ensemblemerge package documentation

Erik Skie

2021-03-29

Contents

Chapter 1

Package Information

1.1 Description

ensemblemerge is a package that implements a common work flow for several single cell RNA sequence integration methods including Seurat V3, Scanorama, Harmony, Liger, and bbknn. The merging process is designed to be as streamlined as possible for the user, only requiring a `SummarizedExperiment` or `SingleCellExperiment` object and a specification of which method should be used to integrate in `setParams()`.

1.1.1 Quick Start Guide

Once **ensemblemerge** is installed, merging batches can be performed by the following:

```
library(ensemblemerge)

data #some SingleCellExperiment or SummarizedExperiment object with batches to be integrated

params = setParams() #different integration methods can be selected by setting method = c("Seurat

merged_data = Merge(params, data)
```

A `SingleCellExperiment` object is returned in `merged_data` with accompanying dimension reductions if applicable for the integration methods.

1.2 Prerequisites

The following R packages are required for installation of **ensemblemerge**.

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install(c("SingleCellExperiment", "SummarizedExperiment", "LoomExperiment"
install.packages("Seurat")
devtools::install_github('theislab/kBET')
devtools::install_github('satijalab/seurat-wrappers')
devtools::install_github("cellgeni/sceasy")
```

If you would like to use the python based integration methods, the following packages must also be installed in your python environment. If you do not have a python environment set, users can also call `ensemblemerge::install_python_packages()` which will install the python prerequisites into a reticulated python environment.

```
pip3 install scanpy
pip3 install bbknn
pip3 install scanorama
```

The **ensemblemerge** package can be installed from CRAN or Github (*package is currently in alpha and only available on github to collaborators*):

```
install.packages("ensemblemerge")
# or the development version
# devtools::install_github("erikjskie/ensemblemerge")
```

Chapter 2

Merge example vignettes

For more information on specific function parameters and class structures, see sections 3 and 4 respectively. This chapter covers basic vignettes of different integration methods using the example dataset provided in our package under `dendrites` from Villani et al. 2017.

2.1 Processing data without merging batches

The following work flow is for processing data without performing batch correction. This can be done to prepare your data for downstream visualization without having to manually perform normalizing and scaling.

```
library(ensemblemerge)

params = setParams(method = "Uncorrected", batch = "batchlb") #any additional processing argument

uncorrected = Merge(params, dendrites)
```

`uncorrected` contains normalized and scaled read counts and a PCA reduction. generating a UMAP reduction can be done by the following.

```
library(Seurat)
uncorrected = as.Seurat(uncorrected, counts = "counts", data = "logcounts")
uncorrected <- RunUMAP(uncorrected, reduction = "PCA", dims = 1:20, seed.use = 1)
uncorrected <- FindNeighbors(uncorrected, reduction = 'umap', dims = 1:2, verbose = FALSE)
uncorrected <- FindClusters(uncorrected, algorithm = 3, resolution = 0.01, verbose = FALSE)
uncorrected$cluster <- Idents(uncorrected)
```

Visualize the UMAP.

```
Idents(uncorrected) = "batchlb"
p1 <- DimPlot(uncorrected, reduction = "umap", pt.size = 0.5, shuffle = TRUE) + ggplot2
Idents(uncorrected) = "CellType"
p2 <- DimPlot(uncorrected, reduction = "umap", pt.size = 0.5, shuffle = TRUE) + ggplot2
p1 + p2
```

2.2 Merging with Seurat

This workflow is for integrating batches with the Seurat V3 batch correction method.

```
library(ensemblemerge)

params = setParams(method = "Seurat", batch = "batchlb") #any additional processing ar

seurat_integrate = Merge(params, dendrites)
```

`seurat_integrate` contains normalized and scaled read counts and a PCA reduction. generating a UMAP reduction can be done by the following.

