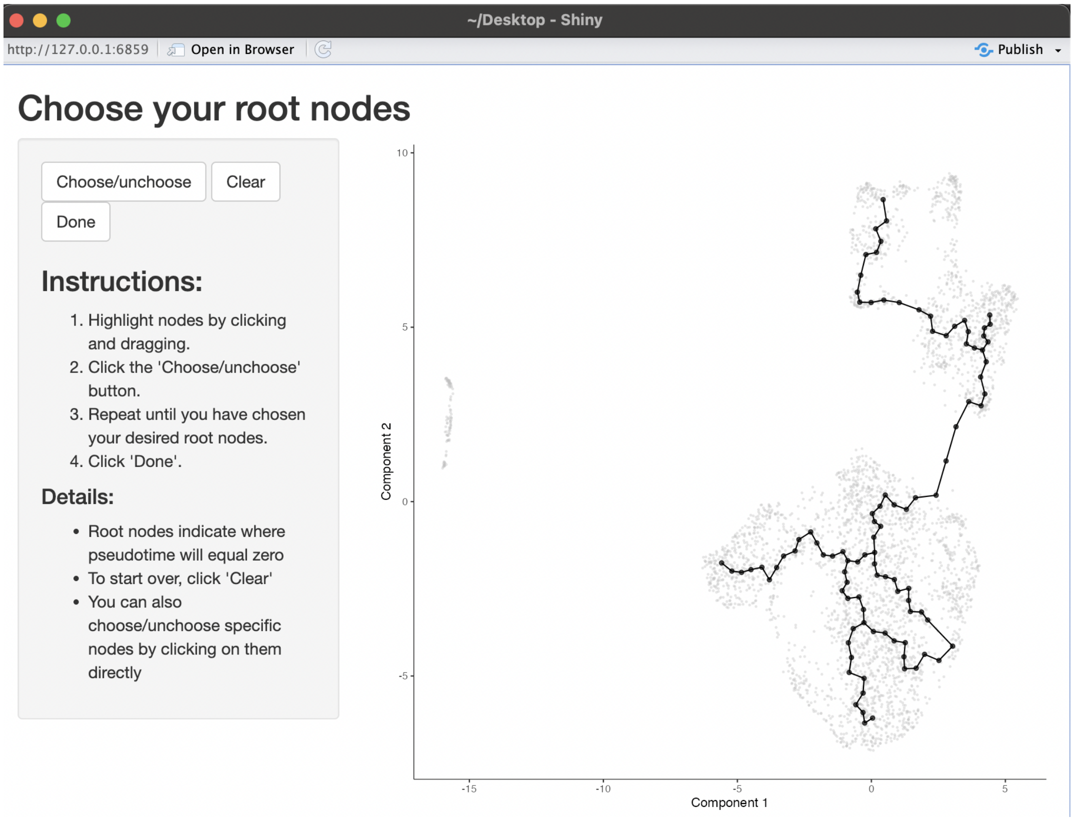
```
# if (!requireNamespace("BiocManager", quietly = TRUE))
# install.packages("BiocManager")
# BiocManager::install(version = "3.14")

# BiocManager::install(c('BiocGenerics', 'DelayedArray', 'DelayedMatrixStats',
#                        'limma', 'lme4', 'S4Vectors', 'SingleCellExperiment',
#                        'SummarizedExperiment', 'batchelor', 'HDF5Array',
#                        'terra', 'ggrastr'))

# install.packages("devtools")
# devtools::install_github('cole-trapnell-lab/monocle3')
# install.packages("remotes")
# remotes::install_github("satijalab/seurat-wrappers")

library(monocle3)

# Useful way for getting started with monocle3: https://cole-trapnell-lab.github.io/monocle3/docs/getti

# Install SeuratWrappers and convert to monocle3 cell_data_set. We need to load the data into monocle3'

# remotes::install_github('satijalab/seurat-wrappers')
library(SeuratWrappers)

cds <- SeuratWrappers::as.cell_data_set(NewLymphocyteClusters)

# Now you can preprocess the data (dimension reduction, cell clustering, etc.). Monocle3 then "learns"

cds <- cluster_cells(cds)
cds <- learn_graph(cds, use_partition = FALSE)

# You can view the UMAP by plotting the cds and showing it on the trajectory graph.

plot_cells(cds,
           show_trajectory_graph = FALSE,
           color_cells_by = "partition")
```



Here, we see that there are 3 main “partitions”. According to our UMAP, these are the CD4, CD8, and B cell clusters. You can then use the function order_cells() to choose the nodes of interest and which clusters you want to analyze.

After selecting the nodes of interest, you can plot it in various ways (i.e. get connect the nodes, label branch points, etc.). You can group by celltype/pseudotime/partition, edit the labels, etc. Grouping by pseudotime allows you to see the transition (dark blue at 0 indicates the root cells, yellow indicates the end cell state). You can also choose how you want to label (roots, leaves, branch points, etc.).

```
cds <- order_cells(cds)

plot_cells(cds,
           color_cells_by = "pseudotime",
           label_roots = F,
           label_leaves = T,
           label_branch_points = F
           )
```



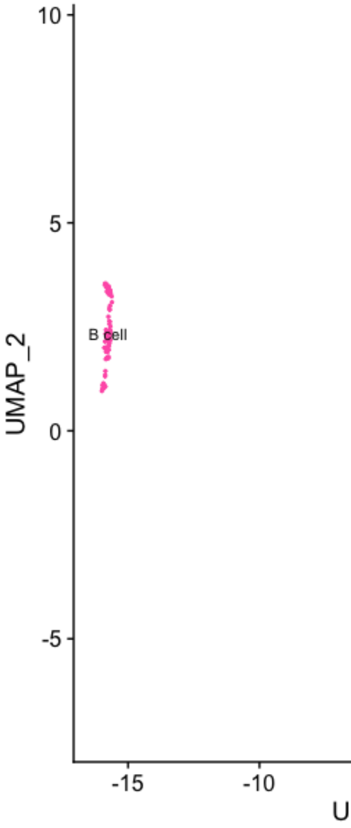
Choose your root nodes

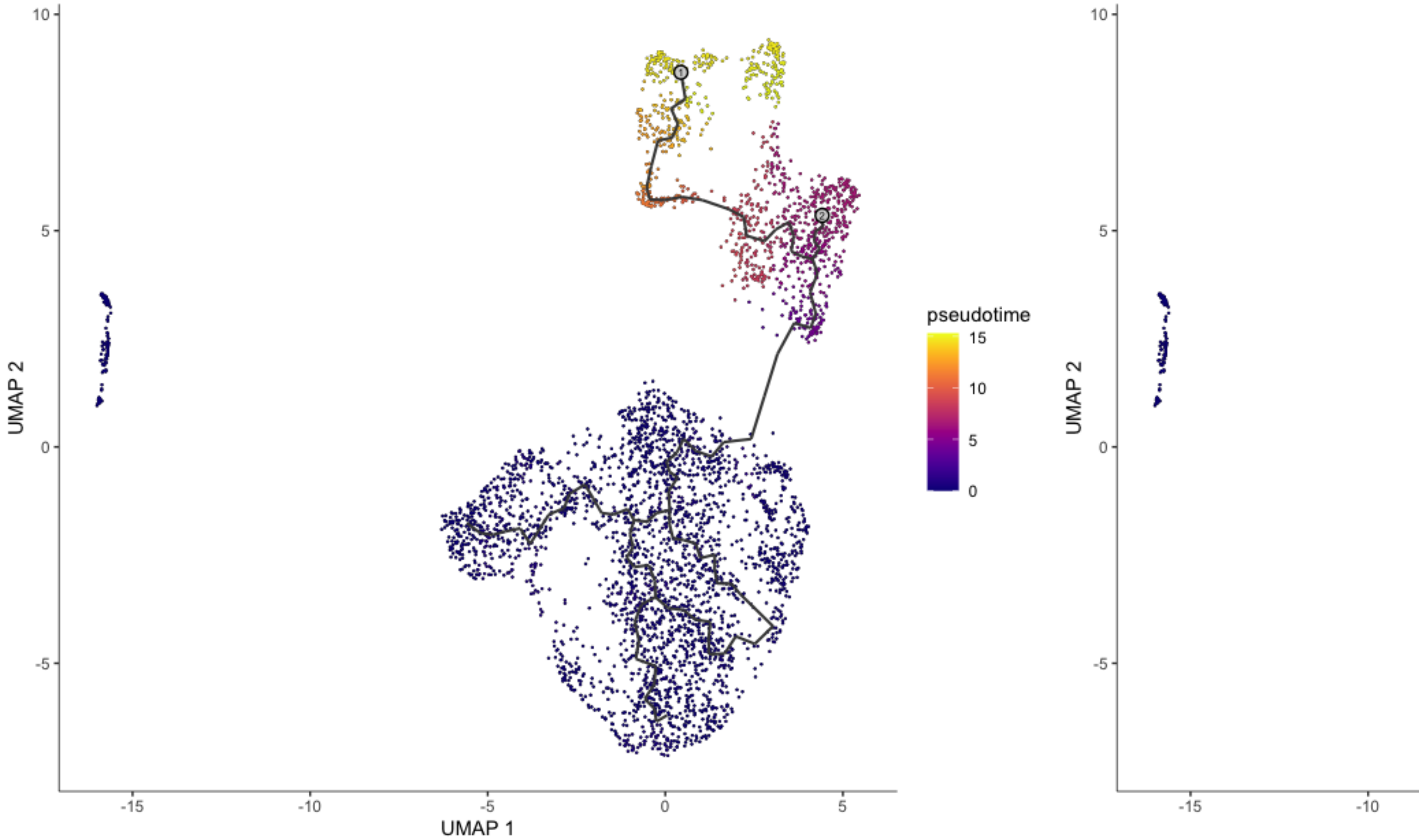
Instructions:

1. Highlight nodes by clicking and dragging.
2. Click the 'Choose/unchoose' button.
3. Repeat until you have chosen your desired root nodes.
4. Click 'Done'.

