BFX scRNA-seq Workshop Day 2

JENNIFER A. FOLTZ, PHD

INSTRUCTOR, SECTION OF COMPUTATIONAL BIOLOGY

JENNIFER.A.FOLTZ@WUSTL.EDU

With content contributions from Dr. Petti

Characterizing clusters using differential gene expression

- Data has zero-inflated negative binomial distribution (lots of zeros, overdispersed) so can't use bulk methods
  - Default in Seurat: Wilcoxon rank-sum test
  - Nonparametric version of t-test
  - For two clusters (A and B), and one gene, rank each cell in each cluster according to expression
  - Determine whether sum-of-ranks for cluster A is significantly different than sum-of-ranks for cluster B
  - Clear explanation of Wilcoxon rank-sum test: http://statweb.stanford.edu/~susan/courses/s141/hononpara.pdf
  - Numerous other tests in Seurat and other packages

    Fold-changes are lower due to noise and low- detection

    Generally accepted to set a minimum detection rate to decrease noise & power

        Most commonly 25% of cells must express the gene for it be detected but can do lower or higher depending on question

Differential Expression in Practice

Other Differential Expression Tests to Consider:

- AUC/ROC
- DeSeq2
- MAST/Logistic Regression
- Scran pairwiseWilcox() by blocking

Things to Consider:

- Pairwise differential gene expression
- What fraction of cells in each sample express a given gene?
- Of the cells in each sample that express a given gene, does the mean expression in those cells differ?
- Does the distribution of cell types differ between samples?
- Do the samples exhibit cell-type-specific differential gene expression?
- Input into gene ontology algorithms (e.g. do you need a universe/background?)

Review of Plots from Homework (see Rmd file)

Analyzing Multiple Samples

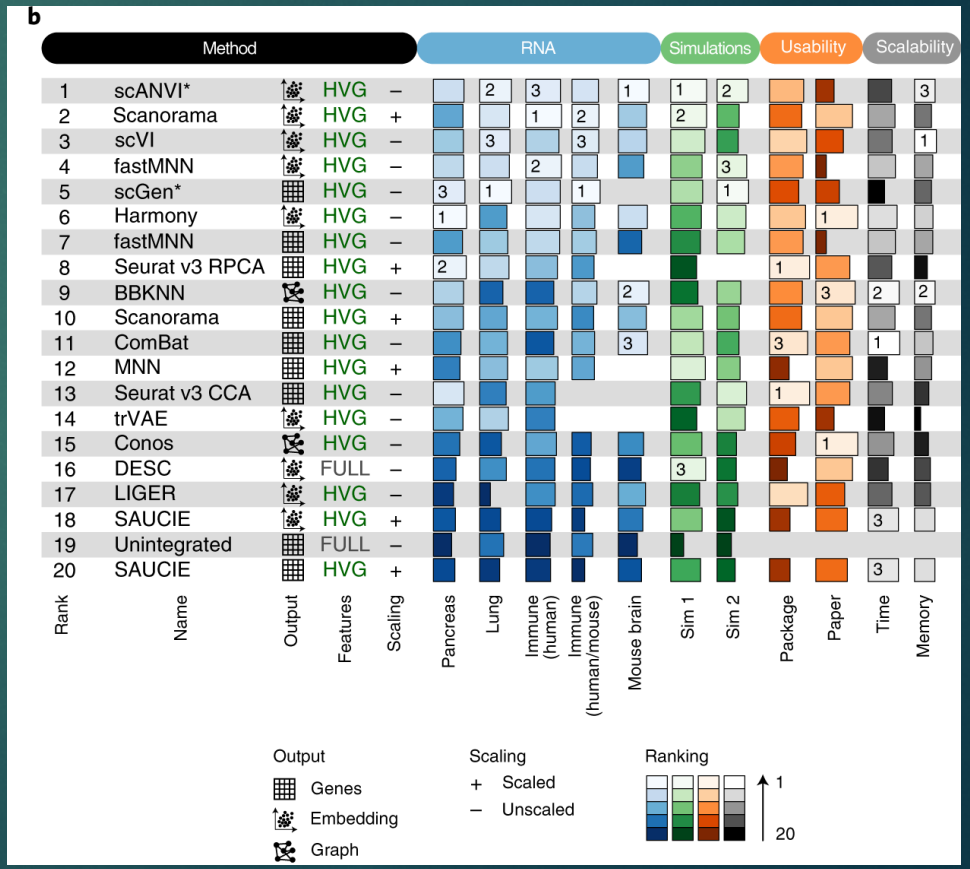
- Merging or batch-correction?
- Avoid batch correction unless absolutely necessary
  - Correct for different technologies (e.g. 3’ and 5’)
  - Correct for different batches
  - Discover conserved biology by finding corresponding cells across different data sets
- Cellranger does faux batch-correction (corrected values are discarded), but batch-corrected tSNE can be visualized in the loupe browser.

Table 1 Description of the 14 batch-effect correction methods

From: A benchmark of batch-effect correction methods for single-cell RNA sequencing data

Tools	Programming language	Batch-effect-corrected output	Methods	Reference package version
Seurat 2 (CCA, MultiCCA)	R	Normalized canonical components (CCs)	Canonical correlation analysis and dynamic time warping	Butler et al. [4], Seurat package version 2.3.4
Seurat 3 (Integration)	R	Normalized gene expression matrix	Canonical correlation analysis and mutual nearest neighbors-anchors	Stuart et al. [12], Seurat package version 3.0.1
Harmony	R	Normalized feature reduction vectors (Harmony)	Iterative clustering in dimensionally reduced space	Korsunsky et al. [13], Harmony version 0.99.9
MNN Correct	R	Normalized gene expression matrix	Mutual nearest neighbor in gene expression space	Haghverdi et al. [5], Scran package version 1.12.0
fastMNN	R	Normalized principal components	Mutual nearest neighbor in dimensionally reduced space	Haghverdi et al. [5], Lun ATL [7], Scran package version 1.12.0
ComBat	R	Normalized gene expression matrix	Adjusts for known batches using an empirical Bayesian framework	Johnson et al. [1]
limma	R	Normalized gene expression matrix	Linear model/empirical Bayes model	Smyth et al. [2], limma version 3.38.3
scGen	Python	Normalized gene expression matrix	Variational auto-encoders neural network model and latent space	Lotfollahi et al. [16], 2019, scGen version 1.0.0
Scanorama	Python/R	Normalized gene expression matrix	Mutual nearest neighbor and panoramic stitching	Hie et al. [9], Scanorama version 1.4.
MND-ResNet	Python	Normalized principal components	Residual neural network for calibration	Shaham et al. [15] updated code to Python 3
ZINB-WaVE	R	Normalized feature reduction vectors (ZINB-WaVE)/normalized gene expression matrix	Zero-inflated negative binomial model, extension of RUV model	Risso et al. [6], ZINB-WaVE version 1.6.0
scMerge	R	Normalized gene expression matrix	Stably expressed genes (scSEGs) and RUVIII model	Lin et al. [18], scMerge version 1.1.3
LIGER	R	Normalized feature reduction vectors (LIGER)	Integrative non-negative matrix factorization (iNMF) and joint clustering + quantile alignment	Welch et al. [14], liger version 1.0
BBKNN	Python/R	Connectivity graph and normalized dimension reduction vectors (UMAP)	Batch balanced *k*-nearest neighbors	Polański et al. [10], bioRxiv. BBKNN version 1.3.2

