scGPS introduction

Quan Nguyen and Michael Thompson

2018-12-18

Contents

1. Installation instruction **1**

2. A simple workflow of the scGPS: **2**

2.1 Create scGPS objects . . . 2
2.2 Run prediction . . . 2
2.3 Summarise results . . . 3

3. A complete workflow of the scGPS: **3**

3.1 Identify clusters in a dataset using CORE . . . 3
3.1 Identify clusters in a dataset using SCORE (Stable Clustering at Optimal REsolution) . . . 3
3.2 Visualise all cluster results in all iterations . . . 5
3.4 Compare clustering results with other dimensional reduction methods (e.g., CIDR) . . . 5
3.5 Find gene markers and annotate clusters . . . 5

4. Relationship between clusters within one sample or between two samples **7**

4.1 Start the scGPS prediction to find relationship between clusters . . . 7
4.2 Display summary results for the prediction . . . 12

test **13**

4.3 Plot the relationship between clusters in one sample . . . 20
4.3 Plot the relationship between clusters in two samples . . . 22
4.4 Annotation: scGPS prediction can be used to compare scGPS clusters with a reference dataset to see which cluster is most similar to the reference . . . 23

1. Installation instruction

```
# Prior to installing scGPS you need to install the SummarizedExperiment
# bioconductor package as the following
# source('https://bioconductor.org/biocLite.R') biocLite('SummarizedExperiment')

# To install scGPS from github (Depending on the configuration of the local
# computer or HPC, possible custom C++ compilation may be required - see
# installation trouble-shootings below)
devtools::install_github("IMB-Computational-Genomics-Lab/scGPS")

# for C++ compilation trouble-shooting, manual download and installation can be
# done from github

git clone https://github.com/IMB-Computational-Genomics-Lab/scGPS

# then check in scGPS/src if any of the precompiled (e.g.  those with *.so and
# *.o) files exist and delete them before recompiling
```

```r
# create a Makevars file in the scGPS/src with one line: PKG_LIBS =
# $(LAPACK_LIBS) $(BLAS_LIBS) $(FLIBS)

# then with the scGPS as the R working directory, manually recompile scGPS in R
# using devtools to load and install functions
devtools::document()
#load the package to the workspace
devtools::load_all()
```

2. A simple workflow of the scGPS:

The purpose of this workflow is to solve the following task: given a mixed population with known subpopulations, estimate transition scores between these subpopulation

2.1 Create scGPS objects

```r
# load mixed population 1 (loaded from sample1 dataset, named it as day2)
# setwd('/Users/quan.nguyen/Documents/Powell_group_MacQuan/AllCodes/scGPS/vignettes/')
devtools::load_all()

day2 <- sample1
mixedpop1 <- NewscGPS(ExpressionMatrix = day2$dat2_counts, GeneMetadata = day2$dat2geneInfo,
    CellMetadata = day2$dat2_clusters)

# load mixed population 2 (loaded from sample2 dataset, named it as day5)
day5 <- sample2
mixedpop2 <- NewscGPS(ExpressionMatrix = day5$dat5_counts, GeneMetadata = day5$dat5geneInfo,
    CellMetadata = day5$dat5_clusters)
```

2.2 Run prediction

```r
# select a subpopulation
c_selectID <- 1
# load gene list (this can be any lists of user selected genes)
genes <- GeneList
genes <- genes$Merged_unique
# load cluster information
cluster_mixedpop1 <- colData(mixedpop1)[,1]
cluster_mixedpop2 <- colData(mixedpop2)[,1]
#run training
LSOLDA_dat <- bootstrap_scGPS(nboots = 2, mixedpop1 = mixedpop1,
    mixedpop2 = mixedpop2, genes = genes, c_selectID  = c_selectID, listData = list(),
     cluster_mixedpop1 = cluster_mixedpop1,
     cluster_mixedpop2 = cluster_mixedpop2)
```

2.3 Summarise results

```
# display the list of result information in the LASOLDA_dat object
names(LSOLDA_dat)
LSOLDA_dat$ElasticNetPredict
LSOLDA_dat$LDAPredict

# summary results LDA
summary_prediction_lda(LSOLDA_dat = LSOLDA_dat, nPredSubpop = 4)

# summary results Lasso to show the percent of cells classified as cells belonging
summary_prediction_lasso(LSOLDA_dat = LSOLDA_dat, nPredSubpop = 4)

# summary accuracy to check the model accuracy in the leave-out test set
summary_accuracy(object = LSOLDA_dat)

# summary maximum deviance explained by the model
summary_deviance(object = LSOLDA_dat)
```

3. A complete workflow of the scGPS:

The purpose of this workflow is to solve the following task: given an unknown mixed population, find clusters and estimate relationship between clusters

3.1 Identify clusters in a dataset using CORE

(skip this step if clusters are known)

```
# find clustering information in an expresion data using CORE
day5 <- sample2
cellnames <- colnames(day5$dat5_counts)
cluster <-day5$dat5_clusters
cellnames <-data.frame("Cluster"=cluster, "cellBarcodes" = cellnames)
mixedpop2 <-NewscGPS(ExpressionMatrix = day5$dat5_counts, GeneMetadata = day5$dat5geneInfo, CellMetadat

CORE_cluster <- CORE_scGPS(mixedpop2, remove_outlier = c(0), PCA=FALSE)
```