Details:

- Root nodes indicate where pseudotime will equal zero
- To start over, click 'Clear'
- You can also choose/unchoose specific nodes by clicking on them directly

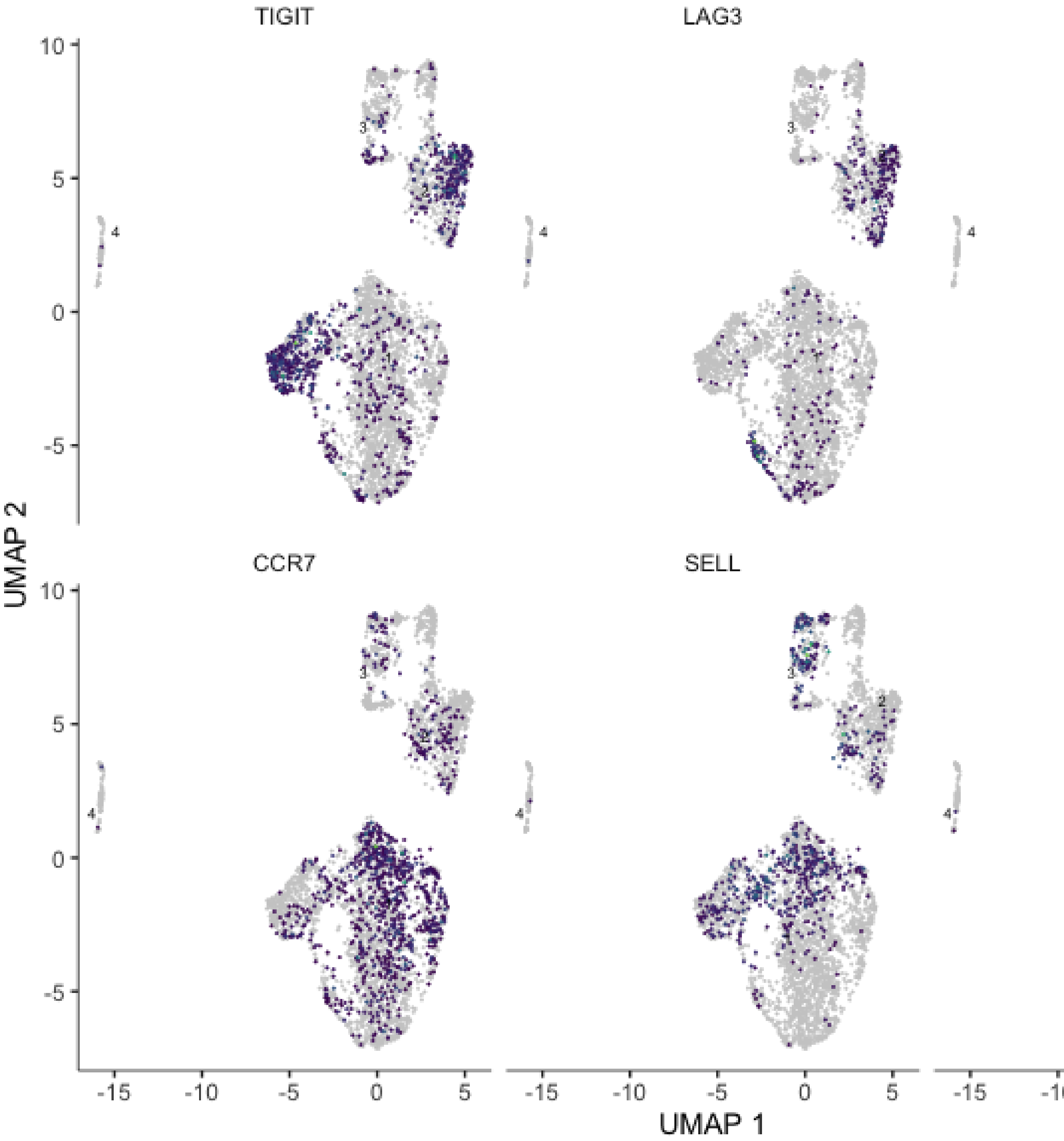




Based on nodes chosen and the pseudotime plot I generated, it shows that the root cells are the Naive CD4 T cell cluster, and the end cell states are at the Treg, EM CD4 T cell, or Th17 cell clusters. On the other hand, choosing the CD8 / NK cell cluster shows the root cells beginning at the stem-like CD8 cells and ending as either an effector CD8 T cell or NK cell. This makes sense because the naive CD4 T cells or stem-like CD8s can differentiate into other cell types.

Monocle3 is also useful for plotting where and when certain genes may be highly expressed (by comparing the expression to other cells in the data). For instance, the expression of a certain gene may increase or decrease over time based on how the cell changes over time. To do this analysis, you first have to specify the gene name and short name column.

```
rowData(cds)$gene_name <- rownames(cds)
rowData(cds)$gene_short_name <- rowData(cds)$gene_name

# You can check to make sure a certain gene name is there by searching it in the row names of the cds t

term_genes <- c("TIGIT", "LAG3", "PDCD1", "CCR7", "SELL", "TCF7")

plot_cells(cds,
           genes = term_genes,
           label_cell_groups=TRUE,
           show_trajectory_graph=FALSE)
```



Because our dataset does not have cells based on different time points, this analysis is not as relevant. It is important to notice, however, that our generated graph does show terminally differentiated effector state genes such as TIGIT, LAG3, and PDCD1 are expressed more in differentiated cell clusters. On the other hand, naive/stem cell-like markers such as CCR7, SELL, and TCF7 are more highly expressed in the naive clusters.

Site for more monocle3 information. You can also do pseudotime analysis with RNA velocity using the Python package, scVelo.

CellChat

CellChat allows us to explore ligand-receptor interactions and cell-cell communications from our scRNAseq dataset. CellChat groups cells by building a shared neighbor graph based on cell-cell distance in the dimensional or pseudotemporal trajectory space.

You can find more information on it here.

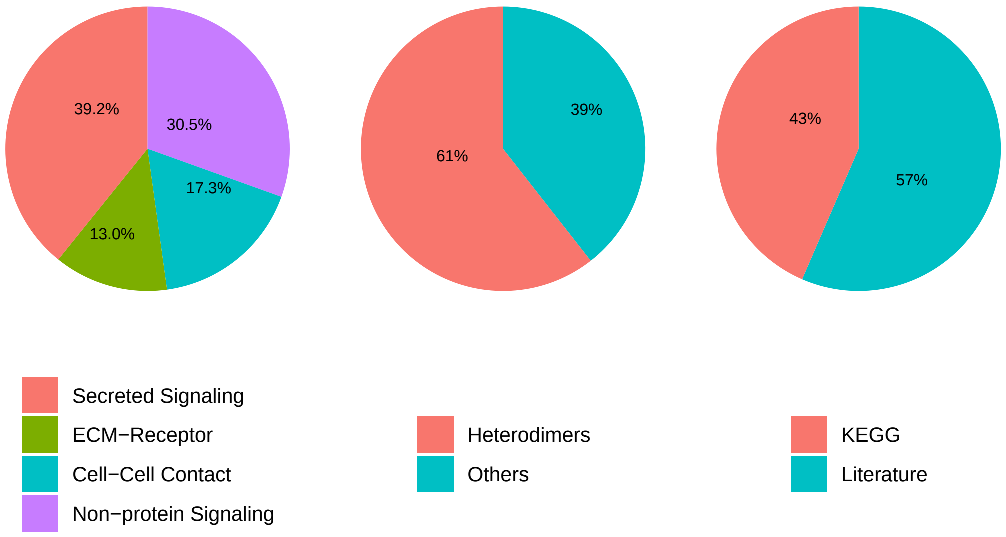
Start off by installing CellChat then input the seurat object into the createCellChat() function, making sure to group by the cell cluster identities.

```r
# Install CellChat and necessary packages

# BiocManager::install("ComplexHeatmap")
# install_github('immunogenomics/presto')
# devtools::install_github("jinworks/CellChat")

library(CellChat)
library(patchwork)
options(stringsAsFactors = FALSE)

cellChat <- createCellChat(object = NewLymphocyteClusters, group.by = "ident", assay = "RNA")
```

```
## [1] "Create a CellChat object from a Seurat object"
## The ‘meta.data‘ slot in the Seurat object is used as cell meta information
## Set cell identities for the new CellChat object
## The cell groups used for CellChat analysis are  Naive CD4 T cell EM CD4 T cell Late EM CD4 T cell Tr
```

Set the ligand-receptor interaction database (in this case, it’s human).

```r
CellChatDB <- CellChatDB.human
showDatabaseCategory(CellChatDB)
```



```
# You can view the database with glimpse()
# glimpse(CellChatDB$interaction)

# use a subset of CellChatDB for cell-cell communication analysis--here I'm choosing to analyze secrete
CellChatDB.use <- subsetDB(CellChatDB, search = "Secreted Signaling")

# set the used database in the object and subset (even if we use the whole data)
cellChat@DB <- CellChatDB.use
cellChat <- subsetData(cellChat)
```

Preprocess the expression data for cell-cell communication analysis. We can use the identifyOverExpressed…() function to identify the overexpressed ligands/receptors and interactions in each cell group.

