01\_visium\_pipeline

# Introduction

We will analyze Visium 10x data from a tissue slide of High-grade serous ovarian carcinoma (HGSOC).

—> @team put the description of the slide/tumor that we are analysing

The tumour sample was collected during interval debulking surgery from HGSOC patient with a good response to taxane- and platinum-based neoadjuvant chemotherapy (NACT) treatment.

# Warming up: setting the environment that you need

Before starting, let’s set the stage to be all on the same ground.

You have been provided with configuration files. Please select “your” environment and the “helper” functions

—> @team here they simply have to understand amongst all files we provided what they may need.

# Loading the package you need

require(ggplot2)  
require(ggsci)  
require(plyr)  
require(dplyr)  
require(patchwork)  
require(Seurat)

# Loading Visium 10x data of HGSOC slide

We start loading the expression data

Files provided by 10x are:

* **the matrix.mtx:**  
  This is a sparse matrix for a more efficient manipulation.  
  Rows: genes  
  Columns: barcodes associated to each spot.
* **features.tsv:**  
  This file contains row indices of the matrix file. Every feature has an ID and a name stored in the first and second columns, respectively. The third column specify the type of feature , which will be one of the following options depending on the feature type:
  + Gene Expression (this case)
  + Antibody Capture
  + CRISPR Guide Capture
  + Multiplexing Capture
  + CUSTOM
* In this analysis, features correspond to Gene Expression data: the ID correspond to gene\_id and the name to the gene\_name.
* **barcodes.tsv:**  
  Barcode sequences correspond to column indices of the matrix file. Each barcode sequence contains a suffix with a dash separator followed by a number, such as ‘-1’

Let’s import the raw counts and the HE image of the slide.

| What | Path to |
| --- | --- |
| raw counts | ../input/GSM6506114\_SP5/ |
| HE image | ../input/GSM6506114\_SP5/spatial/tissue\_lowres\_image.png |

Tip: use the Seurat object Seurat that contains both the spot-level expression data along with the associated image of the tissue slice. A Seurat object is a specialized data structure used in the Seurat package, which is designed for the analysis and visualization of scRNA-seq data. The Seurat object is a container that organizes and stores all the data associated with a single-cell experiment and allows for efficient data management and streamlined analysis workflows. In the context of ST analysis, the Seurat object can be extended to handle spatial information, enabling the study of gene expression in the context of tissue architecture.

–> @team here they should find the way to import data

–> @team here is the solution

By using Read10X() function, we load the sparse data matrix of gene expression. We specify the directory containing the file.  
gene.column refers to the column number in the feature.tsv file where the gene names are stored.  
cell.column indicates the column number where the cell names are specified in the barcodes.tsv file.  
unique.features is a logical parameter to specify whether to make gene names unique and avoid redundant names.  
strip.suffix is a logical parameter (default FALSE) to remove the suffix ‘-1’ from all spot barcodes

counts <- Read10X(data.dir = "../input/GSM6506114\_SP5/"  
 , gene.column = 2  
 , cell.column = 1  
 , unique.features = TRUE  
 , strip.suffix = FALSE) # KEPT to match spot in the img (the reading function does not trim the "-1")  
str(counts)

## Formal class 'dgCMatrix' [package "Matrix"] with 6 slots  
## ..@ i : int [1:5944351] 39 52 54 71 102 154 216 236 277 340 ...  
## ..@ p : int [1:2125] 0 897 5090 7437 9411 14604 15773 17255 19894 21407 ...  
## ..@ Dim : int [1:2] 33538 2124  
## ..@ Dimnames:List of 2  
## .. ..$ : chr [1:33538] "MIR1302-2HG" "FAM138A" "OR4F5" "AL627309.1" ...  
## .. ..$ : chr [1:2124] "AAACAAGTATCTCCCA-1" "AAACAGAGCGACTCCT-1" "AAACAGTGTTCCTGGG-1" "AAACCGTTCGTCCAGG-1" ...  
## ..@ x : num [1:5944351] 1 2 1 1 1 1 1 1 1 1 ...  
## ..@ factors : list()

As we can see the counts table is an object of class dgTMatrix, which is a sparse matrix. This format is used since spatial transcriptomics (ST) data contains many zeros, so sparse matrices are employed to keep the counts table as light as possible.

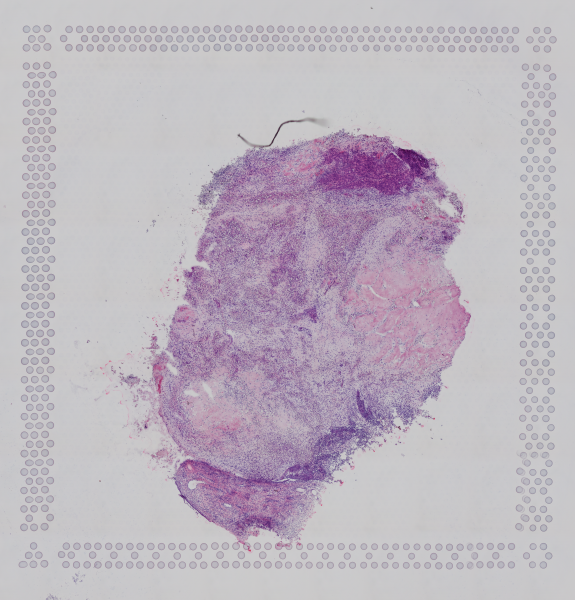
Now we create a Seurat object from raw counts data.

counts parameter takes the matrix-like object with raw counts.  
In project we explicit the project name for the Seurat object.  
in assay we specify the name of initial assay.

The sp object contains:  
- orig.ident: the identifier of the sample.  
- nCount\_spatial: the total number of counts per cell/spot  
- nFeature\_spatial: the number of detected features (genes) per cell/spot

sp <- CreateSeuratObject(counts = counts   
 , project = 'GSM6506114\_SP5'  
 , assay = 'Visium10x\_008um'  
)  
  
DefaultAssay(sp) <- 'Visium10x\_008um' #The default assay is set to `Visium10x\_008um`

Let’s include the low-resultion tissue image, which are outputs of Space Ranger, in the Seurat obj



Now we retrieve the image metadata from 10X Visium experiment and create an img object.  
Load in the 10X Genomics Visium Image (ref: <https://github.com/satijalab/seurat/issues/5806>) in the img object to extrapolate the spatial 2D coordinates of each spot.

image.name is the file name of the PNG we are loading in  
assay specify the associated assay in the Seurat object we are building  
slice is the name for the current image  
filter.matrix is a boolean option to retain only spots in the count matrix that have been detected to be over the tissue image.

img <- Read10X\_Image(image.dir = "../input/GSM6506114\_SP5/spatial/"  
 , image.name = "tissue\_lowres\_image.png"  
 , assay = 'Visium10x\_008um'  
 , slice = 'slice1'  
 , filter.matrix = TRUE  
 )  
  
