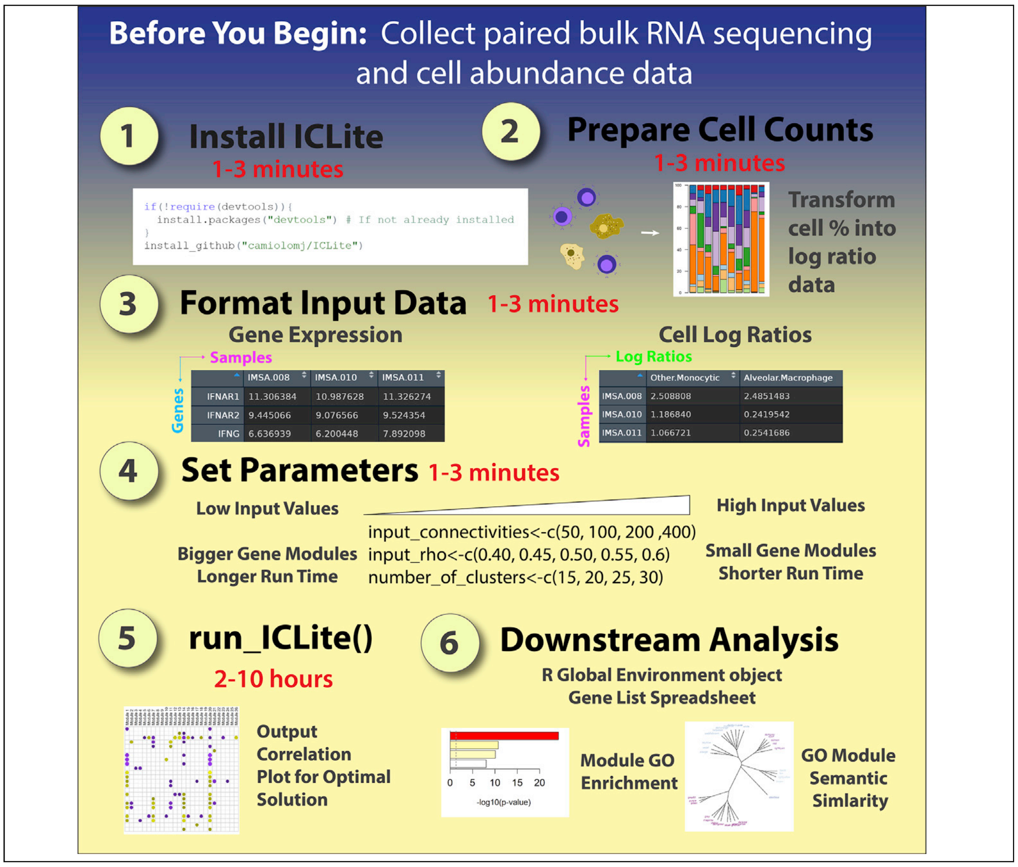
Protocol

Using ICLite for deconvolution of bulk transcriptional data from mixed cell populations



```
if(!require(devtools)){
  install.packages("devtools") # If not already installed
}
install_github("camiolomj/ICLite")
```

	IMSA.008	IMSA.010	IMSA.011
IFNAR1	11.306384	10.987628	11.326274
IFNAR2	9.445066	9.076566	9.524354
IFNG	6.636939	6.200448	7.892098

	Other.Monocytic	Alveolar.Macrophage
IMSA.008	2.508808	2.4851483
IMSA.010	1.186840	0.2419542
IMSA.011	1.066721	0.2541686

```
input_connectivities<-c(50, 100, 200 ,400)
input_rho<-c(0.40, 0.45, 0.50, 0.55, 0.6)
number_of_clusters<-c(15, 20, 25, 30)
```

Matthew J. Camiolo, Sally E. Wenzel, Anuradha Ray

camiolomj@upmc.edu

Highlights

ICLite identifies gene modules in bulk transcriptional data from mixed cell populations

Protocol details how to run and interpret the results of ICLite

Discussion of parameter tuning, data formatting, and solution evaluation

Details for post-run exploration including gene ontology and semantic similarity

Bulk expression data from heterogeneous cell populations pose a challenge for investigators, as differences in cell numbers and transcriptional programs may complicate analysis. To improve the performance of bulk RNA sequencing on mixed populations, we created Immune Cell Linkage through Exploratory Matrices (ICLite). The ICLite package for R constructs modules of correlated genes and identifies their relationship to specific lineages in mixed cell populations. This protocol details formatting, optimization of run parameters, and interpretation of results following implementation of ICLite.

Camiolo et al., STAR Protocols 2, 100847
December 17, 2021 © 2021 The Authors.
https://doi.org/10.1016/j.xpro.2021.100847

Protocol

Using ICLite for deconvolution of bulk transcriptional data from mixed cell populations

Matthew J. Camiolo,[1,2,5,6,*] Sally E. Wenzel,[1,3,4] and Anuradha Ray[1,4]

[1]Division of Pulmonary, Allergy, and Critical Care Medicine, Department of Medicine, University of Pittsburgh School of Medicine, Pittsburgh, PA 15213, USA

[2]Center for Systems Immunology, University of Pittsburgh Medical Center, Pittsburgh, PA 15213, USA

[3]Department of Environmental Medicine and Occupational Health, Graduate School of Public Health, University of Pittsburgh School of Medicine, Pittsburgh, PA 15213, USA

[4]Department of Immunology, University of Pittsburgh School of Medicine, Pittsburgh, PA 15213, USA

[5]Technical contact

[6]Lead contact

*Correspondence: camiolomj@upmc.edu

https://doi.org/10.1016/j.xpro.2021.100847

SUMMARY

Bulk expression data from heterogeneous cell populations pose a challenge for investigators, as differences in cell numbers and transcriptional programs may complicate analysis. To improve the performance of bulk RNA sequencing on mixed populations, we created Immune Cell Linkage through Exploratory Matrices (ICLite). The ICLite package for R constructs modules of correlated genes and identifies their relationship to specific lineages in mixed cell populations. This protocol details formatting, optimization of run parameters, and interpretation of results following implementation of ICLite.
For complete details on the use and execution of this protocol, please refer to Camiolo et al. (2021).

BEFORE YOU BEGIN

While the advent of single cell RNA sequencing has allowed for unprecedented insight into the molecular underpinnings of disease, its cost and availability may limit its implementation, particularly on large scale studies of human cohorts. Bulk sequencing is an affordable and readily available alternative that poses its own unique challenges. Analysis of bulk transcriptional data from heterogenous cell populations can be particularly difficult. Changes to relative composition as well as cell state across samples create multiple drivers of variance that can suppress biological discovery and complicate techniques such as simple differential expression.

The ICLite package for R allows users to gain insight into the function of specific cells from a mixed population by breaking bulk transcriptional data into smaller sets of gene modules. These gene modules allow for inference of biological activity, akin to results from a scRNA sequencing experiment. Unlike algorithms designed to estimate relative cell abundance from bulk transcriptional data using pre-defined gene expression sets, ICLite uses correlation between cell abundance and gene module expression to assemble de novo gene modules and link them to cells. These gene modules can be used for a number of downstream applications, such as biological process enrichment and determination of cell activity in health and disease.

The modules identified by ICLite are derived by comparing gene clustering solutions from an array of user-selected input parameters to determine an optimal solution. ICLite accounts for cluster error, cell-to-module connectivity and representation of the starting transcriptional dataset to identify the combination of input parameters that provides a best-fit solution. Ultimately, however, it remains up to the user to decide what solution best suits their needs. For instance, tuning of input parameters may lead to thorough description of some cell types while providing little to no information on others. In the protocol below, we detail how to format data for use with ICLite and walk users through an initial run. We provide example code for determining the most effective solution from a set of input parameters and guide the user through the process of tuning for the objective best fit as well as homing on particular populations of interest.