3.1 Identify clusters in a dataset using SCORE (Stable Clustering at Optimal REsolution)

(skip this step if clusters are known) (SCORE aims to get stable subpopulation results, by introducing bagging aggregation and bootstrapping to the CORE algorithm)

```
# find clustering information in an expresion data using SCORE
day5 <- sample2
cellnames <- colnames(day5$dat5_counts)
cluster <-day5$dat5_clusters
cellnames <-data.frame("Cluster"=cluster, "cellBarcodes" = cellnames)
mixedpop2 <-NewscGPS(ExpressionMatrix = day5$dat5_counts, GeneMetadata = day5$dat5geneInfo, CellMetadat
SCORE_test <- CORE_scGPS_bagging(mixedpop2, remove_outlier = c(0), PCA=FALSE,
                         bagging_run = 20, subsample_proportion = .8)
#> [1] "Performing 1 round of filtering"
#> [1] "Identifying top variable genes"
#> [1] "Calculating distance matrix"
#> [1] "Performing hierarchical clustering"
#> [1] "Finding clustering information"
#> [1] "No more outliers detected in filtering round 1"
#> [1] "Identifying top variable genes"
#> [1] "Calculating distance matrix"
#> [1] "Performing hierarchical clustering"
#> [1] "Finding clustering information"
#> [1] "500 cells left after filtering"
#> [1] "Running 20 bagging runs, with 0.8 subsampling..."
#> [1] "Done clustering, moving to stability calculation..."
#> [1] "Done calculating stability..."
#> [1] "Start finding optimal clustering..."
#> [1] "Done calculating stability..."
#> [1] "Start finding optimal clustering..."
#> [1] "Done calculating stability..."
#> [1] "Start finding optimal clustering..."
#> [1] "Done calculating stability..."
#> [1] "Start finding optimal clustering..."
#> [1] "Done calculating stability..."
#> [1] "Start finding optimal clustering..."
#> [1] "Done calculating stability..."
#> [1] "Start finding optimal clustering..."
#> [1] "Done calculating stability..."
#> [1] "Start finding optimal clustering..."
#> [1] "Done calculating stability..."
#> [1] "Start finding optimal clustering..."
#> [1] "Done calculating stability..."
#> [1] "Start finding optimal clustering..."
#> [1] "Done calculating stability..."
#> [1] "Start finding optimal clustering..."
#> [1] "Done calculating stability..."
#> [1] "Start finding optimal clustering..."
#> [1] "Done calculating stability..."
#> [1] "Start finding optimal clustering..."
#> [1] "Done calculating stability..."
#> [1] "Start finding optimal clustering..."
#> [1] "Done calculating stability..."
#> [1] "Start finding optimal clustering..."
#> [1] "Done calculating stability..."
#> [1] "Start finding optimal clustering..."
#> [1] "Done calculating stability..."
#> [1] "Start finding optimal clustering..."
#> [1] "Done calculating stability..."
#> [1] "Start finding optimal clustering..."
#> [1] "Done calculating stability..."
#> [1] "Start finding optimal clustering..."
```

3.2 Visualise all cluster results in all iterations

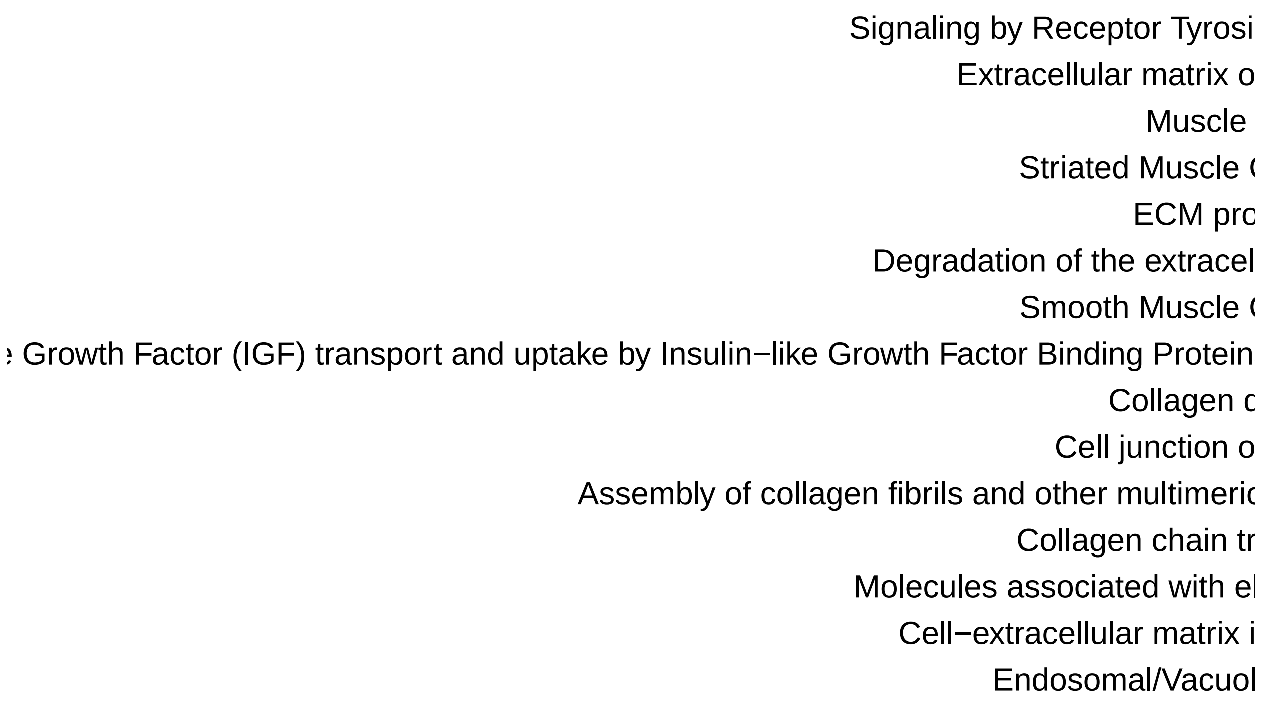
```
##3.2.1 plot CORE clustering
plot_CORE(CORE_cluster$tree, CORE_cluster$Cluster) #plot all clustering bars
#extract optimal index identified by CORE_scGPS
key_height <- CORE_cluster$optimalClust$KeyStats$Height
optimal_res <- CORE_cluster$optimalClust$OptimalRes
optimal_index = which(key_height == optimal_res)
#plot one optimal clustering bar
plot_optimal_CORE(original_tree= CORE_cluster$tree,
                  optimal_cluster = unlist(CORE_cluster$Cluster[optimal_index]), shift = -2000)
# you can customise the cluster color bars (provide color_branch values)
plot_CORE(CORE_cluster$tree, CORE_cluster$Cluster, color_branch = c("#208eb7", "#6ce9d3", "#1c5e39", "#8

##3.2.2 plot SCORE clustering
plot_CORE(SCORE_test$tree, list_clusters = SCORE_test$Cluster)#plot all clustering bars
#plot one stable optimal clustering bar
plot_optimal_CORE(original_tree= SCORE_test$tree,
                  optimal_cluster = unlist(SCORE_test$Cluster[SCORE_test$optimal_index]), shift = -100
```

3.4 Compare clustering results with other dimensional reduction methods (e.g., CIDR)

```
library(cidr)
t <- CIDR_scGPS(expression.matrix=assay(mixedpop2))
p2 <-plotReduced_scGPS(t, color_fac = factor(colData(mixedpop2)[,1]),palletes =1:length(unique(colData(m
p2
```