# Getting the cell (spots) names  
sp\_cells = Cells(sp)  
  
# Retrieve spots associated to features in the SeuratObject `sp` from the image object `img`  
img <- img[sp\_cells]  
  
# Add the image to the Defaulf object `sp`  
sp[["slice"]] <- img

# Test your dataset

Let’s interrogate your data. Please provide a quantitative assessment of the performance of this ST assay

–> @team here they should find QC stastics to describe the data

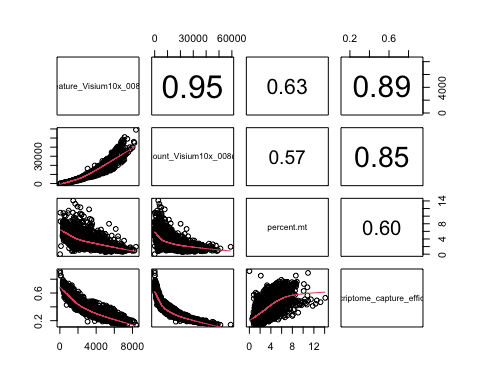
Tip: if you are familiar with single-cell data, you can exploit some statics that are conventionally used in this field.

–> @team here the solution. WE MAY HAVE FEW MORE QCs

Let’s :

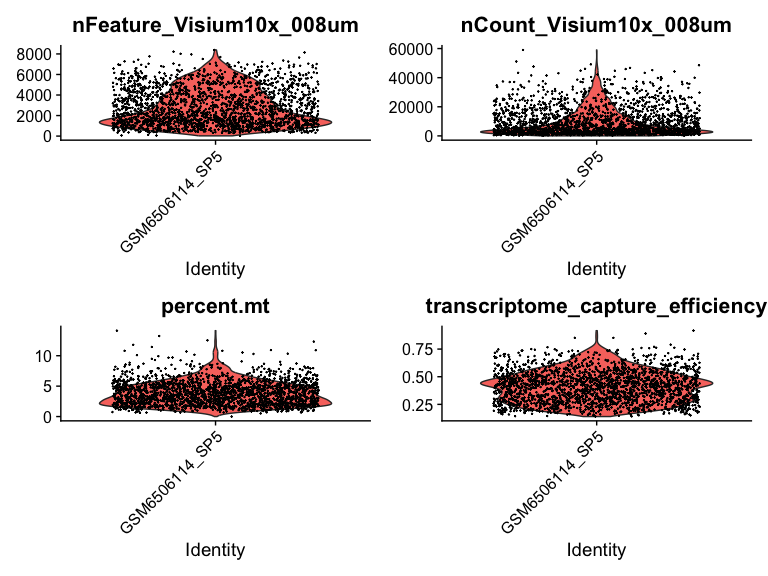
* assess the percentage of MT genes;
* the capture efficiency;
* test the dependencies of the stats;
* visualize the results.

mt\_percentage = PercentageFeatureSet(sp, pattern = "^MT-")  
  
# Adding the mt\_percentage as metadata to the Seurat object  
sp = AddMetaData(object = sp, metadata = mt\_percentage, col.name = "percent.mt")  
  
  
sp@meta.data$transcriptome\_capture\_efficiency = with(sp@meta.data, nFeature\_Visium10x\_008um/nCount\_Visium10x\_008um)  
  
selected\_features = c("nFeature\_Visium10x\_008um","nCount\_Visium10x\_008um", "percent.mt"  
 , "transcriptome\_capture\_efficiency" )  
  
# `pairs` creates a matrix of scatter plots for visualizing relationships between the selected features  
  
pairs(sp@meta.data[,selected\_features]  
 , lower.panel = panel.smooth  
 , upper.panel = panel.cor)



VlnPlot creates violin plots that allow to visualize the distribution of the selected features

VlnPlot(sp, layer = 'counts', features = selected\_features, pt.size = 0.1, ncol = 2)

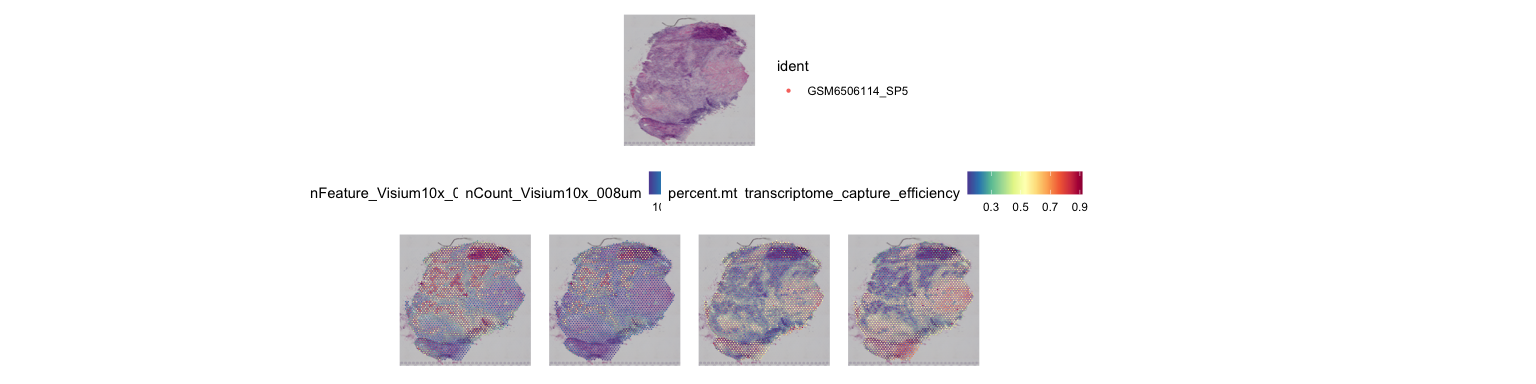


The plot presented above visualizes the distribution of the selected features across all spots.

* nFeature\_Visium10x\_008um illustrates the distribution of the number of unique genes detected per spot.
* nCount\_Visium10x\_008um shows the distribution of total UMI counts per spot.
* percent.mt shows the percentage of reads mapping to mitochondrial genes per spot.
* transcriptome\_capture\_efficiency represents the transcriptome capture efficiency across spots.

Let’s see the statistics on the tissue slide

HE\_slide = SpatialDimPlot(sp, pt.size.factor = 0.5)  
FS\_slide = SpatialFeaturePlot(sp, features = selected\_features, combine = T, ncol = 4 )   
# Combine the plots  
patchwork::wrap\_plots(HE\_slide, FS\_slide, ncol = 1)



# Identification of cluster of spots

Let’s define cluster of spot that share similar trascriptomic profiles. Provide a visualization of clusters on the slide.