To properly execute the functions of ICLite, you will need log normalized RNA sequencing or microarray data as well as matched cell count log ratios or absolute values. Though the execution detailed in Camiolo et al. (2021) employed high dimensional data from mass cytometry analysis, the ICLite algorithm is suitable for use with clinically obtained differential cell count data or other measures of cell prevalence. Given the heterogeneity of immune cells, lower resolution data must be interpreted with caution as discussed below (Limitations). In principle, the algorithm behind ICLite should function on non-immune cell populations as well, as no pre-populated gene lists are required for a successful run. The output of ICLite includes CSV files of gene module membership, a graphical correlation plot describing the module-to-cell connectivity, and several objects in the R global environment that can be used for further analysis. We describe downstream analysis below, including study of module relatedness based on functional similarity.

To properly run an analysis using ICLite, you will need the R statistical software environment installed on your computer (Core Team, 2015).

Prepare your transcriptional data

Timing: 5 h

The algorithm driving ICLite's deconvolution process has been tested on both RNA sequencing and microarray data. In the author's experience, RNA sequencing data tends to have less sample-to-sample variation than microarray profiling, making gene module assembly a more robust process. Below is a general pipeline for preparation of RNA sequencing data for use with ICLite.

CRITICAL: ICLite requires log2 normalized data for execution. The below steps detail how to build properly formatted RNA sequencing data. Microarray data must also be log normalized.

Resources for consistent RNA sequencing results can be found through the ENCODE Data Coordinating Center Uniform Processing Pipelines (Davis et al., 2018) at:

https://www.encodeproject.org/pages/pipelines/#RNA-seq

1. Compile a count library from your cells of interest.
   a. Generate and sequence cDNA libraries from isolated immune cells using the Illumina NextSeq500 System or an equivalent platform
   b. Perform sequence alignment using STAR (Dobin et al., 2013) or an equivalent algorithm
   c. Quantify read counts to uniquely mapped using featureCounts (Liao et al., 2014)
2. Prepare log2 RNA sequencing data using the DESeq2 package in R (Love et al., 2014). Benefits of this approach to normalization are discussed below (troubleshooting). For more complete information on using the DESeq2 suite, please visit:

https://www.bioconductor.org/packages/release/bioc/vignettes/DESeq2/inst/doc/DESeq2.html

```
> library(DESeq2)
>
> filePath <- ~/R_code/New_RNAseq/" ## replace this with your directory
> gtf <- read.csv(paste(filePath,"gtf_abridged.csv", sep = ""))[,-1]
> counts <- read.csv(paste(filePath,"counts", sep = ""))
> metaData <- read.csv(paste(filePath,"metadata", sep = ""))
> metaData$sex <- factor(metaData$sex)
>
> ## The below code features no design input and is not meant for differential
> ## testing.
>
> dds <- DESeqDataSetFromMatrix(countData = counts, colData = metaData,
>         design = ~sex, tidy = TRUE)
>
> ## Variance stabilizing transform
> vsdNobatch <- vst(dds, blind = FALSE)
```

Optional: The below code features adjustment for batch outside DESeq2. This process can also be performed inside DESeq2 by amending the 'design' input to the 'DESeqDataSetFrom-Matrix' function.

3. Additional adjustment to address potential batch effect using algorithms such as COMBAT is encouraged (Leek et al., 2012).

```
> library(sva)
>
> metaData$batch <- factor(metaData$batch)
>
> ## Build a covariate matrix. This can be amended to include other covariates. In this
> ## example we will only be adjusting for batch
>
> modcombat = model.matrix(~1, data = data.frame(metaData))
>
## This step generates normalized and transformed data
> gene_expression_data <- ComBat(assay(vsdNobatch), metadata$batch,
>        mod = modcombat, prior.plots = T)
```

⚠ CRITICAL: Multiple entries for the same gene should be avoided. There are no limitations for the gene nomenclature library used. The sample data set provided by the ICLite package uses universal gene symbols as identifiers. Conversion for gene ontology analysis is covered below.

Obtain paired cell count data

Timing: 3 h

Every sample from the transcriptional data set must have paired cell count data for execution of ICLite. Note that log ratio transformation, as detailed below, is required for compositional data sets such as those created by the popular automated cell clustering package cytofkit for R (Chen et al., 2016). Below is a standardized pipeline for automated cell clustering with cytofkit using the FlowSOM algorithm.

4. Load your FCS files following mass cytometry event acquisition and normalization as described by the cytokit authors: https://github.com/JinmiaoChenLab/cytofkit
5. Perform automated clustering and calculate the relative percentage of each cell type across the samples/individuals in your experiment:

```
> cell_clusters <- cytof_cluster(xdata = marker_intensity_data, method = "FlowSOM’’)

> clustered_unstim_cells<-data.frame(marker_intensity_data, cluster = cell_clusters)

> cell_percentages <- cytof_clusterStat(data = clustered_unstim_cells, cluster = "cluster",

>         statMethod = "percentage")
```

CRITICAL: Cluster percentage values must be log ratio transformed as detailed below prior to use with ICLite.

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
Bulk RNA sequencing from BAL immune cells of IMSA cohort participants	GEO	Accession number: GSE136587
FCS files from paired mass cytometry of BAL immune cells of IMSA cohort participants	FlowRepository	Repository ID: FR-FCM-Z395
Software and algorithms		
ICLite	Camiolo et al. (2021)	https://github.com/camiolomj/ICLite/
Blockcluster (4.4.3)	Bhatia et al. (2017)	https://www.jstatsoft.org/article/view/v076i09
Corrplot (0.84)	Taiyun Wei and Viliam Simko (2017)	https://github.com/taiyun/corrplot
Tidyr (1.1.3)	Wickham et al. (2019)	https://tidyr.tidyverse.org/
R Statistical Software	The R Project for Statistical Computing	https://www.r-project.org/
Other		
Local computer – memory: 8GB required, 16GB recommended; processors: 1 required, 4 recommended	N/A	N/A

MATERIALS AND EQUIPMENT

- Data (Bulk RNA-sequencing log2 normalized gene values and paired cell abundance data– see before you begin)

STEP-BY-STEP METHOD DETAILS

⚠ CRITICAL: All code appearing in this protocol, including troubleshooting vignettes, is available as R script titled 'STAR ICLite Run Code' in Data S1.

Install ICLite and its dependencies

Timing: 5 min

The ICLite package for R is available through the following public repository:

https://github.com/camiolomj/ICLite/

1. Download the most recent version of the R package blockcluster (Bhatia et al., 2017) from one of the following sources and install it into your R library:
   a. https://cran.r-project.org/src/contrib/Archive/blockcluster/
   b. https://www.jstatsoft.org/article/view/v076i09

You may also use the following command lines inside R for installation:

```
> library(devtools)

> install_github("cran/blockcluster")
```

2. Download ICLite to your R library using the code:

```
> library(devtools)

> install_github("camiolomj/ICLite")
```

Load ICLite and example data

Timing: 5 min

3. Load the ICLite package:

```
> library(ICLite)
```

Optional: The ICLite package includes reference the input data from the Immune Mechanisms of Severe Asthma (IMSA) cohort (Camiolo et al., 2021), which can be used as reference for formatting issues or trial runs of the algorithm.

4. Load the example IMSA data set:

```
> load_IMSA_data()
```

5. Verify the presence of 'gene_expression_data' and 'immune_cell_logratios' in the Global Environment. These objects will be used for example code going forward.

Transcriptional data formatting

6. Format your gene expression data so that columns represent samples/individuals and rows represent genes.

Cell count data transformation and formatting

Timing: 10 min

Optional: This step is not necessary if using the included immune cell log ratio data.

Cell count data where each of the values for a given sample or individual represent a fraction of the whole must be log ratio transformed to address the constraints of compositional data (van den Boogaart and Tolosana-Delgado, 2008). Failure to perform this step may lead to unreliable results as discussed below (troubleshooting).

7. Inspect cell count values prior to running ICLite to determine whether your data is compositional. While relative percentages from cell count data must be log ratio transformed as detailed below, absolute cell counts do not.

⚠ CRITICAL: Many R packages commonly used for automated cell cluster analysis of mass cytometry data such as cytofkit, detailed above, provide cluster percentage values for downstream analysis. These data must be log ratio transformed as detailed below prior to use with ICLite.

