library(dplyr)

library(Seurat)

library(patchwork)

library(CellChat)

library(ggplot2)

library(ggalluvial)

library(svglite)

library(SeuratData)

options(stringsAsFactors = FALSE)

library(SingleR)

library(celldex)

library(SingleCellExperiment)

library(magrittr)

library(cowplot)

theme\_set(theme\_cowplot())

library(ggrepel)

library(multtest)

library(metap)

library(monocle3)

library(tidyverse)

library(SeuratWrappers)

library(ggridges)

library(pheatmap)

library(RColorBrewer)

library(viridis)

library(monocle)

library(clusterProfiler)

library(org.Hs.eg.db)

library(sctransform)

library(psych)

library(qgraph)

library(igraph)

library(curl)

library(NMF)

library(pcaMethods)

library(circlize)

library(devtools)

library(remotes)

library(usethis)

library(qs)

library(sctransform)

library(harmony)

library(readxl)

library(openxlsx)

library(proxy)

install.packages("harmony")

------4/15GSE128033normalbal1

setwd("C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal")

#normalbal1

normalbal1 <- Read10X(data.dir = "C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033normalbal1")

normalbal1 <- CreateSeuratObject(counts = normalbal1, project = "normalbal1", min.cells = 3, min.features = 200)

normalbal1

normalbal1 <- SCTransform(normalbal1, vst.flavor = "v2", verbose = FALSE)

#normalbal2

normalbal2 <- Read10X(data.dir = "C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033normalbal2")

normalbal2 <- CreateSeuratObject(counts = normalbal2, project = "normalbal2", min.cells = 3, min.features = 200)

normalbal2

normalbal2 <- SCTransform(normalbal2, vst.flavor = "v2", verbose = FALSE)

#normallung1処理

normallung1 <- Read10X(data.dir = "C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033Normal lung")

normallung1 <- CreateSeuratObject(counts = normallung1, project = "normallung1", min.cells = 3, min.features = 200)

normallung1

normallung1 <- SCTransform(normallung1, vst.flavor = "v2", verbose = FALSE)

#normallung2処理

normallung2 <- Read10X(data.dir = "C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE1280331の2Normallung")

normallung2 <- CreateSeuratObject(counts = normallung2, project = "normallung2", min.cells = 3, min.features = 200)

normallung2

normallung2 <- SCTransform(normallung2, vst.flavor = "v2", verbose = FALSE)

#normallung3処理

normallung3 <- Read10X(data.dir = "C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033normallung3")

normallung3 <- CreateSeuratObject(counts = normallung3, project = "normallung3", min.cells = 3, min.features = 200)

normallung3

normallung3 <- SCTransform(normallung3, vst.flavor = "v2", verbose = FALSE)

#normallung4処理

normallung4 <- Read10X(data.dir = "C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033normallung4")

normallung4 <- CreateSeuratObject(counts = normallung4, project = "normallung4", min.cells = 3, min.features = 200)

normallung4

normallung4 <- SCTransform(normallung4, vst.flavor = "v2", verbose = FALSE)

#normallung5処理

normallung5 <- Read10X(data.dir = "C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033normallung5")

normallung5 <- CreateSeuratObject(counts = normallung5, project = "normallung5", min.cells = 3, min.features = 200)

normallung5

normallung5 <- SCTransform(normallung5, vst.flavor = "v2", verbose = FALSE)

#normallung6処理

normallung6 <- Read10X(data.dir = "C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033normallung6")

normallung6 <- CreateSeuratObject(counts = normallung6, project = "normallung6", min.cells = 3, min.features = 200)

normallung6

normallung6 <- SCTransform(normallung6, vst.flavor = "v2", verbose = FALSE)

#normallung7処理

normallung7 <- Read10X(data.dir = "C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033normallung7")

normallung7 <- CreateSeuratObject(counts = normallung7, project = "normallung7", min.cells = 3, min.features = 200)

normallung7

normallung7 <- SCTransform(normallung7, vst.flavor = "v2", verbose = FALSE)

#normallung8処理

normallung8 <- Read10X(data.dir = "C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033normallung8")

normallung8 <- CreateSeuratObject(counts = normallung8, project = "normallung8", min.cells = 3, min.features = 200)

normallung8

normallung8 <- SCTransform(normallung8, vst.flavor = "v2", verbose = FALSE)

setwd("C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033IPFlungまとめ")

#IPFlung1処理

IPFlung1 <- Read10X(data.dir = "C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033IPFlungまとめ\\GSE128033IPFlung1")

IPFlung1 <- CreateSeuratObject(counts = IPFlung1, project = "IPFlung1", min.cells = 3, min.features = 200)

IPFlung1

IPFlung1 <- SCTransform(IPFlung1, vst.flavor = "v2", verbose = FALSE)

#IPFlung2の処理

IPFlung2 <- Read10X(data.dir = "C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033IPFlungまとめ\\GSE128033IPFlung2")

IPFlung2 <- CreateSeuratObject(counts = IPFlung2, project = "IPFlung2", min.cells = 3, min.features = 200)

IPFlung2

IPFlung2 <- SCTransform(IPFlung2, vst.flavor = "v2", verbose = FALSE)

#IPFlung3の処理

IPFlung3 <- Read10X(data.dir = "C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033IPFlungまとめ\\GSE128033IPFlung3")

IPFlung3 <- CreateSeuratObject(counts = IPFlung3, project = "IPFlung3", min.cells = 3, min.features = 200)

IPFlung3

IPFlung3 <- SCTransform(IPFlung3, vst.flavor = "v2", verbose = FALSE)

#IPFlung4の処理

IPFlung4 <- Read10X(data.dir = "C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033IPFlungまとめ\\GSE128033IPFlung4")

IPFlung4 <- CreateSeuratObject(counts = IPFlung4, project = "IPFlung4", min.cells = 3, min.features = 200)

IPFlung4

IPFlung4 <- SCTransform(IPFlung4, vst.flavor = "v2", verbose = FALSE)

#IPFlung5の処理

IPFlung5 <- Read10X(data.dir = "C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033IPFlungまとめ\\GSE128033IPFlung5")

IPFlung5 <- CreateSeuratObject(counts = IPFlung5, project = "IPFlung5", min.cells = 3, min.features = 200)

IPFlung5

IPFlung5 <- SCTransform(IPFlung5, vst.flavor = "v2", verbose = FALSE)

#IPFlung6の処理

IPFlung6 <- Read10X(data.dir = "C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033IPFlungまとめ\\GSE128033IPFlung6")

IPFlung6 <- CreateSeuratObject(counts = IPFlung6, project = "IPFlung6", min.cells = 3, min.features = 200)

IPFlung6

IPFlung6 <- SCTransform(IPFlung6, vst.flavor = "v2", verbose = FALSE)

#IPFlung7の処理

IPFlung7 <- Read10X(data.dir = "C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033IPFlungまとめ\\GSE128033IPFlung7")

IPFlung7 <- CreateSeuratObject(counts = IPFlung7, project = "IPFlung7", min.cells = 3, min.features = 200)

IPFlung7

IPFlung7 <- SCTransform(IPFlung7, vst.flavor = "v2", verbose = FALSE)

#IPFlung2の処理

IPFlung8 <- Read10X(data.dir = "C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033IPFlungまとめ\\GSE128033IPFlung8")

IPFlung8 <- CreateSeuratObject(counts = IPFlung8, project = "IPFlung8", min.cells = 3, min.features = 200)

IPFlung8

IPFlung8 <- SCTransform(IPFlung8, vst.flavor = "v2", verbose = FALSE)

# normallung1 と normallung3 に sample\_type を追加

normalbal1$sample\_type <- "normallung"

normalbal2$sample\_type <- "normallung"

normallung1$sample\_type <- "normallung"

normallung2$sample\_type <- "normallung"

normallung3$sample\_type <- "normallung"

normallung4$sample\_type <- "normallung"

normallung5$sample\_type <- "normallung"

normallung6$sample\_type <- "normallung"

normallung7$sample\_type <- "normallung"

normallung8$sample\_type <- "normallung"

# IPFlung1 と IPFlung3 に sample\_type を追加

IPFlung1$sample\_type <- "IPFlung"

IPFlung2$sample\_type <- "IPFlung"

IPFlung3$sample\_type <- "IPFlung"

IPFlung4$sample\_type <- "IPFlung"

IPFlung5$sample\_type <- "IPFlung"

IPFlung6$sample\_type <- "IPFlung"

IPFlung7$sample\_type <- "IPFlung"

IPFlung8$sample\_type <- "IPFlung"

# 念のために、これらの変更が正しく行われたかを確認

head(normallung1@meta.data)

head(IPFlung1@meta.data)

