Tutorial

Yufang Liu

2023-09-26

# Introductino for OMIC

This file is a brief tutorial for our proposed method Orthogonal Multimodality Integration and Clustering in Multimodal Single-cell Data (OMIC). This tutorial contains the instructions of the main functions of OMIC, including the data preprocessing, the clustering, and interpretation of model coefficients, which provides quantification of additional prediction power for each variable, and determining which RNAs and ADTs are differentially expressed to provide significant prediction power.

## 1. Load Packages

Load the requaired package for the tutorial. The detailed list of the required packages are presented in the read\_me file.

library(Seurat)

## Attaching SeuratObject

library(SeuratData)

## ── Installed datasets ───────────────────────────────────── SeuratData v0.2.2 ──

## ✔ adiposeref 1.0.0 ✔ ifnb 3.1.0  
## ✔ bmcite 0.3.0 ✔ panc8 3.0.2  
## ✔ bonemarrowref 1.0.0 ✔ pbmc3k 3.1.4  
## ✔ cbmc 3.1.4 ✔ pbmcMultiome 0.1.2  
## ✔ celegans.embryo 0.1.0 ✔ pbmcref 1.0.0  
## ✔ hcabm40k 3.0.0 ✔ pbmcsca 3.0.0

## ────────────────────────────────────── Key ─────────────────────────────────────

## ✔ Dataset loaded successfully  
## ❯ Dataset built with a newer version of Seurat than installed  
## ❓ Unknown version of Seurat installed

library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(ggplot2)  
library(mclust)

## Package 'mclust' version 6.0.0  
## Type 'citation("mclust")' for citing this R package in publications.

library(pROC)

## Type 'citation("pROC")' for a citation.

##   
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

library(caret)

## Loading required package: lattice

## 2. Read data

Load the data from Human Bone Marrow Cells (HBMC). This dataset consists of 30,672 cells and 25 antibodies. We perform normalization to RNA and ADT count matrix and then screen out certain features for RNA.

#Load data  
bm=LoadData('bmcite')  
  
### Normalization, feature selection and get PCA components for RNA  
DefaultAssay(bm) <- 'RNA'  
bm <- NormalizeData(bm) %>% FindVariableFeatures() %>% ScaleData() %>% RunPCA()