3.5 Find gene markers and annotate clusters

```

#load gene list (this can be any lists of user-selected genes)
genes <-GeneList
genes <-genes$Merged_unique

#the gene list can also be objectively identified by differential expression analysis
#cluster information is requied for findMarkers_scGPS. Here, we use CORE results.

colData(mixedpop2)[,1] <- unlist(SCORE_test$Cluster[SCORE_test$optimal_index])

suppressMessages(library(locfit))
suppressMessages(library(DESeq))

DEgenes <- findMarkers_scGPS(expression_matrix=assay(mixedpop2), cluster = colData(mixedpop2)[,1],
                             selected_cluster=unique(colData(mixedpop2)[,1]))
#> [1] "Start estimate dispersions for cluster 1..."
#> [1] "Done estimate dispersions. Start nbinom test for cluster 1..."
#> [1] "Done nbinom test for cluster 1 ..."
#> [1] "Adjust foldchange by subtracting basemean to 1..."
#> [1] "Start estimate dispersions for cluster 2..."
#> [1] "Done estimate dispersions. Start nbinom test for cluster 2..."
#> [1] "Done nbinom test for cluster 2 ..."
#> [1] "Adjust foldchange by subtracting basemean to 1..."
#> [1] "Start estimate dispersions for cluster 3..."
#> [1] "Done estimate dispersions. Start nbinom test for cluster 3..."
#> [1] "Done nbinom test for cluster 3 ..."
#> [1] "Adjust foldchange by subtracting basemean to 1..."
#> [1] "Start estimate dispersions for cluster 4..."
#> [1] "Done estimate dispersions. Start nbinom test for cluster 4..."
#> [1] "Done nbinom test for cluster 4 ..."
#> [1] "Adjust foldchange by subtracting basemean to 1..."

#the output contains dataframes for each cluster.
#the data frame contains all genes, sorted by p-values
names(DEgenes)
#> [1] "DE_Subpop1vsRemaining" "DE_Subpop2vsRemaining" "DE_Subpop3vsRemaining"
#> [4] "DE_Subpop4vsRemaining"

#you can annotate the identified clusters
DEgeneList_3vsOthers <- DEgenes$DE_Subpop3vsRemaining$id

#users need to check the format of the gene input to make sure they are consistent to
#the gene names in the expression matrix
DEgeneList_3vsOthers <-gsub("_.*", "", DEgeneList_3vsOthers )

#the following command saves the file "PathwayEnrichment.xlsx" to the working dir
#use 500 top DE genes
suppressMessages(library(DOSE))
suppressMessages(library(ReactomePA))
suppressMessages(library(clusterProfiler))
enrichment_test <- annotate_scGPS(DEgeneList_3vsOthers[1:500], pvalueCutoff=0.05, gene_symbol=TRUE)
#> [1] "Original gene number in geneList"
#> [1] 500
#> [1] "Number of genes successfully converted"
#> [1] 486

#the enrichment outputs can be displayed by running
dotplot(enrichment_test, showCategory=15)
```



4. Relationship between clusters within one sample or between two samples

The purpose of this workflow is to solve the following task: given one or two unknown mixed population(s) and clusters in each mixed population, estimate and visualise relationship between clusters