8. If your cell count data is compositional:
   a. Install and load the R package "compositions":

```
> install.packages("compositions")

> library(compositions)
```

   b. Transform your cell count data (obtained above in before you begin):

```
> sample_compositions <-acomp(cell_ percentages, total = 100)

> immune_cell_logratios<-cdt(comp_immune)
```

9. The input cell count matrix for ICLite should be formatted such that columns represent cell lineages and rows represent samples/individuals. The "immune_cell_logratios" object loaded by load_IMSA_data()provides an example of properly formatted data.

Setup initial run_ICLite() parameters

Timing: 5 min

Once you have prepared your input data and installed ICLite, you are ready to begin the deconvolution process. We recommend that initial runs with ICLite employ a broad range of run parameters.

10. Create a vector for rho exclusion values. These values should range from 0.3 to 0.9. Gene correlations below this value are converted to 0 in binary space while those above are converted to 1. Higher cutoffs will result in smaller gene modules.

```
> input_rho<-c(0.4, 0.5, 0.6)
```

11. Create a vector of minimum connectivity values. Genes that do not meet a threshold of interactions above the rho exclusion will be removed from the analysis. Higher cutoffs will result in smaller gene modules that are more tightly connected.

```
> input_connectivities<-c(150,300)
```

12. Create a list of assumed number of clusters to be used for blockclustering.

```
> number_of_clusters<-as.list(c(12,24,36))
```

Optional: Example formatting for run parameters is available in Data S1. This .ZIP file contains an R script titled 'STAR ICLite Run Code' related to the the toy example provided in this manuscript as fully executable code.

⚠ CRITICAL: Users should include a vector of at least 2 input parameters for minimum connectivity, rho cutoff and number of clusters.

Execute run_ICLite()

Timing: 5 h

13. To initiate a run of ICLite, use the code below. For the example code detailed in this manuscript, see Data S1.

```
>set.seed(95)
>
>run_ICLite(gene_expression_data = gene_expression_data,
>     immune_cell_logratios = immune_cell_logratios,
>     input_connectivities = input_connectivities,
>     input_rho = input_rho,
>     number_of_clusters = number_of_clusters)
```

Optional: The blockcluster algorithm is sensitive to seed. To ensure consistency, users may choose to set the global R seed value prior to running ICLite.

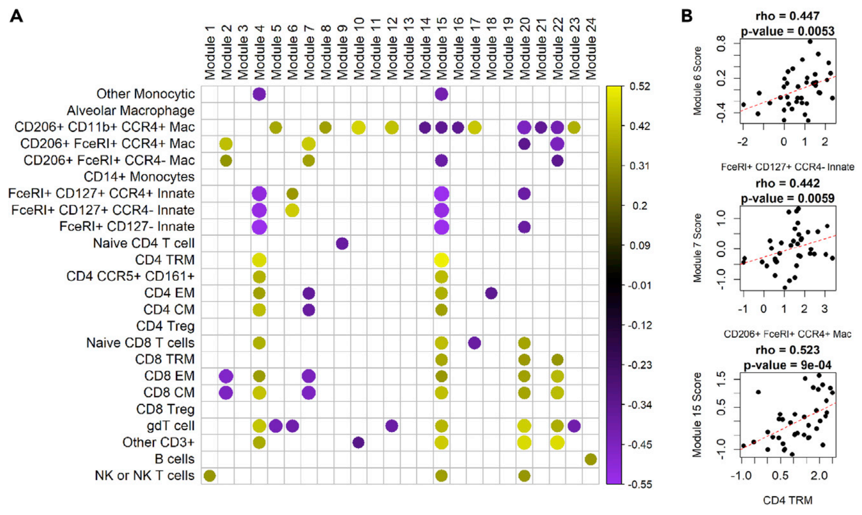


Figure 1. Initial output of ICLite using broad run parameters

(A) Spearman correlation plot generated by execution of ICLite where calculated rho between BAL gene modules (x-axis) and cell lineages (y-axis) is demarcated by circles colored according to the scale detailed to the right. Only associations with FDR corrected p-value of < 0.05 are illustrated. The size of the circle is inversely proportional to p-value of interaction.

(B) Plotting of module scoring vs cell log ratio values with super-imposed linear trend line, Spearman's rho and p-value as indicated in the figure.

Pause point: Saving your workspace after an ICLite run can be done using the command:

```
>save.image("Your_Experiment_Name_Here.Rdata")
```

The session can be reloaded by using the following command:

```
>load("Your_Experiment_Name_Here.Rdata")
```

Inspect ICLite run results

14. Following completion of an ICLite run, the user will be left with output files in the R working directory that include:
    a. A correlation plot describing all relationships between gene modules and cells in the analysis titled "mod solution corrplot.png"
    b. Individual module vs cell lineage dot plots featuring a linear regression trend line, spearman's rho calculation and p-value
    c. Comma separated value (CSV) files for gene module membership

The data shown in Figure 1 illustrates the graphical output from solution generated by ICLite on the IMSA data set included in the package with the parameters described above. Several objects will be created in the Global Environment, including **gene_module_lists,** which contains the gene names for the modules assembled by ICLite.

Should your run produce no connectivity results, see topics below for more information (troubleshooting).

Tune ICLite run parameters

After inspection of ICLite results, users may choose a narrower band of input parameters focused around the optimal solution identified during the initial run.

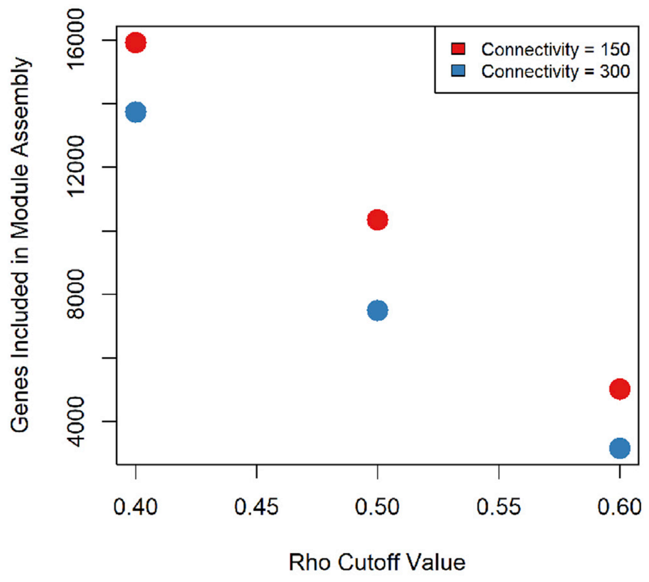


Figure 2. Graphical output from plot_solution_size()
Execution of the plot_solution_size() function will use objects in the R global environment to generate a plot detailing the number of genes included for module assembly by ICLite (y-value) vs rho cutoff value (x-axis). Coloration of dots indicates solution connectivity value.

15. To explore the relationship between input parameters and genes included in module assembly, use the following code:

```
>plot_solution_size()
```

The resultant graph plotting the relationship between gene matrix size, rho cutoff and minimum connectivity value will be saved in the working directory and is demonstrated in Figure 2.

16. To examine run parameters from the optimal solution, use the following code to create a graph of fit scoring. Note that dot size is related to solution fit, with the largest dot equating to the best fit:

```
>plot_fit_score()
```

The graph will again be saved to the working directory. Results from the above example are illustrated in Figure 3. The specific values for the identified optimal run conditions will also be displayed in the R console after executing this command. Note that subsequent ICLite run parameters can be chosen based on these data.

⚠ CRITICAL: We can see from both text and graphical outputs that the optimal solution occurred with a rho cutoff = 0.4, connectivity = 150 and number of clusters = 24.

