Imbalanced integration guidelines tutorial - R

2023-04-24

This tutorial is based on the Imbalanced integration guidelines in the Iniquitate manuscript. This example explores integration in the case of a dataset comprising of two batches of PBMC data, for which a modified version was used in the paper experiments - namely dataset number 5 from the Tran et al. scRNA-seq integration benchmarking paper. To simulate a case where significant cell-type imbalance exists between the batches in this dataset, we first dropped non-prevalent populations from both batches (<200 cells per batch, which included megakaryocytes, Plasmacytoid dendritic cells, and hematopoetic stem cells), and then removed CD4+ T cells and CD14+ Monocytes from batch 1. As demonstrated in the paper experiments, cell-type imbalance combined with proximity to similar cell-types can lead to loss of information about the imbalanced cell-types in the integrated space. Given that CD4+ T cells are in transcriptional proximity to CD8+ T cells and NK cells, and CD14+ Monocytes are in proximity to CD16+ Monocytes, and combined with the fact that they are imbalanced in this data, we would expect that integration methods might lead to loss of biological signal.

These facts are known to us because we know the cell-type labels, but we'll play the role of a researcher that has obtained these two batches in an unsupervised manner and demonstrate how the guidelines can be used.

We'll begin by examining the properties of these two batches.

Dataset exploration

Load the necessary libraries for the tutorial:

```
if (!requireNamespace("remotes", quietly = TRUE)) {
    install.packages("remotes")
}
remotes::install_github("mojaveazure/seurat-disk")
library(SeuratDisk)
library(googledrive)
library(ggplot2)
library(ggthemes)
library(Seurat)
library(bluster)
library(pheatmap)
library(reticulate)
```

Start by downloading the data - both batches:

```
# Create data directory
if (!dir.exists("data")) {
    dir.create("data")
}

# Install files if not already present
if (!(file.exists("data/tran_et_al_batch_1.h5ad"))) {
    # Remove authorization for Google Drive
    drive_deauth()
```

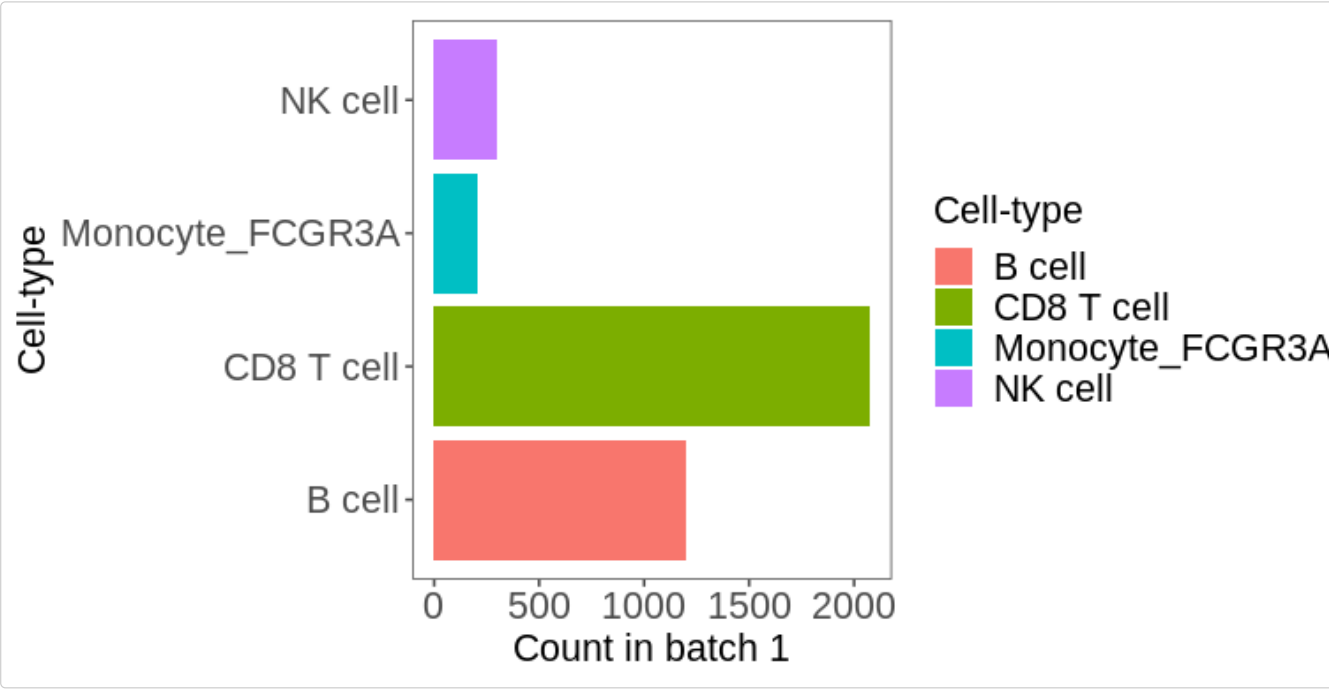
```
    drive_user()

    # Download the files
    file_id_1 <- as_id("1LryRxnVGq2Ny4awDxyiS84oou_zkvQki")
    file_id_2 <- as_id("1JMhR3cGGj4pueSLiFCPYl19jZDUKmznR")
    file_1 <- drive_get(file_id_1)
    file_2 <- drive_get(file_id_2)
    googledrive::drive_download(file_1, path = "data/tran_et_al_batch_1.h5ad",
        overwrite = TRUE)
    googledrive::drive_download(file_2, path = "data/tran_et_al_batch_2.h5ad",
        overwrite = TRUE)
}
```

Read in both files using the SeuratDisk library (we'll need to install it first but this shouldn't take long). We'll use the `Seurat` library for processing and analyzing the data:

```
# Convert the tran et al files to h5seurat format
Convert("data/tran_et_al_batch_1.h5ad", dest = "h5seurat",
    overwrite = TRUE)
Convert("data/tran_et_al_batch_2.h5ad", dest = "h5seurat",
    overwrite = TRUE)

# Load h5ad files for both batches
pbmc_1 <- SeuratDisk::LoadH5Seurat("data/tran_et_al_batch_1.h5seurat")
pbmc_2 <- SeuratDisk::LoadH5Seurat("data/tran_et_al_batch_2.h5seurat")
```

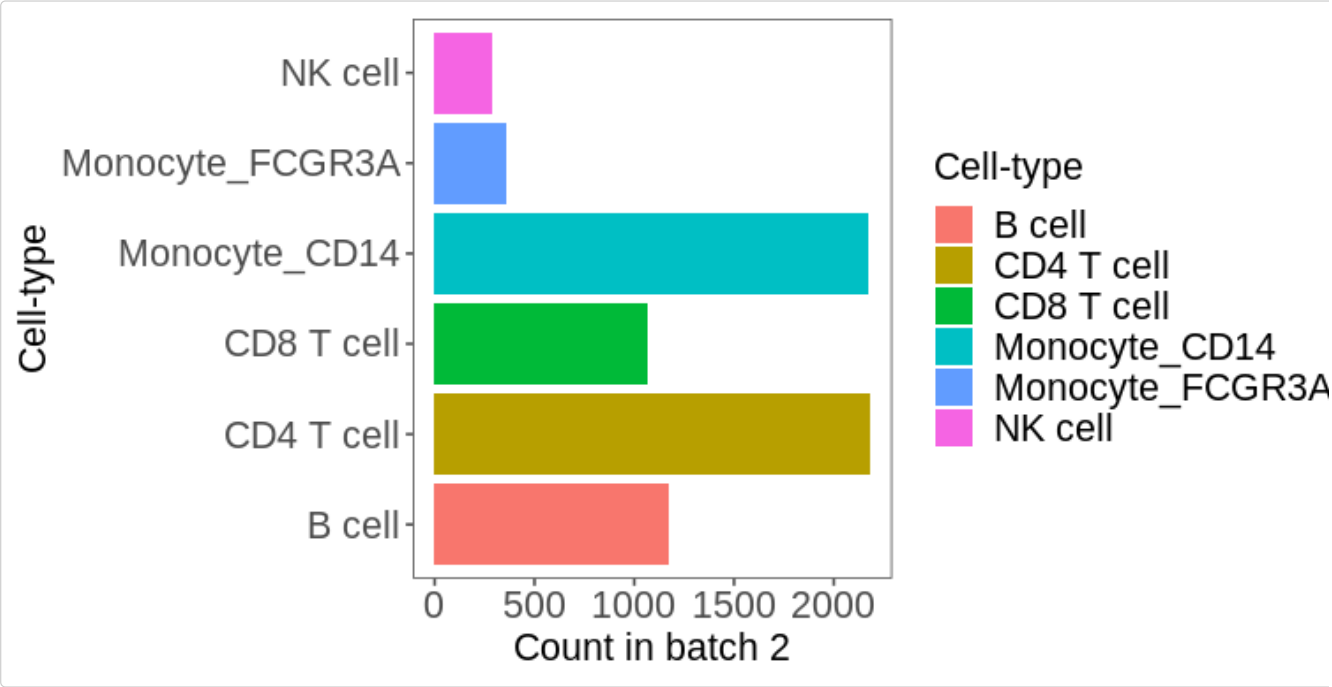
Examine the cell-type distribution for each batch:

```
#>
#>          B cell      CD8 T cell Monocyte_FCGR3A         NK cell
#>            1199            2076             206             303
```




```
#>
#>          B cell      CD4 T cell      CD8 T cell   Monocyte_CD14 Monocyte_FCGR3A
```

```
#>           1172           2183           1066           2176            355
#>        NK cell
#>            290
```



As we can see from the results, there is an obvious **imbalance in terms of the cell-types present in both batches**, and that the **CD4+ T cells** and **CD14+ Monocytes** only exist in batch 2.

Now that we've seen the imbalance present in this dataset, we're going to take the role of a single-cell analysis practitioner that **does not know the underlying cell-type distribution of this dataset**. I.e. we don't know what cell-types are present and our goal is to learn a joint representation of these two batches of pbmc cells that we've obtained from an experiment.

We'll start from the top of the guidelines and work our way down.

