SingleGEO: Query and Integrative Analysis of High-throughput Single Cell Sequencing Data from GEO

Author: Yuanqing Yan bioinformatics.yuanqing@gmail.com

07 January, 2025

Description

Single cell sequencing technologies have advanced the research to examine the genomic alterations from individual cells. Currently a huge amount of single cell genetic data have been deposited in GEO and the number is keeping on exponentially increasing. These single cell high-throughput sequencing data come from different tissues, health status, disease stage and drug treatments. Various machine learning algorithms, such as canonical correlation analysis and reciprocal principal component analysis, have been adapted in single cell data analysis. The implementation of such algorithms correct the potential batch effects and make the data integration between different datasets feasible. SingleGEO is developed to efficiently query, download high-throughput single cell sequencing data from GEO for further integrative analysis. The program will initially download GEO meta database into local drive and implement a fast and light-weight searching engine to query the database. By feeding the program with the keywords relevant to the user's research interests, the query results with specific meta information will be returned for further investigation. Once the targeted GSE IDs have been selected, the single cell sequencing data deposited in GEO supplement can be downloaded into local derive.

Installation

Installing singleGEO from GitHub

```
library(devtools)
devtools::install_github("yuanqingyan/singleGEO")
```

Download GEO meta database

SingleGEO uses sql to query the database. To perform the database searching, GEO meta database should be first downloaded and save in the local drive. The name of the geo meta dataset usually is GEOmetadb.sqlite. If the meta database has been already downloaded and not the in the current working folder, specify the path of this file. In this vignett, a demo file is used for demonstration purpose.

```
library(singleGEO)
##Download the GEO meta database. Remove "Demo=TRUE" or set "Demo=FALSE" when you start your
        own project.
MyDataBase<-GetGeoMetaDatabase(Sqlfile=NULL,Demo=TRUE)
#> [1] "Using demo GEO meta database"
```

Query the database

To query the database, the organisms and a list of keywords should be provided. For the organism, one or multiple organisms can be provided. By providing a list of keywords, the query will return the results with the keywords described in GEO meta data. One or multiple keywords can be provided. For each keyword, to overcome the ambiguity, multiple options can be provided. For example, "pulmonary" could be an alternative word used for lung study. The program will searching the database by recognizing the different options for each key point. The following code is to search the potential available GEO datasets, assuming we are interested in scRNAseq data in lung study from both human and mouse.

```
# Will study mouse and human data
MyOrganism<-c("Mus musculus","Homo sapiens")
# Will study lung. Some study may use "pulmonary" instead of "lung", so two options are
        specified
search_kw1<-c("lung","pulmonary")
KeyList<-list(kw1=search_kw1)
MySearch<-Get_Keyword_Meta(GeoDataBase=MyDataBase,
                           Organism=MyOrganism,
                           InputSearchList=KeyList)
#> [1] "Database searching"
#> [1] "---Step1: Filter organism"
#> [1] "---Step2: Obtain unique GSE"
#> [1] "---Step3: Obtain GSE Info"
```

```
colnames(MySearch)
#> [1] "title"             "gse"               "summary"
#> [4] "type"              "overall_design"    "supplementary_file"
unique(MySearch$gse)
#>  [1] "GSE42564"  "GSE52583"  "GSE69405"  "GSE73121"  "GSE86618"  "GSE94555"
#>  [7] "GSE98048"  "GSE96106"  "GSE97168"  "GSE106960" "GSE109444" "GSE103918"
#> [13] "GSE103919" "GSE104154" "GSE99254"  "GSE117618" "GSE102580" "GSE103354"
#> [19] "GSE117450" "GSE117617" "GSE118704" "GSE118706" "GSE116031" "GSE113320"
#> [25] "GSE121611" "GSE122960" "GSE126205" "GSE126906" "GSE126908" "GSE100412"
#> [31] "GSE115730" "GSE128033" "GSE123838" "GSE128169" "GSE131800" "GSE124258"
#> [37] "GSE135893" "GSE121600" "GSE127803" "GSE136580" "GSE129914" "GSE139229"
#> [43] "GSE139231" "GSE140431" "GSE133992" "GSE108378" "GSE142285" "GSE142286"
#> [49] "GSE124323" "GSE124324" "GSE137026" "GSE138585" "GSE123902" "GSE137912"
#> [55] "GSE129937" "GSE130077" "GSE132771" "GSE135167" "GSE128822" "GSE131907"
#> [61] "GSE134174" "GSE137353" "GSE132533" "GSE132534" "GSE132910" "GSE132914"
#> [67] "GSE143705" "GSE143706" "GSE154869" "GSE154870" "GSE154965" "GSE154966"
#> [73] "GSE154977" "GSE154978" "GSE154989" "GSE123405" "GSE149655" "GSE140032"
#> [79] "GSE149813" "GSE137950" "GSE146981" "GSE141141" "GSE142246" "GSE140203"
#> [85] "GSE158127" "GSE160760" "GSE160876" "GSE161648" "GSE149878" "GSE159354"
#> [91] "GSE160664" "GSE151974" "GSE161934" "GSE162045"


head(MySearch[,c("title","gse")])


#>
title
#> 1                                                                          Gene
expression analysis of lung fibroblasts in bleomycin-induced pulmonary fibrosis
#> 2                                                                          Gene
expression analysis of lung fibroblasts in bleomycin-induced pulmonary fibrosis
#> 3 High throughput quantitative whole transcriptome analysis of distal mouse lung epithelial
cells from various developmental stages (E14.5, E16.5, E18.5 and adult)
#> 4 High throughput quantitative whole transcriptome analysis of distal mouse lung epithelial
cells from various developmental stages (E14.5, E16.5, E18.5 and adult)
#> 5 High throughput quantitative whole transcriptome analysis of distal mouse lung epithelial
cells from various developmental stages (E14.5, E16.5, E18.5 and adult)
#> 6 High throughput quantitative whole transcriptome analysis of distal mouse lung epithelial
cells from various developmental stages (E14.5, E16.5, E18.5 and adult)
#>        gse
#> 1 GSE42564
#> 2 GSE42564
#> 3 GSE52583
#> 4 GSE52583
#> 5 GSE52583
#> 6 GSE52583


MySearch[MySearch$gse=="GSE158127","summary"][1]


#> [1] "Lung transplantation can potentially be a life-saving treatment for patients with non-
resolving COVID-19-associated respiratory failure. Concerns limiting transplant include
recurrence of SARS-CoV-2 infection in the allograft, technical challenges imposed by viral-
mediated injury to the native lung, and potential risk for allograft infection by pathogens
associated with ventilator-associated pneumonia in the native lung. Most importantly, the
native lung might recover, resulting in long-term outcomes preferable to transplant. Here, we
report results of the first successful lung transplantation procedures in patients with non-
resolving COVID-19-associated respiratory failure in the United States. We performed sm-FISH to
detect both positive and negative strands of SARS-CoV-2 RNA in the explanted lung tissue,
extracellular matrix imaging using SHIELD tissue clearance, and single cell RNA-Seq on explant
and warm post-mortem lung biopsies from patients who died from severe COVID-19 pneumonia. Lungs
from patients with prolonged COVID-19 were free of virus but pathology showed extensive
evidence of injury and fibrosis which resembled end-stage pulmonary fibrosis. We used a machine
learning approach to project single cell RNA-Seq data from patients with late stage COVID-19
onto a single cell atlas of pulmonary fibrosis, revealing similarities across cell lineages.
There was no recurrence of SARS-CoV-2 or pathogens associated with pre-transplant ventilator
associated pneumonias following transplantation. Our findings suggest that some patients with
severe COVID-19 develop fibrotic lung disease for which lung transplantation is the only option
for survival."
```