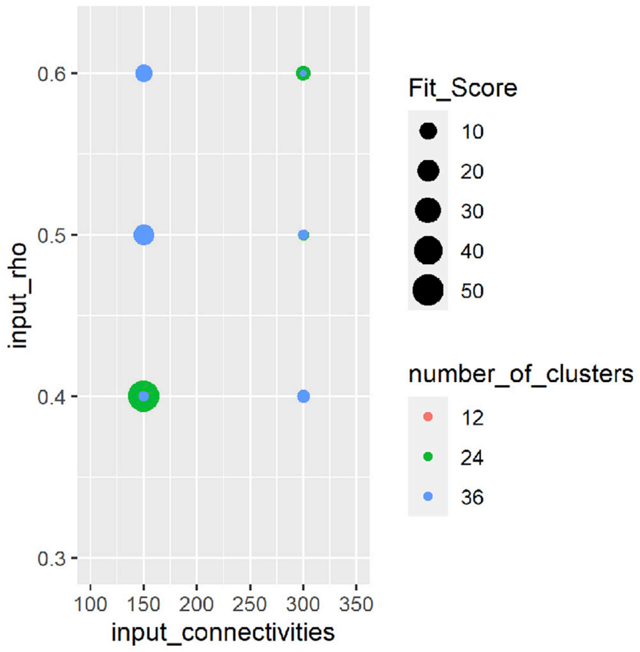


Figure 3. Graphical output from plot_fit_score()
Execution of the plot_fit_score () function will use objects in the R global environment to generate a plot detailing the ICLite solution score by input rho, connectivity and number of clusters. Size of dots in the plot correspond to relative value of fit scoring. Coloration of dots indicates solution connectivity value.

17. Create input parameters with smaller gradients of change focusing around the optimal parameters from the initial run:

```
> input_connectivities<-c(75,150,225)

> input_rho<-c(0.4, 0.425, 0.45)

> number_of_clusters<-as.list(c(22,24,26))
```

18. Re-run ICLite:

```
>set.seed(95)

>

>run_ICLite(gene_expression_data = gene_expression_data,

>      immune_cell_logratios = immune_cell_logratios,

>      input_connectivities = input_connectivities,

>      input_rho = input_rho,

>      number_of_clusters = number_of_clusters)
```

The data shown in Figure 4 illustrates the graphical output from ICLite based on the parameters described above. This time around, we can see that a solution with rho = 0.45, connectivity = 75 and number of clusters = 26 greatly outperformed our initial run.

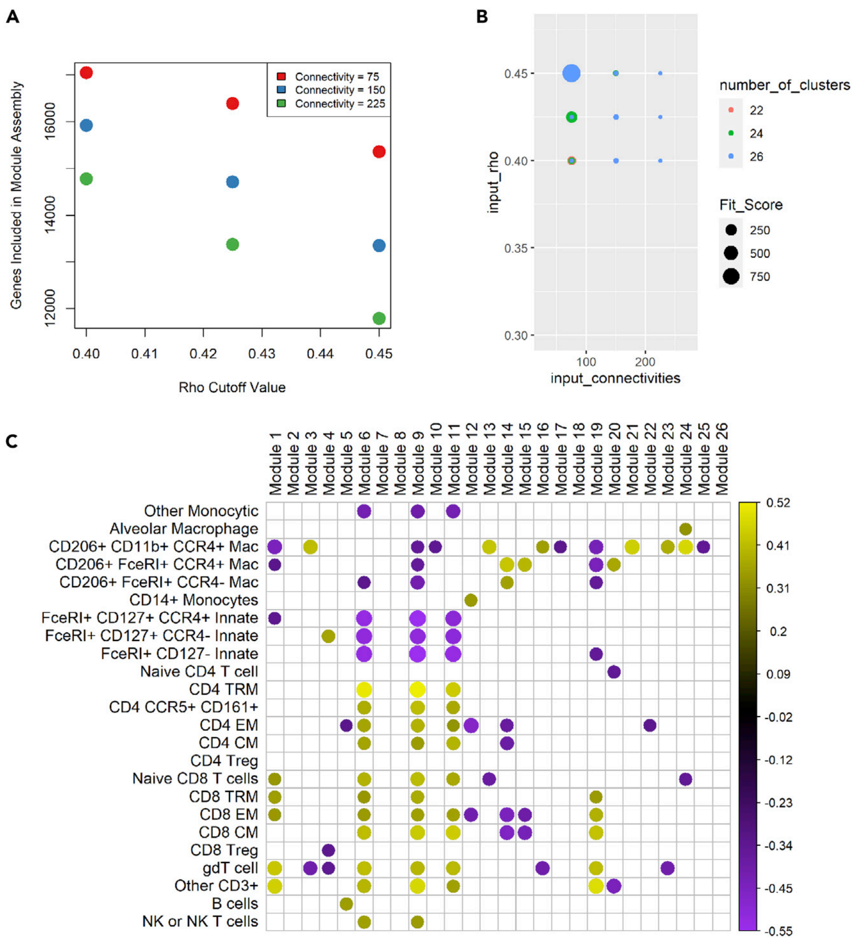


Figure 4. Output of ICLite using focused parameters

(A) Plotting generated by execution of plot_solution_size() from focused run parameters following the initial ICLite run.

(B) Plotting generated by execution of plot_fit_score () from focused run parameters.

(C) Spearman correlation plot from the optimal solution using tuned run parameters in ICLite. Calculated rho between modules (x-axis) and cell lineages (y-axis) is demarcated by circles colored according to the scale detailed to the right. Only associations with FDR corrected p-value of < 0.05 are illustrated. The size of the circle is inversely proportional to p-value of interaction.

Optional: At this time, the user may choose to move onto functional analysis or continue tuning ICLite parameters around the new optimal run.

⚠ CRITICAL: As stated above (before you begin), tuning of input parameters may influence the ability of ICLite to capture relationships to specific cells in your data set. For more information regarding parameter tuning to address these issues, please see troubleshooting below.

Perform gene ontology analysis

The "gene_module_lists" may be used for subsequent Gene Ontology (GO) enrichment analysis as a means of understanding the biological functions represented by ICLite modules. In the analysis presented in Camiolo et al., we used the TopGO package in R (Alexa and Rahnenfuhrer, 2020). Example code for running TopGO is detailed below. More information on TopGO can be found at:

https://www.bioconductor.org/packages/devel/bioc/html/topGO.html

⚠ CRITICAL: Output module membership lists will use whatever input nomenclature was used, which may require conversion prior to running enrichment analysis with popular tools such as fgsea (Korotkevich et al., 2021) or topGO.

19. Prior to GO enrichment using topGO, convert the gene symbols used in the example IMSA data into Entrez ids. To do so, first install and load the conversion database (Carlson, 2019):

```
> BiocManager::install("org.Hs.eg.db")

> library('org.Hs.eg.db')
```

20. Next, convert our genes of interest from the R object containing our gene modules:

```
> GOI<- gene_module_lists[[n]] ## where n = the module index from ICLite

> GOI_converted<-unique(mapIds(org.Hs.eg.db, GOI, 'ENTREZID', 'SYMBOL'))
```

21. Generate a background index of all genes from our transcriptional dataset:

```
> all_genes<-rownames(gene_expression_data)

> background_names<-unique(mapIds(org.Hs.eg.db, all_genes, 'ENTREZID', 'SYMBOL'))
```

22. Install and load topGO:

```
> BiocManager::install("topGO")

> library(topGO)
```

23. Run TopGO for the selected ICLite module:

```
> geneList <- factor(as.integer(background_names %in% GOI_converted))

> names(geneList) <- background_names

> GOdata <- new("topGOdata", allGenes = geneList, nodeSize = 10, ontology = "BP",

>         annot = annFUN.org, mapping = "org.Hs.eg.db")

> resultFisher <- runTest(GOdata, algorithm = "classic", statistic = "fisher")

> upRes <- GenTable(GOdata, classicFisher = resultFisher, ranksOf = "classicFisher",

>         topNodes = 50, numChar = 40)
```

