Overview

PhEMD is a package for comparing multiple heterogeneous single-cell samples to one another. It currently does so by first defining cell subtypes and relating them to one another using Monocle 2. It then computes Earth Mover's Distance (EMD) between each pair of samples, incorporating information about intrinsic cell subtype dissimilarity (i.e. manifold distance between cell subtypes) and differences between samples with respect to relative abundance of each cell subtype. PhEMD uses these pairwise distances to construct a network of single-cell samples that may be visually related in 2D or 3D using a diffusion map and partitioned to identify groups of similar samples.

1. Installation

PhEMD requires R version >= 3.4.0 (recommended 3.5.0+), Bioconductor version >= 3.5 (recommended 3.7+), and Monocle 2 version >= 2.4.0 (recommended 2.8.0)

###Install from Github

```
library(devtools)
install_github("wschen/phemd")
```

###Install from Bioconductor

```
BiocManager::install("phemd")
```

###Load library after installation

```
library('phemd')
library('monocle')
```

2. Preparing data for cell state definition and embedding

PhEMD expects single-cell data to be represented as an R list of samples. Each sample (i.e. list element) is expected to be a matrix of dimension *num_cells* x *num_markers*, where markers may represent genes or cytometry protein markers. For this vignette, we will be demonstrating our analysis pipeline on a melanoma dataset consisting of tumor-infiltrating immune cell scRNA-seq data (selected genes) that were log-transformed following TPM-normalization (first published by Tirosh et al., 2016).

We first start by creating a PhEMD data object, specifying the multi-sample expression data (R list), marker names (i.e. column names of the data matrices in the list of expression data), and sample names (in the order they appear in the list of expression data).

```
load('melanomaData.RData')
myobj <- createDataObj(all_expn_data, all_genes, as.character(snames))
```

We can optionally remove samples in the PhEMD data object that have fewer than min_sz number of cells as follows:

```
myobj <- removeTinySamples(myobj, min_sz = 20)
```

```
## [1] "Mel78 removed because only contains 3 cells"
## [1] "Mel59 removed because only contains 12 cells"
```

Note that samples that don't meet the meet the minimum cell yield criteria are removed from rawExpn(myobj) and from the list of sample names in sampleNames(myobj).

Next, aggregate data from all samples into a matrix that is stored in the PhEMD data object (in slot 'data_aggregate'). This aggregated data will then be used for initial cell subtype definition and embedding. If there are more cells collectively in all samples than max_cells, an equal number of cells from each sample will be subsampled and stored in pooledCells(myobj).

```
myobj <- aggregateSamples(myobj, max_cells=12000)
```

3. Generate Monocle 2 cell embedding with cell state definitions

Now that we have aggregated single-cell data from all samples, we are ready to perform cell subtype definition and dimensionality reduction to visually and spatially relate cells and cell subtypes. For this, we use Monocle 2. Before we begin, we first perform feature selection by selecting 44 important genes. Suggestions on how to choose important genes can be found here: http://cole-trapnell-lab.github.io/monocle-release/docs/#trajectory-step-1-choose-genes-that-define-a-cell-s-progress

```
myobj <- selectFeatures(myobj, selected_genes)
```

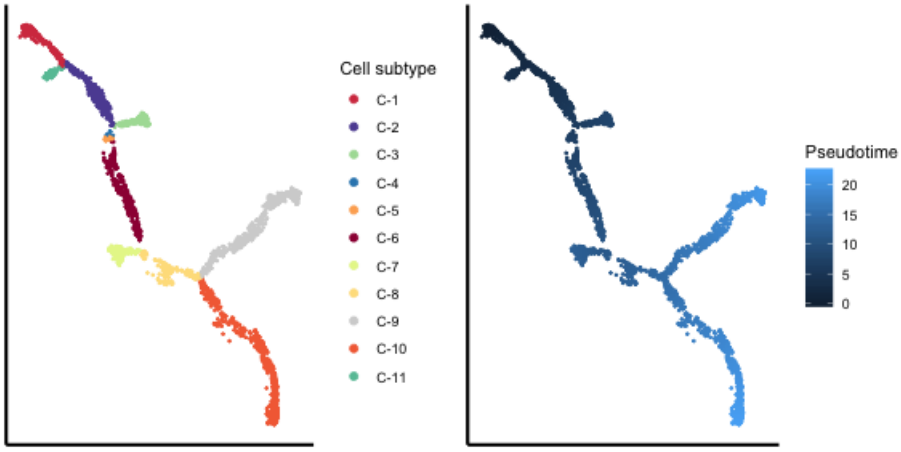
We are now ready to generate a Monocle 2 embedding. Our *embedCells()* function is a wrapper function for the *reduceDimension()* function in Monocle 2. For our example dataset, we specify the expression distribution model as *'gaussianff'* as is recommended in the Monocle 2 tutorial for log-transformed scRNA-seq TPM values (http://cole-trapnell-lab.github.io/monocle-release/docs/#choosing-a-distribution-for-your-data-required). *'negbinomial_sz'* is the recommended data type for most unnormalized scRNA-seq data (raw read counts) and *'gaussianff'* is recommended for log-transformed data or arcsin-transformed mass cytometry data. See above link for more details.