SingleGEO can query scRNAseq and scATACseq datasets simultaneously. The following code searches the datasets which are from either scRANseq or scATACseq and relevant to human adenocarcinoma disease.

```
Organism_hs<-c("Homo sapiens")
Data_platform<-c("scRNAseq","scATACseq")
search_kw_ade<-c("adenocarcinoma")
KeyList_ade<-list(kw_ad=search_kw_ade)
```

```
MySearch_ade<-Get_Keyword_Meta(GeoDataBase=MyDataBase,
                               Organism=Organism_hs,
                               InputSearchList=KeyList_ade,
                               Platform=Data_platform)
#> [1] "Database searching"
#> [1] "---Step1: Filter organism"
#> [1] "---Step2: Obtain unique GSE"
#> [1] "---Step3: Obtain GSE Info"
unique(MySearch_ade$gse)
#>  [1] "GSE69405"  "GSE97168"  "GSE99254"  "GSE117618" "GSE117450" "GSE117617"
#>  [7] "GSE118704" "GSE118706" "GSE126906" "GSE126908" "GSE108378" "GSE142285"
#> [13] "GSE142286" "GSE123902" "GSE137912" "GSE128822" "GSE131907" "GSE154869"
#> [19] "GSE154870" "GSE149655" "GSE141141" "GSE162045"
```

Title of one scATACseq study

```
MySearch_ade[MySearch_ade$gse=="GSE142285","title"][1]
```

```
#> [1] "Designing a single cell ATAC-Seq (scATAC-Seq) dataset to validate long read RNA-Seq
isoforms and benchmark scATAC-Seq analysis methods"
```

For some studies, a combination of both scRNAseq and scATACseq are performed. SingleGEO can identify such datasets by intersecting the query results. The following code is to search for mouse brain study with both scRNAseq and scATACseq.

```
MyOrganism_ms<-c("Mus musculus")
search_brain<-c("brain")
KeyList_brain<-list(kw_brain=search_brain)

MySearch_brain_scRNA<-Get_Keyword_Meta(GeoDataBase=MyDataBase,
                                       Organism=MyOrganism_ms,
                                       InputSearchList=KeyList_brain,
                                       Platform="scRNAseq")
#> [1] "Database searching"
#> [1] "---Step1: Filter organism"
#> [1] "---Step2: Obtain unique GSE"
#> [1] "---Step3: Obtain GSE Info"
MySearch_brain_scATAC<-Get_Keyword_Meta(GeoDataBase=MyDataBase,
                                        Organism=MyOrganism_ms,
                                        InputSearchList=KeyList_brain,
                                        Platform="scATACseq")
#> [1] "Database searching"
#> [1] "---Step1: Filter organism"
#> [1] "---Step2: Obtain unique GSE"
#> [1] "---Step3: Obtain GSE Info"
(Both_RNA_ATAC<-intersect(unique(MySearch_brain_scRNA$gse),
                          unique(MySearch_brain_scATAC$gse)))
#> [1] "GSE126074" "GSE132534"
```

```
MySearch_brain_scRNA[MySearch_brain_scRNA$gse==Both_RNA_ATAC[1],"overall_design"][1]
```

```
#> [1] "Single nuclei RNA-seq and ATAC-seq co-assay for BJ, H1, K562 and GM12878 cell mixture,
and wild type postnatal day 0 mouse brain cerebral cortex."
```

To obtain more specific studies, multiple keywords should be provided. The following codes can be used to query scRNAseq datasets in mouse, and such datasets should focus on lung fibrosis and new or novel subtype of cell population should be reported.

```
search_kw2<-c("fibrosis")
search_kw3<-c("new","novel")
search_kw4<-c("subtype","subpopulation")
KeyList_new<-list(kw1=search_kw1,
                  kw2=search_kw2,
                  kw3=search_kw3,
                  kw4=search_kw4)
MySearch_new<-Get_Keyword_Meta(GeoDataBase=MyDataBase,
                               Organism=MyOrganism_ms,
                               InputSearchList=KeyList_new)
#> [1] "Database searching"
#> [1] "---Step1: Filter organism"
#> [1] "---Step2: Obtain unique GSE"
#> [1] "---Step3: Obtain GSE Info"
```

```r
unique(MySearch_new$gse)
#> [1] "GSE104154"
```

Title of this gse dataset

```r
MySearch_new[1,"title"]
```

```
#> [1] "Single Cell Analysis of Pulmonary Fibrotic Mesenchymal Cells Indicates a New Subtype"
```

Summary of this study

```r
MySearch_new$summary[1]
```

```
#> [1] "Stromal  taxonomy consists of multiple cell subtypes that play a critical role in
tissue repair or regeneration, fibrosis, inflammation, angiogenesis and tumor formation.
However, their identities are not fully understood, as these conventional classified
populations have historically been defined by restricted sets of markers. We performed single-
cell RNA sequencing to investigate stromal mesenchymal cells (MCs) in normal and fibrotic mouse
lung. Interrogated patterns of signature genes, lncRNAs, extracellular and plasma membrane
genes, and top transcription factors expression were defined to characterize and classify 7 MC
types in normal status. A specific new subset was discovered in fibrotic stage. Delineation of
their differentiation trajectory was achieved by a machine learning method. This collection of
transcriptional scRNA-seq data uncovered core information and provided a valuable resource for
understanding the mesenchymal landscape in fibrotic diseases."
```

Design of this study {r,eval=FALSE MySearch_new$overall_design[1]

```
#> [1] "Stratification of lung mesenchymal subtypes in normal and fibrotic stages."
```

We can also obtain the gsm information of each study.

```r
gsm_info<-Get_GSMFromGSE(GeoDataBase=MyDataBase,
                         GSE.ID=MySearch_new$gse[1])
colnames(gsm_info)
#> [1] "title"                 "gsm"                   "gse"
#> [4] "source_name_ch1"       "characteristics_ch1"   "treatment_protocol_ch1"
#> [7] "treatment_protocol_ch2"
```

Gsm title and id

```r
gsm_info[,c("title","gsm")]
```

```
#>         title        gsm
#> 1  d0_SMA+Tm+ GSM2790885
#> 2  d0_SMA-Tm+ GSM2790886
#> 3  d0_SMA-Tm- GSM2790887
#> 4 d21_SMA+TM+ GSM2790888
#> 5 d21_SMA-TM+ GSM2790889
#> 6 d21_SMA-TM- GSM2790890
```

SingleGEO can be used to query different datasets for integrative analysis. Suppose we are also doing a fas-signaling project in lung fibrosis (fas-signaling has been reported to be up-regulated in lung epithelial cells from patients with idiopathic pulmonary fibrosis) and wondering whether there are some scRNAseq datasets available in mouse. If we could find some, we would like to integrate them with the previous searching result (novel cell type in mouse lung fibrosis) and evaluate whether fas-signaling pathway affect the novel cell type (this example will be described in data integration part).

```r
search_kw5<-c("fas-signaling","fas signaling","fas pathway","fas-pathway")
KeyList_fas<-list(kw1=search_kw1,
                  kw2=search_kw2,
                  kw5=search_kw5)
MySearch_fas<-Get_Keyword_Meta(GeoDataBase=MyDataBase,
                               Organism=MyOrganism_ms,
                               InputSearchList=KeyList_fas)
#> [1] "Database searching"
#> [1] "---Step1: Filter organism"
#> [1] "---Step2: Obtain unique GSE"
#> [1] "---Step3: Obtain GSE Info"
unique(MySearch_fas$gse)
#> [1] "GSE161648"
```