## Centering and scaling data matrix

## PC\_ 1   
## Positive: TRBC1, LAT, CD8B, CCL5, KLRB1, IGKC, S100A12, GZMA, S100A8, S100A9   
## MS4A1, S100B, GNLY, CST7, TYROBP, KLRD1, RP11-291B21.2, NKG7, VCAN, CD14   
## IGLC2, CCL4, AC092580.4, FCN1, IGLC3, PRF1, RBP7, SERPINA1, DUSP2, JUN   
## Negative: KIAA0101, TYMS, KLF1, KCNH2, FAM178B, APOC1, CNRIP1, CENPU, GATA1, BIRC5   
## CENPF, EPCAM, CKS2, RP11-620J15.3, TUBA1B, TFR2, CA1, HMGA1, STMN1, HIST1H4C   
## CDT1, AHSP, TOP2A, TK1, GFI1B, TUBB, MKI67, NME4, SMIM1, TMEM56   
## PC\_ 2   
## Positive: RPL3, RPS3, RPS18, RPS5, RPS4X, RPSA, RPS12, RPS23, RPS2, EEF1B2   
## RPL4, LDHB, NPM1, RPS17, RPLP0, TRBC1, LAT, RPL7A, GYPC, HSPA8   
## CD8B, KLRB1, CCL5, HNRNPA1, PEBP1, RPL37A, MYC, NUCB2, SOD1, CD79A   
## Negative: LYZ, FCN1, CST3, TYROBP, S100A9, LST1, S100A8, CSTA, MNDA, VCAN   
## LGALS1, AIF1, S100A12, CFD, SERPINA1, FCER1G, MS4A6A, FOS, S100A6, CD14   
## LGALS2, FTH1, GAPDH, ANXA2, CD36, CPVL, RBP7, HLA-DRA, LINC01272, H3F3A   
## PC\_ 3   
## Positive: CD74, HLA-DRA, HLA-DPB1, HLA-DPA1, IGLL1, ITM2C, SMIM24, RPS23, FABP5, HLA-DRB5   
## TCF4, RPS18, PLD4, HLA-DQA1, RPS4X, SOX4, SPINK2, RPLP1, KIAA0125, HLA-DQB1   
## HLA-DRB1, RPS2, PRSS57, IRF8, CDCA7, C1QTNF4, STMN1, BCL11A, NREP, RPS24   
## Negative: GYPA, ALAS2, SLC4A1, AHSP, HBM, GYPB, CA2, HBD, RHAG, CA1   
## HBA1, TMEM56, SELENBP1, HBB, HBA2, MYL4, HEMGN, HMBS, RHCE, HBQ1   
## DMTN, RFESD, SPTA1, FECH, KLF1, ANK1, CTSE, SMIM1, TSPO2, SLC25A37   
## PC\_ 4   
## Positive: CD79A, MS4A1, CD79B, CD74, HLA-DRA, HLA-DPB1, HLA-DPA1, IGHD, HLA-DQA1, HLA-DQB1   
## BANK1, VPREB3, SPIB, TCL1A, HLA-DRB5, FAM129C, LINC00926, IGHM, IGKC, HLA-DRB1   
## TNFRSF13C, JCHAIN, TSPAN13, IRF8, FCER2, CD24, BLK, CD22, GNG7, FCRLA   
## Negative: NKG7, CST7, PRSS57, GNLY, GZMA, KLRB1, CCL5, LDHB, KLRD1, CMC1   
## CYTL1, PRF1, EGFL7, LYAR, TRBC1, KLRF1, GAPDH, S100A6, NGFRAP1, LAT   
## RPS3, S100B, FGFBP2, GZMH, GATA2, RPS24, NPW, CCL4, SMIM24, CDK6   
## PC\_ 5   
## Positive: RPS2, RPS18, RPS12, RPS23, RPL37A, RPLP1, RPL4, RPS5, RPS4X, CNRIP1   
## RPS17, APOC1, NPM1, MYC, EEF1B2, FAM178B, EPCAM, KCNH2, KLF1, RPLP0   
## RPL7A, CYTL1, MS4A1, LDHB, SMIM10, GATA1, APOE, RPS3, TFR2, RPL3   
## Negative: NKG7, GZMB, CST7, GNLY, GZMA, KLRD1, CMC1, KLRF1, PRF1, CCL4   
## CCL5, GZMH, CLIC3, FGFBP2, SPON2, C12orf75, TRDC, KLRB1, HOPX, XCL2   
## CD160, IL2RB, PLAC8, FCGR3A, TRGC2, KLRG1, DUSP2, RHOC, PLEK, LAIR2

### Normalization and feature selection for ADT  
DefaultAssay(bm) <- 'ADT'  
VariableFeatures(bm) <- rownames(bm[["ADT"]])  
bm <- NormalizeData(bm, normalization.method = 'CLR', margin = 2) %>%  
 ScaleData()

## Normalizing across cells

## Centering and scaling data matrix

## 3. Train the model

n.adt.pc = 25  
  
#Get RNA and ADT scaled matrix  
rna = t(bm@assays$RNA@scale.data)  
adt = t(bm@assays$ADT@scale.data)  
  
#Train the model  
model=lm(adt~rna)  
  
#Get ADT residuals  
adt.reduce=model$residuals  
  
#Get RNA components  
rna.emb = bm@reductions$pca@cell.embeddings[,1:30]  
colnames(adt.reduce)=c(1:25)  
colnames=colnames(adt.reduce)  
colnames=paste("apca\_",colnames,sep="")  
colnames(adt.reduce)=colnames  
  
#Integrate ADT residuals and RNA for clustering  
integrate.data = t(scale(data.frame(cbind(rna.emb,adt.reduce))))

## 4. Clustering

The evaluation of the clustering performance is presented by calculating the adjusted rand index between the clustering result and the manually annotated labels.

