

# ensemblemerge package documentation

Erik Skie

2021-03-29



# Contents

<b>1</b>	<b>Package Information</b>	<b>5</b>
1.1	Description . . . . .	5
1.2	Prerequisites . . . . .	6
<b>2</b>	<b>Merge example vignettes</b>	<b>7</b>
2.1	Processing data without merging batches . . . . .	7
2.2	Merging with Seurat . . . . .	8
<b>3</b>	<b>functions</b>	<b>11</b>
3.1	setParams() . . . . .	11
3.2	Merge() . . . . .	11
<b>4</b>	<b>classes</b>	<b>13</b>
4.1	UncorrectedParams . . . . .	13



# 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", "Scanorama", "Harmony", "Liger", "bbknn")

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 = "batch1b") #any additional processing arguments

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) = "batch1b"
p1 <- DimPlot(uncorrected, reduction = "umap", pt.size = 0.5, shuffle = TRUE) + ggplot2::theme_minimal()
Idents(uncorrected) = "CellType"
p2 <- DimPlot(uncorrected, reduction = "umap", pt.size = 0.5, shuffle = TRUE) + ggplot2::theme_minimal()
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 = "batch1b") #any additional processing arguments go here

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 = 1)
seurat_integrate <- FindNeighbors(seurat_integrate, reduction = 'umap', dims = 1:2, verbose = FALSE)
seurat_integrate <- FindClusters(seurat_integrate, algorithm = 3, resolution = 0.01, verbose = FALSE)
seurat_integrate$cluster <- Idents(seurat_integrate)

```

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

