A Secondary Analysis of Integrated Microglial Transcriptomic Data Sets

Statistical Model Estimation – Spring 2021

Term Project by Alyssa Pybus

Table of Contents

Introduction and Background

Microglia form a diverse population of various subtypes and phenotypes

Much like the axiom from high school biology class, “mitochondria are the powerhouse of the cell,” a phrase repeated over and over again in the field of neuroinflammation is “microglia are the resident immune cells of the brain.” An introductory lesson on the brain’s innate ability to clear pathogens or respond to acute injury often involves a discussion on the role of microglia in interpreting environmental cues and adopting various phenotypes to carry out its critical role in maintaining brain tissue homeostasis[1]. Further complicating the picture, other cell types of the innate immune system are present within brain tissue, to include invading inflammatory monocytes and neutrophils as well as tissue-resident perivascular macrophage[2,3].

In 2000, Mills et. al.[4] proposed two distinct macrophage phenotypes, termed M1 and M2, which parallel the Th1 and Th2 (pro-inflammatory and anti-inflammatory, respectively) cellular response to inflammatory stimuli. For better or worse, this paradigm was extended to the microglial population to explain the many observed behaviors of microglia under different environmental cues[5–7]. For example, lipopolysaccharide (LPS), a pro-inflammatory pathogenic-like signaling molecule, can stimulate microglial secretion of pro-inflammatory cytokines and cellular mechanisms of pathogen clearance while anti-inflammatory cytokines such as IL-4 can trigger clearing of inflammatory signaling and restoration of tissue homeostasis[8] (Figure 1).

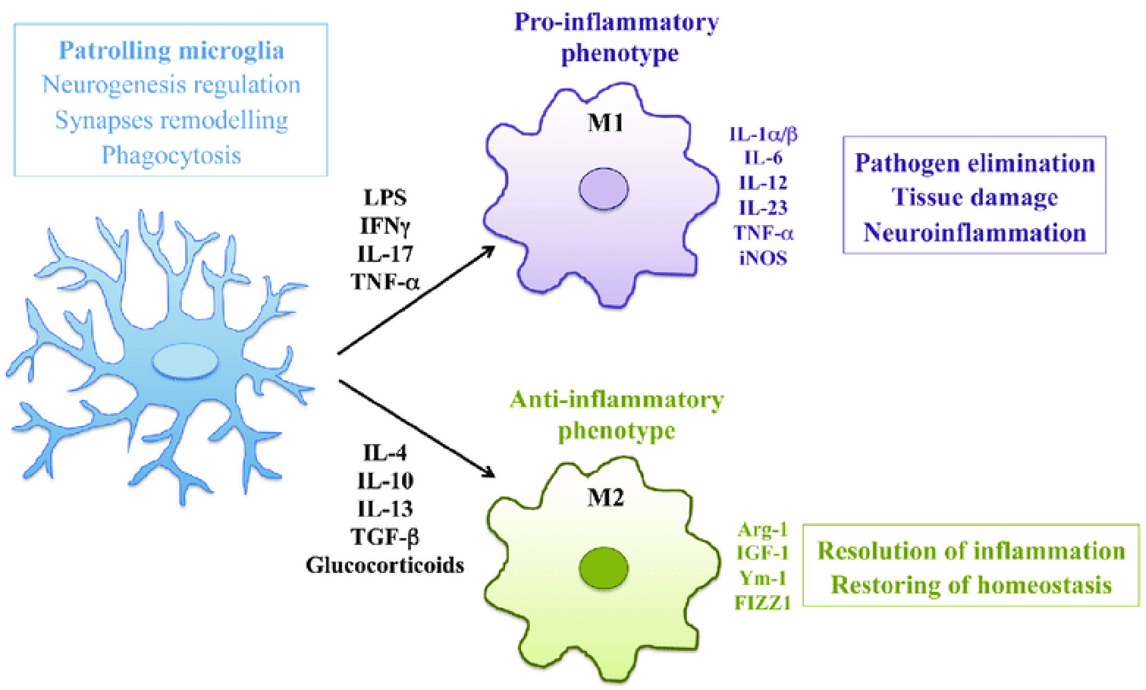


Figure 1. M1/M2 paradigm in microglia. Image from Salvi et. al.**[8]**

Lately, the simplification of microglial (and macrophage) inflammatory phenotypes into M1/M2 classifications has come under intense scrutiny[9,10], as most validating studies are conducted *in vitro* while *in vivo* data proves to be much less convincing and “fail to emerge as isolated pure phenomena” (Ransohoff 2016[10]). Meanwhile, the surge of single-cell methods of genomic, transcriptomic, proteomic, and other methods has resulted in publication of various studies defining diverse microglial subtypes and phenotypes which vary by age, region, and pathological conditions (***Figure 2***)[1,11]. While these studies have proved informative and serve to expand our conceptions of microglial heterogeneity beyond the M1/M2 paradigm, they often are singularly focused on the phenotypic profile of one disease, condition, or injury. Integration of transcriptomic data from multiple studies may serve to elucidate commonalities in microglial phenotypes across various pathological conditions, providing more concise classifications of shared phenotypes and enabling generalized therapeutic strategies of cell modulation.

In this project, two scRNA-Seq studies are combined to analyze commonalities between microglial/monocyte phenotypes in mice subjected to a mild fluid percussive brain injury (Arenson et. al.[12]) or stimulated with LPS (Sousa et. al.[13]). I hypothesize that injury-associated microglia and pathogen-stimulated microglia will display multiple inflammation-associated phenotypes, some of which are common to both stimuli.

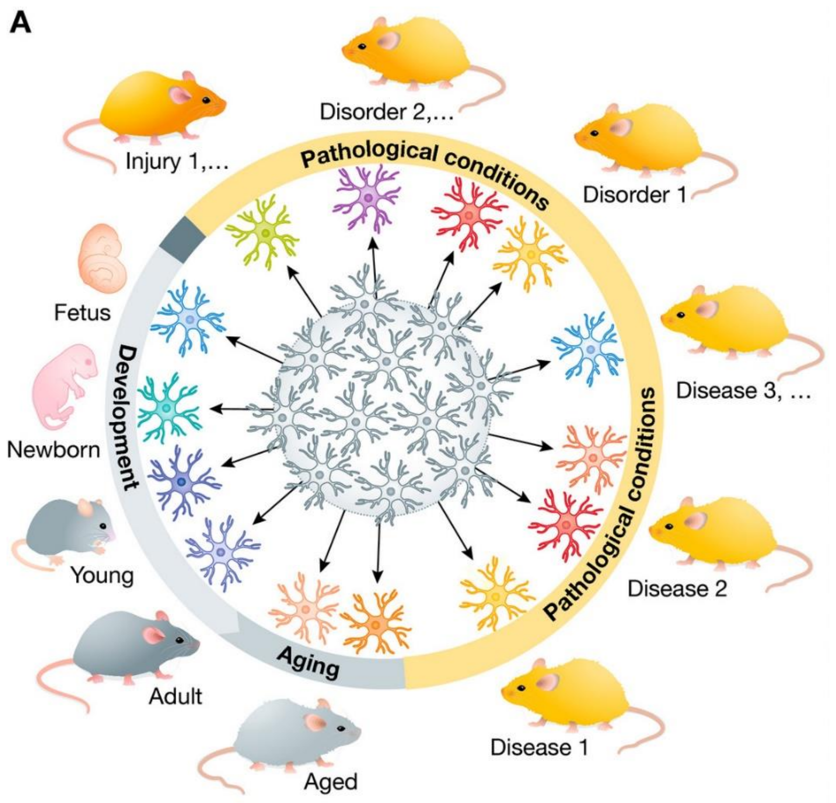


Figure 2. Microglial reaction states. Image from Stratoulias et. al.[11]

scRNA-Seq Data Analysis

Single cell RNA-Sequencing (scRNA-Seq) enables the sequencing of cellular-level transcriptomes of thousands of cells across tens of thousands of genes, vastly improving our ability to define cellular phenotypes and processes in health and pathological states.

Along with the explosion of single-cell publications and increased data and code availability of such studies comes a push for improved methods of data analysis. The analysis workflow for scRNA-Seq data (see ***Figure 3***) typically consists of:

- RNA read alignment/mapping (raw data processing)
- Data pre-processing: filtering, quality control, log normalization, feature selection
- Gene imputation to correct drop-out (Optional)
- Dimensionality reduction of variable features
- Clustering and cell type identification
- Downstream analysis

Variations in data analysis methods exists at each level of the analysis workflow. For example, dimensionality reduction methods can include linear methods such as principal component analysis[14] (PCA, most widely used method) and independent component analysis[15] (ICA), non-linear methods such as t-distributed stochastic neighbor embedding[16] (t-SNE) and uniform manifold approximation and projection[17] (UMAP), deep learning methods such as variational autoencoder[18] (VAE) and deep count autoencoder[19] (DCA), and model-based methods such as zero-inflated factor analysis[20] (ZIFA) (methods reviewed in Xiang et. al.[21]).

Clustering of single cell data can be accomplished via self-organizing maps[22], reversed graph embedding[23], k-means clustering[24], ensemble clustering[25], nearest-neighbor graph clustering[26,27], model-based clustering[28], and various neural network approaches[29–32] (methods reviewed in Duo et. al.[33]).

When selecting an scRNA-Seq workflow, consideration should be given to the dimensionality of the data, the expected number of cell types and subsets, quality of the raw RNA reads and gene alignment, data distribution, and the published performance of the model in comparison to other methods. The majority of methods used in this project derive from the Seurat, an R package collection of various functions for widely-used methods in scRNA-Seq analysis[26,34,35]. An exploratory analysis was conducted with single cell graph neural network[29] (scGNN) for gene imputation and clustering but is not included in this report due to limited computational power and high runtimes.

Dataset Integration Methods for scRNA-Seq Analysis

The increased availability of scRNA-Seq data from various studies on the Gene Expression Ominibus (GEO), a public functional genomics data repository, has enabled integrated secondary analysis of multiple studies to identify commonalities, unique components, and reproducibility in the data. However, challenges to data set integration include addressing batch effects, combining data from multiple scRNA-Seq technologies, and in some cases, accounting for cross-species variability.

While various methods exist for data set integration, this project explored two distinct techniques: Seurat anchor-based integration[34,35] and common factor integration and transfer learning[36,37] (cFIT). Seurat data set integration is widely used and highly cited, recognized as one of the leading methods for integration. cFIT is a recently published method (*PNAS* 2021) which shows promise when compared with Seurat.

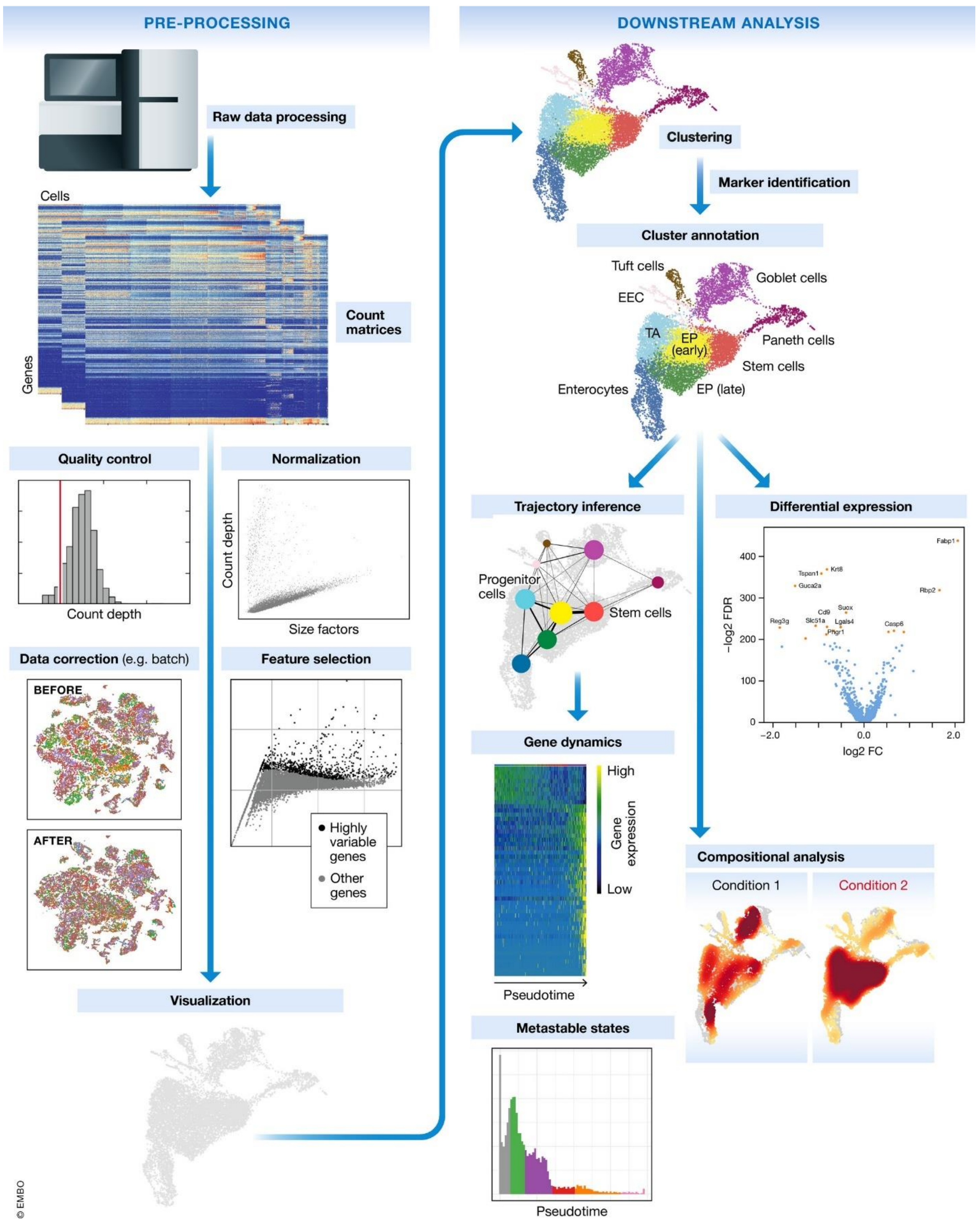


Figure 3. Schematic of typical scRNA-Seq workflow. Image from Luecken et. al.[38]

Selection of Model and Design

Initial analysis of each individual data set and the combined data set is carried out with the Seurat[26,34,35] package in R, a well-established and highly cited collection of scRNA-Seq functions. The standard workflow as recommended by the Seurat developers will be used as a baseline with which to test two data set integration methods (Seurat anchor-based[34] and cFIT[36]) and one clustering method (Seurat nearest-neighbor graph clustering[26,27]).

