SCmap and Seurat single-cell data integration for ABN talk

Virginia Howick

28/02/2021

set working directory and load packages. Scater is used to make the single cell experiment object whilst scmap and Seurat will be used for data integration

Good resource on best R coding practices here: https://style.tidyverse.org/files.html

```
setwd("/Users/virginiahowick/Documents/abn/")
library(scater, quietly = TRUE)
```

```
## Warning: package 'BiocGenerics' was built under R version 4.0.5
```

```
## Warning: package 'GenomeInfoDb' was built under R version 4.0.5
```

```
library(scmap)
library(Seurat)
library(scater)
library(scran)
library(cowplot)
library(gridExtra)
library(viridis)
library(devtools)
```

Read in the data. For today we will be looking at just the IDC (Intra-erythrocytic Developmental Cycle/asexual blood stages) from Pb and Pf. These csvs are the QCed counts and metadata. To quality control Pb cells, we filtered out those with fewer than 230 genes per cell from the filtered expression matrix (generated by cell ranger). For Pf, we used the raw expression matrix from cell ranger and filtered out cells with fewer than 100 genes per cell. Doublets were removed using DoubletFinder.

```
pbcounts <- read.csv("/Users/virginiahowick/Documents/abn/pb10xIDC/pb10xIDC_counts.csv", header=TRUE, r
pbpheno <- read.csv("/Users/virginiahowick/Documents/abn/pb10xIDC/pb10xIDC_pheno.csv", header=TRUE, row

knitr::kable(
  head(pbcounts[ , 1:3]), booktabs = TRUE,
  caption = 'A table of the first 6 rows and 3 columns of the counts table.'
)
```

Table 1: A table of the first 6 rows and 3 columns of the counts table.

	AAACCTGAGCACCGTC	AAACCTGAGCGCTTAT	AAACGGGAGGGTCGAT
PBANKA_0000301	0	0	0
PBANKA_0000600	0	0	0
PBANKA_0001001	0	0	0
PBANKA_0001101	0	0	0
PBANKA_0001201	0	0	0
PBANKA_0006300	0	0	0

```
knitr::kable(
  head(pbpheno[ , 1:3]), booktabs = TRUE,
  caption = 'A table of the first 6 rows and 3 columns of the pheno table.'
)
```

Table 2: A table of the first 6 rows and 3 columns of the pheno table.

	nGene	nUMI	Prediction.Spearman.
AAACCTGAGCACCGTC	1233	2105	10
AAACCTGAGCGCTTAT	1048	1582	8
AAACGGGAGGGTCGAT	1779	3845	10
AAAGATGAGTCACGCC	1278	2201	10
AAAGCAAAGTTAAGTG	1160	1780	10
AAAGCAATCGTAGGAG	1188	1937	4

```
pfcounts <- read.csv("/Users/virginiahowick/Documents/abn/pf10xIDC/pf10xIDC_counts.csv", header=TRUE, r
pfpheno <- read.csv("/Users/virginiahowick/Documents/abn/pf10xIDC/pf10xIDC_pheno.csv", header=TRUE, row

knitr::kable(
  head(pfcounts[ , 1:3]), booktabs = TRUE,
  caption = 'A table of the first 6 rows and 3 columns of the counts table.'
)
```

Table 3: A table of the first 6 rows and 3 columns of the counts table.

	AAACCTGAGCCTCGTG.1	AAACCTGAGCCTTGAT.1	AAACCTGAGCTCCTTC.1
PF3D7_1400200	0	0	0
PF3D7_1400700	0	0	0
PF3D7_1401100	0	0	0
PF3D7_1401200	0	0	0
PF3D7_1401300	0	0	0
PF3D7_1401400	0	0	0

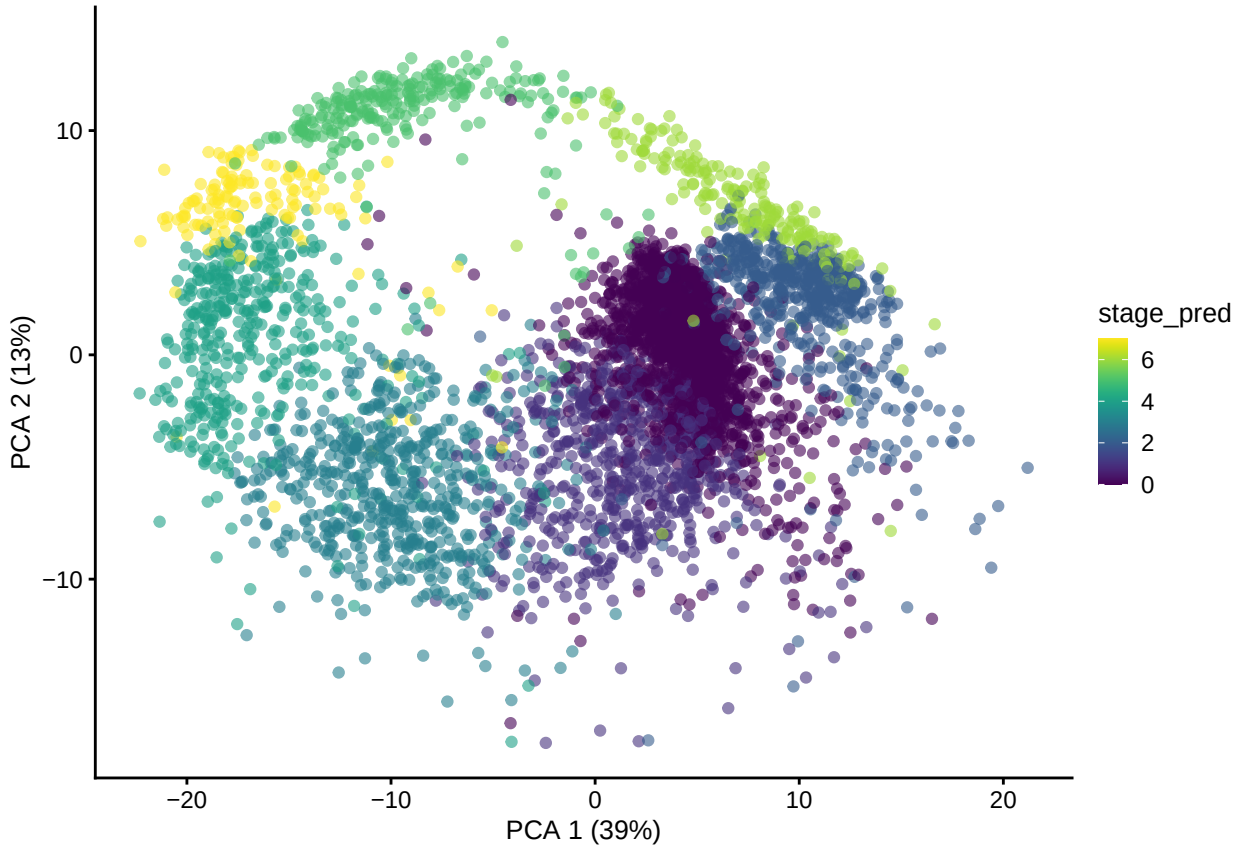
```
knitr::kable(
  head(pfpheno[ , 1:3]), booktabs = TRUE,
  caption = 'A table of the first 6 rows and 3 columns of the pheno table.'
)
```

Table 4: A table of the first 6 rows and 3 columns of the pheno table.

	nGene	nUMI	orig.ident
AAACCTGAGCCTCGTG.1	373	474	threed7
AAACCTGAGCCTTGAT.1	279	377	threed7
AAACCTGAGCTCCTTC.1	354	533	threed7
AAACCTGAGGACATTA.1	317	435	threed7
AAACCTGCAAGACGTG.1	673	1291	threed7
AAACCTGCATGTCCTC.1	443	594	threed7

