Data Science Research Project - Notes

Table of Contents

[Outline 2](#_Toc78005986)

[Week 4 (22/03 – 28/03) 3](#_Toc78005987)

[Week 5 (29/03 – 04/04) 4](#_Toc78005988)

[Week 6 (05/04 – 11/04) 5](#_Toc78005989)

[Week7 (19/04) 6](#_Toc78005990)

[25/06 Meeting with Belinda 20](#_Toc78005991)

[Week 28/06 21](#_Toc78005992)

[28/06 Group Meeting 21](#_Toc78005993)

[02/07 Meeting with Belinda 21](#_Toc78005994)

[Code 21](#_Toc78005995)

[Week 05/07 24](#_Toc78005996)

[05/07 Meeting with Belinda 24](#_Toc78005997)

[Code 24](#_Toc78005998)

[Week 19/07 25](#_Toc78005999)

[code 25](#_Toc78006000)

[23/07 Meeting with Belinda 26](#_Toc78006001)

Timeline

Description automatically generated

# Outline

1. (go over the clustering steps) get the clusters

2. marker gene analysis -> the cell type for each cell cluster (to see the genes upregulated for this cluster compared to all other clusters )

3. gene set testing

Overview of single cell

We are working on marker gene analysis & gene set testing, specially on annotating clusters

Problems: p-values are optimistic (cells within an individual are correlated)

We will try to solve it using permutations

steps:

1. Go through analysis steps on pbmc dataset
2. Explore different test statistics for marker gene analysis,

one-side tests / one-side tests with threshold

1. Explore different test statistics for gene set testing, start with Wilcoxon signed-rank test

Look at the written software

1. How can we do permutation tests that account for different samples?

(block permutation)

Shuffling cell type labels within an individual

# Week 4 (22/03 – 28/03)

# Week 5 (29/03 – 04/04)

# Week 6 (05/04 – 11/04)

# Week7 (19/04)

pmbc data, extract individual “ 689\_690 ”

using Limma => dim(obs.tstat) [1] 9145 8

time: (using t-statistic from treat.all(lfc=0.5)) -> not valid

user system elapsed

1987.212 77.294 2065.082

Using moderated-t test statistic (from ebayes$t)

How can I speed up the process?

What kind of metrics can we use to evaluate the calculated p-values (to set number of permutations / select test-statistics)?

Feedback (Meeting 2 with Alicia & Belinda)

1. Try to do the permutation across all individuals, to allow more cells to DC cell type

2. Vis suggestions:

* A histogram comparing pvalues from Limma Vs p-values from permutation
* A QQ plot instead of a scatter plot
* A Venn diagram compare significant genes reported by Limma vs by permutation

3. Should compare permutation p-values against moderated (adjusted multiple testing) p-values from Limma (instead of raw p-values)

4. Do the permutation without >>> eBays() (can extrat the test statistic from LimFit())

5. Last thing (do at the end): parallelise permutations-> may require server as input

6. Read some papers about multiple testing adjustments

“ mg\_perm\_array\_mtstat\_ind3.Rds"

* 1000 permutation;
* Cover individuals <- c("685\_686", "689\_690","691\_692")
* Takes about 3h
* >>> dim(mg.perm.array)
* [1] 12700 8 1000
* Code for each permuation

all.bct.shuffled <- sample(all.bct, replace = FALSE, size =length(all.bct))

# to do marker analysis using Limma way

design <- model.matrix(~0+all.bct.shuffled)

colnames(design)[1:(length(levels(all.bct.shuffled)))] <- levels(all.bct.shuffled)

mycont <- matrix(0,

ncol=length(levels(all.bct.shuffled)),

nrow=length(levels(all.bct.shuffled)))

colnames(mycont)<-levels(all.bct.shuffled)

diag(mycont)<-1

mycont[upper.tri(mycont)]<- -1/(length(levels(all.bct.shuffled))-1)

mycont[lower.tri(mycont)]<- -1/(length(levels(all.bct.shuffled))-1)

# Fill out remaining rows with 0s

zero.rows <- matrix(0,ncol=length(levels(all.bct.shuffled)),

nrow=(ncol(design)-length(levels(all.bct.shuffled))))

test <- rbind(mycont, zero.rows)

fit <- lmFit(logcounts.all,design)

fit.cont <- contrasts.fit(fit, contrasts=test)

fit.cont <- eBayes(fit.cont,trend=TRUE,robust=TRUE)

fit.cont$genes <- ann.keep.all

# store the statistic for current permutation

mg.perm.array[,,i]<- fit.cont$t

* plots are in meeting2\_update

Should I include

“mg\_perm\_array\_t3\_tsta.Rds” (733.8Mb)

Note this version f t-stat is wrong!!! (extract from LmFit object, should from contrast.fit object instead!!!! )

* t-statistic for individauls, use 1000 permutations
* Cover individuals <- c("685\_686", "689\_690","691\_692")
* Takes about 2h