—> @team We can ask for an assessment of clustering robustness

Tip: select the proper count normalization before detecting the clusters

—> @ team Here’s the solution

We perform standard log-normalization and exploit the findCluster function of Seurat

We need to identify highly variable features in the dataset as implemented in FindVariableFeatures function. Then, we exploit ScaleData function to scale and center the data for downstream analyses

We use FindNeighbors function to compute the nearest neighbours for clustering, it finds the nearest neighbors of each cell based on the first 30 PCs

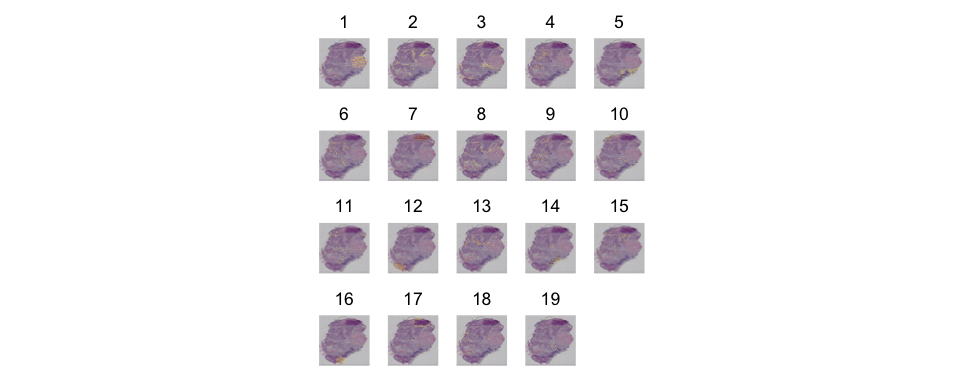
Finally, FindClusters identifies clusters in the data

The identified clusters are visualized over the spatial image, this helps to see the spatial distribution of different gene expression patterns

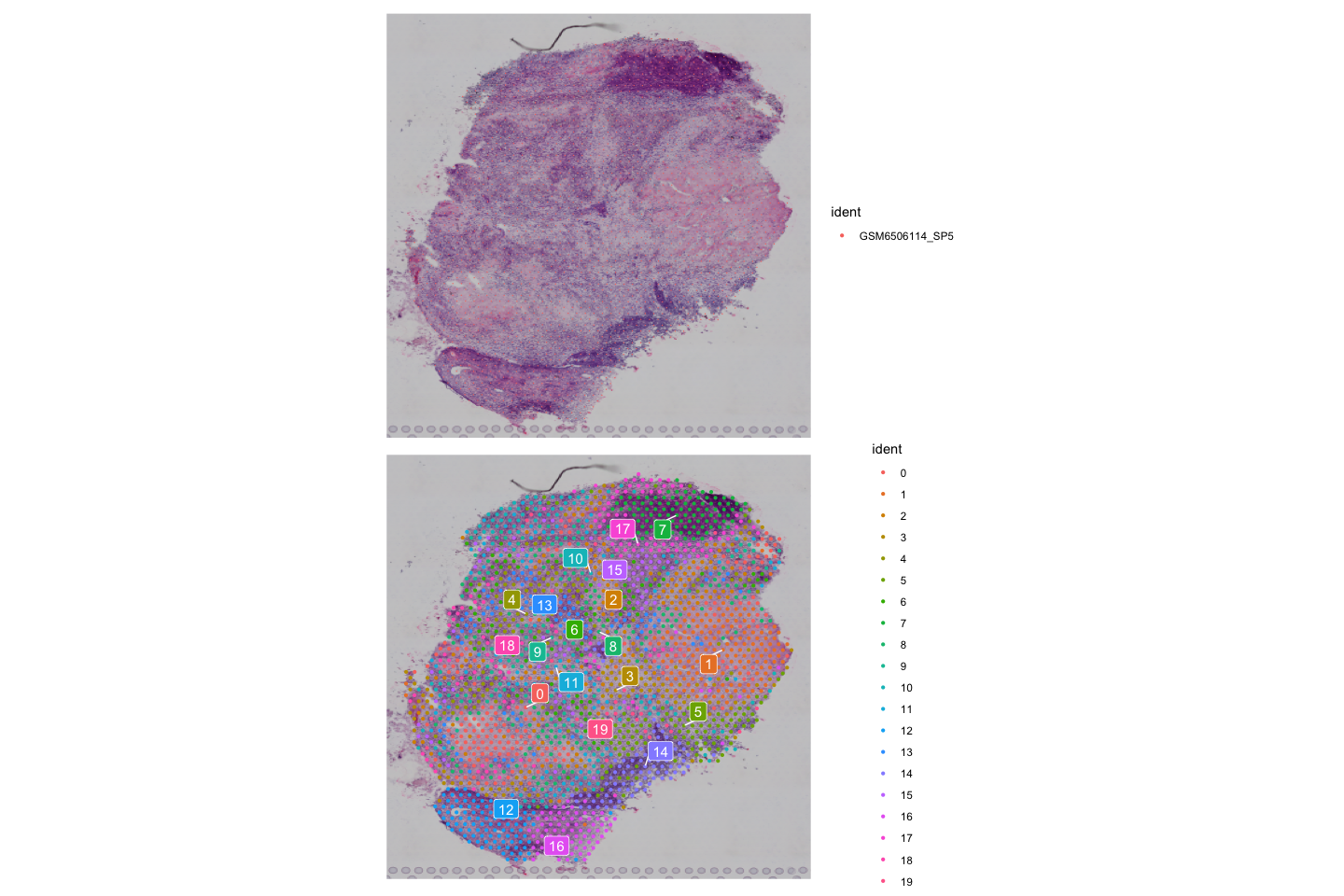
Idents(sp) <- "seurat\_clusters"   
#retrieves and sort cells by their cluster identities, storing the result in the `cells` variable   
cells <- CellsByIdentities(sp, idents = sort(as.numeric(unique(sp@meta.data$seurat\_clusters))))   
str(cells)