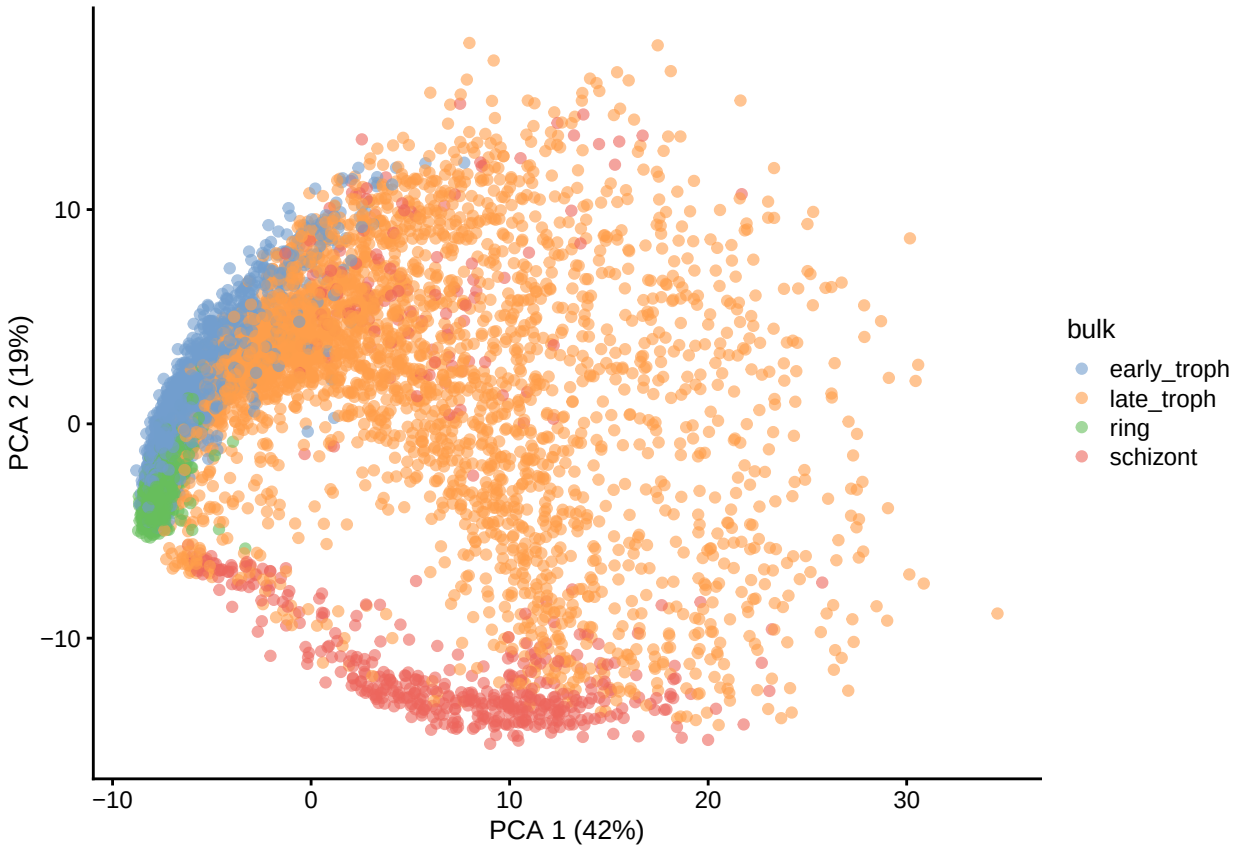
Make both datasets into single cell experiment object

```
pb_sce <- SingleCellExperiment(assays = list(
  counts = as.matrix(pbcounts),
  logcounts = log2(as.matrix(pbcounts) + 1)
), colData = pbpheno)

pb_sce <- runPCA(pb_sce, exprs_values = "logcounts", ntop = 150)
pbpca <- plotPCA(pb_sce, colour_by="stage_pred")
pbpca
```




```
pf_sce <- SingleCellExperiment(assays = list(
  counts = as.matrix(pfcounts),
  logcounts = log2(as.matrix(pfcounts) + 1)
```

```
), colData = pfpheno)

pf_sce <- runPCA(pf_sce, exprs_values = "logcounts", ntop = 150)
pfpca <- plotPCA(pf_sce, colour_by="bulk")
pfpca
```



Now ortho sce for both datasets

```
#Select one-to-one orthologs
orthos <- read.csv("/Users/virginiahowick/Documents/abn/pbpf_orthos.csv")
orthos <- orthos[orthos$paralog_count == 0, ]
orthos <- orthos[isUnique(orthos$ortho_group) == TRUE, ]

#add the ortho group gene name to the pb sce object
rowData(pb_sce)$ortho_group <- orthos[match(rownames(pb_sce), orthos$pb_id), ]$ortho_group
table(is.na(rowData(pb_sce)$ortho_group))
```

```
##
## FALSE  TRUE
##  4265   625
```

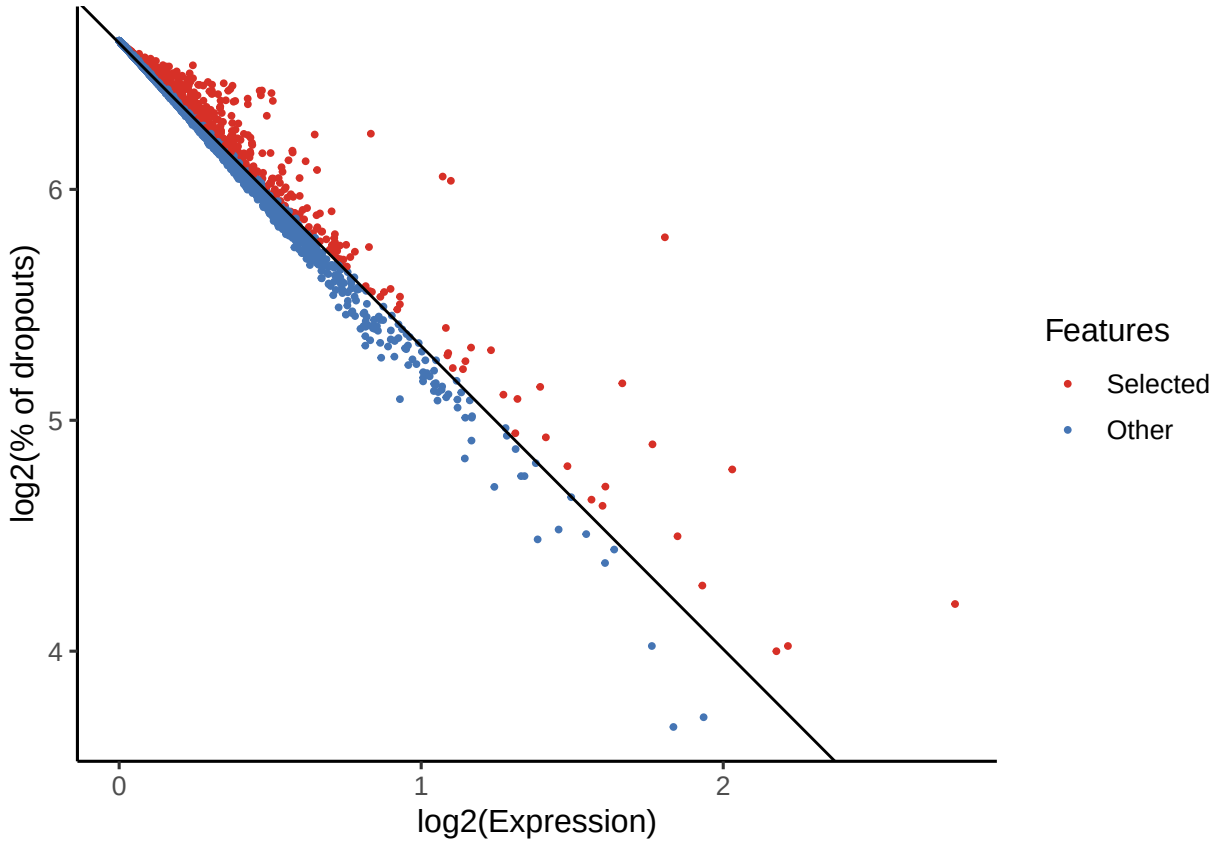
```
rowData(pb_sce)$pb_feature_symbol <- rownames(pb_sce)

#rename to orth object and subset ortho genes
pb_sce_orth <- pb_sce
```

```
pb_sce_orth <- pb_sce_orth[!is.na(rowData(pb_sce_orth)$ortho_group), ]
rownames(pb_sce_orth) <- rowData(pb_sce_orth)$ortho_group

#DO THE SAME THING WITH PF
#add the ortho group gene name to the pf sce object
rowData(pf_sce)$ortho_group <- orthos[match(rownames(pf_sce), orthos$pf_id), ]$ortho_group
table(is.na(rowData(pf_sce)$ortho_group))
```

```
##
## FALSE  TRUE
##  4118   948
```

```
rowData(pf_sce)$pf_feature_symbol <- rownames(pf_sce)

#rename to orth object and subset ortho genes
pf_sce_orth <- pf_sce
pf_sce_orth <- pf_sce_orth[!is.na(rowData(pf_sce_orth)$ortho_group), ]
rownames(pf_sce_orth) <- rowData(pf_sce_orth)$ortho_group

rowData(pf_sce_orth)$feature_symbol <- rownames(pf_sce_orth)
rowData(pb_sce_orth)$feature_symbol <- rownames(pb_sce_orth)
```

```
#build scmap-cell reference index, save this rds
pb_sce_orth <- selectFeatures(pb_sce_orth, suppress_plot = FALSE, n_features = 500)
```



```
table(rowData(pb_sce_orth)$scmap_features)
```

```
##
## FALSE  TRUE
##  3765   500
```

```
set.seed(1)
pb_sce_orth <- indexCell(pb_sce_orth)
```

```
## Parameter M was not provided, will use M = n_features / 10 (if n_features <= 1000), where n_features
```

```
## Parameter k was not provided, will use k = sqrt(number_of_cells)
```

```
names(metadata(pb_sce_orth)$scmap_cell_index)
```

```
## [1] "subcentroids" "subclusters"
```

```
length(metadata(pb_sce_orth)$scmap_cell_index$subcentroids)
```

```
## [1] 50
```

```
dim(metadata(pb_sce_orth)$scmap_cell_index$subcentroids[[1]])
```

```
## [1] 10 69
```

```
metadata(pb_sce_orth)$scmap_cell_index$subcentroids[[1]][,1:5]
```

```
##            1           2           3           4           5
## OG5_127259 0 0.435744777 0.000000000 0.000000000 0.023273170
## OG5_162064 0 0.001974710 0.004535934 0.006655623 0.014899436
## OG5_140540 0 0.000000000 0.434920074 0.000000000 0.611085714
## OG5_150514 0 0.000000000 0.457625987 0.000000000 0.007753617
## OG5_156822 0 0.012414610 0.084968738 0.000000000 0.564964322
## OG5_127428 1 0.899961874 0.024544659 0.999972285 0.111983876
## OG5_164905 0 0.004438794 0.743962901 0.003336584 0.540333561
## OG5_166985 0 0.004170027 0.086319849 0.000000000 0.008504775
## OG5_133090 0 0.000000000 0.167019767 0.000000000 0.038054378
## OG5_166988 0 0.000000000 0.068940316 0.000000000 0.023336668
```