Pre-integration stage

The first stage of the guidelines focuses on checks before integration, to determine whether or not an imbalance might be present in the dataset. This serves as a prior for our integration experiment, and combined with the post-integration diagnostics, helps inform whether or not we need to tune the tradeoff between *preserving biological heterogeneity* and *batch mixing*.

Let's make our way down the flowchart.

Unsupervised clustering within each batch

The first step is to perform unsupervised clustering within each batch. We don't have any labels in this scenario, so we perform unsupervised clustering to gauge if different cell-types might be present based on the number and proportion of clusters between batches.

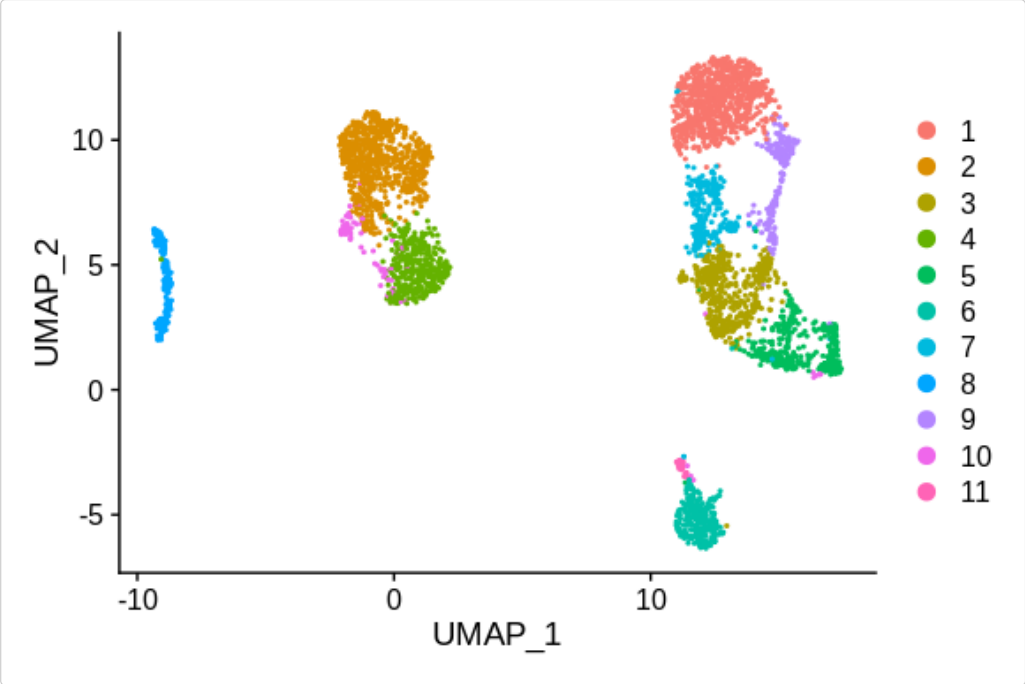
Using `Seurat`, start by performing normalization, highly-variable gene selection, PCA-reduction, and unsupervised clustering of each batch separately. Then we can visualize the results using UMAP:

```
# Preprocess the data using Seurat - Normalize
# and scale
pbmc_1 <- NormalizeData(pbmc_1, normalization.method = "LogNormalize",
    scale.factor = 10000, verbose = FALSE)
```

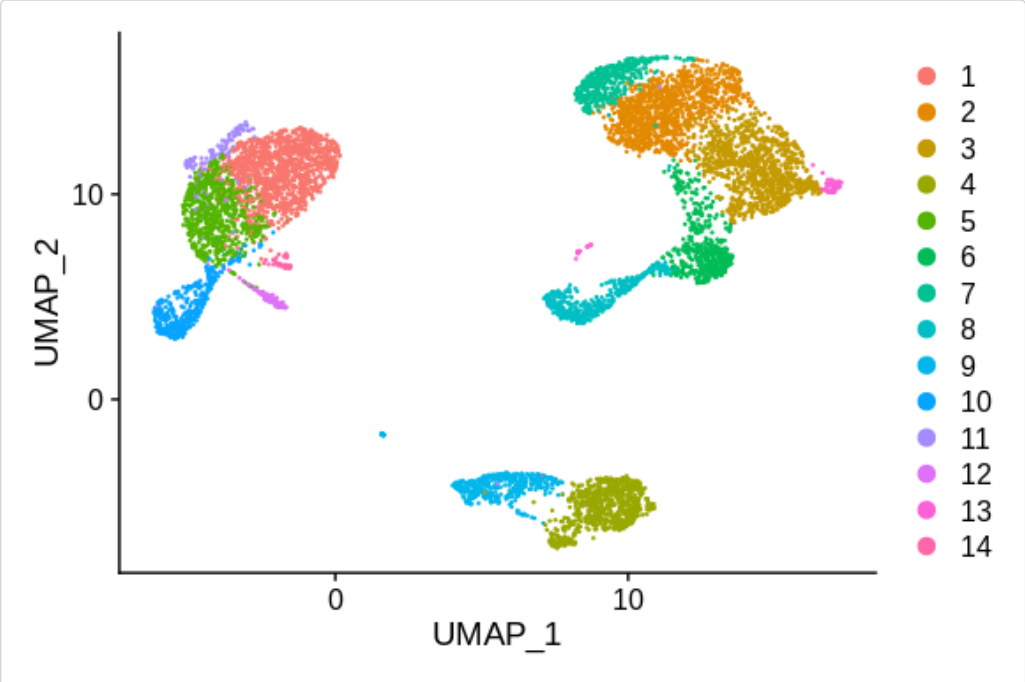
```
pbmc_1 <- FindVariableFeatures(pbmc_1, nfeatures = 2000,
    verbose = FALSE)
all.genes <- rownames(pbmc_1)
pbmc_1 <- ScaleData(pbmc_1, features = all.genes, verbose = FALSE)

# Run PCA, create neighborhood graph and get the
# clusters using the Leiden algorithm
pbmc_1 <- RunPCA(pbmc_1, features = VariableFeatures(pbmc_1),
    verbose = FALSE)
pbmc_1 <- FindNeighbors(pbmc_1, dims = 1:10, verbose = FALSE)
pbmc_1 <- FindClusters(pbmc_1, resolution = 0.5, algorithm = "leiden",
    verbose = FALSE)

# Get the UMAP reduction of the data
pbmc_1 <- RunUMAP(pbmc_1, umap.method = "umap-learn",
    dims = 1:10, verbose = FALSE)

# Do the same for pbmc 2
pbmc_2 <- NormalizeData(pbmc_2, normalization.method = "LogNormalize",
    scale.factor = 10000, verbose = FALSE)
pbmc_2 <- FindVariableFeatures(pbmc_2, nfeatures = 2000,
    verbose = FALSE)
all.genes <- row.names(pbmc_2)
pbmc_2 <- ScaleData(pbmc_2, features = all.genes, verbose = FALSE)
pbmc_2 <- RunPCA(pbmc_2, features = VariableFeatures(pbmc_2),
    verbose = FALSE)
pbmc_2 <- FindNeighbors(pbmc_2, dims = 1:10, verbose = FALSE)
pbmc_2 <- FindClusters(pbmc_2, resolution = 0.5, algorithm = "leiden",
    verbose = FALSE)
pbmc_2 <- RunUMAP(pbmc_2, umap.method = "umap-learn",
    dims = 1:10, verbose = FALSE)
```

Plot the UMAP reduction of the clustering results for each individual batch:

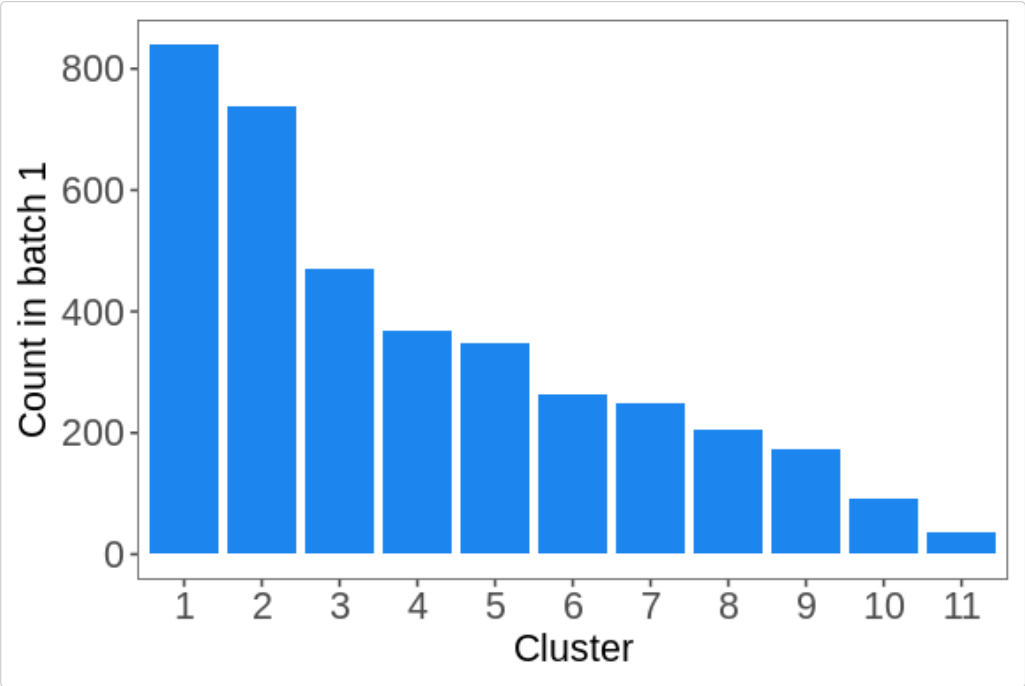
```
DimPlot(pbmc_1, reduction = "umap")
```