Title of this study

```
MySearch_fas[1,"title"]
```

```
#> [1] "Loss of Fas-signaling in pro-fibrotic fibroblasts impairs homeostatic fibrosis
resolution and promotes persistent pulmonary fibrosis"
```

Design of this study {r,eval=FALSE MySearch_fas$overall_design[1]

```
#> [1] "Bulk RNA-seq,: Naïve(4 reps) + 3wk_Fas-- (4 reps) + 3wk_Fas++ (4 reps) + 6wk_Fas-- (4
reps) + 6wk_Fas++ (4 reps)  scRNA-seq: paired Naive, 3wk_Fas--, 3wk_Fas++, 6wk_Fas--,
6wk_Fas++"
```

Summary of this study

```
MySearch_fas$summary[1]
```

```
#> [1] "Idiopathic pulmonary fibrosis (IPF) is a progressive, irreversible fibrotic disease of
the distal lung alveoli that culminates in respiratory failure and reduced lifespan. Unlike
normal lung repair in response to injury, IPF is associated with the accumulation and
persistence of fibroblasts and myofibroblasts and continued production of collagen and other
extracellular matrix (ECM) components. Prior in vitro studies have led to the hypothesis that
the development of resistance to Fas-induced apoptosis by lung fibroblasts and myofibroblasts
contibributes to their accumulation in the distal lung tissues of IPF patients. Here, we test
this hypothesis in vivo in the resolving model of bleomycin-induced pulmonary fibrosis in mice.
Using genetic loss-of-function approaches to inhibit Fas signaling in fibroblasts, novel flow
cytometry strategies to quantify lung fibroblast subsets and transcriptional profiling of lung
fibroblasts by bulk and single cell RNA-sequencing, we show that Fas is necessary for lung
fibroblast apoptosis during homeostatic resolution of bleomycin-induced pulmonary fibrosis in
vivo. Furthermore, we show that loss of Fas signaling leads to the persistence and continued
pro-fibrotic functions of lung fibroblasts. Our studies provide novel insights into the
mechanisms that contribute to fibroblast survival, persistence and continued ECM deposition in
the context of IPF and how failure to undergo Fas-induced apoptosis prevents fibrosis
resolution."
```

Available supplement data

```
MySearch_fas$supplementary_file[1]
```

```
#> [1]
"ftp://ftp.ncbi.nlm.nih.gov/geo/series/GSE161nnn/GSE161648/suppl/GSE161648_RAW.tar;\tftp://ftp.ncbi.nlm.nih.gov/geo/series/GSE
```

Use singleGEO to download the single cell high-throughput sequencing data

By providing the GSE IDs, the single cell high-throughput sequencing data deposited in the supplement will be downloaded for further analysis.

```
myGeoID<-c("GSE140032")
##Set DecompressFile=TRUE if you want to download and decompress the data in the same time
DownloadFileInfo<-Get_Geo_Data(DownGSEID=myGeoID,DecompressFile=FALSE)
#> Directory already exists: /home/yyw9094/github/Rpackage/singleGEO/vignettes/GSE140032
DownloadFileInfo
#>       GSE140032
#>  [1,] "filelist.txt"
#>  [2,] "GSE140032_RAW.tar"
#>  [3,] "GSM4151704_GLI1+_cell_PBS_barcodes.tsv"
#>  [4,] "GSM4151704_GLI1+_cell_PBS_barcodes.tsv.gz"
#>  [5,] "GSM4151704_GLI1+_cell_PBS_genes.tsv"
#>  [6,] "GSM4151704_GLI1+_cell_PBS_genes.tsv.gz"
#>  [7,] "GSM4151704_GLI1+_cell_PBS_matrix.mtx"
#>  [8,] "GSM4151704_GLI1+_cell_PBS_matrix.mtx.gz"
#>  [9,] "GSM4151705_GLI1+_cell_Bleo_barcodes.tsv"
#> [10,] "GSM4151705_GLI1+_cell_Bleo_barcodes.tsv.gz"
#> [11,] "GSM4151705_GLI1+_cell_Bleo_genes.tsv"
#> [12,] "GSM4151705_GLI1+_cell_Bleo_genes.tsv.gz"
#> [13,] "GSM4151705_GLI1+_cell_Bleo_matrix.mtx"
#> [14,] "GSM4151705_GLI1+_cell_Bleo_matrix.mtx.gz"
ExtractFile(ExtractID=myGeoID,DecompressType=c("tar","gz"))
list.files("GSE140032")
#>  [1] "filelist.txt"
#>  [2] "GSE140032_RAW.tar"
#>  [3] "GSM4151704_GLI1+_cell_PBS_barcodes.tsv"
#>  [4] "GSM4151704_GLI1+_cell_PBS_barcodes.tsv.gz"
```

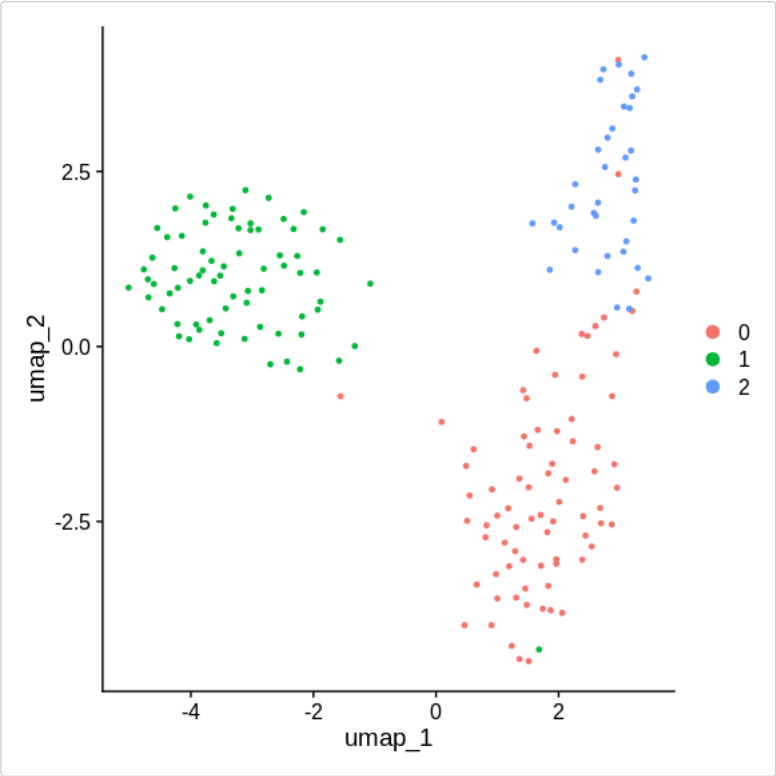
```
#>  [5] "GSM4151704_GLI1+_cell_PBS_genes.tsv"
#>  [6] "GSM4151704_GLI1+_cell_PBS_genes.tsv.gz"
#>  [7] "GSM4151704_GLI1+_cell_PBS_matrix.mtx"
#>  [8] "GSM4151704_GLI1+_cell_PBS_matrix.mtx.gz"
#>  [9] "GSM4151705_GLI1+_cell_Bleo_barcodes.tsv"
#> [10] "GSM4151705_GLI1+_cell_Bleo_barcodes.tsv.gz"
#> [11] "GSM4151705_GLI1+_cell_Bleo_genes.tsv"
#> [12] "GSM4151705_GLI1+_cell_Bleo_genes.tsv.gz"
#> [13] "GSM4151705_GLI1+_cell_Bleo_matrix.mtx"
#> [14] "GSM4151705_GLI1+_cell_Bleo_matrix.mtx.gz"
```

Different groups have different preference in storing and uploading the data to GEO. If a Seurat object is provided by the submitter, the data can be read into Seurat directly for further analysis. For hdf5 format file, R packages of “rhdf5” and/or “SeuratDisk” could be helpful for data access. For the study with only raw count data and/or normalized data provided, SingleGEO provides “MakeSeuObj_FromRawRNAData” to quickly make the Seurat object.