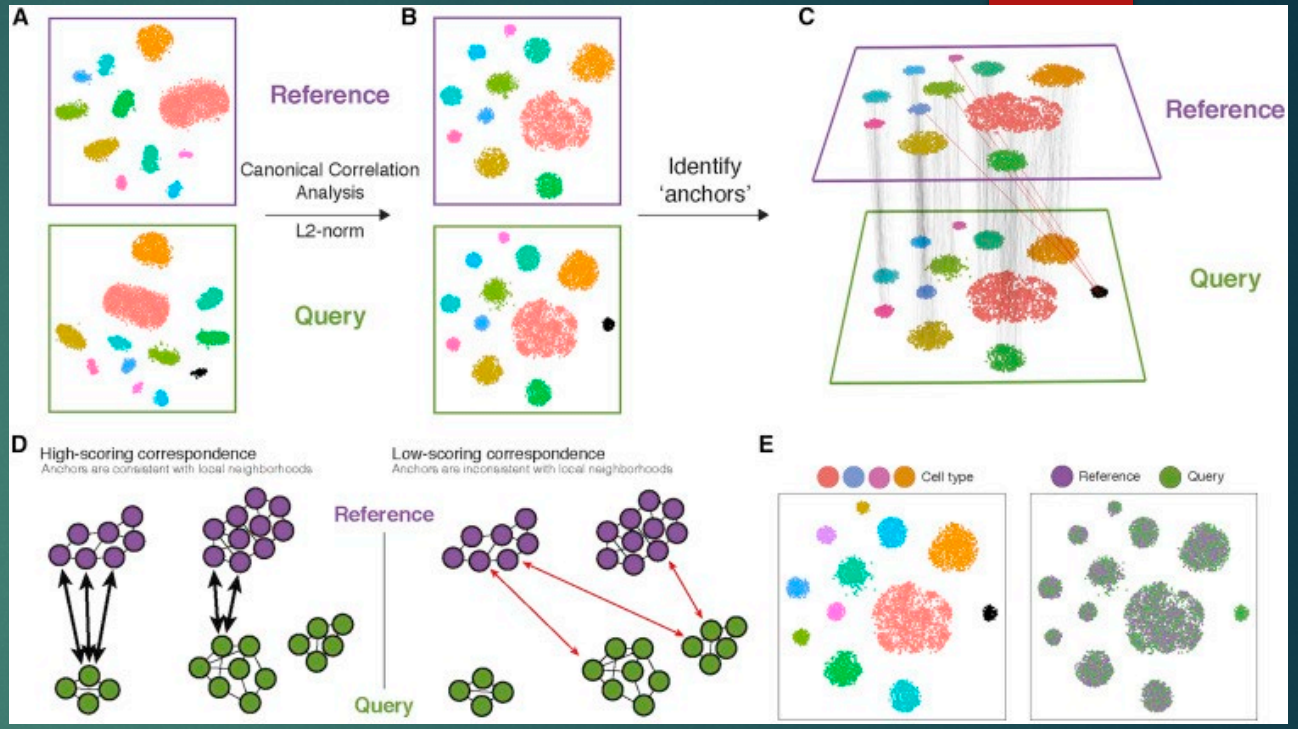
Luecken, et al., 2022



Option 1: Seurat Integration

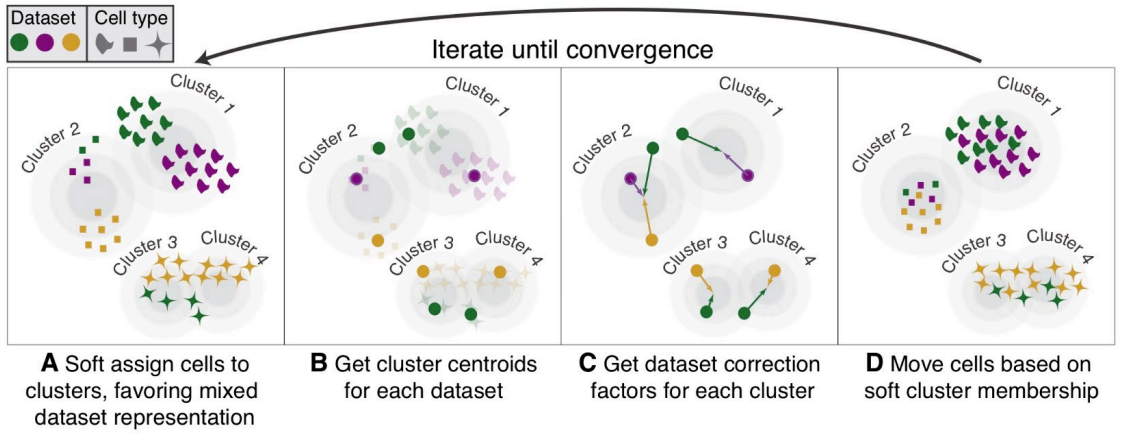
Stuart et al. 2019

```
# matrix.dirs is a list of directories containing 10x data
data.10x = list(); # declare list of 10x data sets
for (i in 1:length(matrix.dirs.)) { # for each data set
        data.10x[[i]] <- Read10X(data.dir = matrix.dirs[i]); # add it to the list
}
scrna.list = list(); # declare list of Seurat objects
for (i in 1:length(data.10x)) { # for each data set…
    scrna.list[[i]] = CreateSeuratObject(counts = data.10x[[i]], min.cells=10, min.features=100,project=batch[i]); # …create a Seurat object for each data set
    scrna.list[[i]][["Batch"]] = batch[i]; # assign a batch label from ‘batch’
    scrna.list[[i]][["Sample"]] = samples[i]; # assign a sample name from ‘samples’
}
# Not shown: Filter each sample separately and normalize (using same method for all samples)
anchors <- FindIntegrationAnchors(object.list = scrna.list, dims = 1:30) # find anchors
scrna.int <- IntegrateData(anchorset = anchors, dims = 1:30) # Integrate data
DefaultAssay(object = scrna.int) <- "integrated” # make integrated data the default for downstream analyses
```