Anchor-based data set integration with Seurat

The Seurat workflow for data set integration[39] implements functions developed in Stuart et. al.[34] for identifying "anchor" cell pairs between data sets that facilitate removal of batch effects (***Figure 4***). Briefly, canonical correlation and L2 normalization of the resulting vectors is performed to project the data into a shared correlation structure. Pairs of closely aligned anchor cells across the data sets are identified to guide integration and are assigned a consistency score within its neighborhood. These anchors and scores are used to compute correction vectors for each cell into the reference space.

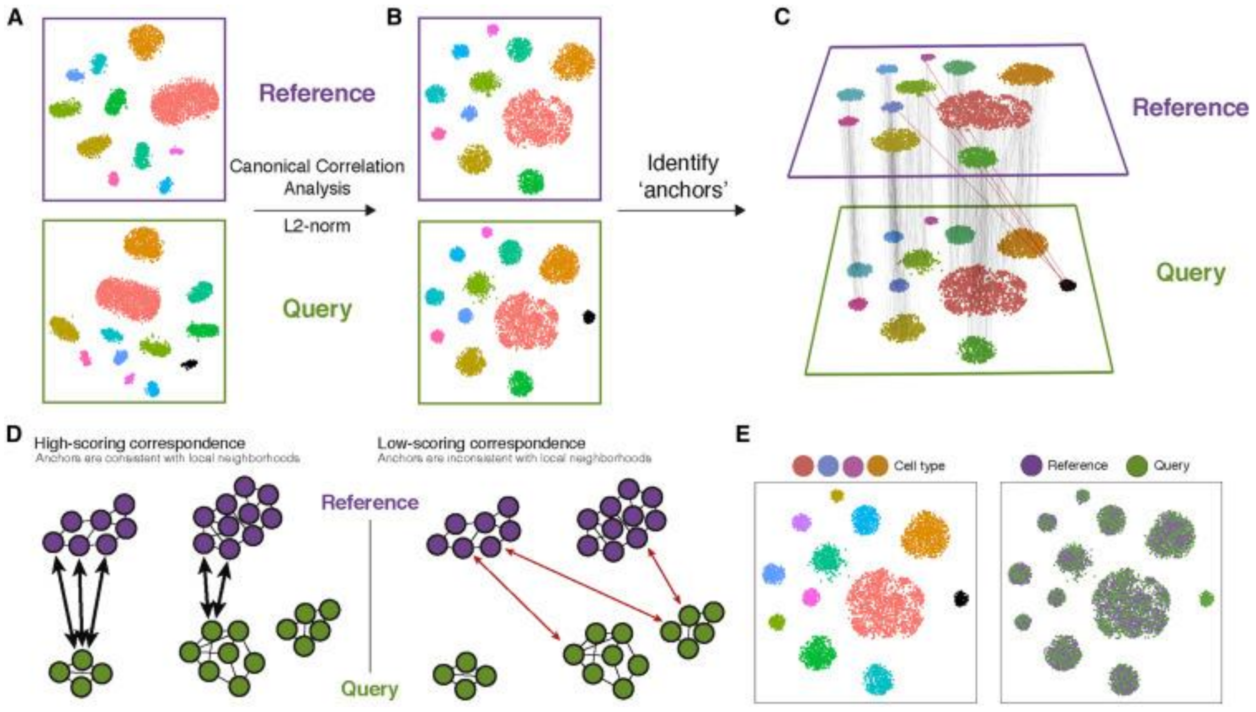

A
B
C
Reference
Canonical Correlation Analysis
L2-norm
Query
Identify 'anchors'
Reference
Query
D
High-scoring correspondence
Anchors are consistent with local neighborhoods
Low-scoring correspondence
Anchors are inconsistent with local neighborhoods
Reference
Query
E
Cell type
Reference
Query

Figure 4. Schematic overview of Seurat anchor-based integration. Image from Stuart et. al.[34]

Anchor pairs are calculated by maximizing the correlation of $\boldsymbol{X}_u$ and $\boldsymbol{Y}_v$, where $\boldsymbol{X}$ and $\boldsymbol{Y}$ represent the two data sets being integrated and u and v represent projection vectors.

$$\max_{u,v} u^T X^T Y v \text{ subject to } \|u\|_2^2 \leq 1, \|v\|_2^2 \leq 1$$

For each anchor (cell *c* and anchor *i*), weighted distances are computed and multiplied by the anchor score ***S***.

$$D_{c,i} = \left(1 - \frac{dist(c,a_i)}{dist(c,a_{k.weight})}\right) S_{a_i}$$

Batch correction is conducted as follows (adapted from Stuart et. al. methods[34]): The matrix ***B*** is calculated, representing the difference between the two expression vectors for every pair of anchor cells. The transformation matrix, ***C***, is then calculated using ***B*** and ***W***. Lastly, the integrated expression matrix $\hat{\boldsymbol{Y}}$ is calculated by subtracting the transformation matrix from the original expression matrix.

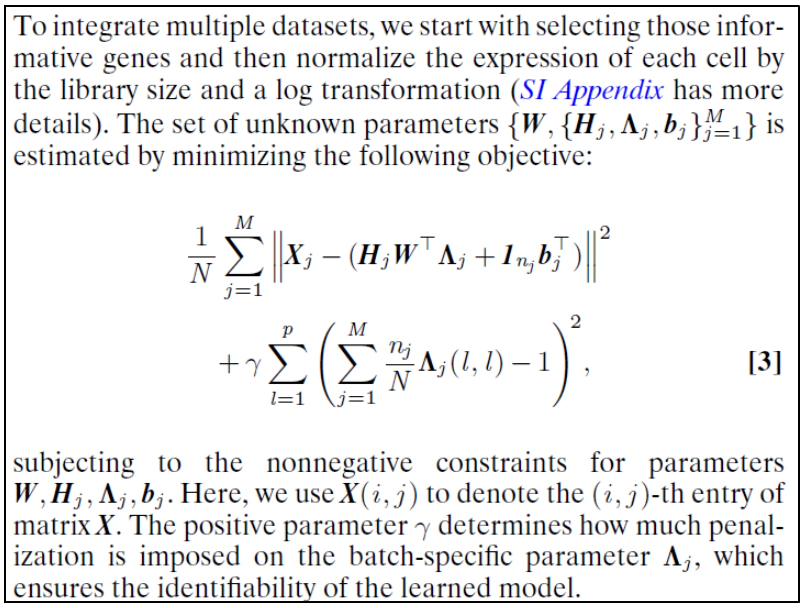
$$B = Y\,[,a] - X\,[,a]$$

$$C = BW^T$$

$$\widehat{Y} = Y - C$$

Common factor integration and transfer learning for combining data sets

Briefly, cFIT captures batch effects across experiments, technologies, and species using an iterative non-negative matrix factorization to identify model parameters for projection into a shared common factor space. The creators explain the “objective function” to be minimized in ***Figure 5*** and the overall workflow is detailed in ***Figure 6***.



To integrate multiple datasets, we start with selecting those informative genes and then normalize the expression of each cell by the library size and a log transformation (*SI Appendix* has more details). The set of unknown parameters $\{\boldsymbol{W}, \{\boldsymbol{H}_j, \boldsymbol{\Lambda}_j, \boldsymbol{b}_j\}_{j=1}^{M}\}$ is estimated by minimizing the following objective:

$$\frac{1}{N}\sum_{j=1}^{M}\left\|\boldsymbol{X}_j - (\boldsymbol{H}_j\boldsymbol{W}^\top\boldsymbol{\Lambda}_j + \boldsymbol{1}_{n_j}\boldsymbol{b}_j^\top)\right\|^2 + \gamma\sum_{l=1}^{p}\left(\sum_{j=1}^{M}\frac{n_j}{N}\boldsymbol{\Lambda}_j(l,l) - 1\right)^2, \quad [3]$$

subjecting to the nonnegative constraints for parameters $\boldsymbol{W}, \boldsymbol{H}_j, \boldsymbol{\Lambda}_j, \boldsymbol{b}_j$. Here, we use $\boldsymbol{X}(i,j)$ to denote the (i,j)-th entry of matrix $\boldsymbol{X}$. The positive parameter γ determines how much penalization is imposed on the batch-specific parameter $\boldsymbol{\Lambda}_j$, which ensures the identifiability of the learned model.

Figure 5. cFIT objective function. Image from Peng et. al.[36] where **W** represents the shared nonnegative factor matrix, **H** is the nonnegative factor loading matrix, **Λ** represents domain-specific effects of gene expression, and **b** captures domain-specific shifts.

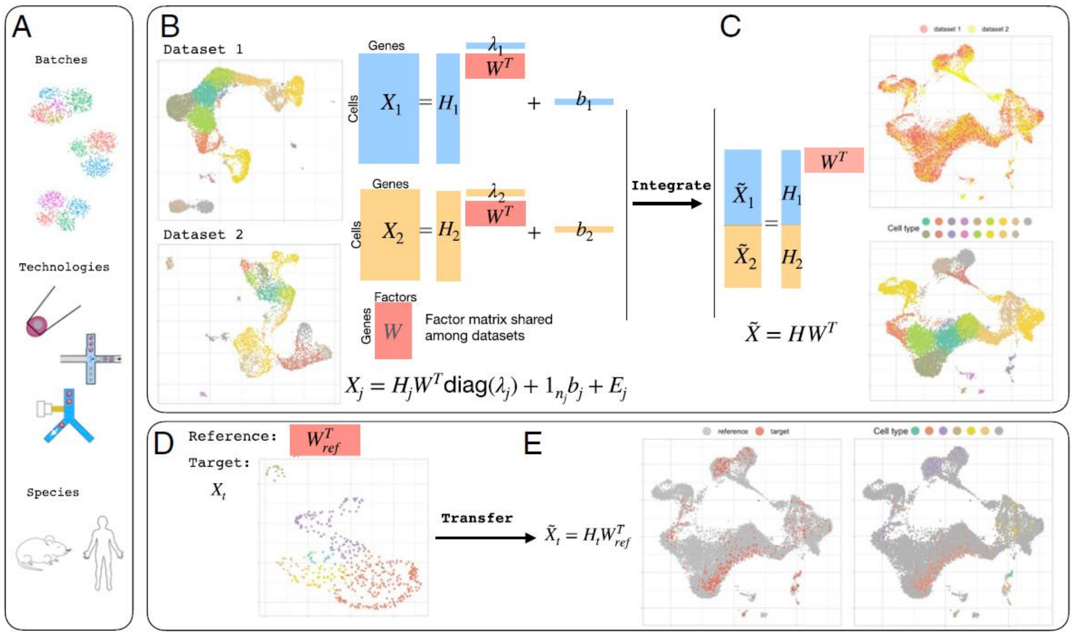


Figure 6. cFIT integration and transfer approach overview. Image from Peng et. al.[36]

Nearest-neighbor graph clustering with Seurat

The Seurat workflow for cell clustering[40] first includes unbiased linear dimensional reduction of selected variable features (usually 2000 genes) via principal component analysis, the eigenvector decomposition of the covariance matrix. The dimensionality of the data is determined by an analysis of the percent variance of the data explained by each principal component. For my analyses, I used the number of principal components needed to explain 90% of the variance (***Figure 7***).