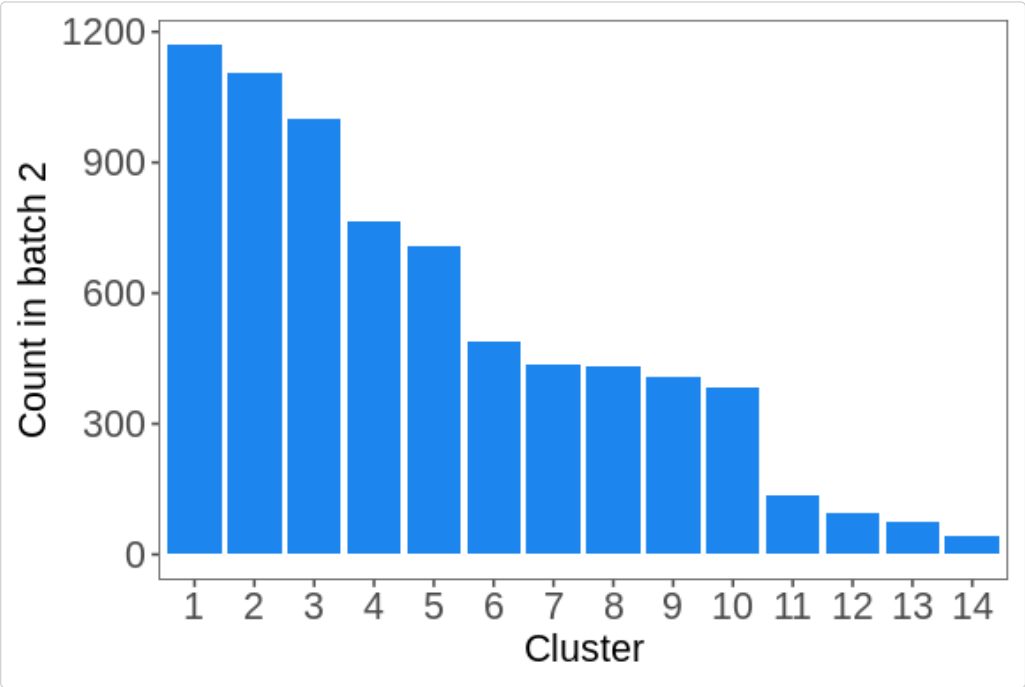



```
DimPlot(pbmc_2, reduction = "umap")
```



As we can see from the plots, there are differences in the number of clusters and possible proportion differences between the batches. We can examine this a bit closer using barplots:





From the barplots, we can see that the number of cells across clusters follows a similar distribution, but the fact that batch 2 contains more clusters at the same Leiden resolution signals that there may be an imbalance present between the batches of this dataset. We'll keep this in mind as we continue.

Continuing down the flowchart for the **pre-integration stage**, we check if we have a reference dataset available. Let's assume we don't for brevity, but if we did, we would annotate each batch individually, and then check the same barplots as above, but with cell-type labels instead of clusters. We'd likely arrive at a similar conclusion.

Given that we found that there may be composition differences, it would be prudent to think about how to tune the tradeoff between *biological heterogeneity preservation* and *batch mixing*. Examining the landmark benchmarking paper from Leuken et al. is another aspect of the guidelines. The major figures to examine would be Figure 3 (for RNA-based integration tasks) and Figure 5. From the paper we know that scGen and scANVI perform best when conserving biological heterogeneity, but their implementations in the paper required cell-type labels, which we do not have in this hypothetical analysis. A dataset of two pbmc batches is not a highly complex scenario, and in relatively simpler scenarios with batch effects, Seuratv3 was found

to integrate strong technical effects (batch mixing), while Harmony was found to better performer in conserving biological signal.

We do not know a-priori how strong the technical effects are between our pbmc batches - we only know that there might be composition differences. We'll use our post-integration diagnostics to assess how well the results meet our criteria for preserving biological heterogeneity and batch mixing. Our **optimal** setup will involve adequate batch correction (not necessarily the strongest) without sacrificing biological signal. As such, we will begin by testing both Seuratv3 canonical correlation analysis (CCA) and Harmony for integration.

Integration stage

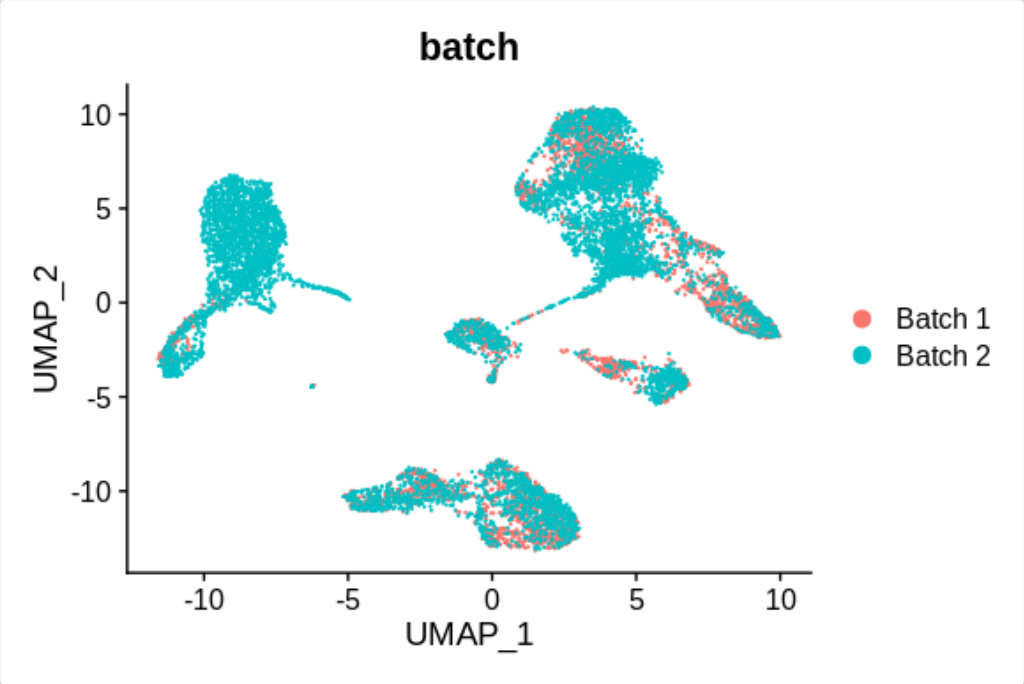
let's start by integrating using Seurat CCA first:

```
# Include a batch covariate for integration in
# the metadata
pbmc_1@meta.data$batch <- "Batch 1"
pbmc_2@meta.data$batch <- "Batch 2"

# Obtain anchors between the two batches and
# perform CCA-base integration using Seurat (the
# default)
pbmc_list <- c(pbmc_1, pbmc_2)
features <- SelectIntegrationFeatures(object.list = pbmc_list,
    verbose = FALSE)
anchors <- FindIntegrationAnchors(object.list = pbmc_list,
    anchor.features = features, verbose = FALSE)
pbmc_integrated <- IntegrateData(anchorset = anchors,
    verbose = FALSE)

# Using the integrated representation, obtain a
# reduced representation through PCA, cluster the
# joint representation and obtain UMAP
# coordinates
pbmc_integrated <- ScaleData(pbmc_integrated, verbose = FALSE)
pbmc_integrated <- RunPCA(pbmc_integrated, verbose = FALSE)
pbmc_integrated <- FindNeighbors(pbmc_integrated, reduction = "pca",
    dims = 1:15, verbose = FALSE)
pbmc_integrated <- FindClusters(pbmc_integrated, resolution = 0.5,
    algorithm = "leiden", verbose = FALSE)
pbmc_integrated <- RunUMAP(pbmc_integrated, reduction = "pca",
    dims = 1:15, verbose = FALSE)
```

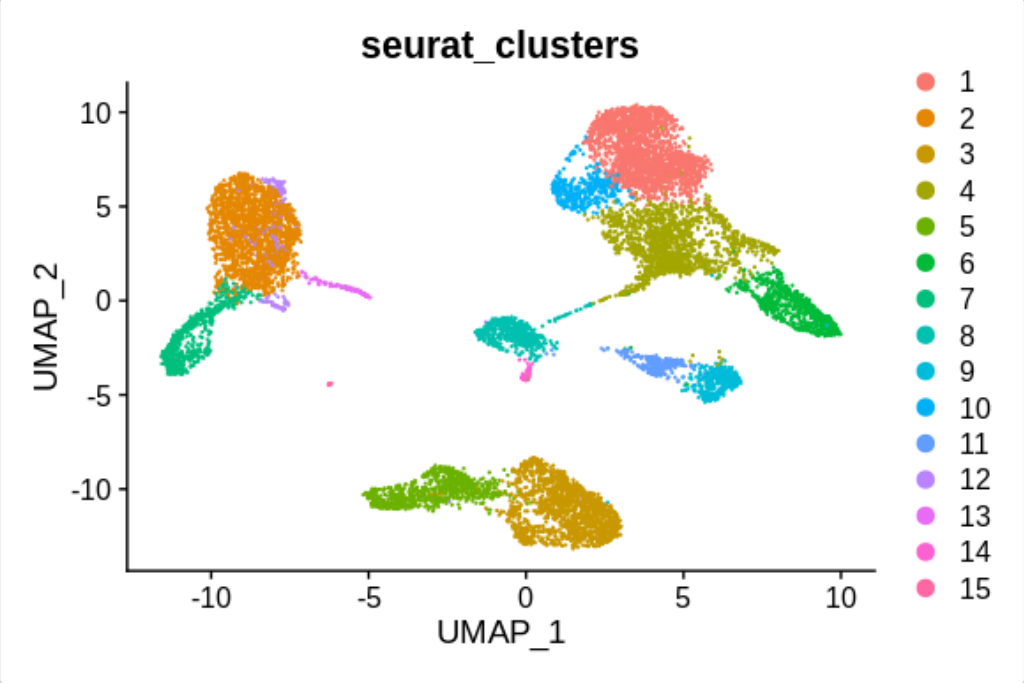
Post-integration stage

Let's examine the Seuratv3 CCA integration results by UMAP:

```
# Plot the UMAP reduction colored by batch and
# the seurat clusters
DimPlot(pbmc_integrated, reduction = "umap", group.by = "batch")
```




```
DimPlot(pbmc_integrated, reduction = "umap", group.by = "seurat_clusters")
```



Measure degree of batch mixing post-integration

From the UMAP plots we can clearly see that there is a strong degree of batch mixing. However, we don't know whether or not this have removed biological signal yet. We can first simply quantify the batch mixing by comparing the cluster labels with the batch labels using the `Adjusted Rand Index` (ARI). We'll use 1 - ARI, as a high score should indicate that the cluster labels and batch labels do not correspond - indicating strong batch mixing.

```
# Using the bluster library, calculate the
# 1-batch ARI
```

```
batch_labels <- pbmc_integrated@meta.data$batch
integrated_cluster_labels <- pbmc_integrated@meta.data$seurat_clusters
batch_ari <- 1 - pairwiseRand(batch_labels, integrated_cluster_labels,
    mode = "index", adjusted = TRUE)
print(paste0("Batch mixing ARI: ", batch_ari))
#> [1] "Batch mixing ARI: 0.952262112242732"
```

A score near 1 would indicate perfect batch mixing, and thus using Seuratv3 CCA, the results show that the batch mixing is very strong.