# 統合のための特徴量を選択

# SCTransformを用いた統合準備

# アンカーを探す

# データを統合

# 統合データの結果を確認

features\_to\_integrate <- SelectIntegrationFeatures(object.list = list(normalbal1, normalbal2, normallung1, normallung2, normallung3, normallung4, normallung5, normallung6, normallung7, normallung8, IPFlung1, IPFlung2, IPFlung3, IPFlung4, IPFlung5, IPFlung6, IPFlung7, IPFlung8), nfeatures = 3000)

data\_list <- PrepSCTIntegration(object.list = list(normalbal1, normalbal2, normallung1, normallung2, normallung3, normallung4, normallung5, normallung6, normallung7, normallung8, IPFlung1, IPFlung2, IPFlung3, IPFlung4, IPFlung5, IPFlung6, IPFlung7, IPFlung8), anchor.features = features\_to\_integrate)

anchors <- FindIntegrationAnchors(object.list = data\_list, normalization.method = "SCT", anchor.features = features\_to\_integrate)

integrated\_data <- IntegrateData(anchorset = anchors, normalization.method = "SCT")

integrated\_data

# データをスケーリング

# PCA を実行

# UMAP を実行

# UMAP プロットを表示

integrated\_data <- ScaleData(integrated\_data, verbose = FALSE)

integrated\_data <- RunPCA(integrated\_data, verbose = FALSE)

integrated\_data <- RunUMAP(integrated\_data, dims = 1:30, verbose = FALSE)

DimPlot(integrated\_data, reduction = "umap")

DimPlot(integrated\_data, reduction = "umap", split.by = "sample\_type", label =TRUE)

FeaturePlot(integrated\_data, features = c("LRP1"), split.by = "sample\_type", max.cutoff = 3, cols = c("grey", "red"))

VlnPlot(integrated\_data, features = "LRP1")

FeaturePlot(integrated\_data, features = c("SFTPB", "COL1A1", "COL1A2"))

# どのアッセイにLRP1が存在するか確認

if("LRP1" %in% rownames(integrated\_data[["SCT"]]@data)) {

print("LRP1 is in SCT assay")

}

# SCT アッセイをデフォルトに設定

DefaultAssay(integrated\_data) <- "SCT"

# デフォルトのアッセイを使用してFeaturePlotを実行

FeaturePlot(integrated\_data, features = "LRP1", slot = "data", max.cutoff = 3, cols = c("grey", "red"))

#SingleRアノテーション

ref <- celldex::HumanPrimaryCellAtlasData()

results <- SingleR(test = as.SingleCellExperiment(integrated\_data), ref = ref, labels = ref$label.main)

integrated\_data$singlr\_labels <- results$labels

DimPlot(integrated\_data, reduction = 'umap', label.size = 2.5, pt.size = 0.1, group.by = 'singlr\_labels', label = TRUE )

head(integrated\_data@meta.data)

str(integrated\_data@meta.data)

# Fibroblastsのみ取り出し再解析

Idents(integrated\_data) <- integrated\_data@meta.data$singlr\_labels

fibroblasts\_data <- subset(integrated\_data, idents = "Fibroblasts")

fibroblasts\_data <- SCTransform(fibroblasts\_data, verbose = FALSE)

fibroblasts\_data <- RunPCA(fibroblasts\_data, verbose = FALSE)

fibroblasts\_data <- RunUMAP(fibroblasts\_data, reduction = "pca", dims = 1:30, verbose = FALSE)

fibroblasts\_data <- FindNeighbors(fibroblasts\_data, reduction = "pca", dims = 1:30, verbose = FALSE)

fibroblasts\_data <- FindClusters(fibroblasts\_data, resolution = 0.5, verbose = FALSE)

fibroblasts\_data <- RenameIdents(fibroblasts\_data, .old = names(Idents(fibroblasts\_data)), .new = paste("Cluster", names(Idents(fibroblasts\_data)), sep = ""))

fibroblasts\_data$seurat\_clusters <- Idents(fibroblasts\_data)

head(fibroblasts\_data@meta.data)

str(fibroblasts\_data@meta.data)

# UMAPプロットの描画

DimPlot(fibroblasts\_data, reduction = "umap", label = TRUE) + NoLegend()

DimPlot(fibroblasts\_data, reduction = "umap", split.by = "sample\_type", label =TRUE)

FeaturePlot(fibroblasts\_data, features = "LRP1", slot = "data", max.cutoff = 3, cols = c("grey", "red"))

FeaturePlot(fibroblasts\_data, features = c("LRP1"), split.by = "sample\_type", max.cutoff = 3, cols = c("grey", "red"))

features <- c("LRP1", "ACTA2", "COL1A1")

# DotPlotを作成する

DotPlot(

object = fibroblasts\_data, # Seuratオブジェクトを指定

features = features, # DotPlotを作成する遺伝子のリストを指定

group.by = "sample\_type", # 使用する細胞腫の情報を指定

dot.scale = 20,

cols = c("lightgray", "darkgreen")

) + RotatedAxis()

# DotPlotの実行

DotPlot(

object = fibroblasts\_data, # Seuratオブジェクトを指定

features = features, # DotPlotを作成する遺伝子のリストを指定

group.by = "seurat\_clusters", # クラスタ情報でグループ化

split.by = "sample\_type", # サンプルタイプで区別（IPFlung, normallung）

dot.scale = 20, # ドットのサイズスケール

cols = c("lightgray", "darkgreen") # ドットの色指定

) + RotatedAxis()

# IPFlungのみを含むサブセットを作成

IPFlung\_data <- subset(fibroblasts\_data, subset = sample\_type == "IPFlung")

# DotPlotを作成

DotPlot(

object = IPFlung\_data, # IPFlungのみのSeuratオブジェクトを指定

features = features, # DotPlotを作成する遺伝子のリストを指定

group.by = "seurat\_clusters", # クラスタ情報でグループ化

dot.scale = 20,

cols = c("lightgray", "darkgreen")

) + RotatedAxis()

-----4/19新しい方法での解析

setwd("C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal")

process\_sample <- function(data\_dir, project\_name) {

data <- Read10X(data.dir = data\_dir)

seurat\_obj <- CreateSeuratObject(counts = data, project = project\_name, min.cells = 3, min.features = 200)

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

seurat\_obj <- subset(seurat\_obj, subset = nFeature\_RNA > 200 & nFeature\_RNA < 2500 & percent.mt < 5)

seurat\_obj <- SCTransform(seurat\_obj, vst.flavor = "v2", verbose = FALSE)

return(seurat\_obj)

}

# すべてのデータセットのリストを作成

dataset\_paths <- c(

"C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033normalbal1",

"C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033normalbal2",

"C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033Normal lung",

"C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE1280331の2Normallung",

"C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033normallung3",

"C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033normallung4",

"C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033normallung5",

"C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033normallung6",

"C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033normallung7",

"C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033normallung8",

"C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033IPFlung1",

"C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033IPFlung2",

"C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033IPFlung3",

"C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033IPFlung4",

"C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033IPFlung5",

"C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033IPFlung6",

"C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033IPFlung7",

"C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033IPFlung8"

)

dataset\_names <- c(

"normalbal1", "normalbal2",

"normallung1", "normallung2", "normallung3", "normallung4", "normallung5",

"normallung6", "normallung7", "normallung8",

"IPFlung1", "IPFlung2", "IPFlung3", "IPFlung4",

"IPFlung5", "IPFlung6", "IPFlung7", "IPFlung8"

)

# mapplyを使用して各データセットを処理

datasets <- mapply(process\_sample, data\_dir = dataset\_paths, project\_name = dataset\_names, SIMPLIFY = FALSE)

datasets <- setNames(datasets, dataset\_names)

# normallung と IPFlung でデータを分ける

normallung <- datasets[1:10]

IPFlung <- datasets[11:18]

# グループごとにデータ統合

normallung\_integrated <- Seurat::IntegrateData(anchorset = FindIntegrationAnchors(object.list = normallung))

IPFlung\_integrated <- Seurat::IntegrateData(anchorset = FindIntegrationAnchors(object.list = IPFlung))

features\_to\_integrate <- SelectIntegrationFeatures(object.list = list(normallung\_integrated, IPFlung\_integrated), nfeatures = 3000)

data\_list <- PrepSCTIntegration(object.list = list(normallung\_integrated, IPFlung\_integrated), anchor.features = features\_to\_integrate)

anchors <- FindIntegrationAnchors(object.list = data\_list, normalization.method = "SCT", anchor.features = features\_to\_integrate)

integrated\_data <- IntegrateData(anchorset = anchors, normalization.method = "SCT")

integrated\_data

# 各Seuratオブジェクトの細胞名に接頭辞を追加して一意にします

normallung\_integrated <- RenameCells(normallung\_integrated, add.cell.id = "normallung")

IPFlung\_integrated <- RenameCells(IPFlung\_integrated, add.cell.id = "IPFlung")

integration\_features <- SelectIntegrationFeatures(object.list = list(normallung\_integrated, IPFlung\_integrated))

# 特徴選択

integration\_features <- SelectIntegrationFeatures(object.list = list(normallung\_integrated, IPFlung\_integrated))