Additional parameters may be passed to Monocle 2 *reduceDimension()* as optional named parameters in *embed_cells()*. We found that Monocle 2 is robust to a range of parameters. Sigma can be thought of as a "noise" parameter and we empirically found that sigma in the range of [0.01, 0.1] often works well for log-transformed scRNA-seq data or normalized CyTOF data. Greater values of sigma generally result in fewer total number of clusters. See Monocle 2 publication (Qiu et al., 2017) for additional details on parameter selection.

```
# generate 2D cell embedding and cell subtype assignments
myobj <- embedCells(myobj, data_model = 'gaussianff', pseudo_expr=0, sigma=0.02)
# generate pseudotime ordering of cells
myobj <- orderCellsMonocle(myobj)
```

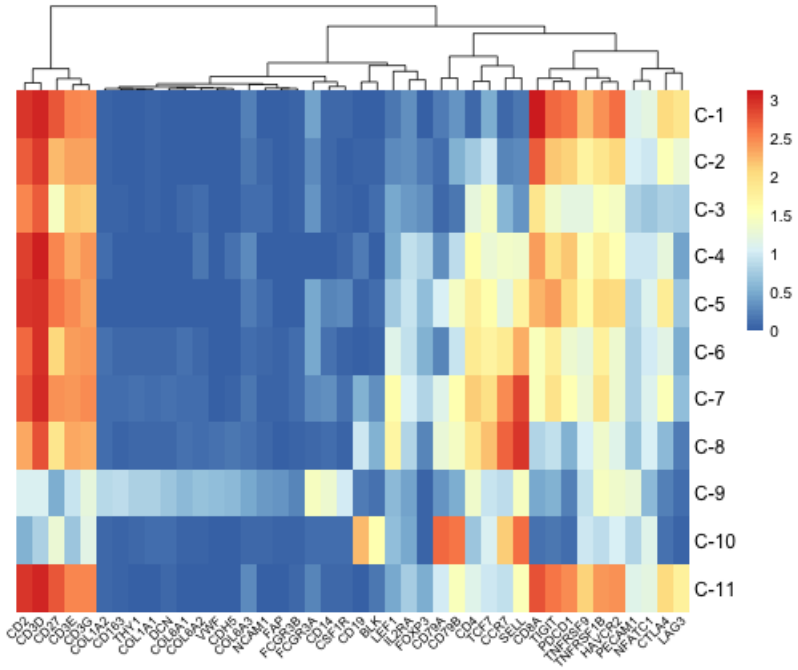
The result of the code above is a Monocle 2 object stored in *monocleInfo(myobj)*. This object contains cell subtype and pseudotime assignments for each cell in the aggregated data matrix (stored in *pooledCells(myobj)*). A 2D embedding of these cells has also been generated. We can visualize the embedding by writing them to file in this way:

```
savedest <- './example_output/'
dir.create(savedest, showWarnings = FALSE) # create folder for output
cmap <- plotEmbeddings(myobj, path=NULL, cell_model='monocle2')
```



To visualize the expression profiles of the cell subtypes, we can plot a heatmap and save to file as such:

```
plotHeatmaps(myobj, path=NULL, cell_model='monocle2',
             selected_genes=heatmap_genes)
```



4. Deconvolute single-cell samples and compare using Earth Mover's Distance

Now that we have identified a comprehensive set of cell subtypes across all single-cell samples and related them in a low-dimensional embedding by aggregating cells from all samples, we want to perform deconvolution to determine the abundance of each cell subtype on a per sample basis. To do so, we call this function:

```
# Determine cell subtype breakdown of each sample
myobj <- clusterIndividualSamples(myobj)
```

The results of this process are stored in *celltypeFreqs(myobj)*. Row *i* column *j* represents the fraction of all cells in sample *i* assigned to cell subtype *j*.