4.1 Start the scGPS prediction to find relationship between clusters

```
#select a subpopulation, and input gene list
c_selectID <- 1
genes = DEgenes$DE_Subpop1vsRemaining$id[1:500]
#format gene names
genes <- gsub("_.*", "", genes)

#run the test bootstrap with nboots = 2 runs

cluster_mixedpop1 <- colData(mixedpop1)[,1]
cluster_mixedpop2 <- colData(mixedpop2)[,1]

sink("temp")
LSOLDA_dat <- bootstrap_scGPS(nboots = 2, mixedpop1 = mixedpop1,
    mixedpop2 = mixedpop2, genes = genes, c_selectID  = c_selectID, listData = list(),
     cluster_mixedpop1 = cluster_mixedpop1,
     cluster_mixedpop2 = cluster_mixedpop2)
#>
#> Call:  glmnet(x = as.matrix(dataset[, -which(colnames(dataset) == "Cluster_class")]),      y = y_cat
#>
```

```
#>        Df      %Dev   Lambda
#>   [1,]  0 -2.563e-15 0.283900
#>   [2,]  1  2.069e-02 0.271000
#>   [3,]  1  3.966e-02 0.258700
#>   [4,]  1  5.716e-02 0.246900
#>   [5,]  2  7.787e-02 0.235700
#>   [6,]  3  1.002e-01 0.225000
#>   [7,]  4  1.232e-01 0.214800
#>   [8,]  4  1.455e-01 0.205000
#>   [9,]  5  1.690e-01 0.195700
#>  [10,]  5  1.925e-01 0.186800
#>  [11,]  6  2.148e-01 0.178300
#>  [12,]  7  2.361e-01 0.170200
#>  [13,]  7  2.564e-01 0.162500
#>  [14,]  7  2.754e-01 0.155100
#>  [15,]  8  2.935e-01 0.148000
#>  [16,]  9  3.134e-01 0.141300
#>  [17,]  9  3.328e-01 0.134900
#>  [18,] 10  3.511e-01 0.128800
#>  [19,] 11  3.695e-01 0.122900
#>  [20,] 11  3.869e-01 0.117300
#>  [21,] 11  4.034e-01 0.112000
#>  [22,] 11  4.191e-01 0.106900
#>  [23,] 12  4.346e-01 0.102000
#>  [24,] 12  4.499e-01 0.097400
#>  [25,] 13  4.646e-01 0.092970
#>  [26,] 13  4.799e-01 0.088740
#>  [27,] 13  4.944e-01 0.084710
#>  [28,] 13  5.083e-01 0.080860
#>  [29,] 14  5.215e-01 0.077190
#>  [30,] 15  5.346e-01 0.073680
#>  [31,] 15  5.472e-01 0.070330
#>  [32,] 15  5.592e-01 0.067130
#>  [33,] 15  5.706e-01 0.064080
#>  [34,] 15  5.815e-01 0.061170
#>  [35,] 15  5.920e-01 0.058390
#>  [36,] 15  6.020e-01 0.055730
#>  [37,] 17  6.117e-01 0.053200
#>  [38,] 17  6.211e-01 0.050780
#>  [39,] 18  6.301e-01 0.048470
#>  [40,] 18  6.390e-01 0.046270
#>  [41,] 21  6.481e-01 0.044170
#>  [42,] 22  6.573e-01 0.042160
#>  [43,] 25  6.662e-01 0.040240
#>  [44,] 27  6.751e-01 0.038420
#>  [45,] 28  6.842e-01 0.036670
#>  [46,] 30  6.933e-01 0.035000
#>  [47,] 31  7.024e-01 0.033410
#>  [48,] 31  7.116e-01 0.031890
#>  [49,] 31  7.205e-01 0.030440
#>  [50,] 32  7.291e-01 0.029060
#>  [51,] 35  7.380e-01 0.027740
#>  [52,] 36  7.468e-01 0.026480
#>  [53,] 40  7.555e-01 0.025270
#>  [54,] 40  7.640e-01 0.024130
#>  [55,] 41  7.722e-01 0.023030
#>  [56,] 43  7.802e-01 0.021980
#>  [57,] 46  7.882e-01 0.020980
#>  [58,] 51  7.966e-01 0.020030
#>  [59,] 52  8.050e-01 0.019120
#>  [60,] 56  8.132e-01 0.018250
#>  [61,] 56  8.212e-01 0.017420
#>  [62,] 55  8.288e-01 0.016630
#>  [63,] 56  8.360e-01 0.015870
#>  [64,] 56  8.430e-01 0.015150
#>  [65,] 57  8.496e-01 0.014460
#>  [66,] 57  8.559e-01 0.013810
#>  [67,] 58  8.620e-01 0.013180
#>  [68,] 59  8.679e-01 0.012580
#>  [69,] 60  8.736e-01 0.012010
#>  [70,] 61  8.790e-01 0.011460
#>  [71,] 61  8.843e-01 0.010940
#>  [72,] 62  8.894e-01 0.010440
#>  [73,] 62  8.943e-01 0.009969
#>  [74,] 62  8.990e-01 0.009516
#>  [75,] 63  9.034e-01 0.009083
#>  [76,] 65  9.078e-01 0.008670
#>  [77,] 65  9.119e-01 0.008276
#>  [78,] 66  9.159e-01 0.007900
#>  [79,] 66  9.196e-01 0.007541
#>  [80,] 66  9.232e-01 0.007198
#>  [81,] 66  9.267e-01 0.006871
#>  [82,] 66  9.300e-01 0.006559
#>  [83,] 66  9.331e-01 0.006261
#>  [84,] 67  9.361e-01 0.005976
#>  [85,] 68  9.390e-01 0.005705
#>  [86,] 69  9.417e-01 0.005445
#>  [87,] 69  9.443e-01 0.005198
#>  [88,] 69  9.469e-01 0.004962
#>  [89,] 70  9.493e-01 0.004736
#>  [90,] 72  9.515e-01 0.004521
#>  [91,] 73  9.537e-01 0.004315
#>  [92,] 73  9.558e-01 0.004119
#>  [93,] 73  9.578e-01 0.003932
#>  [94,] 73  9.597e-01 0.003753
#>  [95,] 75  9.616e-01 0.003583
#>  [96,] 75  9.633e-01 0.003420
#>  [97,] 75  9.650e-01 0.003264
#>  [98,] 75  9.665e-01 0.003116
#>  [99,] 75  9.681e-01 0.002974
#> [100,] 75  9.695e-01 0.002839
#> [1] "done bootstrap 1"
#> 
#> Call:  glmnet(x = as.matrix(dataset[, -which(colnames(dataset) == "Cluster_class")]),      y = y_cat
#> 
#>        Df       %Dev   Lambda
#>    [1,]  0 -2.563e-15 0.291900
#>    [2,]  1  2.188e-02 0.278700
#>    [3,]  1  4.198e-02 0.266000
#>    [4,]  3  6.460e-02 0.253900
#>    [5,]  3  9.013e-02 0.242400
#>    [6,]  5  1.152e-01 0.231300
#>    [7,]  6  1.428e-01 0.220800
#>    [8,]  6  1.703e-01 0.210800
#>    [9,]  6  1.961e-01 0.201200
#>   [10,]  6  2.203e-01 0.192100
#>   [11,]  7  2.433e-01 0.183300
#>   [12,]  7  2.650e-01 0.175000
#>   [13,]  8  2.858e-01 0.167000
#>   [14,]  8  3.059e-01 0.159500
#>   [15,]  8  3.250e-01 0.152200
#>   [16,]  9  3.433e-01 0.145300
#>   [17,]  9  3.607e-01 0.138700
#>   [18,] 11  3.781e-01 0.132400
#>   [19,] 12  3.953e-01 0.126400
#>   [20,] 13  4.117e-01 0.120600
#>   [21,] 14  4.277e-01 0.115100
#>   [22,] 14  4.430e-01 0.109900
#>   [23,] 14  4.577e-01 0.104900
#>   [24,] 14  4.717e-01 0.100100
#>   [25,] 16  4.856e-01 0.095590
#>   [26,] 16  4.989e-01 0.091250
#>   [27,] 16  5.117e-01 0.087100
#>   [28,] 16  5.240e-01 0.083140
#>   [29,] 17  5.358e-01 0.079360
#>   [30,] 17  5.472e-01 0.075750
#>   [31,] 17  5.581e-01 0.072310
#>   [32,] 17  5.686e-01 0.069020
#>   [33,] 17  5.787e-01 0.065890
#>   [34,] 18  5.886e-01 0.062890
#>   [35,] 18  5.990e-01 0.060030
#>   [36,] 20  6.095e-01 0.057310
#>   [37,] 21  6.201e-01 0.054700
#>   [38,] 21  6.308e-01 0.052210
#>   [39,] 23  6.418e-01 0.049840
#>   [40,] 24  6.525e-01 0.047580
#>   [41,] 25  6.629e-01 0.045410
#>   [42,] 26  6.728e-01 0.043350
#>   [43,] 27  6.824e-01 0.041380
#>   [44,] 27  6.917e-01 0.039500
#>   [45,] 29  7.010e-01 0.037700
#>   [46,] 29  7.101e-01 0.035990
#>   [47,] 31  7.189e-01 0.034350
#>   [48,] 33  7.278e-01 0.032790
#>   [49,] 36  7.367e-01 0.031300
#>   [50,] 40  7.459e-01 0.029880
#>   [51,] 42  7.555e-01 0.028520
#>   [52,] 43  7.647e-01 0.027220
#>   [53,] 46  7.739e-01 0.025990
#>  [54,] 49  7.834e-01 0.024810
#>  [55,] 49  7.926e-01 0.023680
#>  [56,] 49  8.013e-01 0.022600
#>  [57,] 50  8.097e-01 0.021580
#>  [58,] 51  8.178e-01 0.020590
#>  [59,] 51  8.257e-01 0.019660
#>  [60,] 53  8.334e-01 0.018760
#>  [61,] 52  8.408e-01 0.017910
#>  [62,] 52  8.477e-01 0.017100
#>  [63,] 52  8.544e-01 0.016320
#>  [64,] 52  8.607e-01 0.015580
#>  [65,] 52  8.668e-01 0.014870
#>  [66,] 53  8.726e-01 0.014190
#>  [67,] 54  8.782e-01 0.013550
#>  [68,] 54  8.835e-01 0.012930
#>  [69,] 54  8.886e-01 0.012350
#>  [70,] 53  8.935e-01 0.011780
#>  [71,] 55  8.982e-01 0.011250
#>  [72,] 55  9.027e-01 0.010740
#>  [73,] 55  9.070e-01 0.010250
#>  [74,] 56  9.111e-01 0.009784
#>  [75,] 58  9.150e-01 0.009339
#>  [76,] 58  9.188e-01 0.008915
#>  [77,] 58  9.225e-01 0.008510
#>  [78,] 59  9.259e-01 0.008123
#>  [79,] 59  9.292e-01 0.007754
#>  [80,] 61  9.324e-01 0.007401
#>  [81,] 61  9.354e-01 0.007065
#>  [82,] 60  9.383e-01 0.006744
#>  [83,] 61  9.410e-01 0.006437
#>  [84,] 61  9.436e-01 0.006145
#>  [85,] 64  9.462e-01 0.005865
#>  [86,] 64  9.486e-01 0.005599
#>  [87,] 64  9.509e-01 0.005344
#>  [88,] 65  9.531e-01 0.005101
#>  [89,] 64  9.552e-01 0.004870
#>  [90,] 65  9.572e-01 0.004648
#>  [91,] 66  9.591e-01 0.004437
#>  [92,] 68  9.609e-01 0.004235
#>  [93,] 68  9.627e-01 0.004043
#>  [94,] 68  9.643e-01 0.003859
#>  [95,] 68  9.660e-01 0.003684
#>  [96,] 67  9.675e-01 0.003516
#>  [97,] 65  9.689e-01 0.003356
#>  [98,] 64  9.703e-01 0.003204
#>  [99,] 64  9.717e-01 0.003058
#> [100,] 64  9.729e-01 0.002919
#> [1] "please check the lambda min output ..."
#> [1] "done bootstrap 2"

sink()
```