# 統合の準備

list(normallung\_integrated, IPFlung\_integrated) <- PrepSCTIntegration(

object.list = list(normallung\_integrated, IPFlung\_integrated),

anchor.features = integration\_features

)

# dimnamesの確認と修正

if (is.null(dimnames(normallung\_integrated@assays$SCT@scale.data))) {

rownames(normallung\_integrated@assays$SCT@scale.data) <- rownames(normallung\_integrated@assays$SCT@counts)

colnames(normallung\_integrated@assays$SCT@scale.data) <- colnames(normallung\_integrated@assays$SCT@counts)

}

# 統合特徴を選択

integration\_features <- SelectIntegrationFeatures(object.list = list(normallung\_integrated, IPFlung\_integrated))

# 適切に dimnames を設定

rownames(normallung\_integrated@assays$SCT@scale.data) <- rownames(normallung\_integrated@assays$SCT@counts)

colnames(normallung\_integrated@assays$SCT@scale.data) <- colnames(normallung\_integrated)

# 行と列の数を確認

print(dim(normallung\_integrated@assays$SCT@counts))

print(dim(normallung\_integrated@assays$SCT@scale.data))

# 選択された特徴が scale.data にあるかどうかを確認

valid\_features <- features %in% rownames(normallung\_integrated@assays$SCT@scale.data)

print(sum(valid\_features)) # 有効な特徴の数を表示

if (sum(valid\_features) > 0) {

valid\_features <- features[valid\_features]

all\_integrated <- PrepSCTIntegration(

object.list = list(normallung\_integrated, IPFlung\_integrated),

anchor.features = valid\_features

)

} else {

print("No valid integration features found in scale.data.")

}

normallung\_integrated <- ScaleData(normallung\_integrated, features = rownames(normallung\_integrated@assays$SCT@counts))

IPFlung\_integrated <- ScaleData(IPFlung\_integrated, features = rownames(IPFlung\_integrated@assays$SCT@counts))

# 特徴選択

features\_to\_integrate <- SelectIntegrationFeatures(object.list = list(normallung\_integrated, IPFlung\_integrated), nfeatures = 3000)

# 統合の準備を実行し、結果を更新

integrated\_results <- PrepSCTIntegration(

object.list = list(normallung\_integrated, IPFlung\_integrated),

anchor.features = features\_to\_integrate

)

# 結果をそれぞれのオブジェクトに割り当てる

normallung\_integrated <- integrated\_results[[1]]

IPFlung\_integrated <- integrated\_results[[2]]

# アンカーの検出

final\_anchors <- FindIntegrationAnchors(

object.list = list(normallung\_integrated, IPFlung\_integrated),

normalization.method = "SCT",

anchor.features = features\_to\_integrate

)

# データの統合

final\_integrated <- IntegrateData(anchorset = final\_anchors)

# データのスケーリング

if ("scale.data" %in% names(final\_integrated@assays$RNA)) {

print("Scale data exists.")

} else {

print("Scale data does not exist, performing scaling.")

final\_integrated <- ScaleData(final\_integrated, features = rownames(final\_integrated@assays$RNA@counts))

}

# 変動遺伝子の選択

final\_integrated <- FindVariableFeatures(final\_integrated, selection.method = "vst", nfeatures = 2000)

final\_integrated <- ScaleData(final\_integrated, features = VariableFeatures(final\_integrated))

# PCAの実行

final\_integrated <- RunPCA(final\_integrated, features = VariableFeatures(final\_integrated))

# 近隣探索とクラスタリング

final\_integrated <- FindNeighbors(final\_integrated, dims = 1:20)

final\_integrated <- FindClusters(final\_integrated, resolution = 0.5)

# UMAPの実行

final\_integrated <- RunUMAP(final\_integrated, dims = 1:20)

# final\_integrated の meta.data にアクセスし、新しい group 列を作成

final\_integrated@meta.data <- final\_integrated@meta.data %>%

mutate(

group = case\_when(

grepl("^normal", orig.ident) ~ "normallung", # "normallung"で始まる場合

grepl("^IPFlung", orig.ident) ~ "IPFlung", # "IPFlung"で始まる場合

TRUE ~ as.character(orig.ident) # それ以外の場合、元の値を保持

)

)

# 更新後の group 列を確認

unique(final\_integrated@meta.data$group)

# 結果を確認

head(final\_integrated@meta.data)

# 結果の視覚化

head(final\_integrated@meta.data)

DimPlot(final\_integrated, reduction = "umap", label = TRUE) + NoLegend()

DimPlot(final\_integrated, reduction = "umap", group.by = "group", label = TRUE) + NoLegend()

DimPlot(final\_integrated, reduction = "umap", split.by = "group", label = TRUE) + NoLegend()

ref <- celldex::HumanPrimaryCellAtlasData()

results <- SingleR(test = as.SingleCellExperiment(final\_integrated), ref = ref, labels = ref$label.main)

final\_integrated$singlr\_labels <- results$labels

DimPlot(final\_integrated, reduction = 'umap', label.size = 2.5, pt.size = 0.1, group.by = 'singlr\_labels', label = TRUE )

FeaturePlot(final\_integrated, features = c("LRP1", "COL1A1", "COL1A2"))

FeaturePlot(final\_integrated, features = c("ACTA2", "MYH11", "TAGLN"))

FeaturePlot(final\_integrated, features = c("HHIP", "MUSTN1", "SCX"))

FeaturePlot(final\_integrated, features = c("ITGAX", "SIGLECF", "CD68"))#AlveolarMacrophage

saveRDS(final\_integrated, file="GSE128033integrated\_obj\_finals\_SingleR\_SCtransform.rds")

# Fibroblastsのみ取り出し再解析

Idents(final\_integrated) <- final\_integrated@meta.data$singlr\_labels

fibroblasts\_data <- subset(final\_integrated, idents = "Fibroblasts")

fibroblasts\_data <- SCTransform(fibroblasts\_data, verbose = FALSE)

fibroblasts\_data <- RunPCA(fibroblasts\_data, verbose = FALSE)

fibroblasts\_data <- RunUMAP(fibroblasts\_data, reduction = "pca", dims = 1:30, verbose = FALSE)

fibroblasts\_data <- FindNeighbors(fibroblasts\_data, reduction = "pca", dims = 1:30, verbose = FALSE)

fibroblasts\_data <- FindClusters(fibroblasts\_data, resolution = 0.5, verbose = FALSE)

fibroblasts\_data <- RenameIdents(fibroblasts\_data, .old = names(Idents(fibroblasts\_data)), .new = paste("Cluster", names(Idents(fibroblasts\_data)), sep = ""))

fibroblasts\_data$seurat\_clusters <- Idents(fibroblasts\_data)

head(fibroblasts\_data@meta.data)

str(fibroblasts\_data@meta.data)

# UMAP表示

DimPlot(fibroblasts\_data, reduction = "umap", label = TRUE) + NoLegend()

DimPlot(fibroblasts\_data, reduction = "umap", split.by = "group", label = TRUE) + NoLegend()

# 遺伝子リスト

genes <- c("LRP1", "COL1A1", "COL1A2", "ACTA2", "MYH11", "TAGLN", "COL13A1", "ITGA8", "CXCL14", "NPNT",

"COL14A1", "PI16", "MMP3", "CYGB", "RPT4", "MEG3", "SNHG18", "ADRP", "PPARG", "FABP1", "FABP4",

"LPL", "TOP2A", "MKI67", "MALAT1", "LOCKD", "HMGB2", "PDGFRB", "HIGD1B", "COX4I2", "NOTCH3",

"FAM162B")

# DotPlot作成

DotPlot(fibroblasts\_data, features = genes, dot.scale = 3, split.by = "group") + RotatedAxis()

# Fibroblasts\_Cluster3 のデータを抽出

cluster3\_data <- subset(fibroblasts\_data, idents = "3")

# normallung と IPFlung のデータを分ける

normallung\_data <- subset(cluster3\_data, subset = group == "normallung")

IPFlung\_data <- subset(cluster3\_data, subset = group == "IPFlung")

# Idents を group 列の値に基づいて設定

Idents(cluster3\_data) <- cluster3\_data$group

# 設定後のアイデンティティを確認

levels(Idents(cluster3\_data))

# 差異発現解析を実行

differential\_expression\_results <- FindMarkers(cluster3\_data, ident.1 = "normallung", ident.2 = "IPFlung", logfc.threshold = 0)

# ボルケーノプロットを作成

ggplot(differential\_expression\_results, aes(x = avg\_log2FC, y = -log10(p\_val))) +

geom\_point(aes(color = p\_val\_adj < 0.05), alpha = 0.5) +

scale\_color\_manual(values = c("grey", "red")) +

theme\_minimal() +

labs(x = "Log2 fold change (IPFlung vs normallung)",

y = "-Log10 p-value",

title = "Volcano plot of differential expression")