#run scmap

```
#Project query data set onto cell index
scmapCell_results <- scmapCell(
  pf_sce_orth,
  list(
    yan = metadata(pb_sce_orth)$scmap_cell_index
  )
)
```

Look into the results For each dataset there are two matricies. cells matrix contains the top 10 (scmap default) cell IDs of the cells of the reference dataset that a given cell of the projection dataset is closest to:

Give assignments in two ways: 1. Take the top cell assignment abs clust, if cosine similarity is less than 0.4 (or adjust if needed) mark as unassigned 2. For the top 3 nearest neighbors, get a mean of the PCA coordinates and snap to the nearest cell of those coordinates. If any of the top three cells are sim below 0.4 then mark as unassigned.

```
##Top cell assignment method
scmapCell_results$yan$cells[, 1:3]
```

```
##       AAACCTGAGCCTCGTG.1 AAACCTGAGCCTTGAT.1 AAACCTGAGCTCCTTC.1
##  [1,]               4438                729                2474
##  [2,]               2021               3226                2495
##  [3,]               4640                168                3880
##  [4,]               4617                212                2621
##  [5,]                208                102                2605
##  [6,]                537                230                2728
##  [7,]               4637               2998                2570
##  [8,]               4650               1180                2552
##  [9,]               2122               1266                2538
## [10,]               4641                236                2608
```

```
getcells <- scmapCell_results$yan$cells[1, ]
cdsce <- colData(pb_sce_orth)[getcells, ]
topsim <- scmapCell_results$yan$similarities[1, ]

pf_sce_orth$topcell <- cdsce$sample_id
pf_sce_orth$topcell_ac <- cdsce$absclust
pf_sce_orth$indexPC1 <- cdsce$PC1
pf_sce_orth$indexPC2 <- cdsce$PC2
pf_sce_orth$pbpt <- cdsce$pseudotime
pf_sce_orth$pbbulk <- cdsce$bulk
pf_sce_orth$topcell_sp <- pf_sce_orth$topcell_ac
pf_sce_orth$topsim <- topsim
pf_sce_orth$topcell_sp[pf_sce_orth$topsim < 0.3] <- "unassigned"
table(pf_sce_orth$topcell_sp)
```

```
##
##          0          1          2          3          4          5          6
##       1983       1199       1148       1048        593         99        168
##          7 unassigned
##         75        424
```

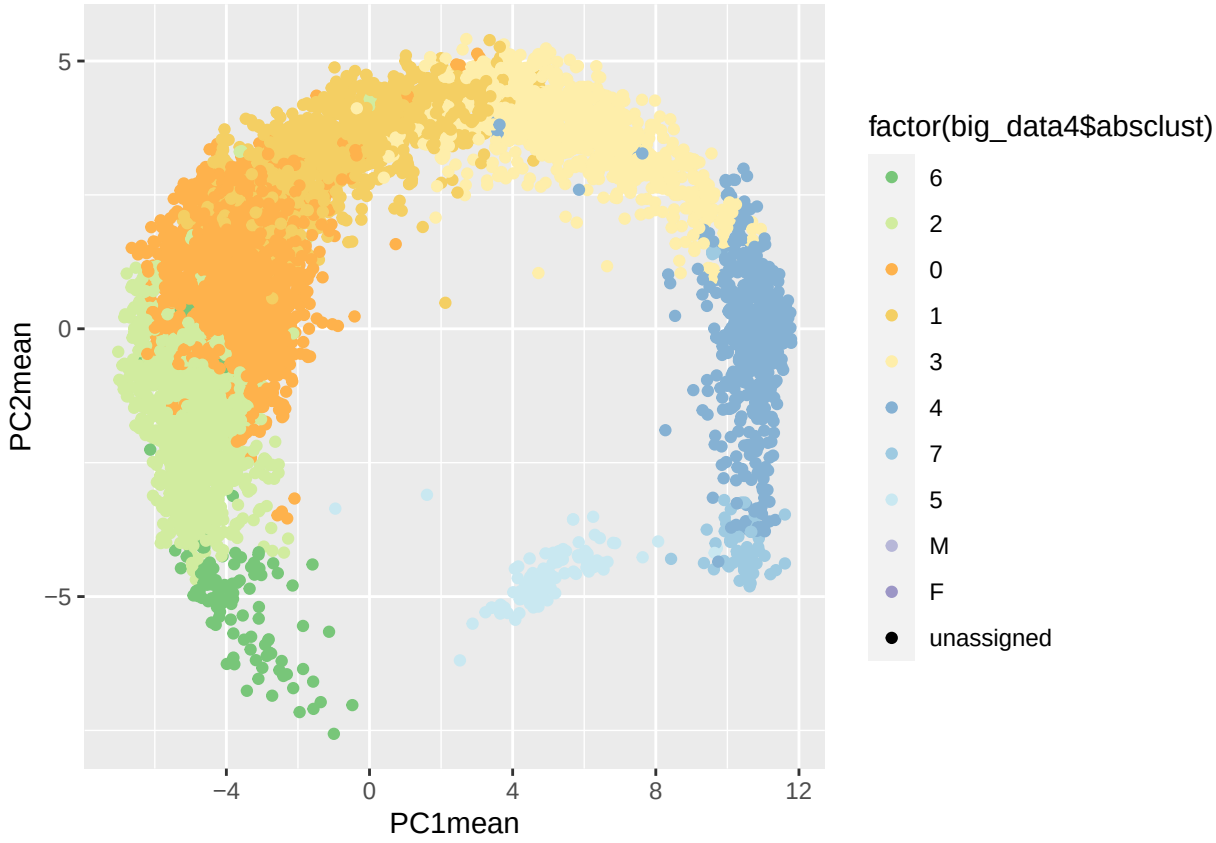
```
#### TOP 3NN method

#This function makes a list of the PC means for each cell and then do.call below rbinds them into a dat


datalist = list()

for (i in colnames(scmapCell_results$yan$cells)) {
```

```
  getcellstest <- scmapCell_results$yan$cells[1:3, i]
  cdscetest <- colData(pb_sce_orth)[getcellstest, ]
  PC1mean <- mean(cdscetest$PC1)
  PC2mean <- mean(cdscetest$PC2)
  # ... make some data
  dat <- data.frame(i, PC1mean, PC2mean)
  dat$i <- i  # maybe you want to keep track of which iteration produced it?
  datalist[[i]] <- dat # add it to your list
}

big_data = do.call(rbind, datalist)
# or big_data <- dplyr::bind_rows(datalist)
# or big_data <- data.table::rbindlist(datalist)

test <- big_data[1, ]

df <- data.frame(X=colData(pb_sce_orth)$PC1, Y=colData(pb_sce_orth)$PC2, row.names = rownames(colData(pb

#the snap function snaps to the nearest cell in PC coordiantes
snap <- function(df, test){
  require(Biobase)
  d <- matchpt(as.matrix(df),
               as.matrix(data.frame(X=test$PC1mean,Y=test$PC2mean)))

  min_row <- rownames(d[d$distance==min(d$distance),])

  test$X_snap <- unique(df[min_row,"X"])
  test$Y_snap <- unique(df[min_row,"Y"])
  test$pb_cell <- min_row

  test
}

#this loops through each cell and in big_data and runs the snap function
datalist2 = list()
colnames(big_data) <- c("sample_id", "PC1mean", "PC2mean")
for (i in rownames(big_data)) {
  test <- big_data[i, ]
  coord <- snap(df, test)
  coord$i <- i
  datalist2[[i]] <- coord
}
big_data2 = do.call(rbind, datalist2)


table(rownames(big_data2)==rownames(colData(pf_sce_orth)))
```

```
##
## TRUE
## 6737
```