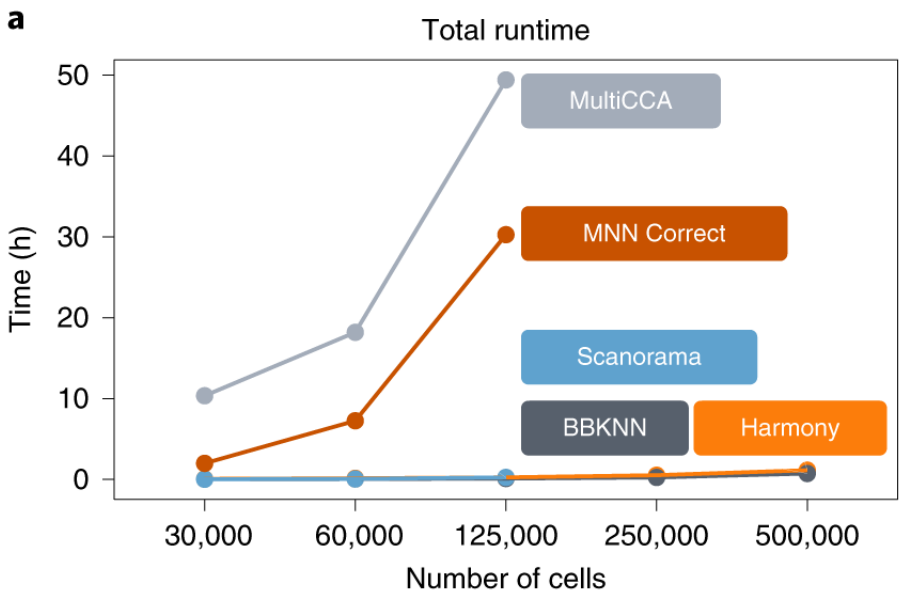


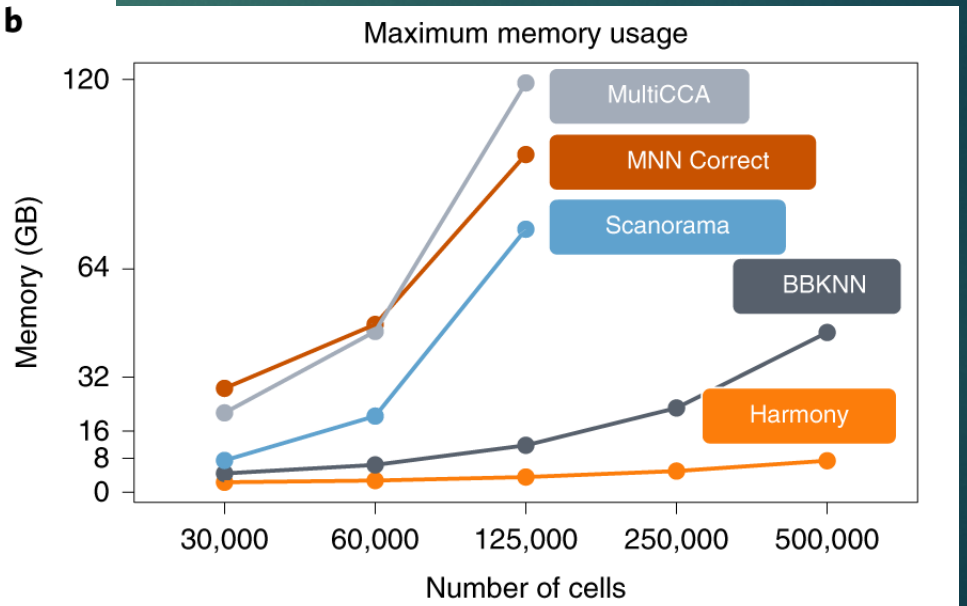
- Integrated values not intended for use with differential expression calculations.
  - We recommend running your differential expression tests on the “unintegrated” data. By default this is stored in the “RNA” Assay. There are several reasons for this.
  - The integration procedure inherently introduces dependencies between data points. This violates the assumptions of the statistical tests used for differential expression.
- TransferData function uses data integration to classify cells based on a reference data set.

Option 2: Harmony Batch Correction (Recommended)



- Ability to batch correct on multiple classes
- Minimal increase in the saved object file size

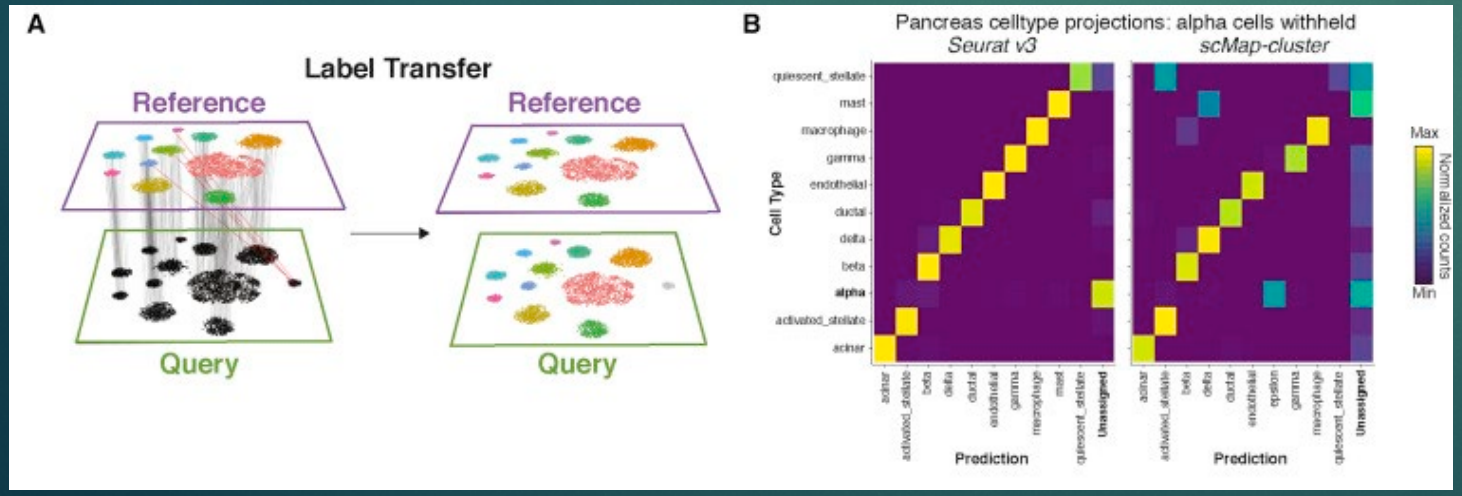




Korsunsky et al., 2019

Label Harmonization/Transfer Approaches

- In lieu of batch correction, can “integrate” data by aligning the sample annotations
  - e.g. look for gene changing between the same cell populations across samples
- Pros: No batch correction required, smaller datasets easier to work with computationally
- Cons: No increase in power by combining samples, annotations may not be 100% consistent

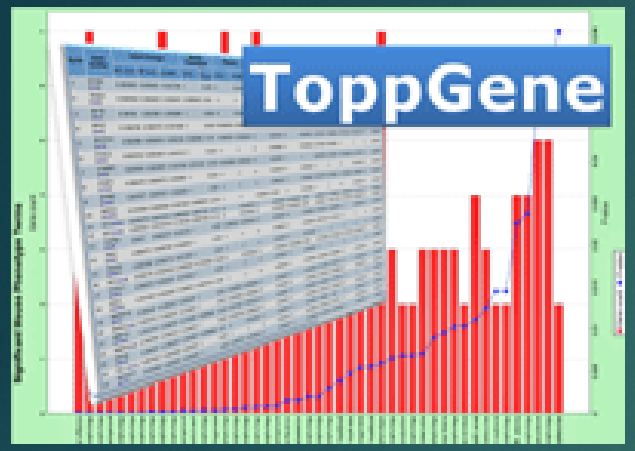


Stuart et al. 2019

Reclustering of Data

- Need to rerun FindVariableFeatures (in order to increase chance of finding additional heterogeneity or cell populations)
- Rerun ScaleData
  - I also rerun CellCycleScoring although likely not necessary
- Rerun dimensionality reduction tests, and batch correction (if using Harmony)

