single_cell_course_sib

IM

11/8/2021

SIB course

#https://sib-swiss.github.io/single-cell-training/2021.11/day1/introduction_scrnaseq/

We will work on the GBM dataset that is: Transcriptomes of human glioblastoma multiforme cells, generated by 10x genomics. This will be our most used dataset, we will use it throughout the course. Source: 10x datasets

We will analyse one sample

Day1 Seurat typical analysis

We will analyse differentially expressed genes within the same sample. Different cells.

We start the analysis from the output made by the Illumina sequencing of the 10x libraries?. First step is to produce the three files by using cellranger pipeline

use filtered matrix

Empty droplets are needed to control for ambient RNA.

Load data with seurat package

show firt 30 cells of 3 genes. we noticed allot of empty spaces.

```
# load 10x data with Seurat package

gbm.data <- Seurat::Read10X(data.dir = "C://Users/imateusg/Documents/WW_courses/02_SingleCELL_SIB2021/s

gbm.data[c("PECAM1", "CD8A", "TSPAN1"), 1:30]
```

```
## 3 x 30 sparse Matrix of class "dgCMatrix"

##    [[ suppressing 30 column names 'AAACCCAAGGCGATAC-1', 'AAACCCACAAGTCCCG-1', 'AAACCCACAGATGCGA-1' .

##
## PECAM1 . . 1 . . . 1 . . . . . . . . . . . 2 . . . . . . . . . . . . .
## CD8A   . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## TSPAN1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 . . .
```

Generate Seurat Object including only genes that contain at least 3 cell and that expression is detected for more than 100 genes

```
gbm <- Seurat::CreateSeuratObject(counts = gbm.data,
                                  project = "gbm",
                                  min.cells = 3,
                                  min.features = 100)
gbm
```

```
## An object of class Seurat
## 24363 features across 5553 samples within 1 assay
## Active assay: RNA (24363 features, 0 variable features)
```

We found that the Run contained 24363 different features across 5553 cells

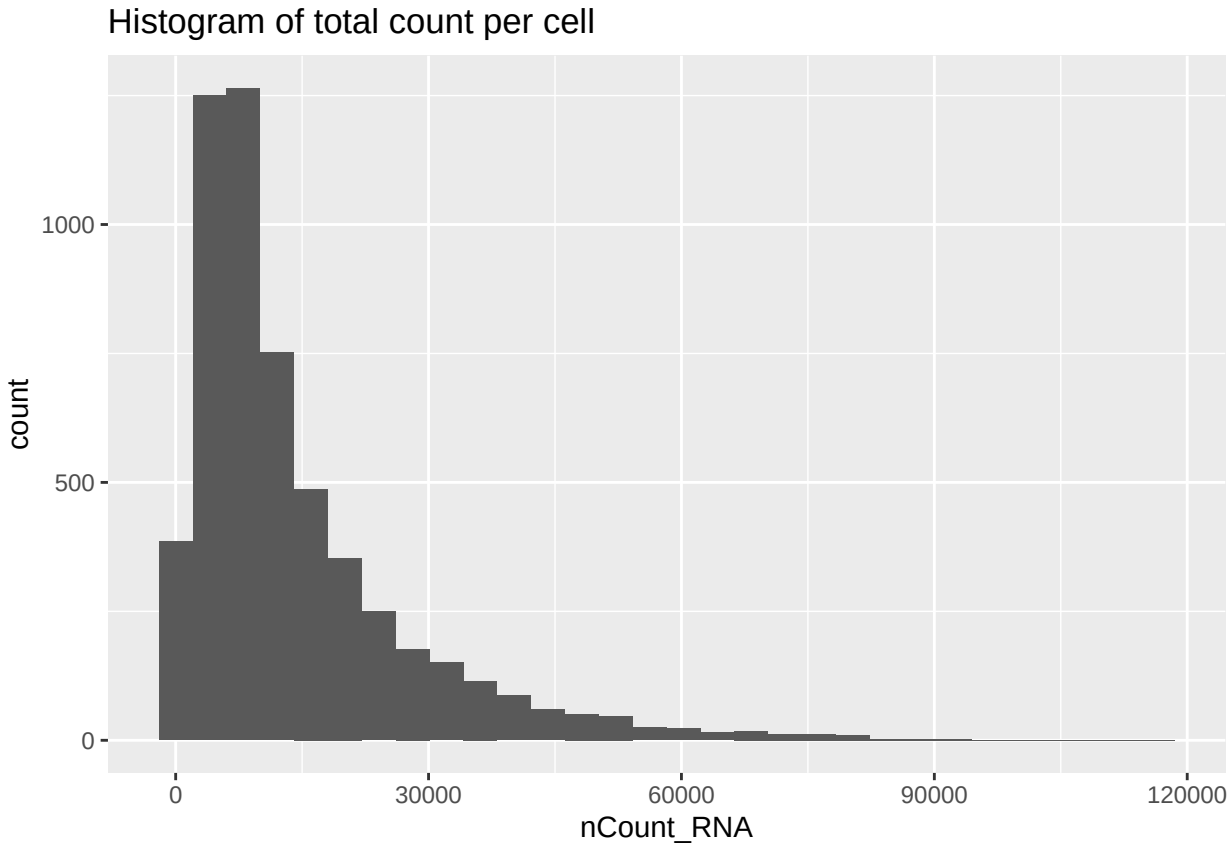
Inspect gbm@meta.data object. we have two columns. nCount which defines total number of counts, and nFeature which define total number of genes.

```
head(gbm@meta.data)
```

```
##                     orig.ident nCount_RNA nFeature_RNA
## AAACCCAAGGCGATAC-1        gbm       2225          815
## AAACCCACAAGTCCCG-1        gbm      17882         4760
## AAACCCACAGATGCGA-1        gbm       8172         2319
## AAACCCACAGGTGAGT-1        gbm       9057         3395
## AAACCCAGTCTTGCGG-1        gbm       5612         2716
## AAACCCATCGATAACC-1        gbm       6254         1659
```

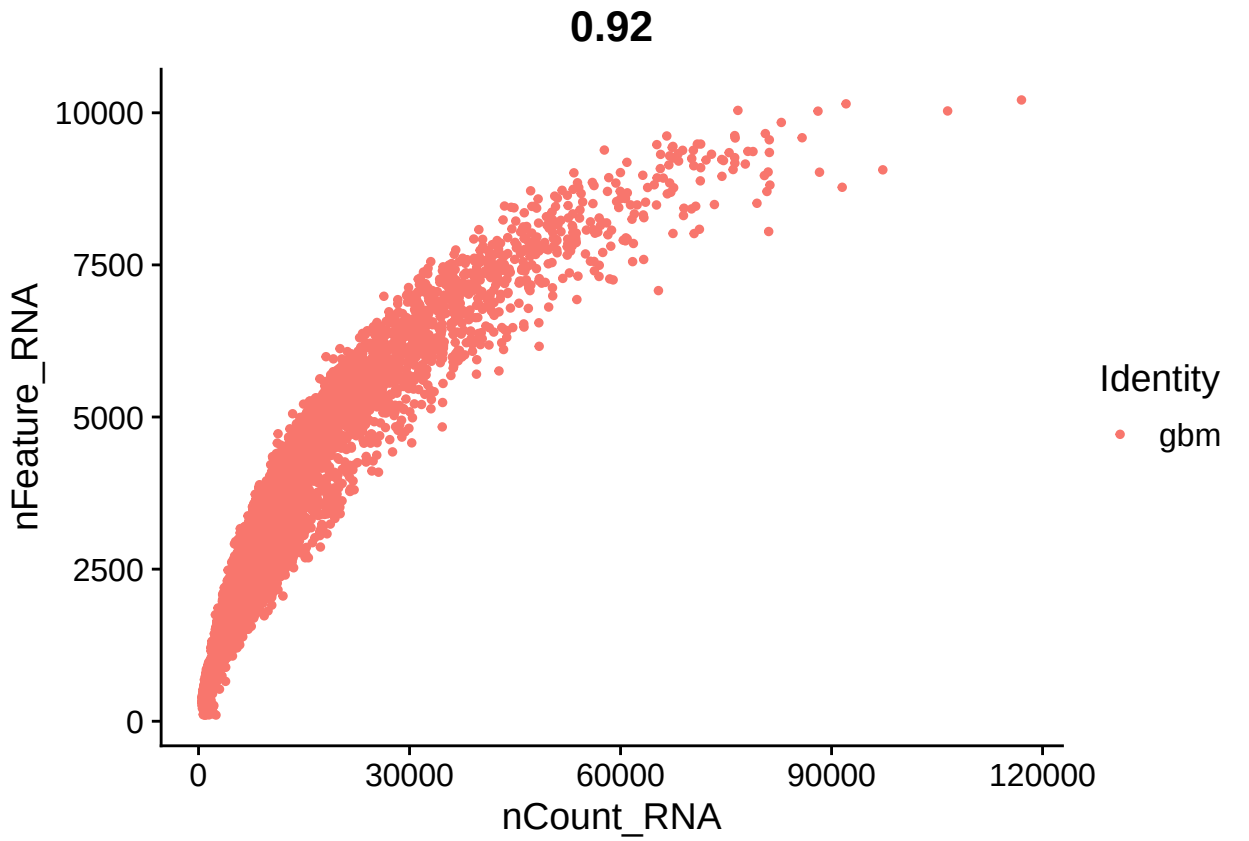
```
ggplot(gbm@meta.data, aes(x=nCount_RNA)) + geom_histogram() + labs( title="Histogram of total count per
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



or you can plot how the two variables correlate via Seurat