Make Seurat object from raw sequencing data

SingleGEO can make the Seurat object from the raw data it downloaded by running “MakeSeuObj_FromRawRNAData”. Before running “MakeSeuObj_FromRawRNAData”, a list of raw data should be constructed. Each element of the list is for each individual sample and the name of the element corresponds to the name of the sample. “MakeSeuObj_FromRawRNAData” performs the scRNAseq data quality control by filtering out cells with potential doublets, dead cells or empty droplets. If all cells want to be kept or the cells have been filtered, set a big number of maxFeature,maxCount,maxMT and small number of minFeature,minCount. SingleGEO also conducts dimension reduction and makes the Seurat object for the downstream analysis.

```
###The following code generate an example dataset from GSE134174 (The same data has been built
        and can be loaded by data(testData_GSE134174)). Note that only 4 samples with 800
        cells were selected for illustration purpose.
# Get_Geo_Data(DownGSEID="GSE134174",DecompressFile="TRUE",DecompressType="All")
# selPatient<-c("T101","T85","T153","T164")
# raw_GSE134174<-read.delim("./GSE134174/GSE134174_Processed_invivo_raw.txt",header=T,sep="\t")
# meta_GSE134174<-
        read.delim("./GSE134174/GSE134174_Processed_invivo_metadata.txt",header=T,sep="\t")
# meta_sel0<-meta_GSE134174[meta_GSE134174$Donor %in% selPatient,]
# raw_sel0<-raw_GSE134174[,colnames(raw_GSE134174) %in% meta_sel0$Cell]
# set.seed(12345);sampleIndex<-sample(1:nrow(meta_sel0),size=800)
# meta_sel<-meta_sel0[sampleIndex,];raw_sel<-raw_sel0[,paste(meta_sel$Cell)]
# raw_sel<-raw_sel[rowSums(raw_sel)>0,]
# testData_GSE134174<-list(TwoRawData=raw_sel,TwoMetaData=meta_sel)

data(testData_GSE134174)
test_meta<-testData_GSE134174$TwoMetaData
test_data<-testData_GSE134174$TwoRawData
list_GSE134174<-Splitdata_MakeDataList(InputData=test_data,Group=test_meta$Donor)
##The following code setting a big number of maxFeature,maxCount,maxMT and small number ##of
        minFeature,minCount will keep all the cells
seu_GSE134174<-MakeSeuObj_FromRawRNAData(RawList=list_GSE134174,
                                         GSE.ID="GSE134174",
                                         MinFeature=1,
                                         MaxFeature=750000,
                                         MinCount=1,
                                         MaxCount=4000000,
                                         MaxMT=100)
names(seu_GSE134174)
#> [1] "T164" "T101" "T85"  "T153"
Seurat::DimPlot(seu_GSE134174[[1]],repel = TRUE)
```

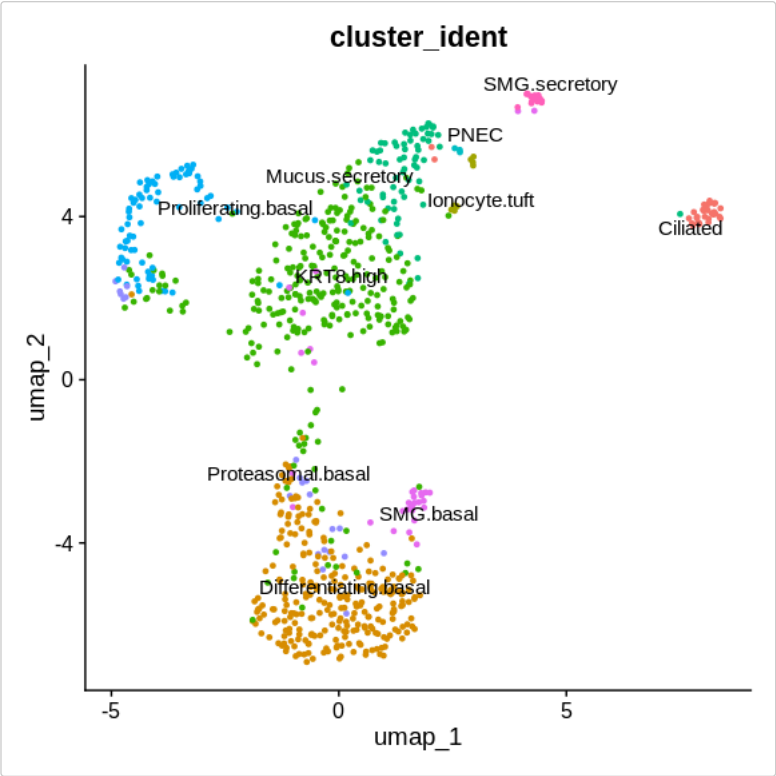



```
####Use sctransform normalization and make Seurat object
seu_GSE134174_sct<-MakeSeuObj_FromRawRNAData(RawList=list_GSE134174,
                                             GSE.ID="GSE134174",
                                             MinFeature=1,
                                             MaxFeature=750000,
                                             MinCount=1,
                                             MaxCount=4000000,
                                             MaxMT=100,
                                             Norm.method = "sct")
names(seu_GSE134174_sct)
#> [1] "T164" "T101" "T85"  "T153"
```