#Use ADT residuals and RNA for clustering  
bm.integrate = bm  
bm.integrate@assays$integrate=CreateAssayObject(data = integrate.data)

## Warning: Feature names cannot have underscores ('\_'), replacing with dashes  
## ('-')

DefaultAssay(bm.integrate) <- 'integrate'  
bm.integrate@assays$integrate@key = "integrate\_"  
VariableFeatures(bm.integrate) <- rownames(bm.integrate[["integrate"]])  
  
bm.integrate@reductions$integrateRed = CreateDimReducObject(embeddings = t(as.matrix(integrate.data)),  
 assay = "integrate",key = 'integrateRed\_')  
  
bm.integrate <- RunUMAP(bm.integrate, reduction = 'integrateRed',dims = 1:50,  
 reduction.name = "integrate.umap", reduction.key = "integrateUMAP\_")

## Warning: The default method for RunUMAP has changed from calling Python UMAP via reticulate to the R-native UWOT using the cosine metric  
## To use Python UMAP via reticulate, set umap.method to 'umap-learn' and metric to 'correlation'  
## This message will be shown once per session

## 09:56:09 UMAP embedding parameters a = 0.9922 b = 1.112

## 09:56:09 Read 30672 rows and found 50 numeric columns

## 09:56:09 Using Annoy for neighbor search, n\_neighbors = 30

## 09:56:09 Building Annoy index with metric = cosine, n\_trees = 50

## 0% 10 20 30 40 50 60 70 80 90 100%

## [----|----|----|----|----|----|----|----|----|----|

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|  
## 09:56:11 Writing NN index file to temp file /var/folders/ms/tsd08\_hj7nj8327733dr3h380000gn/T//RtmpHmavcm/file849774ada4ee  
## 09:56:11 Searching Annoy index using 1 thread, search\_k = 3000  
## 09:56:21 Annoy recall = 100%  
## 09:56:21 Commencing smooth kNN distance calibration using 1 thread with target n\_neighbors = 30  
## 09:56:22 Initializing from normalized Laplacian + noise (using irlba)  
## 09:56:23 Commencing optimization for 200 epochs, with 1291372 positive edges  
## 09:56:43 Optimization finished

bm.integrate <- FindNeighbors(bm.integrate, reduction = 'integrateRed',dims = 1:50  
 , assay = 'integrate',graph.name = "intg\_graph",k.param =20)

## Computing nearest neighbor graph  
## Computing SNN  
## Only one graph name supplied, storing nearest-neighbor graph only

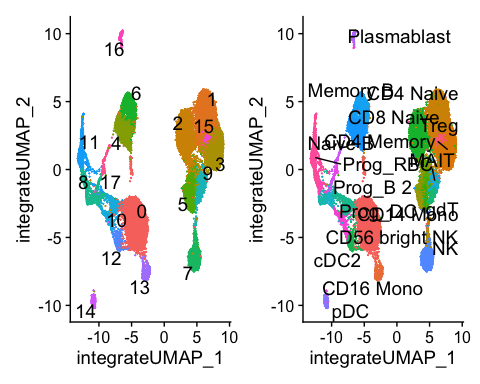
#Louvein algorithm for clustering  
bm.integrate <- FindClusters(bm.integrate , graph.name = "intg\_graph"  
 ,resolution = 1,method=3)

## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck  
##   
## Number of nodes: 30672  
## Number of edges: 279474  
##   
## Running Louvain algorithm...  
## Maximum modularity in 10 random starts: 0.8271  
## Number of communities: 74  
## Elapsed time: 1 seconds

## 56 singletons identified. 18 final clusters.

#Visualization  
p4 <- DimPlot(bm.integrate, reduction = 'integrate.umap', label = TRUE,  
 repel = TRUE, label.size = 5) + NoLegend() +ggtitle(NULL)  
p5 <- DimPlot(bm.integrate, reduction = 'integrate.umap', group.by = 'celltype.l2', label = TRUE,  
 repel = TRUE, label.size = 5) + NoLegend() +ggtitle(NULL)  
p4+p5

## Warning: ggrepel: 9 unlabeled data points (too many overlaps). Consider  
## increasing max.overlaps



adjustedRandIndex(bm.integrate$seurat\_clusters, bm.integrate$celltype.l2)

## [1] 0.8790843

## 5. Intepretability

Now we come to see how our model could detect, which RNAs and ADTs are differentially expressed to provide significant prediction power in clustering analysis. We perform logisitic regression in each cluster and examing the value of coefficients to determine additional prediction power for each variable.