```
Seurat::FeatureScatter(gbm, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
```

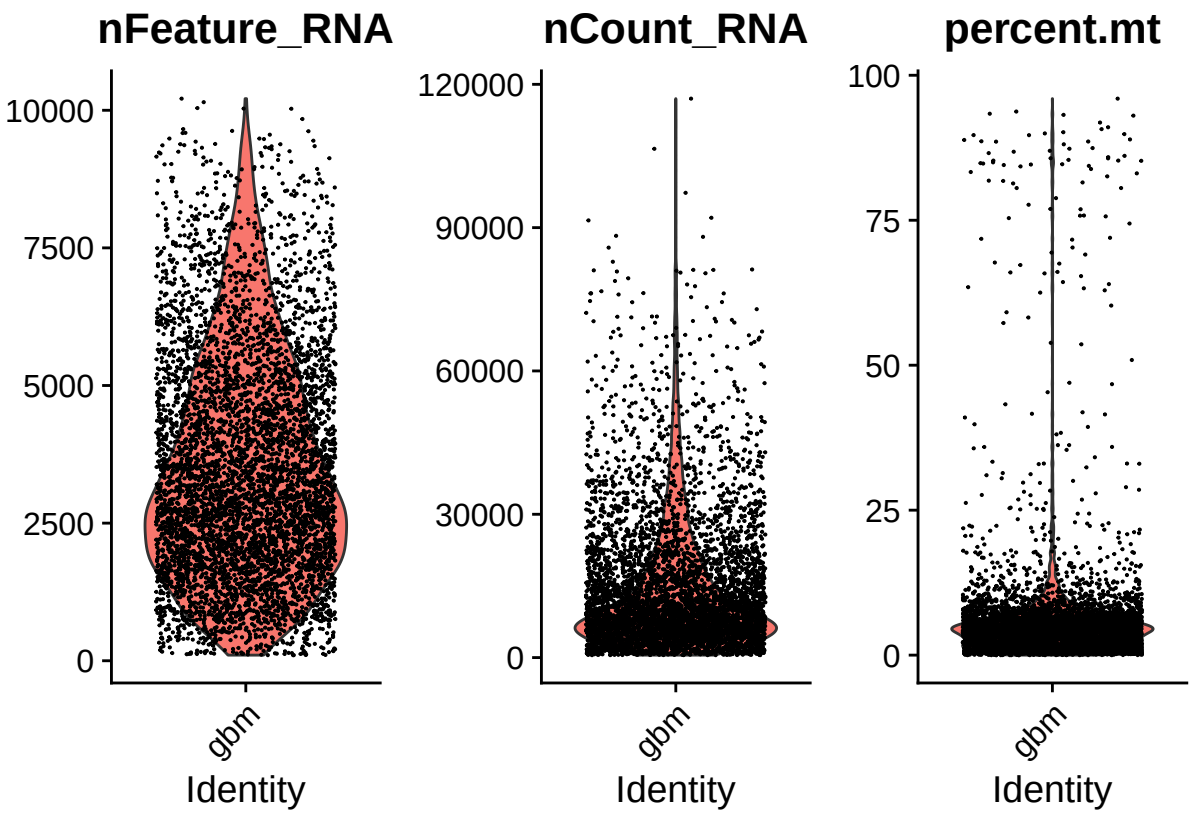


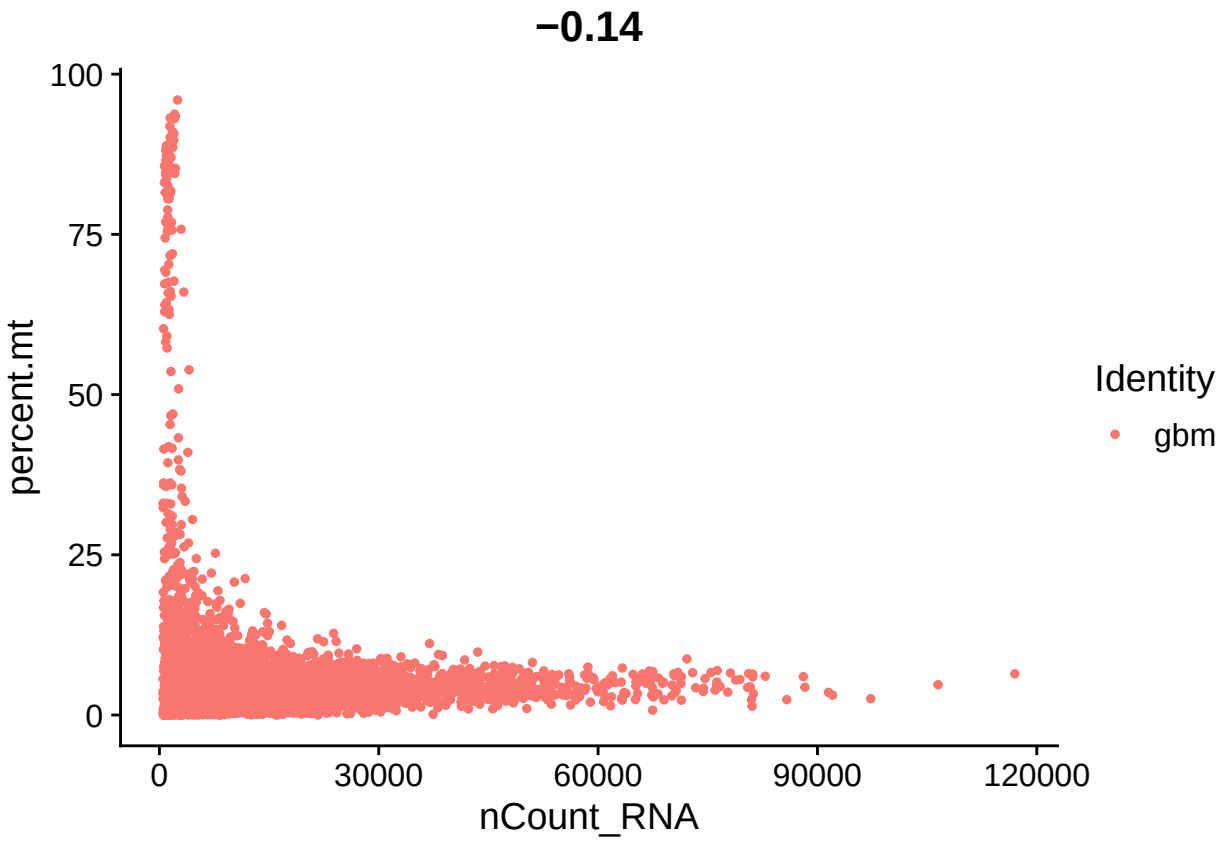
Basic Quality Control

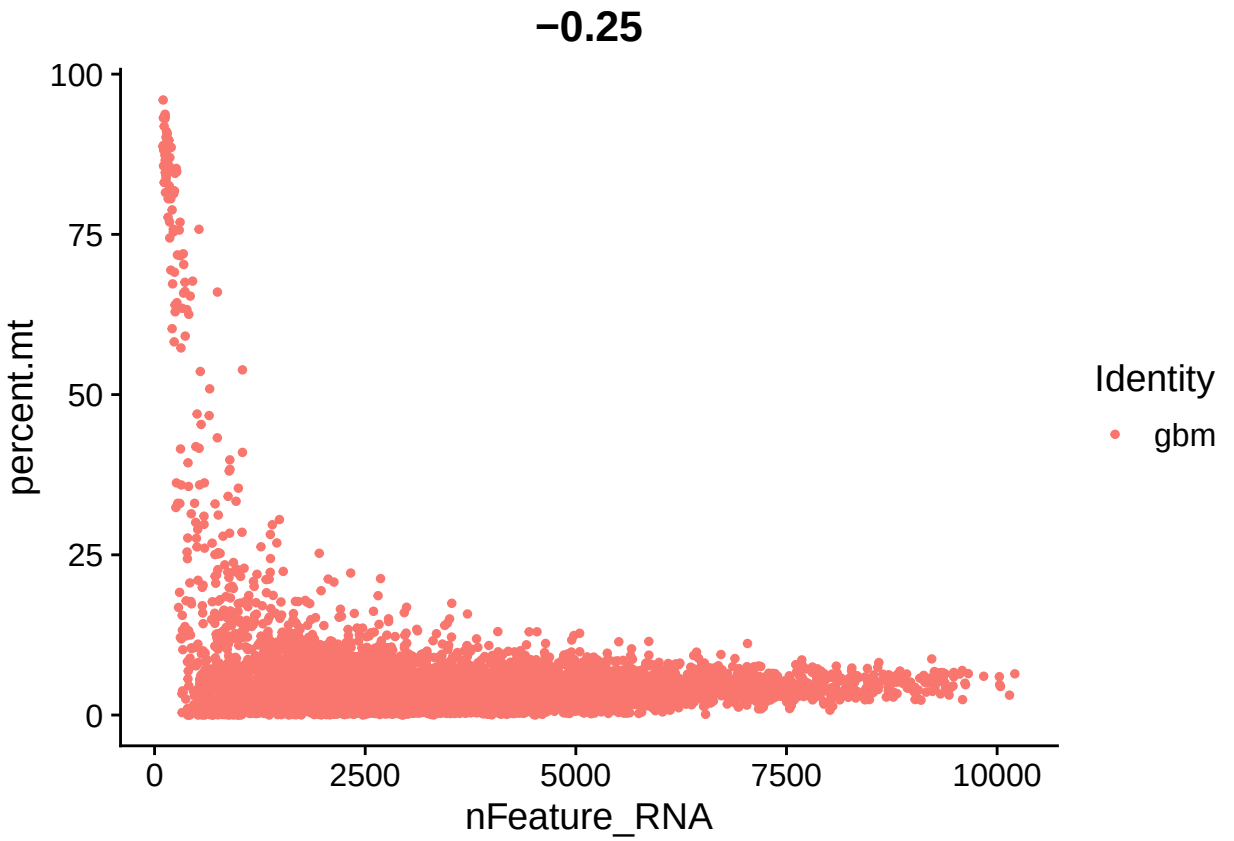
A high number of UMIs originating from mitochondrial genes can point to dying cells. In order to calculate the percentage of UMIs coming from mitochondrial genes for each cell, we use the function PercentageFeatureSet. In our count matrix, the names of mitochondrial genes all start with MT-, so we can use that pattern to search these genes:

We should define thresholds min genes, counts mitochondria based on different thresholds, that could be dataset specific.do not use the same thresholds as other papers. Each dataset should have its own thresholds. A way to make this more reproducible, it could be to make permutations of different thresholds and choose the thresholds that produce similar results.

```
##                     orig.ident nCount_RNA nFeature_RNA percent.mt
## AAACCCAAGGCGATAC-1        gbm       2225          815  4.7191011
## AAACCCACAAGTCCCG-1        gbm      17882         4760  1.5098982
## AAACCCACAGATGCGA-1        gbm       8172         2319  6.4855605
## AAACCCACAGGTGAGT-1        gbm       9057         3395  3.1577785
## AAACCCAGTCTTGCGG-1        gbm       5612         2716  0.4276550
## AAACCCATCGATAACC-1        gbm       6254         1659  0.7195395
```


nFeature_RNA
10000
7500
5000
2500
0
gbm
Identity
nCount_RNA
120000
90000
60000
30000
0
gbm
Identity
percent.mt
100
75
50
25
0
gbm
Identity


−0.14
100
75
50
25
0
percent.mt
0
30000
60000
90000
120000
nCount_RNA
Identity
gbm




```
## [1] 24363  5091
```

We found that 5553-5091(462) cells where filtered out

Normalization

After removing unwanted cells from the dataset, the next step is to normalize the data. By default, Seurat employs a global-scaling normalization method "LogNormalize" that normalizes the feature expression measurements for each cell by the total expression, multiplies this by a scale factor (10,000 by default), and log-transforms the result. Normalized values are stored in the "RNA" slot of the gbm object.

Variable features

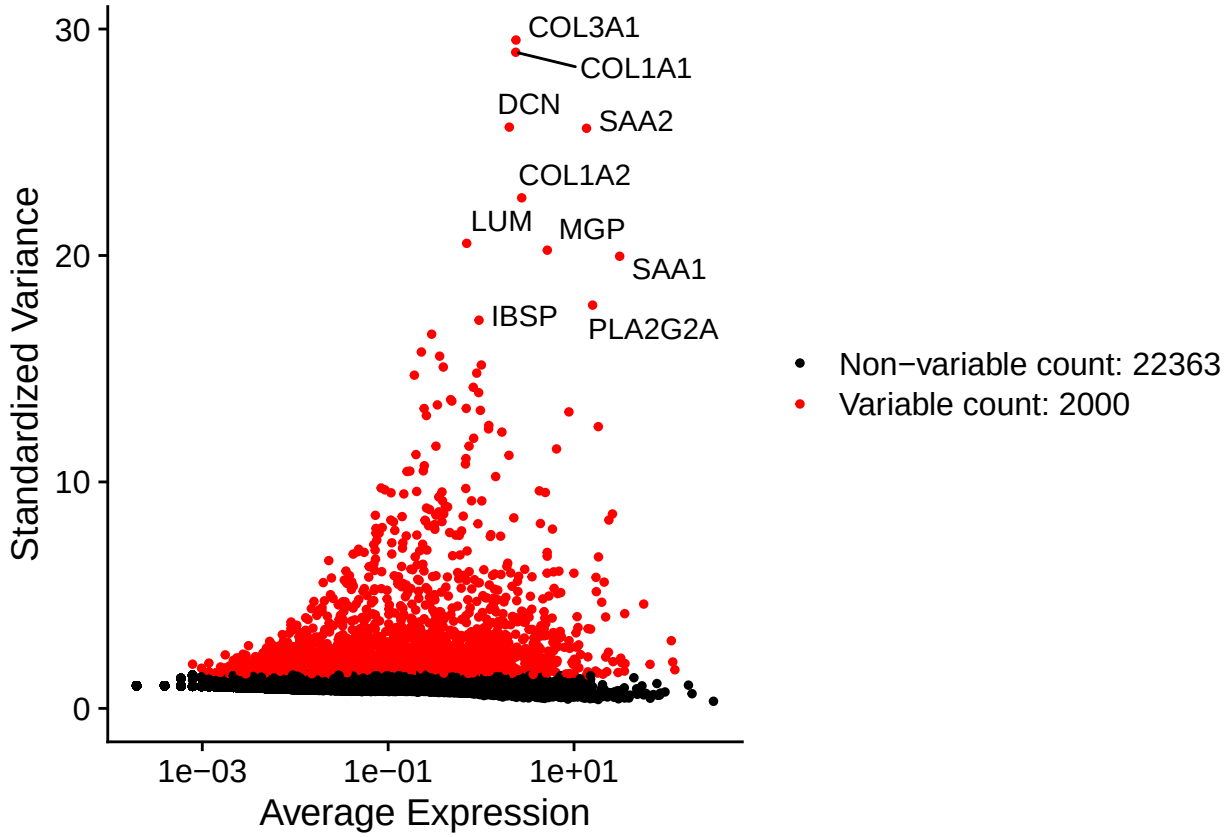
We next calculate a subset of features that exhibit high cell-to-cell variation in the dataset (i.e, they are highly expressed in some cells, and lowly expressed in others). Focusing on these genes in downstream analysis helps to highlight biological signal in single-cell datasets. The procedure in Seurat models the mean-variance relationship inherent in single-cell data, and is implemented in the FindVariableFeatures() function. By default, 2,000 genes (features) per dataset are returned and these will be used in downstream analysis, like PCA.

```
##  [1] "COL3A1"  "COL1A1"  "DCN"     "SAA2"    "COL1A2"  "LUM"     "MGP"
##  [8] "SAA1"    "PLA2G2A" "IBSP"
```

```
## When using repel, set xnudge and ynudge to 0 for optimal results
```

```
## Warning: Transformation introduced infinite values in continuous x-axis
```

```
## Warning: Removed 10 rows containing missing values (geom_point).
```



Scaling

Next, we apply scaling, a linear transformation that is a standard pre-processing step prior to dimensional reduction techniques like PCA. The ScaleData() function

```
shifts the expression of each gene, so that the mean expression across cells is 0
scales the expression of each gene, so that the variance across cells is 1
```

This step gives equal weight in downstream analyses, so that highly-expressed genes do not dominate. The results of this are stored in gbm\protect\T1\textdollarRNA@scale.data This also can be done with Seurat::SCTransform

```
## Centering and scaling data matrix
```

```
##            AAACCCAAGGCGATAC-1 AAACCCACAAGTCCCG-1 AAACCCACAGATGCGA-1
## AL627309.1        -0.04520560        -0.04520560        -0.04520560
## AL627309.5        -0.06113671        -0.06113671        -0.06113671
## AP006222.2        -0.02994402        -0.02994402        -0.02994402
## LINC01409         -0.20815793        -0.20815793        -0.20815793
```

```
## FAM87B            -0.07277531        -0.07277531        -0.07277531
##            AAACCCACAGGTGAGT-1 AAACCCAGTCTTGCGG-1 AAACCCATCGATAACC-1
## AL627309.1        -0.04520560        -0.04520560        -0.04520560
## AL627309.5        -0.06113671        -0.06113671        -0.06113671
## AP006222.2        -0.02994402        -0.02994402        -0.02994402
## LINC01409          4.75584813        -0.20815793        -0.20815793
## FAM87B            -0.07277531        -0.07277531        -0.07277531
##            AAACGAAAGACTCAAA-1 AAACGAAAGTAATCCC-1 AAACGAAAGTATGCAA-1
## AL627309.1        -0.04520560        -0.04520560        -0.04520560
## AL627309.5        -0.06113671        -0.06113671        -0.06113671
## AP006222.2        -0.02994402        -0.02994402        -0.02994402
## LINC01409         -0.20815793        -0.20815793        -0.20815793
## FAM87B            -0.07277531        -0.07277531        -0.07277531
##            AAACGAACACCACATA-1
## AL627309.1        -0.04520560
## AL627309.5        -0.06113671
## AP006222.2        -0.02994402
## LINC01409         -0.20815793
## FAM87B            -0.07277531
```

Quality control

In this part we will use package seurat to learn about:

1. Assign cell cycle phases to a single cell dataset
2. Use the package scater evaluate cell quality based on reads originating from: mitochondrial genes ribosomal genes dissociation-related genes
3. Evaluate confounding effects on expression data by analyzing the explained variance

Cell cycle analysis

Cells could be captured in different cycle phases. From human cells, we know from databases, that some genes are expressed in different cell cycles.

Seurat::cc.genes.updated.2019 contains genes classified by cell cycle we will add this info to the gbm3 dataset

We can visualize the distribution of cell cycle markers (genes characteristic from a cell cycle)

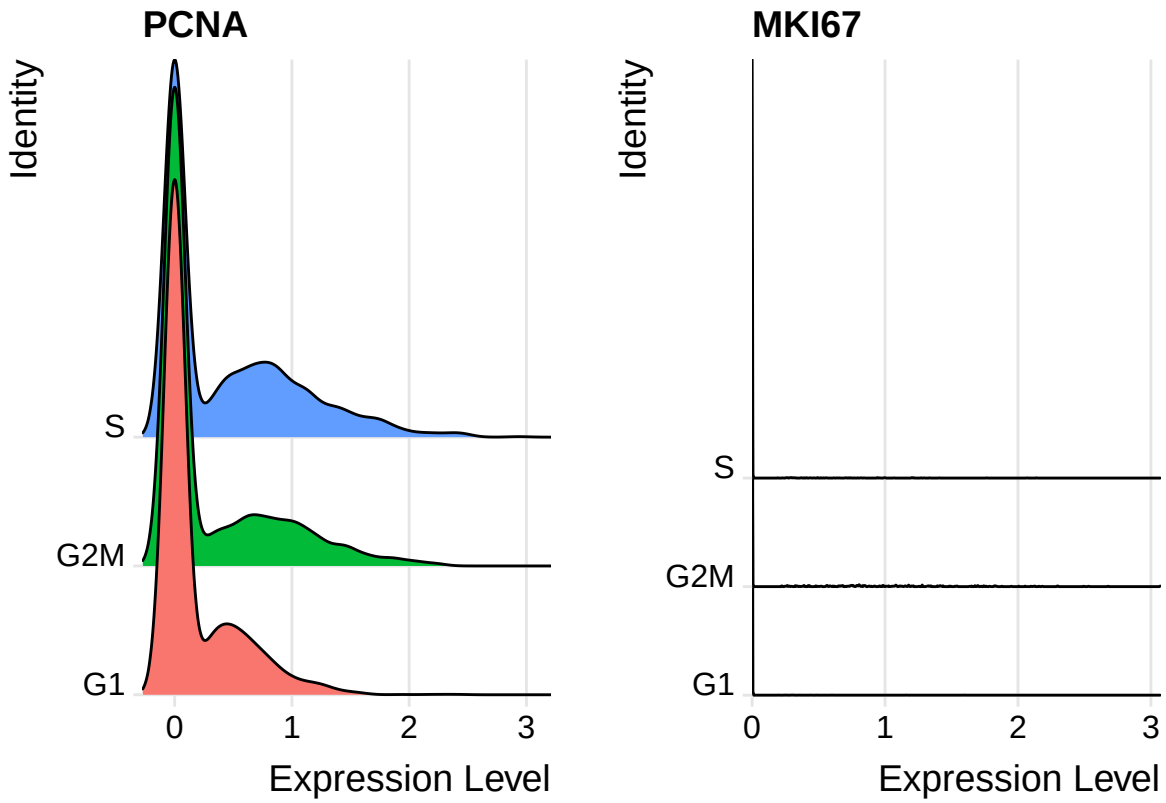
```
s.genes <- Seurat::cc.genes.updated.2019$s.genes
g2m.genes <- Seurat::cc.genes.updated.2019$g2m.genes

gbm <- Seurat::CellCycleScoring(gbm,
                                s.features = s.genes,
                                g2m.features = g2m.genes)
table(gbm$Phase)
```