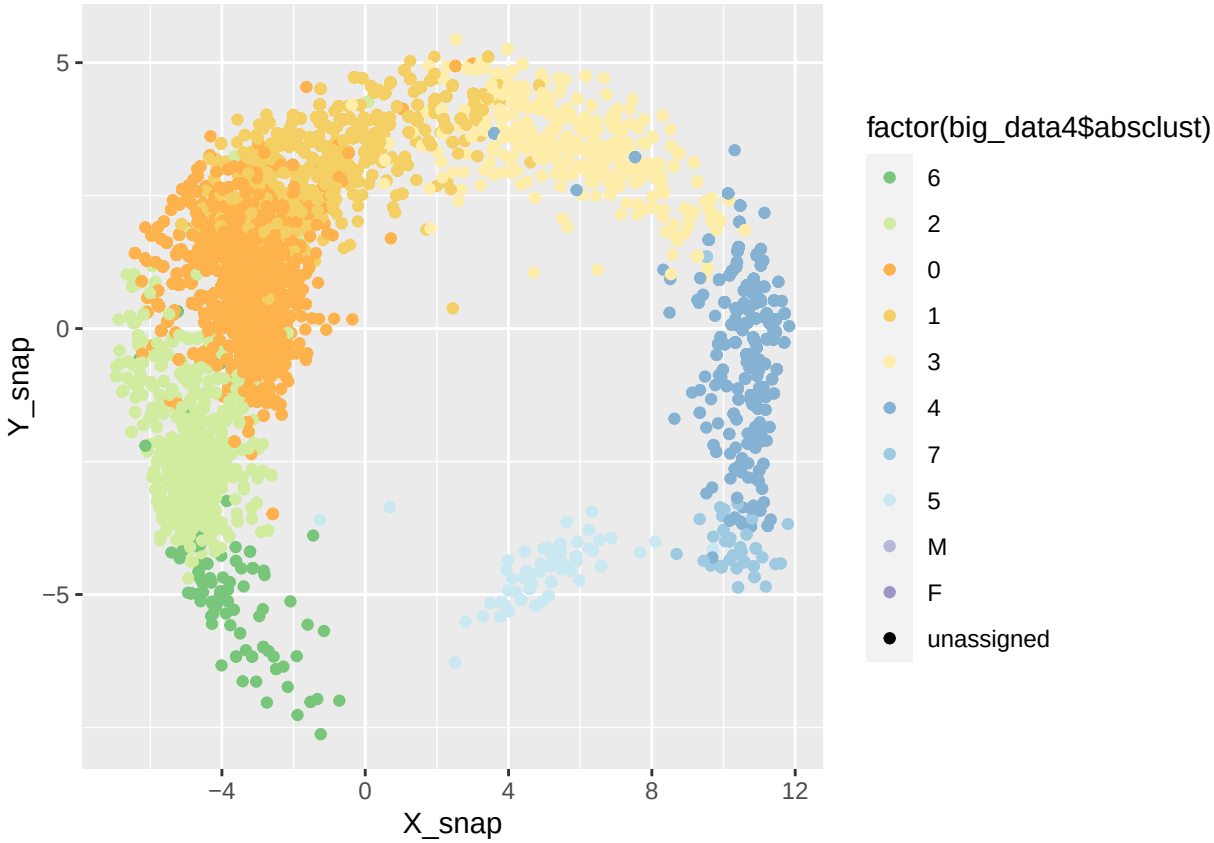
```
allpbcd <- colData(pb_sce_orth)
allpbcd <- as.data.frame(allpbcd)
pbabsclust <- allpbcd[, c("absclust", "clock_pseudotime"), drop=FALSE]
pbabsclust$pb_sample_id <- rownames(pbabsclust)

# Now merge the pc cell asignments with their abs clust and get in the right order
big_data3 <- merge(big_data2, pbabsclust, by.x = "pb_cell", by.y = "pb_sample_id", all.x=TRUE, all.y=FA
big_data4 <- big_data3[match(rownames(big_data2), big_data3$sample_id), ]


colors <- c("6"="#78C679",
            "2"="#D1EC9F",
            "0"="#FEB24C",
            "1"="#F4CF63",
            "3"="#FEEEAA",
            "4"="#85B1D3",
            "7"="#9ecae1",
            "5"="#C9E8F1",
            "M"= "#B7B7D8",
            "F"="#9C96C6",
            "unassigned"="black")


ggplot(big_data4, aes(PC1mean, PC2mean)) + geom_point(aes(colour=factor(big_data4$absclust))) + scale_c
```

```
## Warning: Use of ‘big_data4$absclust‘ is discouraged. Use ‘absclust‘ instead.
```



```
ggplot(big_data4, aes(X_snap, Y_snap)) + geom_point(aes(colour=factor(big_data4$absclust))) + scale_colo
```

```
## Warning: Use of 'big_data4$absclust' is discouraged. Use 'absclust' instead.
```




```
##add info to SCE and save colData, to be an assigned cell all 3NN must have a cos sim >0.4
scmapCell_results$yan$similarities[, 1:3]
```

```
##        AAACCTGAGCCTCGTG.1 AAACCTGAGCCTTGAT.1 AAACCTGAGCTCCTTC.1
##  [1,]          0.4171100          0.4307333          0.4819539
##  [2,]          0.4097487          0.4308877          0.4945335
##  [3,]          0.4209587          0.4326952          0.4946163
##  [4,]          0.4117110          0.4319871          0.4797261
##  [5,]          0.4170785          0.4254663          0.4952173
##  [6,]          0.4163522          0.4329492          0.4795706
##  [7,]          0.4073506          0.4375676          0.4939426
##  [8,]          0.4154736          0.4276614          0.4777773
##  [9,]          0.4094794          0.4459340          0.4815812
## [10,]          0.4203739          0.4652545          0.4786334
```

```
topsim1 <- scmapCell_results$yan$similarities[1, ]
topsim2 <- scmapCell_results$yan$similarities[2, ]
topsim3 <- scmapCell_results$yan$similarities[3, ]

table(big_data4$sample_id==rownames(colData(pf_sce_orth)))
```

```
##
## TRUE
## 6737
```

```
#pf_sce_orth$pb_cell <- big_data4$pb_cell
#pf_sce_orth$PC1mean <- big_data4$PC1mean
bd4 <- big_data4[, c("pb_cell", "sample_id", "PC1mean", "PC2mean", "X_snap", "Y_snap", "absclust", "clo

colData(pf_sce_orth) <- cbind(colData(pf_sce_orth), bd4)

pf_sce_orth$topsim1 <- topsim1
pf_sce_orth$topsim2 <- topsim2
pf_sce_orth$topsim3 <- topsim3
pf_sce_orth$stage_pred <- pf_sce_orth$absclust
pf_sce_orth$stage_pred[pf_sce_orth$topsim1 < 0.3 | pf_sce_orth$topsim2 < 0.3 | pf_sce_orth$topsim3 < 0.
table(pf_sce_orth$stage_pred)
```

```
##
##          0          1          2          3          4          5          6
##       2196       1335       1031        795        600         96         81
##          7 unassigned
##         69        534
```

```
#write.csv(bd4, "pfcellassignmentswithmean3nn_20181029.csv")
#write.csv(colData(pf_sce_orth), "pf3d7100scmapclusts2methodindexn100_20190107.csv")
```

```
pbcounts <- as.data.frame(counts(pb_sce_orth))
pfcounts <- as.data.frame(counts(pf_sce_orth))

pbcd <- as.data.frame(colData(pb_sce_orth))
pfcd <- as.data.frame(colData(pf_sce_orth))

pb_seurat_orth <- CreateSeuratObject(pbcounts, assay = "RNA", meta.data = pbcd)
```

```
## Warning: Feature names cannot have underscores ('_'), replacing with dashes
## ('-')
```

```
## Warning: Feature names cannot have pipe characters ('|'), replacing with dashes
## ('-')
```

```
pf_seurat_orth <- CreateSeuratObject(pfcounts, assay = "RNA", meta.data = pfcd)
```

```
## Warning: Feature names cannot have underscores ('_'), replacing with dashes
## ('-')

## Warning: Feature names cannot have pipe characters ('|'), replacing with dashes
## ('-')
```

```
pb_seurat_orth$species <- rep("Pb", length(pb_seurat_orth$nGene))
pf_seurat_orth$species <- rep("Pf", length(pf_seurat_orth$nGene))
```

```
Idents(pb_seurat_orth) <- "stage_pred"
pb_seurat_orth <- FindVariableFeatures(pb_seurat_orth, selection.method = "vst", nfeatures = 500)
pf_seurat_orth <- FindVariableFeatures(pf_seurat_orth, selection.method = "vst", nfeatures = 500)
```