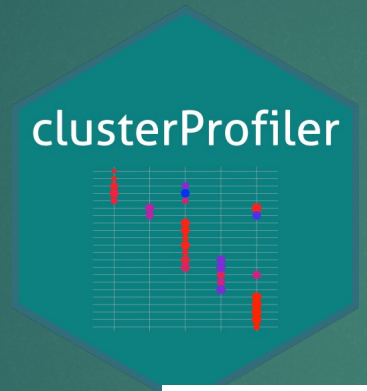
Pathway & Gene Set Analysis

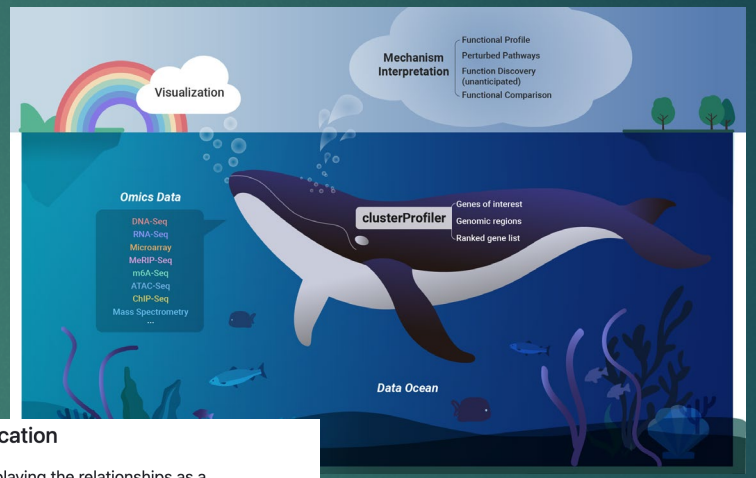


https://toppgene.cchmc.org/enrichment.jsp

ToppFun

Clusterprofiler
https://yulab-smu.top/biomedical-knowledge-mining-book/





15.4 Heatmap-like functional classification

The `heatplot` is similar to `cnetplot`, while displaying the relationships as a heatmap. The gene-concept network may become too complicated if user want to show a large number significant terms. The `heatplot` can simplify the result and more easy to identify expression patterns.

```
p1 <- heatplot(edox, showCategory=5)
p2 <- heatplot(edox, foldChange=geneList, showCategory=5)
cowplot::plot_grid(p1, p2, ncol=1, labels=LETTERS[1:2])
```

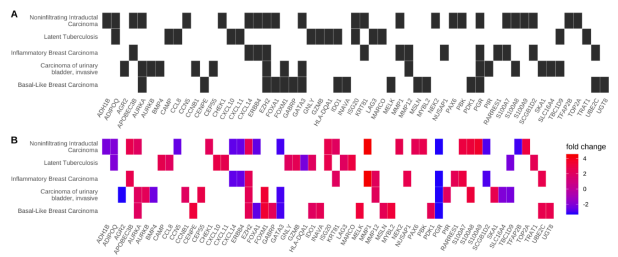
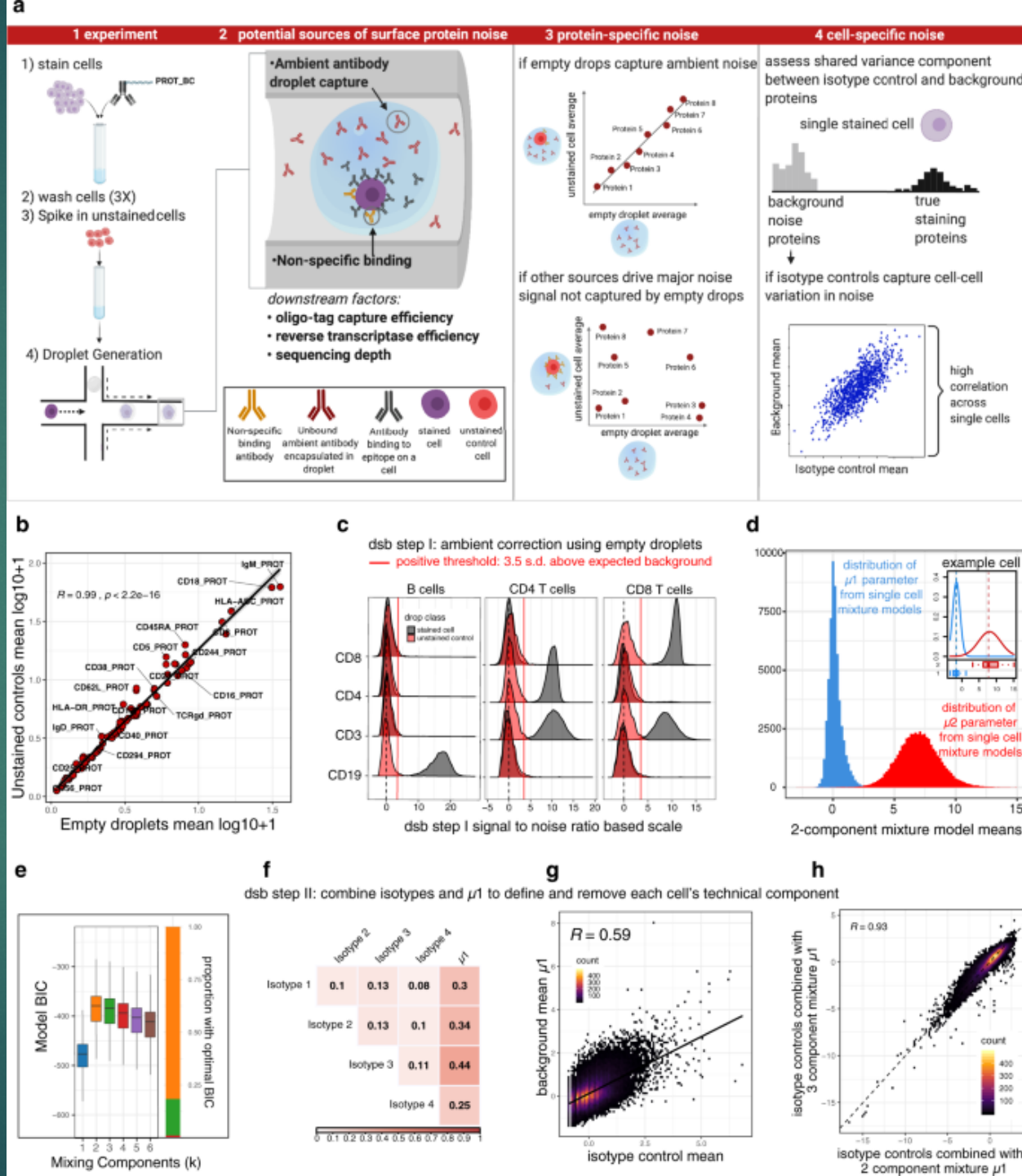


Figure 15.6: **Heatmap plot of enriched terms.** default (A), `foldChange=geneList` (B)