# IPFlungで上昇している遺伝子のみを抽出

upregulated\_genes <- differential\_expression\_results %>%

filter(avg\_log2FC > 0) %>%

arrange(desc(avg\_log2FC)) %>%

head(100)

# 上昇している上位100遺伝子を表示

print(upregulated\_genes)

saveRDS(fibroblasts\_data, file="GSE128033fibroblasts\_finals\_SingleR\_SCtransform.rds")

cell\_names <- colnames(fibroblasts\_data)

common\_cells <- cell\_names[cell\_names %in% colnames(final\_integrated)]

final\_integrated[['Fibroblast\_Clusters']] <- rep("Not Fibroblast", ncol(final\_integrated))

final\_integrated$Fibroblast\_Clusters[common\_cells] <- Idents(final\_integrated)[common\_cells]

table(final\_integrated$Fibroblast\_Clusters)

# Fibroblastsデータのクラスター情報を用いて新しい識別子を作成

final\_integrated$group\_identity <- ifelse(final\_integrated$singlr\_labels == "Fibroblasts",

paste("Fibroblasts\_Cluster", final\_integrated$Fibroblast\_Clusters),

final\_integrated$singlr\_labels)

saveRDS(final\_integrated, file="GSE128033integrated\_obj\_finals\_SingleR\_SCtransform.rds")

# CellChatオブジェクトの作成

DefaultAssay(final\_integrated) <- "SCT" # SCT アッセイをデフォルトに設定

cellchat\_data <- createCellChat(object = final\_integrated, group.by = "group\_identity")

ls("package:CellChat")

data(CellChatDB.human)

cellchat\_data@DB <- CellChatDB.human

cellchat\_data <- subsetData(cellchat\_data)

cellchat\_data <- identifyOverExpressedGenes(cellchat\_data)

cellchat\_data <- identifyOverExpressedInteractions(cellchat\_data)

# ネットワークを計算

cellchat\_data <- computeCommunProb(cellchat\_data)

cellchat\_data <- computeCommunProbPathway(cellchat\_data)

cellchat\_data <- aggregateNet(cellchat\_data)

groupSize <- as.numeric(table(cellchat\_data@idents)) # クラスターごとの細胞数を保存

par(mfrow = c(1,2), xpd=TRUE) # 次のプロットを並べて表示するためのパラメータの設定

p1 <- netVisual\_circle(cellchat\_data@net$count, vertex.weight = groupSize, weight.scale = T, label.edge= T, title.name = "Number of interactions")

p2 <- netVisual\_circle(cellchat\_data@net$weight, vertex.weight = groupSize, weight.scale = T, label.edge= F, title.name = "Interaction weights/strength")

mat <- cellchat\_data@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.name = rownames(mat)[i])

}

pathways.show <- c("CXCL")

vertex.receiver = seq(1,4) # a numeric vector.

netVisual\_aggregate(cellchat\_data, signaling = pathways.show, vertex.receiver = vertex.receiver)

par(mfrow=c(1,1))

netVisual\_aggregate(cellchat\_data, signaling = pathways.show, layout = "circle")

par(mfrow=c(1,1))

netVisual\_aggregate(cellchat\_data, signaling = pathways.show, layout = "chord")

pathways.show <- c("PTN")

vertex.receiver = seq(1,4) # a numeric vector.

netVisual\_aggregate(cellchat\_data, signaling = pathways.show, vertex.receiver = vertex.receiver)

par(mfrow=c(1,1))

netVisual\_aggregate(cellchat\_data, signaling = pathways.show, layout = "circle")

par(mfrow=c(1,1))

netVisual\_aggregate(cellchat\_data, signaling = pathways.show, layout = "chord")

------------ここまで

# Fibroblastsクラスターを抽出

# クラスターのレベルを確認

# cluster3のみを抽出

# UMAPプロットを実行してクラスターを表示

# 例としてLRP1遺伝子の発現を可視化

fibroblasts\_data <- subset(integrated\_data, subset = singlr\_labels == "Fibroblasts")

table(Idents(fibroblasts\_data))

cluster3\_data <- subset(fibroblasts\_data, idents = "3")

DimPlot(cluster3\_data, reduction = "umap", label = TRUE) + NoLegend()

FeaturePlot(cluster3\_data, features = "LRP1", cols = c("lightgray", "blue"))

# 細胞名を取得

# 細胞名が integrated\_data に存在するかどうかの確認

# Fibroblast\_Clusters 列を integrated\_data の meta.data に追加し、デフォルト値を設定

# fibroblasts\_data の各細胞に対応するクラスターIDを integrated\_data に割り当て

# Meta.data にクラスターIDを割り当てる正しい方法

# 割り当てた後のクラスター情報の確認

cell\_names <- colnames(fibroblasts\_data)

common\_cells <- cell\_names[cell\_names %in% colnames(integrated\_data)]

integrated\_data[['Fibroblast\_Clusters']] <- rep("Not Fibroblast", ncol(integrated\_data))

integrated\_data$Fibroblast\_Clusters[common\_cells] <- Idents(fibroblasts\_data)[common\_cells]

table(integrated\_data$Fibroblast\_Clusters)

# Fibroblastsデータのクラスター情報を用いて新しい識別子を作成

integrated\_data$group\_identity <- ifelse(integrated\_data$singlr\_labels == "Fibroblasts",

paste("Fibroblasts\_Cluster", integrated\_data$Fibroblast\_Clusters),

integrated\_data$singlr\_labels)

DimPlot(integrated\_data, reduction = "umap")

DimPlot(integrated\_data, reduction = "umap", split.by = "sample\_type", label =TRUE)

FeaturePlot(integrated\_data, features = c("LRP1"), split.by = "sample\_type", max.cutoff = 3, cols = c("grey", "red"))

FeaturePlot(integrated\_data, features = c("SFTPB", "COL1A1", "COL1A2"))

DimPlot(integrated\_data, reduction = "umap", group.by = "group\_identity", label = TRUE) + NoLegend()

# 表示したい遺伝子リスト

features <- c("LRP1", "COL1A1", "ACTA2")

# 特定のクラスターのみを含むサブセットを作成

target\_clusters <- c("Fibroblasts\_Cluster 1", "Fibroblasts\_Cluster 2",

"Fibroblasts\_Cluster 3", "Fibroblasts\_Cluster 4")

subset\_data <- subset(integrated\_data, subset = group\_identity %in% target\_clusters)

# DotPlot のためのカスタムカラーパレットの作成

colors <- c("Fibroblasts\_Cluster1" = "brown",

"Fibroblasts\_Cluster2" = "lightblue",

"Fibroblasts\_Cluster3" = "navy",

"Fibroblasts\_Cluster4" = "darkgreen")

# DotPlot の描画

DotPlot(subset\_data, features = c("LRP1", "COL1A1", "ACTA2"), group.by = "group\_identity",

dot.scale = 8, cols = colors) + RotatedAxis()

# CellChatオブジェクトの作成

cellchat\_data <- createCellChat(object = integrated\_data, group.by = "group\_identity")

# 使用可能な関数を表示

ls("package:CellChat")

# CellChatデータベースの読み込み

data(CellChatDB.human)

cellchat\_data@DB <- CellChatDB.human

# CellChatのデータ処理

cellchat\_data <- subsetData(cellchat\_data)

cellchat\_data <- identifyOverExpressedGenes(cellchat\_data)

cellchat\_data <- identifyOverExpressedInteractions(cellchat\_data)

# ネットワークを計算

cellchat\_data <- computeCommunProb(cellchat\_data)

cellchat\_data <- computeCommunProbPathway(cellchat\_data)

cellchat\_data <- aggregateNet(cellchat\_data)

# View(cellchat@net$count)

# View(cellchat@net$weight)

groupSize <- as.numeric(table(cellchat\_data@idents)) # クラスターごとの細胞数を保存

par(mfrow = c(1,2), xpd=TRUE) # 次のプロットを並べて表示するためのパラメータの設定

# 細胞間コミュニケーションの数とweight/strength をプロット

p1 <- netVisual\_circle(cellchat\_data@net$count, vertex.weight = groupSize, weight.scale = T, label.edge= T, title.name = "Number of interactions")

p2 <- netVisual\_circle(cellchat\_data@net$weight, vertex.weight = groupSize, weight.scale = T, label.edge= F, title.name = "Interaction weights/strength")

mat <- cellchat\_data@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.name = rownames(mat)[i])

}

pathways.show <- c("CXCL")

# Hierarchy plot

# Here we define `vertex.receive` so that the left portion of the hierarchy plot shows signaling to fibroblast and the right portion shows signaling to immune cells

vertex.receiver = seq(1,4) # a numeric vector.

netVisual\_aggregate(cellchat\_data, signaling = pathways.show, vertex.receiver = vertex.receiver)

# Circle plot

par(mfrow=c(1,1))

netVisual\_aggregate(cellchat\_data, signaling = pathways.show, layout = "circle")