#Find differentially expressed RNA in each cluster  
DefaultAssay(bm.integrate) <- 'RNA'  
bm.markers <- FindAllMarkers(bm.integrate, only.pos = TRUE, min.pct = 0.25, logfc.threshold = 0.75)

## Calculating cluster 0

## Calculating cluster 1

## Calculating cluster 2

## Calculating cluster 3

## Calculating cluster 4

## Calculating cluster 5

## Calculating cluster 6

## Calculating cluster 7

## Calculating cluster 8

## Calculating cluster 9

## Calculating cluster 10

## Calculating cluster 11

## Calculating cluster 12

## Calculating cluster 13

## Calculating cluster 14

## Calculating cluster 15

## Calculating cluster 16

## Calculating cluster 17

bm.markers %>%  
 group\_by(cluster) %>%  
 slice\_max(n = 2, order\_by = avg\_log2FC)

## # A tibble: 36 × 7  
## # Groups: cluster [18]  
## p\_val avg\_log2FC pct.1 pct.2 p\_val\_adj cluster gene   
## <dbl> <dbl> <dbl> <dbl> <dbl> <fct> <chr>   
## 1 0 4.49 0.986 0.338 0 0 S100A9   
## 2 0 4.36 0.981 0.349 0 0 S100A8   
## 3 0 1.17 0.672 0.339 0 1 NOSIP   
## 4 0 1.06 0.361 0.105 0 1 CCR7   
## 5 0 2.18 0.659 0.051 0 2 CD8B   
## 6 0 1.53 0.354 0.006 0 2 RP11-291B21.2  
## 7 0 1.54 0.756 0.251 0 3 IL7R   
## 8 0 1.52 0.872 0.342 0 3 IL32   
## 9 0 3.05 0.753 0.028 0 4 IGHD   
## 10 0 2.81 0.815 0.089 0 4 CD79A   
## # ℹ 26 more rows

marker\_index=list()  
  
for(i in 1:16){  
 marker\_index[[i]]=which(bm.markers$cluster==i-1)  
}

ADT=t(bm.integrate$ADT@scale.data)  
colnames(ADT)=noquote(paste('adt',colnames(ADT)))  
  
test\_adt=c()  
train\_adt=c()  
test\_rna=c()  
train\_rna=c()  
test\_adtrna=c()  
train\_adtrna=c()  
  
#Divide into training set and testing set  
separate <- sample(c(rep(0,21470), rep(1,9202)))  
train=ADT[separate==0,]  
test=ADT[separate==1,]  
   
cluster=rep(0,30672)  
cluster\_list=c()  
index\_list=c()  
model\_list=c()  
  
   
#Perform logistic regression on each cluster using only ADT information  
for(i in 1:16){  
 index\_list[[i]]=which(bm.integrate$seurat\_clusters==(i-1))  
 cluster[index\_list[[i]]]=1  
 cluster\_list[[i]]=cluster  
 cluster=rep(0,30672)  
   
   
 cluster\_train=cluster\_list[[i]][separate==0]  
 cluster\_test=cluster\_list[[i]][separate==1]  
   
 train=cbind(train,cluster\_train)  
 train=as.data.frame(train)  
 test=cbind(test,cluster\_test)  
 test=as.data.frame(test)  
   
 temp <- (0.5\*train$cluster\_train\*(1/sum(train$cluster\_train== 1) - 1/sum(train$cluster\_train == 0)) + 0.5/sum(train$cluster\_train == 0))  
 dat\_weights = temp\*length(train$cluster\_train)  
   
 g1=glm(cluster\_train~.,data=train, weights=dat\_weights, family = 'binomial')  
   
 model\_list[[i]]=g1  
   
 predict=ifelse(g1$fitted.values>0.5,1,0)  
 c1=confusionMatrix(data=factor(predict, levels = c(1,0)), reference = factor(cluster\_train, levels = c(1,0)))  
  
   
   
   
 predict2=predict(g1,newdata=test,'response')  
 predict3=ifelse(predict2>0.5,1,0)  
 c2=confusionMatrix(data=factor(predict3, levels = c(1,0)), reference = factor(cluster\_test, levels = c(1,0)))  
   
   
 train=ADT[separate==0,]  
 test=ADT[separate==1,]  
}  
   
 cluster=rep(0,30672)  
 cluster\_list=c()  
 index\_list=c()  
 model\_list2=c()  
   