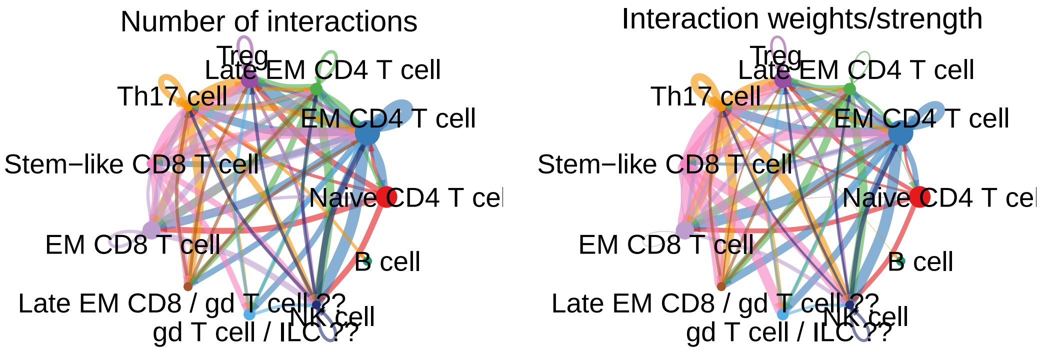
Next we can infer the cell-cell communication network by assigning a probability value to each interaction, which depends on teh average gene expression per cell group. You can also filter out communications with the argument min.cells.

```
cellChat <- identifyOverExpressedGenes(cellChat)
cellChat <- identifyOverExpressedInteractions(cellChat)

cellChat <- computeCommunProb(cellChat) # this function uses truncated mean to calculate average gene e
```

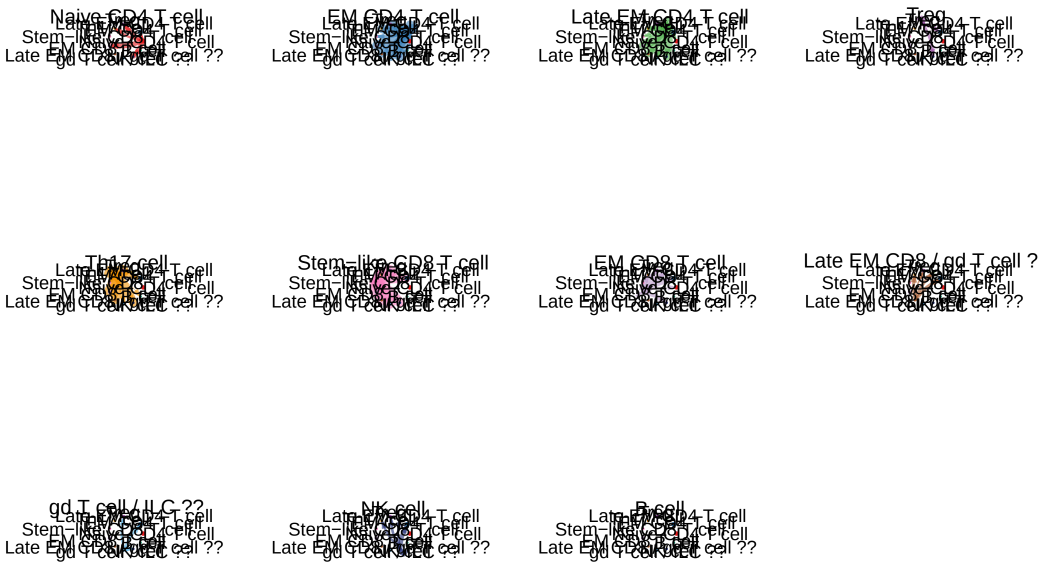
```
## triMean is used for calculating the average gene expression per cell group.
## [1] ">>> Run CellChat on sc/snRNA-seq data <<< [2023-11-27 21:41:39.959477]"
## [1] ">>> CellChat inference is done. Parameter values are stored in ‘object@options$parameter‘ <<< [
cellChat <- filterCommunication(cellChat, min.cells = 10)

cellChat <- computeCommunProbPathway(cellChat) # computes communication probability of ligand-receptor
```

We can then calculate the aggregated cell-cell communication network and visualize it.

```
cellChat <- aggregateNet(cellChat) # calculate aggregated cell-cell communication network and use for v

groupSize <- as.numeric(table(cellChat@idents))

par(mfrow = c(1,2), xpd=TRUE)

netVisual_circle(cellChat@net$count, vertex.weight = groupSize, weight.scale = T, label.edge= F, title.

netVisual_circle(cellChat@net$weight, vertex.weight = groupSize, weight.scale = T, label.edge= F, title
```

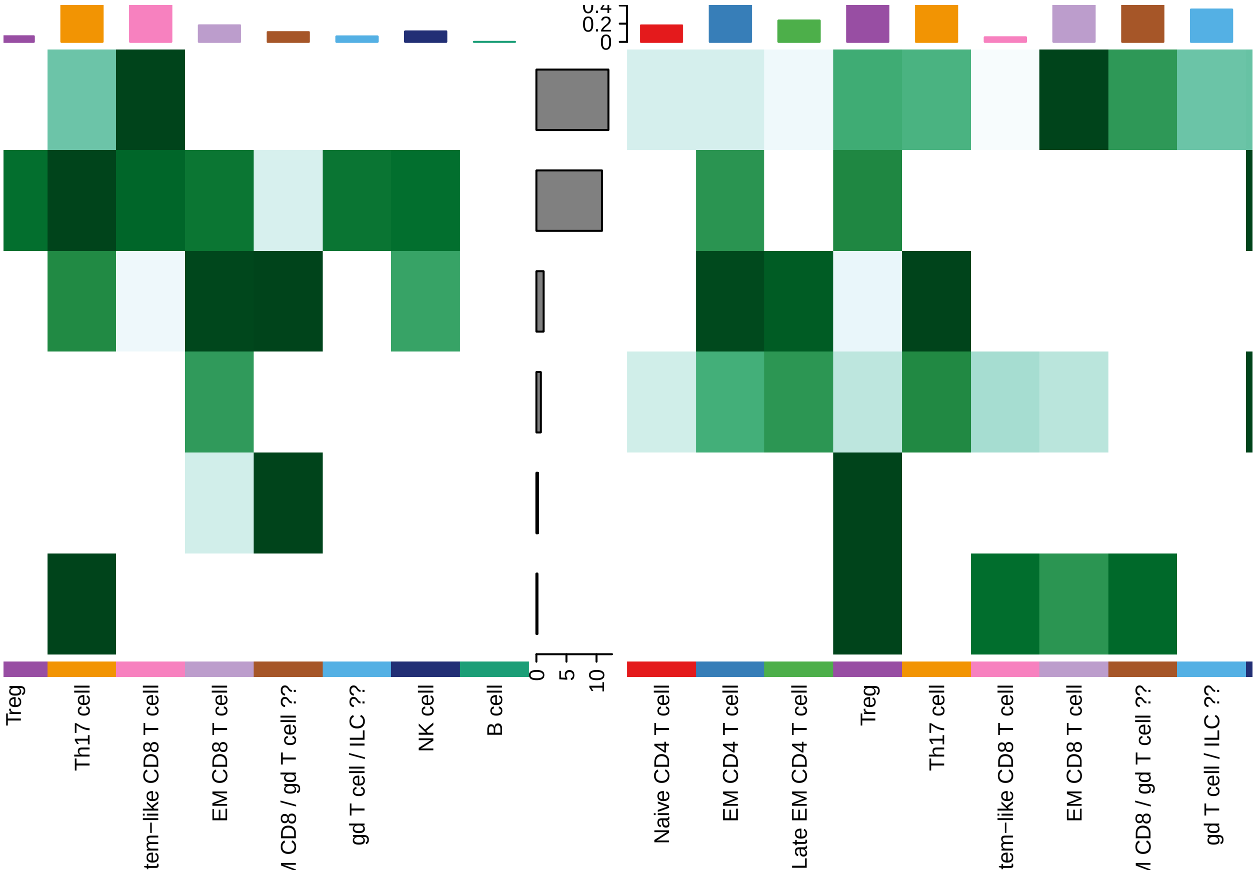


```
mat <- cellChat@net$weight

par(mfrow = c(3,4), xpd=TRUE)
for (i in 1:nrow(mat)) {
  mat2 <- matrix(0, nrow = nrow(mat), ncol = ncol(mat), dimnames = dimnames(mat))
  mat2[i, ] <- mat[i, ]
  netVisual_circle(mat2, vertex.weight = groupSize, weight.scale = T, edge.weight.max = max(mat), title
}
```



We can compare the weight of signaling events from each cell group with the above graphs generated. For further data exploration, we can visualize cell-cell communication in several ways. We can generate heat maps that show the signaling pathways.

```r
# Compute the network centrality scores
cellChat <- netAnalysis_computeCentrality(cellChat, slot.name = "netP")

# Signaling role analysis on the aggregated cell-cell communication network from all signaling pathways
ht1 <- netAnalysis_signalingRole_heatmap(cellChat, pattern = "outgoing")
ht2 <- netAnalysis_signalingRole_heatmap(cellChat, pattern = "incoming")
ht1 + ht2
```