Looks like the t-statistic has worse permance (e,g, observation pvalues are mostly 1 ) compared to moderated t-statistic. The p.adjust does not help observation p values

Correct version to find t-statistic

mg\_perm\_array\_ind3\_tstat\_fitcont.Rds

* t-statistic for individauls, use 1000 permutations
* Cover individuals <- c("685\_686", "689\_690","691\_692")
* Takes about 2h
* for (i in 1:perm.size) {

print(i)

# permutate the all.bct only

all.bct.shuffled <- sample(all.bct, replace = FALSE, size =length(all.bct))

# to do marker analysis using Limma way

design <- model.matrix(~0+all.bct.shuffled)

colnames(design)[1:(length(levels(all.bct.shuffled)))] <- levels(all.bct.shuffled)

mycont <- matrix(0,

ncol=length(levels(all.bct.shuffled)),

nrow=length(levels(all.bct.shuffled)))

colnames(mycont)<-levels(all.bct.shuffled)

diag(mycont)<-1

mycont[upper.tri(mycont)]<- -1/(length(levels(all.bct.shuffled))-1)

mycont[lower.tri(mycont)]<- -1/(length(levels(all.bct.shuffled))-1)

# Fill out remaining rows with 0s

zero.rows <- matrix(0,ncol=length(levels(all.bct.shuffled)),

nrow=(ncol(design)-length(levels(all.bct.shuffled))))

test <- rbind(mycont, zero.rows)

fit <- lmFit(logcounts.all,design)

fit.cont <- contrasts.fit(fit, contrasts=test)

fit.cont$genes <- ann.keep.all

mg.perm.array[,,i]<- fit.cont$coef/ fit.cont$stdev.unscaled / fit.cont$sigma

}

t-stat and mod-t stat are close in general, adjusting permutation pvalues for modt and t makes them slightly less similar. However, adjusting limma pvalues for moderated-t and t makes them way more similar

> table(modt.perm.pval.adj == t.perm.pval.adj)

FALSE TRUE

9698 75734

> table(modt.perm.pval == t.perm.pval)

FALSE TRUE

4548 80884

> table(obs.mod.pval.adj == obs.t.pval.adj)

FALSE TRUE

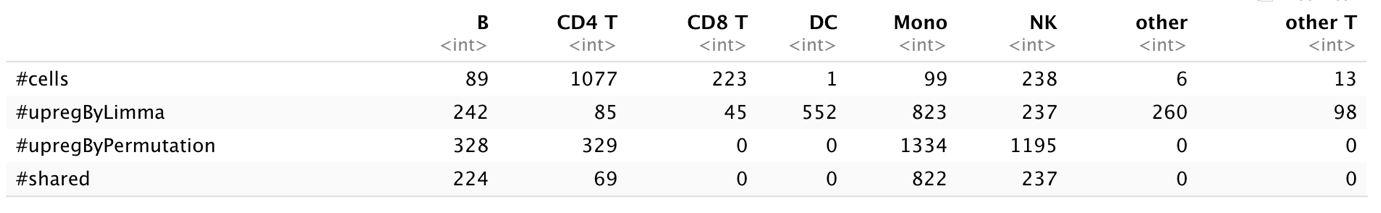
55970 29462

> table(obs.mod.pval == obs.t.pval)

FALSE TRUE

85215 217

* Use t-stat (adjusted pvalue)



B CD4 T CD8 T DC Mono NK other other T

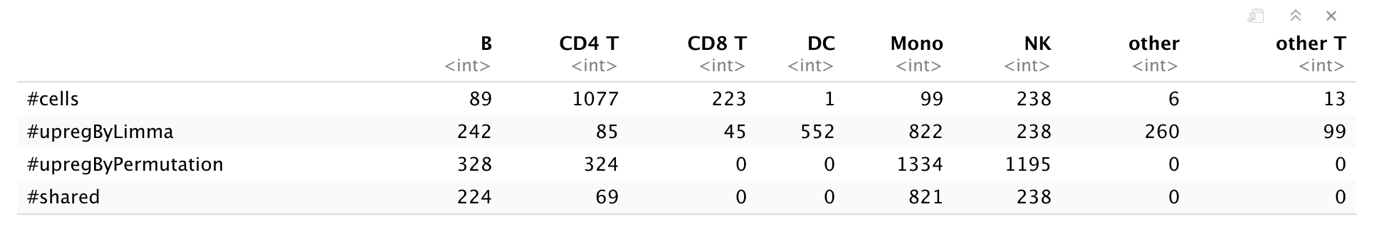
#cells 89 1077 223 1 99 238 6 13

#upregByLimma 242 85 45 552 823 237 260 98

#upregByPermutation 328 329 0 0 1334 1195 0 0

#shared 224 69 0 0 822 237 0 0

* Use mod-t stat(adjusted p value)