## List of 19  
## $ 1 : chr [1:161] "AAAGTGTGATTTATCT-1" "AAAGTTGACTCCCGTA-1" "AACCCTACTGTCAATA-1" "AACCGCTAAGGGATGC-1" ...  
## $ 2 : chr [1:153] "AAAGGCTACGGACCAT-1" "AAAGGCTCTCGCGCCG-1" "AAATTACCTATCGATG-1" "AACAGCTGTGTGGCAA-1" ...  
## $ 3 : chr [1:129] "AAACAAGTATCTCCCA-1" "AAACGGTTGCGAACTG-1" "AAAGTAGCATTGCTCA-1" "AACACACGCTCGCCGC-1" ...  
## $ 4 : chr [1:114] "AACGGCCATCTCCGGT-1" "AACGTACTGTGGGTAC-1" "AAGGAGCGGTTGGTGC-1" "AAGTCAATTGTCGTCA-1" ...  
## $ 5 : chr [1:114] "AAACTCGGTTCGCAAT-1" "AAATCTAGCCCTGCTA-1" "AACGCTGTTGCTGAAA-1" "AAGCCGAAGCGGTTTA-1" ...  
## $ 6 : chr [1:114] "AAACGAGACGGTTGAT-1" "AAATGGTCAATGTGCC-1" "AACCTTTAAATACGGT-1" "AACTCAAGTTAATTGC-1" ...  
## $ 7 : chr [1:112] "AAACAGAGCGACTCCT-1" "AAATAACCATACGGGA-1" "AAATTACACGACTCTG-1" "AACCTTTACGACGTCT-1" ...  
## $ 8 : chr [1:112] "AAACCGTTCGTCCAGG-1" "AAATTAACGGGTAGCT-1" "AACAATACATTGTCGA-1" "AACATATCAACTGGTG-1" ...  
## $ 9 : chr [1:110] "AACAACTGGTAGTTGC-1" "AACGCGGTCTCCAGCC-1" "AACTGGGTCCCGACGT-1" "AATGATGCGACTCCTG-1" ...  
## $ 10: chr [1:108] "AAATTTGCGGGTGTGG-1" "AACAGGAAATCGAATA-1" "AACGATAGAAGGGCCG-1" "AACGCGACCTTGGGCG-1" ...  
## $ 11: chr [1:103] "AAACTGCTGGCTCCAA-1" "AAATACCTATAAGCAT-1" "AAATTGATAGTCCTTT-1" "AACGTCAGACTAGTGG-1" ...  
## $ 12: chr [1:99] "AAACAGTGTTCCTGGG-1" "AACCCGACAACCCGTG-1" "AAGTGCTTCTCTATTG-1" "AATACCGGAGGGCTGT-1" ...  
## $ 13: chr [1:97] "AACGTGCGAAAGTCTC-1" "AAGAGCTCTTTATCGG-1" "AAGAGGCATGGATCGC-1" "AAGCGTCCCTCATCGA-1" ...  
## $ 14: chr [1:88] "AACCTCGCTTTAGCCC-1" "AAGCGCAGGGCTTTGA-1" "AAGTTCACTCCAAGCT-1" "AAGTTTATGGGCCCAA-1" ...  
## $ 15: chr [1:85] "AAATCGTGTACCACAA-1" "AACCGAGCTTGGTCAT-1" "AAGATTGGCGGAACGT-1" "AAGGATGAGGGACCTC-1" ...  
## $ 16: chr [1:84] "AAATGACTGATCAAAC-1" "AACCCGCTGTATTCCA-1" "AATCCCGCTCAGAGCC-1" "ACACATTTCCGTAGAC-1" ...  
## $ 17: chr [1:82] "AAATGGCATGTCTTGT-1" "AACGATATGTCAACTG-1" "AATGATGATACGCTAT-1" "AATTCATAAGGGATCT-1" ...  
## $ 18: chr [1:55] "AAGTGCGTTAGAATCT-1" "ACGCAAACTAATAGAT-1" "AGAAGGTTGCCGAATT-1" "AGGACGACCCATTAGA-1" ...  
## $ 19: chr [1:31] "AAATAGGGTGCTATTG-1" "AACGATAATGCCGTAG-1" "AAGTGTTTGGAGACGG-1" "ACCACGTGCAGCTATA-1" ...

p <- SpatialDimPlot(sp,  
 cells.highlight = cells[setdiff(names(cells), "NA")],  
 cols.highlight = c("#FFFF00", "grey50"), facet.highlight = T, combine = T) + NoLegend()  
p



cluster\_slide = SpatialDimPlot(sp, label = T, repel = T, label.size = 4)  
  
patchwork::wrap\_plots(HE\_slide, cluster\_slide, ncol=1)



# Getting the gene signatures

Get the markers of your clusters and visualize the XXX top-ranked ones for each cluster.

Tip: XXXX top 50, logFC>1, FDR<0.1

—> @ team Here’s the solution

Use the function `FindAllMarkers’ to get the list of genes that represent each cluster.

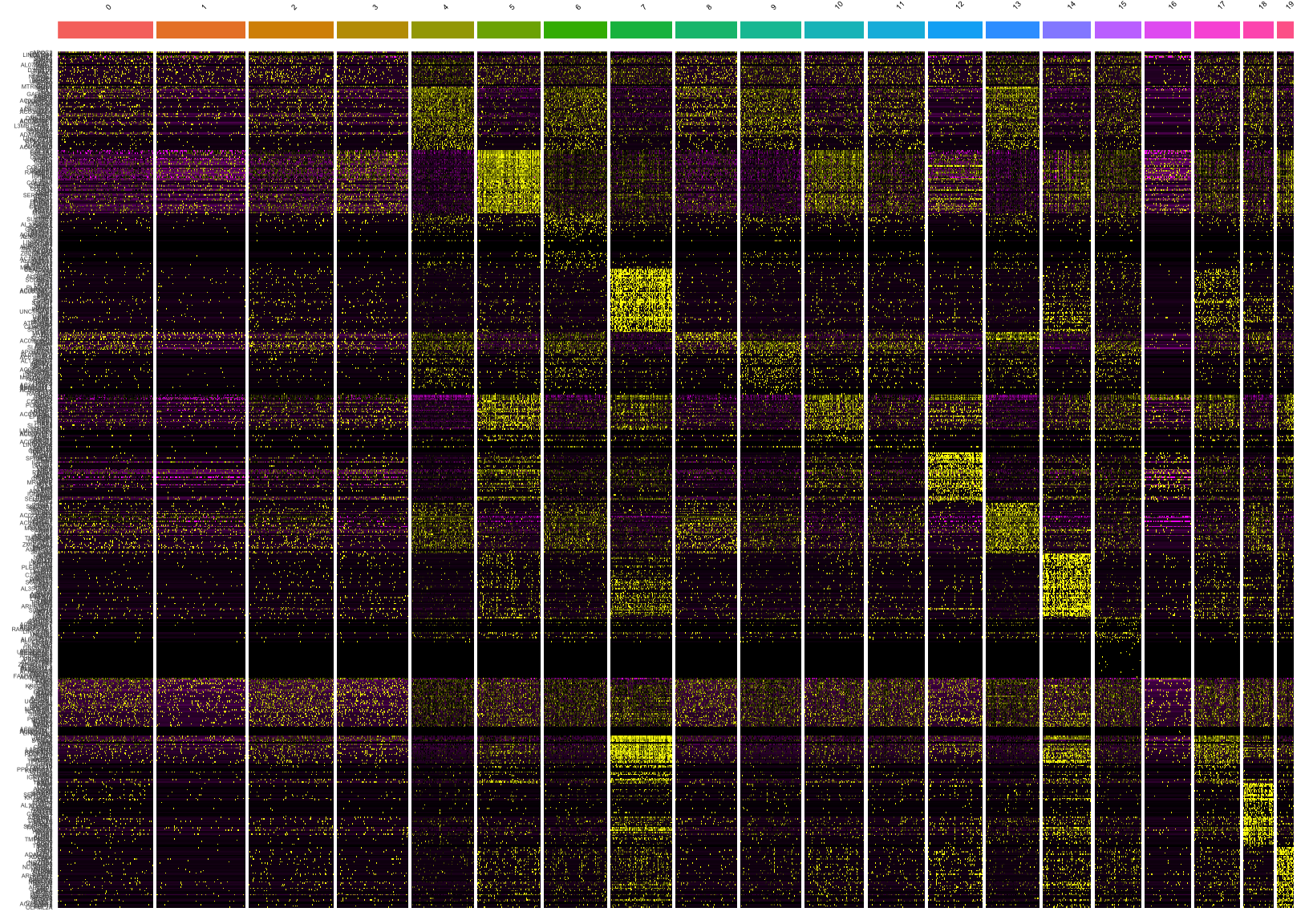
DT::datatable(selected\_markers,rownames = F,style = "auto", class = 'cell-border stripe')

## PhantomJS not found. You can install it with webshot::install\_phantomjs(). If it is installed, please make sure the phantomjs executable can be found via the PATH variable.