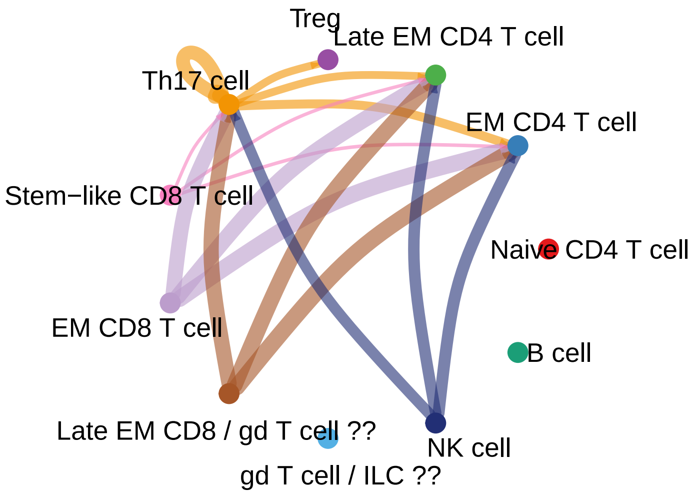


```
# Signaling role analysis on the cell-cell communication networks of interest
ht <- netAnalysis_signalingRole_heatmap(cellChat, signaling = c("CCL"))
```

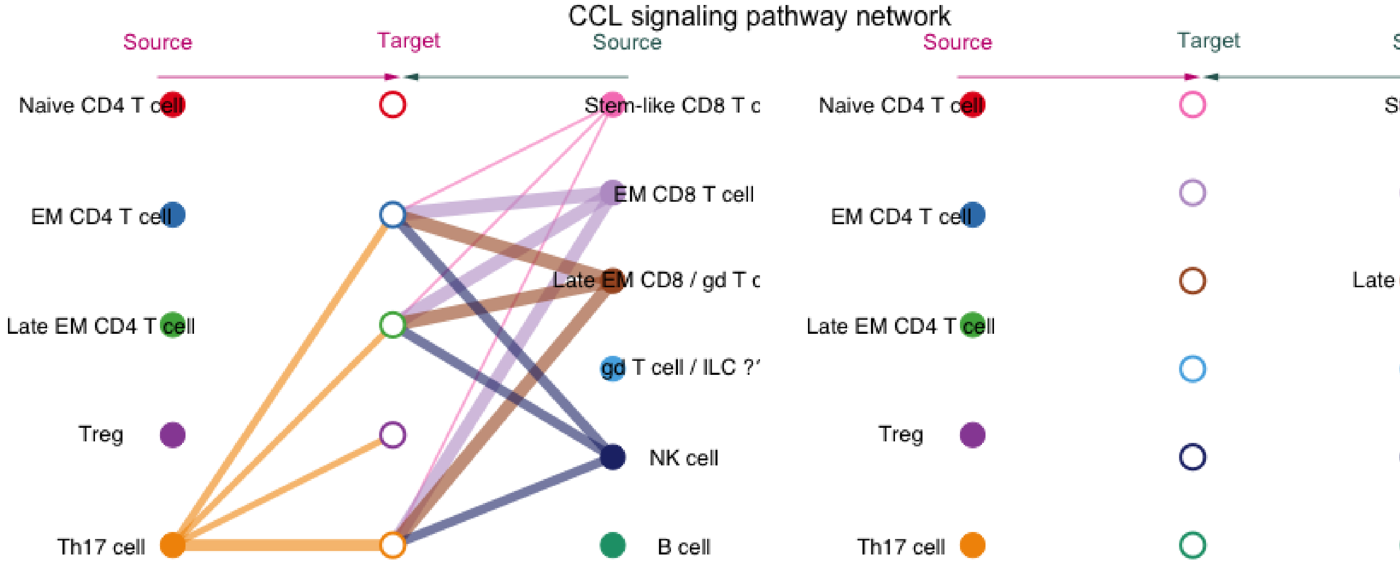
Looks like CCL, TNF, MIF, etc. are signaling pathways that occur in this dataset. We can see where the outgoing signals come from and which cells receive those signals.

In deeper analyses, I'm going to further explore the CCL pathway because we can know which chemokines may be produced by which cells to recruit other lymphocytes.

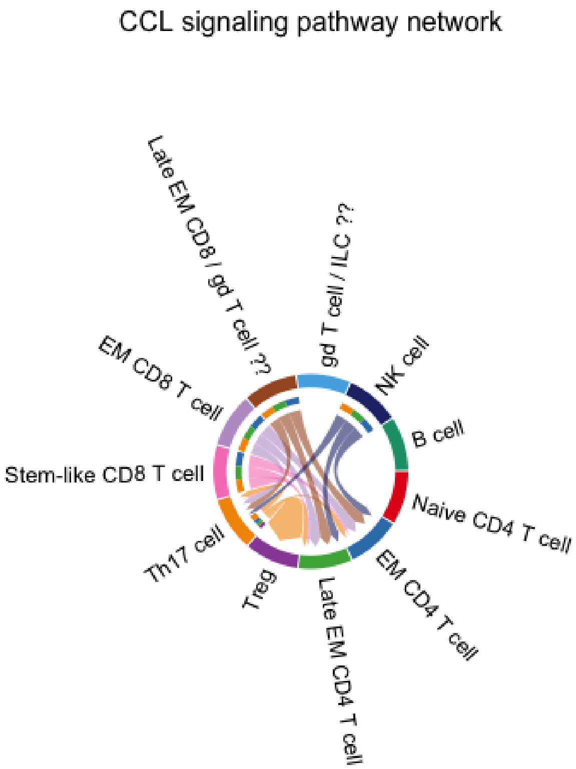
```
pathways.show <- c("CCL")

# Circle plot
par(mfrow=c(1,1))
netVisual_aggregate(cellChat, signaling = pathways.show, layout = "circle")
```



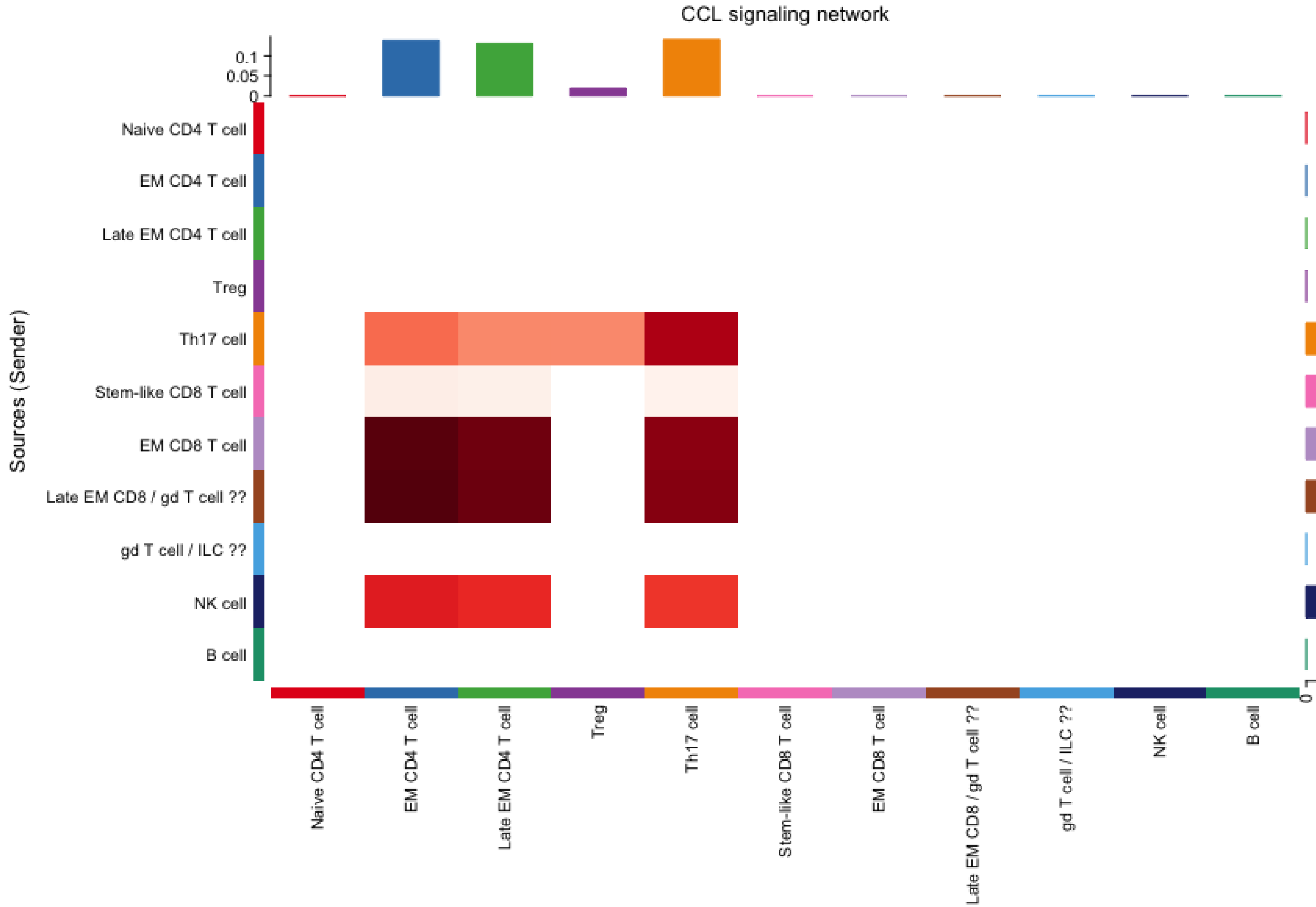
```
# Hierarchy plot
vertex.receiver = seq(1,5) # a numeric vector to show 5 on the left side and the rest on the right side
netVisual_aggregate(cellChat, signaling = pathways.show,  vertex.receiver = vertex.receiver, layout = "
```



```
# Chord diagram
par(mfrow=c(1,1))
netVisual_aggregate(cellChat, signaling = pathways.show, layout = "chord")
```