# Chord diagram

par(mfrow=c(1,1))

netVisual\_aggregate(cellchat\_data, signaling = pathways.show, layout = "chord")

str(CellChatDB.human)

# interaction 内の pathway\_name の内容を表示

if (!is.null(CellChatDB.human$interaction) && !is.null(CellChatDB.human$interaction$pathway\_name)) {

print(CellChatDB.human$interaction$pathway\_name)

} else {

cat("指定されたパスが存在しないか、正しくない可能性があります。\n")

}

pathways.show <- c("PTN")

# Hierarchy plot

# Here we define `vertex.receive` so that the left portion of the hierarchy plot shows signaling to fibroblast and the right portion shows signaling to immune cells

vertex.receiver = seq(1,4) # a numeric vector.

netVisual\_aggregate(cellchat\_data, signaling = pathways.show, vertex.receiver = vertex.receiver)

# Circle plot

par(mfrow=c(1,1))

netVisual\_aggregate(cellchat\_data, signaling = pathways.show, layout = "circle")

# Chord diagram

par(mfrow=c(1,1))

netVisual\_aggregate(cellchat\_data, signaling = pathways.show, layout = "chord")

--------Psuedbulk解析。LRP1高いクラスターについてのボルケーノプロット

# Fibroblasts\_Cluster3 のデータを抽出

cluster3\_data <- subset(fibroblasts\_data, idents = "Fibroblasts\_Cluster3")

# normallung と IPFlung のデータを分ける

normallung\_data <- subset(cluster3\_data, subset = sample\_type == "normallung")

IPFlung\_data <- subset(cluster3\_data, subset = sample\_type == "IPFlung")

# 遺伝子発現の差異を計算

differential\_expression\_results <- FindMarkers(cluster3\_data, ident.1 = "normallung", ident.2 = "IPFlung", logfc.threshold = 0)

# ボルケーノプロットを作成

ggplot(differential\_expression\_results, aes(x = logFC, y = -log10(p\_val))) +

geom\_point(aes(color = p\_val\_adj < 0.05), alpha = 0.5) +

scale\_color\_manual(values = c("grey", "red")) +

theme\_minimal() +

labs(x = "Log2 fold change (IPFlung vs normallung)",

y = "-Log10 p-value",

title = "Volcano plot of differential expression")

# IPFlungで上昇している遺伝子のみを抽出

upregulated\_genes <- differential\_expression\_results %>%

filter(logFC > 0) %>% # Log Fold Changeが正のもののみを選択

arrange(desc(logFC)) %>% # Log Fold Changeで降順に並べ替え

head(100) # 上位100を抽出

# 上昇している上位100遺伝子を表示

print(upregulated\_genes)

pbmc2 <- CreateSeuratObject(counts = pbmc.data2, project = "pbmc3k", min.cells = 3, min.features = 200)

pbmc2

normallung1 <- readRDS("C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\normallung1\_SCtransform.rds")

normallung3 <- readRDS("C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\normallung3\_SCtransform.rds")

-------

saveRDS(integrated\_data, file="GSE128033normallung1-2\_SCtransform.rds")

# meta.dataの内容を表示

head(integrated\_data@meta.data)

str(integrated\_data@meta.data)

------------

------------4/17GSE128033IPFlungintegrated

setwd("C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033IPFlungまとめ")

pbmc.data3 <- Read10X(data.dir = "C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033IPFlungまとめ\\GSE128033IPFlung1")

pbmc3 <- CreateSeuratObject(counts = pbmc.data3, project = "pbmc3k", min.cells = 3, min.features = 200)

pbmc3

pbmc3 <- SCTransform(pbmc3, vst.flavor = "v2", verbose = FALSE)

saveRDS(pbmc, file = "IPFlung1\_SCtransform.rds")

pbmc.data4 <- Read10X(data.dir = "C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033IPFlungまとめ\\GSE128033IPFlung2")

pbmc4 <- CreateSeuratObject(counts = pbmc.data4, project = "pbmc3k", min.cells = 3, min.features = 200)

pbmc4

pbmc4 <- SCTransform(pbmc4, vst.flavor = "v2", verbose = FALSE)

saveRDS(pbmc, file = "IPFlung2\_SCtransform.rds")

IPFlung1 <- readRDS("C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033IPFlungまとめ\\IPFlung1\_SCtransform.rds")

IPFlung2 <- readRDS("C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033IPFlungまとめ\\IPFlung2\_SCtransform.rds")

head(IPFlung@meta.data)

str(IPFlung@meta.data)

-------GSE128033normalとIPFまとめ

# normallung1 と normallung3 に sample\_type を追加

pbmc$sample\_type <- "normallung"

pbmc2$sample\_type <- "normallung"

# IPFlung1 と IPFlung3 に sample\_type を追加

pbmc3$sample\_type <- "IPFlung"

pbmc4$sample\_type <- "IPFlung"

# 念のために、これらの変更が正しく行われたかを確認

head(pbmc@meta.data)

head(pbmc3@meta.data)

# 統合のための特徴量を選択

features\_to\_integrate <- SelectIntegrationFeatures(object.list = list(pbmc, pbmc2, pbmc3, pbmc4), nfeatures = 3000)

# SCTransformを用いた統合準備

data\_list <- PrepSCTIntegration(object.list = list(pbmc, pbmc2, pbmc3, pbmc4), anchor.features = features\_to\_integrate)

# アンカーを探す

anchors <- FindIntegrationAnchors(object.list = data\_list, normalization.method = "SCT", anchor.features = features\_to\_integrate)

# データを統合

integrated\_data <- IntegrateData(anchorset = anchors, normalization.method = "SCT")

# 統合データの結果を確認

integrated\_data

# データをスケーリング

integrated\_data <- ScaleData(integrated\_data, verbose = FALSE)

# PCA を実行

integrated\_data <- RunPCA(integrated\_data, verbose = FALSE)

# UMAP を実行

integrated\_data <- RunUMAP(integrated\_data, dims = 1:30, verbose = FALSE)

# UMAP プロットを表示

DimPlot(integrated\_data, reduction = "umap")

DimPlot(integrated\_data, reduction = "umap")

DimPlot(integrated\_data, reduction = "umap", split.by = "normallung")

DimPlot(integrated\_data, reduction = "umap", split.by = "sample\_type", label =TRUE)

FeaturePlot(integrated\_data, features = c("LRP1"), split.by = "sample\_type", max.cutoff = 3, cols = c("grey", "red"))

VlnPlot(integrated\_data, features = "LRP1")

FeaturePlot(integrated\_data, features = c("SFTPC", "CCL4", "COL1A2"))

# どのアッセイにLRP1が存在するか確認

if("LRP1" %in% rownames(integrated\_data[["SCT"]]@data)) {

print("LRP1 is in SCT assay")

}

# SCT アッセイをデフォルトに設定

DefaultAssay(integrated\_data) <- "SCT"

# デフォルトのアッセイを使用してFeaturePlotを実行

FeaturePlot(integrated\_data, features = "LRP1", slot = "data", max.cutoff = 3, cols = c("grey", "red"))

setwd("C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal")

GSE128033normalbal\_normallung\_integrated <- readRDS("C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033normal\\GSE128033normallung1-2\_SCtransform.rds")

setwd("C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033IPFlungまとめ")

GSE128033IPFlung\_integrated <- readRDS("C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033IPFlungまとめ\\GSE128033IPFlung1-2\_SCtransform.rds")

GSE128033normalbal\_normallung\_integrated <- SCTransform(GSE128033normalbal\_normallung\_integrated, vst.flavor = "v2", verbose = FALSE) %>%

RunPCA(npcs = 30, verbose = FALSE)

GSE128033IPFlung\_integrated <- SCTransform(GSE128033IPFlung\_integrated, vst.flavor = "v2", verbose = FALSE) %>%

RunPCA(npcs = 30, verbose = FALSE)

IPFlung.list <- list(GSE128033normalbal\_normallung\_integrated = GSE128033normalbal\_normallung\_integrated, GSE128033IPFlung\_integrated = GSE128033IPFlung\_integrated)

features <- SelectIntegrationFeatures(object.list = IPFlung.list, nfeatures = 3000)

IPFlung.list <- PrepSCTIntegration(object.list = IPFlung.list, anchor.features = features)

immune.anchors <- FindIntegrationAnchors(object.list = IPFlung.list, normalization.method = "SCT",

anchor.features = features)

immune.combined.sct <- IntegrateData(anchorset = immune.anchors, normalization.method = "SCT")

immune.combined.sct <- RunPCA(immune.combined.sct, verbose = FALSE)

immune.combined.sct <- RunUMAP(immune.combined.sct, reduction = "pca", dims = 1:30, verbose = FALSE)

immune.combined.sct <- FindNeighbors(immune.combined.sct, reduction = "pca", dims = 1:30)

immune.combined.sct <- FindClusters(immune.combined.sct, resolution = 0.3)