ScaleData Scales and centers the data for the selected features (e.g top50$gene)

features specifies the features (genes) to be included in the heatmap

p <- DoHeatmap(sp\_subset, assay = "Visium10x\_008um", features = top50$gene, size = 2.5) +   
 theme(axis.text = element\_text(size = 5.5)) + NoLegend()  
p



# Functional analysis on markers

Let’s identify cancerous and not cancerous cells by annotating markers on hallmarks of cancer and NCG cancer/healthy drivers.  
How many superclass of cell can we spot in the lesion? What are they?

To do so we will exploit:

* the involvement of genes in the Hallmarks of Cancer
* the selective advantage of being cancer and healty drivers genes

Tip: –> @team PLEASE REPORT HERE INFORMATION OF NCG AND HALLMARKS

[NCG](http://network-cancer-genes.org/) is a manually curated collection of cancer genes and healthy drivers.

Hallmark gene sets is collected in the [Human Molecular Signatures Database (MSigDB)](https://www.gsea-msigdb.org/gsea/msigdb/collections.jsp)

–> @team we ask to characterize cluster based on enrichments of hallmarks and cancer/healthy genes. It is nice to see how enrichments can define the cell composition of clusters

—> @ team Here’s the solution

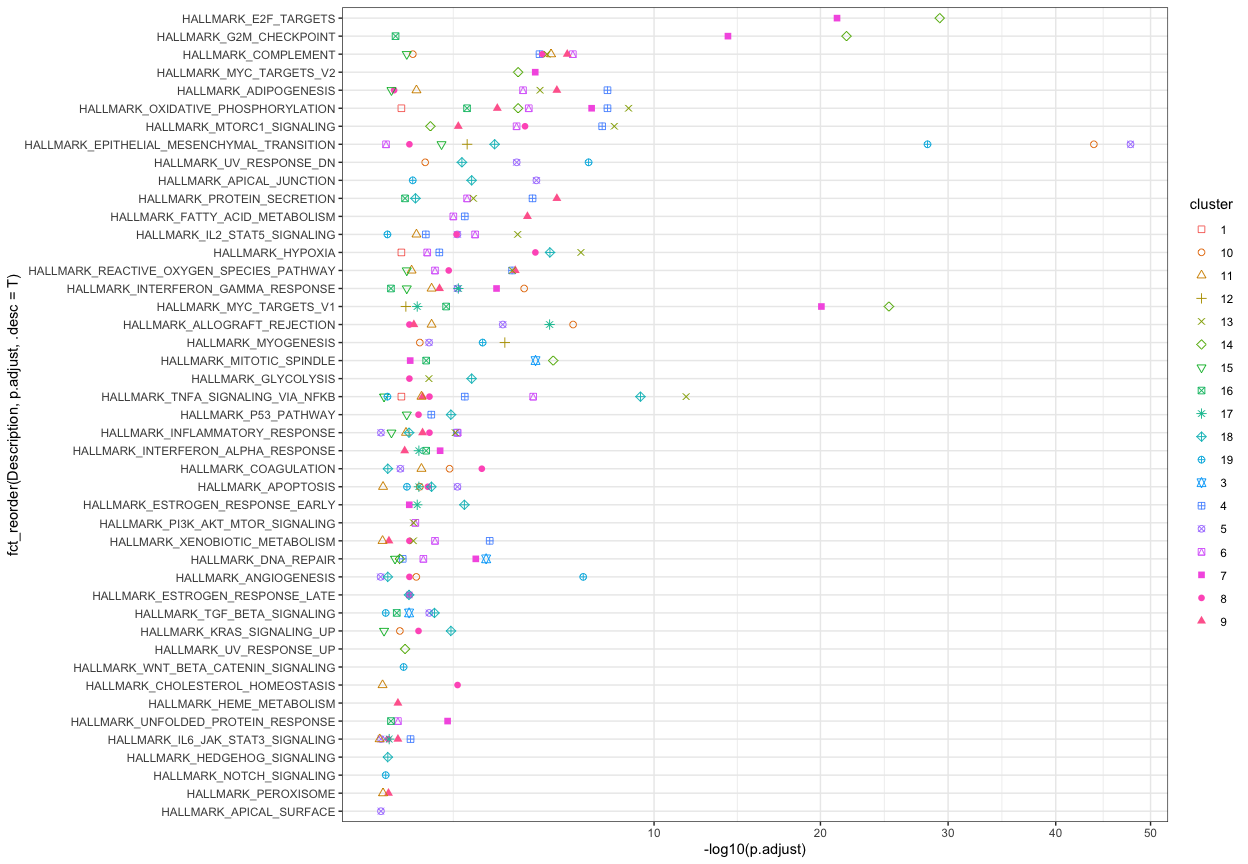
Fetching Hallmark gene sets

enricher performs enrichment analysis the genes of each cluster

DT::datatable(enr,rownames = F,style = "auto", class = 'cell-border stripe')

Visualization of Enrichment results

ggplot(enr,aes(x = -log10(p.adjust),y = fct\_reorder(Description, p.adjust, .desc = T), color=cluster, shape=cluster)) +   
 geom\_point(size=2)+scale\_shape\_manual(values=0:18)+scale\_x\_sqrt()+theme\_bw()



Loading NCG data

cgc = subset(read.delim2('../input/NCG\_cancerdrivers\_annotation\_supporting\_evidence.tsv')  
 , primary\_site %in% c('ovary'))  
healthy = subset(read.delim2('../input/NCG\_healthydrivers\_annotation\_supporting\_evidence.tsv'),  
 organ\_system=='Gynecologic')  
  
message("The dimensions of the cancer genes dataframe are: ", paste(dim(cgc), collapse = " x "))

## The dimensions of the cancer genes dataframe are: 41 x 14

message("The dimensions of the healthy genes dataframe are: ", paste(dim(healthy), collapse = " x "))

## The dimensions of the healthy genes dataframe are: 40 x 8

markers$driver\_cancer\_clone and markers$driver\_not\_cancer\_clone: Flags markers as either cancer drivers or not.

markers$driver\_cancer\_clone = markers$gene%in%cgc$symbol  
markers$driver\_not\_cancer\_clone = markers$gene%in%healthy$symbol

overview: Summarizes the number of markers and the count of cancer and non-cancer drivers per cluster.