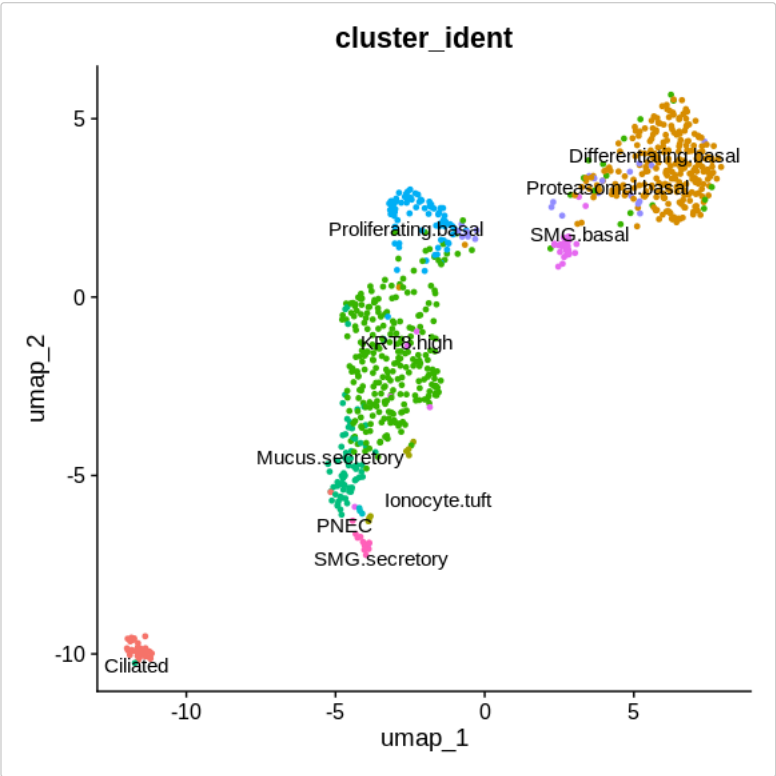
Data integration

Multiple datasets can be integrated for the analysis. These datasets could be obtained from one single GSE ID or multiple GSE IDs. The following examples illustrate the integrative analysis of example datasets from one single GSE ID (GSE134174). For datasets from different studies, please double-check the potential technical discrepancy. For example, different studies can use different genomic build which will lead to gene symbol difference.

```
Int_GSE134174<-SeuObj_integration(Object.list=seu_GSE134174)
###Add additional meta data
row.names(test_meta)<-test_meta$Cell
test_meta<-test_meta[row.names(Int_GSE134174@meta.data),]
Int_GSE134174<-Seurat::AddMetaData(object=Int_GSE134174,metadata =as.data.frame(test_meta))
Seurat::DimPlot(Int_GSE134174,group.by="cluster_ident",label=TRUE,repel = TRUE)+ NoLegend()
```




```
#####Data integration with one single dataset normalized with sctransform method
Int_GSE134174_sct<-SeuObj_integration(Object.list=seu_GSE134174_sct,Frow.which.Norm="sct")
#> [1] 1
#> [1] 2
#> [1] 3
#> [1] 4
test_meta_sct<-test_meta[row.names(Int_GSE134174_sct@meta.data),]
Int_GSE134174_sct<-Seurat::AddMetaData(object=Int_GSE134174_sct,metadata
        =as.data.frame(test_meta_sct))
Seurat::DimPlot(Int_GSE134174_sct,group.by="cluster_ident",label=TRUE,repel = TRUE)+ NoLegend()
```



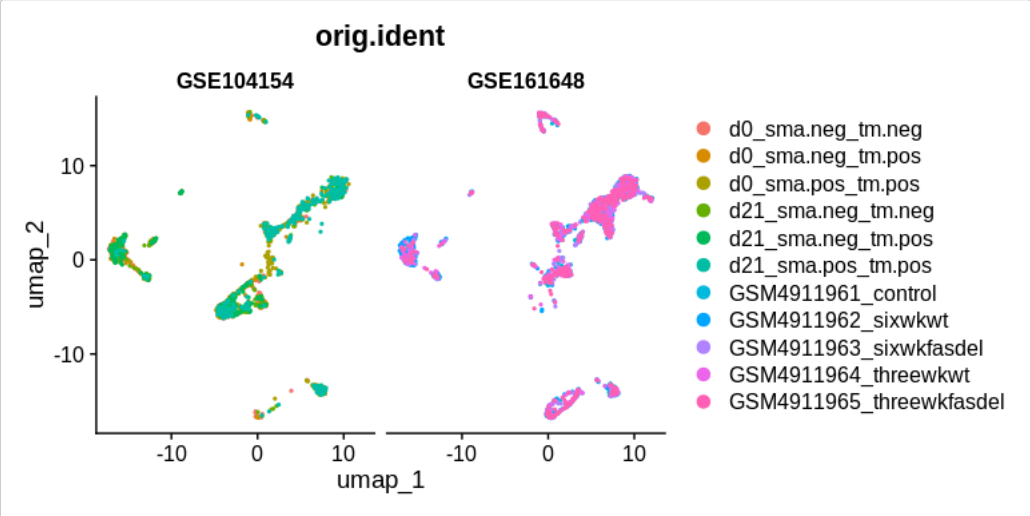
Integrative analysis of datasets from multiple GSE IDs could be powerful to gain additional information. As previous query result, GSE104154 dataset studies the mouse pulmonary fibrotic mesenchymal cells and the

authors identify a new subtype of cells (Pdgfrb_hi) in fibrotic stage(Xie,T. (2018)). Another dataset, GSE161648, studies the impact of homeostatic fibrosis when the fas signaling was lost in fibroblasts(Redente, EF. (2020)). Integrative analysis of these two scRNAseq datasets could be useful to help us understand how the new subtype of cells found from GSE104154 will be affected by fas signaling. For demonstration purpose, we assume that these two datasets using the same genomic build and having the same gene symbol for the same gene.