4.2 Display summary results for the prediction

```
#get the number of rows for the summary matrix
row_cluster <-length(unique(colData(mixedpop2)[,1]))

#summary results LDA to to show the percent of cells classified as cells belonging by LDA classifier
summary_prediction_lda(LSOLDA_dat=LSOLDA_dat, nPredSubpop = row_cluster )
#>                  V1               V2                                names
#> 1             56.25         53.90625 LDA for subpop 1 in target mixedpop2
#> 2 44.1666666666667 46.6666666666667 LDA for subpop 2 in target mixedpop2
#> 3 49.4736842105263 31.5789473684211 LDA for subpop 3 in target mixedpop2
#> 4 44.8275862068966 41.3793103448276 LDA for subpop 4 in target mixedpop2

#summary results Lasso to show the percent of cells classified as cells belonging by Lasso classifier
summary_prediction_lasso(LSOLDA_dat=LSOLDA_dat, nPredSubpop = row_cluster)
#>                  V1               V2
#> 1         52.734375          64.0625
#> 2 51.6666666666667               25
#> 3 61.0526315789474 26.3157894736842
#> 4 48.2758620689655 37.9310344827586
#>                                         names
#> 1 ElasticNet for subpop1 in target mixedpop2
#> 2 ElasticNet for subpop2 in target mixedpop2
#> 3 ElasticNet for subpop3 in target mixedpop2
#> 4 ElasticNet for subpop4 in target mixedpop2

# summary maximum deviance explained by the model during the model training
summary_deviance(object = LSOLDA_dat)
#> $allDeviance
#> [1] "0.6211" "0.966"
#>
#> $DeviMax
#>          Dfd   Deviance        DEgenes
#> 1          0 -2.563e-15 genes_cluster1
#> 2          1    0.04198 genes_cluster1
#> 3          3    0.09013 genes_cluster1
#> 4          5     0.1152 genes_cluster1
#> 5          6     0.2203 genes_cluster1
#> 6          7      0.265 genes_cluster1
#> 7          8      0.325 genes_cluster1
#> 8          9     0.3607 genes_cluster1
#> 9         11     0.3781 genes_cluster1
#> 10        12     0.3953 genes_cluster1
#> 11        13     0.4117 genes_cluster1
#> 12        14     0.4717 genes_cluster1
#> 13        16      0.524 genes_cluster1
#> 14        17     0.5787 genes_cluster1
#> 15        18      0.599 genes_cluster1
#> 16        20     0.6095 genes_cluster1
#> 17        21     0.6308 genes_cluster1
#> 18        23     0.6418 genes_cluster1
#> 19        24     0.6525 genes_cluster1
#> 20        25     0.6629 genes_cluster1
#> 21         26      0.6728 genes_cluster1
#> 22         27      0.6917 genes_cluster1
#> 23         29      0.7101 genes_cluster1
#> 24         31      0.7189 genes_cluster1
#> 25         33      0.7278 genes_cluster1
#> 26         36      0.7367 genes_cluster1
#> 27         40      0.7459 genes_cluster1
#> 28         42      0.7555 genes_cluster1
#> 29         43      0.7647 genes_cluster1
#> 30         46      0.7739 genes_cluster1
#> 31         49      0.8013 genes_cluster1
#> 32         50      0.8097 genes_cluster1
#> 33         51      0.8257 genes_cluster1
#> 34         52      0.8668 genes_cluster1
#> 35         53      0.8935 genes_cluster1
#> 36         54      0.8886 genes_cluster1
#> 37         55       0.907 genes_cluster1
#> 38         56      0.9111 genes_cluster1
#> 39         58      0.9225 genes_cluster1
#> 40         59      0.9292 genes_cluster1
#> 41         60      0.9383 genes_cluster1
#> 42         61      0.9436 genes_cluster1
#> 43         64      0.9729 genes_cluster1
#> 44         65      0.9689 genes_cluster1
#> 45         66      0.9591 genes_cluster1
#> 46         67      0.9675 genes_cluster1
#> 47         68       0.966 genes_cluster1
#> 48 remaining           1        DEgenes
#>
#> $LassoGenesMax
#> NULL

# summary accuracy to check the model accuracy in the leave-out test set
summary_accuracy(object = LSOLDA_dat)
#> [1] 91.96429 84.82143
```