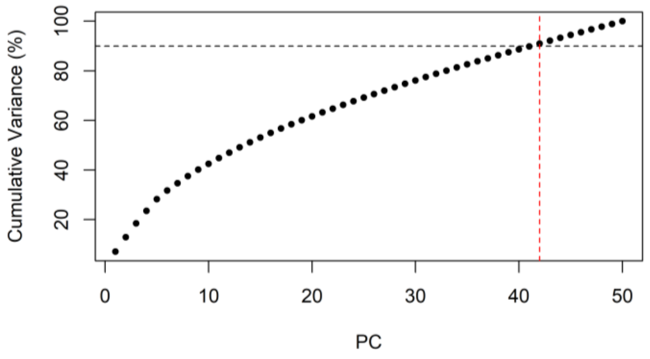
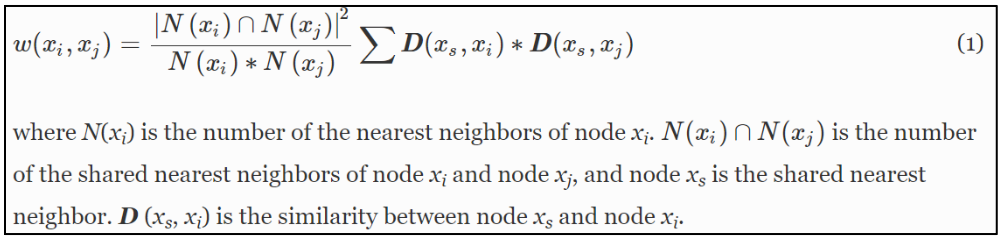


Figure 7. Cumulative variance explained by principal components. Linear dimensional reduction was carried out by PCA. The 'dimension' of the data was defined as the number of PCs needed to reach at least 90% of explained variance. For the Arneson et. al.[12] data, 42 PCs are needed to reach the 90% threshold.

Next, Seurat applies a graph-based clustering method which takes the principal components as input and calculates a distance metric (i.e. Euclidean distance) to drive the clustering. Every cell is embedded in a k-nearest neighbor graph[41] with edges between similar cells and edge weights are refined based on Jaccard similarity (***Figure 8***).

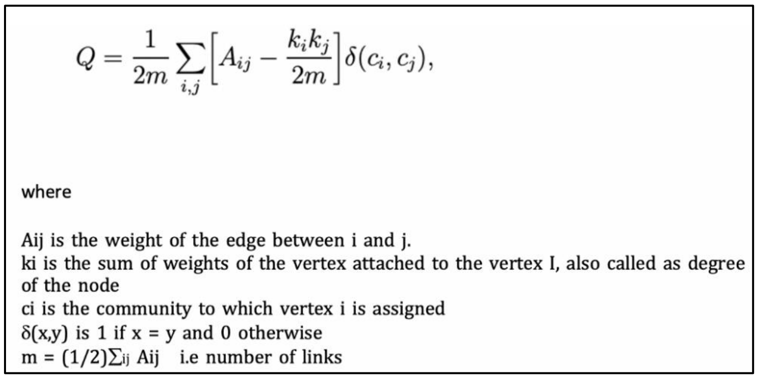


$$w(x_i, x_j) = \frac{|N(x_i) \cap N(x_j)|^2}{N(x_i) * N(x_j)} \sum \boldsymbol{D}(x_s, x_i) * \boldsymbol{D}(x_s, x_j) \qquad (1)$$

where $N(x_i)$ is the number of the nearest neighbors of node x_i. $N(x_i) \cap N(x_j)$ is the number of the shared nearest neighbors of node x_i and node x_j, and node x_s is the shared nearest neighbor. $\boldsymbol{D}(x_s, x_i)$ is the similarity between node x_s and node x_i.

Figure 8. Edge weights for KNN graph. Image from Zhu et. al.[41]

Modularity optimization techniques are then applied to group similar cells together in distinct communities. This is carried out by optimization of a standard modularity function. The default modularity optimization algorithm used is the Louvain method for community detection[42] (***Figure 9***).



$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j),$$

where

Aij is the weight of the edge between i and j.
ki is the sum of weights of the vertex attached to the vertex I, also called as degree of the node
ci is the community to which vertex i is assigned
δ(x,y) is 1 if x = y and 0 otherwise
m = (1/2)$\sum_{ij}$ Aij i.e number of links

Figure 9. Louvain method modularity function. The Louvain method uses modularity (Q) as its optimization function for community detection. Image from Mishra 2019[43].

Assessment of Model Performance

Arneson et. al. analysis with Seurat

The Arneson et. al.[12] data set contains all brain cell types from hippocampal tissue of sham-injured and injured mice subjected to a mild fluid percussive injury. The count matrix was available on the GEO repository with series record GSE101901. The Seurat workflow was conducted to identify clusters of microglial or monocyte cell populations for integration into following analyses. See Appendix 1: Arneson et. al. analysis in R using Seurat.

Quality control of the data included filtering out cells with too many or too few identified genes (***Figure 10***). The number of RNA counts vs the number of RNA features is expected to follow a linear pattern. Too few genes indicate the measurement may have come from a cell-less droplet, whereas too many may indicate multiple cells sticking together during sequencing. In addition, cells with an outsized proportion of mitochondrial RNA indicate cell death and were also removed from the analysis.

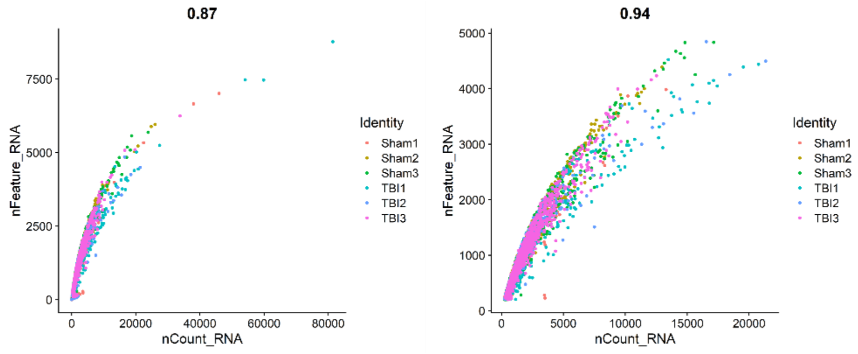


Figure 10. Quality control of Arneson et. al. Cells with greater than 5000 or less than 200 RNA features were removed from the analysis. Cells with >75% mitochondrial DNA were also removed. Linearity is improved post-filter.

The top 2,000 variable features (high cell-to-cell variation) are selected for further analysis (***Figure 11***). According to the Satija lab website for Seurat, "We and others have found that focusing on these genes in downstream analysis helps to highlight biological signal in single-cell datasets."[40]

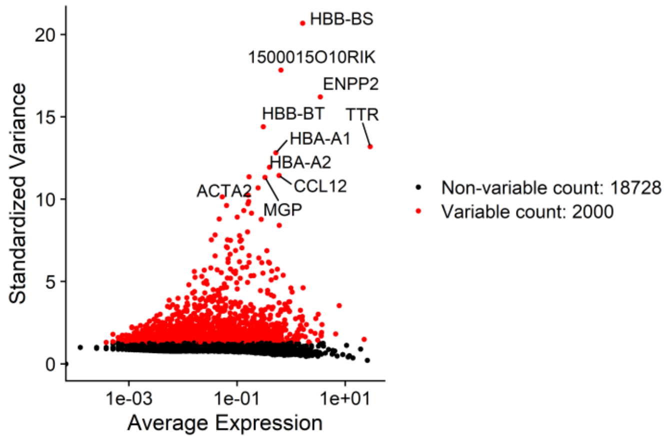


Figure 11. Variable features. The top 2,000 variable genes are used for clustering to improve biological signal.

Next, principal component analysis is conducted on the 2,000 identified features. Loading bar plots, score plots, heatmaps (***Figure 12***), and variance charts (***Figure 7***) are used to determine the appropriate number of PCs to include in clustering and visualization.

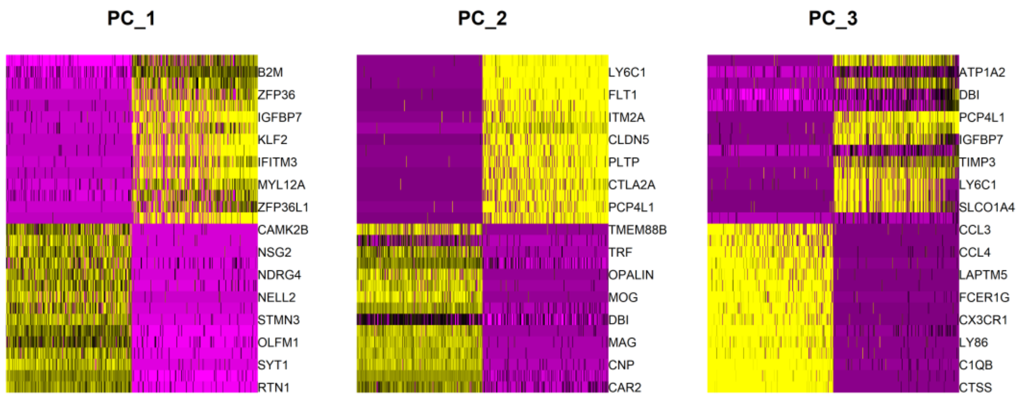


Figure 12. PCA heatmaps. Heatmaps show the genes of highest weightings in each PC.

Next, clustering is conducted with graph-based KNN and the Louvain method, using the 42 top PCs as input (which together explain >90% of variance). The 42 PCs are also subjected to nonlinear dimensional reduction via UMAP for data visualization. The data is projected onto a 2D plot of UMAP components and cells are colored by their assigned cluster (***Figure 13***).

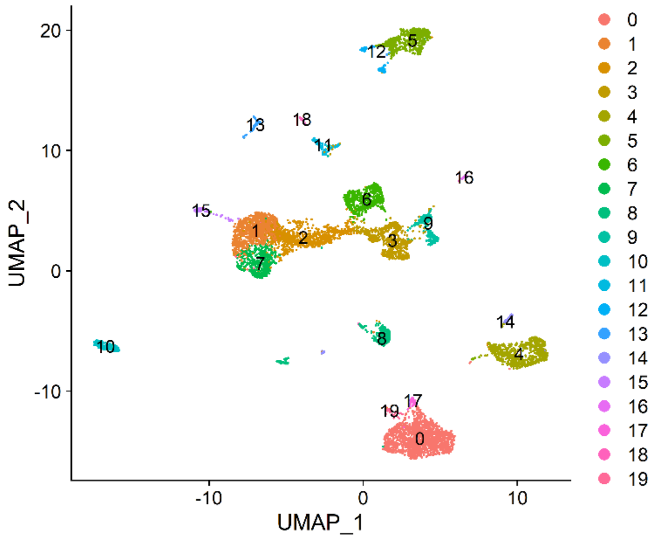


Figure 13. Cell clusters within Arneson et. al. UMAP of 42 PCs colored by identified KNN/Louvain clusters.

To assess the success of the model at distinguishing cell types, microglial and monocyte cell clusters are identified by an analysis of cell markers. TMEM119 and CCR5 (***Figure 14***, ***Figure 15***) are microglia-specific markers, while CTSS and CD14 are expressed in both populations. Analysis of each gene reveals a cluster 4 to be microglia and cluster 14 to be other immune cells. Further analysis is restricted to cells of these clusters.

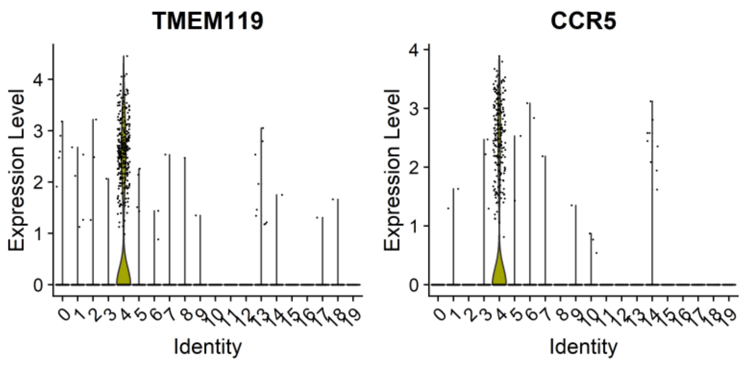


Figure 14. TMEM119 and CCR5 expression by cluster. Cluster 4 contains microglia.

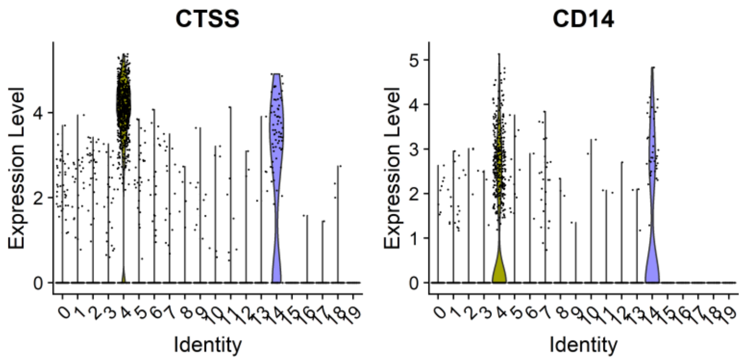


Figure 15. CTSS and CD14 expression by cluster. Clusters 4 and 14 contain immune cells.