```
##
##   G1  G2M    S
## 2887  711 1493
```

```
#
Seurat::RidgePlot(gbm, features = c("PCNA", "MKI67"),
          group.by = "Phase",
          ncol = 2)
```

```
## Picking joint bandwidth of 0.0907
```

```
## Picking joint bandwidth of 2.84e-06
```



Quality control with Scater

Scater includes different types of quality controls: QC and filtering of cells QC and filtering of features (genes) QC of experimental variables

the scater package as well as other bioconductor packages, rely on an object of the class SingleCellExperiment. So first we get the counts with Seurat. Then we use the function SingleCellExperient to import the cts and metadata

```
cts <- Seurat::GetAssayData(gbm, slot = "counts")
gbm_sce <- SingleCellExperiment::SingleCellExperiment(
  assays = list(counts = cts),
  colData = gbm@meta.data,
  rowData = rownames(gbm)
)
```

We can now find variables in the data set that can be used as explanatory variables. These variables can explain a lot of the variance observed in the cells.

We can import info about mitochondrial gene expression. But we can also run a quality check on stressed cells. Genes related to stress are known and could cluster together as a false type of cell. This step is known as dissociation protocol-related gene expression. and are available in data/dissocation_genes.txt

We then compute the info about each type to the summarized experiment. We work now with Scuttle package

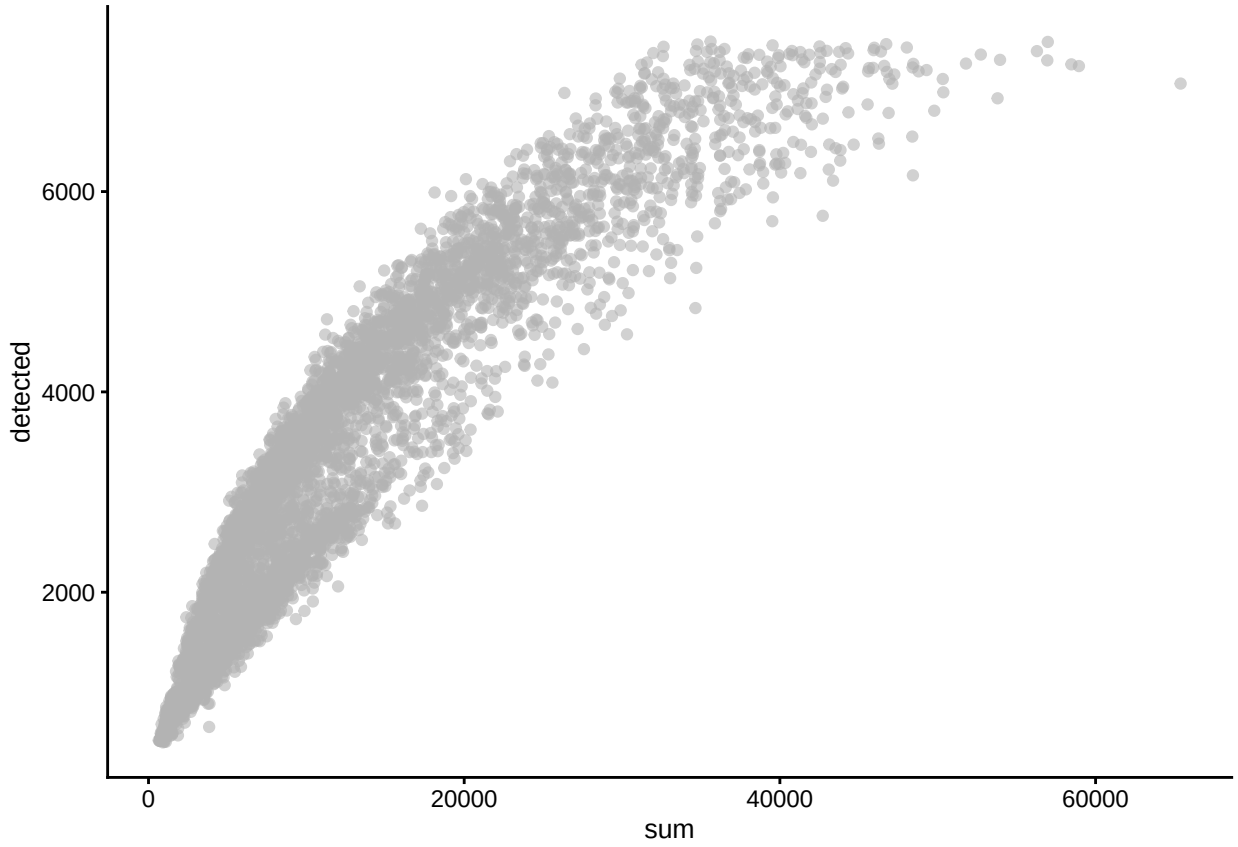
```
dissoc_genes <- readLines("C://Users/imateusg/Documents/WW_courses/02_SingleCELL_SIB2021/single_cell_co
ribo_genes <- rownames(gbm)[grep(pattern = "^RP[S|L]", rownames(gbm), perl = T)]
mito_genes <- rownames(gbm)[grep(pattern = "^MT-", rownames(gbm))]
#
gbm_sce <- scuttle::addPerCellQC(gbm_sce,
                         subsets=list(mito_genes=which(rownames(gbm_sce) %in% mito_genes),
                                      dissoc_genes=which(rownames(gbm_sce) %in% dissoc_genes),
                                      ribo_genes=which(rownames(gbm_sce) %in% ribo_genes)))
#  compute a number of quality control metrics for each cell and feature (i.e gene)
SingleCellExperiment::colData(gbm_sce)
```

```
## DataFrame with 5091 rows and 19 columns
##                     orig.ident nCount_RNA nFeature_RNA percent.mt     S.Score
##                       <factor>  <numeric>    <integer>  <numeric>   <numeric>
## AAACCCAAGGCGATAC-1        gbm       2225          815   4.719101  -0.0231030
## AAACCCACAAGTCCCG-1        gbm      17882         4760   1.509898  -0.0531619
## AAACCCACAGATGCGA-1        gbm       8172         2319   6.485560  -0.0485556
## AAACCCACAGGTGAGT-1        gbm       9057         3395   3.157779   0.0362228
## AAACCCAGTCTTGCGG-1        gbm       5612         2716   0.427655   0.0887003
## ...                       ...        ...          ...        ...         ...
## TTTGGTTTCGCTACAA-1        gbm      10403         3965   0.278766  0.04996560
## TTTGTTGAGGGTACGT-1        gbm       5574         1591   3.229279  0.00958146
## TTTGTTGCATCCGGTG-1        gbm       6387         1827   5.182402 -0.02136678
## TTTGTTGGTCTACACA-1        gbm      25620         5583   5.987510 -0.01650486
## TTTGTTGGTTGGTACT-1        gbm       8555         2325   7.212157 -0.04682965
##                      G2M.Score       Phase       sum  detected
##                      <numeric> <character> <numeric> <integer>
## AAACCCAAGGCGATAC-1 -0.0624157          G1      2225       815
## AAACCCACAAGTCCCG-1 -0.0438394          G1     17882      4760
## AAACCCACAGATGCGA-1 -0.1084447          G1      8172      2319
## AAACCCACAGGTGAGT-1 -0.0725644           S      9057      3395
## AAACCCAGTCTTGCGG-1 -0.0889893           S      5612      2716
## ...                       ...         ...       ...       ...
## TTTGGTTTCGCTACAA-1 -0.0771751           S     10403      3965
## TTTGTTGAGGGTACGT-1 -0.0727691           S      5574      1591
## TTTGTTGCATCCGGTG-1 -0.0481192          G1      6387      1827
## TTTGTTGGTCTACACA-1 -0.0397462          G1     25620      5583
## TTTGTTGGTTGGTACT-1 -0.0468854          G1      8555      2325
##                     subsets_mito_genes_sum subsets_mito_genes_detected
##                                  <numeric>                   <integer>
## AAACCCAAGGCGATAC-1                    105                          11
## AAACCCACAAGTCCCG-1                    270                          13
## AAACCCACAGATGCGA-1                    530                          12
## AAACCCACAGGTGAGT-1                    286                          12
## AAACCCAGTCTTGCGG-1                     24                          10
```

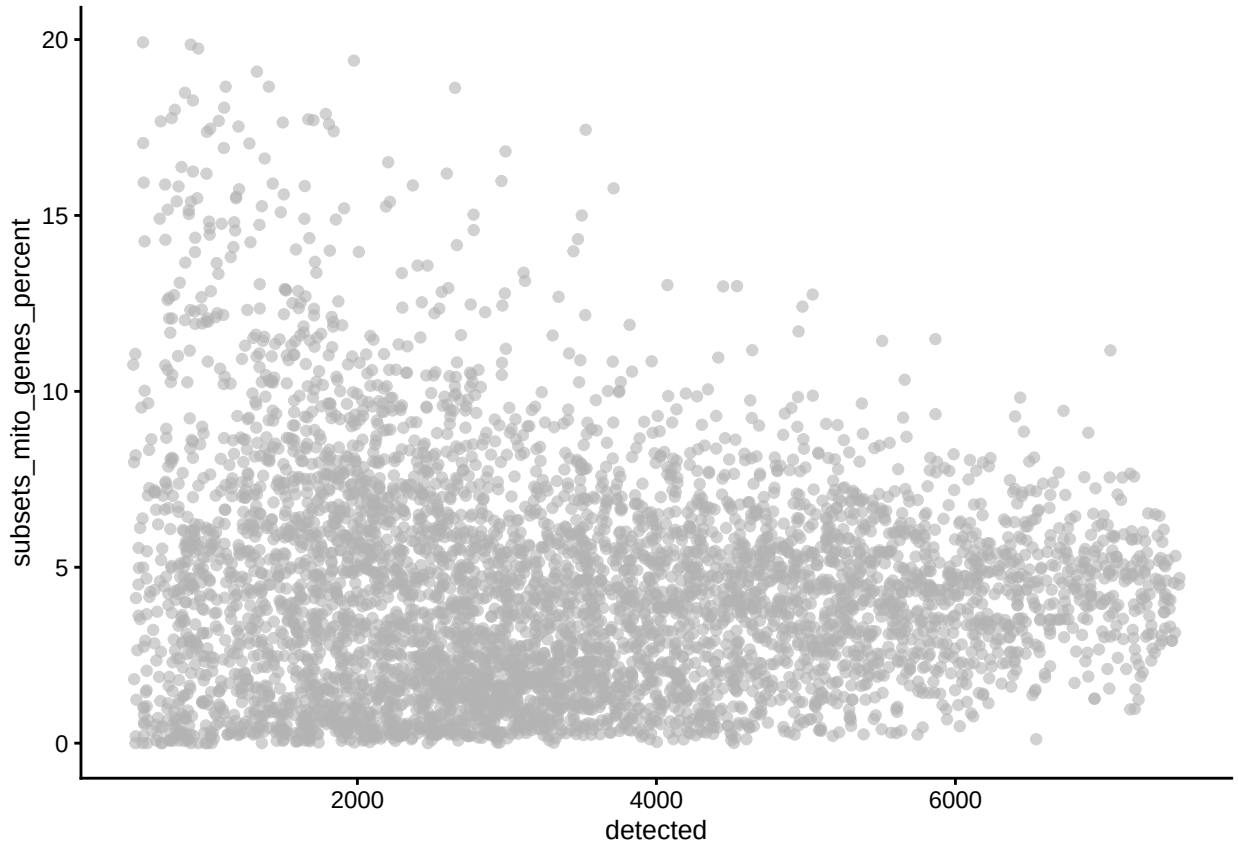
```
## ...                                  ...                          ...
## TTTGGTTTCGCTACAA-1                    29                           9
## TTTGTTGAGGGTACGT-1                   180                          12
## TTTGTTGCATCCGGTG-1                   331                          11
## TTTGTTGGTCTACACA-1                  1534                          13
## TTTGTTGGTTGGTACT-1                   617                          12
##                     subsets_mito_genes_percent subsets_dissoc_genes_sum
##                                      <numeric>                <numeric>
## AAACCCAAGGCGATAC-1                    4.719101                       34
## AAACCCACAAGTCCCG-1                    1.509898                      460
## AAACCCACAGATGCGA-1                    6.485560                      695
## AAACCCACAGGTGAGT-1                    3.157779                      778
## AAACCCAGTCTTGCGG-1                    0.427655                      262
## ...                                        ...                      ...
## TTTGGTTTCGCTACAA-1                    0.278766                      595
## TTTGTTGAGGGTACGT-1                    3.229279                      687
## TTTGTTGCATCCGGTG-1                    5.182402                      581
## TTTGTTGGTCTACACA-1                    5.987510                      730
## TTTGTTGGTTGGTACT-1                    7.212157                      809
##                     subsets_dissoc_genes_detected subsets_dissoc_genes_percent
##                                         <integer>                   <numeric>
## AAACCCAAGGCGATAC-1                             23                     1.52809
## AAACCCACAAGTCCCG-1                             66                     2.57242
## AAACCCACAGATGCGA-1                             79                     8.50465
## AAACCCACAGGTGAGT-1                             73                     8.59004
## AAACCCAGTCTTGCGG-1                             55                     4.66857
## ...                                           ...                         ...
## TTTGGTTTCGCTACAA-1                             62                     5.71950
## TTTGTTGAGGGTACGT-1                             57                    12.32508
## TTTGTTGCATCCGGTG-1                             80                     9.09660
## TTTGTTGGTCTACACA-1                             76                     2.84934
## TTTGTTGGTTGGTACT-1                             80                     9.45646
##                     subsets_ribo_genes_sum subsets_ribo_genes_detected
##                                  <numeric>                  <integer>
## AAACCCAAGGCGATAC-1                    1006                         80
## AAACCCACAAGTCCCG-1                    1388                         84
## AAACCCACAGATGCGA-1                     661                         81
## AAACCCACAGGTGAGT-1                     479                         83
## AAACCCAGTCTTGCGG-1                     468                         80
## ...                                    ...                        ...
## TTTGGTTTCGCTACAA-1                     492                         80
## TTTGTTGAGGGTACGT-1                    1378                         80
## TTTGTTGCATCCGGTG-1                     391                         78
## TTTGTTGGTCTACACA-1                    1561                         89
## TTTGTTGGTTGGTACT-1                     679                         78
##                     subsets_ribo_genes_percent     total
##                                      <numeric> <numeric>
## AAACCCAAGGCGATAC-1                   45.21348      2225
## AAACCCACAAGTCCCG-1                    7.76200     17882
## AAACCCACAGATGCGA-1                    8.08860      8172
## AAACCCACAGGTGAGT-1                    5.28873      9057
## AAACCCAGTCTTGCGG-1                    8.33927      5612
## ...                                       ...       ...
## TTTGGTTTCGCTACAA-1                    4.72940     10403
```