DimPlot(immune.combined.sct, reduction = "umap")

FeaturePlot(immune.combined.sct, features = c("LRP1"), split.by = "sample\_type", max.cutoff = 3, cols = c("grey", "red"))

# メタデータにデータセットを示す列を追加

GSE128033normalbal\_normallung\_integrated$sample\_type <- "Normal"

GSE128033IPFlung\_integrated$sample\_type <- "IPF"

#データセットを並べて表示

DimPlot(immune.combined.sct, reduction = "umap")

DimPlot(immune.combined.sct, reduction = "umap", split.by = "sample\_type")

DimPlot(immune.combined.sct, reduction = "umap", split.by = "sample\_type", label =TRUE)

FeaturePlot(immune.combined.sct, features = c("LRP1"), split.by = "sample\_type", max.cutoff = 3, cols = c("grey", "red"))

FeaturePlot(immune.combined.sct, features = c("COL1A1"), split.by = "sample\_type", max.cutoff = 3, cols = c("grey", "darkgreen"))

FeaturePlot(immune.combined.sct, features = c("COL1A2"), split.by = "sample\_type", max.cutoff = 3, cols = c("grey", "darkblue"))

#マーカーとなる遺伝子の定義 (大文字で！)

Immune <- c("PTPRC") #CD45

Tcell <- c("CD3D", "CD4", "CD8A")

NK <- c("NKG7", "GZMA")

B <- c("CD19", "CD79A")

Myeloid <- c("ITGAM", "CD68")

AlveolarMacrophage <- c("ITGAX", "SIGLECF", "CD68")

DC <- c("ITGAX", "ITGAE")

Neutrophil <- c("LY6G", "NGP")

Epithelial <- c("EPCAM","CDH1")

Myofibroblast <- c("MUSTN1","ACTA2")

Fibroblast <- c("COL1A1", "COL1A2")

Endothelial <- c("PECAM1", "CDH5", "VWF")

Pericyte <- c("MCAM", "PDGFRB", "COX4I2")

NaiveCD4T <- c("IL7R", "CCR7")

CD14mono <- c("CD14", "LYZ")

MemoryCD4T <- c("IL7R", "S100A4")

CD8T <- c("CD8A")

FCGR3Amono <- c("FCGR3A", "MS4A7")

NKT <- c("GNLY", "NKG7")

DC <- c("FCER1A","CST3")

Platelet <- c("PPBP")

Erythrocyte <- c("HBA2", "HBB")

Endothelial <- c("CLDN5", "CDH5", "VWF")

Neutrophil <- c("CHI3L1", "ALPL", "ANXA3", "S100A8", "S100A9", "IFITM2", "FCGR3B")

Basophil <- c("CPA3", "IL4", "TAL1", "MS4A2", "TPSAB1")

Eosinophil <- c("ALOX15", "MYCT1", "CCL23")

AT2 <- c("SFTPB", "SFTPC", "MUC1", "SFTPD")

Basal\_cell <- c("KRT17", "KRT15", "MIR205HG", "KRT5")

DC <- c("HLA-DPB1","HLA-DQA1", "APOC1", "APOE")

Clubcell <- c("AHR", "MYC", "SCGB3A2", "CYP2F2")

AT1 <- c("MYRF", "AGER", "PDPN", "CLIC5")

Pericyte <- c("COX4I2", "TBX5", "CSPG4", "TRPC6", "PDGFRB")

Ciliatedcell <- c("FOXJ1", "TUBB1")

Gobletcell <- c("MUC5B", "MUC5AC", "SPDEF")

Neuroendcrinecell <- c("CALCA", "CHGA", "ASCL1")

Vascular\_smooth\_muscle <- c("CNN1", "ACTA2", "TAGLN", "RGS5")

Aiwway\_smooth\_muscle <- c("CNN1", "ACTA2", "TAGLN", "DES", "LGR6")

#バイオリンプロット(Immune cell)

VlnPlot(integrated\_obj, features = Immune)

#マーカー遺伝子の発現量でUMAPプロットを色塗り(Immune cell)

FeaturePlot(integrated\_obj, features = Immune)

#T cell

FeaturePlot(integrated\_obj, features = Tcell, ncol = 3, label=TRUE)

#NK cell

FeaturePlot(integrated\_obj, features = NK, label=TRUE)

#B cell

FeaturePlot(integrated\_obj, features = B, label=TRUE)

#Myeloid cell

FeaturePlot(integrated\_obj, features = Myeloid, label=TRUE)

#Alveolar Macrophage

FeaturePlot(integrated\_obj, features = AlveolarMacrophage, ncol = 3, label=TRUE)

#DC

FeaturePlot(integrated\_obj, features = DC, label=TRUE)

#Neutrophil

FeaturePlot(integrated\_obj, features = Neutrophil, label=TRUE)

#Epithelial cell

FeaturePlot(integrated\_obj, features = Epithelial, label=TRUE)

#Myofibroblast

FeaturePlot(integrated\_obj, features = Myofibroblast, label=TRUE)

#Fibroblast

FeaturePlot(integrated\_obj, features = Fibroblast, label=TRUE)

#Endothelial cell

FeaturePlot(integrated\_obj, features = Endothelial, ncol = 3, label=TRUE)

#Pericyte

FeaturePlot(integrated\_obj, features = Pericyte, ncol = 3, label=TRUE)

#NaiveCD4T

FeaturePlot(integrated\_obj, features = NaiveCD4T, ncol = 3, label=TRUE)

#CD14mono

FeaturePlot(integrated\_obj, features = CD14mono, ncol = 3, label=TRUE)

#MemoryCD4T

FeaturePlot(integrated\_obj, features = MemoryCD4T, ncol = 3, label=TRUE)

#CD8T

FeaturePlot(integrated\_obj, features = CD8T, ncol = 3, label=TRUE)

#FCGR3Amono

FeaturePlot(integrated\_obj, features = FCGR3Amono, ncol = 3, label=TRUE)

#NKT

FeaturePlot(integrated\_obj, features = NKT, ncol = 3, label=TRUE)

#DC

FeaturePlot(integrated\_obj, features = DC, ncol = 3, label=TRUE)

#Platelet

FeaturePlot(integrated\_obj, features = Platelet, ncol = 3, label=TRUE)

#Erythrocyte

FeaturePlot(integrated\_obj, features = Erythrocyte, ncol = 3, label=TRUE)

#Endothelial

FeaturePlot(integrated\_obj, features = Endothelial, ncol = 3, label=TRUE)

#Neutrophil

FeaturePlot(integrated\_obj, features = Neutrophil, ncol = 3, label=TRUE)

#Basophil

FeaturePlot(integrated\_obj, features = Basophil, ncol = 3, label=TRUE)

#Eosinophil

FeaturePlot(integrated\_obj, features = Eosinophil, ncol = 3, label=TRUE)

#AT2

FeaturePlot(integrated\_obj, features = AT2, ncol = 3, label=TRUE)

#Basal\_cell

FeaturePlot(integrated\_obj, features = Basal\_cell, ncol = 3, label=TRUE)

#DC

FeaturePlot(integrated\_obj, features = DC, ncol = 3, label=TRUE)

#Clubcell

FeaturePlot(integrated\_obj, features = Clubcell, ncol = 3, label=TRUE)

#AT1

FeaturePlot(integrated\_obj, features = AT1, ncol = 3, label=TRUE)

#Pericyte

FeaturePlot(integrated\_obj, features = Pericyte, ncol = 3, label=TRUE)

#Ciliatedcell

FeaturePlot(integrated\_obj, features = Ciliatedcell, ncol = 3, label=TRUE)

#Gobletcell

FeaturePlot(integrated\_obj, features = Gobletcell, ncol = 3, label=TRUE)

#Neuroendcrinecell

FeaturePlot(integrated\_obj, features = Neuroendcrinecell, ncol = 3, label=TRUE)

#Vascular\_smooth\_muscle

FeaturePlot(integrated\_obj, features = Vascular\_smooth\_muscle, ncol = 3, label=TRUE)

#Aiwway\_smooth\_muscle

FeaturePlot(integrated\_obj, features = Aiwway\_smooth\_muscle, ncol = 3, label=TRUE)

#クラスターへの細胞種の割り当て(UMAP印刷して見比べながらつける)

new.cluster.ids <- c("monocyte\_FCGR3A", "CD8Tcell", "monocyte\_CD14", "endothelial", "fibroblast/myofibroblast", "neutrophil", "CD8Tcell\_2", "epithelial", "naiveCD4Tcell", "NKTcell", "naiveCD4Tcell\_2", "alveolarmacrophage", "basophil", "memoryCD4Tcell", "DC", "ciliatedcell", "Bcell", "AT1", "clubcell", "eosinophil")

names(new.cluster.ids) <- levels(integrated\_obj)

integrated\_obj <- RenameIdents(integrated\_obj, new.cluster.ids)