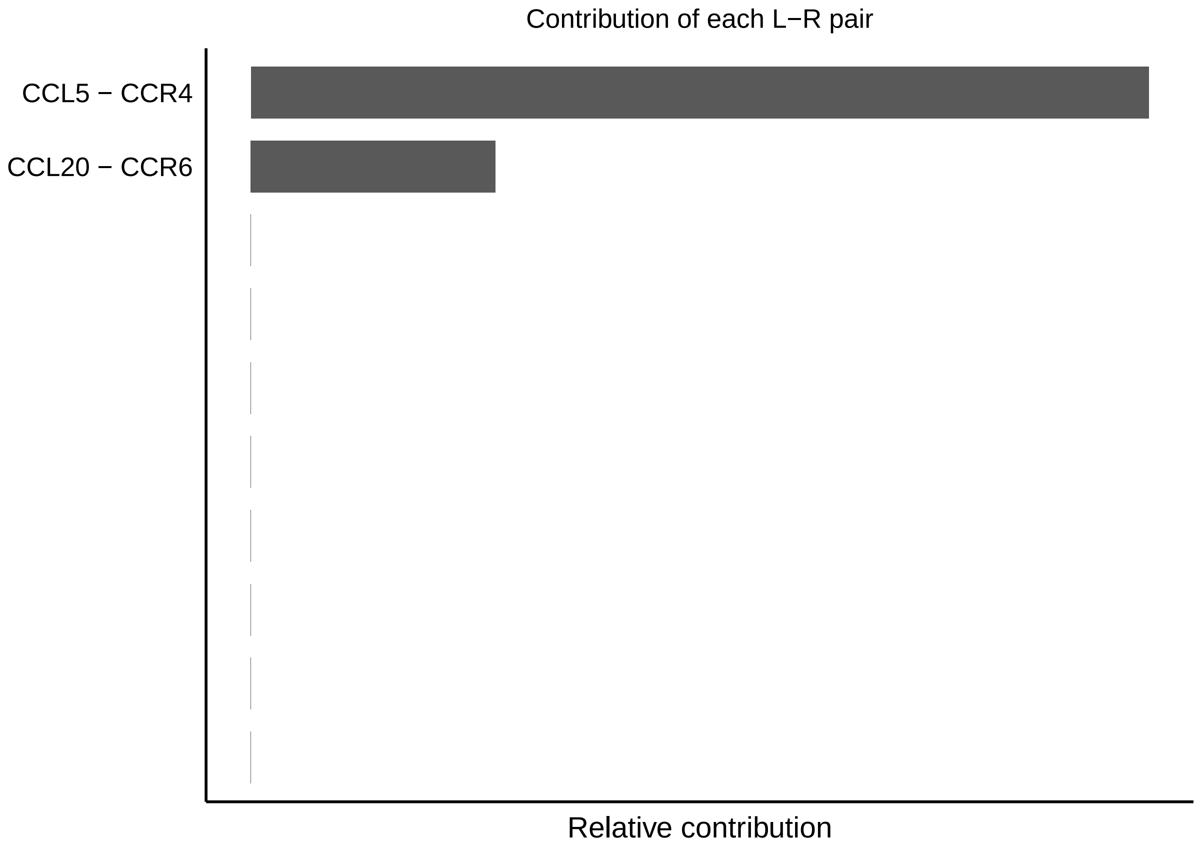
```
# Heatmap
par(mfrow=c(1,1))
netVisual_heatmap(cellChat, signaling = pathways.show, color.heatmap = "Reds")
```



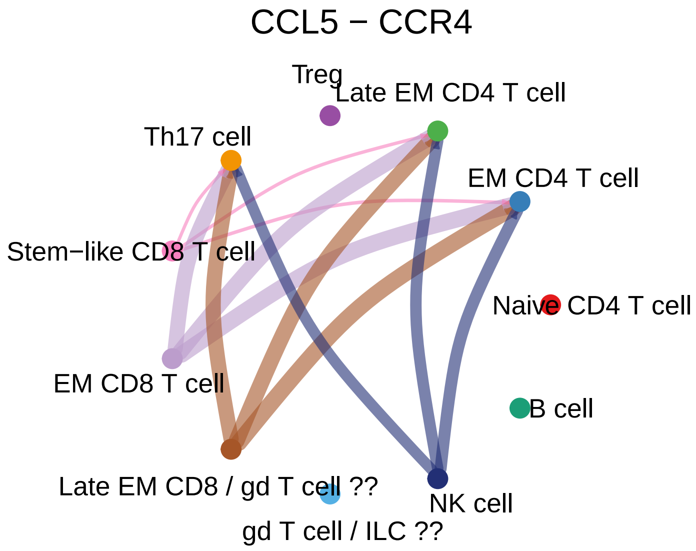
Based on these graphs, it looks like CD8 T cells and NK cells are the "senders" (i.e. the ones producing the chemokines "CCL" and the CD4 T cells are the receivers that express the chemokine receptors "CCR"). This makes sense because cytotoxic responders recruit helper T cells to sites of inflammation.

To go more in depth, we can also take a closer look at the ligand-receptor (L-R) pairs themselves.

```
netAnalysis_contribution(cellChat, signaling = pathways.show) # Here we see CCL5-CCR4 is what contribut
```

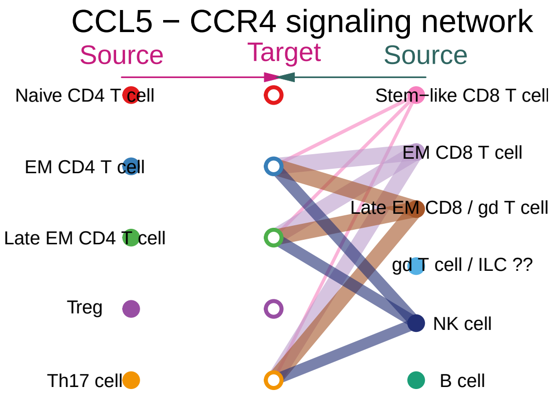


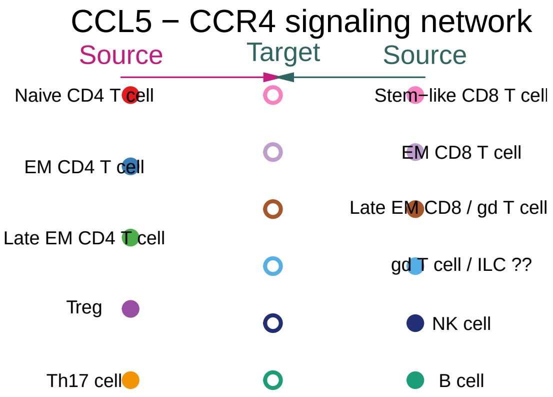
```
pairLR.CCL <- extractEnrichedLR(cellChat, signaling = pathways.show, geneLR.return = FALSE) # extract t

LR.show <- pairLR.CCL[1,] # show one L-R pair

# Circle plot
netVisual_individual(cellChat, signaling = pathways.show,  pairLR.use = LR.show, vertex.receiver = vert
```



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

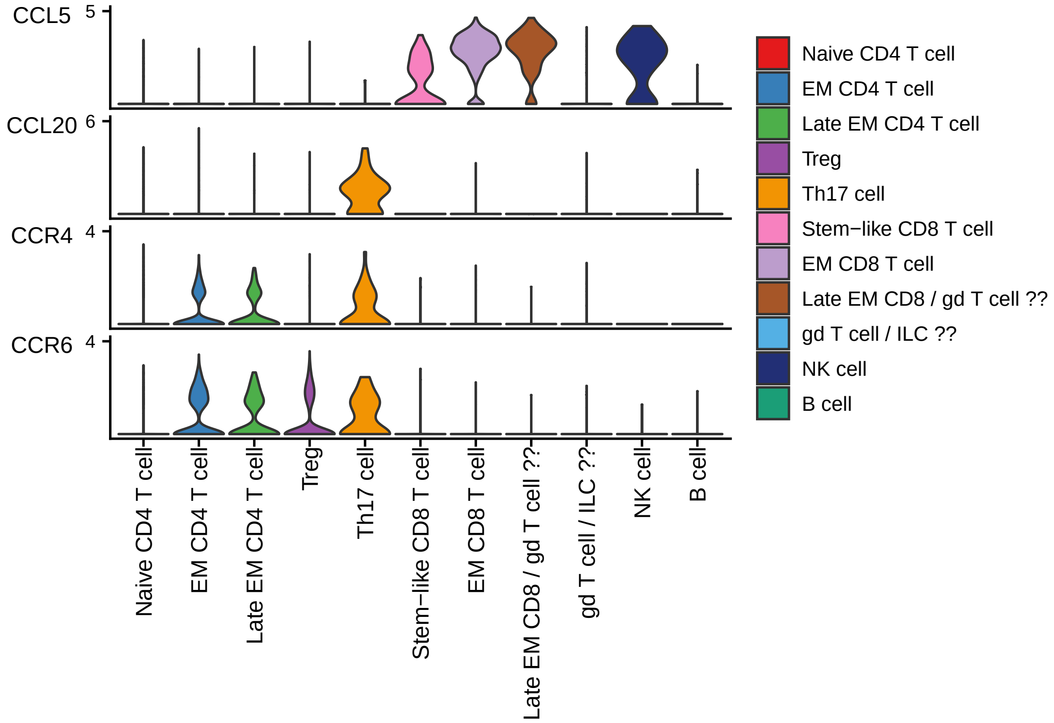
```
# Hierarchy diagram
vertex.receiver = seq(1,5)
netVisual_individual(cellChat, signaling = pathways.show, pairLR.use = LR.show,, vertex.receiver = verte
```