```
## TTTGTTGAGGGTACGT-1                  24.72192      5574
## TTTGTTGCATCCGGTG-1                   6.12181      6387
## TTTGTTGGTCTACACA-1                   6.09290     25620
## TTTGTTGGTTGGTACT-1                   7.93688      8555
```

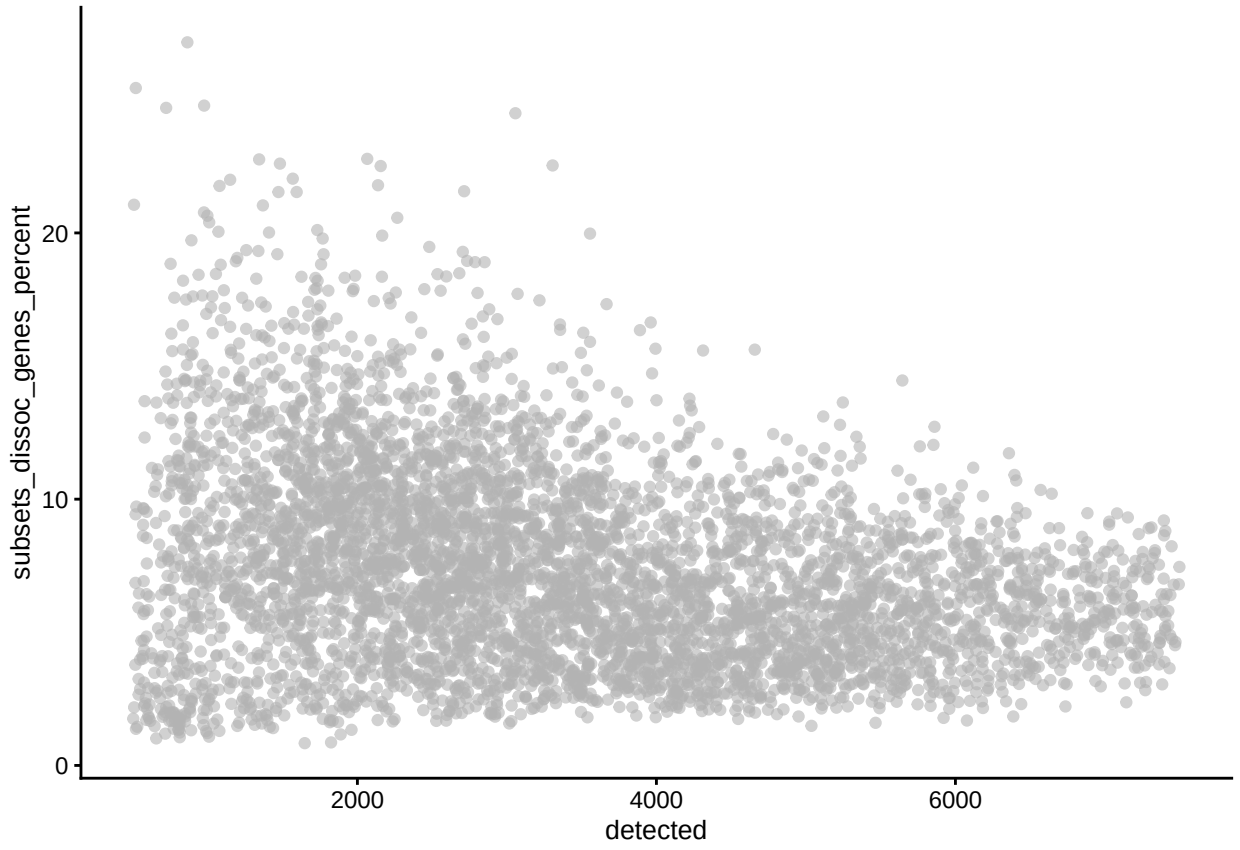
```
# Rarefaction curve
scater::plotColData(gbm_sce, x = "sum", y="detected")
```



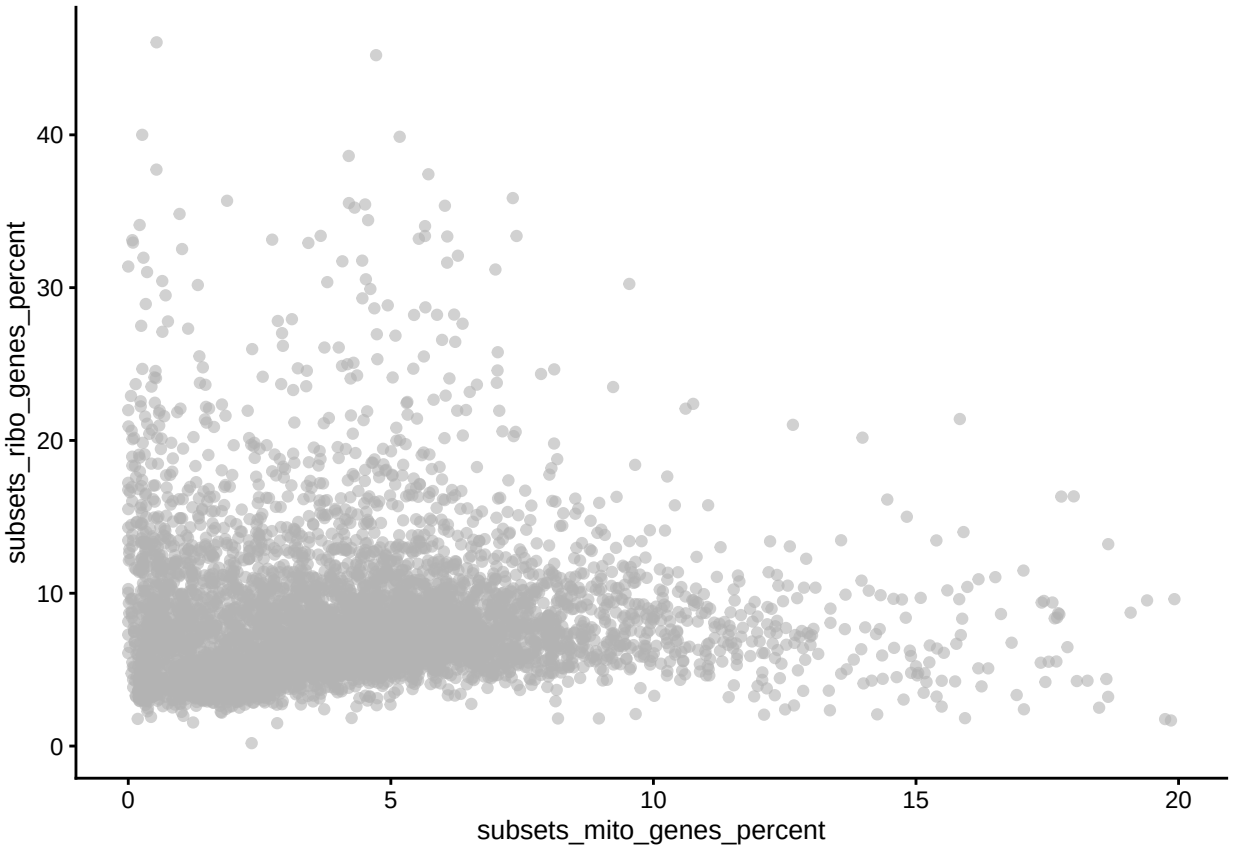

```
#percentage of mitochondrial genes
scater::plotColData(gbm_sce, x = "detected", y="subsets_mito_genes_percent")
```




```
#percentage dissociation genes
scater::plotColData(gbm_sce, x = "detected", y="subsets_dissoc_genes_percent")
```




```
#correlation mitochondrial genes and ribo genes
scater::plotColData(gbm_sce, x = "subsets_mito_genes_percent", y="subsets_ribo_genes_percent")
```



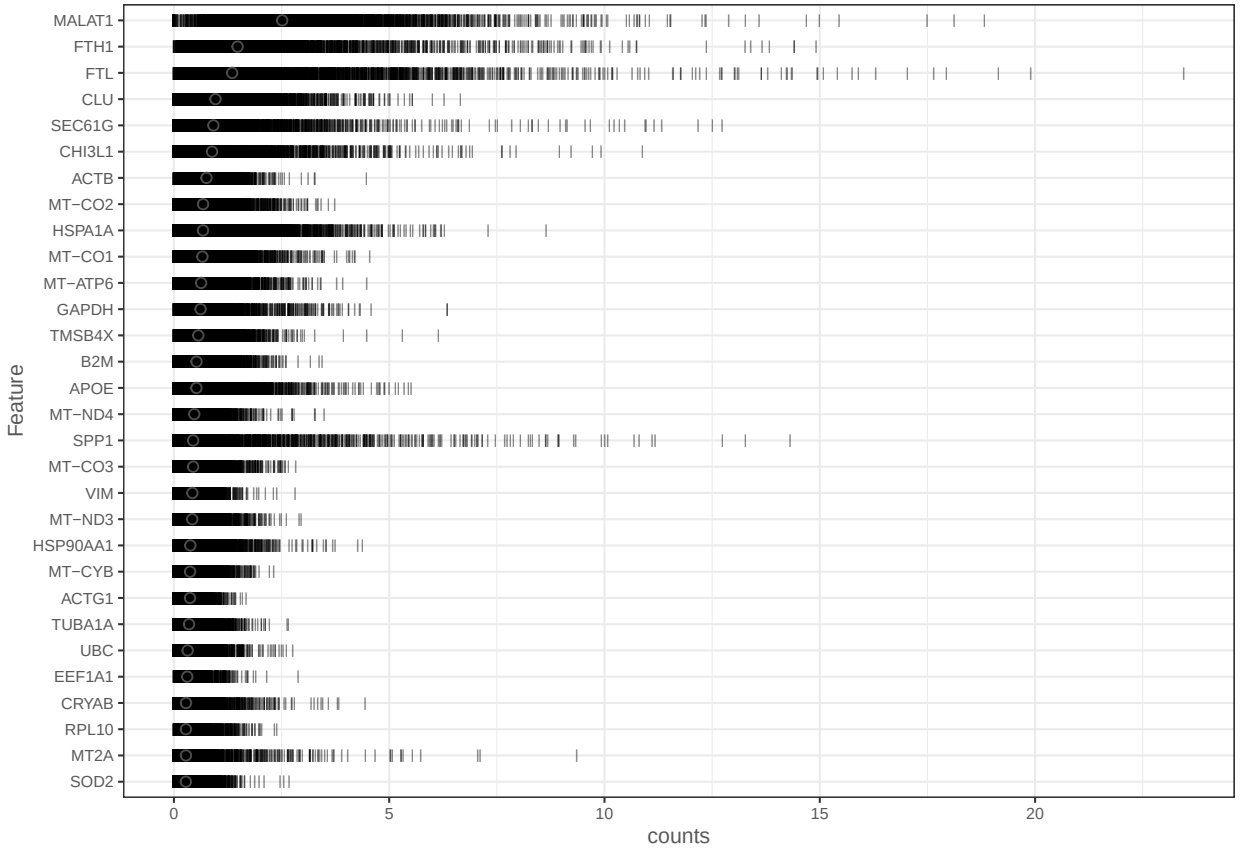
Highly expressed genes

Identification of highly expressed genes. these are the ones that we want to get rid off.

On the gene level, we can look at a plot that shows the top (by default 50) most-expressed genes. Each row in the plot corresponds to a gene; each bar corresponds to the expression of a gene in a single cell; and the circle indicates the median expression of each gene, with which genes are sorted. We expect to see the "usual suspects", i.e., mitochondrial genes, actin, ribosomal protein, MALAT1. If used, few spike-in transcripts may also be present here, though if all of the spike-ins are in the top 50, it suggests that too much spike-in RNA was added. A large number of pseudo-genes or predicted genes may indicate problems with alignment.

visualize high expressed genes

```
scater::plotHighestExprs(gbm_sce, exprs_values = "counts", n = 30)
```



Find explanatory variables

Variable-level metrics are computed by the getVarianceExplained() function (after normalization, see below). This calculates the percentage of variance of each gene's expression that is explained by each variable in the colData of the SingleCellExperiment object. We can then use this to determine which experimental factors are contributing most to the variance in expression. This is useful for diagnosing batch effects or to quickly verify that a treatment has an effect.

First, computing variance explained on the log-counts, so that the statistics reflect changes in relative expression.