test

```
c_selectID <- 1
genes = DEgenes$DE_Subpop1vsRemaining$id[1:500]
#format gene names
genes <- gsub("_.*", "", genes)

#run the test bootstrap with nboots = 2 runs

cluster_mixedpop1 <- colData(mixedpop1)[,1]
cluster_mixedpop2 <- colData(mixedpop2)[,1]

sink("temp")
LSOLDA_dat <- bootstrap_scGPS(nboots = 2, mixedpop1 = mixedpop2,
    mixedpop2 = mixedpop2, genes = genes, c_selectID  = c_selectID, listData = list(),
     cluster_mixedpop1 = cluster_mixedpop2,
     cluster_mixedpop2 = cluster_mixedpop2)
#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear
#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear
#>
#> Call:  glmnet(x = as.matrix(dataset[, -which(colnames(dataset) == "Cluster_class")]),      y = y_cat
#>
#>        Df       %Dev   Lambda
#>   [1,]  0 -1.922e-15 0.349200
#>   [2,]  1  3.129e-02 0.333400
#>   [3,]  2  6.056e-02 0.318200
#>   [4,]  2  9.059e-02 0.303800
#>   [5,]  2  1.183e-01 0.289900
#>   [6,]  2  1.440e-01 0.276800
#>   [7,]  3  1.679e-01 0.264200
#>   [8,]  4  1.920e-01 0.252200
#>   [9,]  5  2.155e-01 0.240700
#>  [10,]  5  2.380e-01 0.229800
#>  [11,]  5  2.592e-01 0.219300
#>  [12,]  5  2.791e-01 0.209400
#>  [13,]  6  2.980e-01 0.199800
#>  [14,]  7  3.162e-01 0.190800
#>  [15,]  8  3.335e-01 0.182100
#>  [16,]  8  3.500e-01 0.173800
#>  [17,]  8  3.657e-01 0.165900
#>  [18,]  9  3.808e-01 0.158400
#>  [19,]  9  3.953e-01 0.151200
#>  [20,] 10  4.090e-01 0.144300
#>  [21,] 12  4.226e-01 0.137700
#>  [22,] 13  4.358e-01 0.131500
#>  [23,] 14  4.486e-01 0.125500
#>  [24,] 14  4.610e-01 0.119800
#>  [25,] 14  4.729e-01 0.114400
#>  [26,] 14  4.842e-01 0.109200
#>  [27,] 15  4.951e-01 0.104200
#>  [28,] 15  5.056e-01 0.099470
#>  [29,] 15  5.157e-01 0.094940
#>  [30,] 15  5.255e-01 0.090630
#>  [31,] 16  5.350e-01 0.086510
#>  [32,] 17  5.446e-01 0.082580
#>  [33,] 17  5.541e-01 0.078820
#>  [34,] 16  5.633e-01 0.075240
#>  [35,] 18  5.723e-01 0.071820
#>  [36,] 20  5.813e-01 0.068560
#>  [37,] 20  5.903e-01 0.065440
#>  [38,] 22  5.992e-01 0.062470
#>  [39,] 22  6.080e-01 0.059630
#>  [40,] 22  6.166e-01 0.056920
#>  [41,] 23  6.249e-01 0.054330
#>  [42,] 23  6.330e-01 0.051860
#>  [43,] 26  6.408e-01 0.049500
#>  [44,] 27  6.485e-01 0.047250
#>  [45,] 28  6.559e-01 0.045110
#>  [46,] 30  6.636e-01 0.043060
#>  [47,] 31  6.722e-01 0.041100
#>  [48,] 33  6.807e-01 0.039230
#>  [49,] 34  6.889e-01 0.037450
#>  [50,] 34  6.968e-01 0.035750
#>  [51,] 35  7.045e-01 0.034120
#>  [52,] 38  7.122e-01 0.032570
#>  [53,] 38  7.199e-01 0.031090
#>  [54,] 38  7.274e-01 0.029680
#>  [55,] 39  7.347e-01 0.028330
#>  [56,] 39  7.417e-01 0.027040
#>  [57,] 41  7.489e-01 0.025810
#>  [58,] 42  7.559e-01 0.024640
#>  [59,] 42  7.626e-01 0.023520
#>  [60,] 46  7.693e-01 0.022450
#>  [61,] 46  7.757e-01 0.021430
#>  [62,] 48  7.822e-01 0.020460
#>  [63,] 50  7.890e-01 0.019530
#>  [64,] 52  7.963e-01 0.018640
#>  [65,] 55  8.045e-01 0.017790
#>  [66,] 56  8.125e-01 0.016980
#>  [67,] 56  8.203e-01 0.016210
#>  [68,] 58  8.280e-01 0.015470
#>  [69,] 59  8.356e-01 0.014770
#>  [70,] 60  8.430e-01 0.014100
#>  [71,] 61  8.500e-01 0.013460
#>  [72,] 62  8.567e-01 0.012850
#>  [73,] 64  8.632e-01 0.012260
#>  [74,] 62  8.694e-01 0.011710
#>  [75,] 63  8.752e-01 0.011170
#>  [76,] 65  8.808e-01 0.010670
#>  [77,] 65  8.862e-01 0.010180
#>  [78,] 65  8.914e-01 0.009718
#>  [79,] 65  8.963e-01 0.009276
#>  [80,] 66  9.009e-01 0.008855
#>  [81,] 65  9.055e-01 0.008452
#>  [82,] 65  9.098e-01 0.008068
#>  [83,] 64  9.139e-01 0.007701
#>  [84,] 65  9.178e-01 0.007351
#>  [85,] 65  9.216e-01 0.007017
#>  [86,] 65  9.252e-01 0.006698
#>  [87,] 65  9.286e-01 0.006394
#>  [88,] 65  9.318e-01 0.006103
#>  [89,] 64  9.349e-01 0.005826
#>  [90,] 66  9.379e-01 0.005561
#>  [91,] 66  9.407e-01 0.005308
#>  [92,] 67  9.434e-01 0.005067
#>  [93,] 67  9.460e-01 0.004837
#>  [94,] 68  9.485e-01 0.004617
#>  [95,] 69  9.509e-01 0.004407
#>  [96,] 71  9.531e-01 0.004207
#>  [97,] 72  9.553e-01 0.004015
#>  [98,] 72  9.573e-01 0.003833
#>  [99,] 72  9.593e-01 0.003659
#> [100,] 71  9.611e-01 0.003492
#> [1] "please check the lambda min output ..."
#> [1] "done bootstrap 1"
#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear
#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear

#> Warning in lda.default(x, grouping, ...): variables are collinear
#>
#> Call:  glmnet(x = as.matrix(dataset[, -which(colnames(dataset) == "Cluster_class")]),      y = y_cat
#>
#>        Df       %Dev   Lambda
#>   [1,]  0 -1.922e-15 0.347500
#>   [2,]  1  3.098e-02 0.331700
#>   [3,]  1  5.935e-02 0.316600
#>   [4,]  2  8.773e-02 0.302200
#>   [5,]  2  1.161e-01 0.288500
#>   [6,]  4  1.427e-01 0.275400
#>   [7,]  4  1.689e-01 0.262900
#>   [8,]  4  1.934e-01 0.250900
#>   [9,]  4  2.164e-01 0.239500
#>  [10,]  6  2.392e-01 0.228600
#>  [11,]  6  2.609e-01 0.218200
#>  [12,]  7  2.815e-01 0.208300
#>  [13,]  7  3.013e-01 0.198800
#>  [14,]  7  3.200e-01 0.189800
#>  [15,]  7  3.378e-01 0.181200
#>  [16,]  8  3.547e-01 0.172900
#>  [17,]  8  3.708e-01 0.165100
#>  [18,]  9  3.861e-01 0.157600
#>  [19,] 10  4.012e-01 0.150400
#>  [20,] 13  4.157e-01 0.143600
#>  [21,] 13  4.303e-01 0.137100
#>  [22,] 14  4.443e-01 0.130800
#>  [23,] 17  4.580e-01 0.124900
#>  [24,] 16  4.710e-01 0.119200
#>  [25,] 17  4.834e-01 0.113800
#>  [26,] 17  4.955e-01 0.108600
#>  [27,] 17  5.070e-01 0.103700
#>  [28,] 18  5.181e-01 0.098960
#>  [29,] 17  5.288e-01 0.094460
#>  [30,] 19  5.394e-01 0.090170
#>  [31,] 20  5.499e-01 0.086070
#>  [32,] 20  5.601e-01 0.082160
#>  [33,] 20  5.700e-01 0.078430
#>  [34,] 20  5.795e-01 0.074860
#>  [35,] 20  5.887e-01 0.071460
#>  [36,] 19  5.975e-01 0.068210
#>  [37,] 20  6.061e-01 0.065110
#>  [38,] 20  6.144e-01 0.062150
#>  [39,] 20  6.225e-01 0.059330
#>  [40,] 21  6.302e-01 0.056630
#>  [41,] 21  6.382e-01 0.054060
#>  [42,] 21  6.459e-01 0.051600
#>  [43,] 21  6.533e-01 0.049250
#>  [44,] 23  6.607e-01 0.047010
#>  [45,] 25  6.687e-01 0.044880
#>  [46,] 26  6.769e-01 0.042840
#>  [47,] 27  6.857e-01 0.040890
#>  [48,] 30  6.947e-01 0.039030
#>  [49,] 30  7.037e-01 0.037260
#>  [50,] 31  7.124e-01 0.035560
#>  [51,] 31  7.209e-01 0.033950
#>  [52,] 34  7.294e-01 0.032410
#>  [53,] 34  7.379e-01 0.030930
#>  [54,] 34  7.465e-01 0.029530
#>  [55,] 35  7.549e-01 0.028180
#>  [56,] 35  7.630e-01 0.026900
#>  [57,] 36  7.708e-01 0.025680
#>  [58,] 36  7.783e-01 0.024510
#>  [59,] 38  7.856e-01 0.023400
#>  [60,] 39  7.927e-01 0.022340
#>  [61,] 39  7.998e-01 0.021320
#>  [62,] 39  8.065e-01 0.020350
#>  [63,] 38  8.130e-01 0.019430
#>  [64,] 38  8.192e-01 0.018540
#>  [65,] 38  8.252e-01 0.017700
#>  [66,] 38  8.310e-01 0.016900
#>  [67,] 40  8.367e-01 0.016130
#>  [68,] 41  8.424e-01 0.015400
#>  [69,] 41  8.480e-01 0.014700
#>  [70,] 42  8.535e-01 0.014030
#>  [71,] 43  8.590e-01 0.013390
#>  [72,] 45  8.643e-01 0.012780
#>  [73,] 46  8.694e-01 0.012200
#>  [74,] 47  8.746e-01 0.011650
#>  [75,] 49  8.798e-01 0.011120
#>  [76,] 49  8.849e-01 0.010610
#>  [77,] 50  8.897e-01 0.010130
#>  [78,] 51  8.945e-01 0.009669
#>  [79,] 52  8.990e-01 0.009229
#>  [80,] 52  9.034e-01 0.008810
#>  [81,] 52  9.075e-01 0.008409
#>  [82,] 54  9.114e-01 0.008027
#>  [83,] 54  9.152e-01 0.007662
#>  [84,] 54  9.189e-01 0.007314
#>  [85,] 55  9.224e-01 0.006982
#>  [86,] 57  9.258e-01 0.006664
#>  [87,] 58  9.292e-01 0.006361
#>  [88,] 60  9.324e-01 0.006072
#>  [89,] 61  9.354e-01 0.005796
#>  [90,] 62  9.384e-01 0.005533
#>  [91,] 62  9.412e-01 0.005281
#>  [92,] 61  9.438e-01 0.005041
#>  [93,] 61  9.464e-01 0.004812
#>  [94,] 60  9.488e-01 0.004593
#>  [95,] 61  9.511e-01 0.004385
#>  [96,] 61  9.534e-01 0.004185
#>  [97,] 62  9.555e-01 0.003995
#>  [98,] 62  9.575e-01 0.003814
#>  [99,] 63  9.595e-01 0.003640
#> [100,] 63  9.613e-01 0.003475
#> [1] "please check the lambda min output ..."
#> [1] "done bootstrap 2"

sink()
```