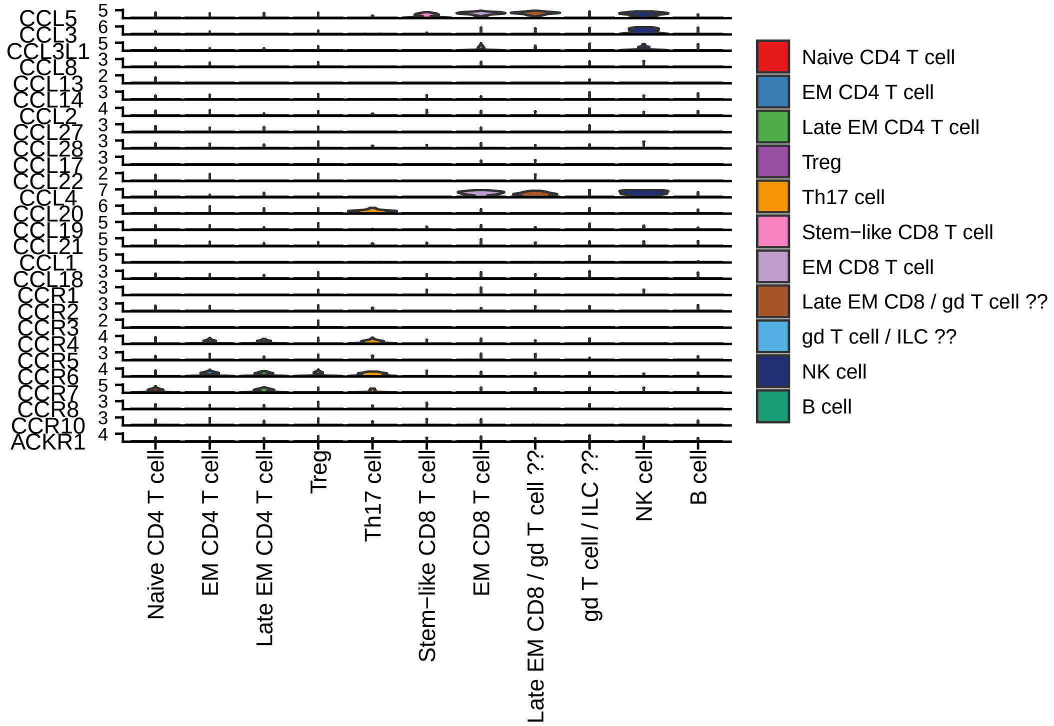


You can also plot the gene expression distribution of these ligands and receptors on violin/dot plots!

```
plotGeneExpression(cellChat, signaling = "CCL")
```

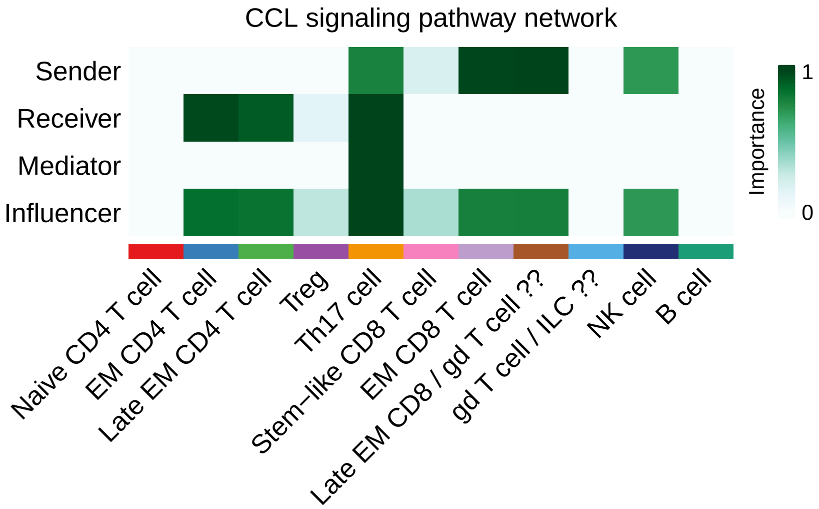


```
plotGeneExpression(cellChat, signaling = "CCL", enriched.only = FALSE) #shows ALL ligands/receptors (no
```



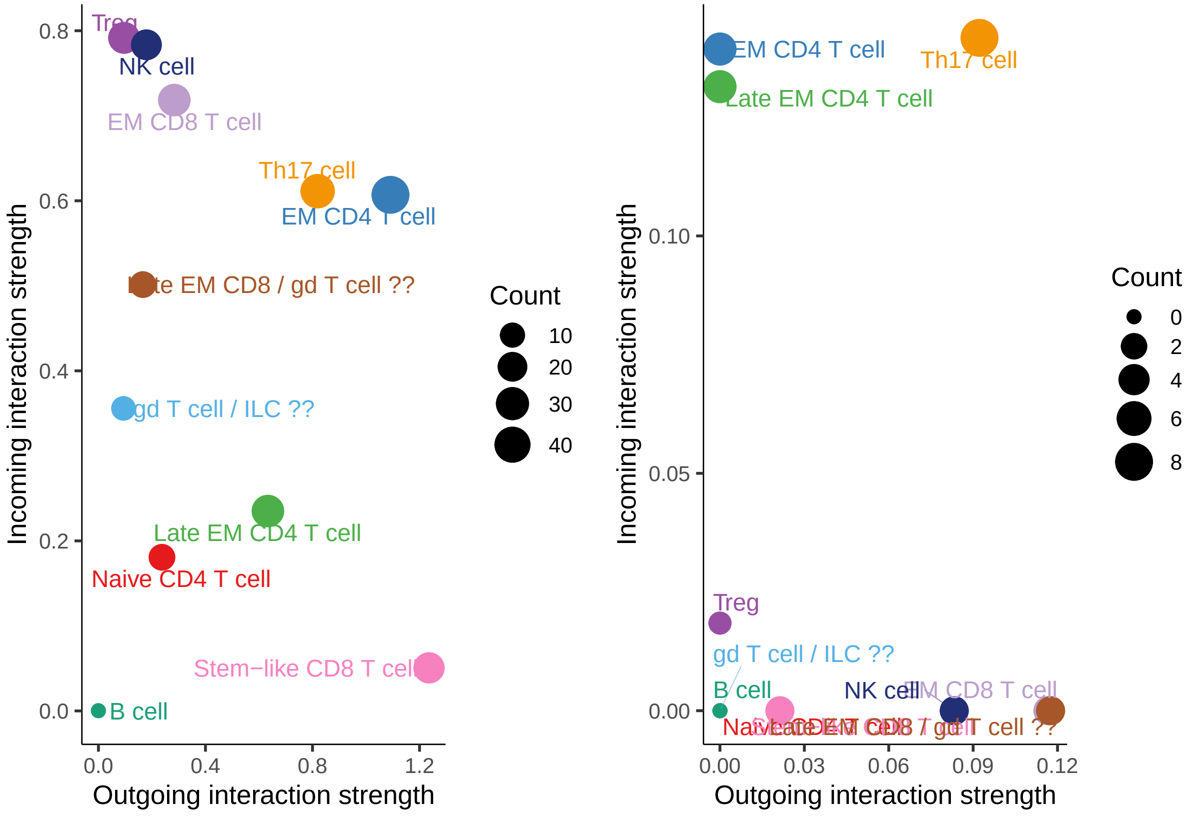
Here you can also identify signaling roles of cell groups (senders, receivers).

```r
# Visualize with heatmap
netAnalysis_signalingRole_network(cellChat, signaling = pathways.show, width = 8, height = 2.5, font.si
```



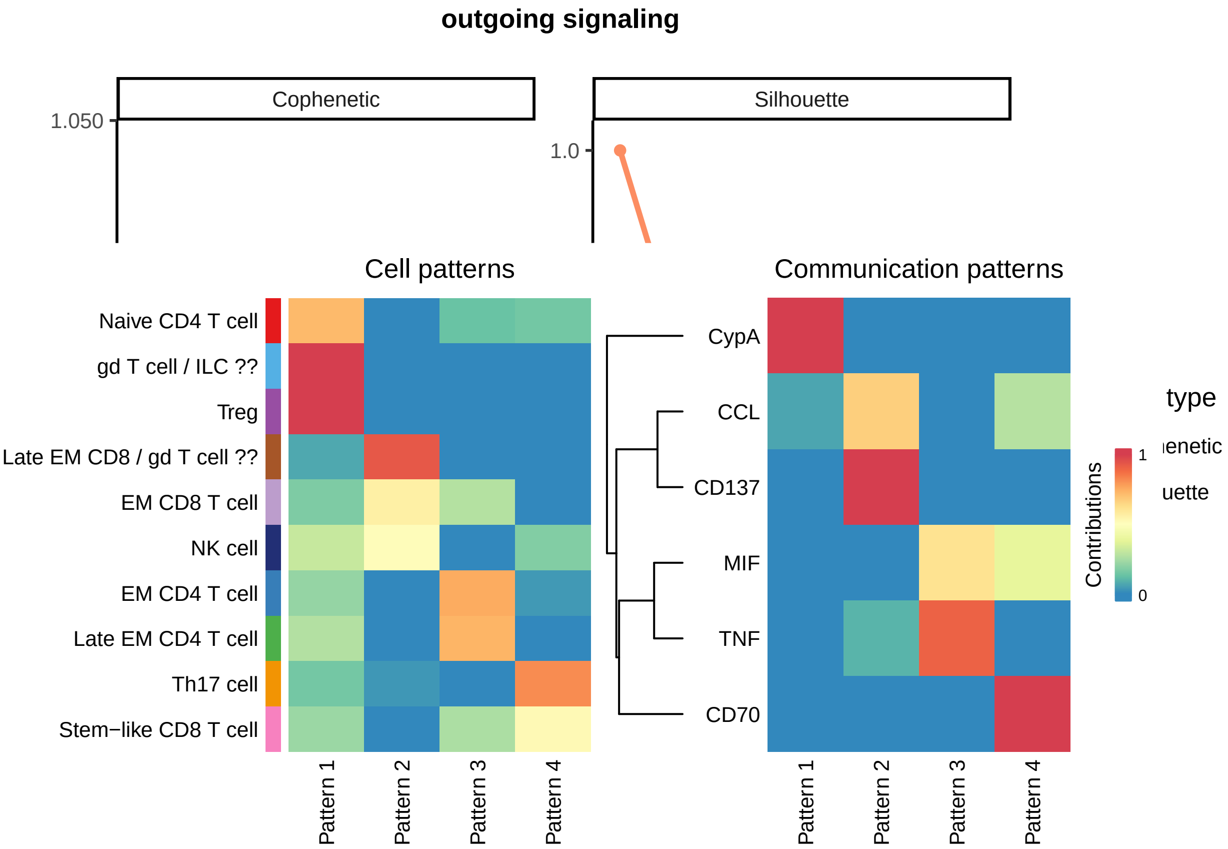
This reaffirms what we saw earlier in that CD8 T cells are the main senders and CD4 T cells are the receivers. You can also visualize it in terms of a 2D space.

```r
# Signaling role analysis on the aggregated cell-cell communication network from all signaling pathways
gg1 <- netAnalysis_signalingRole_scatter(cellChat)

# Signaling role analysis on the cell-cell communication networks of interest (i.e. the CCL pathway)
gg2 <- netAnalysis_signalingRole_scatter(cellChat, signaling = c("CCL"))

gg1 + gg2
```