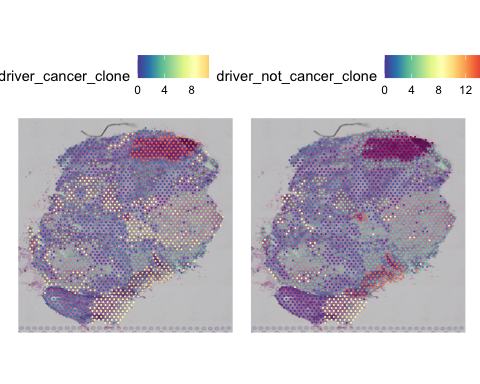
overview = ddply(markers, .(cluster), summarise  
 , markers = n()  
 , driver\_cancer\_clone = sum(driver\_cancer\_clone)  
 , driver\_not\_cancer\_clone = sum(driver\_not\_cancer\_clone)  
) %>% arrange(as.numeric(as.character(cluster)))  
  
DT::datatable(overview,rownames = F,style = "auto", class = 'cell-border stripe')

Adding metadata to Seurat object to visualize cancer driver information

cancer = overview$driver\_cancer\_clone[match(Idents(sp), overview$cluster)]  
names(cancer) <- colnames(sp)  
not\_cancer = overview$driver\_not\_cancer\_clone[match(Idents(sp), overview$cluster)]  
names(not\_cancer) <- colnames(sp)  
  
sp <- AddMetaData(object = sp, metadata = cancer, col.name = 'driver\_cancer\_clone')  
sp <- AddMetaData(object = sp, metadata = not\_cancer, col.name = 'driver\_not\_cancer\_clone')

Visualizing how cancer and non-cancer driver genes are distributed

cgc\_slide = SpatialFeaturePlot(sp, features = c('driver\_cancer\_clone', 'driver\_not\_cancer\_clone')  
 , combine = T, ncol = 2  
 , keep.scale = 'all' )  
cgc\_slide



# Prediction cellular types by unsupervised and supervised approaches

Let’s see whethere we can strengthen the results by inferring cell composition Cell type prediction can be done using:

- an unsupervised approach: CLUSTERMOLE

- a scRNA-guided apporach: RCTD —> @team this may be removed for sake of time and to avoid the use of another dataset

—> @ team Here’s the solution

Loading the required packages

clustermole predicts cell types based on marker enrichment

GetAssayData retrieves assay data from the Seurat object clustermole\_enrichment performs cell type enrichment analysis using the CLUSTERMOLE method

***–>*** @team this is the most relevant step in the analysis, we should drive them towards the selection of this dataset.

Extracting and cleaning ell type information

db\_mrks=clustermole\_markers('hs') #retrieves marker data specific to human cells   
cmh = ddply(subset(db\_mrks, grepl('Human', celltype\_full) & grepl('CellMarker',celltype\_full)), .(db), summarise  
 , celltype\_full=unique(celltype\_full), db=unique(db))  
  
cmh$n\_info = sapply(strsplit(cmh$celltype\_full, "\\|"), length)  
cmh$celltype = sapply(strsplit(cmh$celltype\_full,"\\|"), '[[',1)  
cmh$organ = sapply(strsplit(cmh$celltype\_full,"\\|"), '[[',2)  
selected\_cell\_types = subset(cmh, (n\_info==3 & !grepl("\\(|Embr|Neur|Glia|glia|Purkinje|Pyrami|Germ|germ|Follic|neuro|Adventitial", celltype\_full)) |   
 organ== " Ovary ")$celltype\_full  
rm(cmh, db\_mrks)

Filters and cleans the enrichment results and adds the inferred cell type metadata back to the Seurat object

enr\_res = subset(l, celltype\_full %in% selected\_cell\_types) #includes only the cell types   
enr\_res$celltype\_full = gsub(" | Human | CellMarker", "", enr\_res$celltype\_full)  
enr\_res$celltype\_full = gsub( "\\||", "", enr\_res$celltype\_full)  
  
celltype = enr\_res$celltype\_full[match(Cells(sp), enr\_res$cluster)]  
names(celltype) <- colnames(sp)  
sp <- AddMetaData(object = sp, metadata = celltype, col.name = 'celltype\_full') #Adds the predicted cell types as metadata to the Seurat object

Summarizing enrichment results Identifies the most enriched cell type per cluster

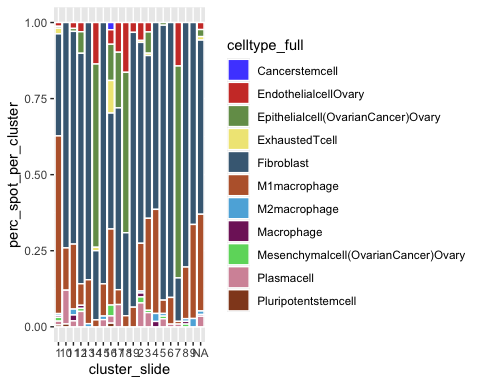
enr\_res = ddply(enr\_res,.(cluster), summarise, celltype\_full=celltype\_full[which(score\_rank==min(score\_rank))])  
  
spots = CellsByIdentities(sp, idents = sort(as.numeric(unique(sp@meta.data$seurat\_clusters))))  
spots = mapply(function(x,y){data.frame(cluster=x, cluster\_slide = y)}, x=spots, y=names(spots), SIMPLIFY = F)  
spots = bind\_rows(spots)

Merges and summarizes the enrichement results with cell spots, calculating the percentage of spots per cluster

enr\_res = left\_join(enr\_res, spots, by = "cluster")  
enr\_res = ddply(enr\_res, .(cluster\_slide, celltype\_full), summarise, n\_spots = n() )   
enr\_res = ddply(enr\_res, .(cluster\_slide), mutate, total\_spots = sum(n\_spots))  
enr\_res$perc\_spot\_per\_cluster= enr\_res$n\_spot/enr\_res$total\_spot

Visualizing cell type composition

ggplot(enr\_res, aes(cluster\_slide, perc\_spot\_per\_cluster, fill=celltype\_full))+  
 geom\_bar(stat='identity', position = position\_stack(), color='white')+  
 ggsci::scale\_fill\_igv()



RCTD: scRNAseq-guided

Loading counts matrix GetAssayData Retrieves raw counts data from the Seurat object

DefaultAssay(sp) <- "Visium10x\_008um"  
counts = GetAssayData(object = sp, assay = 'Visium10x\_008um',layer = 'counts' )

Preparing cell type metadata and reference object for RCTD

Reference Creates a reference object for RCTD with cell type annotations GetTissueCoordinates Retrieves spatial coordinates for tissue sections