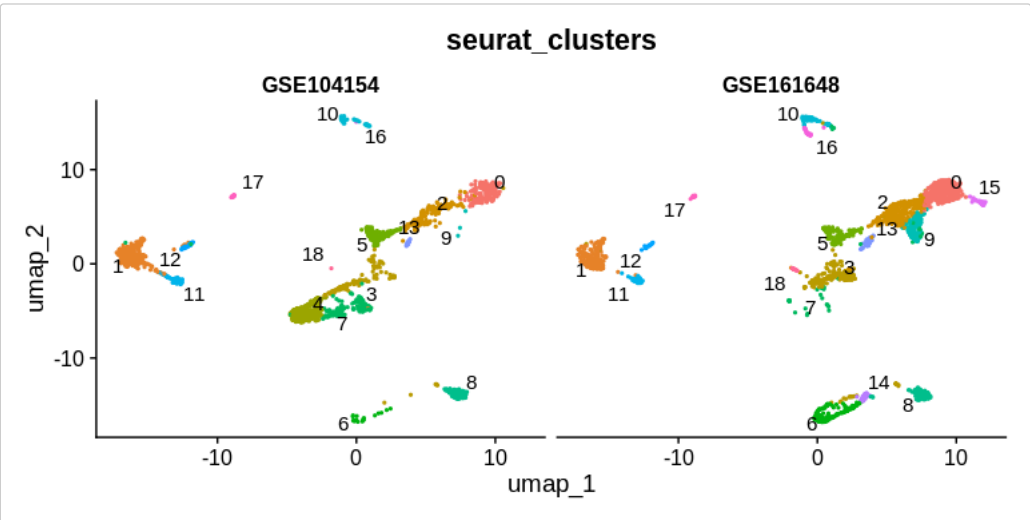
```
# ##The codes were used to generate the example data list of GSE104154 (only a small subset of
        the data were used)
#   myGeoList<-c("GSE104154", "GSE161648")
#   myAllF<-Get_Geo_Data(DownGSEID=myGeoList,DecompressFile=TRUE,DecompressType="All")
#   dat_104_raw<-read.csv("GSE104154/GSE104154_d0_d21_sma_tm_Expr_raw.csv")
#   meta_104_2<-
        as.data.frame(readxl::read_xlsx("GSE104154/GSE104154_cell_type_annotation_d0_d21.xlsx",col_names=T,sheet=2))
#   meta_104_3<-
        as.data.frame(readxl::read_xlsx("GSE104154/GSE104154_cell_type_annotation_d0_d21.xlsx",col_names=T,sheet=3))
#   meta_104_2_3<-rbind(meta_104_2,meta_104_3)
#   meta_104_2_3$ID<-sapply(strsplit(meta_104_2_3$Barcode,split="\\."),function(x) x[2])
#   sleSize=300
#   sampleInd<-unlist(lapply(unique(meta_104_2_3$ID),function(x) {temp<-
        meta_104_2_3[meta_104_2_3$ID %in% x,]
#     set.seed(1234);sleSize<-min(nrow(temp),sleSize);SelectCellInd<-
        sample(1:nrow(temp),size=sleSize,replace=F)
#     returnBarcode<-temp$Barcode[SelectCellInd]}))
#   meta_23<-meta_104_2_3[meta_104_2_3$Barcode %in% sampleInd,]
#   dat_104Raw_withAno<-dat_104_raw[,paste(meta_23$Barcode)]
#   rowSum_dat104Raw<-rowSums(dat_104Raw_withAno)
#   dat_104Raw_f<-dat_104Raw_withAno[rowSum_dat104Raw>0,]
#   removeDupGene<-(!duplicated(dat_104_raw[rowSum_dat104Raw>0,2]))
#   dat_104Raw_f2<-dat_104Raw_f[removeDupGene,]
#   row.names(dat_104Raw_f2)<-dat_104_raw[rowSum_dat104Raw>0,2][removeDupGene]
#   t_datRaw<-
        data.frame(barcode=sapply(strsplit(colnames(dat_104Raw_f2),split="\\."),function(x)
        x[2]),t(dat_104Raw_f2))
#   t_datRaw$barcode<-
        dplyr::recode(t_datRaw$barcode,"1"="d0_sma.pos_tm.pos","2"="d0_sma.neg_tm.pos",
#   "3"="d0_sma.neg_tm.neg",
        "4"="d21_sma.pos_tm.pos","5"="d21_sma.neg_tm.pos","6"="d21_sma.neg_tm.neg")
#   RawDataList_GSE104154<-
        Splitdata_MakeDataList(InputData=t(t_datRaw[,2:ncol(t_datRaw)]),Group=t_datRaw$barcode)
#   meta_GSE104154<-meta_23
#
# ##The codes were used to generate the example data list of GSE161648
#   GEO_ID<-"GSE161648"
#   list_barcodefile<-list.files(GEO_ID,pattern="*_barcodes.tsv.gz")
#   (sampleName<-gsub("_barcodes.tsv.gz","",list_barcodefile))
#   sapply(sampleName,function(x){
#     NewSampleFolder<-sprintf("%s/%s",GEO_ID,x)
#     if (!file.exists(NewSampleFolder)){dir.create(NewSampleFolder)}
#     file.copy(sprintf("%s/%s_barcodes.tsv.gz",GEO_ID,x),
        sprintf("%s/barcodes.tsv.gz",NewSampleFolder))
#     file.copy(sprintf("%s/%s_matrix.mtx.gz",GEO_ID,x),
        sprintf("%s/matrix.mtx.gz",NewSampleFolder))
#     file.copy(sprintf("%s/%s_features.tsv.gz",GEO_ID,x),
        sprintf("%s/features.tsv.gz",NewSampleFolder))
#   })
#   GSE161648_10xRawData<-sapply(sampleName,function(x){rawdata<- Seurat::Read10X(data.dir
        =sprintf("%s/%s",GEO_ID,x))})
#   sleSize1=500
#   RawDataList_GSE161648<-lapply(GSE161648_10xRawData,function(x){sleSize1<-
        min(ncol(as.matrix(x)),sleSize1)
#     set.seed(1234);SelectCellInd<-sample(1:ncol(as.matrix(x)),size=sleSize1,replace=F)
#     returnCell<-(as.matrix(x))[,SelectCellInd]})
```

```
##load the example data
data(Dat_GSE54_GSE48)
RawDataList_GSE104154<-Dat_GSE54_GSE48$dat_GSE104154$RawDataList
meta_GSE104154<-Dat_GSE54_GSE48$dat_GSE104154$MetaData
row.names(meta_GSE104154)<-meta_GSE104154$Barcode
RawDataList_GSE161648<-Dat_GSE54_GSE48$dat_GSE161648$RawDataList
###make Seurat object fro the raw data
seu_GSE104154_raw<-MakeSeuObj_FromRawRNAData(RawList=RawDataList_GSE104154,
                                             GSE.ID="GSE104154",
                                             MtPattern='^mt-',
                                             MinFeature=1,
                                             MaxFeature=750000,
                                             MinCount=1,
                                             MaxCount=4000000,
                                             MaxMT=100)
seu_GSE161648_raw<-
      MakeSeuObj_FromRawRNAData(RawList=RawDataList_GSE161648,GSE.ID="GSE161648",MtPattern='^mt-
      ')
###Data integration
```

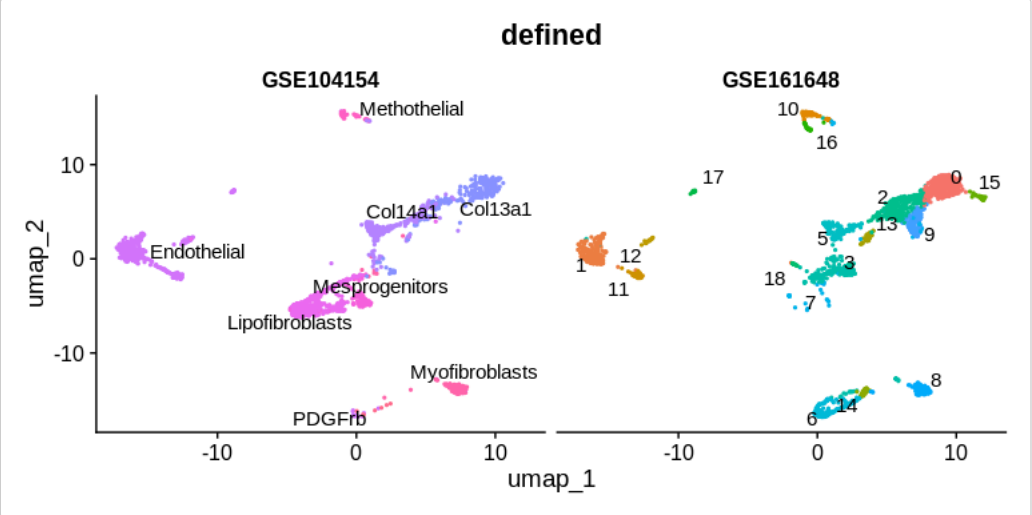
```
Int_GSE54_GSE48<-
      SeuObj_integration(Object.list=seu_GSE104154_raw,Object.list2=seu_GSE161648_raw)

###Evaluate the potential batch effect
Seurat::DimPlot(Int_GSE54_GSE48,group.by="orig.ident",split.by="GSE",repel = TRUE)
```