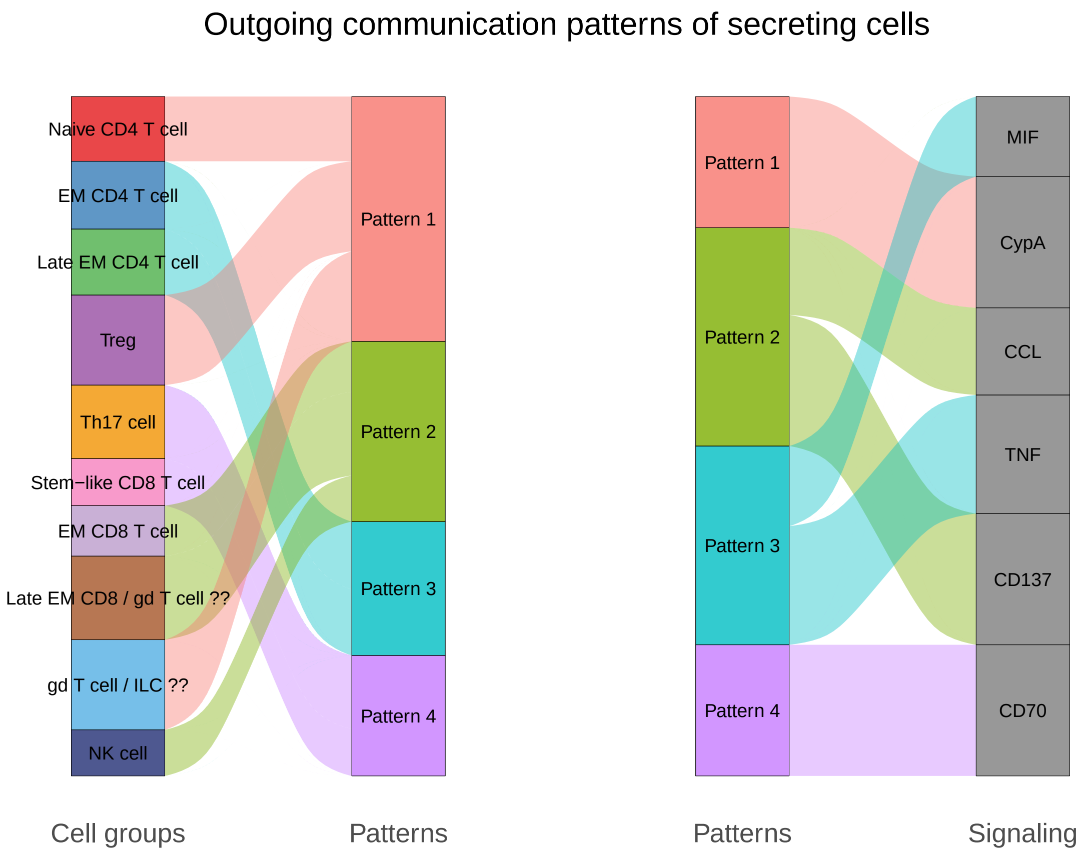


You can also visualize outgoing and incoming signal communication patterns to show how cells coordinate with each other or with certain signaling pathways to drive/respond to the communication.

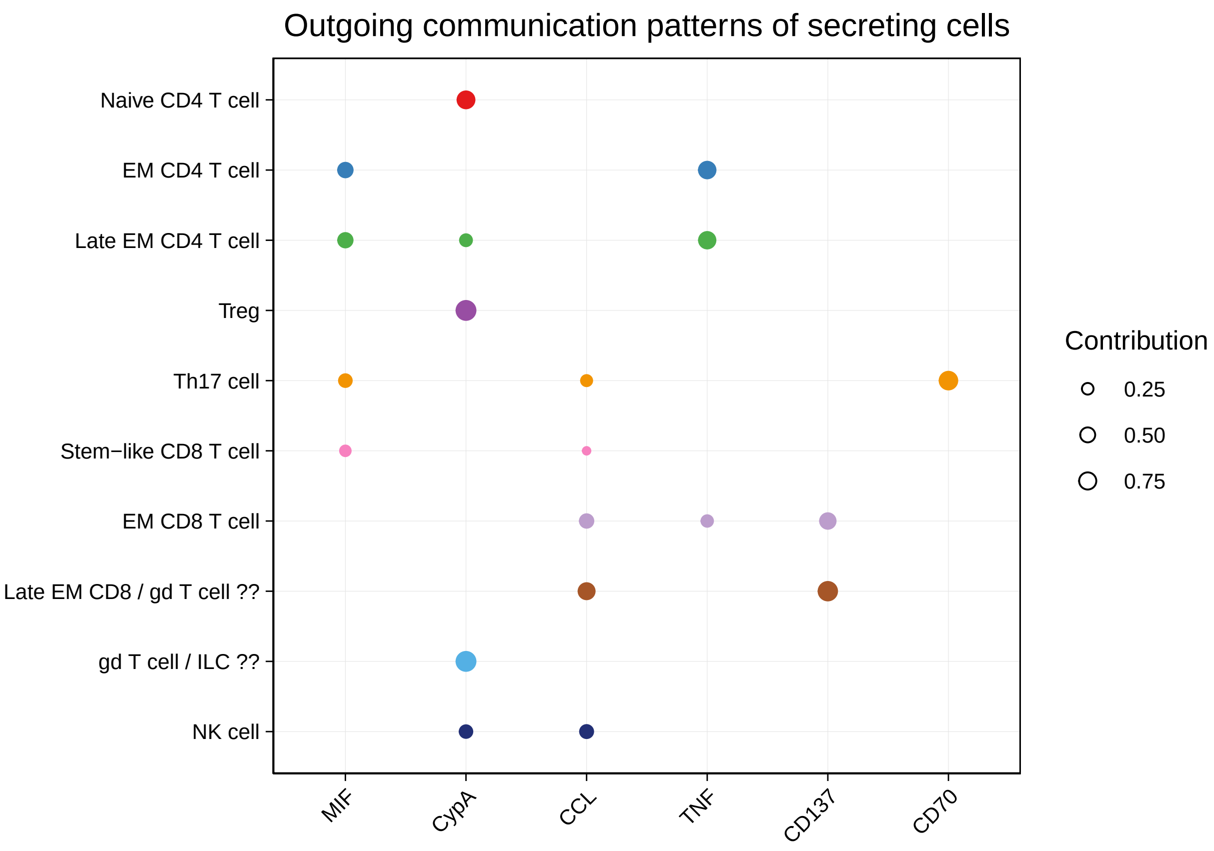
```
library(NMF)
library(ggalluvial)

# Outgoing communication signals

selectK(cellChat, pattern = "outgoing")
nPatterns = 4
cellChat <- identifyCommunicationPatterns(cellChat, pattern = "outgoing", k = nPatterns)
```



