Simulation data generation scenario 1

Yunshan Duan

2024-03-14

Load library

```
library(ggplot2); theme_set(theme_bw())
library(dplyr)
library(Seurat)
library(scDesign2)
library(ggpubr)

data_dir <- "../data/simulation_sc1"

# colors
pal <- c("#ffb6db", "#33A02C", "#b66dff", "#FEC44F", "#41B6C4", "#8E0152", "#0868AC", "#807DBA", "#E7298
         "#00441B", "#525252", "#4D9221", "#8B5742", "#D8DAEB", "#7cdd2d", "#980043", "#8C96C6", "#EC70
         "#FDAE61", "#1D91C0", "#A6DBA0", "#4292C6", "#BF812D", "#01665E", "#41AB5D", "#FE9929", "#2525
names(pal) <- 1:30
```

Preprocess PBMC data set

```
if (file.exists(paste0(data_dir, "/pbmc3k_final.rds"))) {
  pbmc <- readRDS(paste0(data_dir, "/pbmc3k_final.rds"))
} else {
  # Load the PBMC dataset
  pbmc.data <- Read10X("../data/filtered_gene_bc_matrices/hg19/")
  # Initialize the Seurat object with the raw (non-normalized data).
  pbmc <- CreateSeuratObject(counts = pbmc.data, project = "pbmc3k", min.cells = 3, min.features = 200)
  pbmc
  # The [[ operator can add columns to object metadata. This is a great place to stash QC stats
  pbmc[["percent.mt"]] <- PercentageFeatureSet(pbmc, pattern = "^MT-")
  # Visualize QC metrics as a violin plot
  VlnPlot(pbmc, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
  # FeatureScatter is typically used to visualize feature-feature relationships, but can be used
  # for anything calculated by the object, i.e. columns in object metadata, PC scores etc.

  plot1 <- FeatureScatter(pbmc, feature1 = "nCount_RNA", feature2 = "percent.mt")
  plot2 <- FeatureScatter(pbmc, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
  plot1 + plot2