```

Idents(uncorrected) = "batch1b"
p1 <- DimPlot(uncorrected, reduction = "umap", pt.size = 0.5, shuffle = TRUE) + ggplot2::theme_minimal()
Idents(uncorrected) = "CellType"
p2 <- DimPlot(uncorrected, reduction = "umap", pt.size = 0.5, shuffle = TRUE) + ggplot2::theme_minimal()
Idents(seurat_integrate) = "batch1b"
p3 <- DimPlot(seurat_integrate, reduction = "umap", pt.size = 0.5, shuffle = TRUE) + ggplot2::theme_minimal()
Idents(seurat_integrate) = "CellType"
p4 <- DimPlot(seurat_integrate, reduction = "umap", pt.size = 0.5, shuffle = TRUE) + ggplot2::theme_minimal()
p1 + p2 + p3 + p4

```



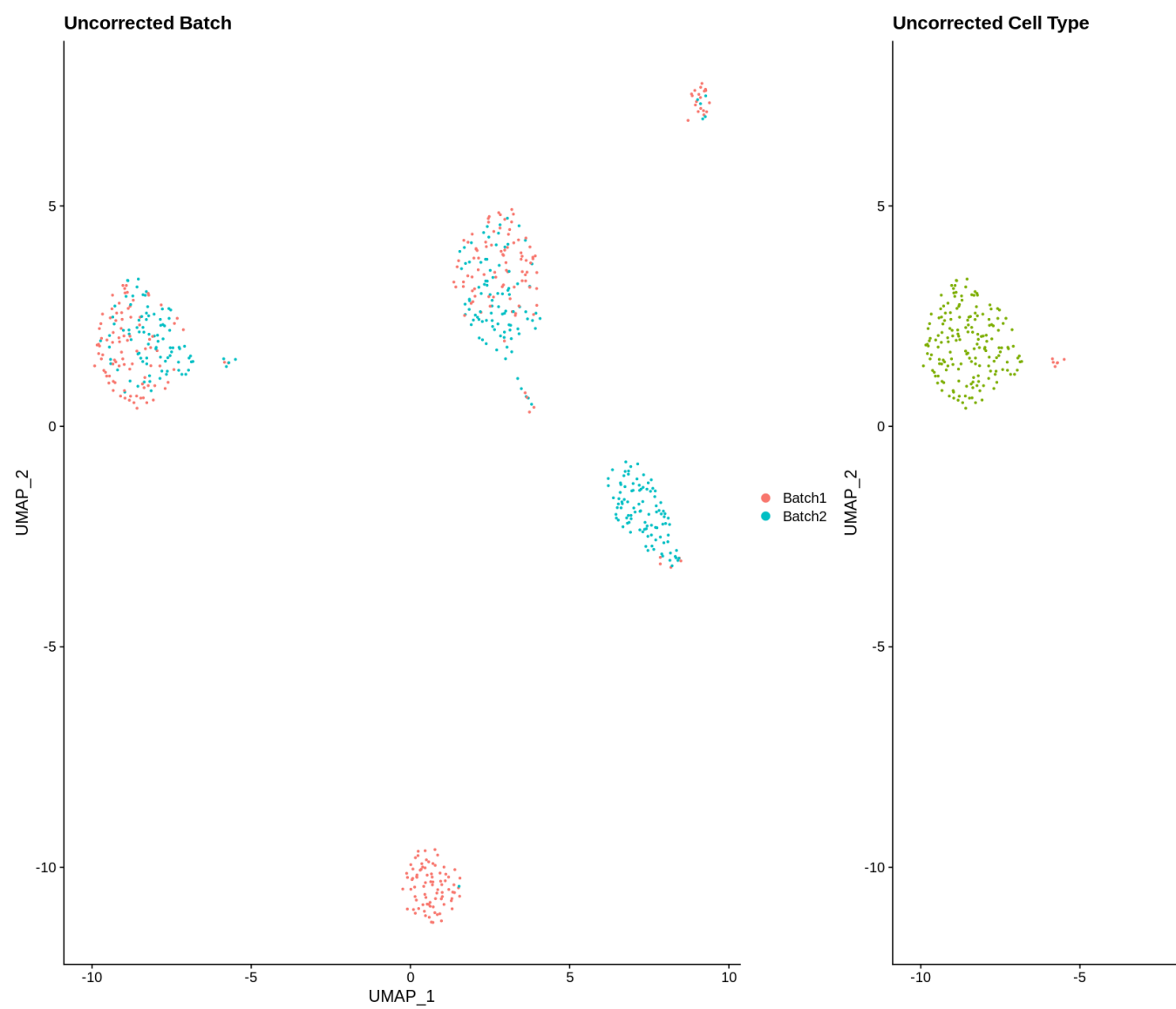


Figure 2.1: Uncorrected UMAP of single cell data by batch and cell type

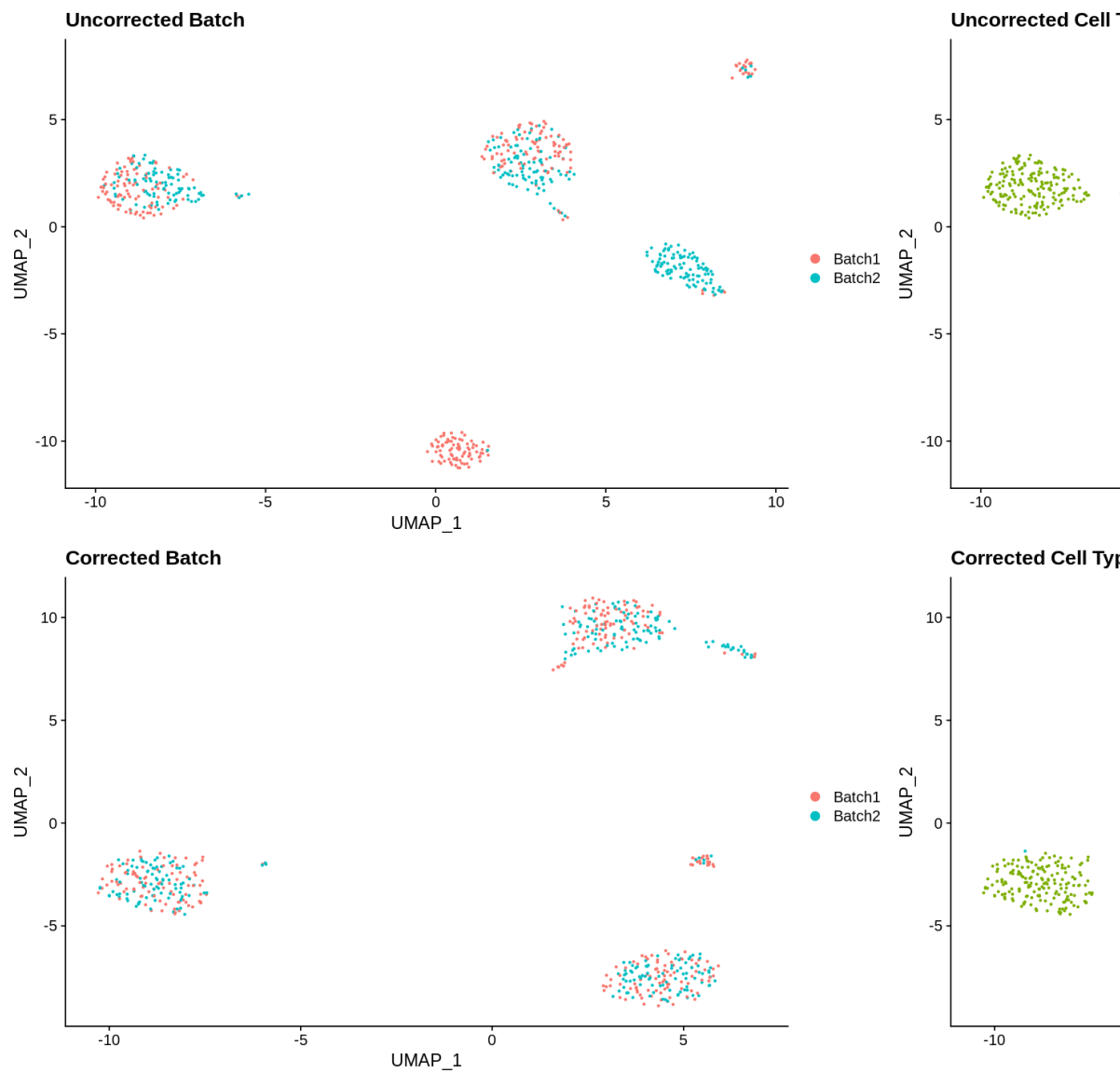


Figure 2.2: Seurat V3 corrected UMAP of single cell data by batch and cell type

## 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
<b>method</b>	character vector specifying the integration method, currently the following methods are available <code>c("Seurat", "Harmony", "Liger", "Scanorama", "BBKNN")</code>
<b>...</b>	additional arguments that modify the slots of the <code>Params</code> object specified by the <code>methods</code> 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
<b>params</b>	A <code>params</code> object generated by the <code>setParams</code> function
<b>data</b>	A <code>SummarizedExperiment</code> or <code>SingleCellExperiment</code> 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
<b>batch</b>	metadata name specifying the batches
<b>dimreduc_names</b>	Names of the dimension reduction objects to return
<b>npcs</b>	Number of PCA dimensions to return in PCA reduction
<b>scaling</b>	boolean specifying whether scaling should be performed

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