CITE-seq Data

- 3 main types of normalization:
  - 1. CLR (centered log ratio) in Seurat, with margin=1, plots low to high for each features
  - CLR in Seurat, with margin 2, requires the assumption that each cell is stained with roughly the same amount of antibodies- normalizes per cell
  - dsb (separate package)
    - This requires the raw output with empty droplets from 10x to specify a background
    - Recommended to have isotype controls as well

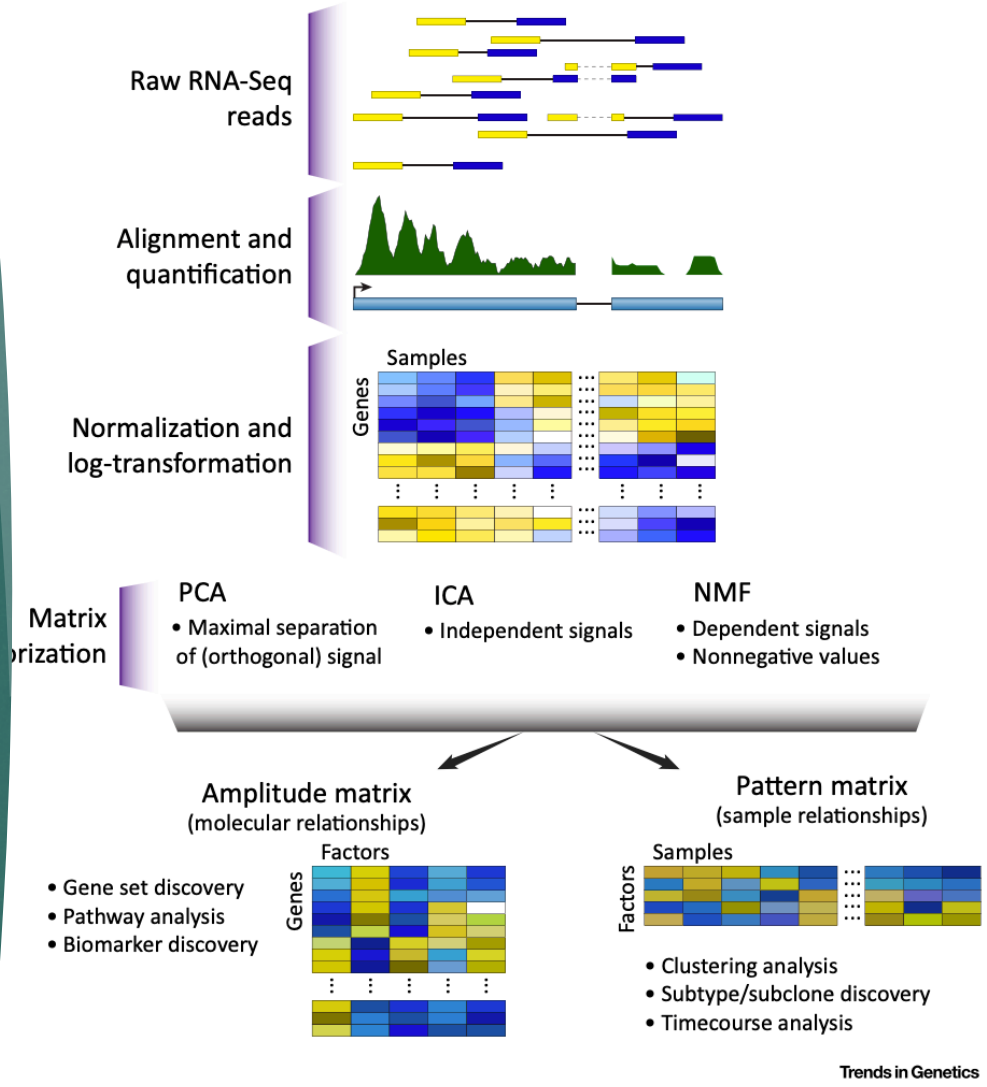


Mule et al., 2019

Extended Applications of scRNA-seq

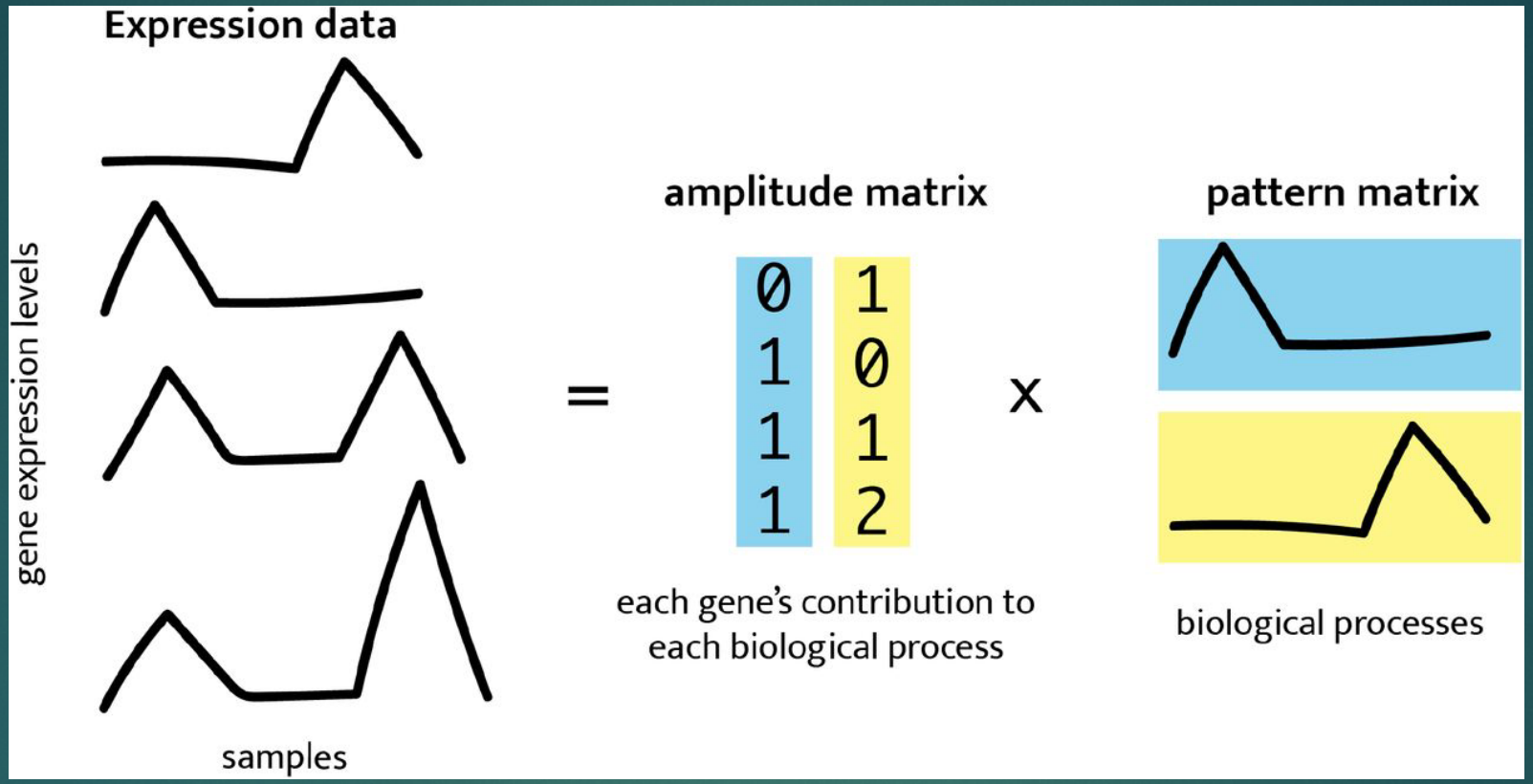
Non-Negative Matrix Factorization

- Complementary technique to PCA
- Identifies patterns in your dataset that are unbiased
  - Looks for genes that change in an organized and non-negative fashion within a set of cells and defines this as a pattern
  - Does not look for patterns different between your groups- this is performed after the fact
- Similar to PC, has no hard black/white truth for the number of patterns that best represents your data- this depends on your experimental question
- Requires a priori to tell the algorithm how many patterns you want to find