##Perform integration We then identify anchors using the FindIntegrationAnchors function, which takes a list of Seurat objects as input, and use these anchors to integrate the two datasets together with IntegrateData.

```
p_anchors <- FindIntegrationAnchors(object.list = list(pb_seurat_orth, pf_seurat_orth),
    dims = 1:20)
```

```
## Computing 2000 integration features

## Scaling features for provided objects

## Finding all pairwise anchors

## Running CCA

## Merging objects

## Finding neighborhoods

## Finding anchors

##  Found 9650 anchors

## Filtering anchors

##  Retained 1844 anchors
```

```
p_combined <- IntegrateData(anchorset = p_anchors, dims = 1:20)
```

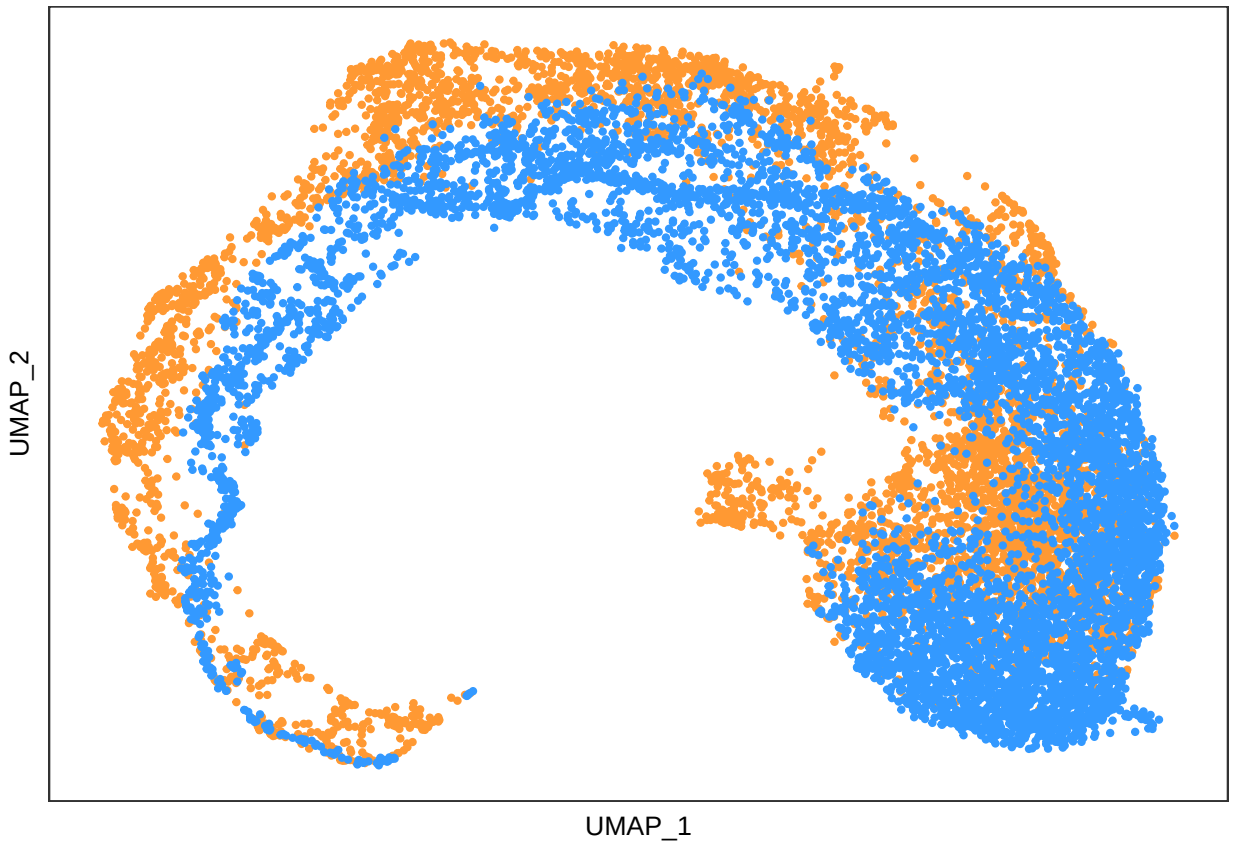
```
## Merging dataset 1 into 2

## Extracting anchors for merged samples

## Finding integration vectors

## Finding integration vector weights

## Integrating data
```