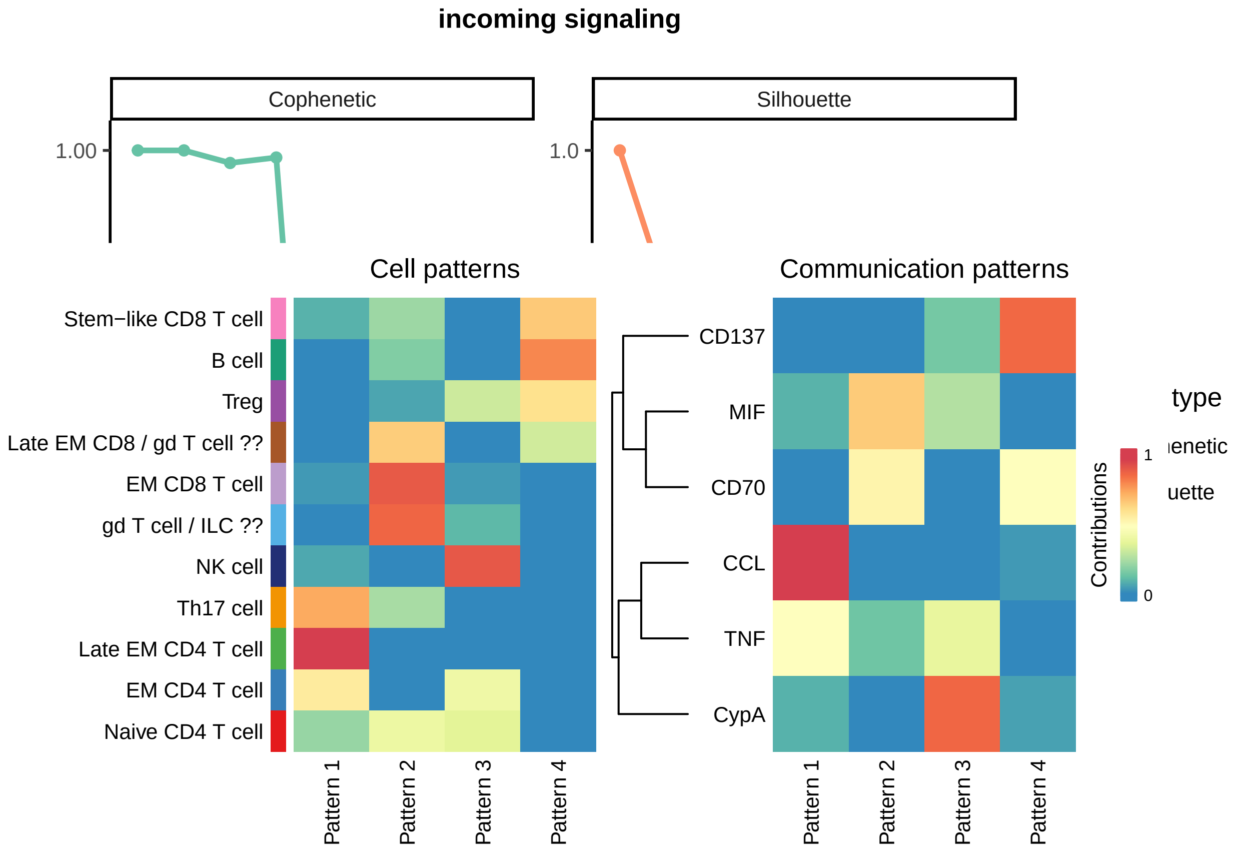
```
# river plot
netAnalysis_river(cellChat, pattern = "outgoing")
```



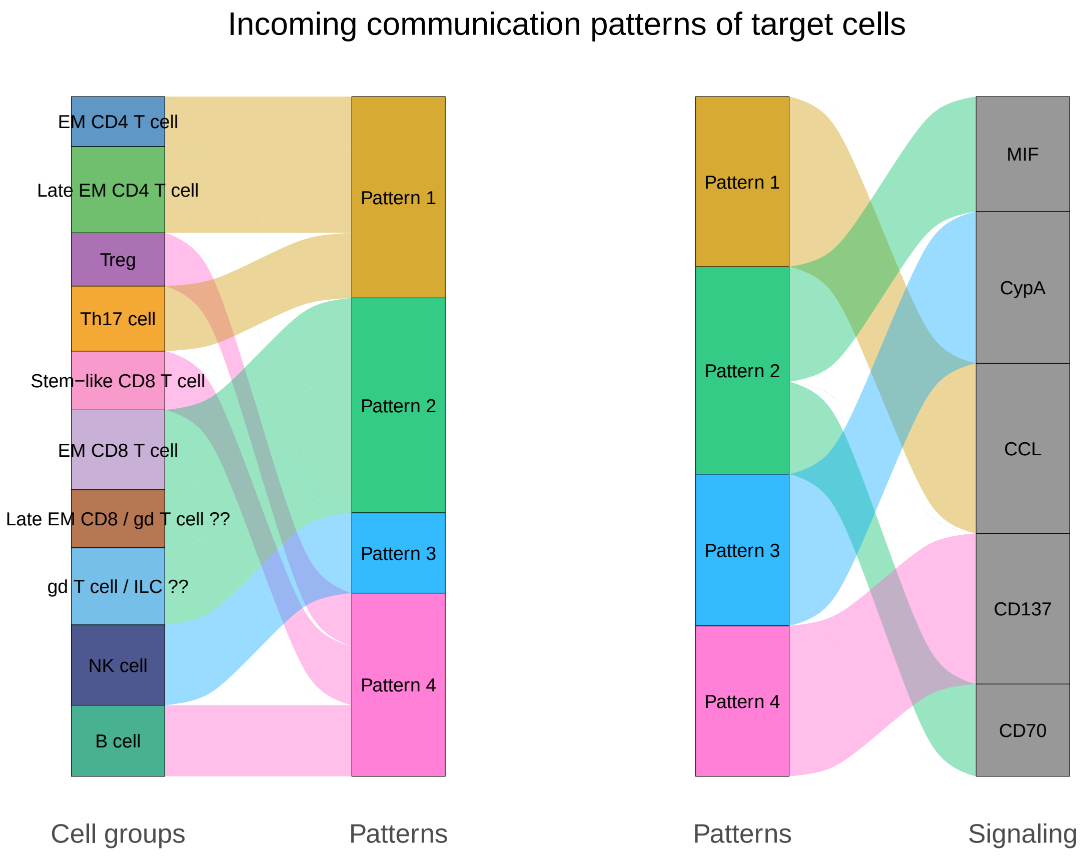
```
# dot plot
netAnalysis_dot(cellChat, pattern = "outgoing")
```



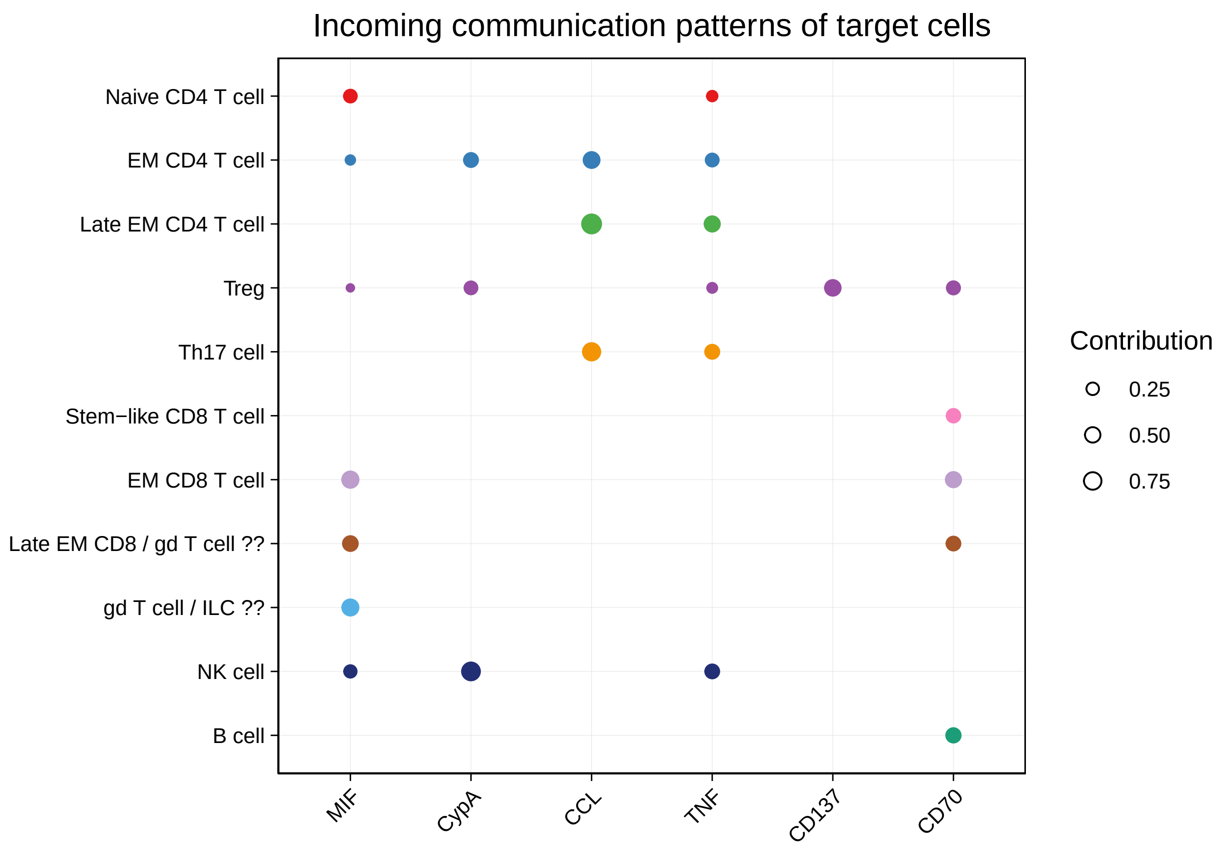
```
# Incoming communication patterns

selectK(cellChat, pattern = "incoming")
nPatterns = 4
cellChat <- identifyCommunicationPatterns(cellChat, pattern = "incoming", k = nPatterns)
```



```
# river plot
netAnalysis_river(cellChat, pattern = "incoming")
```



```
# dot plot
netAnalysis_dot(cellChat, pattern = "incoming")
```



Overall, CellChat is a powerful tool to analyze how cells communicate with each other to contribute to pathogenesis. Just from this dataset alone, we're able to understand how lymphocytes interact/communicate with each other.

More information on this CellChat and its features can be found here.

Conclusion

As I've shown in these past 2 projects, there are so many ways we can analyze single cell data. Analysis of the genome and transcriptome of a single cell allows for so much discovery and elucidation of disease pathogenesis.