```
library(Seurat)
seurat_integrate = as.Seurat(seurat_integrate, counts = "logcounts", data = "logcounts
seurat_integrate <- RunUMAP(seurat_integrate, reduction = "PCA", dims = 1:20, seed.use
seurat_integrate <- FindNeighbors(seurat_integrate, reduction = 'umap', dims = 1:2, ve
seurat_integrate <- FindClusters(seurat_integrate, algorithm = 3, resolution = 0.01, v
seurat_integrate$cluster <- Idents(seurat_integrate)
```

Visualize the UMAP (*including the uncorrected data for reference*).

```
Idents(uncorrected) = "batchlb"
p1 <- DimPlot(uncorrected, reduction = "umap", pt.size = 0.5, shuffle = TRUE) + ggplot2
Idents(uncorrected) = "CellType"
p2 <- DimPlot(uncorrected, reduction = "umap", pt.size = 0.5, shuffle = TRUE) + ggplot2
Idents(seurat_integrate) = "batchlb"
p3 <- DimPlot(seurat_integrate, reduction = "umap", pt.size = 0.5, shuffle = TRUE) + gg
Idents(seurat_integrate) = "CellType"
p4 <- DimPlot(seurat_integrate, reduction = "umap", pt.size = 0.5, shuffle = TRUE) + gg
p1 + p2 + p3 + p4
```

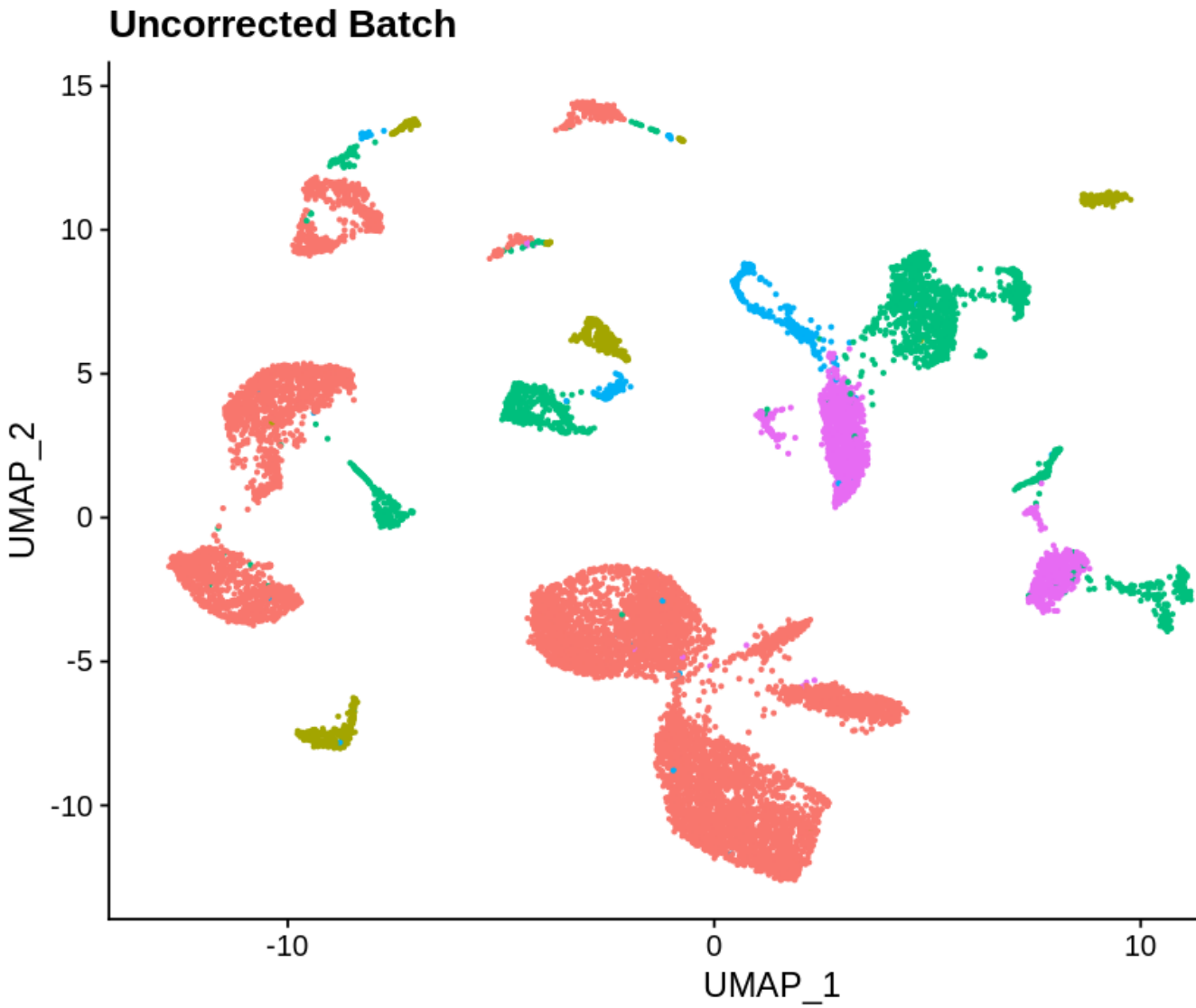

Figure 2.1: Uncorrected UMAP of single cell data by batch and cell type

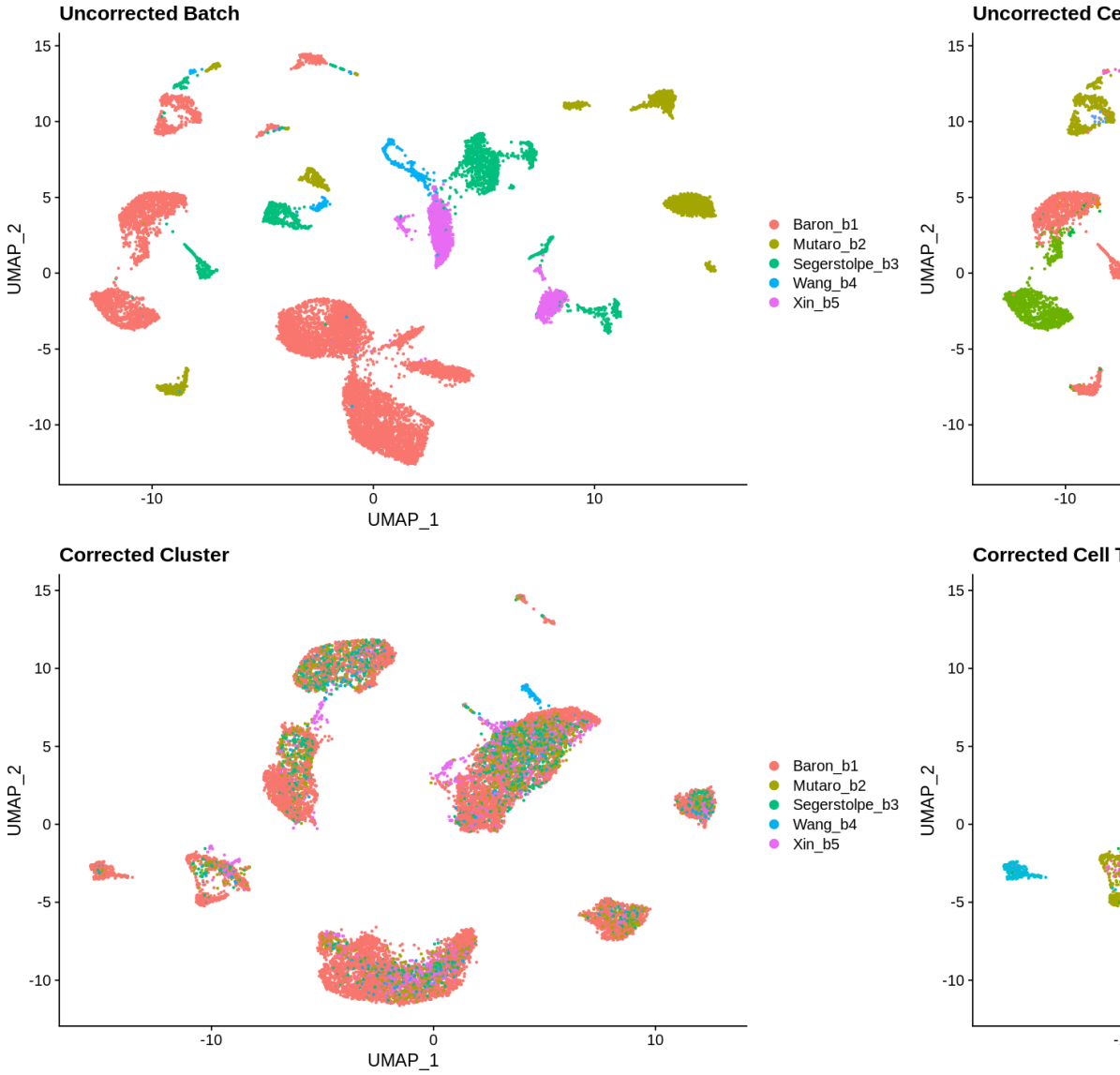

Figure 2.2: Seurat V3 corrected UMAP of single cell data by batch and cell type

2.3 Merging with bbknn

This workflow is for integrating batches with the bbknn batch correction method.

To take full advantage of all available features of bbknn integration, install the following packages.