After identification of the immune cells within the Arneson et. al. data, the Seurat workflow was re-implemented on this filtered subset (identification of variable features, PCA, clustering, data visualization with UMAP). As before, cell types are identified by analysis of cell-specific markers (***Figure 16***, ***Figure 17***, ***Figure 18***).

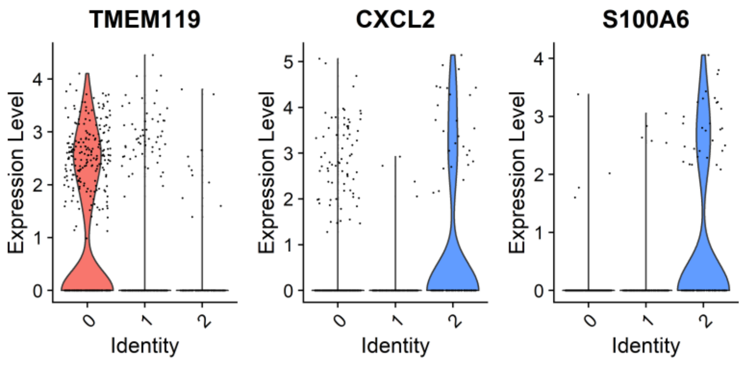


Figure 16. Cell-specific markers identify immune cell clusters. TMEM119 represents microglia while S100A6 and CXCL2 are found in peripheral monocytes/macrophage.

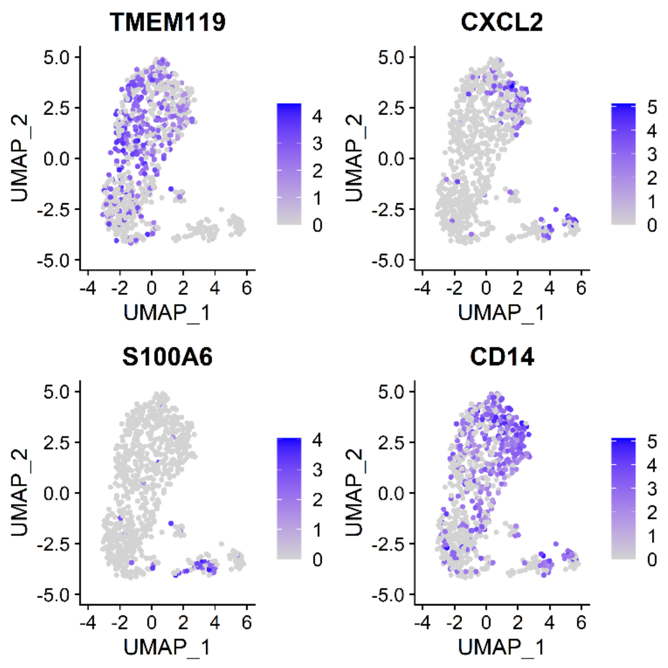


Figure 17. Gene expression of cell specific markers in UMAP space.

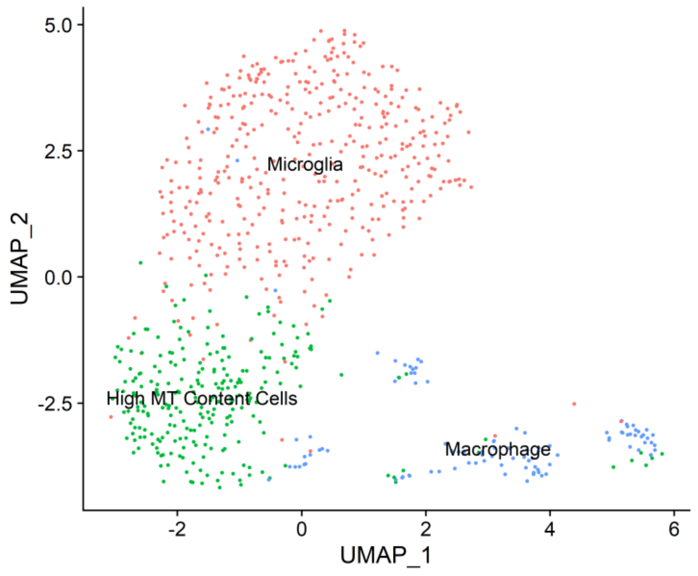


Figure 18. Cell type clusters in Arneson et. al. immune cells.

Sousa et. al. analysis with Seurat

The Sousa et. al. data contains microglia (isolated by flow cytometry) from mice injected with LPS or saline control. The count matrices were available on the GEO repository with series record GSE 115571. The Seurat workflow was conducted to identify microglial phenotypes. See Appendix 2: Sousa et. al. analysis in R using Seurat.

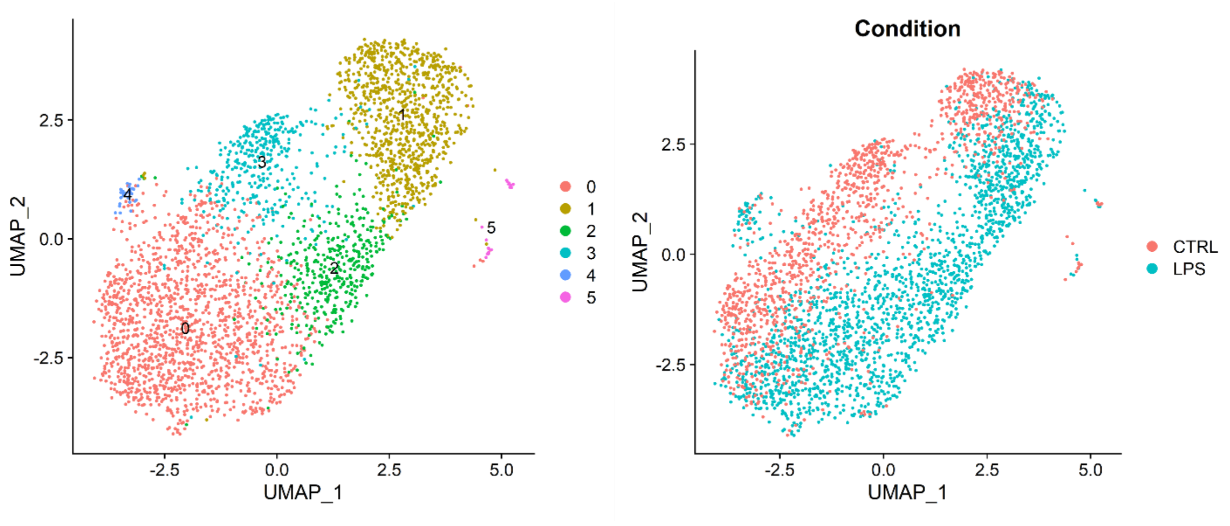


Figure 19. Sousa et. al. UMAP by cell clusters and condition. Visual analysis indicates that batch effects may remain within clusters. Clusters 0 and 1 contain shared microglial phenotypes while 2 and 3 are specific to condition.

Anchor-based integration with Seurat

Seurat integration was applied to the Arneson et. al. and Sousa et. al. data sets using recommended parameters from the published workflow[39]. See Appendix 3: Data set integration in R using Seurat.

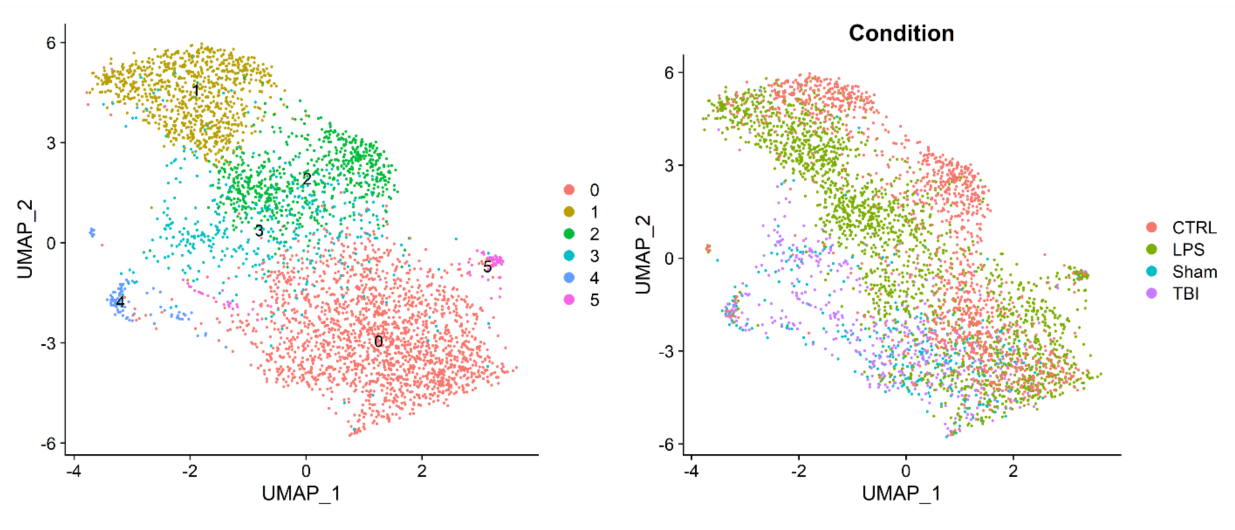


Figure 20. Integrated data from Arneson et. al. and Sousa et. al. UMAP by cluster and condition. Clusters vary by experiment and condition. Cluster 0 contains cells from all 4 conditions.

Table 1. Top gene markers of each cluster.

Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
H3F3B	MALAT1	CTSS	TMSB4X	PTN	IFIT1
UBC	GM26924	HEXB	RPLP1	SPARCL1	RSAD2
CST3	MT-RNR2	C1QB	MT-ND4	CLU	IFIT3
JUNB	DST	ITM2B	ACTB	CAR2	H2-AA
EGR1	MACF1	B2M	MT-ND1	ATP1A2	H2-EB1
C1QC	MT-RNR1	CX3CR1	MT-ND2	SLC1A2	PLAC8
ZFP36	RBM25	TAX1BP1	MT-ND5	ALDOC	IFIT2
TMSB4X	ZEB2	CD68	CST3	TTYH1	H2-AB1
LY86	CT030684.1	CALR	RPS12	GPM6B	S100A6
C1QB	ASH1L	PDIA6	MT-CYTB	PPAP2B	MX1
RPLP1	ATRX	CYBA	C1QC	SCG3	OASL1
SLC25A5	FYB	YWHAH	RPS29	CPE	ISG15
UBB	LUC7L3	SIRPA	RPS9	MT3	CRIP1
HSPA8	SFRS18	ARHGDIB	LY86	CAMK2A	PYHIN1
CD53	SON	ADAM10	RPS27A	NDRG2	MNDA
CYBA	ARHGAP5	MBNL1	FTH1	MT2	IRF7
IER2	NKTR	LRRC58	RPS21	PCSK1N	IFITM3
RPL32	MYCBP2	PYCARD	RPL27A	SCD2	MX2
CD14	XIST	CAMK1	RPL13A	GSTM1	RTP4
EIF1	DIAP2	NPC2	RPS23	PRNP	VIM
FCER1G	UBN2	CD2AP	EEF1A1	MOBP	SLFN5
RPS14	RRBP1	MSN	RPL11	PLP1	IFI27L2A
RPS19	PRPF4B	EVI2A	TPT1	DBI	CXCL10
RPL3	CHD7	HEXA	RPL41	MBP	PHF11B
JUN	SNORD49B	C1QA	RPLP0	MT-TP	CD74
RPS18	RBM26	ATP6V0B	FCER1G	KIF5A	LGALS1
RPL14	STAB1	LCP1	RPS2	MT1	PHF11D
RPLP2	CHD9	CYFIP1	RPL13	PLA2G7	IFI204
C1QA	WNK1	FERMT3	B2M	MT-CYTB	OASL2
FTH1	GOLGA4	TMEM59	RPS26	RPS27	SAMD9L
CCL3	PRRC2C	HK2	RPLP2	MT-ND4	MS4A4C
RPL26	BOD1L	P2RY13	RPS15	MEG3	ZBP1
CFL1	RANBP2	TMEM50A	RPS5	PTGDS	IL1B
RPL35A	ZFC3H1	CD37	CD81	MT-ND1	SAP30
SEPP1	RP24-	MYLIP	CALM1	APOE	AI607873
FOS	312B12.1	LPCAT2	RPL18A	MT-CO1	SP100
RPS11	SRRM2	LAPTM4A	RPL14	MT-TC	TOR3A
RPL13	ITGAM	LAMP2	C1QA	CKB	IFIH1
NFKBIA	ROCK2	DNAJC7	PFN1	RPS27A	H2-T23
CD68	ZCCHC7	TSC22D4	RPL22	SYT11	NAMPT
RPS3A1	SSH2	KMT2E	RPS24	MT-TV	TGFBI
BTG2	CEP110	MANF	RPSA	RPL21	BST2
CLTA	ZFML	F11R	MT-CO1	SEPW1	DDX58
RPL41	AKAP9	SEC62	RPS11	RPL31	CD52
RPL37	NSD1	SPARC	RPL32	TPT1	H2-D1
TYROBP	NAV3	TMED10	RPS15A	RPL9	TPT1
RPL34	PRPF38B	MLEC	RPS3A1	RPL11	ATOX1
LGMN	FUS	CTSD	PPIA	RPS29	CALM1
RPL37A	HERC2	SLFN2	RPS8	MT-ND2	SRGN
RPS9	ELMO1	IL6RA	RPL35A	MT-ND5	B2M

Table 2. Gene ontology enrichment analysis of each cluster.

Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
-peptide biosynthetic process -peptide metabolic process -regulation of cell death -response to organic substance -regulation of neuron death -regulation of defense response	-RNA splicing -RNA metabolic process -regulation of cytoskeleton organization -organic cyclic compound metabolic process -regulation of organelle organization	-regulation of localization -leukocyte activation -regulation of cellular component organization -leukocyte cell-cell adhesion -positive regulation of phagocytosis -immune system process	-peptide biosynthetic process -cellular nitrogen compound biosynthetic process -rRNA processing -ncRNA processing -electron transport coupled proton transport	-peptide metabolic process -ATP metabolic process -glial cell differentiation -response to stress -regulation of tau-protein kinase activity -response to drug	-peptide biosynthetic process -response to other organism - innate immune response - response to interferon-gamma - regulation of tumor necrosis factor production

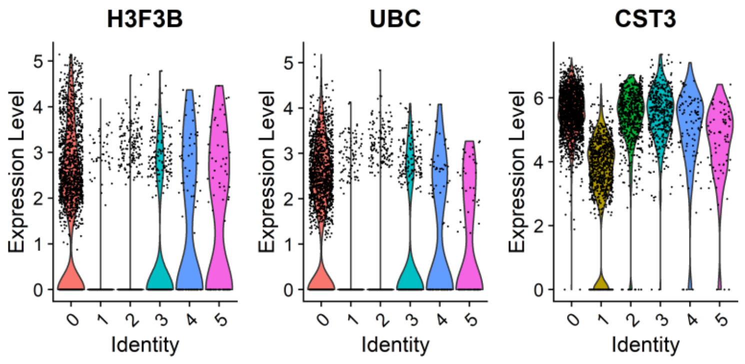


Figure 21. Top genes in Cluster 0.

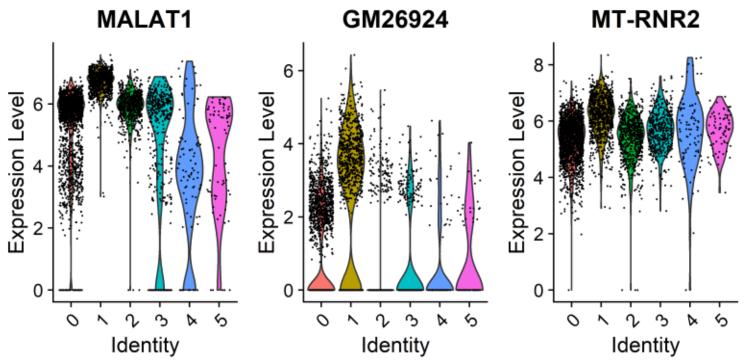


Figure 22. Top genes in Cluster 1.

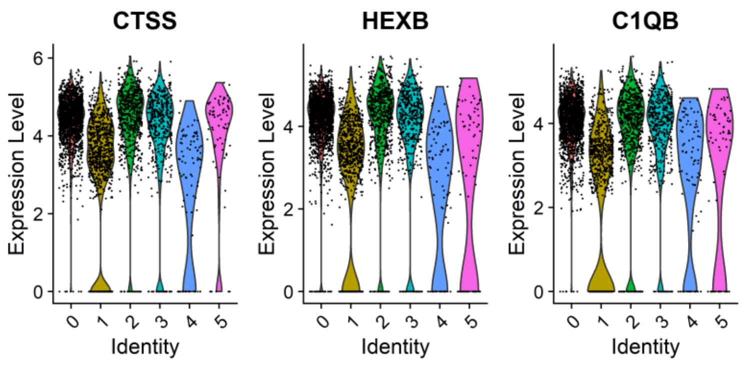


Figure 23. Top genes in Cluster 2.

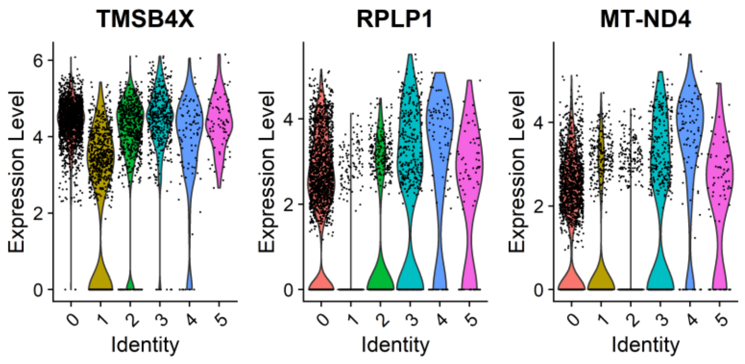


Figure 24. Top genes in Cluster 3.

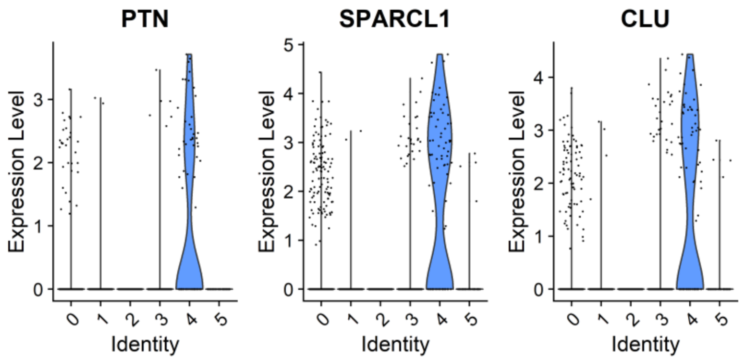


Figure 25. Top genes in Cluster 4.

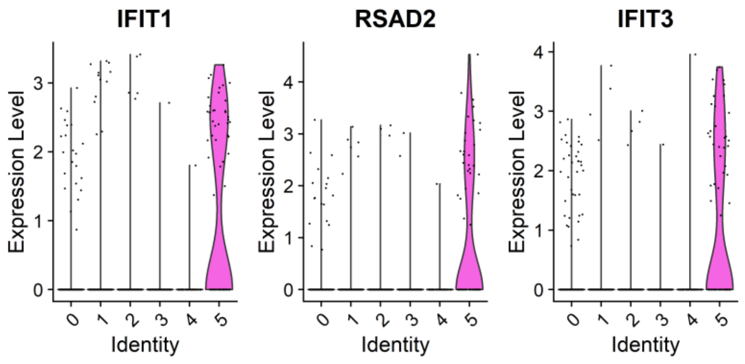


Figure 26. Top genes in Cluster 5.

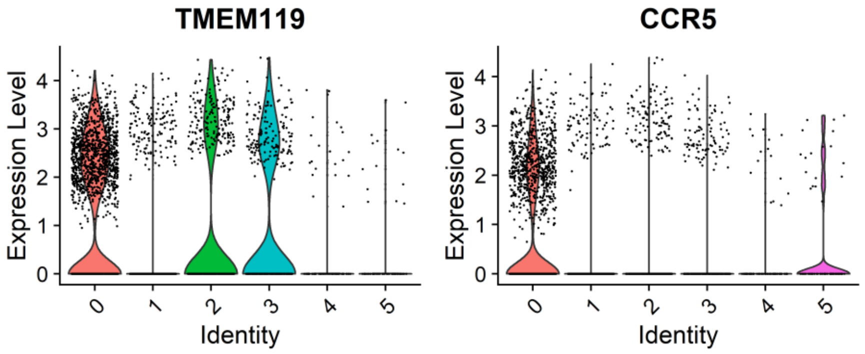


Figure 27. Microglial-specific genes by cluster.

Integration with cFIT

The Arneson et. al. and Sousa et. al. data sets were input to the cFIT workflow as described in the cFIT tutorial[44]. After integration, results were visualized via UMAP projection and colored by experimental condition and Seurat cluster. See Appendix 4: Combining the Data Sets Using cFIT in R.

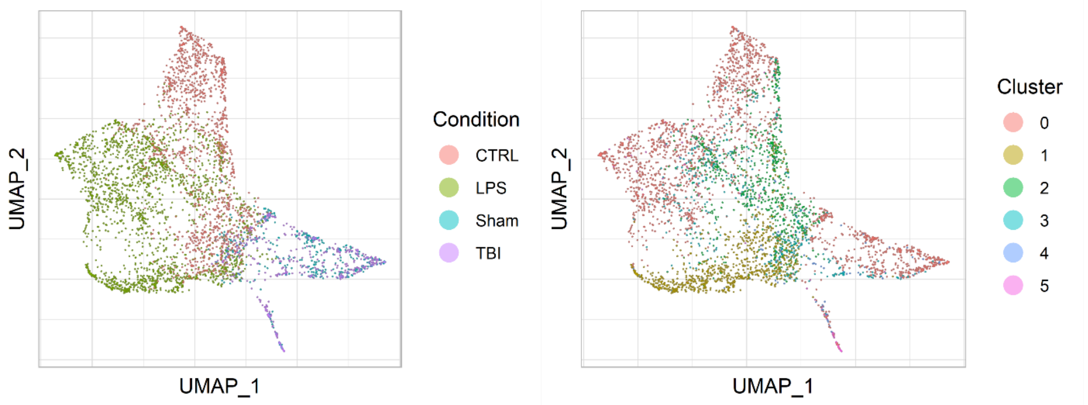


Figure 28. UMAP projections of data integrated via cFIT. Graphical separation of clusters is less apparent than with Seurat integration.

Conclusion

Single cell RNA-Sequencing analysis was successfully conducted for each data set individually and after integration via two methods, the anchor-based Seurat method and cFIT. The integrated myeloid data set containing data from both Arneson et. al. (mild fluid percussive injury) and Sousa et. al. (LPS stimulation) organized into six distinct clusters of various proportions from each experiment. Gene ontology and analysis of individual gene markers help understand the microglial/myeloid phenotypes within each cluster.

Cluster 0 contains sizeable populations of cells from each of the four conditions (TBI, sham, LPS, saline control). TMEM119 and CCR5 show high expression within this cluster, indicating a microglial cell type. Gene ontology processes enriched in cluster 0 include regulation of cell death and neuron death, response to organic substances, and metabolic upregulation, indicating a potential deleterious population.

Cluster 1, 2, and 5 appear to be evenly mixed between LPS and saline control cells, indicating the likelihood of leftover batch effects not removed by integration. This is similar to Cluster 4 which contains a mix of TBI and sham cells. Cluster 2 expresses relatively high levels of TMEM119 and CCR5, indicating microglial cells. Its gene ontology processes include leukocyte activation and cell-cell adhesion and immune system processes. This cluster may represent pro-inflammatory responses to LPS.

Cluster 3 contains mostly TBI cells, indicating a likely injury-associated phenotype. TMEM119 and (to a lesser extent) CCR5 indicate high expression, indicating a microglial cell population. Gene ontology processes include peptide biosynthesis, rRNA processing, and electron transport coupled proton transport, indicating a possible “constructive” phenotype.

One of the limitations of the data selected is the uneven distribution of cell populations from each experiment. After selecting for myeloid cells, the Arneson et. al. data set was limited to 773 cells total, while the Sousa et. al. set contained 3167. This uneven split might complicate the identification of anchor cells and resulting transformations. As the Sousa data deliberately selected for only microglia, inclusion of other myeloid cells within the Arneson set may have also confounded the transformation.

A second limitation is the high number of mitochondrial RNAs found in the Arneson data set. Abundance of MT-RNA may indicate dead or dying cells at the time of scRNAseq reading, and the Seurat workflow recommends filtering out all cells with >5% MT-RNA. An unusually high number of samples contained MT-RNA levels greater than 25%, suggesting that many cells may have died in the process of reading their RNA, which may confound the results of the study.