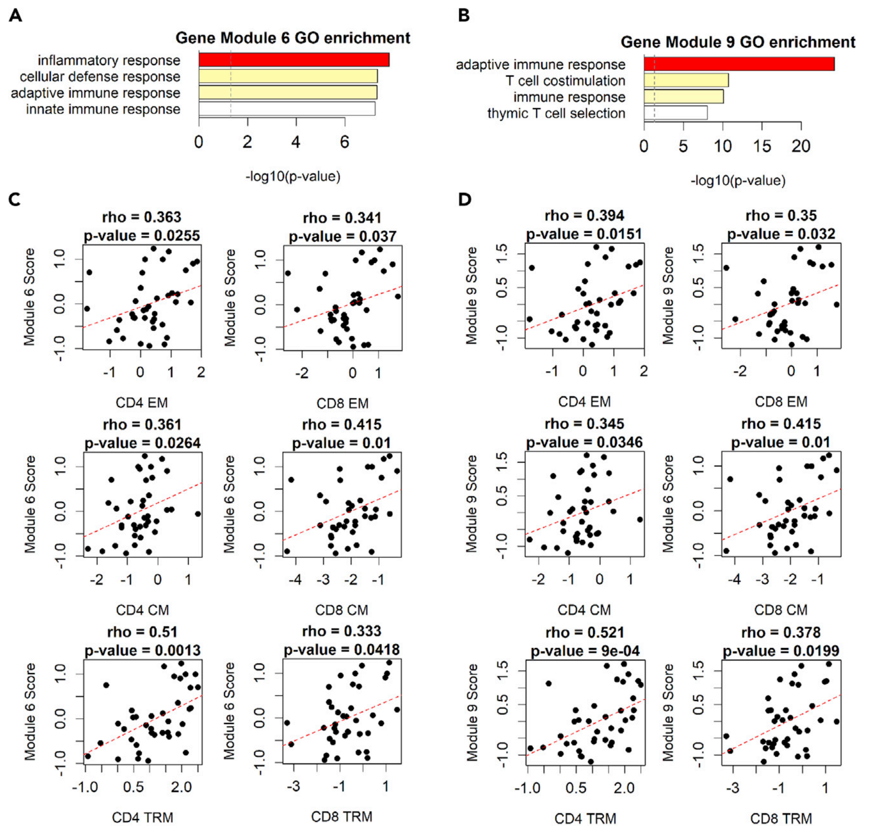


Figure 5. Pathway analysis of modules linked to T cell populations

(A) Barplot of -log10(p-values) for GO term enrichment of ICLite modules 6 which correlates with CD4 and CD8 EMs, CMs and TRMs.

(B) Barplot of -log10(p-values) for GO term enrichment of ICLite module 9, which correlates with CD4 and CD8 EMs, CMs and TRMs.

(C) Plotting of module 6 scoring vs cell log ratio values with super-imposed linear trend line, Spearman's rho and p-value as indicated in the figure.

(D) Plotting of module 9 scoring vs cell log ratio values with super-imposed linear trend line, Spearman's rho and p-value as indicated in the figure.

Optional: The above example uses pre-defined values for node size, enrichment algorithm and test statistic. Please refer to the topGO website above for more information regarding parameter selection.

Example GO enrichment results and corresponding cell log ratio vs module scoring plots from our optimized ICLite solution are illustrated in Figure 5.

Perform semantic similarity clustering

Gene module membership from ICLite solutions is exclusive, meaning that a single gene may be a member of only one module. GO semantic similarity allows us to take sets of gene ontology results and compare them for overlap. This can be used to generate distance measures for dendrogram construction to better understand functional relationships between gene modules.

The GOSemSim package is not provided with ICLite and must be downloaded and installed separately (Yu et al., 2010). Information on this package can be found at:

https://bioconductor.org/packages/release/bioc/html/GOSemSim.html

24. Install and load the GOSemSim package:

```
> BiocManager::install("GOSemSim")

> library(GOSemSim)
```

25. After identifying a satisfactory ICLite solution, obtain GO results for all modules as detailed above and create a list object containing all GO enrichment terms:

```
> all_GO_terms<-list(module_1_terms, module_2_terms....)
```

26. Create a GO library object:

```
> hsGO <- godata('org.Hs.eg.db', ont="BP")
```

27. Create a matrix for semantic similarity results:

```
> ##Where num_clust = the number of modules from the accepted ICLite solution

> GO_semantic_mat<-matrix(0, ncol = num_clust, nrow = num_clust)
```

28. Perform semantic similarity for pairwise comparison between GO term lists corresponding to ICLite modules:

```
> for(c in 1:ncol(GO_semantic_mat)){

>

>  gs1<- all_GO_terms [[c]]

>

>  for(r in 1:nrow(GO_semantic_mat)){

>

>    gs2<- all_GO_terms [[r]]

>    GO_semantic_mat[r,c]<-mgoSim(gs1, gs2, semData=hsGO, measure="Wang",

>          combine="BMA")

>

>  }

> }
```

29. Create a phylogram based on the sematic similarity results to illustrate similarities in the biological process represented by each ICLite module (Paradis and Schliep, 2019):

```
> install_github("cran/ape")
> library("ape")
>
> GO_hclust = hclust(dist(GO_semantic_mat), method = "ward.D2")
> GO_phylo<-as.phylo(GO_hclust)
>
> png(‘‘GO semantic phylogram,.png", height = 2400, width = 2400, res = 300)
> plot(GO_phylo, type = "unrooted", cex = 1.2,
>  no.margin = F, label.offset = 0.03)
> dev.off()
```

While not necessary, semantic similarity clustering can provide reassurance that the modules identified by ICLite are biologically plausible. As demonstrated in Figure 6, ICLite has linked modules of functionally related genes to similar cells. In principle, we expect that modules linked to the same cell lineages should themselves share ontological similarity.

Use modules from an ICLite solution for sample scoring

Once an ICLite solution has been validated, the modules created may be used for scoring individual samples. These scores may be treated as continuous variables.

⚠ CRITICAL: Module scoring may be done on external transcriptional data sets, as described in Camiolo et al., (2021). To do so, gene names must be consistent. Please see the section above on converting nomenclature.

30. Create module score values for the example cohort. Using this code, we call the function mod_score() from the ICLite package on the gene_module_lists object from a successful ICLite run.

```
> individual_mod_scores<-do.call(cbind, lapply(gene_module_lists, mod_score))
> colnames(individual_mod_scores)<-paste0("Mod_",
>     unique(accepted_solution@rowclass)+1)
```

EXPECTED OUTCOMES

After completion of an ICLite run, the resultant module to gene connectivity solution, correlation plots and output CSV files will be stored in the R working directory. Output .CSV files can be opened in R or excel for downstream analysis using tools such as gene ontology enrichment. Though the examples above used additional R coding for ontology analysis, public internet databases are also suitable. Inspection of optimal run parameters can be made using the plot_fit_score() function.

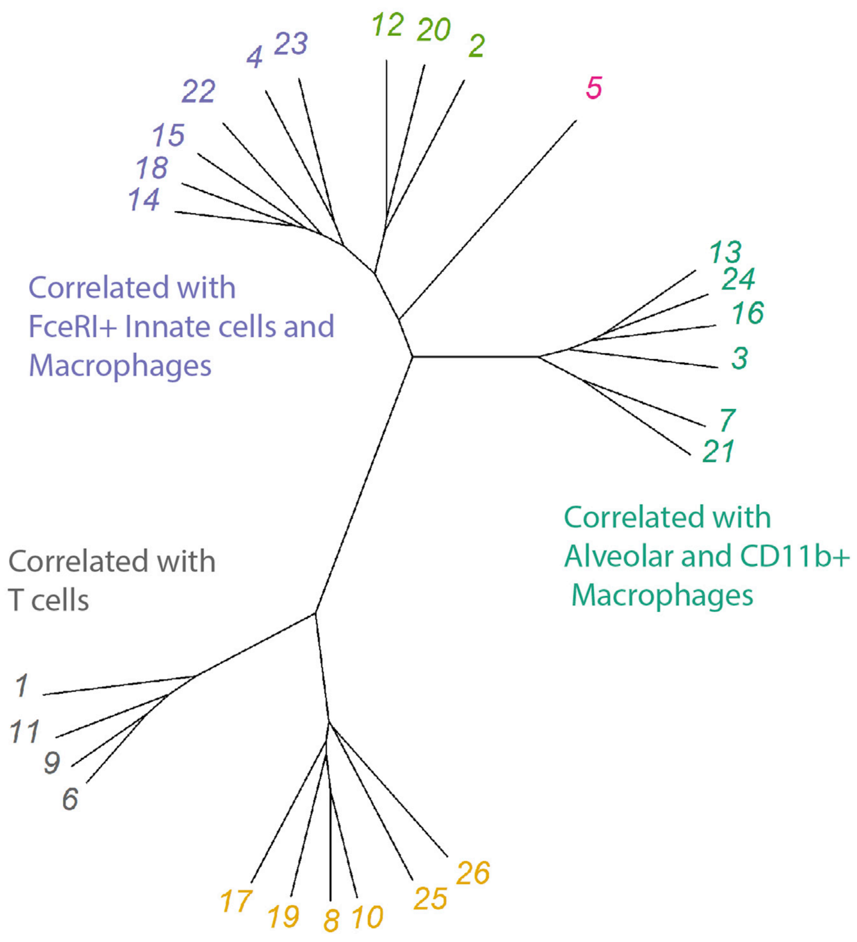

4
23
12
20
2
5
22
15
18
14
Correlated with FceRI+ Innate cells and Macrophages
13
24
16
3
7
21
Correlated with Alveolar and CD11b+ Macrophages
Correlated with T cells
1
11
9
6
17
19
8
10
25
26