##Perform an integrated analysis Now we can run a single integrated analysis on all cells!

```r
DefaultAssay(p_combined) <- "integrated"

# Run the standard workflow for visualization and clustering
p_combined <- ScaleData(p_combined, verbose = FALSE)
p_combined <- RunPCA(p_combined, npcs = 30, verbose = FALSE)
# t-SNE and Clustering p.combined <- FindVariableFeatures(p.combined,
# selection.method = 'vst', nfeatures = 500) hvg <- HVFInfo(object = p.combined)
# hvgfeat <- rownames(hvg)
p_combined <- RunUMAP(p_combined, reduction = "pca", dims = 1:20, umap.method = "uwot",
    n.neighbors = 10, min.dist = 0.3, spread = 1, seed.use = 222)
```

```
## Warning: The default method for RunUMAP has changed from calling Python UMAP via reticulate to the R
## To use Python UMAP via reticulate, set umap.method to 'umap-learn' and metric to 'correlation'
## This message will be shown once per session

## 09:54:58 UMAP embedding parameters a = 0.9922 b = 1.112

## 09:54:58 Read 11500 rows and found 20 numeric columns

## 09:54:58 Using Annoy for neighbor search, n_neighbors = 10

## 09:54:58 Building Annoy index with metric = cosine, n_trees = 50

## 0%   10   20   30   40   50   60   70   80   90   100%

## [----|----|----|----|----|----|----|----|----|----|

## **************************************************|
## 09:54:59 Writing NN index file to temp file /var/folders/4p/hz2xpdw94j39xw648pf35t3w0000gn/T//RtmpQs
## 09:54:59 Searching Annoy index using 1 thread, search_k = 1000
## 09:55:01 Annoy recall = 100%
## 09:55:02 Commencing smooth kNN distance calibration using 1 thread
## 09:55:03 Initializing from normalized Laplacian + noise
## 09:55:03 Commencing optimization for 200 epochs, with 166252 positive edges
## 09:55:08 Optimization finished
```

```r
p <- DimPlot(p_combined, reduction = "umap", group.by = "species")
pdat <- p$data

colors = c(Pf = "#3399FF", Pb = "#FF9933")

ggplot(pdat, aes(UMAP_1, UMAP_2)) + geom_point(aes(colour = species), size = 0.8) +
    theme_bw() + scale_colour_manual(values = colors) + theme(axis.text = element_blank(),
    axis.ticks = element_blank(), panel.grid.major = element_blank(), panel.grid.minor = element_blank()
    axis.title = element_text(size = 10), legend.position = "none")
```



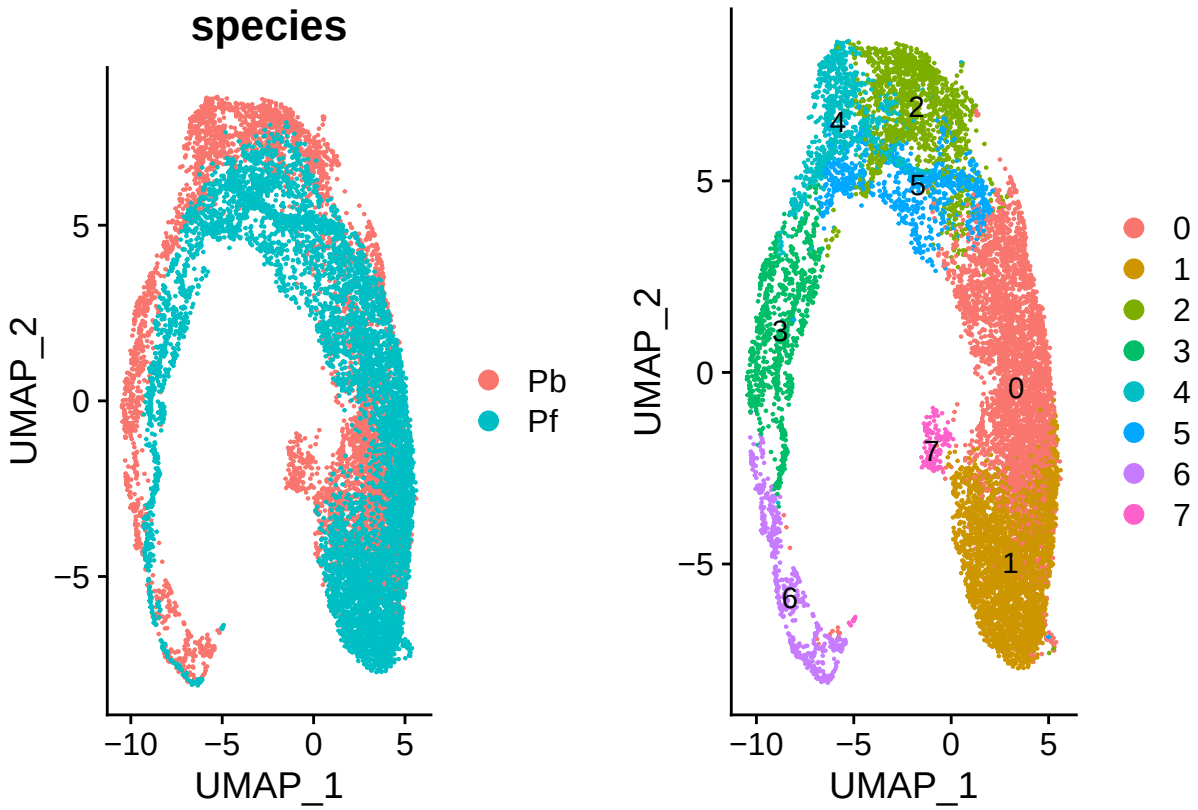

```
p_combined <- FindNeighbors(p_combined, reduction = "pca", dims = 1:20)
```

```
## Computing nearest neighbor graph
## Computing SNN
```