  pbmc <- subset(pbmc, subset = nFeature_RNA > 200 & nFeature_RNA < 2500 & percent.mt < 5)
  pbmc <- NormalizeData(pbmc, normalization.method = "LogNormalize", scale.factor = 10000)
  pbmc <- NormalizeData(pbmc)
  pbmc <- FindVariableFeatures(pbmc, selection.method = "vst", nfeatures = 2000)

  # Identify the 10 most highly variable genes
```

```r
top10 <- head(VariableFeatures(pbmc), 10)

# plot variable features with and without labels
plot1 <- VariableFeaturePlot(pbmc)
plot2 <- LabelPoints(plot = plot1, points = top10, repel = TRUE)
plot1 + plot2

all.genes <- rownames(pbmc)
pbmc <- ScaleData(pbmc, features = all.genes)

pbmc <- RunPCA(pbmc, features = VariableFeatures(object = pbmc))
# Examine and visualize PCA results a few different ways
print(pbmc[["pca"]], dims = 1:5, nfeatures = 5)

VizDimLoadings(pbmc, dims = 1:2, reduction = "pca")

DimPlot(pbmc, reduction = "pca")
DimHeatmap(pbmc, dims = 1, cells = 500, balanced = TRUE)

# NOTE: This process can take a long time for big datasets, comment out for expediency. More
# approximate techniques such as those implemented in ElbowPlot() can be used to reduce
# computation time
pbmc <- JackStraw(pbmc, num.replicate = 100)
pbmc <- ScoreJackStraw(pbmc, dims = 1:20)

JackStrawPlot(pbmc, dims = 1:15)
pdf("./figures/elbowplot.pdf", width = 6, height = 6)
ElbowPlot(pbmc)
dev.off()

pbmc <- FindNeighbors(pbmc, dims = 1:10)
pbmc <- FindClusters(pbmc, resolution = 0.5)

# Look at cluster IDs of the first 5 cells
head(Idents(pbmc), 5)

# If you haven't installed UMAP, you can do so via reticulate::py_install(packages =
# 'umap-learn')
pbmc <- RunUMAP(pbmc, dims = 1:10)