B CD4 T CD8 T DC Mono NK other other T

#cells 89 1077 223 1 99 238 6 13

#upregByLimma 242 85 45 552 822 238 260 99

#upregByPermutation 328 324 0 0 1334 1195 0 0

#shared 224 69 0 0 821 238 0 0

With all individuals (modt\_perm\_array\_indall.Rds / t\_perm\_array\_indall.Rds)

Mod-t

B CD4 T CD8 T DC Mono NK other other T

#cells 1747 7988 3154 28 747 2759 46 220

#upregByLimma 1036 589 418 1294 2598 1892 939 514

#upregByPermutation 1112 920 479 607 2633 2403 346 380

#shared 1032 585 399 606 2570 1891 346 378

>

t-stat

B CD4 T CD8 T DC Mono NK other other T

#cells 1747 7988 3154 28 747 2759 46 220

#upregByLimma 1036 590 418 1294 2599 1892 939 514

#upregByPermutation 1112 920 479 607 2633 2403 346 380

#shared 1032 586 399 606 2570 1891 346 378

Histogram for pvalues (from limma/permutation) for one individual (689\_690) for each cell type

hist\_obs.t.pval.adj\_ind1.i.pdf (i=1,…8)

hist\_t.perm.pval.adj\_ind1.i.pdf (i=1,…8)

Simulation

1. remove DC & other cells, keep the rest

# B CD4 T CD8 T Mono NK other T

# 0.051178838 0.619321449 0.128234618 0.056929270 0.136860265 0.007475561

Simulation setup

params <- setParam(params, "group.prob", ct\_prop)

params <- splatEstimate(sce,params = params)

# SplatParams

sim <- splatSimulate(params, method = "groups", nGenes = 5000,

batchCells = 1000, dropout.type = "none", out.prob = 0,

de.prob = c(0.02, 0.02, 0, 0, 0, 0),

group.prob = c(9/10, 1/50, 1/50,1/50,1/50,1/50),

seed = 12202)

# Group1 Group2 Group3 Group4 Group5 Group6

# 903 12 29 19 13 24

121 up-reg genes in Group1

102 up-reg genes in Group2

Obs p-values vs perm p values (Saved as “sim1\_p”)

Diagram, engineering drawing, line chart

Description automatically generated

Adj Obs p-values vs adj perm p values (Saved as “sim1\_adjp”)

Engineering drawing

Description automatically generated with medium confidence

Group1 Group2 Group3 Group4 Group5 Group6

795 205 1 2 0 0

Chart, histogram

Description automatically generated

# B cells = 800

|  |
| --- |
|  |

Chart, line chart

Description automatically generatedpvalue (“sim2\_p.pdf”)

pvalue (“sim2\_p.pdf”)

Chart, line chart, scatter chart

Description automatically generated

# B cells = 795

# CD4T cells = 205

* With p-value

"number of up-reg genes for B cells = 346 (limma)"

"number of up-reg genes for B cells = 375 (permutation)"

"# up-reg genes overlap (limma & permutation) for B cells = 335 genes"

"#true up-reg genes for B cells = 20 (17.094%) (limma)"

"#true up-reg genes for B cells = 20 (17.094%) (permutation)"

number of up-reg genes for CD4T cells = 274 (limma)"

number of up-reg genes for CD4T cells = 247 (permutation)"

#up-reg genes overlap (limma & permutation) for CD4T cells = 239 genes"

#true up-reg genes for CD4T cells = 36 (29.752%) (limma)"

"#true up-reg genes for CD4T cells = 35 (28.926%) (permutation)"

* With adjusted p-value

"number of up-reg genes for B cells = 27 (limma)"

[1] "number of up-reg genes for B cells = 0 (permutation)"

[1] "# up-reg genes overlap (limma & permutation) for B cells = 0 genes"

[1] "#true up-reg genes for B cells = 9 (7.692%) (limma)"

[1] "#true up-reg genes for B cells = 0 (0%) (permutation)"

[1] "number of up-reg genes for CD4T cells = 52 (limma)"

[1] "number of up-reg genes for CD4T cells = 0 (permutation)"

[1] "# up-reg genes overlap (limma & permutation) for CD4T cells = 0 genes"

[1] "#true up-reg genes for CD4T cells = 18 (14.876%) (limma)"

[1] "#true up-reg genes for CD4T cells = 0 (0%) (permutation)"