Running RCTD

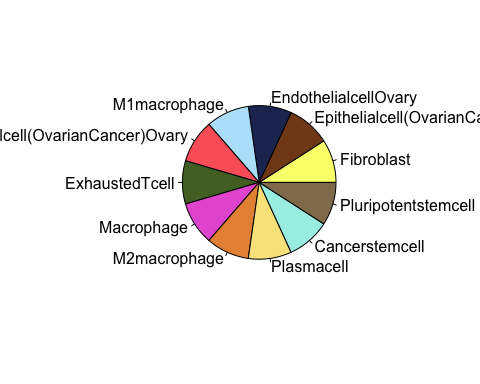
create.RCTD and run.RCTD Runs the RCTD algorithm to predict cell types and add the results to the Seurat object

Add results back to Seurat object

sp <- AddMetaData(sp, metadata = RCTD@results$results\_df)

Visualizing RCTD results

unique\_cell\_type = unique(celltype)  
my\_cols = ggsci::pal\_rickandmorty()(11)  
names(my\_cols) = unique\_cell\_type  
x=rep(1,11);names(x)=names(my\_cols)  
pie(x, col = my\_cols)



# Identification of Ligand receptors

Let’s identify possible cell-to-cell communications via their ligands and receptors

Tip: XXX

—> @ team Here’s the solution

require(tidyverse)  
require(magrittr)

require(liana)

require(dplyr)  
  
sp <- SetIdent(sp, value = "seurat\_clusters")  
sp@meta.data$clusters = as.numeric(as.character(sp@meta.data$seurat\_clusters))+1

Running LIANA.

liana\_wrap performs the ligand-receptor interactions analysis using multiple methods

liana = liana\_wrap(sp,  
 method = c( "natmi" # Network Analysis Toolkit for the Multicellular Interactions Built on connectomeDB2020   
 # of 2,293 human ligand-receptor interactions with primary literature support (in the current manuscript)   
 # and additional 1,778 putative pairs (available as a part of NATMI only), which builds on our previous draft   
 # and a database of human cell interactions (Ramilowski, J. A., et al. Nat Commun 6, 7866 (2015)).   
 # Edges between any pair of cell types are then predicted based on the expression of the ligand in one cell type   
 # and the expression of its cognate receptor in the other cell type. NATMI introduces the concept of   
 # cell-connectivity-summary networks that merge the many ligand–receptor edges drawn from one cell type to   
 # another into a combined weighted cell-connectivity-summary edge to summarise how strongly (or specifically)   
 # each cell type is communicating to another cell type   
 # , "connectome"  
 , "call\_italk" # capture highly abundant ligand-receptor gene (or transcript) pairs, identify gains or losses   
 # of cellular interactions by comparative analysis, and track the dynamic changes of intercellular communication   
 # signals in longitudinal samples.  
 , "call\_cellchat" # quantitatively characterize and compare the inferred cell-cell communication networks using   
 # a systems approach by combining social network analysis, pattern recognition, and manifold learning approache  
 # , "logfc"  
 , "sca" # SingleCellSignalR facilitates the transformation of complex data into higher-order information  
 # SingleCellSignalR regularized score achieves better control of the FP compared to other solutions  
 # , "cellphonedb"  
 ),  
 resource = c("OmniPath"), # Use 'all' to run all 'human' resources in one go  
 idents\_col = 'celltype\_full',  
 # idents\_col = 'clusters',  
 min\_cells = 5,  
 return\_all = FALSE,  
 supp\_columns = c("ligand.expr", "receptor.expr", "ligand.pval", "receptor.pval", "ligand.FDR", "receptor.FDR"),  
 verbose = TRUE,  
 assay = "Visium10x\_008um",  
 .simplify = TRUE,  
 cell.adj = NULL, # DOBBIAMO CALCOLARE QUESTA MATRICE   
 # base = NULL  
 # parallelize = T,  
 # workers = 10,  
 # seed=30580,  
 # assay.type = 'logcounts'  
)

## [1] "Create a CellChat object from a data matrix"  
## Set cell identities for the new CellChat object   
## The cell groups used for CellChat analysis are EndothelialcellOvary, Epithelialcell(OvarianCancer)Ovary, ExhaustedTcell, Fibroblast, M1macrophage, M2macrophage, Macrophage, Mesenchymalcell(OvarianCancer)Ovary, Plasmacell   
## Issue identified!! Please check the official Gene Symbol of the following genes:   
## 2   
## Issue identified!! Please check the official Gene Symbol of the following genes:   
## DEFB4B CGB8 DEFB106B CCL4L1   
## The number of highly variable ligand-receptor pairs used for signaling inference is 1508   
## triMean is used for calculating the average gene expression per cell group.   
## [1] ">>> Run CellChat on sc/snRNA-seq data <<< [2024-07-29 12:56:21.238829]"  
## [1] ">>> CellChat inference is done. Parameter values are stored in `object@options$parameter` <<< [2024-07-29 12:58:42.900845]"

Liana returns a list of results, each element of which corresponds to a method

liana.summary = liana %>% dplyr::glimpse()