# note that you can set `label = TRUE` or use the LabelClusters function to help label
# individual clusters
DimPlot(pbmc, reduction = "umap")
saveRDS(pbmc, file = "./data/pbmc_tutorial.rds")

# find all markers of cluster 2
cluster2.markers <- FindMarkers(pbmc, ident.1 = 2, min.pct = 0.25)
head(cluster2.markers, n = 5)

# find all markers distinguishing cluster 5 from clusters 0 and 3
cluster5.markers <- FindMarkers(pbmc, ident.1 = 5, ident.2 = c(0, 3), min.pct = 0.25)
head(cluster5.markers, n = 5)
```

```
  # find markers for every cluster compared to all remaining cells, report only the positive
  # ones
  pbmc.markers <- FindAllMarkers(pbmc, only.pos = TRUE, min.pct = 0.25, logfc.threshold = 0.25)
  pbmc.markers %>%
    group_by(cluster) %>%
    slice_max(n = 2, order_by = avg_log2FC)

  cluster0.markers <- FindMarkers(pbmc, ident.1 = 0, logfc.threshold = 0.25, test.use = "roc", only.pos

  VlnPlot(pbmc, features = c("MS4A1", "CD79A"))

  # you can plot raw counts as well
  VlnPlot(pbmc, features = c("NKG7", "PF4"), slot = "counts", log = TRUE)

  FeaturePlot(pbmc, features = c("MS4A1", "GNLY", "CD3E", "CD14", "FCER1A", "FCGR3A", "LYZ", "PPBP",
                                 "CD8A"))

  pbmc.markers %>%
    group_by(cluster) %>%
    top_n(n = 10, wt = avg_log2FC) -> top10
  DoHeatmap(pbmc, features = top10$gene) + NoLegend()

  new.cluster.ids <- c("Naive CD4 T", "CD14+ Mono", "Memory CD4 T", "B", "CD8 T", "FCGR3A+ Mono",
                       "NK", "DC", "Platelet")
  names(new.cluster.ids) <- levels(pbmc)
  pbmc <- RenameIdents(pbmc, new.cluster.ids)
  DimPlot(pbmc, reduction = "umap", label = TRUE, pt.size = 0.5) + NoLegend()