```
gbm_sce <- scater::logNormCounts(gbm_sce)  # alternative to Seurat's normalization here using scater
```

In the gbm dataset, we only have 1 patient, so we cannot calculate the effect of experimental variables like sex or donor id, but in case we would have several variables, here is the method with the cell cycle phase as example: Look at the percentage of variance explained by phase or detected expression.

Variance due to the cell type (Detected)

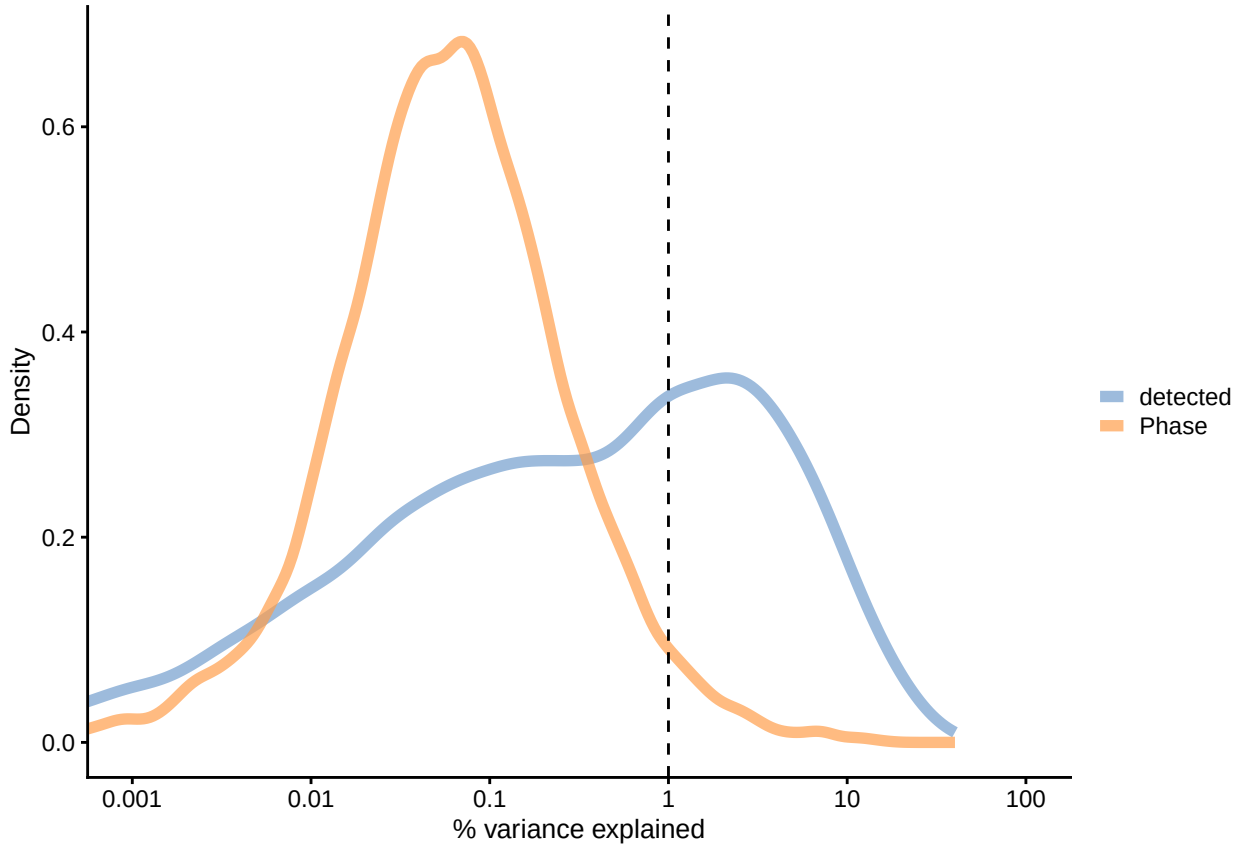
```
vars <- scater::getVarianceExplained(gbm_sce,
                                     variables = c("detected","Phase") )
head(vars) # show how each variable is  explained for each gene
```

```
##              detected      Phase
```

```
## AL627309.1 0.002821331 0.03755442
## AL627309.5 0.083208606 0.01721037
## AP006222.2 0.018017527 0.01993929
## LINC01409  0.234686147 0.07512823
## FAM87B     0.144580624 0.01490441
## LINC01128  0.361184234 0.05493752
```

```
scater::plotExplanatoryVariables(vars)
```

```
## Warning: Removed 20 rows containing non-finite values (stat_density).
```



A distribution of percentage variance explained by each gene is shown, and can indicate whether one or the other experimental variable has high contribution to the variance in the data:

The cell cycle explain around 1% of variance so it should be interesting to take this into account.

If we think that the cell cycling has an effect on the analysis, and if we want to regress out this effect so that cycling cells are integrated into the rest of the cells and not clustering apart anymore, we can regress out the cell cycling phase at the moment of scaling the data using ScaleData. This might be slow to compute.

do not run

```
#gbm_cc <- Seurat::ScaleData(gbm, vars.to.regress = c("S.Score", "G2M.Score"))
```

save work with RDS file

```
#saveRDS(gbm, "C://Users/imateusg/Documents/WW_courses/02_SingleCELL_SIB2021/gbm_day1.rds")
```

Day 2 . Dimensionality reduction

Once the data is normalized, scaled and variable features have been identified, we can start to reduce the dimensionality of the data. For the PCA, by default, only the previously determined variable features are used as input, but can be defined using features argument if you wish to specify a vector of genes. The PCA will only be run on the variable features, that you can check with VariableFeatures(gbm).

open work space with RDS file

```
#openRDS(gbm, "C://Users/imateusg/Documents/WW_courses/02_SingleCELL_SIB2021/gbm_day1.rds")
```

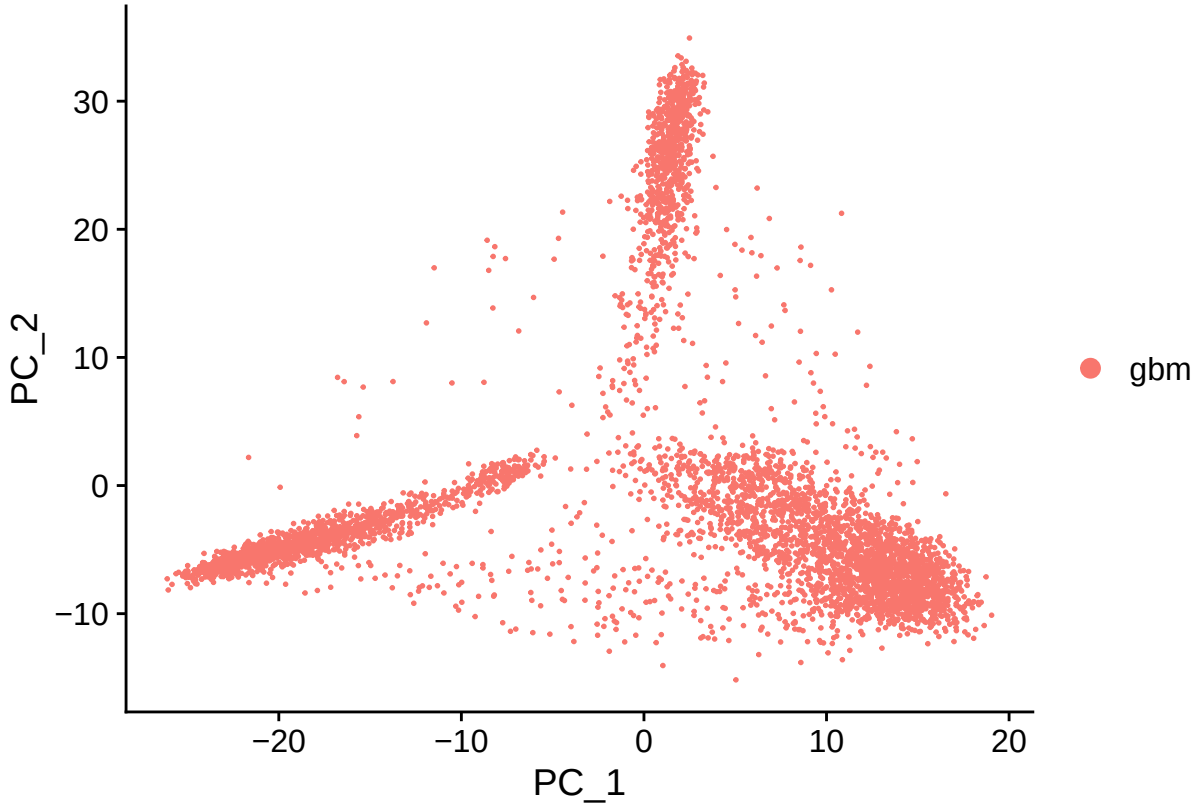
Perform a pca

```
gbm <- Seurat::RunPCA(gbm)
```

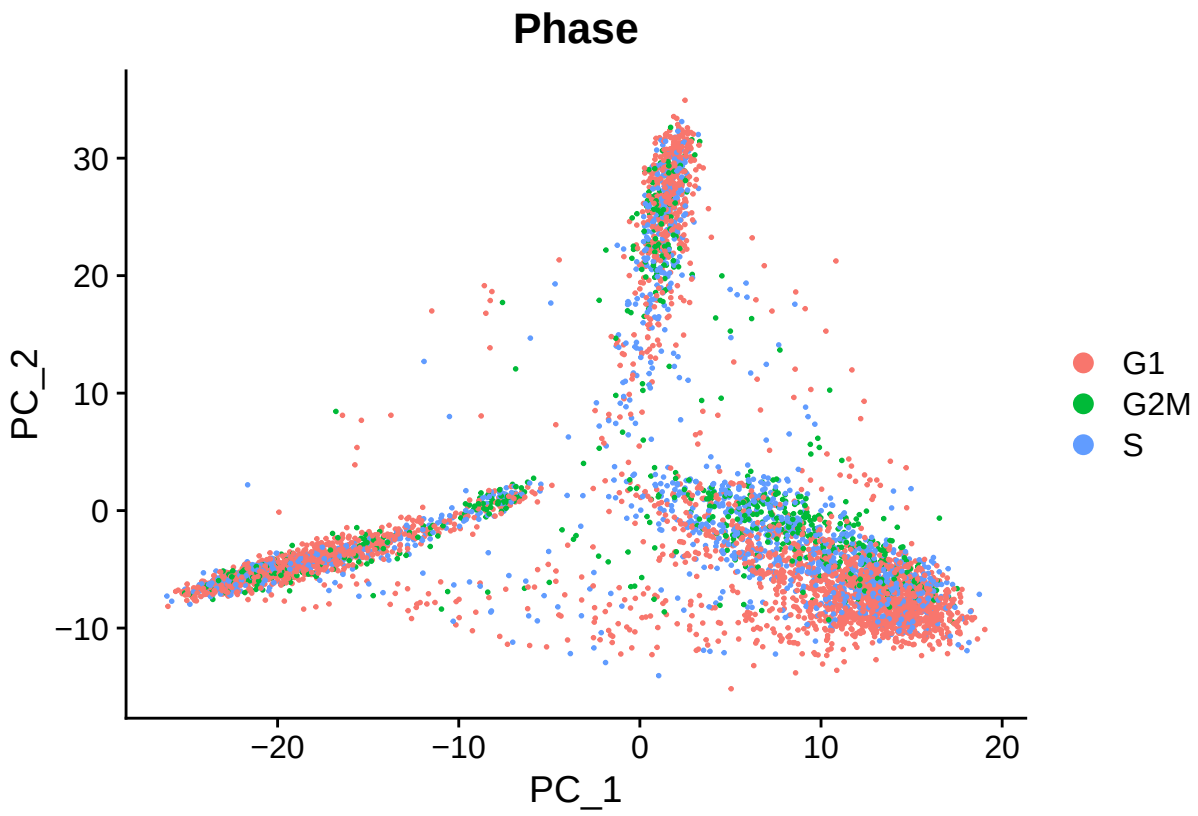
```
## PC_ 1
## Positive:  CLU, PCSK1N, PMP2, PTN, CALD1, ITM2C, SCRG1, GAP43, SEC61G, S100A16
##     PTPRZ1, GFAP, C1R, TUBB3, CHI3L1, CRYAB, SPARCL1, DNER, CHL1, AQP4
##     FABP7, SERPINA3, ID4, IGFBP7, SPARC, TRIM47, CKB, C1orf61, F3, DTNA
## Negative:  TYROBP, SRGN, LAPTM5, FCER1G, CYBA, AIF1, C1QA, IFI30, C1QC, C1QB
##     MS4A7, BCL2A1, PLAUR, CD14, RGS1, CCL3, FCGR3A, TYMP, VAMP8, RNASET2
##     CD68, PLEK, CTSZ, VSIG4, ITGB2, ALOX5AP, HLA-DQB1, MS4A6A, CTSS, CD163
## PC_ 2
## Positive:  MAG, NKX6-2, TF, CLDN11, MOG, KLK6, PLP1, PPP1R14A, EDIL3, ERMN
##     CNDP1, CARNS1, FGFR2, ENPP2, ABCA2, ANLN, DBNDD2, MBP, HCN2, EFHD1
##     HAPLN2, CNTN2, SPOCK3, TMEM144, AIF1L, MYRF, CDK18, RAPGEF5, KCNMB4, PLPP2
## Negative:  ANXA1, S100A10, SOD2, CST3, BST2, C3, HLA-DRA, ACTN1, SAT1, A2M
##     HLA-DPA1, HLA-DRB1, IGFBP7, PLTP, CHI3L1, SPARCL1, FLNA, SERPING1, ITM2C, GFAP
##     C1R, TNC, HLA-DRB5, C1S, ID3, CD74, PDPN, GLRX, MGST1, SEC61G
## PC_ 3
## Positive:  TOP2A, UBE2C, CENPF, BIRC5, PBK, PIMREG, MKI67, NCAPG, ASPM, NUF2
##     UBE2T, NUSAP1, MAD2L1, RRM2, GTSE1, TPX2, CENPK, DLGAP5, PTTG1, ASCL1
##     HMGB2, MELK, TYMS, CENPA, SPC25, CDKN3, KIF2C, CCNA2, C2orf80, SGO1
## Negative:  IFITM3, MYL9, CSRP1, TAGLN, SERPING1, CP, IGFBP7, MGST1, C1S, SOD2
##     STOM, FN1, RRAS, S100A13, MXRA8, FBXO32, SPARC, LTF, SERPINA3, GGT5
##     CFH, C3, PTGDS, CYP1B1, NUPR1, PLAAT4, CHI3L2, VCAM1, SAA1, DCN
## PC_ 4
## Positive:  AQP4, MAOB, C3, GFAP, MLC1, MYBPC1, CLU, CA3, APLNR, RAMP1
##     HP, PMP2, TRIM47, DNER, RAMP3, DTNA, EMID1, CHI3L2, BBOX1, CRB2
##     HLA-DRA, GATM, LRRC2, ITM2C, PPP1R1B, LTF, SCRG1, SERPINA3, TGFB2, CDO1
## Negative:  COL3A1, DCN, COL1A1, COL6A3, FBLN1, LUM, PDGFRB, COL1A2, PLAC9, PDLIM1
##     LAMC3, MYO1B, NID1, EBF1, C7, COL6A2, ITGA1, COX7A1, IGFBP4, COL4A1
##     ASPN, GNG11, CFH, CTSK, CD248, COL4A2, UACA, ISLR, PLXDC1, RGS5
## PC_ 5
## Positive:  ERO1A, CA9, AKAP12, VEGFA, NRN1, GBE1, HILPDA, SLC2A1, TRIB3, NDRG1
```