To compare single-cell samples, we use Earth Mover's Distance, which is a metric that takes into account both the difference in relative frequencies of matching cell subtypes (e.g. % of all cells in each sample that are CD8+ T-cells) and the dissimilarity of the cell subtypes themselves (e.g. intrinsic dissimilarity between CD8+ and CD4+ T-cells). To compute the intrinsic dissimilarity between cell subtypes, we call the following function:

```
# Determine (dis)similarity of different cell subtypes
myobj <- generateGDM(myobj)
```

generateGDM() stores the pairwise dissimilarity (i.e. “ground-distance” or “tree-distance”) between cell subtypes in *GDM(myobj)*.

We are now ready to compare single-cell samples using EMD. To do so, we simply call the function *compareSamples()*:

```
# Perform inter-sample comparisons using EMD
my_distmat <- compareSamples(myobj)
```

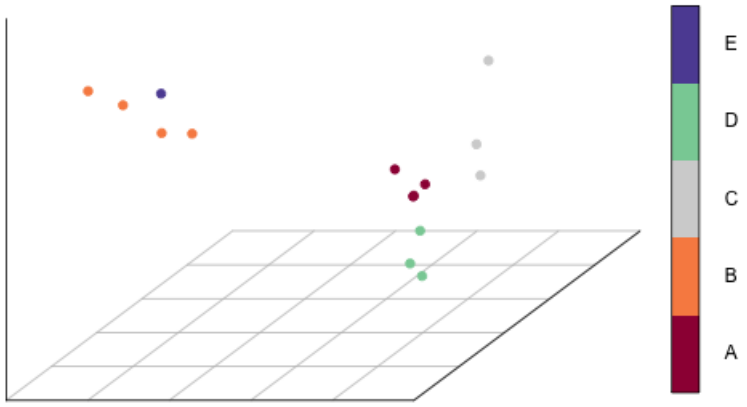
compareSamples() returns a distance matrix representing the pairwise EMD between single-cell samples; *my_distmat[i,j]* represents the dissimilarity between samples *i* and *j* (i.e. samples represented by rows *i* and *j* in celltypeFreqs(myobj)). We can use this distance matrix to identify similar groups of samples as such:

```
## Identify similar groups of inhibitors (hierarchical clustering for now)
group_assignments <- groupSamples(my_distmat, distfun = 'hclust', ncluster=5)
# Write inhibitor groups to file
printClusterAssignments(group_assignments, myobj, savedest)
```

5. Visualize single-cell samples based on PhEMD-based similarity

We can also use the PhEMD-based distance matrix to generate an embedding of single-cell samples, colored by group assignments.

```
dmap_obj <- plotGroupedSamplesDmap(my_distmat, group_assignments, dest=NULL,
                                   pt_sz = 1.5)
```



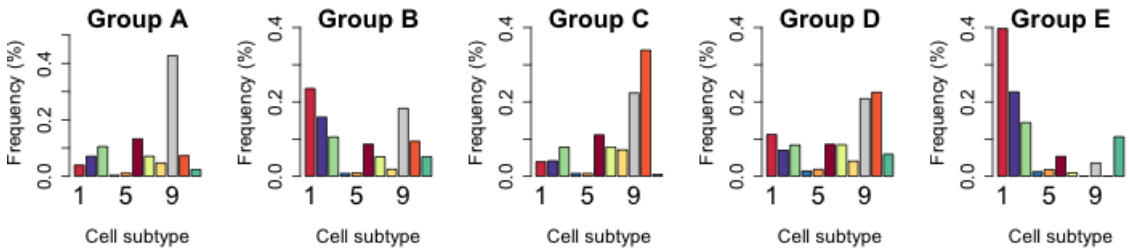
To plot the cell subtype distribution on a per-sample basis, use the following plotting function.

```
# Plot cell subtype distribution (histogram) for each sample
saveSampleHistograms(myobj, group_assignments, savedest,
                     cell_model='monocle2', cmap)
```

Note that the *saveSampleHistograms()* function saves each histogram as a separate file in the *savedest* output folder rather than plotting them in a single plot (due to an often large number of samples).