Appendix 1: Arneson et. al. analysis in R using Seurat

The following code was run in R to conduct both the initial Arneson et. al.[12] analysis of multiple cell types as well as the filtered myeloid cell type analysis.

```
# PRELIMINARIES ################################

rm(list=ls())
setwd(dirname(rstudioapi::getActiveDocumentContext()$path))

#source("C:/Users/apybus3/Box/Wood Lab R Functions/WoodLabFunctions.R")

if (!require("pacman")) install.packages("pacman")
if (!require("BiocManager")) install.packages("BiocManager")
pacman::p_load(readxl,tidyverse,rio,Seurat,data.table,cFIT)

# 1 - PREP ARNESON DATA ##########

# Import the data
df_TBI_in <- import("GEO Data Sets/DropSeqTBI.digital_expression.txt")
genes_TBI = str_to_upper(df_TBI_in$V1)

# Create count matrix
TBI_counts = df_TBI_in[,2:ncol(df_TBI_in)] %>% as.matrix()
rownames(TBI_counts)=genes_TBI

# Create data frame of metadata
meta_TBI = data.frame(Condition = case_when(
  str_detect(colnames(TBI_counts),"TBI") ~ "TBI",
  str_detect(colnames(TBI_counts),"Sham") ~ "Sham"
),
MouseID = case_when(
  str_detect(colnames(TBI_counts),"TBI1") ~ "TBI1",
  str_detect(colnames(TBI_counts),"TBI2") ~ "TBI2",
  str_detect(colnames(TBI_counts),"TBI3") ~ "TBI3",
  str_detect(colnames(TBI_counts),"Sham1") ~ "Sham1",
  str_detect(colnames(TBI_counts),"Sham2") ~ "Sham2",
  str_detect(colnames(TBI_counts),"Sham3") ~ "Sham3"
))
rownames(meta_TBI) = colnames(TBI_counts)

# Create Seurat object for use in Seurat package codes
TBI_SO <- CreateSeuratObject(counts = TBI_counts, project = "TBI",meta.data =
meta_TBI) #, min.cells = 3, min.features = 200)

# Quality control: see distribution of number of features, number of counts
in all samples
TBI_SO[["percent.mt"]] <- PercentageFeatureSet(TBI_SO, pattern = "^MT-")
png("Figures/1 - Prep Arneson Data/MT RNA
Violins.png",res=600,units="in",height=5,width=7)
VlnPlot(TBI_SO, features ="percent.mt")
dev.off()
png("Figures/1 - Prep Arneson Data/nFeature
Violins.png",res=600,units="in",height=5,width=7)
VlnPlot(TBI_SO, features ="nFeature_RNA")
dev.off()
```

```
png("Figures/1 - Prep Arneson Data/nCount
Violins.png",res=600,units="in",height=5,width=7)
VlnPlot(TBI_SO, features = "nCount_RNA")
dev.off()
png("Figures/1 - Prep Arneson Data/Features vs Counts Pre-
Filter.png",res=600,units="in",height=5,width=6)
FeatureScatter(TBI_SO, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
dev.off()

# Filter out cells with too few (possibly droplets) or too many counts
(possibly multiple cells) and cells with too much mito RNA
TBI_SO <- subset(TBI_SO, subset = nFeature_RNA > 200 & nFeature_RNA < 5000 &
percent.mt < 75)

png("Figures/1 - Prep Arneson Data/Features vs Counts Post-
Filter.png",res=600,units="in",height=5,width=6)
FeatureScatter(TBI_SO, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
dev.off()

png("Figures/1 - Prep Arneson Data/MT RNA Violins Post-
Filter.png",res=600,units="in",height=5,width=7)
VlnPlot(TBI_SO, features ="percent.mt")
dev.off()

# Normalize data with log transform
TBI_SO <- NormalizeData(TBI_SO)
TBI_SO <- FindVariableFeatures(TBI_SO, selection.method = "vst", nfeatures =
2000)

# Identify the 10 most highly variable genes
top10 <- head(VariableFeatures(TBI_SO), 10)

# plot variable features
png("Figures/1 - Prep Arneson Data/Variable
Features.png",res=600,units="in",height=4,width=6)
VariableFeaturePlot(TBI_SO)
LabelPoints(plot = VariableFeaturePlot(TBI_SO), points = top10, repel = TRUE)
dev.off()

#Scale the data for PCA
all.genes <- rownames(TBI_SO)
TBI_SO <- ScaleData(TBI_SO, features = all.genes)

# Conduct PCA
TBI_SO <- RunPCA(TBI_SO, features = VariableFeatures(object = TBI_SO))

png("Figures/1 - Prep Arneson Data/PCA Loadings Charts PC1-
2.png",res=600,units="in",height=4.5,width=6)
VizDimLoadings(TBI_SO, dims = 1:2, reduction = "pca")
dev.off()

png("Figures/1 - Prep Arneson Data/PCA Scores Plot PC1-
2.png",res=600,units="in",height=4.5,width=6)
DimPlot(TBI_SO, reduction = "pca",group.by = "MouseID")
dev.off()
```

```
png("Figures/1 - Prep Arneson Data/PCA Heatmaps PC1-
12.png",res=600,units="in",height=11.5,width=7)
DimHeatmap(TBI_SO, dims = 1:12, cells = 500, balanced = TRUE)
dev.off()

# # Determine appropriate dimension of the data by JackStraw analysis of PCs
# TBI_SO <- JackStraw(TBI_SO, num.replicate = 100, dims=30)
# TBI_SO <- ScoreJackStraw(TBI_SO, dims = 1:30)
#
# png("Figures/1 - Prep Arneson Data/Jack Straw
Plot.png",res=600,units="in",height=4,width=6)
# JackStrawPlot(TBI_SO, dims = 1:30)
# dev.off()

# PC Percent Variance Analysis
pct <- TBI_SO[["pca"]]@stdev / sum(TBI_SO[["pca"]]@stdev) * 100 # Determine
percent of variation associated with each PC
cumu <- cumsum(pct)  # Calculate cumulative percents for each PC

# Determine which PC exhibits cumulative percent greater than 90% and %
variation associated with the PC as less than 5
co1 <- which(cumu > 90 & pct < 5)[1]

png("Figures/1 - Prep Arneson Data/PCA Cumulative
Variance.png",res=600,units="in",height=4,width=6)
plot(x=1:50,y=cumu,xlab="PC",ylab="Cumulative Variance (%)",pch=20)+
  abline(h=90,lty=2) +
  abline(v=co1,lty=2,col="red")
dev.off()

png("Figures/1 - Prep Arneson Data/PCA Elbow
Plot.png",res=600,units="in",height=4,width=6)
ElbowPlot(TBI_SO,ndims = 50)
dev.off()

# Cluster the data using KNN of PCs and Louvain modularity optimization
TBI_SO <- FindNeighbors(TBI_SO, dims = 1:co1) %>%
  FindClusters(resolution = 0.5)

# Data visualization with nonlinear dim reduction UMAP
TBI_SO <- RunUMAP(TBI_SO, dims = 1:co1)

png("Figures/1 - Prep Arneson Data/UMAP
Clusters.png",res=600,units="in",height=5,width=6)
DimPlot(TBI_SO, reduction = "umap",label=TRUE)
dev.off()

png("Figures/1 - Prep Arneson Data/UMAP by
MouseID.png",res=600,units="in",height=5,width=6)
DimPlot(TBI_SO, reduction = "umap",group.by = "MouseID")
dev.off()

# Identify monocyte clusters

png("Figures/1 - Prep Arneson Data/Monocyte Genes by Cluster CTSS
CD14.png",res=600,units="in",height=3.5,width=7)
VlnPlot(TBI_SO, features = c("CTSS", "CD14"))
```

```
dev.off()

png("Figures/1 - Prep Arneson Data/Monocyte Genes by Cluster AIF1
FCRLS.png",res=600,units="in",height=3.5,width=7)
VlnPlot(TBI_SO, features = c("AIF1","FCRLS"))
dev.off()

png("Figures/1 - Prep Arneson Data/Monocyte Genes by Cluster TMEM119
CCR5.png",res=600,units="in",height=3.5,width=7)
VlnPlot(TBI_SO, features = c("TMEM119","CCR5"))
dev.off()

png("Figures/1 - Prep Arneson Data/UMAP Monocyte Genes by
Cluster.png",res=600,units="in",height=8,width=6)
FeaturePlot(TBI_SO, features = c("CTSS","CD14",
"AIF1","FCRLS","TMEM119","CCR5"))
dev.off()

# Monocyte clusters are 4 and 14


# 2 - ANALYZE ARNESON MONOCYTES #############

# Filter out all cells not in clusters 4 or 14
TBI_M <- TBI_SO[,which(TBI_SO$seurat_clusters %in% c(4,14))]

# Verify that count vs feature is linear
png("Figures/2 - Analyze Arneson Monocytes/Features vs
Counts.png",res=600,units="in",height=5,width=6)
FeatureScatter(TBI_M, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
dev.off()

# Normalize data with log transform
TBI_M <- NormalizeData(TBI_M)
TBI_M <- FindVariableFeatures(TBI_M, selection.method = "vst", nfeatures =
2000)

# Identify the 10 most highly variable genes
top10 <- head(VariableFeatures(TBI_M), 10)

# Plot variable features
png("Figures/2 - Analyze Arneson Monocytes/Variable
Features.png",res=600,units="in",height=4,width=6)
VariableFeaturePlot(TBI_M)
LabelPoints(plot = VariableFeaturePlot(TBI_M), points = top10, repel = TRUE)
dev.off()

# Scale the data for PCA
all.genes <- rownames(TBI_M)
TBI_M <- ScaleData(TBI_M, features = all.genes)

# Conduct PCA
TBI_M <- RunPCA(TBI_M, features = VariableFeatures(object = TBI_M))

png("Figures/2 - Analyze Arneson Monocytes/PCA Loadings Charts PC1-
2.png",res=600,units="in",height=4.5,width=6)
VizDimLoadings(TBI_M, dims = 1:2, reduction = "pca")
```

```
dev.off()

png("Figures/2 - Analyze Arneson Monocytes/PCA Scores Plot PC1-
2.png",res=600,units="in",height=4.5,width=6)
DimPlot(TBI_M, reduction = "pca",group.by = "MouseID")
dev.off()

png("Figures/2 - Analyze Arneson Monocytes/PCA Heatmaps PC1-
12.png",res=600,units="in",height=11.5,width=7)
DimHeatmap(TBI_M, dims = 1:12, cells = 500, balanced = TRUE)
dev.off()

# # Determine appropriate dimension of the data by JackStraw analysis of PCs
# TBI_M <- JackStraw(TBI_M, num.replicate = 100, dims=30)
# TBI_M <- ScoreJackStraw(TBI_M, dims = 1:30)
#
# png("Figures/2 - Analyze Arneson Monocytes/Jack Straw
Plot.png",res=600,units="in",height=4,width=6)
# JackStrawPlot(TBI_M, dims = 1:30)
# dev.off()

# PC Percent Variance Analysis
pct <- TBI_M[["pca"]]@stdev / sum(TBI_M[["pca"]]@stdev) * 100 # Determine
percent of variation associated with each PC
cumu <- cumsum(pct)  # Calculate cumulative percents for each PC

# Determine which PC exhibits cumulative percent greater than 90% and %
variation associated with the PC as less than 5
co1 <- which(cumu > 90 & pct < 5)[1]

png("Figures/2 - Analyze Arneson Monocytes/PCA Cumulative
Variance.png",res=600,units="in",height=4,width=6)
plot(x=1:50,y=cumu,xlab="PC",ylab="Cumulative Variance (%)",pch=20)+
  abline(h=90,lty=2) +
  abline(v=co1,lty=2,col="red")
dev.off()

png("Figures/2 - Analyze Arneson Monocytes/PCA Elbow
Plot.png",res=600,units="in",height=4,width=6)
ElbowPlot(TBI_M,ndims = 50)
dev.off()

# Cluster the data using KNN of PCs and Louvain modularity optimization
TBI_M <- FindNeighbors(TBI_M, dims = 1:co1) %>%
  FindClusters(resolution = 0.5)

# Data visualization with nonlinear dim reduction UMAP
TBI_M <- RunUMAP(TBI_M, dims = 1:co1)

png("Figures/2 - Analyze Arneson Monocytes/UMAP
Clusters.png",res=600,units="in",height=5,width=6)
DimPlot(TBI_M, reduction = "umap",label=TRUE)
dev.off()

# Find markers for the clusters
TBI_M.markers <- FindAllMarkers(TBI_M, only.pos = TRUE, min.pct = 0.25,
logfc.threshold = 0.25)
```

```r
TBI_M.markers %>% group_by(cluster) %>% top_n(n = 2, wt = avg_log2FC)

png("Figures/2 - Analyze Arneson Monocytes/Monocyte Genes by Cluster
0.png",res=600,units="in",height=3.5,width=7)
VlnPlot(TBI_M, features =
TBI_M.markers$gene[which(TBI_M.markers$cluster==0)[1:3]])
dev.off()

png("Figures/2 - Analyze Arneson Monocytes/Monocyte Genes by Cluster
1.png",res=600,units="in",height=3.5,width=7)
VlnPlot(TBI_M, features =
TBI_M.markers$gene[which(TBI_M.markers$cluster==1)[1:3]])
dev.off()

png("Figures/2 - Analyze Arneson Monocytes/Monocyte Genes by Cluster
2.png",res=600,units="in",height=3.5,width=7)
VlnPlot(TBI_M, features =
TBI_M.markers$gene[which(TBI_M.markers$cluster==2)[1:3]])
dev.off()

png("Figures/2 - Analyze Arneson Monocytes/Monocyte Genes by Cluster
Custom.png",res=600,units="in",height=3.5,width=7)
VlnPlot(TBI_M, features = c("TMEM119","CXCL2","S100A6"))
dev.off()

png("Figures/2 - Analyze Arneson Monocytes/UMAP Monocyte Genes by
Cluster.png",res=600,units="in",height=6,width=6)
FeaturePlot(TBI_M, features = c("TMEM119","CXCL2","S100A6","CD14"))
dev.off()

# Assign labels to clusters
new.cluster.ids <- c("Microglia", "High MT Content Cells", "Macrophage")
names(new.cluster.ids) <- levels(TBI_M)
TBI_M <- RenameIdents(TBI_M, new.cluster.ids)

png("Figures/2 - Analyze Arneson Monocytes/UMAP Clusters
Labeled.png",res=600,units="in",height=5,width=6)
DimPlot(TBI_M, reduction = "umap", label = TRUE, pt.size = 0.5) + NoLegend()
dev.off()
```