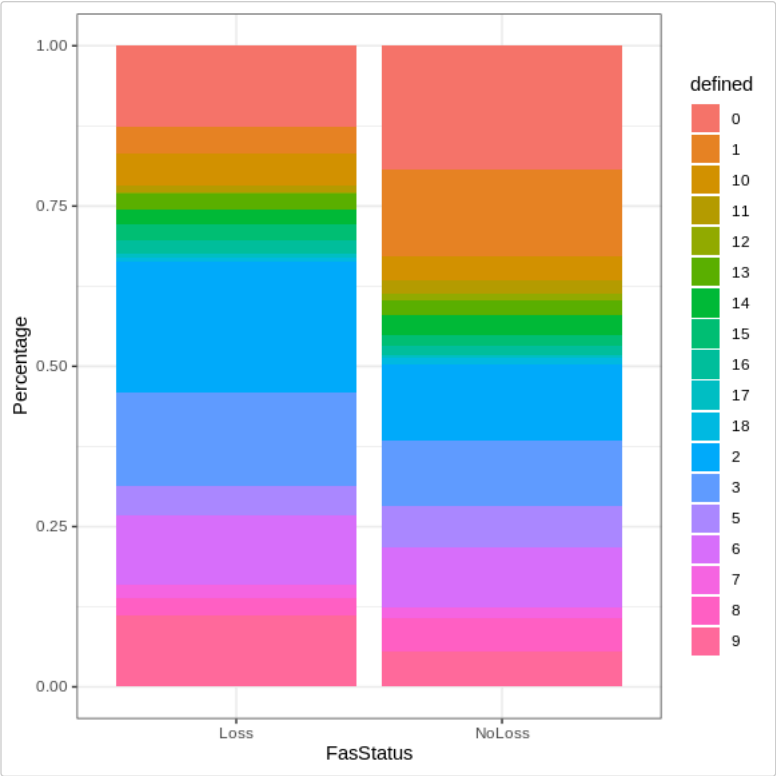

```
Seurat::DimPlot(Int_GSE54_GSE48,group.by="seurat_clusters",split.by="GSE",label=TRUE,repel =
      TRUE)+ NoLegend()
```




```
###Add annotation to GSE104154 as provided by the data submitter
Int_GSE54_GSE48$rowname<-row.names(Int_GSE54_GSE48@meta.data)
meta_subsetOfInt_GSE54<-subset(Int_GSE54_GSE48,GSE=="GSE104154")@meta.data
meta_subsetOfInt_GSE54$Barcode<-
      sapply(strsplit(meta_subsetOfInt_GSE54$rowname,split="\\_"),function(x) x[1])
ano_meta_subsetOfInt_GSE54<-
      dplyr::left_join(meta_subsetOfInt_GSE54,meta_GSE104154,by="Barcode")
row.names(ano_meta_subsetOfInt_GSE54)<-ano_meta_subsetOfInt_GSE54$rowname
ano_meta_subsetOfInt_GSE54<-ano_meta_subsetOfInt_GSE54[,c("rowname","defined")]
Int_GSE54_GSE48@meta.data<-
      dplyr::left_join(Int_GSE54_GSE48@meta.data,ano_meta_subsetOfInt_GSE54,by="rowname")
row.names(Int_GSE54_GSE48@meta.data)<-Int_GSE54_GSE48@meta.data$rowname
Int_GSE54_GSE48$defined<-ifelse(is.na(Int_GSE54_GSE48$defined),

      paste(Int_GSE54_GSE48$seurat_clusters),Int_GSE54_GSE48$defined)

Seurat::DimPlot(Int_GSE54_GSE48,group.by="defined",split.by="GSE",label=TRUE,repel = TRUE)+
      NoLegend()
```



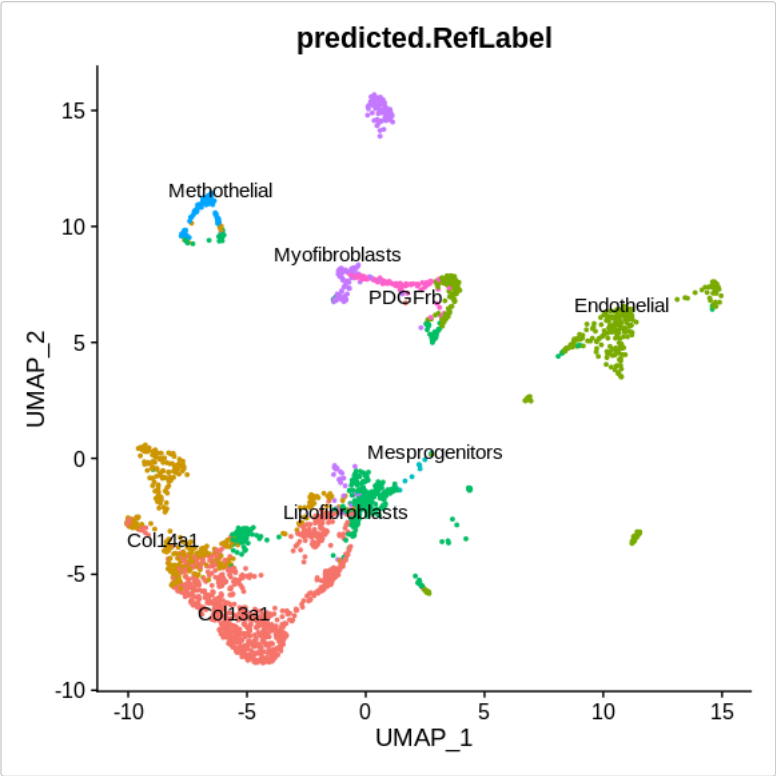

```
  ###Test whether loss Fas signaling will affect PDGFrb cell fraction in GSE161648 dataset
  Subset_GSE161648FromInt<-subset(Int_GSE54_GSE48,GSE=="GSE161648")
  Subset_GSE161648FromInt$FasStatus<-ifelse(Subset_GSE161648FromInt$ID %in%
        c("GSM4911963_sixwkfasdel","GSM4911965_threewkfasdel"),"Loss","NoLoss")
  tab_dat<-
        as.matrix(as.data.frame(unclass(table(Subset_GSE161648FromInt@meta.data[,c("FasStatus","defined")]))))
  ###PDGFrb is located in the cluster of 6. No obvious different is observed and loss of Fas
        signaling probably won't affect PDGFrb cell fraction.
  (tab_pop<-prop.table(tab_dat, margin = 1))
#>                0          1         10         11          12         13
#> Loss   0.1272727 0.04155844 0.04935065 0.01038961 0.002597403 0.02337662
#> NoLoss 0.1929348 0.13496377 0.03804348 0.02083333 0.009963768 0.02264493
#>               14         15         16          17          18          2
#> Loss   0.02337662 0.02467532 0.02207792 0.006493506 0.006493506 0.2038961
#> NoLoss 0.03170290 0.01630435 0.01449275 0.005434783 0.009057971 0.1186594
#>                3          5         6          7          8          9
#> Loss   0.1441558 0.04675325 0.1077922 0.02077922 0.02857143 0.11038961
#> NoLoss 0.1032609 0.06431159 0.0942029 0.01630435 0.05253623 0.05434783
  dat_barplot<-data.frame(FasStatus=rep(c("Loss","NoLoss"),ncol(tab_pop)),
                          defined=rep(colnames(tab_pop),each=2),
                          Percentage=as.numeric(tab_pop))


  ###Barplot. Note that PDGFrb is located in the cluster of 6.
  pl<-ggplot2::ggplot(dat_barplot, aes(fill=defined, y=Percentage, x=FasStatus))+
    geom_bar(position="stack", stat="identity") +
    theme_bw()
  print(pl)
```



Transferring learning

Instead of the data integration by modifying the query expression data, transferring learning could be an alternative approach to study the multiple datasets from different GSE ids. The following example takes GSE104154 as the reference while GSE161648 as the query.

```
 data(Data_Ref_transfer)
 data(Data_Query_transfer)

Transfer_Result<-SeuObj_transfer(RefObj=Data_Ref_transfer,
                                 QueryObj=Data_Query_transfer,
                                 RefLabel="defined")
Seurat::DimPlot(Transfer_Result,
                group.by="predicted.RefLabel",
                reduction = "umap",
                label=TRUE,
                repel = TRUE)+ NoLegend()
```