To plot cell subtype histograms summarizing groups of similar samples (bin-wise mean of each cell subtype across all samples assigned to a particular group), use the following plotting function:

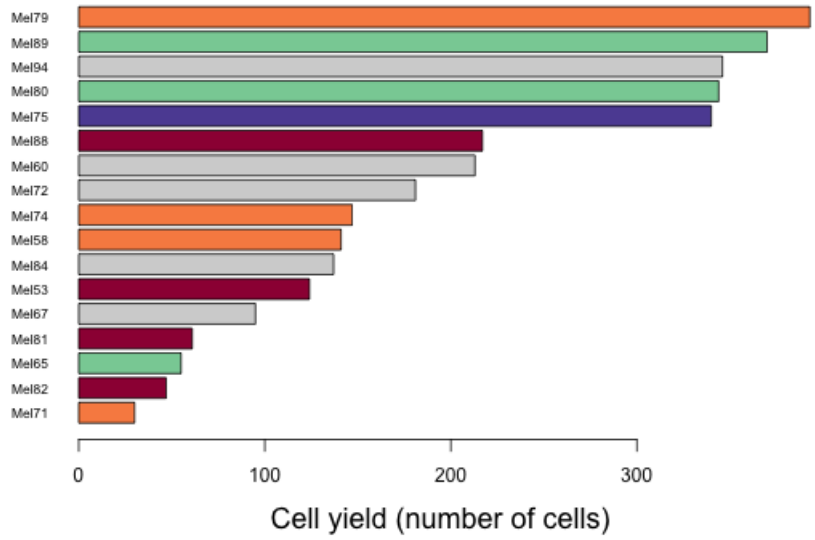
```
# Plot representative cell subtype distribution for each group of samples
plotSummaryHistograms(myobj, group_assignments, savedest,
                      cell_model='monocle2', cmap, ncol.plot = 5,
                     ax.lab.sz=1.5, title.sz=2)
```



Note that the *plotSummaryHistograms()* function optionally saves each histogram as a separate file in the *savedest* output folder.