Appendix 2: Sousa et. al. analysis in R using Seurat

These lines were run in R following the code from Appendix 1 to conduct analysis on the Sousa et. al.[13] data.

```
# Import the data
LPS <- fread("GEO Data Sets/GSM3182556_LPS.txt.gz")
CTRL <- fread("GEO Data Sets/GSM3182555_CTRL.txt.gz")

# Create count matrix
LPS_all = inner_join(LPS,CTRL,by="GENE")
LPS_counts = LPS_all[,2:ncol(LPS_all)] %>% as.matrix()
rownames(LPS_counts) = str_to_upper(LPS_all$GENE)
colnames(LPS_counts) = c(
  str_c(rep("LPS",ncol(LPS)-1),c(1:{ncol(LPS)-1})),
  str_c(rep("CTRL",ncol(CTRL)-1),c(1:{ncol(CTRL)-1})))

# Create data frame of metadata
meta_LPS <- data.frame(Condition = case_when(
  str_detect(colnames(LPS_counts),"LPS") ~ "LPS",
  str_detect(colnames(LPS_counts),"CTRL") ~ "CTRL"
))
rownames(meta_LPS) = colnames(LPS_counts)



# Create Seurat object for use in Seurat package codes
LPS_SO <- CreateSeuratObject(counts = LPS_counts, project = "LPS",meta.data =
meta_LPS) #, min.cells = 3, min.features = 200)

# Quality control: see distribution of number of features, number of counts
in all samples
png("Figures/3 - Prep Sousa Data/nFeature
Violins.png",res=600,units="in",height=5,width=7)
VlnPlot(LPS_SO, features ="nFeature_RNA",group.by = "Condition")
dev.off()
png("Figures/3 - Prep Sousa Data/nCount
Violins.png",res=600,units="in",height=5,width=7)
VlnPlot(LPS_SO, features = "nCount_RNA",group.by = "Condition")
dev.off()
png("Figures/3 - Prep Sousa Data/Features vs Counts Pre-
Filter.png",res=600,units="in",height=5,width=6)
FeatureScatter(LPS_SO, feature1 = "nCount_RNA", feature2 =
"nFeature_RNA",group.by = "Condition")
dev.off()

# Filter out cells with too few (possibly droplets) or too many counts
(possibly multiple cells)
LPS_SO <- subset(LPS_SO, subset = nFeature_RNA > 200 & nFeature_RNA < 4000)

png("Figures/3 - Prep Sousa Data/Features vs Counts Post-
Filter.png",res=600,units="in",height=5,width=6)
FeatureScatter(LPS_SO, feature1 = "nCount_RNA", feature2 =
"nFeature_RNA",group.by="Condition")
dev.off()
```

```
# Perform Integration of LPS and CTRL data
# split the dataset into a list of two seurat objects
LPS.list <- SplitObject(LPS_SO, split.by = "Condition")

# normalize and identify variable features for each dataset independently
LPS.list <- lapply(X = LPS.list, FUN = function(x) {
  x <- NormalizeData(x)
  x <- FindVariableFeatures(x, selection.method = "vst", nfeatures = 2000)
})

# select features that are repeatedly variable across datasets for
integration
features <- SelectIntegrationFeatures(object.list = LPS.list)

anchors <- FindIntegrationAnchors(object.list = LPS.list, anchor.features =
features)
# this command creates an 'integrated' data assay
LPS.combined <- IntegrateData(anchorset = anchors)
DefaultAssay(LPS.combined) <- "integrated"

# Find variable features
LPS.combined <- FindVariableFeatures(LPS.combined, selection.method = "vst",
nfeatures = 2000)

# Identify the 10 most highly variable genes
top10 <- head(VariableFeatures(LPS.combined), 10)

# plot variable features
png("Figures/3 - Prep Sousa Data/Variable
Features.png",res=600,units="in",height=4,width=6)
VariableFeaturePlot(LPS.combined)
LabelPoints(plot = VariableFeaturePlot(LPS.combined), points = top10, repel =
TRUE)
dev.off()

#Scale the data for PCA
all.genes <- rownames(LPS.combined)
LPS.combined <- ScaleData(LPS.combined, features = all.genes)

# Conduct PCA
LPS.combined <- RunPCA(LPS.combined, features = VariableFeatures(object =
LPS.combined))

png("Figures/3 - Prep Sousa Data/PCA Loadings Charts PC1-
2.png",res=600,units="in",height=4.5,width=6)
VizDimLoadings(LPS.combined, dims = 1:2, reduction = "pca")
dev.off()

png("Figures/3 - Prep Sousa Data/PCA Scores Plot PC1-
2.png",res=600,units="in",height=4.5,width=6)
DimPlot(LPS.combined, reduction = "pca",group.by = "Condition")
dev.off()

png("Figures/3 - Prep Sousa Data/PCA Heatmaps PC1-
12.png",res=600,units="in",height=11.5,width=7)
DimHeatmap(LPS.combined, dims = 1:12, cells = 500, balanced = TRUE)
dev.off()
```

```
# PC Percent Variance Analysis
pct <- LPS.combined[["pca"]]@stdev / sum(LPS.combined[["pca"]]@stdev) * 100
cumu <- cumsum(pct)  # Calculate cumulative percents for each PC

# Determine which PC exhibits cumulative percent greater than 90% and %
variation associated with the PC as less than 5
co1 <- which(cumu > 90 & pct < 5)[1]

png("Figures/3 - Prep Sousa Data/PCA Cumulative
Variance.png",res=600,units="in",height=4,width=6)
plot(x=1:50,y=cumu,xlab="PC",ylab="Cumulative Variance (%)",pch=20)+
  abline(h=90,lty=2) +
  abline(v=co1,lty=2,col="red")
dev.off()

png("Figures/3 - Prep Sousa Data/PCA Elbow
Plot.png",res=600,units="in",height=4,width=6)
ElbowPlot(LPS.combined,ndims = 50)
dev.off()

# Cluster the data using KNN of PCs and Louvain modularity optimization
LPS.combined <- FindNeighbors(LPS.combined, dims = 1:co1) %>%
  FindClusters(resolution = 0.5)

# Data visualization with nonlinear dim reduction UMAP
LPS.combined <- RunUMAP(LPS.combined, dims = 1:co1)

png("Figures/3 - Prep Sousa Data/UMAP
Clusters.png",res=600,units="in",height=5,width=6)
DimPlot(LPS.combined, reduction = "umap",label=TRUE)
dev.off()

png("Figures/3 - Prep Sousa Data/UMAP by
Condition.png",res=600,units="in",height=5,width=6)
DimPlot(LPS.combined, reduction = "umap",group.by = "Condition")
dev.off()

# Identifyclusters

# Find markers for the clusters
LPS.markers <- FindAllMarkers(LPS.combined, only.pos = TRUE, min.pct = 0.25,
logfc.threshold = 0.25)
LPS.markers %>% group_by(cluster) %>% top_n(n = 2, wt = avg_log2FC)

for(i in 0:{length(unique(LPS.combined$seurat_clusters))-1}){
  png(paste0("Figures/3 - Prep Sousa Data/Monocyte Genes by Cluster ",i,"
.png"),res=600,units="in",height=3.5,width=7)
  print({VlnPlot(LPS.combined, features =
LPS.markers$gene[which(LPS.markers$cluster==i)[1:3]])})
  dev.off()
}
```

Appendix 3: Data set integration in R using Seurat

These lines were run in R following the code from Appendix 1 and 2 to integrate both data sets using the anchor-based method provided in Seurat.

```
# Perform Integration of LPS and CTRL data
combined_SO <- merge(TBI_M, y = LPS.combined, add.cell.ids = c("Arnseon",
"Sousa"), project = "Integration")

# split the dataset into a list of two seurat objects
SO.list <- SplitObject(combined_SO, split.by = "Condition")

# normalize and identify variable features for each dataset independently
SO.list <- lapply(X = SO.list, FUN = function(x) {
  x <- NormalizeData(x)
  x <- FindVariableFeatures(x, selection.method = "vst", nfeatures = 2000)
})

# select features that are repeatedly variable across datasets for
integration
features <- SelectIntegrationFeatures(object.list = SO.list)
anchors <- FindIntegrationAnchors(object.list = SO.list, anchor.features =
features)

# this command creates an 'integrated' data assay
SO.combined <- IntegrateData(anchorset = anchors)
DefaultAssay(SO.combined) <- "integrated"

# Find variable features
SO.combined <- FindVariableFeatures(SO.combined, selection.method = "vst",
nfeatures = 2000)

# Identify the 10 most highly variable genes
top10 <- head(VariableFeatures(SO.combined), 10)

# plot variable features
png("Figures/4 - Integrate Data Sets/Variable
Features.png",res=600,units="in",height=4,width=6)
VariableFeaturePlot(SO.combined)
LabelPoints(plot = VariableFeaturePlot(SO.combined), points = top10, repel =
TRUE)
dev.off()

#Scale the data for PCA
all.genes <- rownames(SO.combined)
SO.combined <- ScaleData(SO.combined, features = all.genes)

# Conduct PCA
SO.combined <- RunPCA(SO.combined, features = VariableFeatures(object =
SO.combined))

png("Figures/4 - Integrate Data Sets/PCA Loadings Charts PC1-
2.png",res=600,units="in",height=4.5,width=6)
VizDimLoadings(SO.combined, dims = 1:2, reduction = "pca")
dev.off()
```

```
png("Figures/4 - Integrate Data Sets/PCA Scores Plot PC1-
2.png",res=600,units="in",height=4.5,width=6)
DimPlot(SO.combined, reduction = "pca",group.by = "Condition")
dev.off()

png("Figures/4 - Integrate Data Sets/PCA Heatmaps PC1-
12.png",res=600,units="in",height=11.5,width=7)
DimHeatmap(SO.combined, dims = 1:12, cells = 500, balanced = TRUE)
dev.off()

# # Determine appropriate dimension of the data by JackStraw analysis of PCs
# SO.combined <- JackStraw(SO.combined, num.replicate = 100, dims=30)
# SO.combined <- ScoreJackStraw(SO.combined, dims = 1:30)
#
# png("Figures/4 - Integrate Data Sets/Jack Straw
Plot.png",res=600,units="in",height=4,width=6)
# JackStrawPlot(SO.combined, dims = 1:30)
# dev.off()

# PC Percent Variance Analysis
pct <- SO.combined[["pca"]]@stdev / sum(SO.combined[["pca"]]@stdev) * 100 #
Determine percent of variation associated with each PC
cumu <- cumsum(pct)  # Calculate cumulative percents for each PC

# Determine which PC exhibits cumulative percent greater than 90% and %
variation associated with the PC as less than 5
co1 <- which(cumu > 90 & pct < 5)[1]

png("Figures/4 - Integrate Data Sets/PCA Cumulative
Variance.png",res=600,units="in",height=4,width=6)
plot(x=1:50,y=cumu,xlab="PC",ylab="Cumulative Variance (%)",pch=20)+
  abline(h=90,lty=2) +
  abline(v=co1,lty=2,col="red")
dev.off()

png("Figures/4 - Integrate Data Sets/PCA Elbow
Plot.png",res=600,units="in",height=4,width=6)
ElbowPlot(SO.combined,ndims = 50)
dev.off()

# Cluster the data using KNN of PCs and Louvain modularity optimization
SO.combined <- FindNeighbors(SO.combined, dims = 1:co1) %>%
  FindClusters(resolution = 0.5)

# Data visualization with nonlinear dim reduction UMAP
SO.combined <- RunUMAP(SO.combined, dims = 1:co1)

png("Figures/4 - Integrate Data Sets/UMAP
Clusters.png",res=600,units="in",height=5,width=6)
DimPlot(SO.combined, reduction = "umap",label=TRUE)
dev.off()

png("Figures/4 - Integrate Data Sets/UMAP by
Condition.png",res=600,units="in",height=5,width=6)
DimPlot(SO.combined, reduction = "umap",group.by = "Condition")
dev.off()
```

```
# Identify clusters
# For performing differential expression after integration, we switch back to
the original data
DefaultAssay(SO.combined) <- "RNA"

# Find markers for the clusters
SO.markers <- FindAllMarkers(SO.combined, only.pos = TRUE, min.pct = 0.25,
logfc.threshold = 0.25)
SO.markers %>% group_by(cluster) %>% top_n(n = 2, wt = avg_log2FC)

for(i in 0:{length(unique(SO.combined$seurat_clusters))-1}){
  png(paste0("Figures/4 - Integrate Data Sets/Monocyte Genes by Cluster ",i,"
.png"),res=600,units="in",height=3.5,width=7)
  print({VlnPlot(SO.combined, features =
SO.markers$gene[which(SO.markers$cluster==i)[1:3]])})
  dev.off()

  gene_export = as.matrix(SO.markers$gene[which(SO.markers$cluster==i)])
  export(gene_export,file=paste0("Figures/4 - Integrate Data Sets/Top Genes
Cluster ",i,".txt"),)
}

# png("Figures/4 - Integrate Data Sets/UMAP Monocyte Genes by
Cluster.png",res=600,units="in",height=8,width=6)
# FeaturePlot(SO.combined, features = c("CTSS","CD14",
"AIF1","FCRLS","TMEM119","CCR5"))
# dev.off()

png(paste0("Figures/4 - Integrate Data Sets/MG-Specific Genes by
Cluster.png"),res=600,units="in",height=3,width=7)
VlnPlot(SO.combined, features = c("TMEM119","CCL5"))
dev.off()

VlnPlot(SO.combined, features = c("TMEM119","CCL5"))
```