```
remotes::install_github("rstudio/reticulate") #increases available memory
system("pip3 install pynndescent") #optimizes dimension reduction
system("pip3 install leidenalg") #optional clustering algorithm that improves bbknn performance
```

Perform default bbknn integration.

```
library(ensemblemerge)

params = setParams(method = "BBKNN", batch = "batchlb", return = "Seurat") #any additional proces

bbknn_integrate = Merge(params, dendrites)
```

`bbknn_integrate` contains normalized and scaled read counts and a *bbknn* neighbor graph. generating a UMAP reduction can be done by the following

```
bbknn_integrate <- RunUMAP(bbknn_integrate, nn.name = "bbknn")
```

Visualize the UMAP (*including the uncorrected data for reference*).

```
Idents(uncorrected) = "batchlb"
p1 <- DimPlot(uncorrected, reduction = "umap", pt.size = 0.5, shuffle = TRUE) + ggplot2::ggtitle(
Idents(uncorrected) = "CellType"
p2 <- DimPlot(uncorrected, reduction = "umap", pt.size = 0.5, shuffle = TRUE) + ggplot2::ggtitle(
Idents(bbknn_integrate) = "batchlb"
p3 <- DimPlot(bbknn_integrate, reduction = "umap", pt.size = 0.5, shuffle = TRUE) + ggplot2::ggti
Idents(bbknn_integrate) = "CellType"
p4 <- DimPlot(bbknn_integrate, reduction = "umap", pt.size = 0.5, shuffle = TRUE) + ggplot2::ggti
p1 + p2 + p3 + p4
```

By default, bbknn integration uses ridge regression, which can improve integration by combining both biological and technical effects. This step can be removed by

```
library(ensemblemerge)

params = setParams(method = "BBKNN", batch = "batchlb", ridge_regress = FALSE) #any additional pr