  saveRDS(pbmc, file = paste0(data_dir, "/pbmc3k_final.rds"))
}
```

Simulation single cell data given proportions of true cell types

```
pbmc <- RunPCA(pbmc, npcs = 10)

# count matrix
count <- as.matrix(pbmc@assays$RNA@counts)
dim(count)
```

```
## [1] 13714  2638
```

```
# [1] 13714  2638

# feature matrix after PCA (dim reduction)
gene <- pbmc@reductions$pca@cell.embeddings
dim(gene)
```

```
## [1] 2638   10
```

```
# [1] 2638   10

# umap embeddings
gene_umap <- pbmc@reductions$umap@cell.embeddings
dim(gene_umap)
```

```
## [1] 2638    2
```

```
# [1] 2638    2

# cell names
cell_names <- colnames(count)

new.cluster.ids <- c("Naive CD4 T", "CD14+ Mono", "Memory CD4 T", "B", "CD8 T", "FCGR3A+ Mono",
                     "NK", "DC", "Platelet")

cell_types <- pbmc@meta.data$seurat_clusters
cell_types_char <- new.cluster.ids[cell_types]

cell_selected <- cell_names[which(cell_types_char %in% c("Naive CD4 T", "Memory CD4 T", "CD8 T", "NK") &
                                    gene_umap[,1] < 5 & gene_umap[,2] < 0)]
length(cell_selected)
```

```
## [1] 1601
```

```
# [1] 1601

pbmc_sel <- pbmc[,cell_selected]

##################
# count matrix
count <- as.matrix(pbmc_sel@assays$RNA@counts)
dim(count)
```

```
## [1] 13714  1601
```

```
# [1] 13714  1601

# feature matrix after PCA (dim reduction)
gene <- pbmc_sel@reductions$pca@cell.embeddings
dim(gene)
```

```
## [1] 1601   10
```

```
# [1] 1601   10

# umap embeddings
gene_umap <- pbmc_sel@reductions$umap@cell.embeddings
dim(gene_umap)
```

```
## [1] 1601    2
```

```
# [1] 1601    2

# cell names
cell_names <- colnames(count)