Compare pre and post-integration clusters

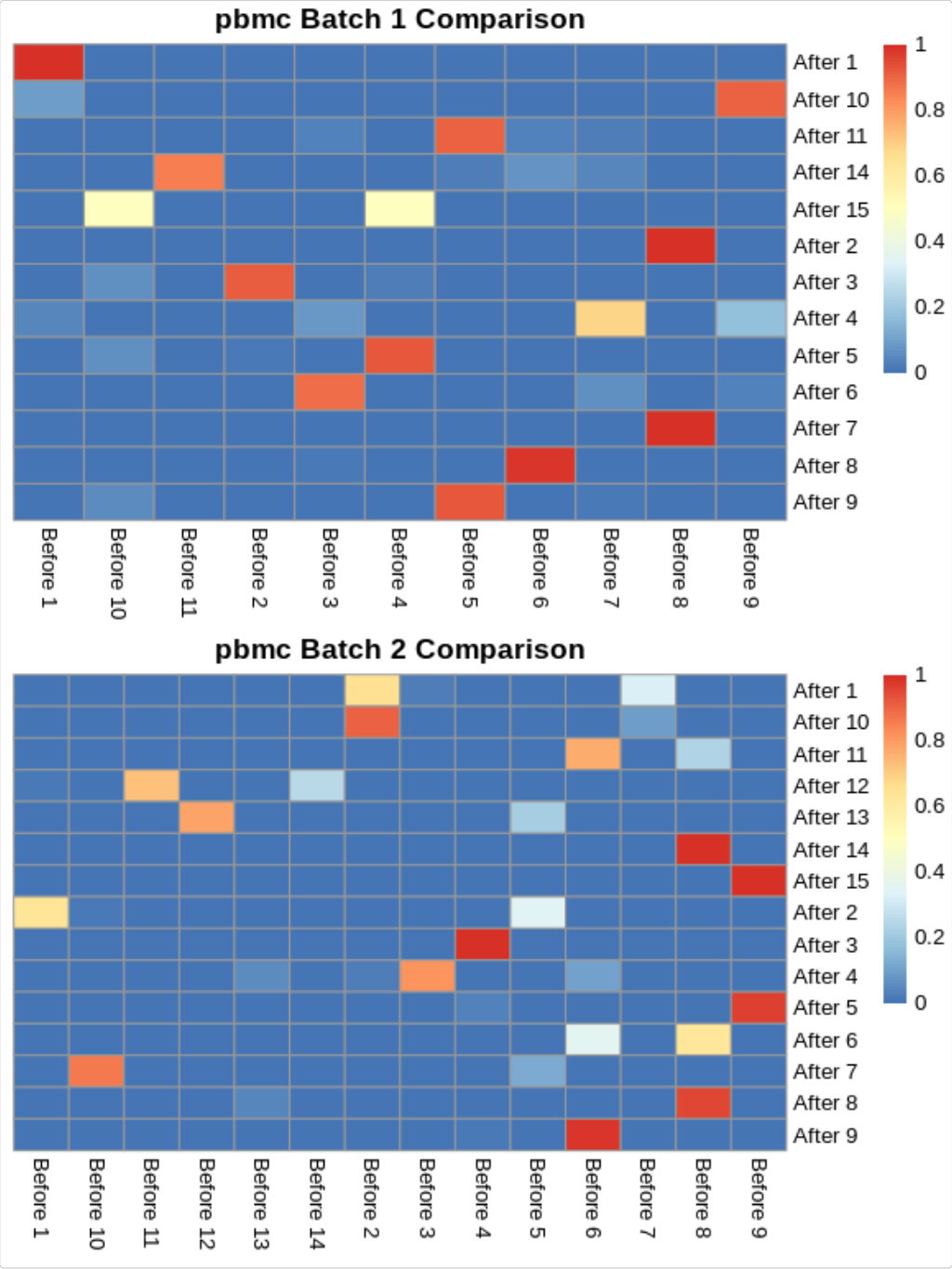
Despite the strong batch mixing results, we still don't have an idea about how well the integrated results might be preserving biological heterogeneity. For this we'll use a workflow outlined in the Orchestrating Single-Cell Analysis book.

Essentially, as we don't have access to cell-type labels, we're going to use the degree of concordance of unsupervised clusters within each batch (before integration) and unsupervised clusters after integration. We can assess this overlap based on heatmaps, as well as directly quantify it for each batch using the ARI between unsupervised clusters before integration and after integration. This readout is used as a surrogate for preservation of cell-type heterogeneity, because if cells from one pre-integration unsupervised cluster are being broken up or merged with cells from other pre-integration unsupervised clusters, this indicates a potential loss of biological signal. In general, we do expect some degree of information loss because of the assumptions and limitations of unsupervised clustering compared to expert annotated cell-type labels, but this technique works well as a post-integration diagnostic.

let's compare the pre and post-integration unsupervised clustering results:

```
# Get the cluster labels for each batch before
# integration
cluster_labels_b1_before <- pbmc_1@meta.data$seurat_clusters
cluster_labels_b2_before <- pbmc_2@meta.data$seurat_clusters

# Subset the post-integration cluster labels for
# only those that correspond to each batch
cluster_labels_b1_after <-
        pbmc_integrated@meta.data$seurat_clusters[pbmc_integrated@meta.data$batch ==
    "Batch 1"]
cluster_labels_b2_after <-
        pbmc_integrated@meta.data$seurat_clusters[pbmc_integrated@meta.data$batch ==
    "Batch 2"]

# Create heatmaps for correspondence of
# unsupervised clusters before and after
# integration
tab_b1 <- nestedClusters(ref = paste("Before", cluster_labels_b1_before),
    alt = paste("After", cluster_labels_b1_after))
tab_b2 <- nestedClusters(ref = paste("Before", cluster_labels_b2_before),
    alt = paste("After", cluster_labels_b2_after))

heat_b1 <- pheatmap(tab_b1$proportions, cluster_row = FALSE,
    cluster_col = FALSE, main = "pbmc Batch 1 Comparison",
    silent = TRUE)
heat_b2 <- pheatmap(tab_b2$proportions, cluster_row = FALSE,
    cluster_col = FALSE, main = "pbmc Batch 2 Comparison",
    silent = TRUE)
```

```
gridExtra::grid.arrange(heat_b1[[4]], heat_b2[[4]])
```



The rows in these heatmaps sum to 1, and the key aspect to examine is whether or not each row (each unsupervised cluster post-integration) corresponds to more than one pre-integration cluster (columns). If this is the case, that means that cell-types may have been merged together after integration, leading to loss of biological information.

Further, if columns are split across rows, this may indicate that cell-types are being split across clusters after integration.

Examining the comparison for **batch 1**, we can see that every post-integrated cluster except for *4* correspond to their own pre-integration batches. Further, only a few pre-integration clusters are split across post-integration clusters. Therefore, the biological heterogeneity from batch 1 is likely being well-conserved after integration.

However, when examining **batch 2**, we can see that there are clearly many post-integration clusters that contain data from different pre-integration batch 2 clusters, including post-integration cluster 1, 10, 11, 12,

13, 2, 4, 6, 7, and 8. This means that certain cell-types from batch 2 might be getting mixed with others post-integration. There are also many more unsupervised clusters pre-integration (columns) being split up across post-integration clusters.

We'll now check the results numerically by calculating the ARI for pre and post-integration clusters of cells from each batch. Given the plots, we expect that the value for batch 2 will be lower.

```
# Calculate and print ARI values for each batch
# and the clusters the cells belong to pre and
# post integration
b1_ari <- pairwiseRand(cluster_labels_b1_before, cluster_labels_b1_after,
    mode = "index", adjusted = TRUE)
b2_ari <- pairwiseRand(cluster_labels_b2_before, cluster_labels_b2_after,
    mode = "index", adjusted = TRUE)
print(paste0("Batch 1 biological heterogeneity conservation ARI: ",
    b1_ari))
#> [1] "Batch 1 biological heterogeneity conservation ARI: 0.86625019801584"
print(paste0("Batch 2 biological heterogeneity conservation ARI: ",
    b2_ari))
#> [1] "Batch 2 biological heterogeneity conservation ARI: 0.663233742844355"
```

This seems to be exactly the case. The ARI for batch 1 is quite high, while the ARI for batch 2 is substantially lower. Given our ideal tradeoff between batch correction and biological heterogeneity conservation, the results for batch 1 are quite adequete. However, we do see that batch 2 loses quite a bit of signal based on the almost 25% decrease in ARI value.

Given that this tradeoff is not adequete, we will not proceed yet, and instead look to **tune the heterogeneity preservation and batch mixing tradeoff**, and **increase biological heterogeneity preservation**.

The canonical correlation analysis (CCA) variant of Seurat enforces stronger integration and batch-mixing, as CCA enforces a representation space where data between the two batches is correlated, and this may lead to loss of biological information if unshared variation is present due to overcorrection. Thus, instead of using CCA, we'll use the reciprocal PCA (rPCA) approach. In this setting, a PCA reduction is trained on one of the two batches, and then the same reduction is applied to the second batch. Because the PCA space does not enforce correlation between the batches, heterogeneity specific to the second batch is more likely to be preserved when projected onto the PCA space of the first.