#ラベル付きでUMAPプロットを出力

png("umap\_cls.png", width = 800, height = 500)

DimPlot(integrated\_obj, reduction = "umap", label = TRUE, pt.size = 0.5)

dev.off()

#Legendなしラベルあり

DimPlot(integrated\_obj, reduction = "umap", label = TRUE, label.size = 4, repel = TRUE) + NoLegend()

#データセットを並べて表示

DimPlot(integrated\_obj, reduction = "umap", split.by = "sample\_type")

DimPlot(integrated\_obj, reduction = "umap", split.by = "sample\_type", label =TRUE)

FeaturePlot(integrated\_obj, features = c("LRP1"), split.by = "sample\_type", max.cutoff = 3, cols = c("grey", "red"))

#lungオブジェクトをファイルに保存します

saveRDS(integrated\_obj, file="GSE128033integrated\_obj\_finals\_SCtransform.rds")

-----------SinglRによるアノテーション

ref <- celldex::HumanPrimaryCellAtlasData()

results <- SingleR(test = as.SingleCellExperiment(integrated\_obj), ref = ref, labels = ref$label.main)

integrated\_obj$singlr\_labels <- results$labels

DimPlot(integrated\_obj, reduction = 'umap', label.size = 2.5, pt.size = 0.1, group.by = 'singlr\_labels', label = TRUE )

saveRDS(integrated\_obj, file="GSE128033integrated\_obj\_finals\_SingleR\_SCtransform.rds")

----Dot plotの色々な解析など

setwd("C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033IPFlungまとめ")

integrated\_obj <- readRDS("C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033IPFlungまとめ\\GSE128033integrated\_obj\_finals\_SingleR\_SCtransform.rds")

# meta.dataを表示する

head(Idents(integrated\_obj))

#ラベル付きでUMAPプロットを出力

DimPlot(integrated\_obj, reduction = 'umap', label.size = 2.5, pt.size = 0.1, group.by = 'singlr\_labels', label = TRUE )

# cluster\_centersデータフレーム内のseurat\_clusters列のユニークな値を表示

unique(cluster\_centers$seurat\_clusters)

# UMAPプロットの作成

umap\_plot <- DimPlot(integrated\_obj, reduction = "umap", label = FALSE, pt.size = 1)

# 各クラスターの中心点を計算

cluster\_centers <- as.data.frame(Embeddings(integrated\_obj[["umap"]]))

cluster\_centers$active.ident <- as.character(integrated\_obj@active.ident) # ファクターを文字列に変換してクラスター名を取得

# 特定のクラスターに名前を付ける

umap\_plot +

geom\_point(data = subset(cluster\_centers, active.ident == "Fibroblasts"),

aes(x = UMAP\_1, y = UMAP\_2), color = "darkblue", size = 1) +

# 名前を付けたクラスターの目立つ色合い

scale\_color\_manual(values = c("Fibroblasts" = "darkblue"), guide = FALSE) +

# 他のクラスターの地味な色合い

scale\_color\_manual(values = c("Other\_Clusters" = "grey"), guide = FALSE)

# 特定のクラスターのみにラベルを付ける

geom\_text\_repel(data = subset(integrated\_obj, active.ident == "Fibroblasts"),

aes(label = "Fibroblasts", x = UMAP\_1, y = UMAP\_2),

color = "darkblue", size = 3)

head(rownames(integrated\_obj), 20)

lrp1\_gene <- "ACTA2"

lrp1\_present <- lrp1\_gene %in% rownames(integrated\_obj)

lrp1\_present

# 検索する遺伝子名のリストを作成する

gene\_names <- c("LRP1", "LRP2", "LRP3", "LRP4", "LRP5", "LRP6") # LRPファミリーの例

# 各遺伝子名を含む行が存在するかどうかを調べる

gene\_presence <- sapply(gene\_names, function(gene) gene %in% rownames(integrated\_obj))

# 各遺伝子の存在を表示する

gene\_presence

#Legendなしラベルあり

DimPlot(integrated\_obj, reduction = "umap", label = TRUE, label.size = 4, repel = TRUE) + NoLegend()

DimPlot(integrated\_obj, reduction = "umap", label = TRUE, label.size = 6, pt.size = 3, repel = TRUE) + NoLegend()

#データセットを並べて表示

DimPlot(integrated\_obj, reduction = "umap", split.by = "sample\_type")

DimPlot(integrated\_obj, reduction = "umap", split.by = "sample\_type", label =TRUE)

FeaturePlot(integrated\_obj, features = c("LRP1"), split.by = "sample\_type", min.cutoff = 0, max.cutoff = 0.5, cols = c("grey", "darkgreen"))

FeaturePlot(integrated\_obj, features = c("LRP1"), label.size = 6, pt.size = 3, repel = TRUE, min.cutoff = 0, max.cutoff = 0.5, cols = c("grey", "darkgreen"))

FeaturePlot(integrated\_obj, features = c("BPIFB1"), split.by = "sample\_type", max.cutoff = 3, cols = c("grey", "darkblue"))

FeaturePlot(GSE128033IPFlung\_integrated, features = c("LRP1"), max.cutoff = 3, cols = c("grey", "darkblue"))

# Violin plots can also be split on some variable. Simply add the splitting variable to object

# metadata and pass it to the split.by argument

VlnPlot(integrated\_obj, features = "percent.mt", split.by = "sample\_type")

features <- c("LRP1")

# Ridge plots - from ggridges. Visualize single cell expression distributions in each cluster

RidgePlot(integrated\_obj, features = features, ncol = 2)

# Violin plot - Visualize single cell expression distributions in each cluster

VlnPlot(integrated\_obj, features = features)

# Feature plot - visualize feature expression in low-dimensional space

FeaturePlot(integrated\_obj, features = features)

# Dot plots - the size of the dot corresponds to the percentage of cells expressing the

# feature in each cluster. The color represents the average expression level

DotPlot(integrated\_obj, features = features) + RotatedAxis()

# SplitDotPlotGG has been replaced with the `split.by` parameter for DotPlot

DotPlot(integrated\_obj, features = features, split.by = "sample\_type") + RotatedAxis()

# Seuratオブジェクトからメタデータを取得

meta\_data <- integrated\_obj@meta.data

# メタデータの最初の数行を表示

head(meta\_data)

# メタデータの最初の数行を表示

head(meta\_data$seurat\_clusters)

#色々なドットプロットやUMAPの色付け

features <- c("LRP1")

DotPlot(integrated\_obj, features = features) + RotatedAxis()

DotPlot(integrated\_obj, features = c("CD3D", "MS4A1", "CD14"),

cols = c("lightgray", "orange", "red", "darkred", "black"))

DotPlot(integrated\_obj, features = c("LRP1"),

cols = c("lightgray", "orange", "red", "darkred", "black"))

DotPlot(integrated\_obj, features = c("LRP1")) &

scale\_color\_gradientn(colours = c("lightgray", "orange", "red", "darkred", "black"))

DotPlot(integrated\_obj, features = c("LRP1")) &

scale\_color\_viridis\_c(option= "D")

FeaturePlot(integrated\_obj, features = c("LRP1"), ncol = 3) &

scale\_color\_gradientn(colours = c("lightgray", "gray", "orange", "red", "darkred", "black"))

FeaturePlot(integrated\_obj, features = c("LRP1"), ncol = 2) &

scale\_color\_viridis\_c(option = "D")

col <- rev(brewer.pal(n = 11, name = "RdBu"))

col <- c(col, rep(col[length(col)], times = 11)) # timesの回数は好きに決定

FeaturePlot(integrated\_obj, features = c("LRP1"), ncol = 3) &

scale\_colour\_gradientn(colours = col)

FeaturePlot(integrated\_obj, features = c("LRP1"), ncol = 3, order =T) &

scale\_colour\_gradientn(colours = rev(brewer.pal(n = 11, name = "RdBu")))

VlnPlot(integrated\_obj,

features = c("LRP1","COL1A1","COL1A2","BPIFB1","SFTPB"),

stack = T)

DimPlot(integrated\_obj,

cells.highlight = WhichCells(integrated\_obj, idents = c("fibroblast/myofibroblast")),

cols.highlight = c("darkgreen"))

#active.identスロットに保存された名前の確認方法

levels(integrated\_obj)

features <- c("LRP1")

# DotPlotを作成する

DotPlot(

object = integrated\_obj, # Seuratオブジェクトを指定

features = features, # DotPlotを作成する遺伝子のリストを指定

group.by = "sample\_type" # 使用する細胞腫の情報を指定

) + RotatedAxis()

# Seuratオブジェクトから特定の細胞群を抽出

# 以下の例では、"ClusterName" が特定のクラスタ名（例: "Cluster1"）に一致する細胞を抽出しています

cluster\_name\_to\_extract <- "fibroblast/myofibroblast"

fibroblast\_integrated\_obj <- subset(integrated\_obj, idents = cluster\_name\_to\_extract)