bbknn_integrate = Merge(params, dendrites)
```

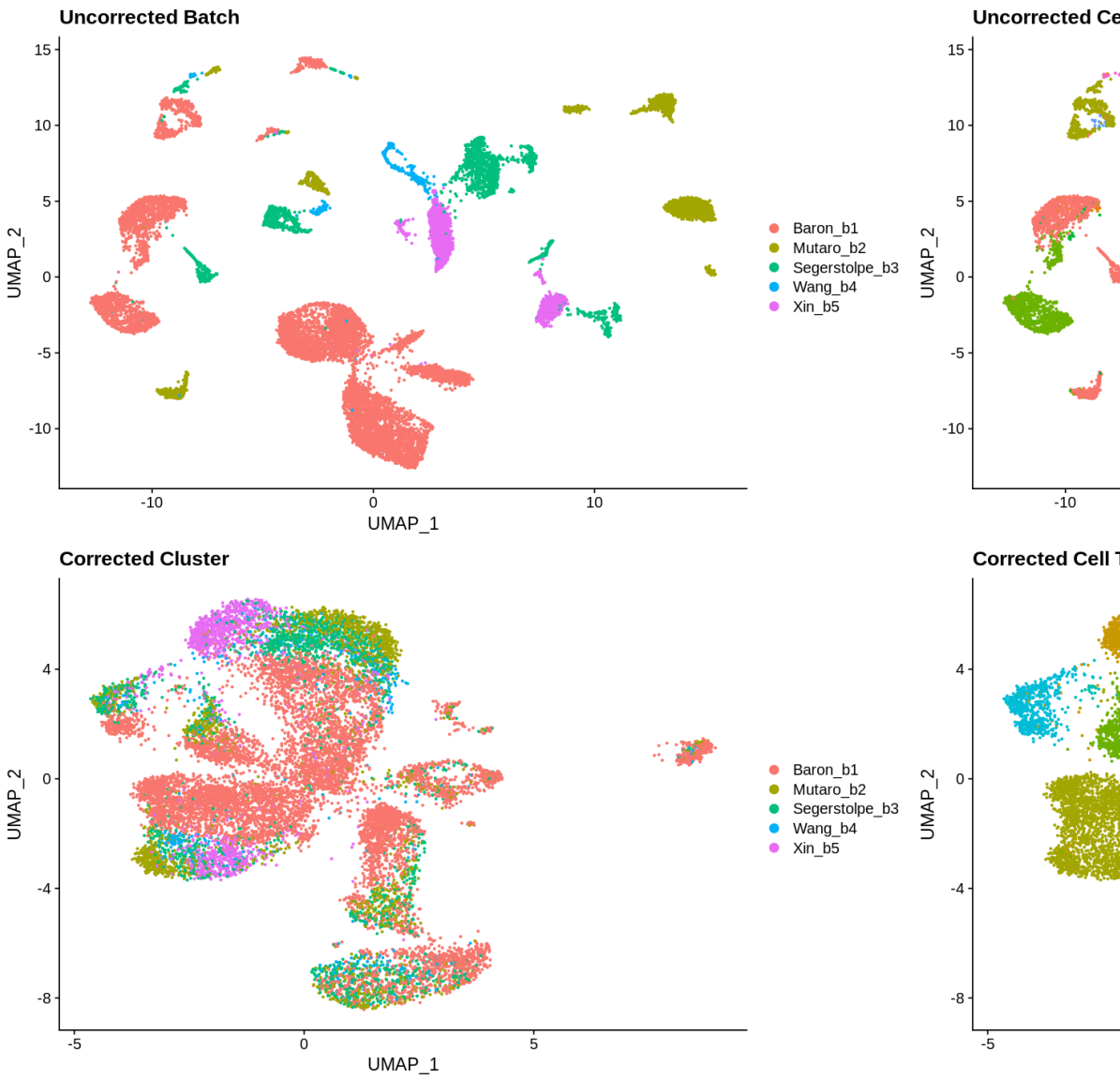

Figure 2.3: bbknn corrected UMAP of single cell data by batch and cell type

`bbknn_integrate` contains normalized and scaled read counts and a *bbknn* neighbor graph. generating a UMAP reduction can be done by the following

```
bbknn_integrate <- RunUMAP(bbknn_integrate, nn.name = "bbknn")
```

Visualize the UMAP (*including the uncorrected data for reference*).