```
##     IGFBP3, ENO2, EIF4EBP1, INSIG2, PLOD2, ANGPTL4, SERPINE1, AL133453.1, UPP1, BNIP3
##     GRB10, EGLN3, DNAJB9, STC2, SLC38A1, IGFBP5, FAM162A, TUBA4A, ADM, HK2
## Negative:  APOE, CST3, C3, PLTP, HLA-DRA, HLA-DPA1, S100B, CD74, HLA-DRB1, NFIA
##     HLA-DRB5, SERPINF1, THY1, RAMP1, A2M, TTYH1, AGT, PLAU, MAOB, IFITM3
##     AQP4, MYBPC1, C1QB, ID3, AQP1, APLNR, TSC22D4, HLA-DQA1, HLA-DMB, SCRG1
```

```
Seurat::DimPlot(gbm, reduction = "pca")
```



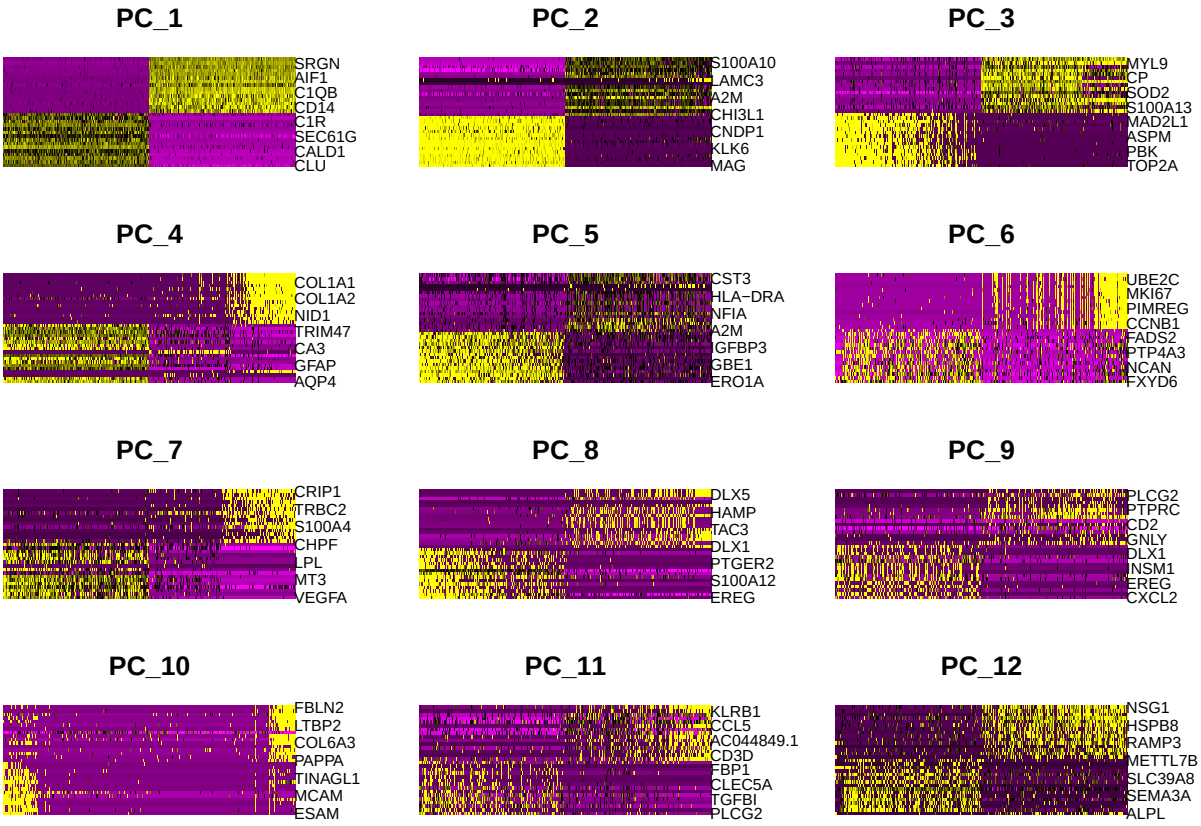

```
# With colors
Seurat::DimPlot(gbm, reduction = "pca", group.by = "Phase")
```



Coming back to the cell cycle analysis, we can check the distribution of the different cell cycle phases over the PCA, and eventually regress it out using the ScaleData() function. But here, the PCA doesn't seem to cluster according to the cell cycle phase. Which is concordant with the less than 1% of variability due to the cell cycle. In consequence, we do not need to regress this variable.

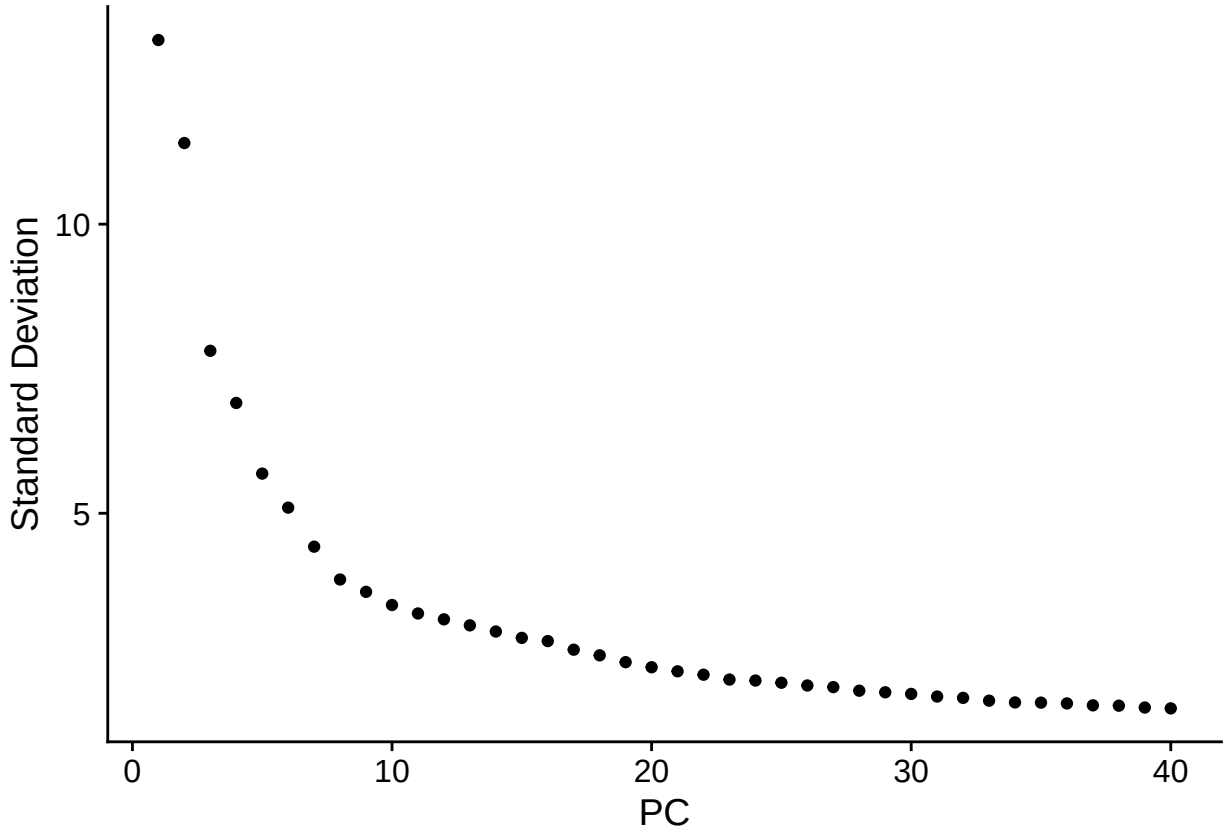
We can see how foe each axis of the PCA the cells correlates to each other.

```
Seurat::DimHeatmap(gbm, dims = 1:12, cells = 500, balanced = TRUE)
```



The elblow plot can help to determine how many PCA axis to use in downstream analysis.

```
Seurat::ElbowPlot(gbm, ndims = 40)
```




```
# we can check the percentage of variance of each PCA axis by using
test <-Seurat::ElbowPlot(gbm, ndims = 40)
test$data$stdev/sum(test$data$stdev)
```

```
##  [1] 0.10054185 0.08695373 0.05955441 0.05267584 0.04335814 0.03887077
##  [7] 0.03372266 0.02939643 0.02776596 0.02603348 0.02491427 0.02413922
## [13] 0.02334632 0.02253185 0.02169708 0.02127612 0.02013977 0.01939621
## [19] 0.01849293 0.01782520 0.01728733 0.01684126 0.01620118 0.01606799
## [25] 0.01578196 0.01542055 0.01520423 0.01472987 0.01451740 0.01430692
## [31] 0.01396067 0.01378444 0.01341494 0.01318329 0.01315741 0.01304710
## [37] 0.01279972 0.01275188 0.01251099 0.01239865
```

The elblow plot ranks principle components based on the percentage of variance explained by each one. Where we observe an "elblow" or flattening curve, the majority of true signal is captured by this number of PCs, eg around 25 PCs for the gbm dataset.

Dimentionality reduction with PCA: PCA doesn't fit for single cell data because • It is a LINEAR method of dimensionalityreduction • It is an interpretable dimensionality reduction • Data is usually SCALEDprior to PCA (Z-score | see ScaleData in the Seurat) • The TOPprincipal components contain higher variance from the data • Can be used as FILTERING,by selecting only the top significant PCs • PCs that explain at least 1%of variance • Jackstraw of significantp-values • The first 5-10 PCs • Scater library describes correlation between PCs and metadata, take PCs until metadata information is covered

The Problems: • The two first PC in SC-RNAseq often account for only few percent of the total variance • It performs poorly to separate cells in 0-inflated data types (because of it non-linearity nature) • Cell sizes and sequencing depth are usually captured in the top principal components

Dimensionality reduction with Umap atype of neighbourg graphs

Capture local structure rather than global structure approaches as PCA. Discriminate better among types.

UMAP is similar to t-SNE but faster!

UMAP: UniformManifold Approximation and Projection • It is a NON-LINEAR graph-based method of dimensionality reduction • UMAP assumes that there is a manifold in the dataset, it could also tend to cluster noise. • Very efficient - O(n) • Can be run from the top PCs (e.g.: PC1 to PC10) • Can use any distance metrics! • Can integrate between different data types (text, numbers, classes) • It is no longer completely stochastic as t-SNE • Defines both LOCAL and GLOBAL distances • Can be applied to newdata points

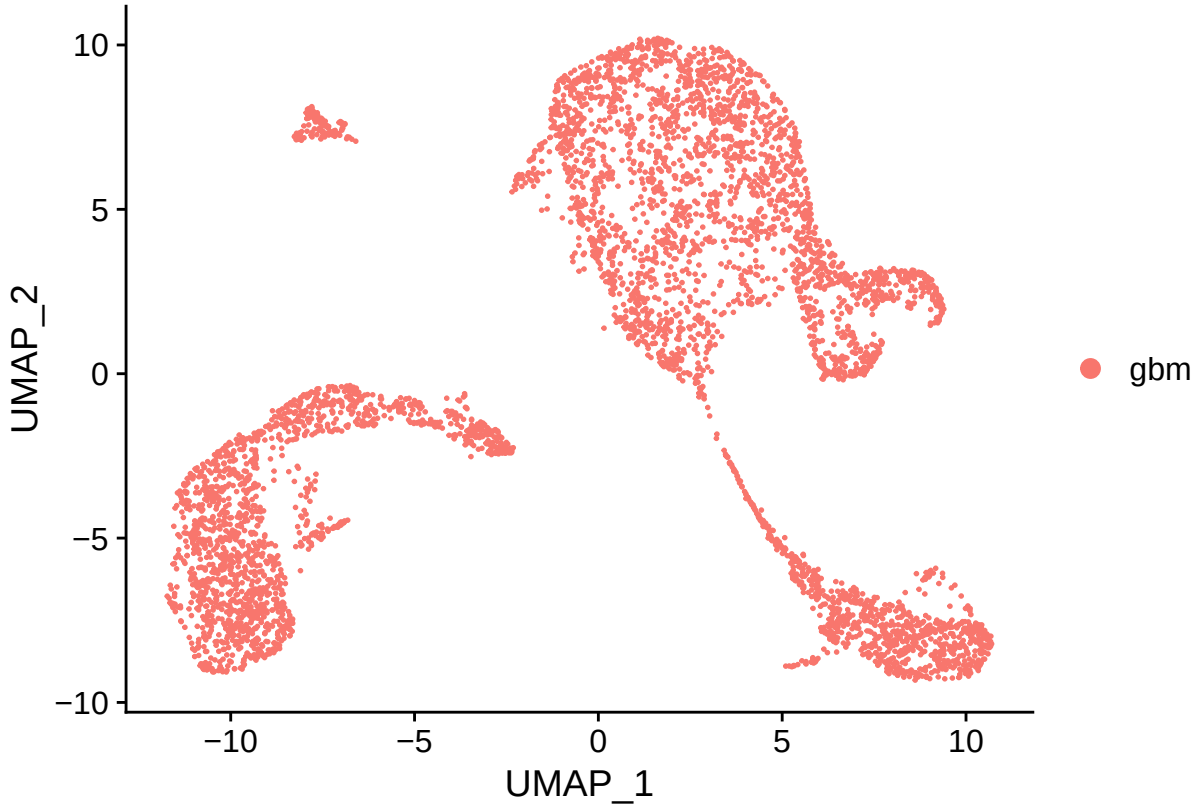
Including too many PCs usually does not affect much the result, while including too few PCs can affect the results very much.

UMAP: The goal of these algorithms is to learn the underlying manifold of the data in order to place similar cells together in low-dimensional space. UMAP uses the previous PCA calculations

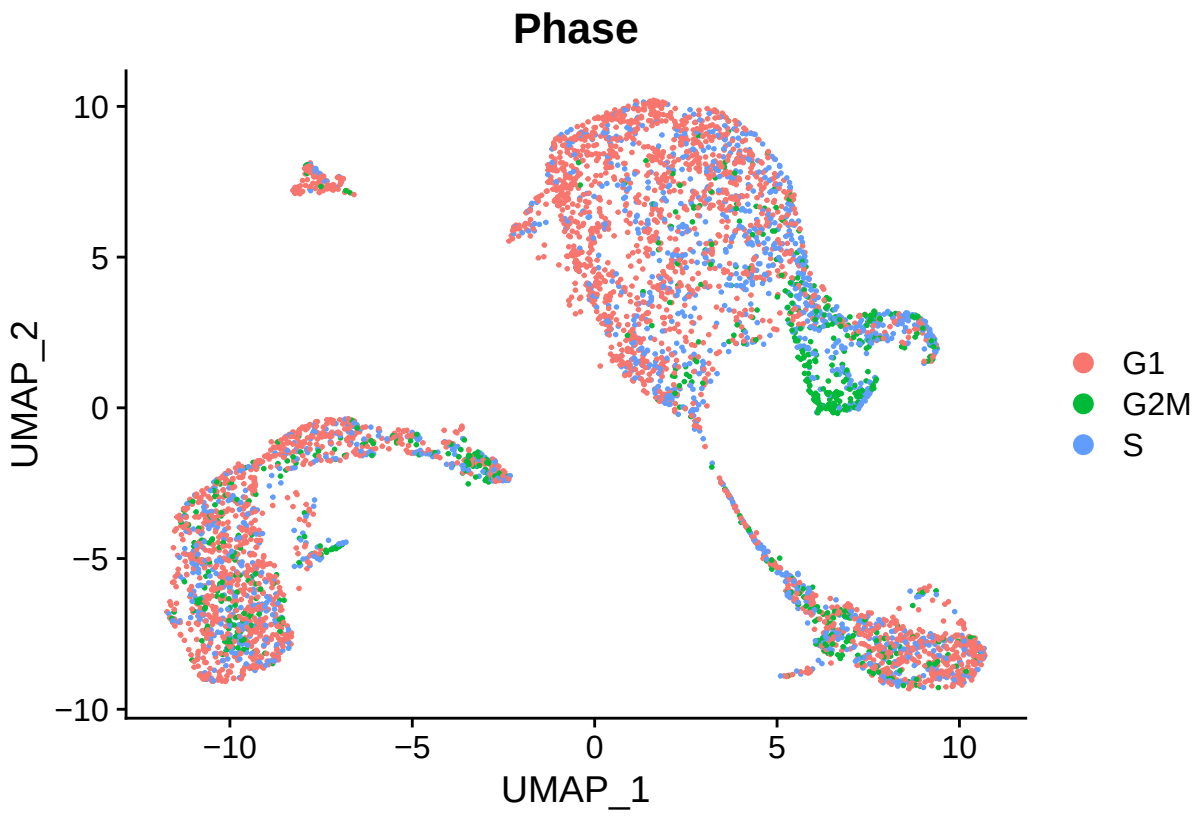
```
gbm <- Seurat::RunUMAP(gbm, dims = 1:5,n.neighbors = 50)
```