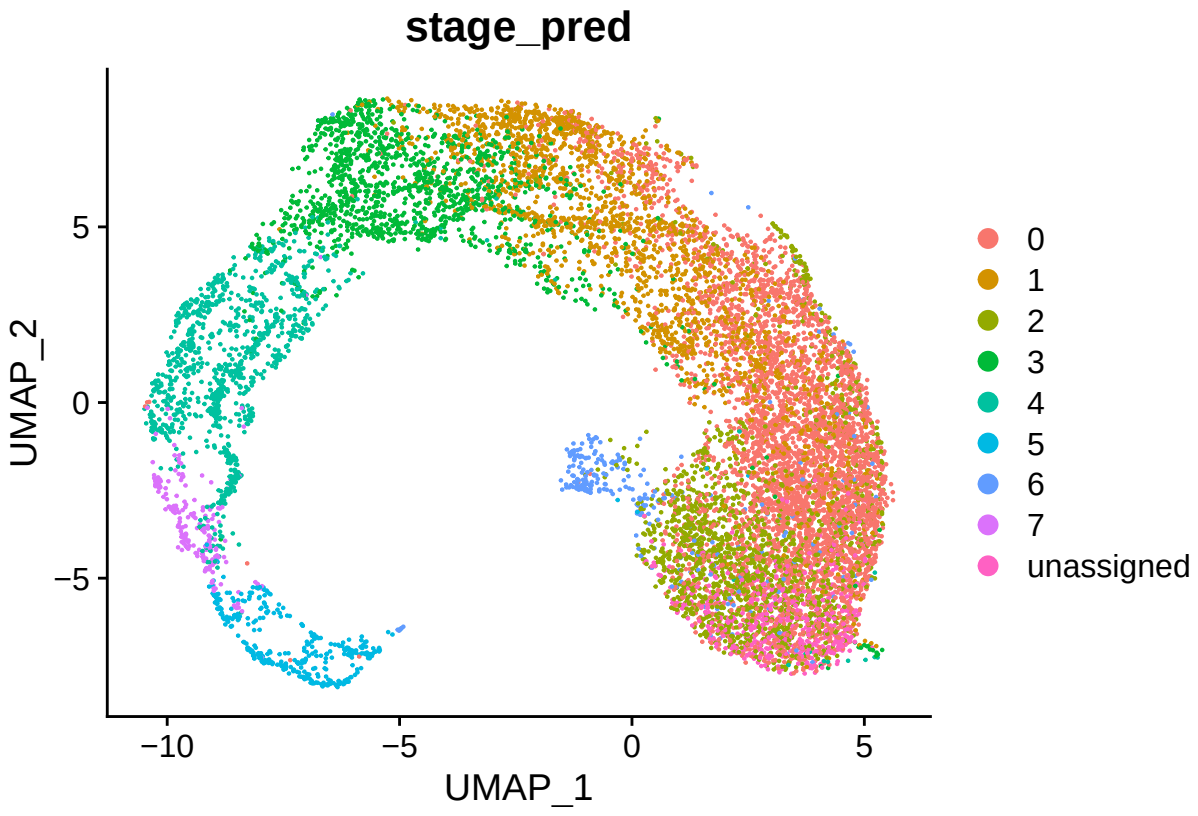
```
p_combined <- FindClusters(p_combined, resolution = 0.3)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 11500
## Number of edges: 396245
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8976
## Number of communities: 8
## Elapsed time: 1 seconds
```

```
p1 <- DimPlot(p_combined, reduction = "umap", group.by = "species")
p2 <- DimPlot(p_combined, reduction = "umap", label = TRUE)
plot_grid(p1, p2)
```




```
DimPlot(p_combined, reduction = "umap", group.by = "stage_pred")
```



```
session_info()
```

```
## - Session info ---------------------------------------------------------------
##  setting  value
##  version  R version 4.0.3 (2020-10-10)
##  os       macOS Big Sur 10.16
##  system   x86_64, darwin17.0
##  ui       X11
##  language (EN)
##  collate  en_GB.UTF-8
##  ctype    en_GB.UTF-8
##  tz       Europe/London
##  date     2022-02-28
##
## - Packages -------------------------------------------------------------------
##  package              * version  date       lib source
##  abind                  1.4-5    2016-07-21 [1] CRAN (R 4.0.2)
##  assertthat             0.2.1    2019-03-21 [1] CRAN (R 4.0.2)
##  beachmat               2.6.3    2020-12-12 [1] Bioconductor
##  beeswarm               0.2.3    2016-04-25 [1] CRAN (R 4.0.2)
##  Biobase              * 2.50.0   2020-10-27 [1] Bioconductor
##  BiocGenerics         * 0.36.1   2021-04-16 [1] Bioconductor
##  BiocNeighbors          1.8.2    2020-12-07 [1] Bioconductor
##  BiocParallel           1.24.1   2020-11-06 [1] Bioconductor
##  BiocSingular           1.6.0    2020-10-27 [1] Bioconductor
##  bitops                 1.0-7    2021-04-24 [1] CRAN (R 4.0.2)
##  bluster                1.0.0    2020-10-27 [1] Bioconductor
##  callr                  3.5.1    2020-10-13 [1] CRAN (R 4.0.2)
##  class                  7.3-17   2020-04-26 [1] CRAN (R 4.0.3)
##  cli                    3.0.1    2021-07-17 [1] CRAN (R 4.0.2)
##  cluster                2.1.0    2019-06-19 [1] CRAN (R 4.0.3)
##  codetools              0.2-18   2020-11-04 [1] CRAN (R 4.0.2)
##  colorspace             2.0-2    2021-06-24 [1] CRAN (R 4.0.2)
##  cowplot              * 1.1.0    2020-09-08 [1] CRAN (R 4.0.2)
##  crayon                 1.4.1    2021-02-08 [1] CRAN (R 4.0.2)
##  data.table             1.13.4   2020-12-08 [1] CRAN (R 4.0.2)
##  DBI                    1.1.0    2019-12-15 [1] CRAN (R 4.0.2)
##  DelayedArray           0.16.3   2021-03-24 [1] Bioconductor
##  DelayedMatrixStats     1.12.1   2020-11-24 [1] Bioconductor
##  deldir                 0.2-3    2020-11-09 [1] CRAN (R 4.0.2)
##  desc                   1.2.0    2018-05-01 [1] CRAN (R 4.0.2)
##  devtools             * 2.3.2    2020-09-18 [1] CRAN (R 4.0.2)
##  digest                 0.6.27   2020-10-24 [1] CRAN (R 4.0.2)
##  dplyr                  1.0.7    2021-06-18 [1] CRAN (R 4.0.2)
##  dqrng                  0.2.1    2019-05-17 [1] CRAN (R 4.0.2)
##  e1071                  1.7-8    2021-07-28 [1] CRAN (R 4.0.2)
##  edgeR                  3.32.0   2020-10-27 [1] Bioconductor
##  ellipsis               0.3.2    2021-04-29 [1] CRAN (R 4.0.2)
##  evaluate               0.14     2019-05-28 [1] CRAN (R 4.0.1)
##  fansi                  0.5.0    2021-05-25 [1] CRAN (R 4.0.2)
##  farver                 2.1.0    2021-02-28 [1] CRAN (R 4.0.2)
##  fastmap                1.0.1    2019-10-08 [1] CRAN (R 4.0.2)
##  fitdistrplus           1.1-3    2020-12-05 [1] CRAN (R 4.0.2)
```

```
##  formatR               1.7      2019-06-11 [1] CRAN (R 4.0.2)
##  fs                    1.5.0    2020-07-31 [1] CRAN (R 4.0.2)
##  future                1.21.0   2020-12-10 [1] CRAN (R 4.0.3)
##  future.apply          1.6.0    2020-07-01 [1] CRAN (R 4.0.2)
##  generics              0.1.0    2020-10-31 [1] CRAN (R 4.0.2)
##  GenomeInfoDb        * 1.26.7   2021-04-08 [1] Bioconductor
##  GenomeInfoDbData      1.2.4    2020-12-11 [1] Bioconductor
##  GenomicRanges       * 1.42.0   2020-10-27 [1] Bioconductor
##  ggbeeswarm            0.6.0    2017-08-07 [1] CRAN (R 4.0.2)
##  ggplot2             * 3.3.5    2021-06-25 [1] CRAN (R 4.0.2)
##  ggrepel               0.8.2    2020-03-08 [1] CRAN (R 4.0.2)
##  ggridges              0.5.2    2020-01-12 [1] CRAN (R 4.0.2)
##  globals               0.14.0   2020-11-22 [1] CRAN (R 4.0.2)
##  glue                  1.4.2    2020-08-27 [1] CRAN (R 4.0.2)
##  goftest               1.2-2    2019-12-02 [1] CRAN (R 4.0.2)
##  googleVis             0.6.10   2021-02-19 [1] CRAN (R 4.0.2)
##  gridExtra           * 2.3      2017-09-09 [1] CRAN (R 4.0.2)
##  gtable                0.3.0    2019-03-25 [1] CRAN (R 4.0.2)
##  highr                 0.8      2019-03-20 [1] CRAN (R 4.0.2)
##  htmltools             0.5.0    2020-06-16 [1] CRAN (R 4.0.2)
##  htmlwidgets           1.5.3    2020-12-10 [1] CRAN (R 4.0.3)
##  httpuv                1.5.4    2020-06-06 [1] CRAN (R 4.0.2)
##  httr                  1.4.2    2020-07-20 [1] CRAN (R 4.0.2)
##  ica                   1.0-2    2018-05-24 [1] CRAN (R 4.0.2)
##  igraph                1.2.6    2020-10-06 [1] CRAN (R 4.0.2)
##  IRanges             * 2.24.1   2020-12-12 [1] Bioconductor
##  irlba                 2.3.3    2019-02-05 [1] CRAN (R 4.0.2)
##  jsonlite              1.7.2    2020-12-09 [1] CRAN (R 4.0.3)
##  KernSmooth            2.23-18  2020-10-29 [1] CRAN (R 4.0.2)
##  knitr                 1.30     2020-09-22 [1] CRAN (R 4.0.2)
##  labeling              0.4.2    2020-10-20 [1] CRAN (R 4.0.2)
##  later                 1.1.0.1  2020-06-05 [1] CRAN (R 4.0.2)
##  lattice               0.20-41  2020-04-02 [1] CRAN (R 4.0.3)
##  lazyeval              0.2.2    2019-03-15 [1] CRAN (R 4.0.2)
##  leiden                0.3.6    2020-12-07 [1] CRAN (R 4.0.2)
##  lifecycle             1.0.0    2021-02-15 [1] CRAN (R 4.0.2)
##  limma                 3.46.0   2020-10-27 [1] Bioconductor
##  listenv               0.8.0    2019-12-05 [1] CRAN (R 4.0.2)
##  lmtest                0.9-38   2020-09-09 [1] CRAN (R 4.0.2)
##  locfit                1.5-9.4  2020-03-25 [1] CRAN (R 4.0.2)
##  magrittr              2.0.1    2020-11-17 [1] CRAN (R 4.0.2)
##  MASS                  7.3-53   2020-09-09 [1] CRAN (R 4.0.3)
##  Matrix                1.2-18   2019-11-27 [1] CRAN (R 4.0.3)
##  MatrixGenerics      * 1.2.1    2021-01-30 [1] Bioconductor
##  matrixStats         * 0.60.0   2021-07-26 [1] CRAN (R 4.0.2)
##  memoise               1.1.0    2017-04-21 [1] CRAN (R 4.0.2)
##  mgcv                  1.8-33   2020-08-27 [1] CRAN (R 4.0.3)
##  mime                  0.9      2020-02-04 [1] CRAN (R 4.0.2)
##  miniUI                0.1.1.1  2018-05-18 [1] CRAN (R 4.0.2)
##  munsell               0.5.0    2018-06-12 [1] CRAN (R 4.0.2)
##  nlme                  3.1-150  2020-10-24 [1] CRAN (R 4.0.2)
##  parallelly            1.21.0   2020-10-27 [1] CRAN (R 4.0.2)
##  patchwork             1.1.0    2020-11-09 [1] CRAN (R 4.0.2)
##  pbapply               1.4-3    2020-08-18 [1] CRAN (R 4.0.2)
```

```
##  pillar                1.6.2    2021-07-29 [1] CRAN (R 4.0.2)
##  pkgbuild              1.1.0    2020-07-13 [1] CRAN (R 4.0.2)
##  pkgconfig             2.0.3    2019-09-22 [1] CRAN (R 4.0.2)
##  pkgload               1.1.0    2020-05-29 [1] CRAN (R 4.0.2)
##  plotly                4.9.2.1  2020-04-04 [1] CRAN (R 4.0.2)
##  plyr                  1.8.6    2020-03-03 [1] CRAN (R 4.0.2)
##  png                   0.1-7    2013-12-03 [1] CRAN (R 4.0.2)
##  polyclip              1.10-0   2019-03-14 [1] CRAN (R 4.0.2)
##  prettyunits           1.1.1    2020-01-24 [1] CRAN (R 4.0.2)
##  processx              3.4.5    2020-11-30 [1] CRAN (R 4.0.2)
##  promises              1.1.1    2020-06-09 [1] CRAN (R 4.0.2)
##  proxy                 0.4-26   2021-06-07 [1] CRAN (R 4.0.2)
##  ps                    1.5.0    2020-12-05 [1] CRAN (R 4.0.2)
##  purrr                 0.3.4    2020-04-17 [1] CRAN (R 4.0.2)
##  R6                    2.5.0    2020-10-28 [1] CRAN (R 4.0.2)
##  randomForest          4.6-14   2018-03-25 [1] CRAN (R 4.0.2)
##  RANN                  2.6.1    2019-01-08 [1] CRAN (R 4.0.2)
##  RColorBrewer          1.1-2    2014-12-07 [1] CRAN (R 4.0.2)
##  Rcpp                  1.0.7    2021-07-07 [1] CRAN (R 4.0.2)
##  RcppAnnoy             0.0.18   2020-12-15 [1] CRAN (R 4.0.2)
##  RCurl                 1.98-1.3 2021-03-16 [1] CRAN (R 4.0.2)
##  remotes               2.2.0    2020-07-21 [1] CRAN (R 4.0.2)
##  reshape2              1.4.4    2020-04-09 [1] CRAN (R 4.0.2)
##  reticulate            1.18     2020-10-25 [1] CRAN (R 4.0.2)
##  rlang                 0.4.11   2021-04-30 [1] CRAN (R 4.0.2)
##  rmarkdown             2.5      2020-10-21 [1] CRAN (R 4.0.3)
##  ROCR                  1.0-11   2020-05-02 [1] CRAN (R 4.0.2)
##  rpart                 4.1-15   2019-04-12 [1] CRAN (R 4.0.3)
##  rprojroot             2.0.2    2020-11-15 [1] CRAN (R 4.0.2)
##  RSpectra              0.16-0   2019-12-01 [1] CRAN (R 4.0.2)
##  rstudioapi            0.13     2020-11-12 [1] CRAN (R 4.0.2)
##  rsvd                  1.0.3    2020-02-17 [1] CRAN (R 4.0.2)
##  Rtsne                 0.15     2018-11-10 [1] CRAN (R 4.0.2)
##  S4Vectors           * 0.28.1   2020-12-09 [1] Bioconductor
##  scales                1.1.1    2020-05-11 [1] CRAN (R 4.0.2)
##  scater              * 1.18.3   2020-11-08 [1] Bioconductor
##  scattermore           0.7      2020-11-24 [1] CRAN (R 4.0.2)
##  scmap               * 1.12.0   2020-10-27 [1] Bioconductor
##  scran               * 1.18.2   2020-12-09 [1] Bioconductor
##  sctransform           0.3.2    2020-12-16 [1] CRAN (R 4.0.2)
##  scuttle               1.0.3    2020-11-23 [1] Bioconductor
##  sessioninfo           1.1.1    2018-11-05 [1] CRAN (R 4.0.2)
##  Seurat              * 4.0.1    2021-03-18 [1] CRAN (R 4.0.2)
##  SeuratObject        * 4.0.0    2021-01-15 [1] CRAN (R 4.0.2)
##  shiny                 1.5.0    2020-06-23 [1] CRAN (R 4.0.2)
##  SingleCellExperiment * 1.12.0  2020-10-27 [1] Bioconductor
##  sparseMatrixStats     1.2.0    2020-10-27 [1] Bioconductor
##  spatstat.core         2.1-2    2021-04-18 [1] CRAN (R 4.0.2)
##  spatstat.data         2.1-0    2021-03-21 [1] CRAN (R 4.0.2)
##  spatstat.geom         2.1-0    2021-04-15 [1] CRAN (R 4.0.2)
##  spatstat.sparse       2.0-0    2021-03-16 [1] CRAN (R 4.0.2)
##  spatstat.utils        2.1-0    2021-03-15 [1] CRAN (R 4.0.2)
##  statmod               1.4.35   2020-10-19 [1] CRAN (R 4.0.2)
##  stringi               1.7.3    2021-07-16 [1] CRAN (R 4.0.2)
```

```
##  stringr                1.4.0   2019-02-10 [1] CRAN (R 4.0.2)
##  SummarizedExperiment * 1.20.0  2020-10-27 [1] Bioconductor
##  survival               3.2-7   2020-09-28 [1] CRAN (R 4.0.3)
##  tensor                 1.5     2012-05-05 [1] CRAN (R 4.0.2)
##  testthat               3.0.0   2020-10-31 [1] CRAN (R 4.0.2)
##  tibble                 3.1.3   2021-07-23 [1] CRAN (R 4.0.2)
##  tidyr                  1.1.2   2020-08-27 [1] CRAN (R 4.0.2)
##  tidyselect             1.1.1   2021-04-30 [1] CRAN (R 4.0.2)
##  usethis              * 2.0.0   2020-12-10 [1] CRAN (R 4.0.2)
##  utf8                   1.2.2   2021-07-24 [1] CRAN (R 4.0.2)
##  uwot                   0.1.9   2020-11-15 [1] CRAN (R 4.0.2)
##  vctrs                  0.3.8   2021-04-29 [1] CRAN (R 4.0.2)
##  vipor                  0.4.5   2017-03-22 [1] CRAN (R 4.0.2)
##  viridis              * 0.5.1   2018-03-29 [1] CRAN (R 4.0.2)
##  viridisLite          * 0.4.0   2021-04-13 [1] CRAN (R 4.0.2)
##  withr                  2.4.2   2021-04-18 [1] CRAN (R 4.0.2)
##  xfun                   0.19    2020-10-30 [1] CRAN (R 4.0.2)
##  xtable                 1.8-4   2019-04-21 [1] CRAN (R 4.0.2)
##  XVector                0.30.0  2020-10-28 [1] Bioconductor
##  yaml                   2.2.1   2020-02-01 [1] CRAN (R 4.0.2)
##  zlibbioc               1.36.0  2020-10-28 [1] Bioconductor
##  zoo                    1.8-8   2020-05-02 [1] CRAN (R 4.0.2)
##
## [1] /Library/Frameworks/R.framework/Versions/4.0/Resources/library
```