  - Best Practice: Run at multiple patterns
- Requires removal of genes with many 0s, Recommend remove other genes as well to decrease run-time, since computationally expensive



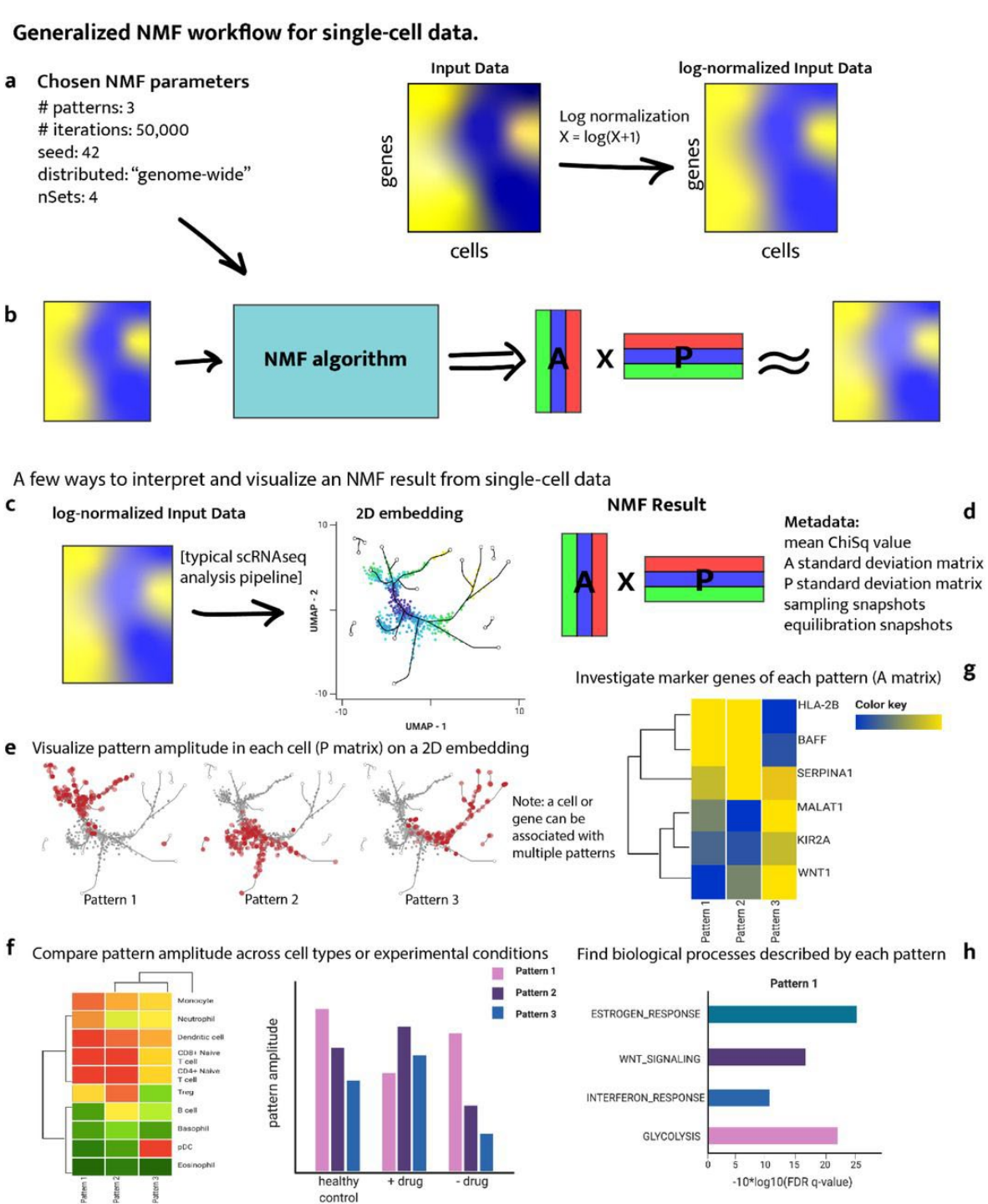
ure 1. Omics Technologies Yield a Data Matrix That Can Be Interpreted through MF. The data matrix from cs has each sample as a column and each observed molecular value (expression counts, methylation Levels, protein ncentrations, etc.) as a row. This data matrix is preprocessed with techniques specific to each measurement chnology, and is then input to a matrix factorization (MF) technique for analysis. MF decomposes the preprocessed ta matrix into two related matrices that represent its sources of variation: an amplitude matrix and a pattern matrix. The ws of the amplitude matrix quantify the sources of variation among the molecular observations, and the columns of the attern matrix quantify the sources of variation among the samples. Abbreviations: ICA, independent component analysis; MF, non-negative matrix factorization; PCA, principal component analysis.