Integration stage

Method tuning

Let's go ahead and re-run Seurat using rPCA instead:

```
# Get the integration features
features <- SelectIntegrationFeatures(object.list = pbmc_list,
    verbose = FALSE)

# Get distinct PCA representations for both
# batches
pbmc_list <- lapply(pbmc_list, function(x) {
    x <- ScaleData(x, features = features, verbose = FALSE)
    x <- RunPCA(x, features = features, verbose = FALSE)
})

# Find integration anchors using the rPCA method
```

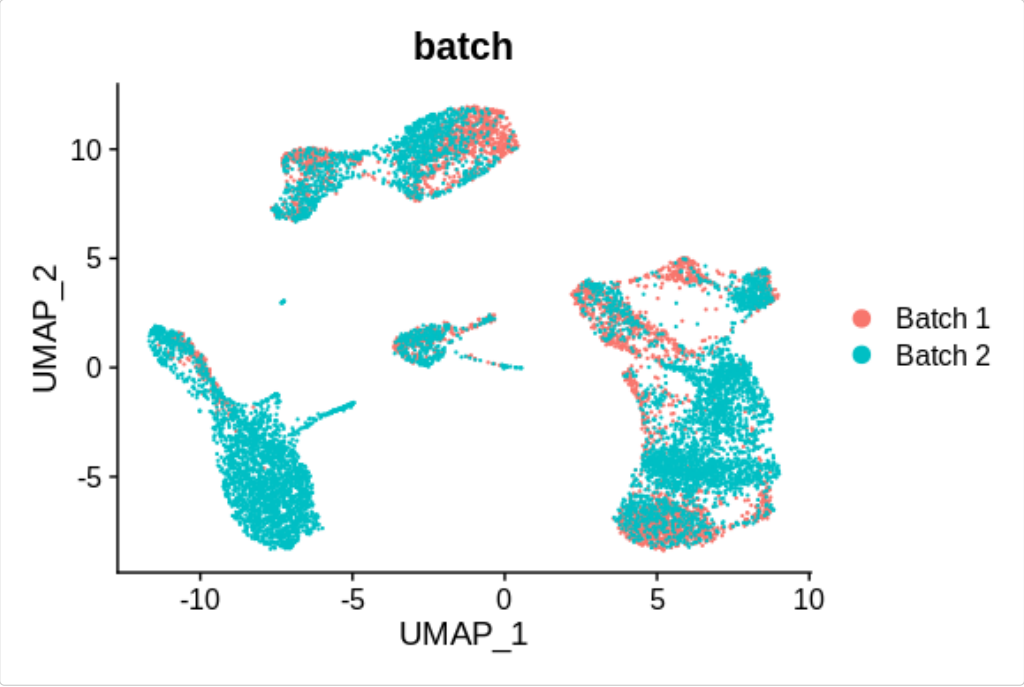
```
anchors <- FindIntegrationAnchors(object.list = pbmc_list,
    anchor.features = features, reduction = "rpca",
    verbose = FALSE)

# Integrate the data
pbmc_integrated <- IntegrateData(anchorset = anchors,
    verbose = FALSE)

# Using the integrated data, perform
# dimensionality reduction, clustering and UMAP
DefaultAssay(pbmc_integrated) <- "integrated"
pbmc_integrated <- ScaleData(pbmc_integrated, verbose = FALSE)
pbmc_integrated <- RunPCA(pbmc_integrated, npcs = 15,
    verbose = FALSE)
pbmc_integrated <- FindNeighbors(pbmc_integrated, reduction = "pca",
    dims = 1:15, verbose = FALSE)
pbmc_integrated <- FindClusters(pbmc_integrated, resolution = 0.5,
    algorithm = "leiden", verbose = FALSE)
pbmc_integrated <- RunUMAP(pbmc_integrated, reduction = "pca",
    dims = 1:15, verbose = FALSE)
```

Post-integration stage

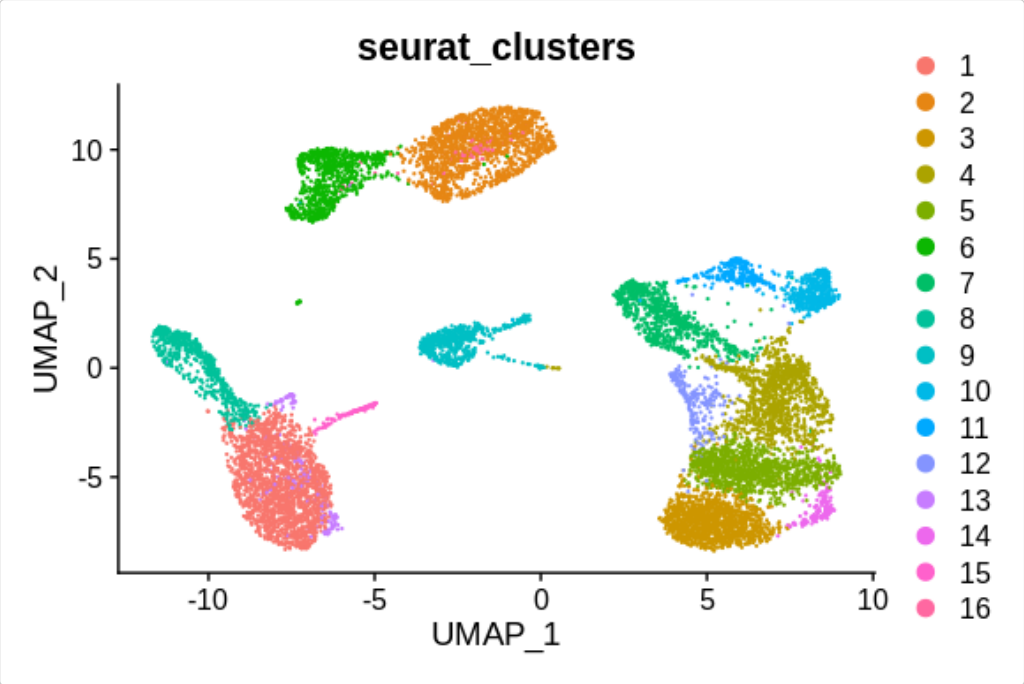
Measure degree of batch mixing post-integration

let's go ahead and see what the results look like using the UMAP reduction:

```
DimPlot(pbmc_integrated, reduction = "umap", group.by = "batch")
```




```
DimPlot(pbmc_integrated, reduction = "umap", group.by = "seurat_clusters")
```



We can already see that the batch mixing might not be as strong here compared to the CCA approach. let's quantify this:

```
batch_labels <- pbmc_integrated@meta.data$batch
integrated_cluster_labels <- pbmc_integrated@meta.data$seurat_clusters
batch_ari <- 1 - pairwiseRand(batch_labels, integrated_cluster_labels,
    mode = "index", adjusted = TRUE)
print(paste0("Batch mixing ARI: ", batch_ari))
#> [1] "Batch mixing ARI: 0.930467436410688"
```

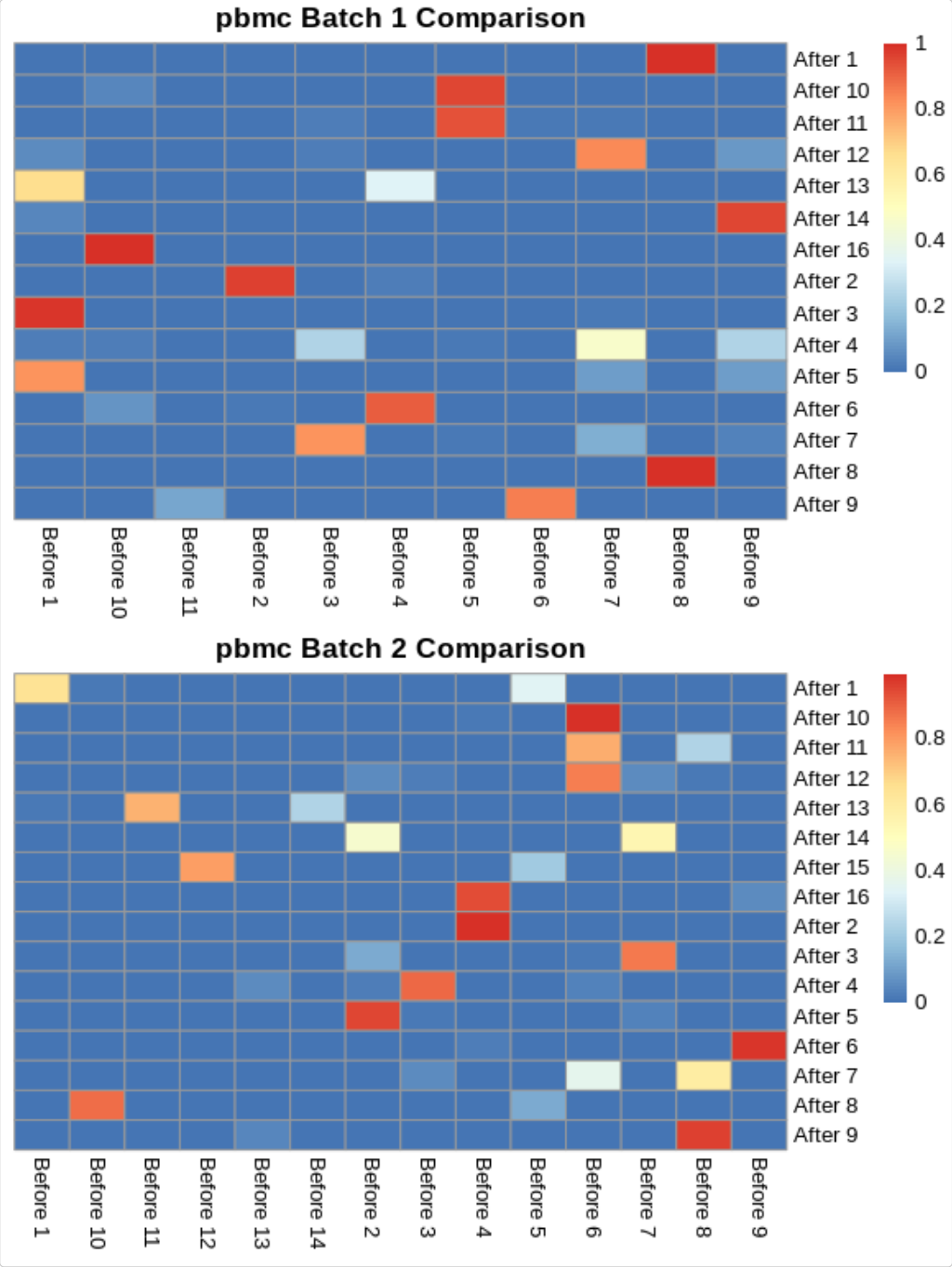
The batch mixing ARI is definitely less in this case compared to using CCA (0.93 vs 0.95), but this drop is not significant enough for us to reconsider testing this approach.