4.3 Plot the relationship between clusters in one sample

Here we look at one example use case to find relationship between clusters within one sample or between two sample

```
#run prediction for 3 clusters
cluster_mixedpop1 <- colData(mixedpop1)[,1]
cluster_mixedpop2 <- as.numeric(as.vector(colData(mixedpop2)[,1]))

c_selectID <- 1
genes = DEgenes$DE_Subpop1vsRemaining$id[1:200] #top 200 gene markers distinguishing cluster 1


LSOLDA_dat1 <- bootstrap_scGPS(nboots = 2, mixedpop1 = mixedpop2, mixedpop2 = mixedpop2, genes=genes, c

c_selectID <- 2
genes = DEgenes$DE_Subpop2vsRemaining$id[1:200]

LSOLDA_dat2 <- bootstrap_scGPS(nboots = 2,mixedpop1 = mixedpop2, mixedpop2 = mixedpop2, genes=genes, c_
     cluster_mixedpop2 = cluster_mixedpop2)

c_selectID <- 3
genes = DEgenes$DE_Subpop3vsRemaining$id[1:200]
#genes <- gsub("_.*", "", genes)
LSOLDA_dat3 <- bootstrap_scGPS(nboots = 2,mixedpop1 = mixedpop2, mixedpop2 = mixedpop2, genes=genes, c_
     cluster_mixedpop2 = cluster_mixedpop2)

c_selectID <- 4
genes = DEgenes$DE_Subpop4vsRemaining$id[1:200]
#genes <- gsub("_.*", "", genes)
LSOLDA_dat4 <- bootstrap_scGPS(nboots = 2,mixedpop1 = mixedpop2, mixedpop2 = mixedpop2, genes=genes, c_
     cluster_mixedpop2 = cluster_mixedpop2)
```

```r
#prepare table input for sankey plot

LASSO_C1S2  <- reformat_LASSO(c_selectID=1, mp_selectID = 2, LSOLDA_dat=LSOLDA_dat1,
                           nPredSubpop = length(unique(colData(mixedpop2)[,1])),
                           Nodes_group ="#7570b3")

LASSO_C2S2  <- reformat_LASSO(c_selectID=2, mp_selectID =2, LSOLDA_dat=LSOLDA_dat2,
                           nPredSubpop = length(unique(colData(mixedpop2)[,1])),
                           Nodes_group ="#1b9e77")

LASSO_C3S2  <- reformat_LASSO(c_selectID=3, mp_selectID =2, LSOLDA_dat=LSOLDA_dat3,
                           nPredSubpop = length(unique(colData(mixedpop2)[,1])),
                           Nodes_group ="#e7298a")

LASSO_C4S2  <- reformat_LASSO(c_selectID=4, mp_selectID =2, LSOLDA_dat=LSOLDA_dat4,
                           nPredSubpop = length(unique(colData(mixedpop2)[,1])),
                           Nodes_group ="#00FFFF")

combined <- rbind(LASSO_C1S2,LASSO_C2S2,LASSO_C3S2, LASSO_C4S2 )
combined <- combined[is.na(combined$Value) != TRUE,]

nboots = 2
#links: source, target, value
#source: node, nodegroup
combined_D3obj <-list(Nodes=combined[,(nboots+3):(nboots+4)], Links=combined[,c((nboots+2):(nboots+1),nc

library(networkD3)

Node_source <- as.vector(sort(unique(combined_D3obj$Links$Source)))
Node_target <- as.vector(sort(unique(combined_D3obj$Links$Target)))
Node_all <-unique(c(Node_source, Node_target))

#assign IDs for Source (start from 0)
Source <-combined_D3obj$Links$Source
Target <- combined_D3obj$Links$Target

for(i in 1:length(Node_all)){
  Source[Source==Node_all[i]] <-i-1
  Target[Target==Node_all[i]] <-i-1
}

combined_D3obj$Links$Source <- as.numeric(Source)
combined_D3obj$Links$Target <- as.numeric(Target)
combined_D3obj$Links$LinkColor <- combined$NodeGroup

#prepare node info
node_df <-data.frame(Node=Node_all)
node_df$id <-as.numeric(c(0, 1:(length(Node_all)-1)))

suppressMessages(library(dplyr))
Color <- combined %>% count(Node, color=NodeGroup) %>% select(2)
node_df$color <- Color$color
```

```
suppressMessages(library(networkD3))
p1<-sankeyNetwork(Links =combined_D3obj$Links, Nodes = node_df,  Value = "Value", NodeGroup ="color", Li
                  fontSize = 22 )
p1

#saveNetwork(p1, file = paste0(path,'Subpopulation_Net.html'))
##R Setting Information
#sessionInfo()
#rmarkdown::render("/Users/quan.nguyen/Documents/Powell_group_MacQuan/AllCodes/scGPS/vignettes/vignette
#rmarkdown::render("/Users/quan.nguyen/Documents/Powell_group_MacQuan/AllCodes/scGPS/vignettes/vignette
```