Figure 6. Using semantic similarity to explore functional similarity of ICLite modules
Phylogram of GO term semantic similarity for ICLite modules described in the solution from Figure 4. Distance is independent of cell associations and based only on functional enrichment in GO terms from recovered ICLite modules. Module coloring is based off hierarchical clustering of semantic similarity. Terms adjacent to phylogram summarize cell lineages attached to GO semantic families.

LIMITATIONS

Given a broad enough array of starting parameters, ICLite should be successful at deriving an output set of modules. It is, however, possible for the blockclustering process to fail at reaching a suitable result. Scenarios for this will be covered below (troubleshooting).

ICLite was initially built for use on high-dimensional cell count data derived from mass cytometry experiments. The analysis of immune cells detailed in Camiolo et al. (2021) featured 33 surface markers with more than 20 distinct cell lineages identified from the bronchoalveolar lavage of cohort participants. Using more simplified immune cell data, such as clinically obtained differential cell counts, may lead to loss of resolution during the solution selection phase of ICLite. Because gene modules are derived independently, it may still be possible to capture heterogeneous processes within a "parent" cell type. We highly recommend careful gene ontology analysis following these runs to ensure the biological plausibility of your results. Data generated by ICLite is meant to enable further hypothesis testing and guide downstream validation.

TROUBLESHOOTING

Below we will cover some common issues that may give rise to errors when running ICLite.

⚠ CRITICAL: Code for troubleshooting vignettes is available as R script titled 'STAR ICLite Run Code' in Data S1.

Problem 1

ICLite Fails to Connect ANY Cells to Modules.

Potential solution

In the example provided by load_IMSA_data(), 39 individuals are used for pooled analysis. The power to discern relationships will be directly related to the number of samples in your dataset. Though a strong relationship may be identified with fewer samples, we would caution against using data sets with less than 10 individuals. Successful solution finding may take several iterations using varied inputs. In the example provided, we started with broad parameters and focused more finely afterward. An alternate approach would be running two or more sets of focused parameters up front. Details on how parameters may impact relationship resolution are detailed below. In addition, properly normalized data is critical to ICLite functionality.

ICLite employs Spearman's rank correlation, which does not assume linear relationship between cell count and gene expression values. Log scaling of expression data does allow for more practical visualization of correlations by graphical output, however. Furthermore, accounting for RNA composition is recommended for accurate comparison of expression between samples, making normalization a critical step towards ensuring reliable results (Anders and Huber, 2010). In the example provided above, we have employed DESeq's normalization algorithm prior to analysis with ICLite.

It is important to note that other commonly used normalization methods, such as reads per kilobase of exon per million reads mapped (RPKM) or transcripts per kilobase million (TPM), are less suitable for use with ICLite. Both of these methods describe the relative abundance of a transcript among the population of sequence transcripts, and therefore depend on the composition of the RNA population in the sample (Zhao et al., 2020). As RNA repertoires may differ across samples, particularly with mixed cell population data, use of DESeq2 for normalization is preferable.

Problem 2

ICLite Fails to Describe My Cells of Interest.

Potential solution

Under certain circumstances, describing the activity of specific cells lineages may be of interest to investigators. The below example is illustrated using the IMSA dataset to demonstrate how input parameters may influence the ability to resolve these relationships.

Consider a run with the following parameters:

```
> input_connectivities<- c(100, 200)

> input_rho<- c(0.55, 0.65)

> number_of_clusters<- c(12,24,36)

> run_ICLite(gene_expression_data, immune_cell_logratios, input_connectivities,

>     input_rho, number_of_clusters)
```

The resultant correlation plot is presented in Figure 7. GO analysis demonstrates appropriate functional enrichment in modules linked to T cell populations, however ICLite as failed to link many other cell types to gene modules. Looking back at our run parameters, we see that we have selected high rho exclusion values (0.55 and 0.65), which will remove all but the strongest gene-gene interactions.

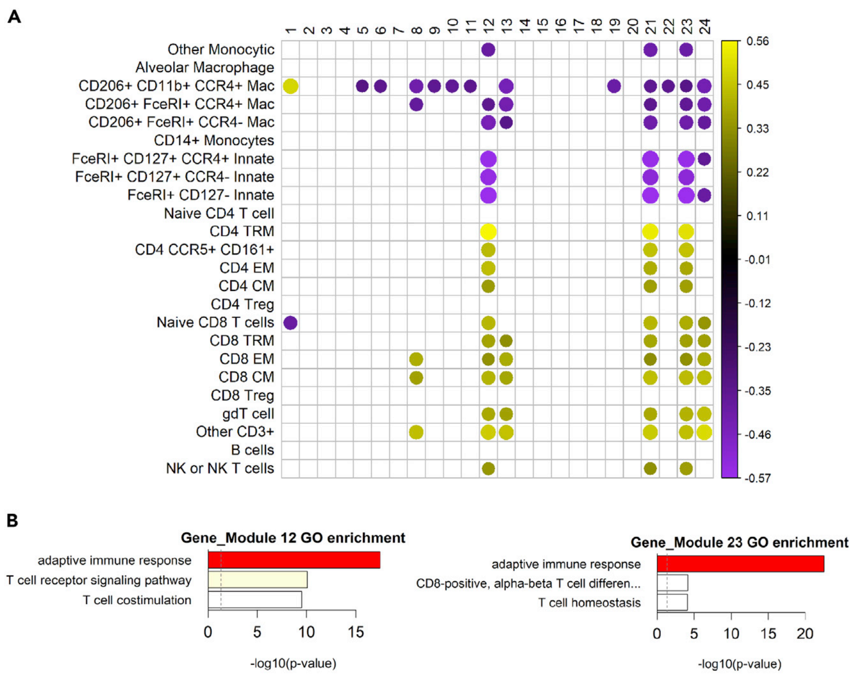


Figure 7. High rho cutoffs result in poor description of macrophage populations
(A) Spearman correlation plot generated by execution of ICLite on the example data set using high rho exclusion values only. Calculated rho between modules (x-axis) and cell lineages (y-axis) is demarcated by circles colored according to the scale detailed to the right. Only associations with FDR corrected p-value of < 0.05 are illustrated. The size of the circle is inversely proportional to p-value of interaction.
(B) Barplot of -log10(p-values) for GO term enrichment of ICLite modules 12 and 23, which correlate with CD4 and CD8 EMs, CMs and TRMs.

As noted above (Tuning ICLite Run Parameters), selection of input parameters will influence the total number of genes included in module generation. We will rerun the analysis, this time also considering solutions with rho exclusion values of 0.45.