#Perform logistic regression on each cluster using only RNA information  
for(i in 1:16){  
 pbmc <- CreateSeuratObject(counts = bm.integrate@assays$RNA@counts[bm.markers$gene[marker\_index[[i]]],])  
 pbmc <- NormalizeData(pbmc)%>% ScaleData()  
   
 rna.emb=t(pbmc@assays$RNA@scale.data)  
 colnames(rna.emb) <- noquote(paste('rna',colnames(rna.emb)))  
   
  
 separate <- sample(c(rep(0,21470), rep(1,9202)))  
 train=rna.emb[separate==0,]  
 test=rna.emb[separate==1,]  
   
 index\_list[[i]]=which(bm.integrate$seurat\_clusters==(i-1))  
 cluster[index\_list[[i]]]=1  
 cluster\_list[[i]]=cluster  
 cluster=rep(0,30672)  
   
 cluster\_train=cluster\_list[[i]][separate==0]  
 cluster\_test=cluster\_list[[i]][separate==1]  
   
 train=cbind(train,cluster\_train)  
 train=as.data.frame(train)  
 test=cbind(test,cluster\_test)  
 test=as.data.frame(test)  
   
 temp <- (0.5\*train$cluster\_train\*(1/sum(train$cluster\_train== 1) - 1/sum(train$cluster\_train == 0)) + 0.5/sum(train$cluster\_train == 0))  
 dat\_weights = temp\*length(train$cluster\_train)  
   
 g1=glm(cluster\_train~.,data=train, weights=dat\_weights, family = 'binomial')  
   
 model\_list2[[i]]=g1  
  
   
 predict=ifelse(g1$fitted.values>0.5,1,0)  
 c1=confusionMatrix(data=factor(predict, levels = c(1,0)), reference = factor(cluster\_train, levels = c(1,0)))  
   
 predict2=predict(g1,newdata=test,'response')  
 predict3=ifelse(predict2>0.5,1,0)  
 c2=confusionMatrix(data=factor(predict3, levels = c(1,0)), reference = factor(cluster\_test, levels = c(1,0)))  
   
 train=rna.emb[separate==0,]  
 test=rna.emb[separate==1,]  
   
}

## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix

cluster=rep(0,30672)  
 cluster\_list=c()  
 index\_list=c()  
 model\_list3=c()  
   
#Perform logistic regression on each cluster using RNA and ADT information  
for(i in 1:16){  
  
 pbmc <- CreateSeuratObject(counts = bm.integrate@assays$RNA@counts[bm.markers$gene[marker\_index[[i]]],])  
 pbmc <- NormalizeData(pbmc) %>% ScaleData()  
   
 rna.emb0=t(pbmc@assays$RNA@scale.data)  
   
 colnames(rna.emb0) <- noquote(paste('rna',colnames(rna.emb0)))  
   
 rna.emb = cbind(rna.emb0,ADT)  
   
  
 separate <- sample(c(rep(0,21470), rep(1,9202)))  
 train=rna.emb[separate==0,]  
 test=rna.emb[separate==1,]  
   
   
   
   
 index\_list[[i]]=which(bm.integrate$seurat\_clusters==(i-1))  
 cluster[index\_list[[i]]]=1  
 cluster\_list[[i]]=cluster  
 cluster=rep(0,30672)  
   
   
 cluster\_train=cluster\_list[[i]][separate==0]  
 cluster\_test=cluster\_list[[i]][separate==1]  
   
 train=cbind(train,cluster\_train)  
 train=as.data.frame(train)  
 test=cbind(test,cluster\_test)  
 test=as.data.frame(test)  
   
 temp <- (0.5\*train$cluster\_train\*(1/sum(train$cluster\_train== 1) - 1/sum(train$cluster\_train == 0)) + 0.5/sum(train$cluster\_train == 0))  
 dat\_weights = temp\*length(train$cluster\_train)  
   
 g1=glm(cluster\_train~.,data=train, weights=dat\_weights, family = 'binomial')  
   
 model\_list3[[i]]=g1  
   
 predict=ifelse(g1$fitted.values>0.5,1,0)  
 c1=confusionMatrix(data=factor(predict, levels = c(1,0)), reference = factor(cluster\_train, levels = c(1,0)))  
  
 predict2=predict(g1,newdata=test,'response')  
 predict3=ifelse(predict2>0.5,1,0)  
 c2=confusionMatrix(data=factor(predict3, levels = c(1,0)), reference = factor(cluster\_test, levels = c(1,0)))  
   
 train=rna.emb[separate==0,]  
 test=rna.emb[separate==1,]  
 }

## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix  
## Centering and scaling data matrix