## List of 4  
## $ natmi : tibble [63,478 × 18] (S3: tbl\_df/tbl/data.frame)  
## ..$ source : chr [1:63478] "Fibroblast" "Fibroblast" "Fibroblast" "Fibroblast" ...  
## .. ..- attr(\*, "levels")= chr [1:9] "EndothelialcellOvary" "Epithelialcell(OvarianCancer)Ovary" "ExhaustedTcell" "Fibroblast" ...  
## ..$ target : chr [1:63478] "Fibroblast" "Fibroblast" "Fibroblast" "Fibroblast" ...  
## .. ..- attr(\*, "levels")= chr [1:9] "EndothelialcellOvary" "Epithelialcell(OvarianCancer)Ovary" "ExhaustedTcell" "Fibroblast" ...  
## ..$ ligand.complex : chr [1:63478] "ADAM10" "ADAM10" "ADAM10" "ADAM10" ...  
## ..$ ligand : chr [1:63478] "ADAM10" "ADAM10" "ADAM10" "ADAM10" ...  
## ..$ receptor.complex: chr [1:63478] "TSPAN15" "ERBB2" "APP" "IL6R" ...  
## ..$ receptor : chr [1:63478] "TSPAN15" "ERBB2" "APP" "IL6R" ...  
## ..$ receptor.prop : num [1:63478] 0.146 0.137 0.512 0.24 0.134 ...  
## ..$ ligand.prop : num [1:63478] 0.52 0.52 0.52 0.52 0.52 ...  
## ..$ ligand.expr : num [1:63478] 0.542 0.542 0.542 0.542 0.542 ...  
## ..$ receptor.expr : num [1:63478] 0.1032 0.114 0.5267 0.1731 0.0966 ...  
## ..$ ligand.sum : num [1:63478] 4.07 4.07 4.07 4.07 4.07 ...  
## ..$ receptor.sum : num [1:63478] 0.826 1.719 3.793 0.968 1.267 ...  
## ..$ ligand.pval : num [1:63478] 0.461 0.461 0.461 0.461 0.461 ...  
## ..$ receptor.pval : num [1:63478] 0.581 0.666 0.991 0.857 0.688 ...  
## ..$ ligand.FDR : num [1:63478] 1 1 1 1 1 1 1 1 1 1 ...  
## ..$ receptor.FDR : num [1:63478] 1 1 1 1 1 1 1 1 1 1 ...  
## ..$ prod\_weight : num [1:63478] 0.0559 0.0618 0.2854 0.0938 0.0523 ...  
## ..$ edge\_specificity: num [1:63478] 0.01662 0.00882 0.01847 0.02378 0.01014 ...  
## $ call\_italk : tibble [21,571 × 9] (S3: tbl\_df/tbl/data.frame)  
## ..$ source : Factor w/ 9 levels "EndothelialcellOvary",..: 4 4 4 4 4 4 4 4 4 4 ...  
## ..$ ligand : chr [1:21571] "PRNP" "PTPRT" "ADAM10" "ADAM10" ...  
## ..$ target : Factor w/ 9 levels "EndothelialcellOvary",..: 4 4 4 4 4 4 4 4 4 4 ...  
## ..$ receptor : chr [1:21571] "TNFRSF25" "CDH1" "TSPAN12" "ERBB2" ...  
## ..$ logFC\_from: num [1:21571] 0.253 -1.967 0.222 0.222 0.222 ...  
## ..$ logFC\_to : num [1:21571] -0.239 -1.333 -0.952 -0.837 0.11 ...  
## ..$ qval\_from : num [1:21571] 0.000147 1 0.245621 0.245621 0.245621 ...  
## ..$ qval\_to : num [1:21571] 1.00 3.34e-07 1.00 2.81e-03 1.00 ...  
## ..$ logfc\_comb: num [1:21571] 0.00548 0.00548 0.00548 0.00548 0.00548 ...  
## $ call\_cellchat: tibble [24,159 × 6] (S3: tbl\_df/tbl/data.frame)  
## ..$ source : Factor w/ 9 levels "EndothelialcellOvary",..: 1 2 4 5 6 7 8 9 1 2 ...  
## ..$ target : Factor w/ 9 levels "EndothelialcellOvary",..: 1 1 1 1 1 1 1 1 2 2 ...  
## ..$ ligand : chr [1:24159] "PRNP" "PRNP" "PRNP" "PRNP" ...  
## ..$ receptor: chr [1:24159] "TNFRSF25" "TNFRSF25" "TNFRSF25" "TNFRSF25" ...  
## ..$ prob : num [1:24159] 0.0026 0.00281 0.00363 0.00313 0.00441 ...  
## ..$ pval : num [1:24159] 0 0 0 0 0 0 0.01 0 0 0 ...  
## $ sca : tibble [63,478 × 16] (S3: tbl\_df/tbl/data.frame)  
## ..$ source : chr [1:63478] "Fibroblast" "Fibroblast" "Fibroblast" "Fibroblast" ...  
## .. ..- attr(\*, "levels")= chr [1:9] "EndothelialcellOvary" "Epithelialcell(OvarianCancer)Ovary" "ExhaustedTcell" "Fibroblast" ...  
## ..$ target : chr [1:63478] "Fibroblast" "Fibroblast" "Fibroblast" "Fibroblast" ...  
## .. ..- attr(\*, "levels")= chr [1:9] "EndothelialcellOvary" "Epithelialcell(OvarianCancer)Ovary" "ExhaustedTcell" "Fibroblast" ...  
## ..$ ligand.complex : chr [1:63478] "ADAM10" "ADAM10" "ADAM10" "ADAM10" ...  
## ..$ ligand : chr [1:63478] "ADAM10" "ADAM10" "ADAM10" "ADAM10" ...  
## ..$ receptor.complex: chr [1:63478] "TSPAN15" "ERBB2" "APP" "IL6R" ...  
## ..$ receptor : chr [1:63478] "TSPAN15" "ERBB2" "APP" "IL6R" ...  
## ..$ receptor.prop : num [1:63478] 0.146 0.137 0.512 0.24 0.134 ...  
## ..$ ligand.prop : num [1:63478] 0.52 0.52 0.52 0.52 0.52 ...  
## ..$ ligand.expr : num [1:63478] 0.542 0.542 0.542 0.542 0.542 ...  
## ..$ receptor.expr : num [1:63478] 0.1032 0.114 0.5267 0.1731 0.0966 ...  
## ..$ global\_mean : num [1:63478] 0.14 0.14 0.14 0.14 0.14 ...  
## ..$ ligand.pval : num [1:63478] 0.461 0.461 0.461 0.461 0.461 ...  
## ..$ receptor.pval : num [1:63478] 0.581 0.666 0.991 0.857 0.688 ...  
## ..$ ligand.FDR : num [1:63478] 1 1 1 1 1 1 1 1 1 1 ...  
## ..$ receptor.FDR : num [1:63478] 1 1 1 1 1 1 1 1 1 1 ...  
## ..$ LRscore : num [1:63478] 0.628 0.639 0.792 0.686 0.62 ...

liana.summary <- liana.summary %>% liana\_aggregate()

liana.summary=ddply(liana.summary, .(source,target), mutate, ranking = rank(aggregate\_rank))

Visualize the different components of liana

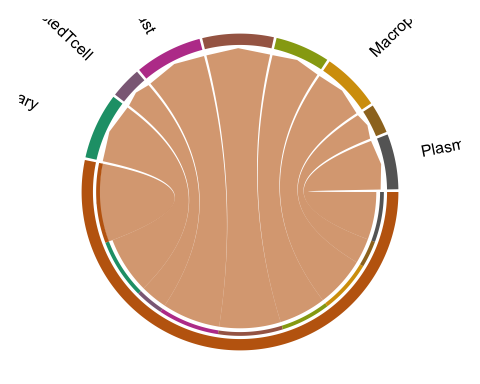
We can aggregate these results into a tibble with consensus ranks

# liana.summary = readRDS("liana\_summary.rds")  
  
heat\_freq(liana.summary)

A pink squares with white text

Description automatically generated

chord\_freq(liana.summary,  
 source\_groups = c("Epithelialcell(OvarianCancer)Ovary"))



–> @team here the most recurrent receptor is DDR1 (@Francesca did it remind you anything?)

top.ranked = subset(liana.summary, ranking<=5)  
  
lr = subset(top.ranked, source%in%c("Macrophage", "M1macrophage", "M2macrophage","ExhaustedTcell") & target=="Epithelialcell(OvarianCancer)Ovary")  
DT::datatable(lr,rownames = F,style = "auto", class = 'cell-border stripe')

–> @team PLEASE PLOT THE EXPRESSION OF DDR1 ON THE SLIDE

SpatialFeaturePlot(sp, features = "DDR1" ) + ggtitle("DDR1 expression")