B CD4T CD8T Mono NK otherT

903 97 0 0 0 0

Chart, line chart

Description automatically generated

pvalue

B CD4T

#cells 903.000 97.000

#upRegGenes 117.000 121.000

#upreg(limma) 248.000 339.000

#trueUpreg(limma) 19.000 29.000

%trueUpreg(limma) 16.239 23.967

#upreg(permutation) 334.000 295.000

#trueUpreg(permutation) 24.000 28.000

%trueUpreg(permutation) 20.513 23.140

#overlap(limma&permutation) 242.000 289.000

Adjusted pvalue

B CD4T

#cells 903.000 97.000

#upRegGenes 117.000 121.000

#upreg(limma) 13.000 49.000

#trueUpreg(limma) 5.000 13.000

%trueUpreg(limma) 4.274 10.744

#upreg(permutation) 0.000 0.000

#trueUpreg(permutation) 0.000 0.000

%trueUpreg(permutation) 0.000 0.000

#overlap(limma&permutation) 0.000 0.000

Result for different cell proportions,

Note that the up-regulated genes proportion is 20% for both B cells and CD4T cells, however the value of up-regulation is 0.01 (thus low recovery rate )

Week 11

First run: set de.prob = 0.2 for all six groups

Summary resul > Simulation\_100upreggenes

“raw\_pvalue\_result\_1000upreg.csv” & “adj\_pvalue\_result\_1000upreg.csv”

Sensitivity = True positive rate (TPR) = TP / (TP + FN)

Precision = Positive predictive value (PPV) = TP / (TP + FP)

False positive rate (FPR), [fall-out](https://en.wikipedia.org/wiki/Information_retrieval), probability of false alarm= FP / (FP + TN)

Summary result > “Simulation\_100upreggenes” used in meeting 17/05

adj\_pvalue\_result\_100upreg.csv

raw\_pvalue\_result\_100upreg.csv

TODO

* Volcano plot
* Ranking of genes (smallest pvalue to largest pvalue)
* Violin plot (y=logFC, x = TP/ FP)

## 25/06 Meeting with Belinda

**What I have done over the semester: make simulation, general framework with Limma; compare the impact of cell type proportions on marker gene detection (Limma versus permutation).**

**A general plan: Aim to get most of the project done in July & August**

**- try different statistics on marker analysis (one idea: logFC\*(1-p\_value)), and use the optimal one for gene set testing. t-statistic = logFC/variance**

**logFC is the coefficient (in linear model is logFC) >>> fit.contrast$coef**

**- a threshold t test**

**- to see if two-sided t-test is not necessary (we hope two-sided will give more false positives as we are normally more interested in up-regulated genes instead of down-regulated genes)**

# Week 28/06

## 28/06 Group Meeting

**- Better to try a simple simulation E.g. pick one cell type, estimate parameters (e.g. library size) and use this parameter set to simulate two groups, one of them has 100 up-regulated genes, then change the related proportion**

**This is because with a complicated design (different sets of params for each cell type will impact the analysis, and we want to get rid of all biology just investigate whether cell proportion will affect the power to detect DE genes)**

* onect\_3logfac

contains one cell type (CD4T);

simulation details: de.prob = c(0.02, 0), de.downProb = c(0, 0), de.facLoc = c(3,0),

* onect\_10logfac

contains one cell type (CD4T);

simulation details: de.prob = c(0.02, 0), de.downProb = c(0, 0), de.facLoc = c(10,0),

* onect\_5logfac

contains one cell type (CD4T);

simulation details: de.prob = c(0.02, 0), de.downProb = c(0, 0), de.facLoc = c(5,0),

## 02/07 Meeting with Belinda

It’s necessary to do QC after simulation!!! Need to set droupout and outliers to be none.

Pay more attention two the following metrics: tp/(tp + fn) & tp/(tp+fp)

## Code

* Folder simplesimulation\_10kgenes\_withqc

groups\_lst<- list(c(0.5, 0.5), c(0.01, 0.99), c(0.1, 0.9), c(0.2, 0.8), c(0.3, 0.7), c(0.4, 0.6))

perm.size = 1000

for (i in 2:length(groups\_lst)){

print(i)

sim <- splatSimulate(params, method = "groups", nGenes = 10000,

batchCells = 1000, dropout.type = "none", out.prob = 0,

de.prob = c(0.02, 0.02), de.downProb = c(0, 0),

group.prob = groups\_lst[[i]],

seed = 12202,

de.facLoc = c(2,2),

#de.facScale = c(0.001, 0.001, 0.001, 0.001, 0.001, 0.001),

verbose = FALSE)

* folder: simplesimulation\_10kgenes\_withqc\_15cp

groups\_lst<- list(c(0.5, 0.5), c(0.01, 0.99), c(0.02, 0.98), c(0.03, 0.97), c(0.04, 0.96), c(0.05, 0.95), c(0.06, 0.94), c(0.07, 0.93), c(0.08, 0.92), c(0.09, 0.91), c(0.1, 0.9), c(0.15, 0.85), c(0.2, 0.8), c(0.3, 0.7), c(0.4, 0.6) )

perm.size = 1000

set.seed(12202)

seed\_number <- sample(x = 100:10000, size = length(groups\_lst))

for (m in 1:length(groups\_lst))

sim <- splatSimulate(params, method = "groups", nGenes = 10000,

batchCells = 1000, dropout.type = "none", out.prob = 0,

de.prob = c(0.02, 0.02), de.downProb = c(0, 0),

group.prob = groups\_lst[[m]],

seed = seed\_number[m],

de.facLoc = c(2,2),

verbose = FALSE)