#Find coefficients and p-value for Treg cluster  
model\_treg\_both=model\_list[[16]]  
summary=summary(model\_treg\_both)  
pvalue=summary$coefficients[,4]  
coefficients=data.frame(summary$coefficients[,1])  
predictors=rownames(coefficients)[-1]  
coefficients=as.data.frame(coefficients[-1,])  
coefficients <- as.data.frame(cbind(predictors, coefficients))  
coefficients <- coefficients %>% mutate(sign=factor(ifelse(.$`coefficients[-1, ]`>0,'positive','negative'), levels=c('positive','negative')))  
coefficients=data.frame(cbind(coefficients, pvalue[-1]))  
coefficients <- coefficients %>% mutate(abs.value=ifelse(.$pvalue..1.<0.01, abs(.$coefficients..1...), NA))  
coefficients <- subset(coefficients, coefficients$pvalue..1.<0.01)  
coefficients <- cbind(coefficients, 'Treg')  
colnames(coefficients)[6]='Cluster'  
coefficients1 <- coefficients  
  
#Find coefficients and p-value for CD4 Naive cluster  
model\_cd4n\_both=model\_list[[2]]  
summary=summary(model\_cd4n\_both)  
pvalue=summary$coefficients[,4]  
coefficients=data.frame(summary$coefficients[,1])  
predictors=rownames(coefficients)[-1]  
coefficients=as.data.frame(coefficients[-1,])  
coefficients <- as.data.frame(cbind(predictors, coefficients))  
coefficients <- coefficients %>% mutate(sign=factor(ifelse(.$`coefficients[-1, ]`>0,'positive','negative'), levels=c('positive','negative')))  
coefficients=data.frame(cbind(coefficients, pvalue[-1]))  
coefficients <- coefficients %>% mutate(abs.value=ifelse(.$pvalue..1.<0.01, abs(.$coefficients..1...), NA))  
coefficients <- cbind(coefficients, 'CD4 Naive')  
colnames(coefficients)[6]='Cluster'  
coefficients2 <- coefficients  
  
#Find coefficients and p-value for CD4 Memory cluster  
model\_cd4m\_both=model\_list[[4]]  
summary=summary(model\_cd4m\_both)  
pvalue=summary$coefficients[,4]  
pvalue[-1]

## `adt CD11a` `adt CD11c` `adt CD123` `adt CD127-IL7Ra`   
## 3.176129e-02 9.267629e-08 4.643550e-11 3.763791e-23   
## `adt CD14` `adt CD16` `adt CD161` `adt CD19`   
## 4.170361e-08 6.653955e-01 8.983927e-32 3.181544e-02   
## `adt CD197-CCR7` `adt CD25` `adt CD27` `adt CD278-ICOS`   
## 1.703057e-24 2.091327e-04 6.592479e-11 1.811856e-07   
## `adt CD28` `adt CD3` `adt CD34` `adt CD38`   
## 7.542126e-03 1.519091e-59 3.896030e-08 8.944946e-41   
## `adt CD4` `adt CD45RA` `adt CD45RO` `adt CD56`   
## 1.098058e-38 1.035618e-118 2.437914e-12 5.665747e-17   
## `adt CD57` `adt CD69` `adt CD79b` `adt CD8a`   
## 9.029198e-13 4.326541e-06 6.259641e-03 9.705457e-02   
## `adt HLA.DR`   
## 1.017759e-02

coefficients=data.frame(summary$coefficients[,1])  
predictors=rownames(coefficients)[-1]  
coefficients=as.data.frame(coefficients[-1,])  
coefficients <- as.data.frame(cbind(predictors, coefficients))  
coefficients <- coefficients %>% mutate(sign=factor(ifelse(.$`coefficients[-1, ]`>0,'positive','negative'), levels=c('positive','negative')))  
coefficients=data.frame(cbind(coefficients, pvalue[-1]))  
coefficients <- coefficients %>% mutate(abs.value=ifelse(.$pvalue..1.<0.01, abs(.$coefficients..1...), NA))  
coefficients <- cbind(coefficients, 'CD4 Memory')  
colnames(coefficients)[6]='Cluster'  
coefficients3 <- coefficients  
  
#Plot significant coefficients values  
coefficients=as.data.frame(rbind(coefficients1, coefficients2, coefficients3))  
ggplot(coefficients, aes(x=Cluster,y=predictors, colour=sign, size=abs.value)) + geom\_point()+labs(x=NULL)