```
Idents(uncorrected) = "batchlb"
p1 <- DimPlot(uncorrected, reduction = "umap", pt.size = 0.5, shuffle = TRUE) + ggplot2::ggtitle(
Idents(uncorrected) = "CellType"
p2 <- DimPlot(uncorrected, reduction = "umap", pt.size = 0.5, shuffle = TRUE) + ggplot2::ggtitle(
Idents(bbknn_integrate) = "batchlb"
p3 <- DimPlot(bbknn_integrate, reduction = "umap", pt.size = 0.5, shuffle = TRUE) + ggplot2::ggti
Idents(bbknn_integrate) = "CellType"
p4 <- DimPlot(bbknn_integrate, reduction = "umap", pt.size = 0.5, shuffle = TRUE) + ggplot2::ggti
p1 + p2 + p3 + p4
```

which can improve performance.

If you have additional cell data you would like to support the integration (i.e. cell type), you can add that by the following

```
library(ensemblemerge)

params = setParams(method = "BBKNN", batch = "batchlb", confounder_key = "CellType") #any additio

bbknn_integrate = Merge(params, dendrites)
```

`bbknn_integrate` contains normalized and scaled read counts and a *bbknn* neighbor graph. generating a UMAP reduction can be done by the following

```
bbknn_integrate <- RunUMAP(bbknn_integrate, nn.name = "bbknn")
```

Visualize the UMAP (*including the uncorrected data for reference*).