* folder simplesimulation\_10kgenes\_withqc\_11cp

groups\_lst<- list(c(0.5, 0.5), c(0.01, 0.99), c(0.03, 0.97), c(0.05, 0.95), c(0.07, 0.93), c(0.09, 0.91), c(0.1, 0.9), c(0.15, 0.85), c(0.2, 0.8), c(0.3, 0.7), c(0.4, 0.6) )

perm.size = 1000

set.seed(12202)

seed\_number <- sample(x = 100:10000, size = length(groups\_lst))

for (m in 1:length(groups\_lst)){

print(m)

sim <- splatSimulate(params, method = "groups", nGenes = 10000,

batchCells = 1000, dropout.type = "none", out.prob = 0,

de.prob = c(0.02, 0.02), de.downProb = c(0, 0),

group.prob = groups\_lst[[m]],

seed = seed\_number[m],

de.facLoc = c(2,2),

#de.facScale = c(0.001, 0.001, 0.001, 0.001, 0.001, 0.001),

verbose = FALSE)

* folder: simplesimulation\_10kgenes\_withqc\_15cp\_new

groups\_lst<- list(c(0.01, 0.99), c(0.05, 0.95), c(0.1, 0.9), c(0.15, 0.85), c(0.2, 0.8), c(0.3, 0.7), c(0.4, 0.6), c(0.5, 0.5), c(0.6, 0.4), c(0.7, 0.3), c(0.8, 0.2), c(0.85, 0.15), c(0.9, 0.1), c(0.95, 0.05), c(0.99, 0.01) )

perm.size = 1000

set.seed(12202)

seed\_number <- sample(x = 100:10000, size = length(groups\_lst))

for (m in 1:length(groups\_lst)){

print(m)

sim <- splatSimulate(params, method = "groups", nGenes = 10000,

batchCells = 1000, dropout.type = "none", out.prob = 0,

de.prob = c(0.02, 0.02), de.downProb = c(0, 0),

group.prob = groups\_lst[[m]],

seed = seed\_number[m],

de.facLoc = c(2,2),

#de.facScale = c(0.001, 0.001, 0.001, 0.001, 0.001, 0.001),

verbose = FALSE)

df\_raw\_adj$group<- factor(df\_raw\_adj$group)

df\_raw\_adj$simulation<- df\_raw\_adj$group

levels(df\_raw\_adj$simulation) <- gsub("16\_984", "simulation1", levels(df\_raw\_adj$simulation))

levels(df\_raw\_adj$simulation) <- gsub("65\_935", "simulation2", levels(df\_raw\_adj$simulation))

levels(df\_raw\_adj$simulation) <- gsub("95\_905", "simulation3", levels(df\_raw\_adj$simulation))

levels(df\_raw\_adj$simulation) <- gsub("131\_869", "simulation4", levels(df\_raw\_adj$simulation))

levels(df\_raw\_adj$simulation) <- gsub("196\_804", "simulation5", levels(df\_raw\_adj$simulation))

levels(df\_raw\_adj$simulation) <- gsub("297\_703", "simulation6", levels(df\_raw\_adj$simulation))

levels(df\_raw\_adj$simulation) <- gsub("380\_620", "simulation7", levels(df\_raw\_adj$simulation))

levels(df\_raw\_adj$simulation) <- gsub("504\_496", "simulation8", levels(df\_raw\_adj$simulation))

levels(df\_raw\_adj$simulation) <- gsub("606\_394", "simulation9", levels(df\_raw\_adj$simulation))

levels(df\_raw\_adj$simulation) <- gsub("698\_302", "simulation10", levels(df\_raw\_adj$simulation))

levels(df\_raw\_adj$simulation) <- gsub("813\_187", "simulation11", levels(df\_raw\_adj$simulation))

levels(df\_raw\_adj$simulation) <- gsub("853\_147", "simulation12", levels(df\_raw\_adj$simulation))

levels(df\_raw\_adj$simulation) <- gsub("907\_93", "simulation13", levels(df\_raw\_adj$simulation))

levels(df\_raw\_adj$simulation) <- gsub("957\_43", "simulation14", levels(df\_raw\_adj$simulation))

levels(df\_raw\_adj$simulation) <- gsub("987\_13", "simulation15", levels(df\_raw\_adj$simulation))

df\_raw\_adj$simulation <- factor(df\_raw\_adj$simulation,

levels = c("simulation1", "simulation2",

"simulation3", "simulation4",

"simulation5",

"simulation6","simulation7","simulation8","simulation9","simulation10","simulation11",