```
## 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

## 16:23:49 UMAP embedding parameters a = 0.9922 b = 1.112

## 16:23:49 Read 5091 rows and found 5 numeric columns

## 16:23:49 Using Annoy for neighbor search, n_neighbors = 50

## 16:23:49 Building Annoy index with metric = cosine, n_trees = 50

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

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

## **************************************************|
## 16:23:51 Writing NN index file to temp file C:\Users\imateusg\AppData\Local\Temp\RtmpYhrpl1\file1100
## 16:23:51 Searching Annoy index using 1 thread, search_k = 5000
## 16:23:58 Annoy recall = 100%
## 16:23:59 Commencing smooth kNN distance calibration using 1 thread
## 16:24:04 Initializing from normalized Laplacian + noise
## 16:24:05 Commencing optimization for 500 epochs, with 308004 positive edges
## 16:24:45 Optimization finished
```

```
Seurat::DimPlot(gbm, reduction = "umap")
```




```
# and with coloured samples based on cell cycle
Seurat::DimPlot(gbm, reduction = "umap", group.by = "Phase")
```



Clustering

The method implemented in Seurat first constructs a SNN graph based on the euclidean distance in PCA space, and refine the edge weights between any two cells based on the shared overlap in their local neighborhoods (Jaccard similarity). This step is performed using the FindNeighbors() function, and takes as input the previously defined dimensionality of the dataset.

To cluster the cells, Seurat next implements modularity optimization techniques such as the Louvain algorithm (default) or SLM [SLM, Blondel et al., Journal of Statistical Mechanics], to iteratively group cells together, with the goal of optimizing the standard modularity function. The FindClusters() function implements this procedure, and contains a resolution parameter that sets the 'granularity' of the downstream clustering, with increased values leading to a greater number of clusters.

```
gbm <- Seurat::FindNeighbors(gbm, dims = 1:25)
```

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

```
## Computing SNN
```

```
# determine different cluster parameters from 0.1 to 0.8
gbm <- Seurat::FindClusters(gbm, resolution = seq(0.1, 0.8, by=0.1))
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 5091
## Number of edges: 165105
```

```
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9646
## Number of communities: 6
## Elapsed time: 0 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 5091
## Number of edges: 165105
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9370
## Number of communities: 8
## Elapsed time: 0 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 5091
## Number of edges: 165105
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9214
## Number of communities: 11
## Elapsed time: 1 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 5091
## Number of edges: 165105
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9081
## Number of communities: 12
## Elapsed time: 1 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 5091
## Number of edges: 165105
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8986
## Number of communities: 15
## Elapsed time: 1 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 5091
## Number of edges: 165105
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8899
## Number of communities: 15
## Elapsed time: 1 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 5091
## Number of edges: 165105
```

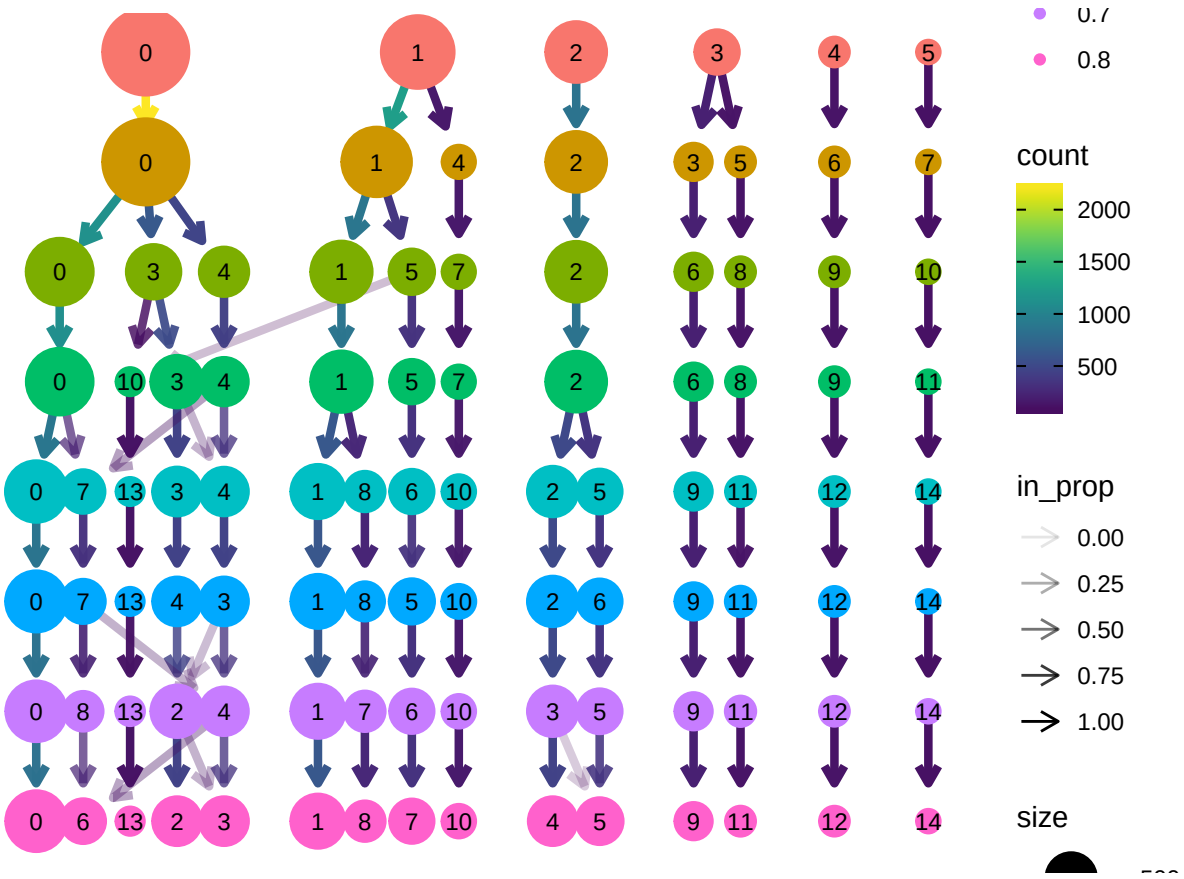
```
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8808
## Number of communities: 15
## Elapsed time: 1 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 5091
## Number of edges: 165105
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8722
## Number of communities: 15
## Elapsed time: 1 seconds
```

```
#Cluster id of each cell is added to the metadata object, as a new column for each resolution tested:
head(gbm@meta.data)
```

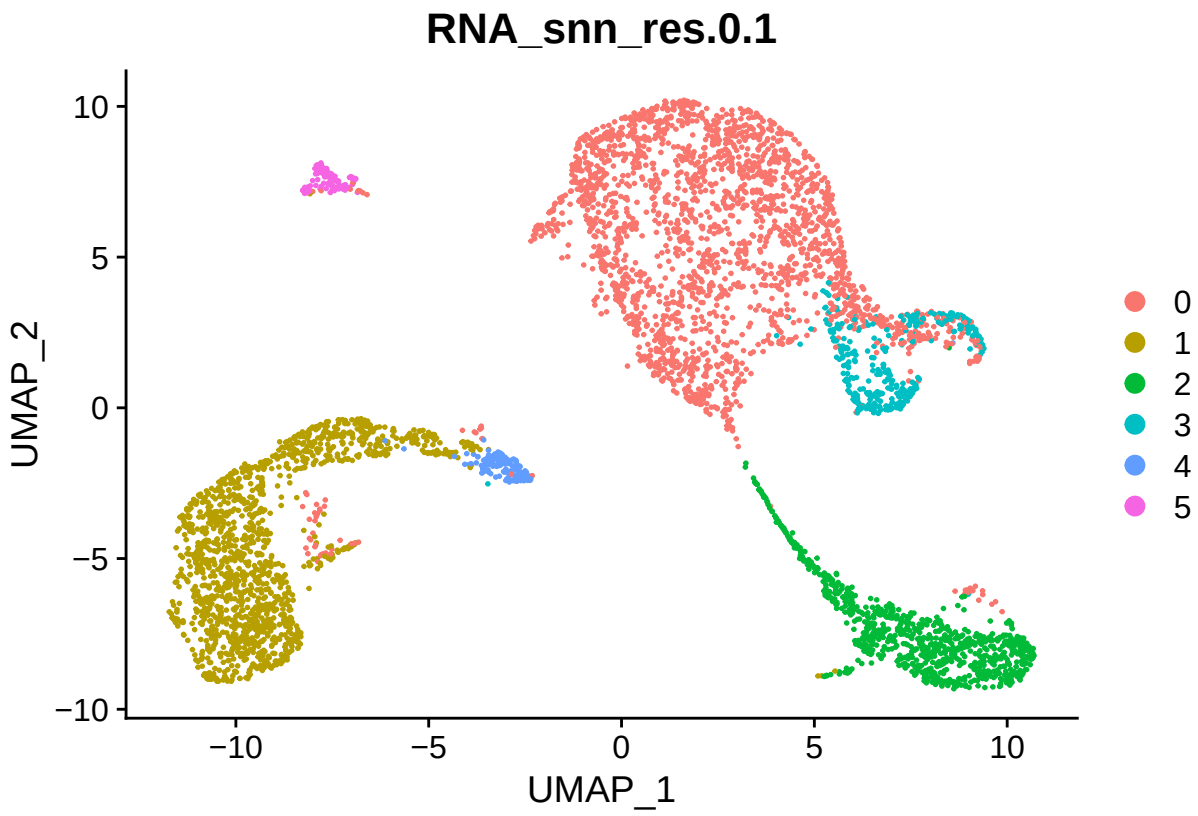
```
##                     orig.ident nCount_RNA nFeature_RNA percent.mt     S.Score
## AAACCCAAGGCGATAC-1        gbm       2225          815  4.7191011 -0.02310297
## AAACCCACAAGTCCCG-1        gbm      17882         4760  1.5098982 -0.05316192
## AAACCCACAGATGCGA-1        gbm       8172         2319  6.4855605 -0.04855563
## AAACCCACAGGTGAGT-1        gbm       9057         3395  3.1577785  0.03622280
## AAACCCAGTCTTGCGG-1        gbm       5612         2716  0.4276550  0.08870032
## AAACCCATCGATAACC-1        gbm       6254         1659  0.7195395 -0.03328401
##                       G2M.Score Phase RNA_snn_res.0.1 RNA_snn_res.0.2
## AAACCCAAGGCGATAC-1 -0.06241569    G1               4               6
## AAACCCACAAGTCCCG-1 -0.04383938    G1               0               0
## AAACCCACAGATGCGA-1 -0.10844466    G1               1               1
## AAACCCACAGGTGAGT-1 -0.07256437     S               2               2
## AAACCCAGTCTTGCGG-1 -0.08898933     S               0               0
## AAACCCATCGATAACC-1 -0.08258553    G1               1               1
##                    RNA_snn_res.0.3 RNA_snn_res.0.4 RNA_snn_res.0.5
## AAACCCAAGGCGATAC-1               9               9              12
## AAACCCACAAGTCCCG-1               0               0               0
## AAACCCACAGATGCGA-1               1               1               1
## AAACCCACAGGTGAGT-1               2               2               2
## AAACCCAGTCTTGCGG-1               0               0               7
## AAACCCATCGATAACC-1               5               5               6
##                    RNA_snn_res.0.6 RNA_snn_res.0.7 RNA_snn_res.0.8
## AAACCCAAGGCGATAC-1              12              12              12
## AAACCCACAAGTCCCG-1               0               0               0
## AAACCCACAGATGCGA-1               1               1               1
## AAACCCACAGGTGAGT-1               2               3               4
## AAACCCAGTCTTGCGG-1               7               8               6
## AAACCCATCGATAACC-1               5               6               7
##                    seurat_clusters
## AAACCCAAGGCGATAC-1              12
## AAACCCACAAGTCCCG-1               0
## AAACCCACAGATGCGA-1               1
## AAACCCACAGGTGAGT-1               4
## AAACCCAGTCTTGCGG-1               6
## AAACCCATCGATAACC-1               7
```