cell_types <- pbmc_sel@meta.data$seurat_clusters
cell_types_str <- new.cluster.ids[cell_types]
```

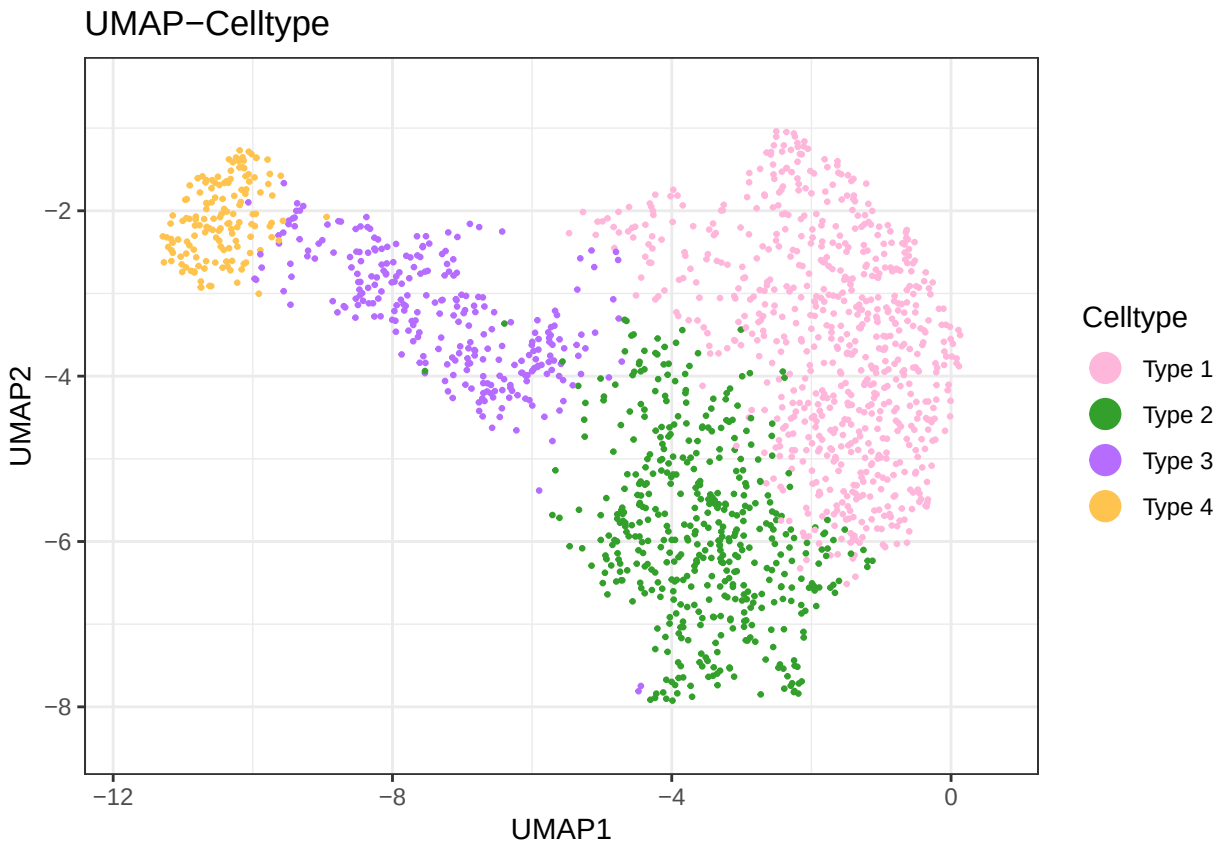
Simulation conditions for single cell data and save the simulated data set

```
set.seed(1999)

ncells <- nrow(gene)
celltype_sim <- cell_types
celltype_sim[which(cell_types == 0)] <- 1
celltype_sim[which(cell_types == 2)] <- 2
celltype_sim[which(cell_types == 4)] <- 3
celltype_sim[which(cell_types == 6)] <- 4
condition_sim <- rep(NA, ncells)
for (ii in 4) {
  ind <- which(celltype_sim == ii)
  ind_LO <- sample(ind, round(length(ind)*0.52), replace = F)
  ind_YO <- ind[!ind %in% ind_LO]
  condition_sim[ind_LO] <- "B"
  condition_sim[ind_YO] <- "A"
}
for (ii in c(1)) {
  ind <- which(celltype_sim == ii)
  ind_LO <- sample(ind, round(length(ind)*0.01), replace = F)
  ind_YO <- ind[!ind %in% ind_LO]
  condition_sim[ind_LO] <- "B"
  condition_sim[ind_YO] <- "A"
}
for (ii in c(2:3)) {
  ind <- which(celltype_sim == ii)
  ind_LO <- sample(ind, round(length(ind)*0.99), replace = F)
  ind_YO <- ind[!ind %in% ind_LO]
  condition_sim[ind_LO] <- "B"
  condition_sim[ind_YO] <- "A"
}

celltype_sim_str <- paste0("Type ", celltype_sim)
sim_data <- pbmc_sel
sim_data@meta.data$condition_sim <- condition_sim
sim_data@meta.data$celltype_sim <- celltype_sim
sim_data@meta.data$celltype_sim_str <- celltype_sim
sim_data@meta.data$cell_types_str <- cell_types_str

celltype_CLE <- rep(NA, length(celltype_sim))
celltype_CLE[which(celltype_sim == 1)] <- "A"
celltype_CLE[which(celltype_sim %in% c(2,3))] <- "B"
celltype_CLE[which(celltype_sim %in% c(4))] <- "C"

sim_data@meta.data$celltype_CLE <- celltype_CLE