Compare pre and post-integration clusters

Now we can once again check the pre and post integration clustering concordance to get a sense of how well this approach preserved biological heterogeneity:

```
cluster_labels_b1_before <- pbmc_1@meta.data$seurat_clusters
cluster_labels_b2_before <- pbmc_2@meta.data$seurat_clusters

cluster_labels_b1_after <-
        pbmc_integrated@meta.data$seurat_clusters[pbmc_integrated@meta.data$batch ==
    "Batch 1"]
cluster_labels_b2_after <-
        pbmc_integrated@meta.data$seurat_clusters[pbmc_integrated@meta.data$batch ==
    "Batch 2"]

tab_b1 <- nestedClusters(ref = paste("Before", cluster_labels_b1_before),
    alt = paste("After", cluster_labels_b1_after))
tab_b2 <- nestedClusters(ref = paste("Before", cluster_labels_b2_before),
    alt = paste("After", cluster_labels_b2_after))

heat_b1 <- pheatmap(tab_b1$proportions, cluster_row = FALSE,
    cluster_col = FALSE, main = "pbmc Batch 1 Comparison",
```

```
    silent = TRUE)
heat_b2 <- pheatmap(tab_b2$proportions, cluster_row = FALSE,
    cluster_col = FALSE, main = "pbmc Batch 2 Comparison",
    silent = TRUE)

gridExtra::grid.arrange(heat_b1[[4]], heat_b2[[4]])
```




```
b1_ari <- pairwiseRand(cluster_labels_b1_before, cluster_labels_b1_after,
    mode = "index", adjusted = TRUE)
b2_ari <- pairwiseRand(cluster_labels_b2_before, cluster_labels_b2_after,
    mode = "index", adjusted = TRUE)
print(paste0("Batch 1 ARI: ", b1_ari))
#> [1] "Batch 1 ARI: 0.85803713474977"
```

```
print(paste0("Batch 2 ARI: ", b2_ari))
#> [1] "Batch 2 ARI: 0.745231800827657"
```

Skipping the heatmaps and examining the cell-type heterogeneity conservation metrics, we can clearly see this improves significantly for batch 2 (0.745 vs 0.663) and is stable for batch 1 (0.858 vs 0.866). This result meets our criteria for conserving biological heterogeneity and its tradeoff between batch mixing much better.

We'll test one more approach for integration, which is Harmony, and then decide whether or not to iterate further.

Integration stage

Method selection

Let's integrate the data using Harmony instead of Seurat.

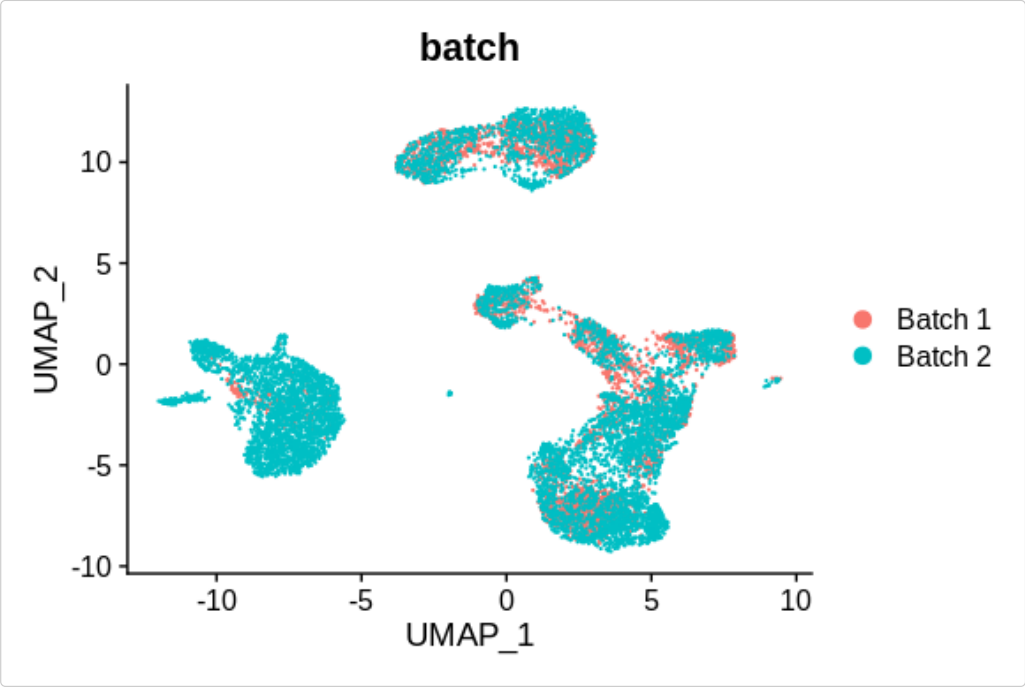
```
library(harmony)

# Remove previous integration objects to save
# space/memory
rm(pbmc_integrated)
rm(pbmc_list)
gc()
#>             used   (Mb) gc trigger    (Mb)   max used   (Mb)
#> Ncells   4631939  247.4    8241467   440.2    8241467  440.2
#> Vcells 549342592 4191.2 1476055682 11261.5 1229973884 9384.0

# Merge the pbmc objects to learn a joint PCA
# which harmony will correct
pbmc_combined <- merge(pbmc_1, pbmc_2)
pbmc_combined <- NormalizeData(pbmc_combined, normalization.method = "LogNormalize",
    scale.factor = 10000)
pbmc_combined <- FindVariableFeatures(pbmc_combined,
    nfeatures = 2000)
all.genes <- rownames(pbmc_combined)
pbmc_combined <- ScaleData(pbmc_combined, features = all.genes)
pbmc_combined <- RunPCA(pbmc_combined, features = VariableFeatures(pbmc_combined))

# Perform harmony-based correction/integration in
# the PCA space
pbmc_combined <- RunHarmony(pbmc_combined, "batch")

# Use the harmony reduction to perform clustering
# and UMAP
pbmc_combined <- FindNeighbors(pbmc_combined, reduction = "harmony",
    dims = 1:15)
pbmc_combined <- FindClusters(pbmc_combined, resolution = 0.5,
    algorithm = "leiden")
pbmc_combined <- RunUMAP(pbmc_combined, reduction = "harmony",
    dims = 1:15)
```

Post-integration stage

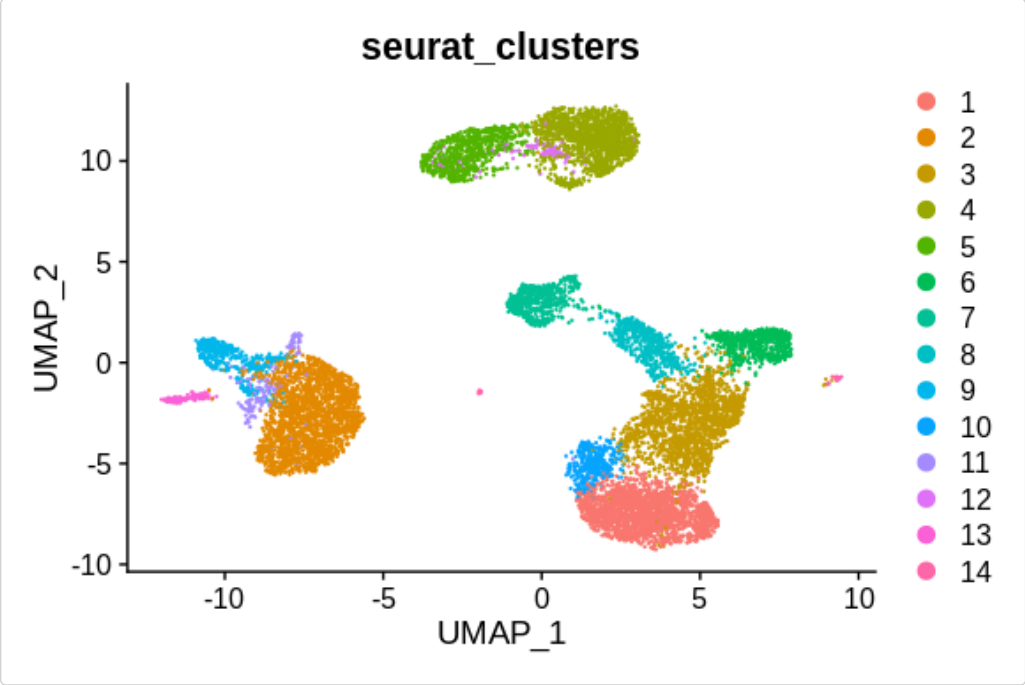
Measure degree of batch mixing post-integration

let's visualize the results of Harmony integration and quantify batch mixing:

```
library(bluster)

# Plot UMAP representation of batch and
# unsupervised clustering results
DimPlot(pbmc_combined, reduction = "umap", group.by = "batch")
```




```
DimPlot(pbmc_combined, reduction = "umap", group.by = "seurat_clusters")
```