CoGAPS



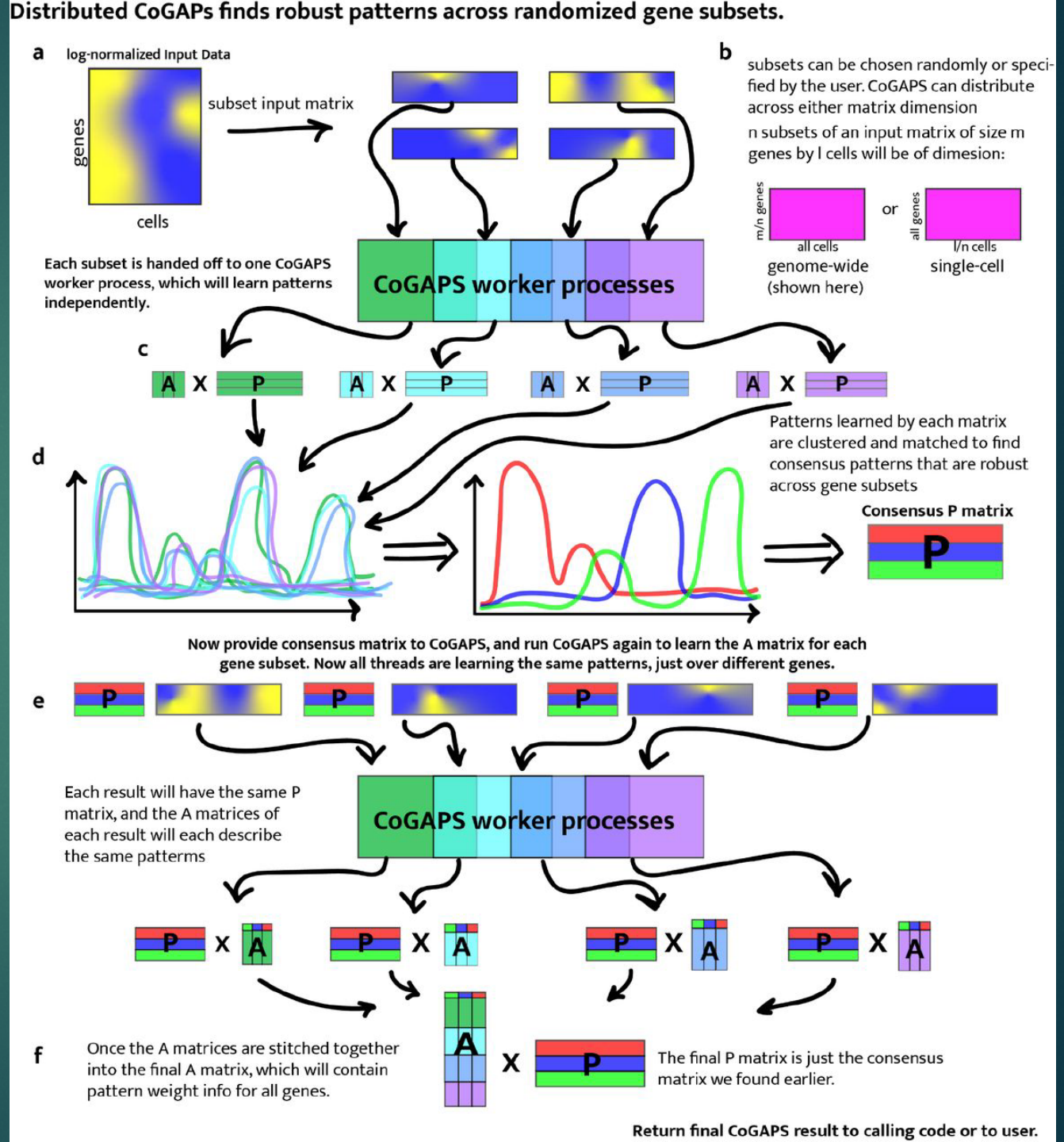
E.g. in T cells, you might see Granzymes, Perforin, and cytokine genes all increase, this could be Pattern 1, that you could examine across samples, treatment conditions, and identify the genes associated with this pattern

doi: https://doi.org/10.1101/2022.07.09.499398

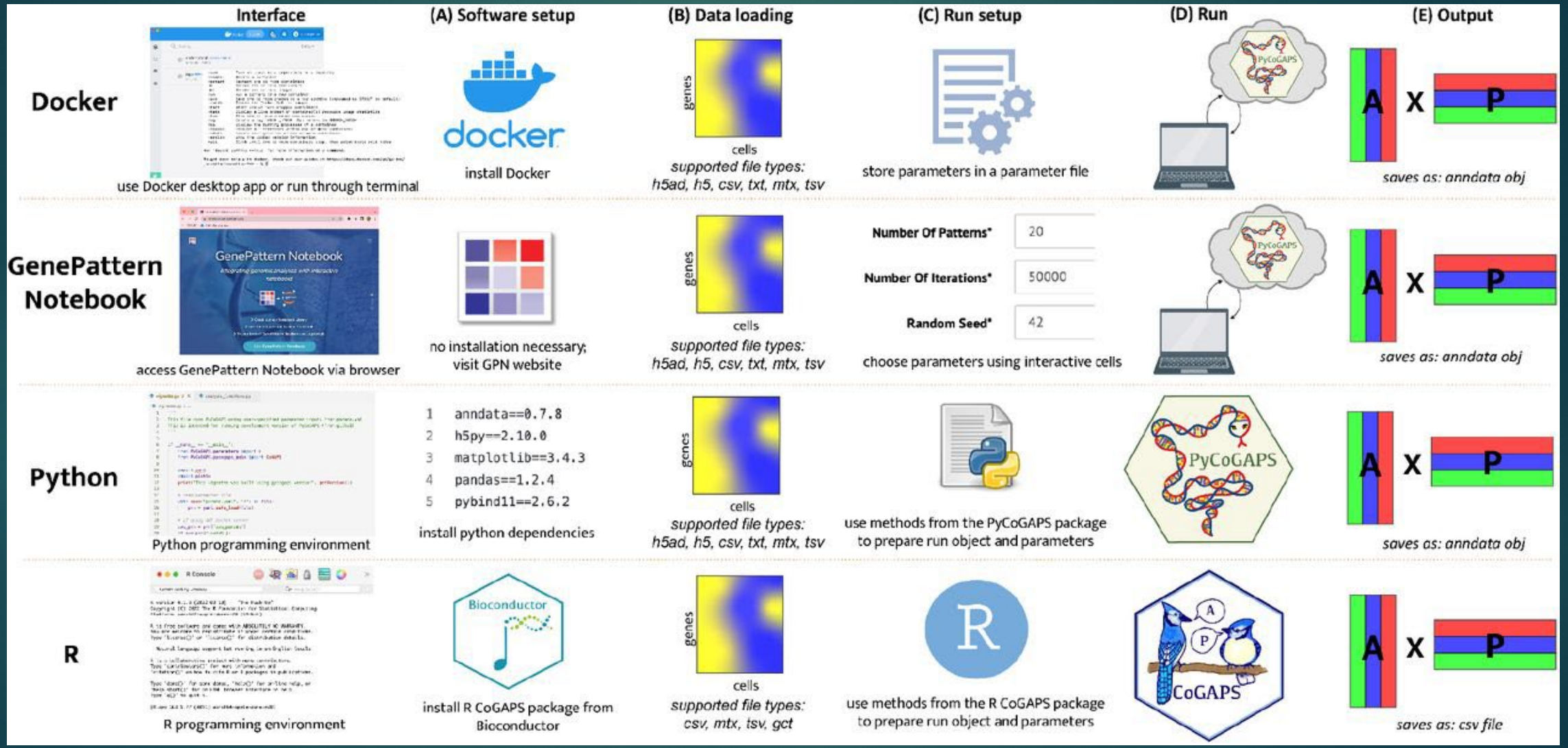

Generalized NMF workflow for single-cell data.
a
Chosen NMF parameters
patterns: 3
iterations: 50,000
seed: 42
distributed: "genome-wide"
nSets: 4
Input Data
log-normalized Input Data
Log normalization
X = log(X+1)
genes
cells
b
NMF algorithm
A
X
P
A few ways to interpret and visualize an NMF result from single-cell data
c
log-normalized Input Data
[typical scRNAseq analysis pipeline]
2D embedding
UMAP - 1
UMAP - 2
NMF Result
d
Metadata:
mean ChiSq value
A standard deviation matrix
P standard deviation matrix
sampling snapshots
equilibration snapshots
e
Visualize pattern amplitude in each cell (P matrix) on a 2D embedding
Pattern 1
Pattern 2
Pattern 3
Note: a cell or gene can be associated with multiple patterns
g
Investigate marker genes of each pattern (A matrix)
Color key
HLA-2B
BAFF
SERPINA1
MALAT1
KIR2A
WNT1
f
Compare pattern amplitude across cell types or experimental conditions
Monocyte
Neutrophil
Dendritic cell
CD8+ Naive T cell
CD4+ Naive T cell
Treg
B cell
Basophil
pDC
Eosinophil
pattern amplitude
healthy control
+ drug
- drug
h
Find biological processes described by each pattern
Pattern 1
ESTROGEN_RESPONSE
WNT_SIGNALING
INTERFERON_RESPONSE
GLYCOLYSIS
-10*log10(FDR q-value)

CoGAPS

Addressing the computational cost of NMF



CoGAPs in Practice



Final Thoughts

- Don’t be afraid to not know
- Everyone started at ground zero sometime
- Think of coding as learning how to use a computer- each software/package has its’ own set of quirks
- Not everything published works in real life

Questions on the Homework?