4.3 Plot the relationship between clusters in two samples

Here we look at one example use case to find relationship between clusters within one sample or between two sample

```
#run prediction for 3 clusters
cluster_mixedpop1 <- colData(mixedpop1)[,1]
cluster_mixedpop2 <- as.numeric(as.vector(colData(mixedpop2)[,1]))
row_cluster <-length(unique(colData(mixedpop2)[,1]))

c_selectID <- 1
genes = DEgenes$DE_Subpop1vsRemaining$id[1:200] #top 200 gene markers distinguishing cluster 1
genes <- gsub("_.*", "", genes)

LSOLDA_dat1 <- bootstrap_scGPS(nboots = 1, mixedpop1 = mixedpop1, mixedpop2 = mixedpop2, genes=genes, c_


c_selectID <- 2
genes = DEgenes$DE_Subpop2vsRemaining$id[1:200]
genes <- gsub("_.*", "", genes)
LSOLDA_dat2 <- bootstrap_scGPS(nboots = 1,mixedpop1 = mixedpop1, mixedpop2 = mixedpop2, genes=genes, c_s
     cluster_mixedpop2 = cluster_mixedpop2)

c_selectID <- 3
genes = DEgenes$DE_Subpop3vsRemaining$id[1:200]
genes <- gsub("_.*", "", genes)
LSOLDA_dat3 <- bootstrap_scGPS(nboots = 1,mixedpop1 = mixedpop1, mixedpop2 = mixedpop2, genes=genes, c_s
     cluster_mixedpop2 = cluster_mixedpop2)

#prepare table input for sankey plot

LASSO_C1S1  <- reformat_LASSO(c_selectID=1, mp_selectID = 1, LSOLDA_dat=LSOLDA_dat1,
                          nPredSubpop = row_cluster, Nodes_group = "#7570b3")

LASSO_C2S1  <- reformat_LASSO(c_selectID=2, mp_selectID = 1, LSOLDA_dat=LSOLDA_dat2,
                          nPredSubpop = row_cluster, Nodes_group = "#1b9e77")

LASSO_C3S1  <- reformat_LASSO(c_selectID=3, mp_selectID = 1, LSOLDA_dat=LSOLDA_dat3,
                          nPredSubpop = row_cluster, Nodes_group = "#e7298a")


combined <- rbind(LASSO_C1S1,LASSO_C2S1,LASSO_C3S1)
combined <- combined[is.na(combined$Value) != TRUE,]
combined_D3obj <-list(Nodes=combined[,4:5], Links=combined[,c(3,2,1)])

library(networkD3)

Node_source <- as.vector(sort(unique(combined_D3obj$Links$Source)))
Node_target <- as.vector(sort(unique(combined_D3obj$Links$Target)))
Node_all <-unique(c(Node_source, Node_target))

#assign IDs for Source (start from 0)
Source <-combined_D3obj$Links$Source
Target <- combined_D3obj$Links$Target

for(i in 1:length(Node_all)){
  Source[Source==Node_all[i]] <-i-1
  Target[Target==Node_all[i]] <-i-1
}

combined_D3obj$Links$Source <- as.numeric(Source)
combined_D3obj$Links$Target <- as.numeric(Target)
combined_D3obj$Links$LinkColor <- combined$NodeGroup

#prepare node info
node_df <-data.frame(Node=Node_all)
node_df$id <-as.numeric(c(0, 1:(length(Node_all)-1)))

suppressMessages(library(dplyr))
Color <- combined %>% count(Node, color=NodeGroup) %>% select(2)

n <- length(unique(node_df$Node))
Color = RColorBrewer::brewer.pal(n,"Set2")

node_df$color <- Color

suppressMessages(library(networkD3))
p1<-sankeyNetwork(Links =combined_D3obj$Links, Nodes = node_df,  Value = "Value", NodeGroup ="color", Li
                  fontSize = 22 )
p1

#saveNetwork(p1, file = paste0(path,'Subpopulation_Net.html'))
##R Setting Information
#sessionInfo()
#rmarkdown::render("/Users/quan.nguyen/Documents/Powell_group_MacQuan/AllCodes/scGPS/vignettes/vignette
#rmarkdown::render("/Users/quan.nguyen/Documents/Powell_group_MacQuan/AllCodes/scGPS/vignettes/vignette
```

4.4 Annotation: scGPS prediction can be used to compare scGPS clusters with a reference dataset to see which cluster is most similar to the reference