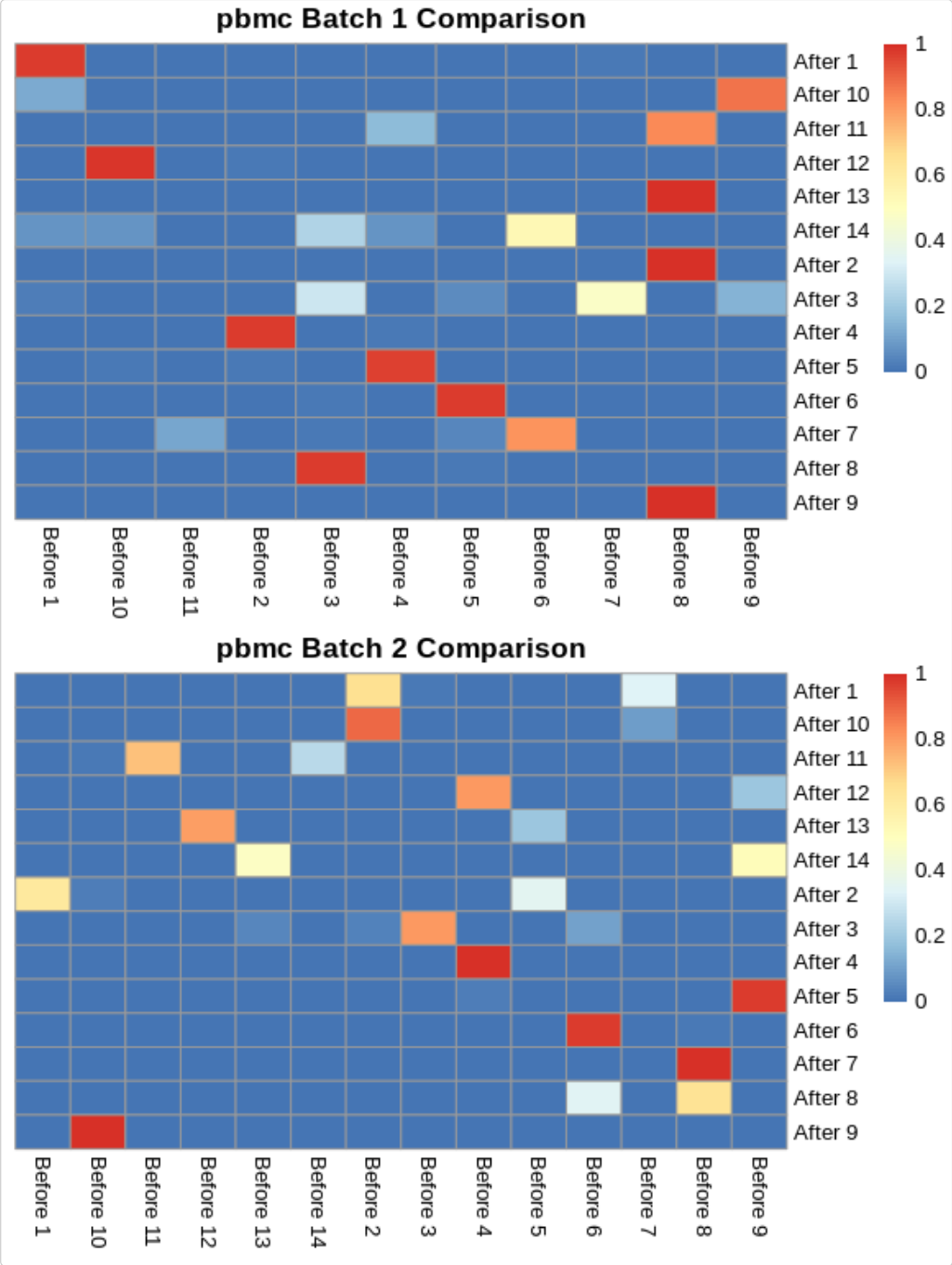


```
# Get the value for batch mixing ARI
batch_labels <- pbmc_combined@meta.data$batch
integrated_cluster_labels <- pbmc_combined@meta.data$seurat_clusters
batch_ari <- 1 - pairwiseRand(batch_labels, integrated_cluster_labels,
    mode = "index", adjusted = TRUE)
print(paste0("Batch mixing ARI: ", batch_ari))
#> [1] "Batch mixing ARI: 0.960896384634549"
```

The batch mixing is very strong, similar to Seurat CCA.

Now we can check how well the harmony embeddings retain biological variation post-integration:

```
cluster_labels_b1_after <-
        pbmc_combined@meta.data$seurat_clusters[pbmc_combined@meta.data$batch ==
    "Batch 1"]
cluster_labels_b2_after <-
        pbmc_combined@meta.data$seurat_clusters[pbmc_combined@meta.data$batch ==
    "Batch 2"]

tab_b1 <- nestedClusters(ref = paste("Before", cluster_labels_b1_before),
    alt = paste("After", cluster_labels_b1_after))
tab_b2 <- nestedClusters(ref = paste("Before", cluster_labels_b2_before),
    alt = paste("After", cluster_labels_b2_after))

heat_b1 <- pheatmap(tab_b1$proportions, cluster_row = FALSE,
    cluster_col = FALSE, main = "pbmc Batch 1 Comparison",
    silent = TRUE)
heat_b2 <- pheatmap(tab_b2$proportions, cluster_row = FALSE,
    cluster_col = FALSE, main = "pbmc Batch 2 Comparison",
    silent = TRUE)

gridExtra::grid.arrange(heat_b1[[4]], heat_b2[[4]])
```




```
b1_ari <- pairwiseRand(cluster_labels_b1_before, cluster_labels_b1_after,
    mode = "index", adjusted = TRUE)
b2_ari <- pairwiseRand(cluster_labels_b2_before, cluster_labels_b2_after,
    mode = "index", adjusted = TRUE)
print(paste0("Batch 1 ARI: ", b1_ari))
#> [1] "Batch 1 ARI: 0.848217273400374"
print(paste0("Batch 2 ARI: ", b2_ari))
#> [1] "Batch 2 ARI: 0.656938798995398"
```

As we can see from the results, although Harmony does well on Batch 1, like Seurat CCA, it performs very poorly on Batch 2. This result also gives us more confidence that our assessment of loss of biological information from batch 2 was correct, as it was recapitulated from multiple methods.

As the rPCA version of Seurat performed best, and we are mostly okay with the tradeoff between batch mixing and preserving biological heterogeneity, we can now **proceed with Seurat rPCA for downstream**

analysis. Observing that batch 2 had potentially lost information in the integrated space, we must keep this fact in mind even with the better performing rPCA integrated space when performing downstream analyses, and be vigilant with respect to annotation and subsequent pipelines. Specifically, we observed in the manuscript experiments that fine-grained cell-types are very difficult to recover in integration scenarios where their more coarse counterparts are affected by the transcriptomic similarity and cell-type support properties (see Results Section II - Query-to-reference projection and cell-type annotation). Therefore, due to the observed loss of information in batch 2, fine-grained annotations within the integrated space must be carefully considered, and it may be best to annotate the batches individually if fine-grained cell-types need to be recovered.

This case study demonstrates how the guidelines for single-cell integration in imbalanced settings can be utilized, and how users can tune *the tradeoff* based on their preferences.

Pulling the curtain

Now let's revisit what we knew about this data before we pretended to forget that we had cell-type labels - in batch 2 we did not have CD4+ T cells and CD14+ Monocytes, but they were present in batch 1. From the manuscript, we know that *relative cell-type support* and *minimum cell-type center distance* are the two facts that contribute to quantitation differences post-integration. In fact, this case-study used the same data (minus some tweaks and balancing) as the main pbmc perturbation experiments in the paper - the major difference being that batch 2 had two cell-types ablated compared to batch 1, instead of just 1 cell-type being downsampled or ablated. In the paper experiments, loss of biological heterogeneity was observed subsequently through the KNN-classification experiments for the downsampled/ablated cell-types and their related cell-types. As such, it should be no surprise that integrating with Seurat/Harmony leads to worse biological heterogeneity conservation scores for the cells from batch 2. We demonstrated how to use the guidelines to alleviate these effects even when we don't have cell-type labels, but it must be made clear that these are heuristics and must be utilized carefully. One factor that is not included is whether or not the researchers have a prior on whether or not there is going to be an imbalance present - such as in the case of developmental or tumor data. In this case, it may be prudent to actually consider tuning the integration method extensively before the first iteration.

Counterfactual - what if we didn't use the guidelines?

To conclude, let's consider a scenario where we didn't use the guidelines. What would happen to the biological heterogeneity in the batches after integration? What might our conclusions look like and what might they miss? To test such a case, let's go ahead and do integration with Seurat CCA without any checks:

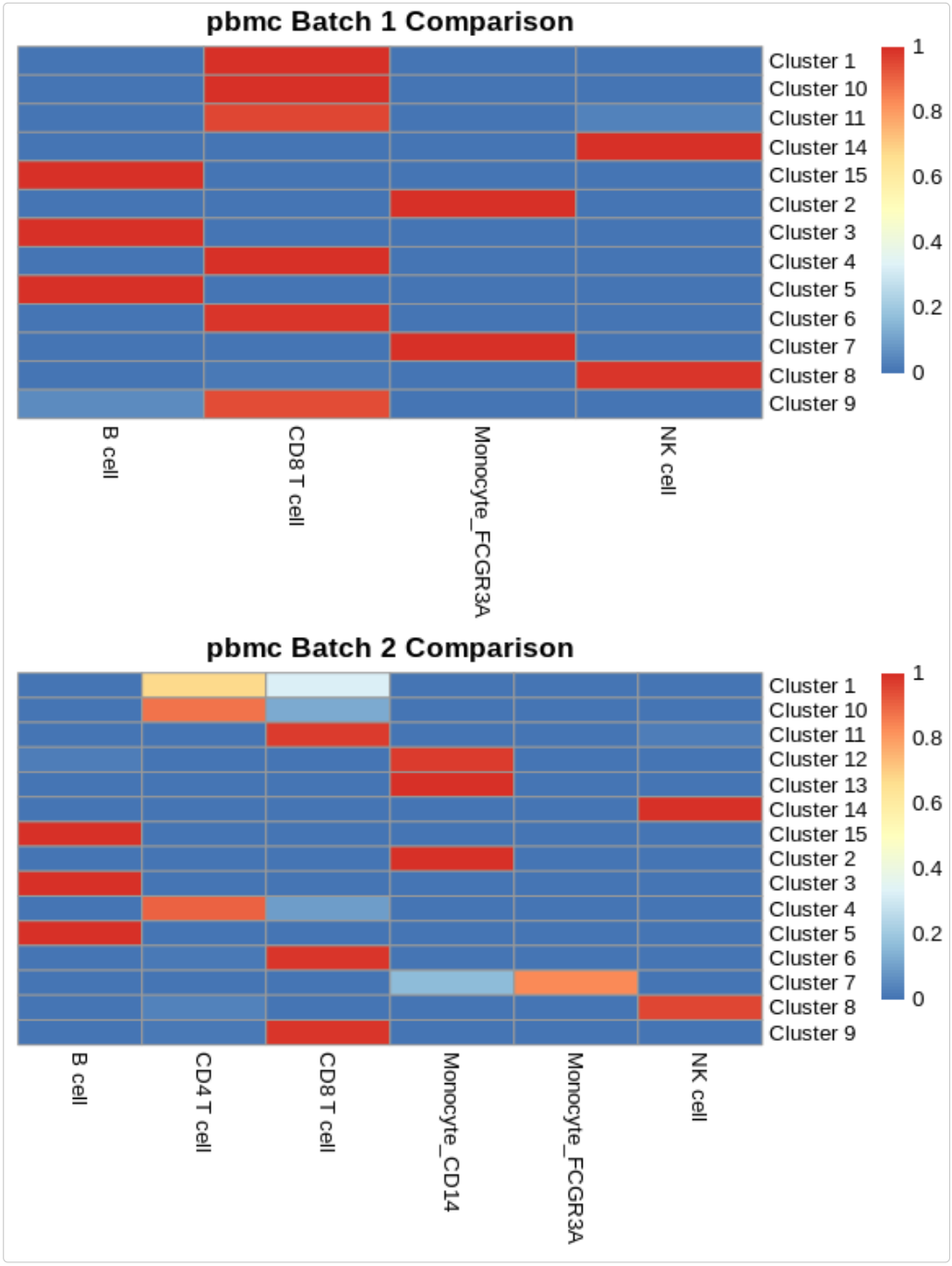
```
# Clean up some files and reload the pbmc data
# from scratch
rm(pbmc_combined)
rm(pbmc_1)
rm(pbmc_2)
gc()
#>             used   (Mb) gc trigger    (Mb)   max used    (Mb)
#> Ncells   4679377  250.0    8241467   440.2    8241467   440.2
#> Vcells 519809458 3965.9 1417077455 10811.5 1476055137 11261.5

pbmc_1 <- SeuratDisk::LoadH5Seurat("data/tran_et_al_batch_1.h5seurat")
pbmc_2 <- SeuratDisk::LoadH5Seurat("data/tran_et_al_batch_2.h5seurat")

pbmc_1@meta.data$batch <- "Batch 1"
pbmc_2@meta.data$batch <- "Batch 2"

# Re-normalize and get variable genes again
pbmc_1 <- NormalizeData(pbmc_1, normalization.method = "LogNormalize",
```

```
    scale.factor = 10000, verbose = FALSE)
pbmc_1 <- FindVariableFeatures(pbmc_1, nfeatures = 2000,
    verbose = FALSE)
pbmc_2 <- NormalizeData(pbmc_2, normalization.method = "LogNormalize",
    scale.factor = 10000, verbose = FALSE)
pbmc_2 <- FindVariableFeatures(pbmc_2, nfeatures = 2000,
    verbose = FALSE)

# Obtain anchors between the two batches and
# perform CCA-base integration using Seurat (the
# default)
pbmc_list <- c(pbmc_1, pbmc_2)
features <- SelectIntegrationFeatures(object.list = pbmc_list,
    verbose = FALSE)
anchors <- FindIntegrationAnchors(object.list = pbmc_list,
    anchor.features = features, verbose = FALSE)
pbmc_integrated <- IntegrateData(anchorset = anchors,
    verbose = FALSE)

# Using the integrated representation, obtain a
# reduced representation through PCA, cluster the
# joint representation and obtain UMAP
# coordinates
pbmc_integrated <- ScaleData(pbmc_integrated, verbose = FALSE)
pbmc_integrated <- RunPCA(pbmc_integrated, verbose = FALSE)
pbmc_integrated <- FindNeighbors(pbmc_integrated, reduction = "pca",
    dims = 1:15, verbose = FALSE)
pbmc_integrated <- FindClusters(pbmc_integrated, resolution = 0.5,
    algorithm = "leiden", verbose = FALSE)
pbmc_integrated <- RunUMAP(pbmc_integrated, reduction = "pca",
    dims = 1:15, verbose = FALSE)
```

Now that we've pulled the curtain, we can actually assess what the concordance between the ground-truth cell-types in each batch and the unsupervised clusters in the integrated space might look like. Let's go ahead and do that:

```
# Get post-integration cluster labels and
# cell-type labels
cluster_labels_b1_after <-
        pbmc_integrated@meta.data$seurat_clusters[pbmc_integrated@meta.data$batch ==
    "Batch 1"]
cluster_labels_b2_after <-
        pbmc_integrated@meta.data$seurat_clusters[pbmc_integrated@meta.data$batch ==
    "Batch 2"]
celltype_labels_b1 <- pbmc_1@meta.data$celltype
celltype_labels_b2 <- pbmc_2@meta.data$celltype

tab_b1 <- nestedClusters(ref = celltype_labels_b1,
    alt = paste("Cluster", cluster_labels_b1_after))
tab_b2 <- nestedClusters(ref = celltype_labels_b2,
    alt = paste("Cluster", cluster_labels_b2_after))

# Assess concordance using the same heatmaps
heat_b1 <- pheatmap(tab_b1$proportions, cluster_row = FALSE,
    cluster_col = FALSE, main = "pbmc Batch 1 Comparison",
```

```
    silent = TRUE)
heat_b2 <- pheatmap(tab_b2$proportions, cluster_row = FALSE,
    cluster_col = FALSE, main = "pbmc Batch 2 Comparison",
    silent = TRUE)

gridExtra::grid.arrange(heat_b1[[4]], heat_b2[[4]])
```




```
# Get the ARI value for each batch
# post-integration
b1_ari <- pairwiseRand(celltype_labels_b1, cluster_labels_b1_after,
    mode = "index", adjusted = TRUE)
b2_ari <- pairwiseRand(celltype_labels_b2, cluster_labels_b2_after,
    mode = "index", adjusted = TRUE)
```

```r
print(paste0("Batch 1 ARI: ", b1_ari))
#> [1] "Batch 1 ARI: 0.370744446296072"
print(paste0("Batch 2 ARI: ", b2_ari))
#> [1] "Batch 2 ARI: 0.581535711707347"
```

As we can see from both the correspondence of the ground-truth cell-types and to the unsupervised clusters through the heatmaps and the ARI value, there is a significant loss of information after integration. The ARI value for batch 2 is quite low (0.58) and it's even lower for batch 1 (0.37). In particular, we can see that in batch 2, the CD14+ and CD16+ (FCGR3A) monocytes are mixed together in several unsupervised clusters, as are the CD4+ and CD8+ T cells. This result is not expected for PBMC data with only 2 batches - the major driver of these results is once again imbalance. From this result, we know we'd end up losing a lot of information post-integration if we didn't seek to determine if there was an imbalance in the first place and try to account for it.

We just saw how the ARI value looks like, but there's something a bit off - the ARI value for batch 1 was much lower than that of batch 2, but that contradicts our unsupervised clustering comparisons in the case where we didn't know the cell-type labels. Further, we can clearly see that the batch 2 cell-types demonstrate problematic behavior in the integrated space due to overlap in the unsupervised clusters. This is happening because even within batches, there is imbalance present. As indicated in the manuscript, the ARI value does not take into account cell-type imbalance (see Results Section V) and may lead to skewed interpretations in terms of integration performance for imbalanced scenarios. In batch 1, most of the cells are from CD8+ T cells, and because we see that they are being mixed with CD4+ T cells from the batch 2 heatmap, this is driving down the ARI value due their prevalence. We can reexamine the results using the balanced ARI, which is available in the balanced clustering python package. Let's use `reticulate` to run the balanced ARI on our Seurat CCA integration results:

```python
from balanced_clustering import balanced_adjusted_rand_index
import numpy as np

celltype_labels_b1_arr = np.array(r.celltype_labels_b1)
celltype_labels_b2_arr = np.array(r.celltype_labels_b2)

cluster_labels_b1_arr = np.array(r.cluster_labels_b1_after)
cluster_labels_b2_arr = np.array(r.cluster_labels_b2_after)

bal_ari_celltype_b1 = balanced_adjusted_rand_index(
  celltype_labels_b1_arr,
  cluster_labels_b1_arr
)
bal_ari_celltype_b2 = balanced_adjusted_rand_index(
  celltype_labels_b2_arr,
  cluster_labels_b2_arr
)

print("Batch 1 balanced ARI: " + str(bal_ari_celltype_b1))
#> Batch 1 balanced ARI: 0.7096428398912037
print("Batch 2 balanced ARI: " + str(bal_ari_celltype_b2))
#> Batch 2 balanced ARI: 0.6759698088497215
```

As we can see from the results, the balanced ARI values are overall higher. This is because the ARI, which will mostly account for the most prevalent cell-types, strongly penalized the results as the CD8+ T cells in batch 1and CD14+ Monocytes and CD4+ T cells in batch 2. However, the B cells and NK cells are largely integrated well, and even though they are less prevalent, they should be accounted for. The **most important aspect** that the balanced ARI captures more faithfully than the base ARI is that the information loss from batch 2 is overall greater than that of batch 1, when considering all cell-types to have equal weight, as shown by the lower score for batch 2.

Although the balanced ARI and other balanced metrics cannot be effectively utilized in the guidelines because cell-type labels are not known a-priori in these integration scenarios, they have substantiative importance in benchmarking and tested methods across datasets.

Through the guidelines, we've alleviated some of this information loss, but this is never going to be perfect. Nonetheless, it's imperative that discrepancies in proportions within and across batches for cell-types and states be accounted for when considering integration. We didn't address extremely challenging scenarios such as development and cancer biology, and in many of those cases the best course of action may not be to integrate at all and instead opt for individual-sample-level analysis.