Citations

Xie T, Wang Y, Deng N, Huang G et al. Single-Cell Deconvolution of Fibroblast Heterogeneity in Mouse Pulmonary Fibrosis. Cell Rep 2018 Mar 27;22(13):3625-3640. PMID: 29590628

Redente EF, Chakraborty S, Sajuthi S, Black BP et al. Loss of Fas signaling in fibroblasts impairs homeostatic fibrosis resolution and promotes persistent pulmonary fibrosis. JCI Insight 2020 Dec 8;6(1). PMID: 33290280

SessionInfo()

```
sessionInfo()
#> R version 4.4.0 (2024-04-24)
#> Platform: x86_64-pc-linux-gnu
#> Running under: Red Hat Enterprise Linux Server 7.9 (Maipo)
#>
#> Matrix products: default
#> BLAS:   /hpc/software/spack_v20d1/spack/opt/spack/linux-rhel7-x86_64/gcc-12.3.0/r-4.4.0-
       aaqsqjfnzw5p5c4twtddj4sxqi23pyfw/rlib/R/lib/libRblas.so
#> LAPACK: /hpc/software/spack_v20d1/spack/opt/spack/linux-rhel7-x86_64/gcc-12.3.0/r-4.4.0-
       aaqsqjfnzw5p5c4twtddj4sxqi23pyfw/rlib/R/lib/libRlapack.so;  LAPACK version 3.12.0
#>
#> locale:
#>  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
#>  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
#>  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
#>  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
#>  [9] LC_ADDRESS=C               LC_TELEPHONE=C
#> [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
#>
#> time zone: America/Chicago
#> tzcode source: system (glibc)
#>
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
#>  [1] singleGEO_0.1.1     glmGamPoi_1.16.0    curl_5.2.3
#>  [4] ggplot2_3.5.1       sctransform_0.4.1   knitr_1.48
#>  [7] devtools_2.4.5      usethis_3.0.0       GEOmetadb_1.66.0
#> [10] GEOquery_2.72.0     Biobase_2.64.0      BiocGenerics_0.50.0
#> [13] Seurat_5.1.0        SeuratObject_5.0.2  sp_2.1-4
#> [16] RSQLite_2.3.9       Matrix_1.7-0        R.utils_2.12.3
#> [19] R.oo_1.26.0         R.methodsS3_1.8.2
#>
#> loaded via a namespace (and not attached):
```

```
#>   [1] RcppAnnoy_0.0.22           splines_4.4.0
#>   [3] later_1.3.2                tibble_3.2.1
#>   [5] polyclip_1.10-7            fastDummies_1.7.4
#>   [7] lifecycle_1.0.4            globals_0.16.3
#>   [9] lattice_0.22-6             MASS_7.3-61
#>  [11] magrittr_2.0.3             limma_3.60.6
#>  [13] plotly_4.10.4              sass_0.4.9
#>  [15] rmarkdown_2.28             jquerylib_0.1.4
#>  [17] yaml_2.3.10                remotes_2.5.0
#>  [19] httpuv_1.6.15              spam_2.11-0
#>  [21] sessioninfo_1.2.2          pkgbuild_1.4.4
#>  [23] spatstat.sparse_3.1-0      reticulate_1.39.0
#>  [25] cowplot_1.1.3              pbapply_1.7-2
#>  [27] DBI_1.2.3                  RColorBrewer_1.1-3
#>  [29] zlibbioc_1.50.0            abind_1.4-8
#>  [31] pkgload_1.4.0              GenomicRanges_1.56.1
#>  [33] Rtsne_0.17                 purrr_1.0.2
#>  [35] GenomeInfoDbData_1.2.12    IRanges_2.38.1
#>  [37] S4Vectors_0.42.1           ggrepel_0.9.6
#>  [39] irlba_2.3.5.1              listenv_0.9.1
#>  [41] spatstat.utils_3.1-0       goftest_1.2-3
#>  [43] RSpectra_0.16-2            spatstat.random_3.3-2
#>  [45] fitdistrplus_1.2-1         parallelly_1.38.0
#>  [47] DelayedMatrixStats_1.26.0  DelayedArray_0.30.1
#>  [49] leiden_0.4.3.1             codetools_0.2-20
#>  [51] xml2_1.3.6                 tidyselect_1.2.1
#>  [53] UCSC.utils_1.0.0           farver_2.1.2
#>  [55] matrixStats_1.4.1          stats4_4.4.0
#>  [57] spatstat.explore_3.3-2     jsonlite_1.8.9
#>  [59] ellipsis_0.3.2             progressr_0.14.0
#>  [61] ggridges_0.5.6             survival_3.7-0
#>  [63] tools_4.4.0                ica_1.0-3
#>  [65] Rcpp_1.0.13                glue_1.8.0
#>  [67] SparseArray_1.4.8          gridExtra_2.3
#>  [69] xfun_0.48                  MatrixGenerics_1.16.0
#>  [71] GenomeInfoDb_1.40.1        dplyr_1.1.4
#>  [73] withr_3.0.1                fastmap_1.2.0
#>  [75] fansi_1.0.6                digest_0.6.37
#>  [77] R6_2.5.1                   mime_0.12
#>  [79] colorspace_2.1-1           scattermore_1.2
#>  [81] tensor_1.5                 spatstat.data_3.1-2
#>  [83] utf8_1.2.4                 tidyr_1.3.1
#>  [85] generics_0.1.3             data.table_1.16.0
#>  [87] S4Arrays_1.4.1             httr_1.4.7
#>  [89] htmlwidgets_1.6.4          uwot_0.2.2
#>  [91] pkgconfig_2.0.3            gtable_0.3.5
#>  [93] blob_1.2.4                 lmtest_0.9-40
#>  [95] XVector_0.44.0             htmltools_0.5.8.1
#>  [97] profvis_0.4.0              dotCall64_1.2
#>  [99] scales_1.3.0               png_0.1-8
#> [101] spatstat.univar_3.0-1      rstudioapi_0.16.0
#> [103] tzdb_0.4.0                 reshape2_1.4.4
#> [105] nlme_3.1-166               cachem_1.1.0
#> [107] zoo_1.8-12                 stringr_1.5.1
#> [109] KernSmooth_2.23-24         parallel_4.4.0
#> [111] miniUI_0.1.1.1             pillar_1.9.0
#> [113] grid_4.4.0                 vctrs_0.6.5
#> [115] RANN_2.6.2                 urlchecker_1.0.1
#> [117] promises_1.3.0             xtable_1.8-4
#> [119] cluster_2.1.6              evaluate_1.0.0
#> [121] readr_2.1.5                cli_3.6.3
#> [123] compiler_4.4.0             crayon_1.5.3
#> [125] rlang_1.1.4                future.apply_1.11.2
#> [127] labeling_0.4.3             plyr_1.8.9
#> [129] fs_1.6.4                   stringi_1.8.4
#> [131] viridisLite_0.4.2          deldir_2.0-4
#> [133] munsell_0.5.1              lazyeval_0.2.2
#> [135] spatstat.geom_3.3-3        RcppHNSW_0.6.0
#> [137] hms_1.1.3                  patchwork_1.3.0
#> [139] sparseMatrixStats_1.16.0   bit64_4.5.2
#> [141] future_1.34.0              statmod_1.5.0
#> [143] shiny_1.9.1                highr_0.11
#> [145] SummarizedExperiment_1.34.0 ROCR_1.0-11
#> [147] igraph_2.0.3               memoise_2.0.1
#> [149] bslib_0.8.0                bit_4.5.0.1
```