To plot cell yield of each samples as a barplot, use the following function:

```
# Plot cell yield of each experimental condition
plotCellYield(myobj, dest=NULL, group_assignments, font_sz = 0.7, w=8, h=5)
```




```
sessionInfo()
```

```
## R version 3.5.0 (2018-04-23)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: OS X El Capitan 10.11.6
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib
##
## locale:
## [1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
##  [1] splines   stats4    parallel  stats     graphics  grDevices utils
##  [8] datasets  methods   base
##
## other attached packages:
##  [1] EBImage_4.22.1      phemd_0.99.13       monocle_2.8.0
##  [4] DDRTree_0.1.5       irlba_2.3.2         VGAM_1.0-5
##  [7] ggplot2_3.0.0       Biobase_2.40.0      BiocGenerics_0.26.0
## [10] Matrix_1.2-14
##
## loaded via a namespace (and not attached):
##   [1] reticulate_1.10             R.utils_2.6.0
##   [3] tidyselect_0.2.4            htmlwidgets_1.2
##   [5] grid_3.5.0                  combinat_0.0-8
##   [7] trimcluster_0.1-2.1         docopt_0.6
##   [9] BiocParallel_1.14.2         Rtsne_0.13
##  [11] munsell_0.5.0               destiny_2.10.2
##  [13] codetools_0.2-15            ica_1.0-2
##  [15] withr_2.1.2                 colorspace_1.3-2
##  [17] fastICA_1.2-1               highr_0.7
##  [19] knitr_1.20                  rstudioapi_0.7
##  [21] SingleCellExperiment_1.2.0  Seurat_2.3.4
##  [23] ROCR_1.0-7                  robustbase_0.93-2
##  [25] dtw_1.20-1                  vcd_1.4-4
##  [27] VIM_4.7.0                   TTR_0.23-3
##  [29] gbRd_0.4-11                 labeling_0.3
##  [31] Rdpack_0.9-0                lars_1.2
##  [33] slam_0.1-43                 GenomeInfoDbData_1.1.0
##  [35] bit64_0.9-7                 pheatmap_1.0.10
##  [37] ggthemes_4.0.0              diptest_0.75-7
##  [39] R6_2.2.2                    GenomeInfoDb_1.16.0
```

```
##  [41] RcppEigen_0.3.3.4.0         locfit_1.5-9.1
##  [43] hdf5r_1.0.0                 flexmix_2.3-14
##  [45] bitops_1.0-6                DelayedArray_0.6.4
##  [47] assertthat_0.2.0            SDMTools_1.1-221
##  [49] scales_0.5.0                nnet_7.3-12
##  [51] gtable_0.2.0                rlang_0.2.1
##  [53] scatterplot3d_0.3-41        lazyeval_0.2.1
##  [55] acepack_1.4.1               checkmate_1.8.5
##  [57] reshape2_1.4.3              abind_1.4-5
##  [59] backports_1.1.2             Hmisc_4.1-1
##  [61] tools_3.5.0                 gplots_3.0.1
##  [63] RColorBrewer_1.1-2          proxy_0.4-22
##  [65] ggridges_0.5.0              Rcpp_0.12.18
##  [67] plyr_1.8.4                  base64enc_0.1-3
##  [69] zlibbioc_1.26.0             purrr_0.2.5
##  [71] RCurl_1.95-4.11             densityClust_0.3
##  [73] rpart_4.1-13                pbapply_1.3-4
##  [75] viridis_0.5.1               cowplot_0.9.3
##  [77] S4Vectors_0.18.3            zoo_1.8-3
##  [79] SummarizedExperiment_1.10.1 haven_1.1.2
##  [81] ggrepel_0.8.0               cluster_2.0.7-1
##  [83] magrittr_1.5                data.table_1.11.4
##  [85] openxlsx_4.1.0              lmtest_0.9-36
##  [87] RANN_2.6                    mvtnorm_1.0-8
##  [89] fitdistrplus_1.0-9          matrixStats_0.54.0
##  [91] fftwtools_0.9-8             hms_0.4.2
##  [93] evaluate_0.11               smoother_1.1
##  [95] jpeg_0.1-8                  rio_0.5.10
##  [97] mclust_5.4.1                sparsesvd_0.1-4
##  [99] readxl_1.1.0                IRanges_2.14.10
## [101] gridExtra_2.3               HSMMSingleCell_0.114.0
## [103] compiler_3.5.0              transport_0.9-4
## [105] tibble_1.4.2                KernSmooth_2.23-15
## [107] crayon_1.3.4                R.oo_1.22.0
## [109] htmltools_0.3.6             tiff_0.1-5
## [111] segmented_0.5-3.0           Formula_1.2-3
## [113] snow_0.4-2                  tidyr_0.8.1
## [115] MASS_7.3-50                 fpc_2.1-11.1
## [117] boot_1.3-20                 car_3.0-0
## [119] R.methodsS3_1.7.1           gdata_2.18.0
## [121] metap_1.0                   bindr_0.1.1
## [123] igraph_1.2.2                GenomicRanges_1.32.6
## [125] forcats_0.3.0               pkgconfig_2.0.1
## [127] foreign_0.8-71              laeken_0.4.6
## [129] sp_1.3-1                    foreach_1.4.4
## [131] XVector_0.20.0              bibtex_0.4.2
## [133] stringr_1.3.1               digest_0.6.15
## [135] tsne_0.1-3                  pracma_2.1.4
## [137] cellranger_1.1.0            htmlTable_1.12
## [139] curl_3.2                    kernlab_0.9-26
## [141] gtools_3.8.1                modeltools_0.2-22
## [143] jsonlite_1.5                nlme_3.1-137
## [145] bindrcpp_0.2.2              carData_3.0-1
## [147] viridisLite_0.3.0           limma_3.36.2
## [149] pillar_1.3.0                lattice_0.20-35
## [151] httr_1.3.1                  DEoptimR_1.0-8
## [153] survival_2.42-6             glue_1.3.0
## [155] xts_0.11-0                  qlcMatrix_0.9.7
## [157] zip_1.0.0                   FNN_1.1.2
## [159] png_0.1-7                   prabclus_2.2-6
## [161] iterators_1.0.10            bit_1.1-14
## [163] class_7.3-14                stringi_1.2.4
## [165] mixtools_1.1.0              doSNOW_1.0.16
## [167] latticeExtra_0.6-28         caTools_1.17.1.1
## [169] dplyr_0.7.6                 e1071_1.7-0
## [171] maptree_1.4-7               ape_5.1
```

```
## [171] maptree_1.4-7                ape_5.1
```