⚠ CRITICAL: As number of genes included in module assembly increases, so does overall run time and system memory requirements.

```
> input_connectivities<- c(100, 200)

> input_rho<- c(0.45, 0.55)

> number_of_clusters<- c(20,24,26)

>

> run_ICLite(gene_expression_data, immune_cell_logratios, input_connectivities,

>     input_rho, number_of_clusters)
```

By varying our input parameters, we were able to maintain relationships with T cell populations while capturing additional information about cell lineages that were previously poorly described (Figure 8). For a similar number of assumed clusters and minimum connectivity, we are able to describe relationships between several macrophage and innate lineages and ICLite gene modules. Performing GO analysis, we see that these new module connections represent interesting biological functions that we may now ascribe to other immune cells from our data set. Table 1 illustrates the

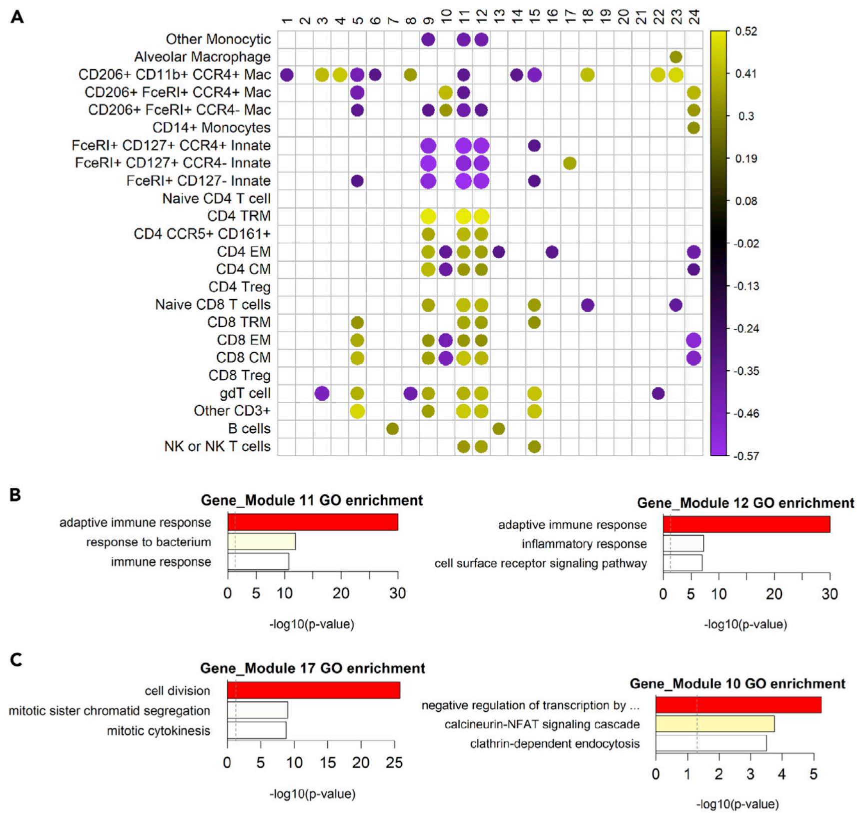


Figure 8. Decreasing input stringency identifies additional relationships

(A) Spearman correlation plot generated by execution of ICLite on the example data set using a broader set of rho exclusion values in parameters used to generate Figure 7. Calculated rho between modules (x-axis) and cell lineages (y-axis) is demarcated by circles colored according to the scale detailed to the right. Only associations with FDR corrected p-value of < 0.05 are illustrated. The size of the circle is inversely proportional to p-value of interaction.

(B) Barplot of -log10(p-values) for GO term enrichment of ICLite modules 11 and 22, which correlate with CD4 and CD8 EMs, CMs and TRMs.

(C) Barplot of -log10(p-values) for GO term enrichment of ICLite modules 17 and 10, which correlate with FceRI+ CD127+ innate cells and FceRI+ macrophages, respectively.

differences in outcome we get by considering a lower rho exclusion value, giving us a favorable solution that describes a greater breadth of cells in our experiment. Note that the relationship between output GO terms and genes included is not linear!

Problem 3

ICLite Connect Cells and Modules with Disparate Ontological Function.

Potential solution

Extensive work has demonstrated that spurious correlation or anti-correlation may occur when using compositional data, thereby impacting the results obtained from ICLite. If you find that ICLite is detecting inappropriate relationships between modules and cells, verify that you have addressed the need for log ratio transformation as detailed above. In brief, if cell count data for every sample adds up to 1 or 100, it is likely compositional and requires transformation (Cell Count Data Transformation and Formatting).

Problem 4

Gene modules returned by ICLite seem functionally redundant and connect to similar cells.

Table 1. Comparison of input and output measures from clustering solutions demonstrated in Figures 7 "Unfavorable Solution" and 8 "Favorable Solution", respectively

Favorable solution	Unfavorable solution
num_clust = 24	num_clust = 24
rho_cutoff = 0.45	rho_cutoff = 0.65
min_connectivity = 200	min_connectivity = 200
Successful Positive Correlations: 56	Successful Positive Correlations: 48
Number of Genes in Solution: 13151	Number of Genes in Solution: 2704
GO Terms Enriched in Modules: 3625	GO Terms Enriched in Modules: 1905
GO Terms Linked to Cells: 994	GO Terms Linked to Cells: 436

Potential solution

The number of assumed gene clusters should be considered in relation to the total size of the transcriptional data set. Though ICLite does penalize for over-clustering, it will only consider solutions from the input vector. Therefore, initial runs may benefit from a broad array of values that may be narrowed on successive iterations. Subsequent runs may focus on finer gradients in parameters around previously successful solutions. Note that the blockclustering algorithm that ICLite employs is sensitive to seed. To ensure consistency between runs, make sure to set a global seed beforehand.

For the sake of illustration, 3 correlation plots have been drawn showing how number of clusters will influence solution generation (Figure 9). As might be expected, a greater number of clusters will lead to a greater number of total positive correlations. Examining the plots, we see that many of the relationships appear to be redundant as number of assumed clusters increases. Specifically, we see that the T cells in our data set share connection to increasing number of modules as total number of assumed clusters increases. Ontologically, these modules are highly related and consist of genes that are clustered together in other solutions (demarcated by red boxes). Though ICLite penalizes for potential over-clustering, evaluation of ontology followed by semantic similarity clustering may help the user avoid scenarios where many small gene clusters are functionally redundant.

Problem 5

ICLite terminates mid-run without successful solution generation.

Potential solution

Combinations of high connectivity and rho cutoff values will eventually generate empty test matrices. For example:

```
> input_connectivities<- c(1000, 2000)

> input_rho<- c(0.85, 0.95)

> number_of_clusters<- c(24,36)
```

The empty matrix will cause an error in blockcluster:

```
>Error in blockcluster::coclusterBinary(test_mat, semisupervised = FALSE, :

+ Number of Row clusters exceeds numbers of rows.
```

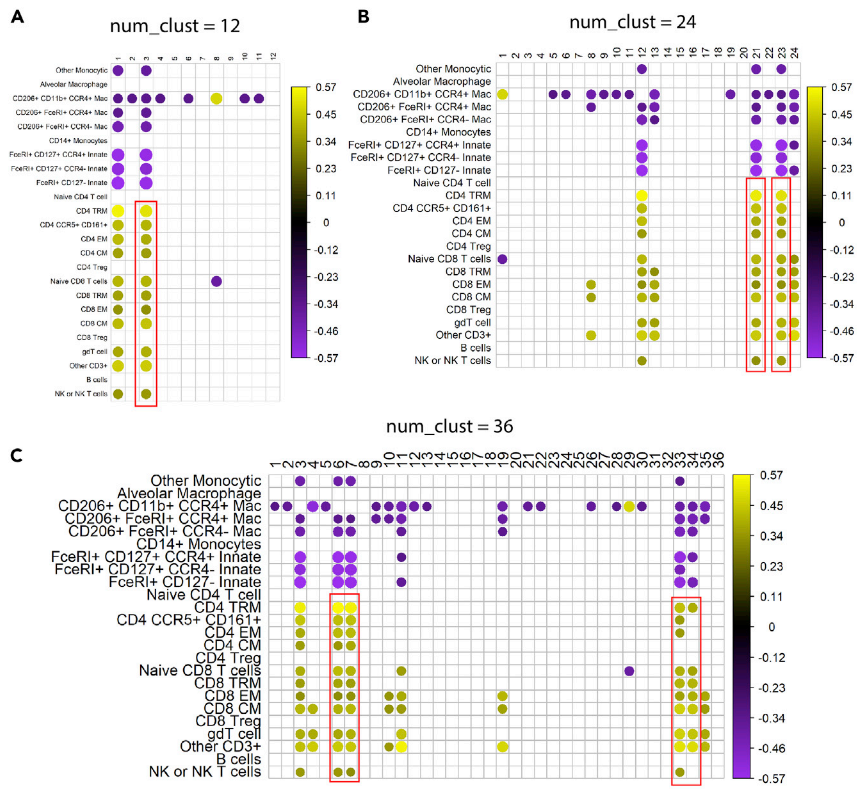


Figure 9. Assessing for relative overclustering in ICLite solutions

Correlation plots from solutions generated by execution of ICLite with similar rho exclusion and connectivity values but varying number of assumed clusters. Calculated rho between modules (x-axis) and cell lineages (y-axis) is demarcated by circles colored according to the scale detailed to the right. The size of the circle is inversely proportional to p-value of interaction.