"simulation12","simulation13","simulation14", "simulation15"))

df\_raw\_adj$precision<- df\_raw\_adj$precision/100

df\_raw\_adj$sensitivity <- df\_raw\_adj$sensitivity/100

df\_raw\_adj$FPR<-df\_raw\_adj$FPR/100

df\_raw\_adj$group<- factor(df\_raw\_adj$group)

df\_raw\_adj$cell.type<- factor(df\_raw\_adj$cell.type)

df\_raw\_adj$test.name<- factor(df\_raw\_adj$test.name)

table(pbmc.data[,colnames(cm.processed)]$predicted.celltype.l1)

B CD4 T CD8 T DC Mono NK other other T

89 1077 223 1 99 238 6 13

# Week 05/07

## 05/07 Meeting with Belinda

**- Up-regulated genes by simulation is set randomly, do not based on high-expression level. Important: some true DE genes are removed by filtering process**

**- de.facLoc and de.facScale seem to control the overall distribution, do not directly control logFC. Also, de.facLoc=1 is already considered as large, the ratio of de.facLoc/de.facScale relates to the logFC.**

**- Notice that the raw p-value distribution for permutation looks nice, and the adjustment step somehow does not work well.**

**Another thing to do: instead of p-value adjustment, we can do a simple cutoff (e.g. regard genes <0.01 as DE). We can test several cutoff values (e.g. 0.001-0.05)**

**- To do: repeat the procedure of one cell proportion for several times, to see if any of then have non-zero TP with permutation (concern about the chance of luck with simulation)**

## Code

Repeat the simulation for the same cell proportion for 30 times:

adj\_pvalue\_result\_repeat\_simple\_7030.csv

raw\_pvalue\_result\_repeat\_simple\_7030.csv

set.seed(12202)

seed.number<- sample(x = 100:10000, size = 30, replace = FALSE) #length(groups\_lst))

sim <- splatSimulate(curr\_params, method = "groups", nGenes = 5000,

batchCells = 1000, dropout.type = "none", out.prob = 0,

de.prob = c(0.02, 0.02), de.downProb = c(0, 0),

group.prob = c(70, 30),

seed = seed.number[r],

de.facLoc = c(2,2),

#de.facScale = c(0.3, 0.3),

verbose = FALSE)

Observation: Most cases (29/30) have nonzero TP for permutation !!

Folder: 5repeat\_simulation\_for\_15cp\_cutoff005

groups\_lst<- list(c(0.01, 0.99), c(0.05, 0.95), c(0.1, 0.9), c(0.15, 0.85), c(0.2, 0.8), c(0.3, 0.7),c(0.4, 0.6),c(0.5, 0.5),c(0.6, 0.4),c(0.7, 0.3),c(0.8, 0.2),c(0.85, 0.15), c(0.9, 0.1), c(0.95, 0.05), c(0.99, 0.01) )

perm.size = 1000

set.seed(12202)

seed.number.cp <- sample(x = 100:10000, size = length(groups\_lst), replace = FALSE) #length(groups\_lst))

rep<- 5

for (m in 1:length(groups\_lst)){

print(m)

set.seed(seed.number.cp[m])

seed.number <- sample(x = 80:seed.number.cp[m], size = rep, replace = FALSE)

for (r in 1:rep){

sim <- splatSimulate(curr\_params, method = "groups", nGenes = 5000,

batchCells = 1000, dropout.type = "none", out.prob = 0,

de.prob = c(0.02, 0.02), de.downProb = c(0, 0),

group.prob = groups\_lst[[m]],

seed = seed.number[r],

de.facLoc = c(2,2),

#de.facScale = c(0.3, 0.3),

verbose = FALSE)

Repeat the simulation for the same cell proportion (90-10%) for 30 times, cutoff=0.05:

Repeat\_simulation/adj\_pvalue\_result\_repeat\_simple\_1090.csv

Repeat\_simulation/raw\_pvalue\_result\_repeat\_simple\_1090.csv

Repeat the simulation for the same cell proportion (90-10%) for 30 times, cutoff=0.005:

Repeat\_simulation/adj\_pvalue\_result\_repeat\_simple\_1090\_2.csv

Repeat\_simulation/raw\_pvalue\_result\_repeat\_simple\_1090\_2.csv

perm.size = 1000

set.seed(12202)

seed.number.cp <- sample(x = 100:10000, size = length(groups\_lst), replace = FALSE) #length(groups\_lst))

rep<- 30

for (m in 1:1){

print(m)

set.seed(seed.number.cp[3])

seed.number <- sample(x = 80:seed.number.cp[3], size = rep, replace = FALSE)

for (r in 1:rep){

sim <- splatSimulate(curr\_params, method = "groups", nGenes = 5000,

batchCells = 1000, dropout.type = "none", out.prob = 0,

de.prob = c(0.02, 0.02), de.downProb = c(0, 0),

group.prob = c(0.1, 0.9),

seed = seed.number[r],

de.facLoc = c(2,2),

#de.facScale = c(0.3, 0.3),

verbose = FALSE)

# Week 12/07

## Group Meeting

- Do a boxplot to compare across simulations instead of a line chart

- Calculate AUC-ROC

## Meeting with Belinda

# Week 19/07

## code

Folder comparative\_s336\_s1940\_s2972 contains MA plot, ROC-AUC, for seed number 336 (good case, permutation after multiple testing adjustment has high performance) and seed number 1940, 2972 when multiple testing adjustment fails

Interesting Note: seed=336, minimum adjusted p value for permutation is about 0.0441 and 0.0495, slightly smaller than 0.05; whereas seed=1940/2972 produce a just over 0.05 minimum adjusted p-value.