```
Idents(uncorrected) = "batchlb"
p1 <- DimPlot(uncorrected, reduction = "umap", pt.size = 0.5, shuffle = TRUE) + ggplot2::ggtitle(
Idents(uncorrected) = "CellType"
p2 <- DimPlot(uncorrected, reduction = "umap", pt.size = 0.5, shuffle = TRUE) + ggplot2::ggtitle(
Idents(bbknn_integrate) = "batchlb"
p3 <- DimPlot(bbknn_integrate, reduction = "umap", pt.size = 0.5, shuffle = TRUE) + ggplot2::ggti
Idents(bbknn_integrate) = "CellType"
p4 <- DimPlot(bbknn_integrate, reduction = "umap", pt.size = 0.5, shuffle = TRUE) + ggplot2::ggti
p1 + p2 + p3 + p4
```

which can improve the quality of integration.

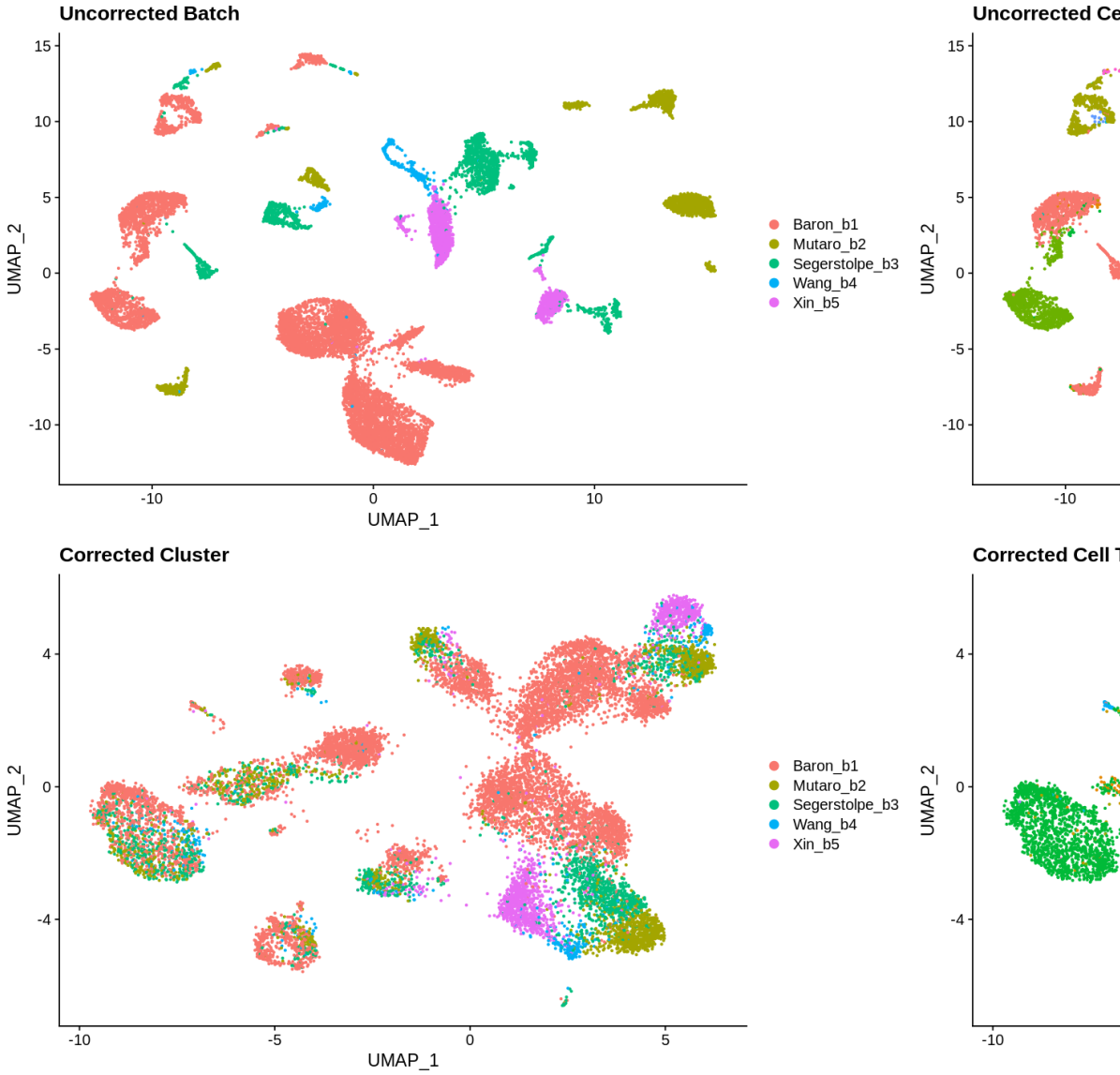

Figure 2.4: bbknn without ridge regression corrected UMAP of single cell data by batch and cell type

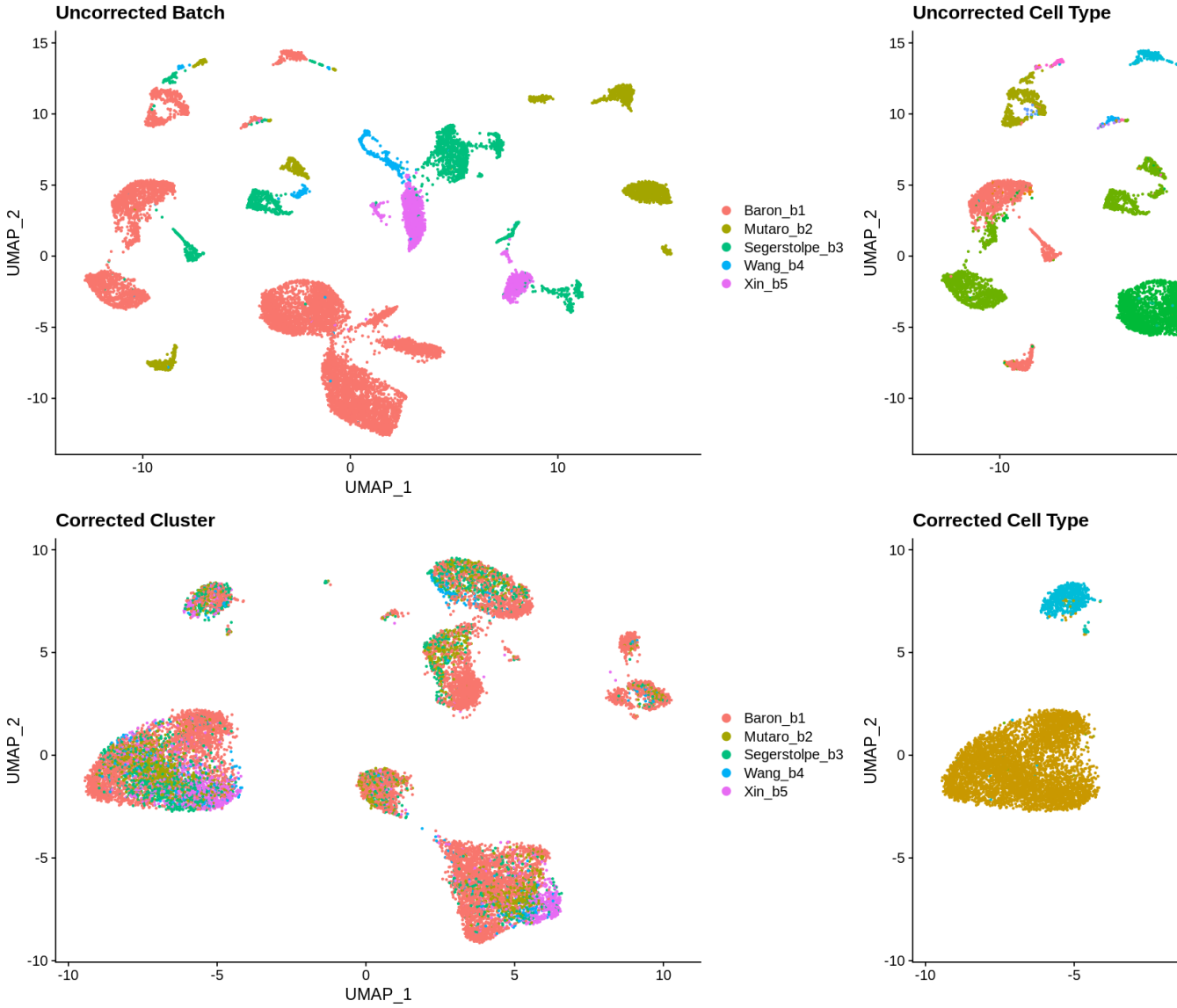

Figure 2.5: bbknn with cell type confound corrected UMAP of single cell data by batch and cell type

Chapter 3

functions

3.1 setParams()

`setParams()` generates a `Params` object of class specified by the `method` argument, i.e. `method = "Seurat"` returns a `SeuratParams` object. The slots of the object can be modified on calling `setParams()` by passing in the name of the slot and the value into the function. i.e `batch = "batch"`