Appendix 4: Combining the Data Sets Using cFIT in R

The cFIT code[36,37] was run in R on a Lenovo T450s with Windows 10. Data sets were downloaded either directly from the journal website or from the Gene Expression Omnibus.

```
# install the package using devtools (install that too if needed)
library("devtools")
devtools::install_github("pengminshi/cFIT")

# load the package
library(cFIT)

# assemble your data sets into one list
data.list =
split_dataset_by_batch(X=t(as.matrix(SO.combined@assays$RNA@counts)),
                                   batch = SO.combined@meta.data$Condition,
                                   labels =
SO.combined@meta.data$seurat_clusters,
                                   metadata = SO.combined@meta.data,
                                   dataset.name = 'Integrated')

# select 2000 highly variable genes
genes = select_genes(data.list$X.list, ngenes=2000, verbose=F)

# data preprocessing
exprs.list = preprocess_for_integration(data.list$X.list, genes,
scale.factor=10^4, scale=F, center=F)

# run cFIT
int.out = CFITIntegrate(X.list=exprs.list, r=15, verbose=F, max.niter = 100,
seed=0,n.cores=1)

# save the results
saveRDS(int.out, file='cFIT Arneson Sousa.rds')

# Obtain the ncell-by-ngene expression matrix
exprs.int = do.call(rbind, int.out$H.list) %*% t(int.out$W)

# Obstain the ncell-by-r low dimensional representation
Hnorm = do.call(rbind, int.out$H.list) %*% diag(colSums(int.out$W))

# Visualize the data with UMAP

p1 = plot_umap(labels=SO.combined@meta.data$Condition, point.size = 0.5,
alpha=0.5, legend.name='Condition', emb=umap.out$emb)$p
p2 = plot_umap(labels=SO.combined@meta.data$seurat_clusters, point.size =
0.5, alpha=0.5, legend.name='Cluster', emb=umap.out$emb)$p
```

Bibliography

1. Masuda T, Sankowski R, Staszewski O, Prinz M. Microglia Heterogeneity in the Single-Cell Era. Cell Rep. 2020 Feb 4;30(5):1271–1281. PMID: 32023447

2. Lenz KM, Nelson LH. Microglia and Beyond: Innate Immune Cells As Regulators of Brain Development and Behavioral Function. Front Immunol [Internet]. Frontiers; 2018 [cited 2021 May 5];9. Available from: https://www.frontiersin.org/articles/10.3389/fimmu.2018.00698/full

3. Lapenna A, De Palma M, Lewis CE. Perivascular macrophages in health and disease. Nat Rev Immunol. Nature Publishing Group; 2018 Nov;18(11):689–702.

4. Mills CD, Kincaid K, Alt JM, Heilman MJ, Hill AM. M-1/M-2 macrophages and the Th1/Th2 paradigm. J Immunol Baltim Md 1950. 2000 Jun 15;164(12):6166–6173. PMID: 10843666

5. Tang Y, Le W. Differential Roles of M1 and M2 Microglia in Neurodegenerative Diseases. Mol Neurobiol. 2016 03;53(2):1181–1194. PMID: 25598354

6. Orihuela R, McPherson CA, Harry GJ. Microglial M1/M2 polarization and metabolic states. Br J Pharmacol. 2016 Feb;173(4):649–665. PMCID: PMC4742299

7. Mittelbronn M. The M1/M2 immune polarization concept in microglia: a fair transfer? Neuroimmunol Neuroinflammation. 2014 Jun 27;1:6–7.

8. Salvi V, Sozio F, Sozzani S, Del Prete A. Role of Atypical Chemokine Receptors in Microglial Activation and Polarization. Front Aging Neurosci [Internet]. Frontiers; 2017 [cited 2021 Apr 30];9. Available from: https://www.frontiersin.org/articles/10.3389/fnagi.2017.00148/full

9. Martinez FO, Gordon S. The M1 and M2 paradigm of macrophage activation: time for reassessment. F1000Prime Rep [Internet]. 2014 Mar 3 [cited 2021 May 5];6. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3944738/ PMCID: PMC3944738

10. Ransohoff RM. A polarizing question: do M1 and M2 microglia exist? Nat Neurosci. Nature Publishing Group; 2016 Aug;19(8):987–991.

11. Stratoulias V, Venero JL, Tremblay M-È, Joseph B. Microglial subtypes: diversity within the microglial community. EMBO J. John Wiley & Sons, Ltd; 2019 Sep 2;38(17):e101997.

12. Arneson D, Zhang G, Ying Z, Zhuang Y, Byun HR, Ahn IS, Gomez-Pinilla F, Yang X. Single cell molecular alterations reveal target cells and pathways of concussive brain injury. Nat Commun. Nature Publishing Group; 2018 Sep 25;9(1):3894.

13. Sousa C, Golebiewska A, Poovathingal SK, Kaoma T, Pires-Afonso Y, Martina S, Coowar D, Azuaje F, Skupin A, Balling R, Biber K, Niclou SP, Michelucci A. Single-cell transcriptomics reveals distinct inflammation-induced microglia signatures. EMBO Rep. 2018 Nov;19(11). PMCID: PMC6216255

14. Jolliffe IT. Principal Component Analysis [Internet]. 2nd ed. New York: Springer-Verlag; 2002 [cited 2021 May 5]. Available from: https://www.springer.com/gp/book/9780387954424

15. Liebermeister W. Linear modes of gene expression determined by independent component analysis. Bioinformatics. 2002 Jan 1;18(1):51–60.

16. Maaten L van der, Hinton G. Visualizing Data using t-SNE. J Mach Learn Res. 2008;9(86):2579–2605.

17. McInnes L, Healy J, Saul N, Großberger L. UMAP: Uniform Manifold Approximation and Projection. J Open Source Softw. 2018 Sep 2;3(29):861.

18. Hu Q, Greene CS. Parameter tuning is a key part of dimensionality reduction via deep variational autoencoders for single cell RNA transcriptomics. Pac Symp Biocomput Pac Symp Biocomput. 2019;24:362–373. PMCID: PMC6417816

19. Eraslan G, Simon LM, Mircea M, Mueller NS, Theis FJ. Single-cell RNA-seq denoising using a deep count autoencoder. Nat Commun. 2019 Jan 23;10(1):390. PMCID: PMC6344535

20. Pierson E, Yau C. ZIFA: Dimensionality reduction for zero-inflated single-cell gene expression analysis. Genome Biol. 2015 Nov 2;16(1):241.

21. Xiang R, Wang W, Yang L, Wang S, Xu C, Chen X. A Comparison for Dimensionality Reduction Methods of Single-Cell RNA-seq Data. Front Genet [Internet]. 2021 Mar 23 [cited 2021 May 5];12. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8021860/ PMCID: PMC8021860

22. Van Gassen S, Callebaut B, Van Helden MJ, Lambrecht BN, Demeester P, Dhaene T, Saeys Y. FlowSOM: Using self-organizing maps for visualization and interpretation of cytometry data. Cytom Part J Int Soc Anal Cytol. 2015 Jul;87(7):636–645. PMID: 25573116

23. Qiu X, Mao Q, Tang Y, Wang L, Chawla R, Pliner HA, Trapnell C. Reversed graph embedding resolves complex single-cell trajectories. Nat Methods. Nature Publishing Group; 2017 Oct;14(10):979–982.

24. Hartigan JA, Wong MA. Algorithm AS 136: A K-Means Clustering Algorithm. J R Stat Soc Ser C Appl Stat. [Wiley, Royal Statistical Society]; 1979;28(1):100–108.

25. Yang Y, Huh R, Culpepper HW, Lin Y, Love MI, Li Y. SAFE-clustering: Single-cell Aggregated (From Ensemble) Clustering for Single-cell RNA-seq Data. bioRxiv. Cold Spring Harbor Laboratory; 2018 Aug 6;215723.

26. Satija R, Farrell JA, Gennert D, Schier AF, Regev A. Spatial reconstruction of single-cell gene expression data. Nat Biotechnol. Nature Publishing Group; 2015 May;33(5):495–502.

27. Macosko EZ, Basu A, Satija R, Nemesh J, Shekhar K, Goldman M, Tirosh I, Bialas AR, Kamitaki N, Martersteck EM, Trombetta JJ, Weitz DA, Sanes JR, Shalek AK, Regev A, McCarroll SA. Highly Parallel Genome-wide Expression Profiling of Individual Cells Using Nanoliter Droplets. Cell. Elsevier; 2015 May 21;161(5):1202–1214. PMID: 26000488

28. Ji Z, Ji H. TSCAN: Pseudo-time reconstruction and evaluation in single-cell RNA-seq analysis. Nucleic Acids Res. 2016 Jul 27;44(13):e117. PMCID: PMC4994863

29. Wang J, Ma A, Chang Y, Gong J, Jiang Y, Qi R, Wang C, Fu H, Ma Q, Xu D. scGNN is a novel graph neural network framework for single-cell RNA-Seq analyses. Nat Commun. Nature Publishing Group; 2021 Mar 25;12(1):1882.

30. Li X, Wang K, Lyu Y, Pan H, Zhang J, Stambolian D, Susztak K, Reilly MP, Hu G, Li M. Deep learning enables accurate clustering with batch effect removal in single-cell RNA-seq analysis. Nat Commun. Nature Publishing Group; 2020 May 11;11(1):2338.

31. Peng J, Wang X, Shang X. Combining gene ontology with deep neural networks to enhance the clustering of single cell RNA-Seq data. BMC Bioinformatics. 2019 Jun 10;20(8):284.

32. Li J, Jiang W, Han H, Liu J, Liu B, Wang Y. ScGSLC: An unsupervised graph similarity learning framework for single-cell RNA-seq data clustering. Comput Biol Chem. 2021 Feb;90:107415. PMID: 33307360

33. Duò A, Robinson MD, Soneson C. A systematic performance evaluation of clustering methods for single-cell RNA-seq data. F1000Research [Internet]. 2020 Nov 16 [cited 2021 May 5];7. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6134335/ PMCID: PMC6134335

34. Stuart T, Butler A, Hoffman P, Hafemeister C, Papalexi E, Mauck WM, Hao Y, Stoeckius M, Smibert P, Satija R. Comprehensive Integration of Single-Cell Data. Cell. 2019 Jun 13;177(7):1888-1902.e21.

35. satijalab. satijalab/seurat [Internet]. 2021 [cited 2021 Apr 30]. Available from: https://github.com/satijalab/seurat

36. Peng M, Li Y, Wamsley B, Wei Y, Roeder K. Integration and transfer learning of single-cell transcriptomes via cFIT. Proc Natl Acad Sci [Internet]. National Academy of Sciences; 2021 Mar 9 [cited 2021 Apr 30];118(10). Available from: https://www.pnas.org/content/118/10/e2024383118 PMID: 33658382

37. PENG M. pengminshi/cFIT [Internet]. 2021 [cited 2021 Apr 30]. Available from: https://github.com/pengminshi/cFIT

38. Luecken MD, Theis FJ. Current best practices in single-cell RNA-seq analysis: a tutorial. Mol Syst Biol. John Wiley & Sons, Ltd; 2019 Jun 1;15(6):e8746.

39. Introduction to scRNA-seq integration [Internet]. [cited 2021 May 5]. Available from: https://satijalab.org/seurat/articles/integration_introduction.html

40. Seurat - Guided Clustering Tutorial [Internet]. [cited 2021 May 5]. Available from: https://satijalab.org/seurat/articles/pbmc3k_tutorial.html

41. Zhu X, Zhang J, Xu Y, Wang J, Peng X, Li H-D. Single-Cell Clustering Based on Shared Nearest Neighbor and Graph Partitioning. Interdiscip Sci Comput Life Sci. 2020 Jun 1;12(2):117–130.

42. Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. J Stat Mech Theory Exp. IOP Publishing; 2008 Oct;2008(10):P10008.

43. Mishra A. Demystifying Louvain’s Algorithm and Its implementation in GPU [Internet]. Medium. 2019 [cited 2021 May 5]. Available from: https://medium.com/walmartglobaltech/demystifying-louvains-algorithm-and-its-implementation-in-gpu-9a07cdd3b010

44. cFIT Tutorial [Internet]. [cited 2021 May 5]. Available from: https://htmlpreview.github.io/?https://github.com/pengminshi/cFIT/blob/master/vignettes/vignette.html