The distribution of raw p-values in all tested seed numbers are similar to each other. No particular trend that differentiate the working case and the rest.

All cases have high AUC.

groups\_lst<- list(c(0.01, 0.99), c(0.05, 0.95), c(0.1, 0.9), c(0.15, 0.85), c(0.2, 0.8), c(0.3, 0.7),c(0.4, 0.6),c(0.5, 0.5),c(0.6, 0.4),c(0.7, 0.3),c(0.8, 0.2),c(0.85, 0.15), c(0.9, 0.1), c(0.95, 0.05), c(0.99, 0.01) )

perm.size = 1000

set.seed(12202)

seed.number.cp <- sample(x = 100:10000, size = length(groups\_lst), replace = FALSE) #length(groups\_lst))

rep<- 30

#for (m in 1:length(groups\_lst)){

for (m in 1:1){

print(m)

set.seed(seed.number.cp[3])

seed.number <- sample(x = 80:seed.number.cp[3], size = rep, replace = FALSE)

for (r in 1:1){

sim <- splatSimulate(curr\_params, method = "groups", nGenes = 5000,

batchCells = 1000, dropout.type = "none", out.prob = 0,

de.prob = c(0.02, 0.02), de.downProb = c(0, 0),

group.prob = groups\_lst[[3]],

seed = 336,

de.facLoc = c(2,2),

#de.facScale = c(0.3, 0.3),

verbose = FALSE)

numzero.genes <- rowSums(counts(sim) ==0)

keep.genes <- numzero.genes < (ncol(counts(sim))-10)

sim<- sim[keep.genes, ]

## 23/07 Meeting with Belinda

**TODO**

**- To test whether increase permutation times can generate a lower adjusted p-value**

**??? will there be a correlation between number of genes and number of permutations required, try from #perm=#genes, as well as #perm = 10,000**

**- There is a parallel R package (might help)**

**- A boxplot of logFC of True DE in working case versus logFC in unworking cases**

**- A cutoff plot (just like what I have) including a cutoff = min(t.perm.value); can also change the text labels to a line so that we can know the exact precision/sensitivity value**

**- An interesting trend: the sensitivity for a smaller cell group (100 cells) is higher than that for a larger cell group (900 cells), but the precision is the other way.**

**Folder 8cellprop\_compare contains the result for comparative simulation for checking the impact of eight cell proportion.**

groups\_lst<- list(c(0.01, 0.99), c(0.05, 0.95), c(0.1, 0.9), c(0.15, 0.85), c(0.2, 0.8), c(0.3, 0.7), c(0.4, 0.6), c(0.5, 0.5))

perm.size = 2000

set.seed(12202)

seed.number.cp <- sample(x = 100:10000, size = length(groups\_lst), replace = FALSE) #length(groups\_lst))

rep<- 1

for (m in 1:length(groups\_lst)){

#for (m in 1:1){

print(m)

#set.seed(seed.number.cp[3])

#seed.number <- sample(x = 80:seed.number.cp[3], size = rep, replace = FALSE)

#for (r in 1:rep){

sim <- splatSimulate(curr\_params, method = "groups", nGenes = 5000,

batchCells = 1000, dropout.type = "none", out.prob = 0,

de.prob = c(0.02, 0.02), de.downProb = c(0, 0),

group.prob = groups\_lst[[m]],

seed = seed.number.cp[m],

de.facLoc = c(0.3,0.3),

de.facScale = c(0.3, 0.3),

verbose = FALSE)

numzero.genes <- rowSums(counts(sim) ==0)

keep.genes <- numzero.genes < (ncol(counts(sim))-10)

sim<- sim[keep.genes, ]