```
setParams(method = "Seurat", ...)
```

Arguments

Argument	Description
method	character vector specifying the integration method, currently the following methods are available `c("Seurat", "Harmony", "Liger", "Scanorama", "BBKNN")`
...	additional arguments that modify the slots of the `Params` object specified by the `methods` argument

3.2 Merge()

`Merge()` generates a `SingleCellExperiment` object that has been batch integrated by the method specified in the `params` object.

```
Merge(params, data)
```

Arguments

Argument	Description
params	A `params` object generated by the `setParams` function
data	A `SummarizedExperiment` or `SingleCellExperiment` object with a metadata identifying the batches

Chapter 4

classes

4.1 UncorrectedParams

Class generated from `setParams(method = "Uncorrected")` for normalizing and scaling data without batch correction.

```
UncorrectedParams(batch = "batch",
          dimreduc_names = c("PCA" = "pca",
                             "UMAP" = "umap",
                             "tSNE" = "tsne"),
          npcs = 20,
          norm_data = TRUE,
          scaling = TRUE,
          regressUMI = FALSE,
          min_cells = 10,
          min_genes = 300,
          norm_method = "LogNormalize",
          scale_factor = 10000,
          numHVG = 2000)
```

Class Structure

Slot	Description
batch	metadata name specifying the batches
dimreduc_names	Names of the dimension reduction objects to return
npcs	Number of PCA dimensions to return in PCA reduction
scaling	boolean specifying whether scaling should be performed

Slot	Description
regressUMI	boolean specifying whether the number of UMI should be regressed
norm_method	character specifying what normalization method to use
min_cells	minimum number of cells expressing a gene to keep for downstream analysis
min_genes	minimum number of genes a cell must express to keep for downstream analysis
scale_factor	Number to scale gene expression to
numHVG	Top N number of genes to assign as highly variable