# "Fibroblast" クラスターの細胞群に対してUMAPを再計算

fibroblast\_integrated\_obj <- subset(integrated\_obj, idents = cluster\_name\_to\_extract)

fibroblast\_integrated\_obj <- RunUMAP(fibroblast\_integrated\_obj)

# 詳細なUMAPのプロット

umap\_plot <- DimPlot(fibroblast\_integrated\_obj, group.by = "sample\_type")

# UMAPプロットの表示

print(umap\_plot)

DimPlot(fibroblast\_integrated\_obj, reduction = "umap", split.by = "sample\_type")

# クラスタリング結果を取得

cluster\_results <- fibroblast\_integrated\_obj$seurat\_clusters

features <- c("LRP1")

# DotPlotを作成する

DotPlot(

object = fibroblast\_integrated\_obj, # Seuratオブジェクトを指定

features = features, # DotPlotを作成する遺伝子のリストを指定

group.by = "sample\_type" # 使用する細胞腫の情報を指定

) + RotatedAxis()

# オブジェクトの名前を変更する

integrated\_obj <- fibroblast\_integrated\_obj

rm(fibroblast\_integrated\_obj)

# クラスタリングを再実行

integrated\_obj <- FindNeighbors(integrated\_obj, dims = 1:20)

integrated\_obj <- FindClusters(integrated\_obj, resolution = 0.5) # resolutionを調整

integrated\_obj <- RunUMAP(integrated\_obj, dims = 1:20)

# 新しいクラスターごとに遺伝子発現の差異を評価

cluster\_marker\_genes <- FindAllMarkers(integrated\_obj, only.pos = TRUE, assay = "RNA")

# UMAPを再計算

integrated\_obj <- RunUMAP(integrated\_obj, dims = 1:20) # ディメンションの範囲を適切に調整

# UMAPプロット

DimPlot(integrated\_obj, group.by = "sample\_type") # "sample\_type" はクラスター情報のカラム名に合わせて調整

DimPlot(integrated\_obj, reduction = "umap", split.by = "sample\_type")

DimPlot(integrated\_obj, reduction = "umap", split.by = "sample\_type", label =TRUE)

FeaturePlot(integrated\_obj, features = c("LRP1"), split.by = "sample\_type", max.cutoff = 3, cols = c("grey", "red"))

features <- c("LRP1")

# DotPlotを作成する

DotPlot(

object = integrated\_obj, # Seuratオブジェクトを指定

features = features, # DotPlotを作成する遺伝子のリストを指定

group.by = "sample\_type", # 使用する細胞腫の情報を指定

dot.scale = 20,

cols = c("lightgray", "darkgreen")

) + RotatedAxis()

DotPlot(integrated\_obj, features = c("LRP1")) &

scale\_color\_gradientn(colours = c("lightgray", "orange", "red", "darkred", "black"))

DotPlot(integrated\_obj, features = c("LRP1"), dot.scale = 20, cols = c("lightgray", "orange", "red", "darkred", "black"))

DotPlot(integrated\_obj, features = c("LRP1"), dot.scale = 20, cols = c("lightgray", "darkgreen"))

# クラスター列をファクターとして再設定し、指定した順序になるようにする

integrated\_obj@meta.data$seurat\_clusters <- factor(integrated\_obj@meta.data$seurat\_clusters, levels = c("0", "3", "4", "2", "1"))

# DotPlotを作成する

DotPlot(integrated\_obj,

features = c("LRP1", "ACTA2"),

dot.scale = 20,

cols = c("lightgray", "darkgreen"),

group.by = "seurat\_clusters" # クラスター情報を指定

)

# クラスター列をファクターとして再設定し、指定した順序になるようにする

integrated\_obj@meta.data$seurat\_clusters <- factor(integrated\_obj@meta.data$seurat\_clusters, levels = c("0", "3", "4", "2", "1"))

# DotPlotを作成する

DotPlot(integrated\_obj,

features = c("LRP1"),

dot.scale = 20,

cols = c("lightgray", "darkgreen"),

group.by = "seurat\_clusters" # クラスター情報を指定

)

FeaturePlot(integrated\_obj, features = c("LRP1"), split.by = "sample\_type", ncol = 3, pt.size = 1.5) &

scale\_color\_gradientn(colours = c("lightgray", "gray", "orange", "red", "darkred", "black"))

FeaturePlot(

object = integrated\_obj,

features = c("LRP1"),

split.by = "sample\_type",

ncol = 3,

pt.size = 1.5, # ドットのサイズを大きくする

cols = c("lightgray", "gray", "orange", "red", "darkred", "black") # カラーパレットを設定する

)

FeaturePlot(

object = integrated\_obj,

features = c("LRP1"),

split.by = "sample\_type",

ncol = 3,

pt.size = 3, # ドットのサイズを大きくする

cols = c("lightgray", "darkgreen") # カラーパレットを設定する

)

FeaturePlot(

object = integrated\_obj,

features = c("ACTA2"),

split.by = "sample\_type",

ncol = 3,

pt.size = 3, # ドットのサイズを大きくする

cols = c("lightgray", "darkred") # カラーパレットを設定する

)

FeaturePlot(

object = integrated\_obj,

features = c("ACTA2"),

split.by = "sample\_type",

ncol = 3,

pt.size = 3, # ドットのサイズを大きくする

cols = c("lightgray", "#929600") # カラーパレットを設定する

)

# UMAPのdot plotを大きくカラフルに表示する

DimPlot(

object = integrated\_obj,

split.by = "sample\_type", # Seuratオブジェクトを指定

reduction = "umap", # 使用する次元削減の方法を指定

label = TRUE, # ラベルの表示

pt.size = 3, # ドットのサイズ

cols = c("red", "blue", "green", "orange", "purple"), # ドットの色

label.size = 10 # ラベルのサイズ

)

# fibroblastクラスターのデータを取得

fibroblast\_data <- integrated\_obj@assays$integrated@data[, integrated\_obj@meta.data$seurat\_clusters == "0"]

# 各遺伝子の平均発現量を計算

gene\_mean\_expression <- rowMeans(fibroblast\_data)

# 平均発現量の高い順に遺伝子を並び替え

top50\_genes <- names(sort(gene\_mean\_expression, decreasing = TRUE))[1:50]

# トップ50の遺伝子を表示

print(top50\_genes)

# 遺伝子名を取得

gene\_names <- rownames(fibroblast\_data)

# すべての遺伝子名を表示

print(gene\_names)

# 各遺伝子の平均発現量を計算

gene\_mean\_expression <- rowMeans(fibroblast\_data)

# 発現量の高い順に遺伝子名を並べ替え

sorted\_gene\_names <- names(sort(gene\_mean\_expression, decreasing = TRUE))

# 並べ替えられた遺伝子名を表示

print(sorted\_gene\_names)

# RDSファイルにオブジェクトを保存

saveRDS(integrated\_obj, file = "fibroblast\_recluster\_SCtransform.rds")

#RDSファイル呼び出し

setwd("C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033IPFlungまとめ")

integrated\_obj <- readRDS("C:\\Users\\yamas\\OneDrive\\デスクトップ\\GSE128033IPFlungまとめ\\fibroblast\_recluster\_SCtransform.rds")

#ファイルを呼び出して再度UMAPを作成する場合は　クラスタリングを再実行　からスクリプトを流していく

# openxlsxパッケージをインストール

install.packages("openxlsx")

# Install sctransform from CRAN

install.packages("sctransform")

# CRAN version compile from source (recommended)

remotes::install\_cran("qs", type = "source", configure.args = "--with-simd=AVX2")

#BiocManager::install("ComplexHeatmap")

#devtools::install\_github("jinworks/CellChat")

usethis::create\_github\_token()

usethis::edit\_r\_environ()

GITHUB\_PAT=your\_new\_pat\_here

remotes:::github\_pat()

usethis::browse\_github\_pat()

install\_github("<user>/<repository>")

install\_github("jokergoo/ComplexHeatmap")

devtools::install\_github("jokergoo/circlize")

install.packages("circlize")

if (!require("BiocManager", quietly = TRUE))

install.packages("BiocManager")

BiocManager::install("pcaMethods")

#パッケージのインストール

install.packages("qgraph")

install.packages( "psych" )

remotes::install\_version("SeuratObject", "4.1.4", repos = c("https://satijalab.r-universe.dev", getOption("repos")))

remotes::install\_version("Seurat", "4.4.0", repos = c("https://satijalab.r-universe.dev", getOption("repos")))

3

if (!require("BiocManager", quietly = TRUE))

install.packages("BiocManager")

BiocManager::install("org.Hs.eg.db")

if (!require("BiocManager", quietly = TRUE))

install.packages("BiocManager")

BiocManager::install("clusterProfiler")

a

install.packages("GO.db", dependencies = TRUE)

if (!require("BiocManager", quietly = TRUE))

install.packages("BiocManager")

BiocManager::install("glmGamPoi")