**Short conclusion: MOST permutation fails with p-value adjustment(none sig genes detected).**

**AUC increases as the cell proportion becomes more even.**

**Both limma and permutation have similar AUC and raw p-value distribution**

folder 8cellprop\_compare\_1000

groups\_lst<- list(c(0.01, 0.99), c(0.05, 0.95), c(0.1, 0.9), c(0.15, 0.85), c(0.2, 0.8), c(0.3, 0.7), c(0.4, 0.6), c(0.5, 0.5))

perm.size = 1000

set.seed(12202)

seed.number.cp <- sample(x = 100:10000, size = length(groups\_lst), replace = FALSE) #length(groups\_lst))

rep<- 1

for (m in 1:length(groups\_lst)){

#for (m in 1:1){

print(m)

sim <- splatSimulate(curr\_params, method = "groups", nGenes = 5000,

batchCells = 1000, dropout.type = "none", out.prob = 0,

de.prob = c(0.02, 0.02), de.downProb = c(0, 0),

group.prob = groups\_lst[[m]],

seed = seed.number.cp[m],

de.facLoc = c(0.5,0.5),

#de.facScale = c(0.3, 0.3),

verbose = FALSE)

numzero.genes <- rowSums(counts(sim) ==0)

keep.genes <- numzero.genes < (ncol(counts(sim))-10)

sim<- sim[keep.genes, ]

folder 8cellprop\_compare\_2000

groups\_lst<- list(c(0.01, 0.99), c(0.05, 0.95), c(0.1, 0.9), c(0.15, 0.85), c(0.2, 0.8), c(0.3, 0.7), c(0.4, 0.6), c(0.5, 0.5))

perm.size = 2000

set.seed(12202)

seed.number.cp <- sample(x = 100:10000, size = length(groups\_lst), replace = FALSE) #length(groups\_lst))

rep<- 1

for (m in 1:length(groups\_lst)){

#for (m in 1:1){

print(m)

sim <- splatSimulate(curr\_params, method = "groups", nGenes = 5000,

batchCells = 1000, dropout.type = "none", out.prob = 0,

de.prob = c(0.02, 0.02), de.downProb = c(0, 0),

group.prob = groups\_lst[[m]],

seed = seed.number.cp[m],

de.facLoc = c(0.5,0.5),

#de.facScale = c(0.3, 0.3),

verbose = FALSE)

numzero.genes <- rowSums(counts(sim) ==0)

keep.genes <- numzero.genes < (ncol(counts(sim))-10)

sim<- sim[keep.genes, ]

folder 8cellprop\_compare\_2000\_updatedplots

Same setup as 8cellprop\_compare\_2000 but use a neat version to generate plots, include facet\_zoom to zoom in the x axis;

Folder 7cellprop\_compare\_1000, an updated version of plots used in meeting 09/08 (fixing the plots with ambiguous jittered plots and overlapped boxplot)

groups\_lst<- list(c(0.01, 0.99), c(0.05, 0.95), c(0.1, 0.9), c(0.2, 0.8), c(0.3, 0.7), c(0.4, 0.6), c(0.5, 0.5) )

perm.size = 1000

set.seed(12202)

seed.number.cp <- sample(x = 100:10000, size = length(groups\_lst), replace = FALSE)

rep<- 1

for (m in 1:length(groups\_lst)){

sim <- splatSimulate(curr\_params, method = "groups", nGenes = 5000,

batchCells = 1000, dropout.type = "none", out.prob = 0,

de.prob = c(0.02, 0.02), de.downProb = c(0, 0),

group.prob = groups\_lst[[m]],

seed = seed.number.cp[m],

de.facLoc = c(0.5,0.5),

verbose = FALSE)}

Folder: 7cellprop\_compare\_2000, same setup as previous as with 2000 permutations

groups\_lst<- list(c(0.01, 0.99), c(0.05, 0.95), c(0.1, 0.9), c(0.2, 0.8), c(0.3, 0.7), c(0.4, 0.6), c(0.5, 0.5) )

perm.size = 2000

set.seed(12202)

seed.number.cp <- sample(x = 100:10000, size = length(groups\_lst), replace = FALSE)

rep<- 1

for (m in 1:length(groups\_lst)){

sim <- splatSimulate(curr\_params, method = "groups", nGenes = 5000,

batchCells = 1000, dropout.type = "none", out.prob = 0,

de.prob = c(0.02, 0.02), de.downProb = c(0, 0),

group.prob = groups\_lst[[m]],

seed = seed.number.cp[m],

de.facLoc = c(0.5,0.5),

verbose = FALSE)}

Folder cp1090\_loc05scale01\_100k

perm.size = 100000

sim <- splatSimulate(curr\_params, method = "groups", nGenes = 5000,

batchCells = 1000, dropout.type = "none", out.prob = 0,

de.prob = c(0.02, 0.02),de.downProb = c(0, 0),

group.prob = c(0.1, 0.9),

seed = 336,

de.facLoc = c(0.5, 0.5),

de.facScale = c(0.1,0.1),

verbose = FALSE)

numzero.genes <- rowSums(counts(sim)==0)

keep.genes <- numzero.genes < (ncol(counts(sim))-10)

keep.genes <- setdiff(row.names(sim)[keep.genes],

intersect(row.names(sim)[sim@rowRanges@elementMetadata$DEFacGroup1 > 1],

row.names(sim)[sim@rowRanges@elementMetadata$DEFacGroup2 > 1]))

sim<- sim[keep.genes, ]

result<- runComparison(sim, perm.size)