(A–C) (A) 12 clusters assumed for blockclustering, (B) 24 clusters assumed for blockclustering or (C) 36 clusters assumed for blockclustering. Red boxes indicate modules with similar ontologic function and gene membership across solutions.

We may decrease the rho exclusion cutoffs (below) to resolve this error:

```
> input_connectivities<- c(1000, 2000)

> input_rho<- c(0.85, 0.95)

> number_of_clusters<- c(24,36)
```

This time, the run generates 3 successful clustering solutions but fails on the third matrix:

```
> Co-Clustering Failed!

> Co-Clustering successfully terminated!

> Co-Clustering successfully terminated!

> Error in blockcluster::coclusterBinary(test_mat, semisupervised = FALSE, :

+ Number of Row clusters exceeds numbers of rows.
```

ICLite creates an environmental object named "test_mat_list" that includes the input matrices for all solutions fed into blockcluster for module assembly. Inspec this value in RStudio using this code:

```
> for(m in 1:length(test_mat_list)){
>
>    print(attributes(test_mat_list[[m]])$dim)
>
>}
```

Which results in:

```
+ }
[1] 4081 4081
[1] 1426 1426
[1] 910 910
[1] 0 0
>
```

We see now that the first 3 matrices were generated without issue but the final has no column or row values. This can again be corrected by liberalizing our cutoffs. This time, decreasing minimum connectivity as below yields a successful run:

```
> input_connectivities<- c(750, 1000)
> input_rho<- c(0.45, 0.55)
> number_of_clusters<- c(24,36)
```

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources and reagents should be directed to and will be fulfilled by the lead contact, Matthew Camiolo (camiolomj@upmc.edu).

Materials availability

This study did not generate new unique reagents. Example data referenced in the text can be loaded into R using the command ICLite::load_IMSA_data(). Unprocessed FCS files and gene expression data can be accessioned as detailed above.

Data and code availability

The GEO accession number for the RNA-seq data is GSE136587. FCS files of mass cytometry data are available through FlowRepository. The ICLite package and documentation are available for download at https://github.com/camiolomj/ICLite/. Immune cell count and BAL transcriptional data from IMSA participants are included as part of the R package and are preformatted to run in ICLite. Code detailed in this Protocol, including Troubleshooting, is available as an R script through supplemental information.

SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.xpro.2021.100847.

ACKNOWLEDGMENTS

This work was supported by NIH grants P01AI106684 (to A.R. and S.E.W.), R01HL113956 (to A.R. and P.R.) and R01AI048927 (to A.R.), U10HL109152 (S.E.W.), and 1F32HL14741501 (to M.J.C).

AUTHOR CONTRIBUTIONS

Conceptualization, M.J.C.; methodology, M.J.C.; data analysis, M.J.C.; writing, M.J.C.; supervision, A.R. and S.E.W.; funding acquisition, A.R., S.E.W., and M.J.C.

DECLARATION OF INTERESTS

A.R. has a research agreement with Pieris Pharmaceuticals. S.E.W. is a consultant for AstraZeneca, Glaxo Smith-Kline, and Sanofi. She is also involved in clinical trials being run by Knopp, Sanofi, and AstraZeneca. She has a research agreement with Pieris Pharmaceuticals. M.J.C. is a consultant for Pieris Pharmaceuticals.

REFERENCES

Alexa, A., and Rahnenfuhrer, J. (2020). topGO: Enrichment Analysis for Gene Ontology.

Anders, S., and Huber, W. (2010). Differential expression analysis for sequence count data. Genome Biol. *11*, R106.

Bhatia, P.S., Iovleff, S., and Govaert, G. (2017). Blockcluster: an R Package for Model-Based Co-Clustering. Journal of Statistical Software, *76*. https://doi.org/10.18637/jss.v076.i09.

Camiolo, M.J., Zhou, X., Oriss, T.B., Yan, Q., Gorry, M., Horne, W., Trudeau, J.B., Scholl, K., Chen, W., Kolls, J.K., et al. (2021). High-dimensional profiling clusters asthma severity by lymphoid and non-lymphoid status. Cell Rep. *35*, 108974.

Carlson, M. (2019). org.Hs.eg.db: Genome Wide Annotation for Human.

Chen, H., Lau, M.C., Wong, M.T., Newell, E.W., Poidinger, M., and Chen, J. (2016). Cytofkit: a Bioconductor package for an integrated mass cytometry data analysis pipeline. PLoS Comput. Biol. *12*, e1005112.

Core Team, R. (2015). R: A Language and Environment for Statistical Computing (R Foundation for Statistical Computing).

Davis, C.A., Hitz, B.C., Sloan, C.A., Chan, E.T., Davidson, J.M., Gabdank, I., Hilton, J.A., Jain, K., Baymuradov, U.K., Narayanan, A.K., et al. (2018). The Encyclopedia of DNA elements (ENCODE): data portal update. Nucleic Acids Res. *46*, D794–d801.

Dobin, A., Davis, C.A., Schlesinger, F., Drenkow, J., Zaleski, C., Jha, S., Batut, P., Chaisson, M., and Gingeras, T.R. (2013). STAR: ultrafast universal RNA-seq aligner. Bioinformatics *29*, 15–21.

Korotkevich, G., Sukhov, V., Budin, N., Shpak, B., Artyomov, M.N., and Sergushichev, A. (2021). Fast gene set enrichment analysis. bioRxiv, 060012. https://doi.org/10.1101/060012.

Leek, J.T., Johnson, W.E., Parker, H.S., Jaffe, A.E., and Storey, J.D. (2012). The sva package for removing batch effects and other unwanted variation in high-throughput experiments. Bioinformatics *28*, 882–883.

Liao, Y., Smyth, G.K., and Shi, W. (2014). featureCounts: an efficient general purpose program for assigning sequence reads to genomic features. Bioinformatics *30*, 923–930.

Love, M.I., Huber, W., and Anders, S. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biol. *15*, 550.

Paradis, E., and Schliep, K. (2019). Ape 5.0: an environment for modern phylogenetics and evolutionary analyses in R. Bioinformatics *35*, 526–528.

van Den Boogaart, K.G., and Tolosana-Delgado, R. (2008). "compositions": A unified R package to analyze compositional data. Comput. Geosci. *34*, 320–338.

Wickham, Hadley & Averick; Mara & Bryan; Jennifer & Chang; Winston & McGowan; Lucy & François; Romain & Grolemund; Garrett & Hayes; Alex & Henry; Lionel & Hester; Jim & Kuhn; Max & Pedersen; Thomas & Miller; Evan & Bache; Stephan & Müller; Kirill & Ooms; Jeroen & Robinson; David & Seidel; Dana & Spinu; Vitalie & Yutani; Hiroaki (2019). Welcome to the Tidyverse. Journal of Open Source Software *4*. https://doi.org/10.21105/joss.01686.

Yu, G., Li, F., Qin, Y., Bo, X., Wu, Y., and Wang, S. (2010). GOSemSim: an R package for measuring semantic similarity among GO terms and gene products. Bioinformatics *26*, 976–978.

Zhao, S., Ye, Z., and Stanton, R. (2020). Misuse of RPKM or TPM normalization when comparing across samples and sequencing protocols. RNA *26*, 903–909.