saveRDS(sim_data, file = paste0(data_dir, "/sim_data.rds"))
```

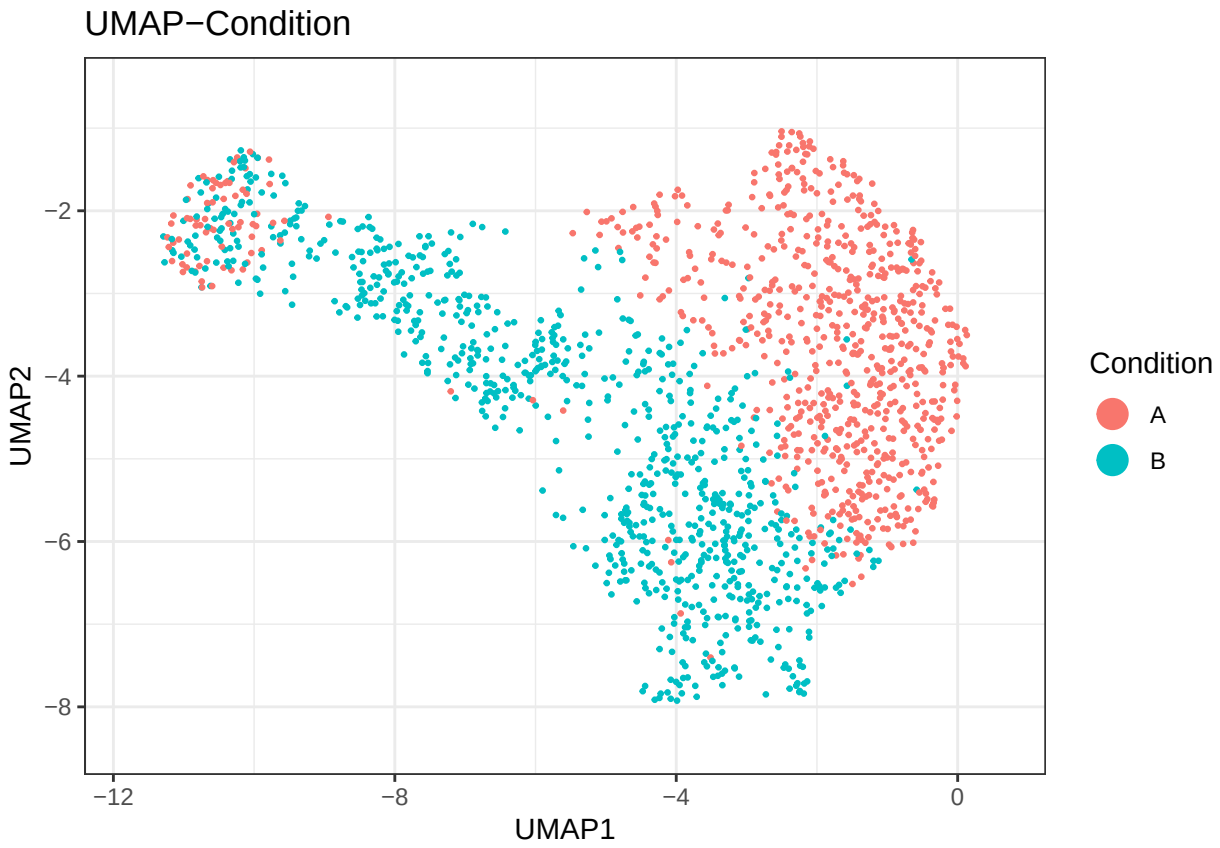
Visualization of the simulated data

```
count <- as.matrix(sim_data@assays$RNA@counts)
gene <- sim_data@reductions$pca@cell.embeddings
```

```r
gene_umap <- sim_data@reductions$umap@cell.embeddings
condition <- sim_data@meta.data$condition_sim
cell_names <- colnames(count)
celltype_sim <- sim_data@meta.data$celltype_sim
celltype_sim_string <- sim_data@meta.data$celltype_sim_string
celltype_sim_string <- paste0("Type ", celltype_sim)

types <- paste0("Type ", 1:4)

#####################################

pointsize <- 0.5

df <- data.frame(PCA1 = gene[,1], PCA2 = gene[,2],
                 UMAP1 = gene_umap[,1], UMAP2 = gene_umap[,2],
                 Condition = condition,
                 Celltype = celltype_sim_string)

xrange <- c(min(df$UMAP1) - 0.5 , max(df$UMAP1) + 0.5)
yrange <- c(min(df$UMAP2) - 0.5 , max(df$UMAP2) + 0.5)

pal0 <- pal[1:length(types)]
names(pal0) <- types

# plot
p1 <- ggplot(df, aes(x = UMAP1, y = UMAP2, color = Celltype)) +
  geom_point(size = pointsize) +
  guides(colour=guide_legend(override.aes=list(size = 5))) +
  scale_color_manual(values=c(pal0)) +
  ylim(yrange) + xlim(xrange) +
  ggtitle("UMAP-Celltype")
p1
```

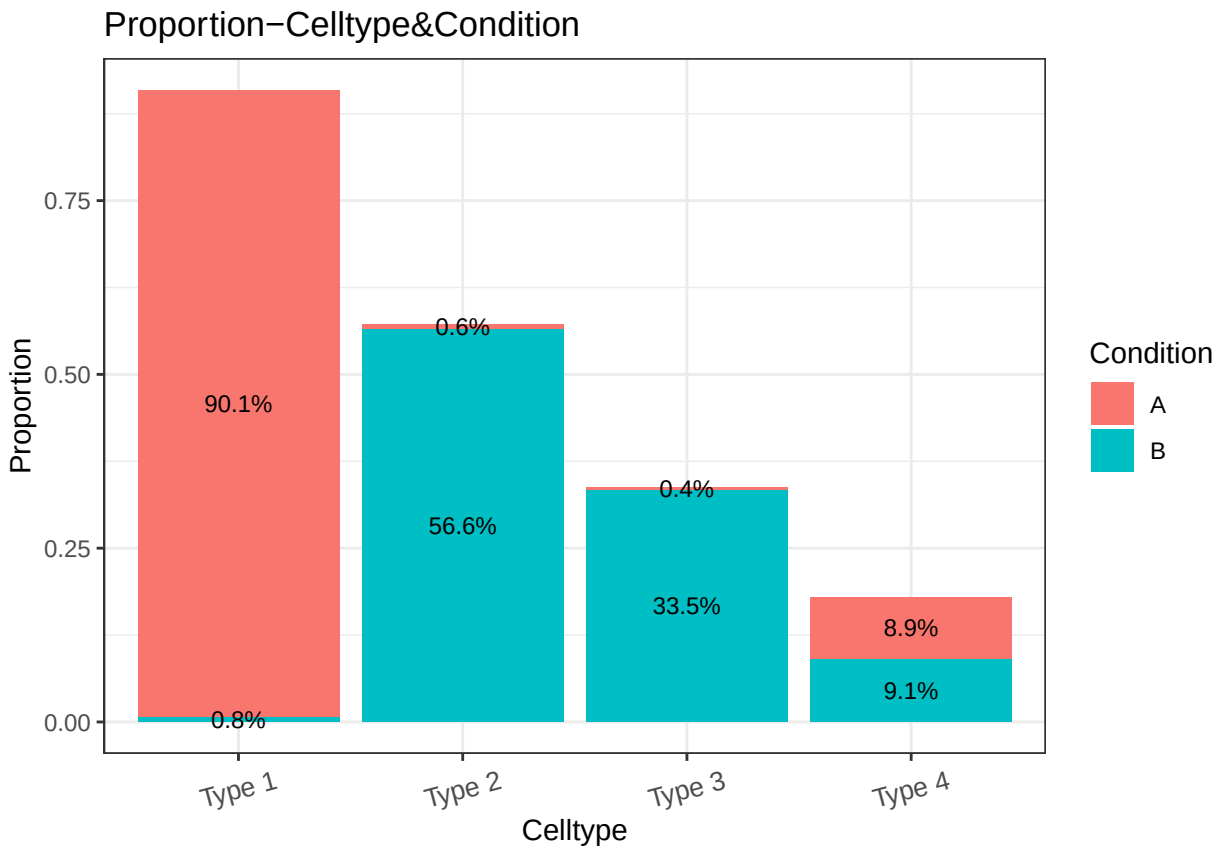



```
# plot
p2 <- ggplot(df, aes(x = UMAP1, y = UMAP2, color = Condition)) +
  geom_point(size = pointsize) +
  guides(colour=guide_legend(override.aes=list(size = 5))) +
  ylim(yrange) + xlim(xrange) +
  ggtitle("UMAP-Condition")
p2
```

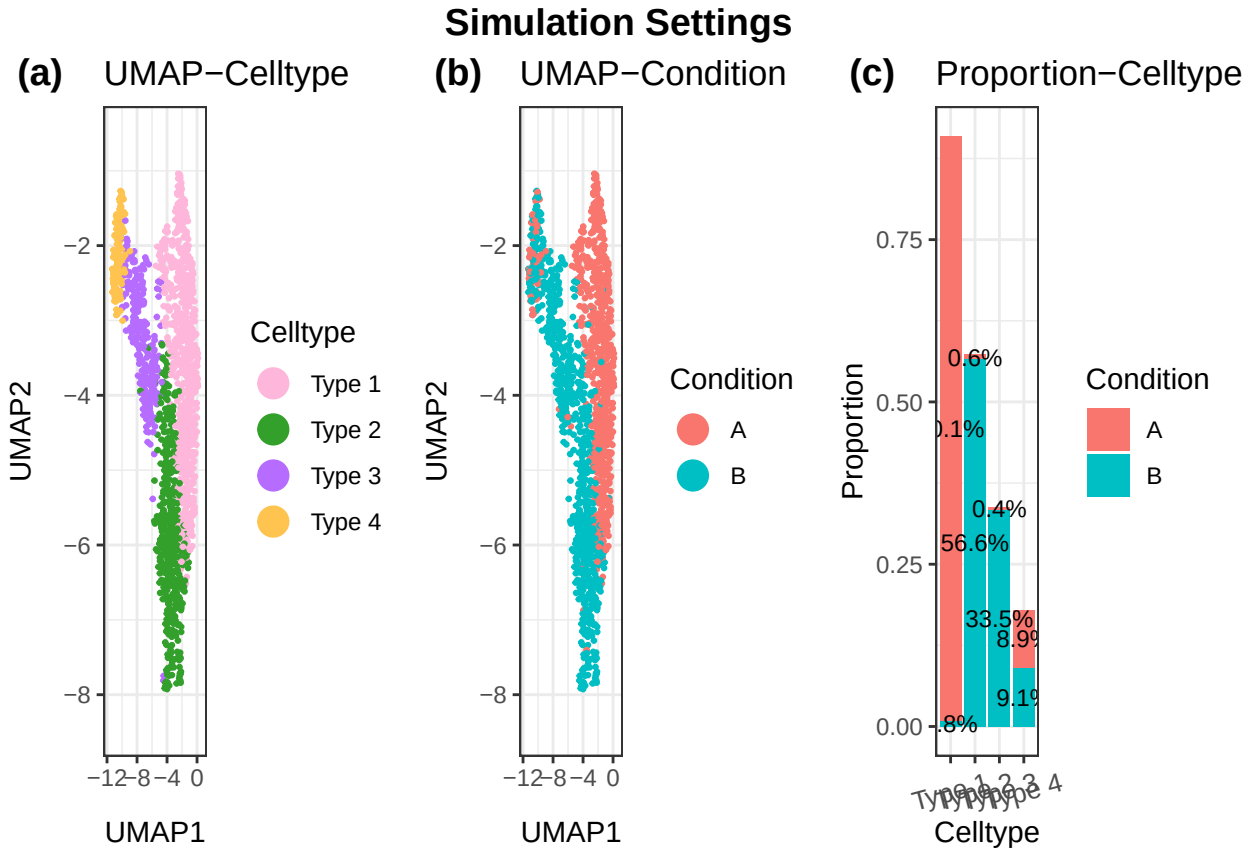



```