Seurat object: meta data

Seurat essential commands:
https://satijalab.org/seurat/articles/essential_commands.html

Meta data (scrna@meta.data) contains:
- summary statistics
- sample name
- cluster membership for each cell
- cell cycle phase for each cell
- batch or sample for each cell
- other custom labels for each cell

Access using:
- `scrna[[]]`
- `scrna@meta.data`
- `str(scrna@meta.data)`

Combine with R commands such as head and str, e.g. str(scrna[[]])

Example: Access number of genes (“Features”) for each cell:
- `head(scrna@meta.data$nFeature_RNA)`

Example: Access number of UMIs for each cell:
- `head(scrna@meta.data$nCount_RNA)`

What are the items in the current default cell identity class?

`Table(Idents(object))`

How many clusters are there?
- `length(unique(scrna@meta.data$seurat_clusters))`
- `levels(x=scrna)`

What batches are included in this data set?

`unique(scrna@meta.data$Batch)`

Can add meta data

```
> str(scrna@meta.data)
'data.frame':   13049 obs. of  14 variables:
 $ orig.ident           : Factor w/ 1 level "452198": 1 1 1 1 1 1 1 1 1 1 ...
 $ nCount_RNA           : num  11846 3535 2850 9158 5607 ...
 $ nFeature_RNA         : int  3557 1740 1617 2810 2298 3834 1737 2526 2745 2947 ...
 $ percent.mito         : num  0.0386 0.0526 0.0589 0.0717 0.0462 ...
 $ percent.ribo         : num  0.1033 0.097 0.0821 0.1902 0.0885 ...
 $ S.Score              : num  -0.00337 -0.06351 0.30666 -0.09634 -0.0508 ...
 $ G2M.Score            : num  -0.428 -0.112 0.109 -0.28 -0.233 ...
 $ Phase                : Factor w/ 3 levels "G1","G2M","S": 1 1 3 1 1 1 3 1 1 1 ...
 $ CC.Difference        : num  0.4243 0.0486 0.1976 0.1833 0.182 ...
 $ Sample               : Factor w/ 2 levels "452198_P","452198_R": 1 1 1 1 1 1 1 1 1 1 ...
 $ RNA_snn_res.0.7      : Factor w/ 14 levels "0","1","2","3",..: 3 4 7 3 3 7 3 6 3 3 ...
 $ seurat_clusters      : Factor w/ 14 levels "0","1","2","3",..: 3 4 7 3 3 7 3 6 3 3 ...
 $ ClusterNames_0.7_17PC: Factor w/ 14 levels "0","1","2","3",..: 3 4 7 3 3 7 3 6 3 3 ...
 $ CellType             : Factor w/ 15 levels "B-CELL","BASO",..: 11 4 11 13 13 4 11 11 11 11 ...
```

Seurat object: Assay data and Dimensionality reduction

- Assay data (i.e. RNA-seq measurements): e.g. ‘RNA’
  - Assay = data type
  - Each assay has multiple “slots” that hold the raw data (‘counts’), scaled data (‘scale.data’) or normalized data (‘data)
  - Data transformation, such as scaling and normalization, adds new ‘slots’ to the assay data
  - Access values in the slots as follows:
    - raw RNA counts: `scrna[['RNA']]@counts[1:3,1:3]`
    - scaled data after SCTransform: `scrna[['SCT']]@scale.data[1:3,1:3]`
    - corrected UMI count data after SCTransform: `scrna[['SCT']]@counts[1:3,1:3]`
    - log-normalized data after SCTransform: scrna[['SCT']]@data[1:3,1:3]
  - Alternatively, use the function GetAssayData:
    - `GetAssayData(object = scrna, slot = 'scale.data')[1:3, 1:3]`
    - `GetAssayData(object = scrna, slot = 'counts')[1:3, 1:3]`
  - It’s often important to know what ‘slot’ a function is using. Sometimes you can change it.
- Dimensionality reduction data, e.g. PCA, tSNE, UMAP data
  - Access using scrna[[‘pca’]], scrna[[‘tsne’]], scrna[[‘umap’]]

Common functions for the Seurat object

```
GetAssayData(object = scrna, slot = "counts")
GetAssayData(object = scrna, slot = "scale.data")
FetchData(object = scrna) # returns a data frame
colnames(x = scrna)
rownames(x = scrna)
VariableFeatures(object = scrna)
scrna[["assay.name"]] eg scrna[[“RNA”]]
scrna[["pca"]]
Embeddings(object = scrna, reduction = "pca")
Loadings(object = scrna, reduction = "pca")
Idents(object = scrna) # get default cell identities
Idents(object = scrna) <- "new.idents” # change the identities
Idents(object = scrna, cells = 1:10) <- "new.idents” # change the identities for specific cells
scrna$saved.idents <- Idents(object = scrna) # change current identities for an existing identity class
levels(x = scrna) # get a list of the default identities (e.g. a list of clusters)
RenameIdents(object = scrna, "old.ident" = "new.ident") # change name of identity classes
WhichCells(object = scrna, idents = "ident.keep") # which cells are in cluster x?
WhichCells(object = scrna, idents = "ident.remove", invert = TRUE)
WhichCells(object = scrna, downsample = 500)
WhichCells(object = scrna, expression = name > low & name < high)
subset(x = scrna, cells = cellist, idents=‘keep’); # subset seurat object and return seurat object
subset(x = scrna, subset = name > low & name < high)
merge(x = object1, y = object2)
```

Additional ways to access data

```
Cells(scrna) # get list of cells
# choose cells that have a given characteristic, here, that are in “sample1”:
subcells <- Cells(scrna)[(which(scrna[["Sample"]]$Sample == sample1))]
# alternative approach, choosing cells in cluster 4:
Idents(object=scrna) <- "ClusterNames” # set default identity to ClusterNames
WhichCells(scrna,idents="4") # use WhichCells
# extract raw expression matrix for all cells in cluster 4
as.matrix(GetAssayData(scrna, slot = "counts")[, WhichCells(scrna, ident = '4')])
FetchData # subset Seurat object and return a data frame
subset # subset Seurat object and return a Seurat object
levels(scrna) # reorder the columns in Do.Heatmap
Command(scrna) # lists the functions that were applied to the Seurat object
```