# data

LOCRC_cells <- cell_names[which(condition == "B")]
y0 <- as.matrix(gene[LOCRC_cells, ])
YOCRC_cells <- cell_names[which(condition == "A")]
y1 <- as.matrix(gene[YOCRC_cells, ])

# prop of EO/LO in the cell types
num_types <- length(types)
num_cells_LO <- num_cells_YO <- num_cells <- rep(0, num_types)
names(celltype_sim_string) <- cell_names
for (ii in 1:num_types) {
  num_cells_LO[ii] <- length(which(celltype_sim_string[LOCRC_cells] == types[ii]))
  num_cells_YO[ii] <- length(which(celltype_sim_string[YOCRC_cells] == types[ii]))
  num_cells[ii] <- length(which(celltype_sim_string == types[ii]))
}

## proportion of celltypes within condition
prop_cells <- c(num_cells_LO/sum(num_cells_LO), num_cells_YO/sum(num_cells_YO))
# plot
df_bar <- data.frame(Condition = c(rep("B" , num_types), rep("A" , num_types) ),
                     Celltype = rep(types , 2),
                     Proportion = prop_cells)

cluster_names <- types
# Grouped bar plot
p3 <- ggplot(df_bar, aes(fill=Condition, y=Proportion, x=Celltype)) +
```

```
  geom_bar(stat="identity") +
  scale_x_discrete(limits = cluster_names) +
  geom_text(aes(label=paste0(sprintf("%1.1f", prop_cells*100),"%")), position = position_stack(vjust = 
  theme(axis.text.x = element_text(angle = 15, vjust = 0.5, hjust=0.5, size = 10)) +
  ggtitle("Proportion-Celltype&Condition")
p3
```




```
plot <- ggarrange(p1, p2, p3, ncol = 3, nrow = 1, align = "hv",
                  labels = c("(a)", "(b)", "(c)"))

annotate_figure(plot, top = text_grob("Simulation Settings", face = "bold", size = 14))
```




```
ggsave(filename = paste0(data_dir, "/figures/sim_setting.pdf"), width = 11, height = 3)
```