```
# we can view how clusters subdivise at increased resolution
clustree::clustree(gbm@meta.data[,grep("RNA_snn_res", colnames(gbm@meta.data))],
                   prefix = "RNA_snn_res.")
```

```
## Warning: The ‘add‘ argument of ‘group_by()‘ is deprecated as of dplyr 1.0.0.
## Please use the ‘.add‘ argument instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was generated.
```




```
# can view the UMAP with the clustering info
Seurat::DimPlot(gbm, group.by = "RNA_snn_res.0.1")
```



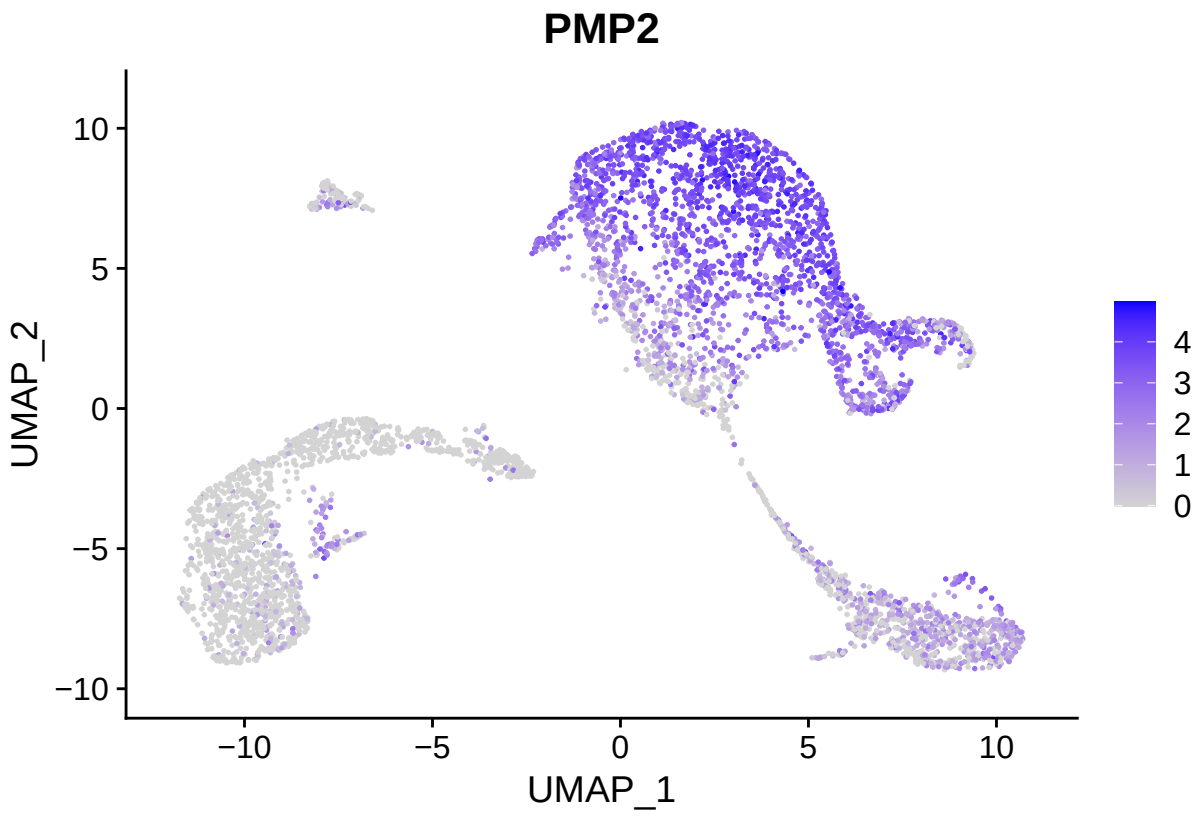

```
# we saw that probably clustering at a resolution of 0.2 gave the most sensible results. Let's therefor
gbm <- Seurat::SetIdent(gbm, value = gbm$RNA_snn_res.0.2)
```

Cell annotation

From now on, grouping (e.g. for plotting) is done by the active identity (set at @active.ident) by default

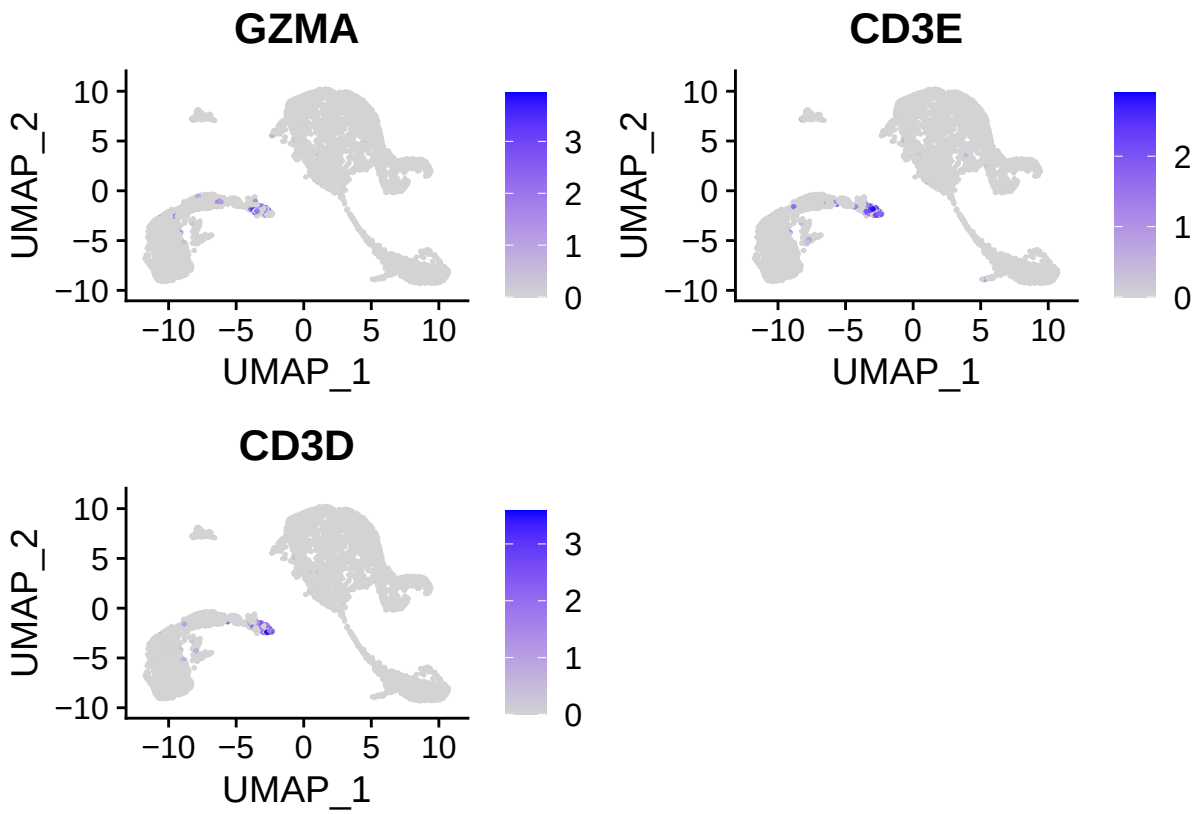
Based on the UMAP we have generated, we can visualize expression for a gene in each cluster:

```
Seurat::FeaturePlot(gbm, "PMP2")
```



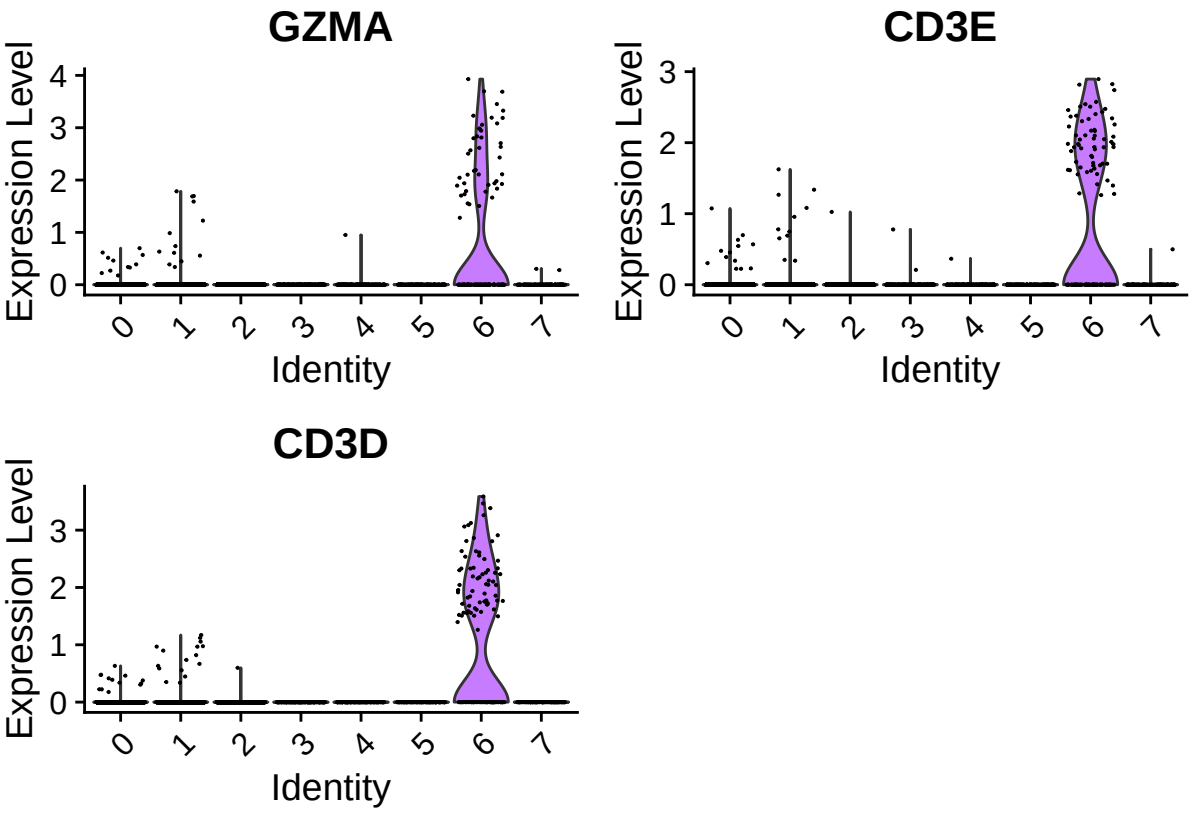
Based on expression of sets of genes you can do a manual cell type annotation. If you know the marker genes for some cell types, you can check whether they are up-regulated in one or the other cluster. Here we have some marker genes for two different cell types:

```
immune_genes<-c("GZMA", "CD3E", "CD3D")
microglia_genes<-c("CCL4", "CCL3", "P2RY12", "C1QB", "CSF1ER", "CY3CR1")
#Let's have a look at the expression of the three immune genes:
Seurat::FeaturePlot(gbm, immune_genes, ncol=2)
```



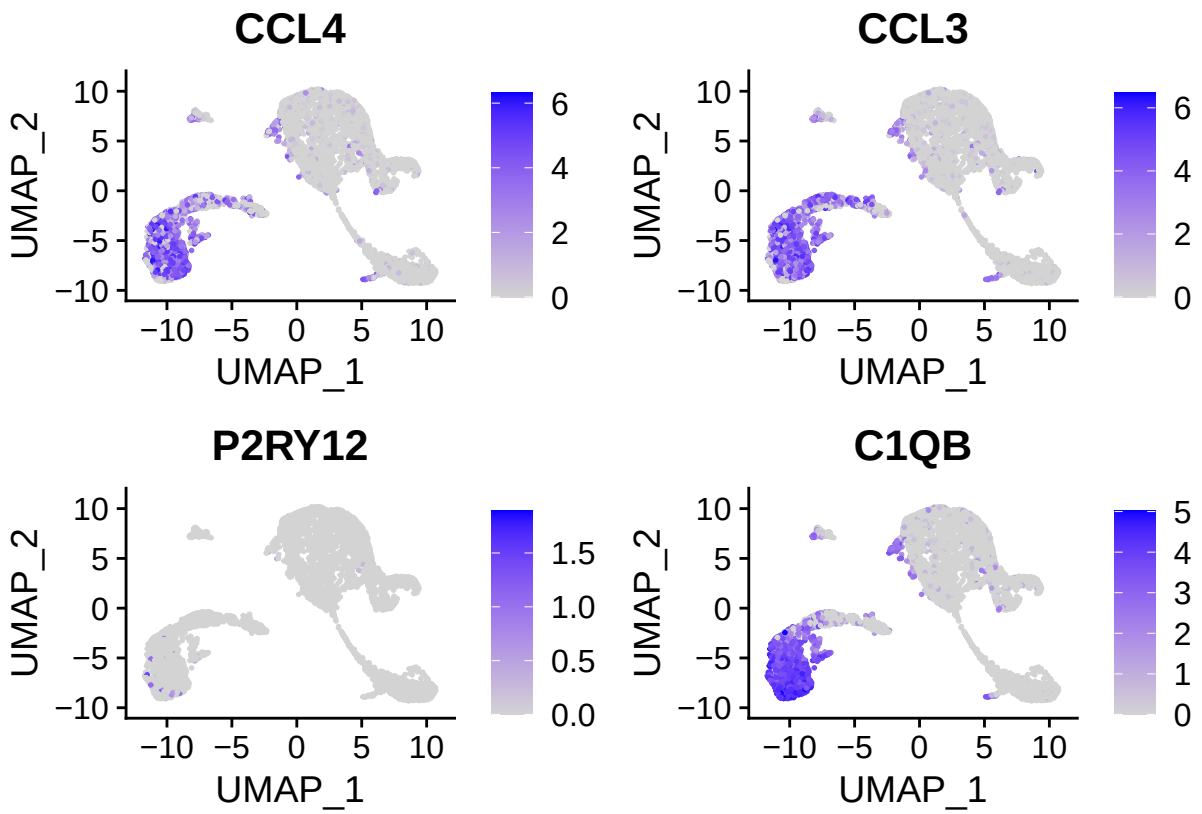
These cells are almost all in cluster 6. Which becomes clearer when looking at the violin plot:

```
Seurat::VlnPlot(gbm,
                features = immune_genes,
                ncol = 2)
```



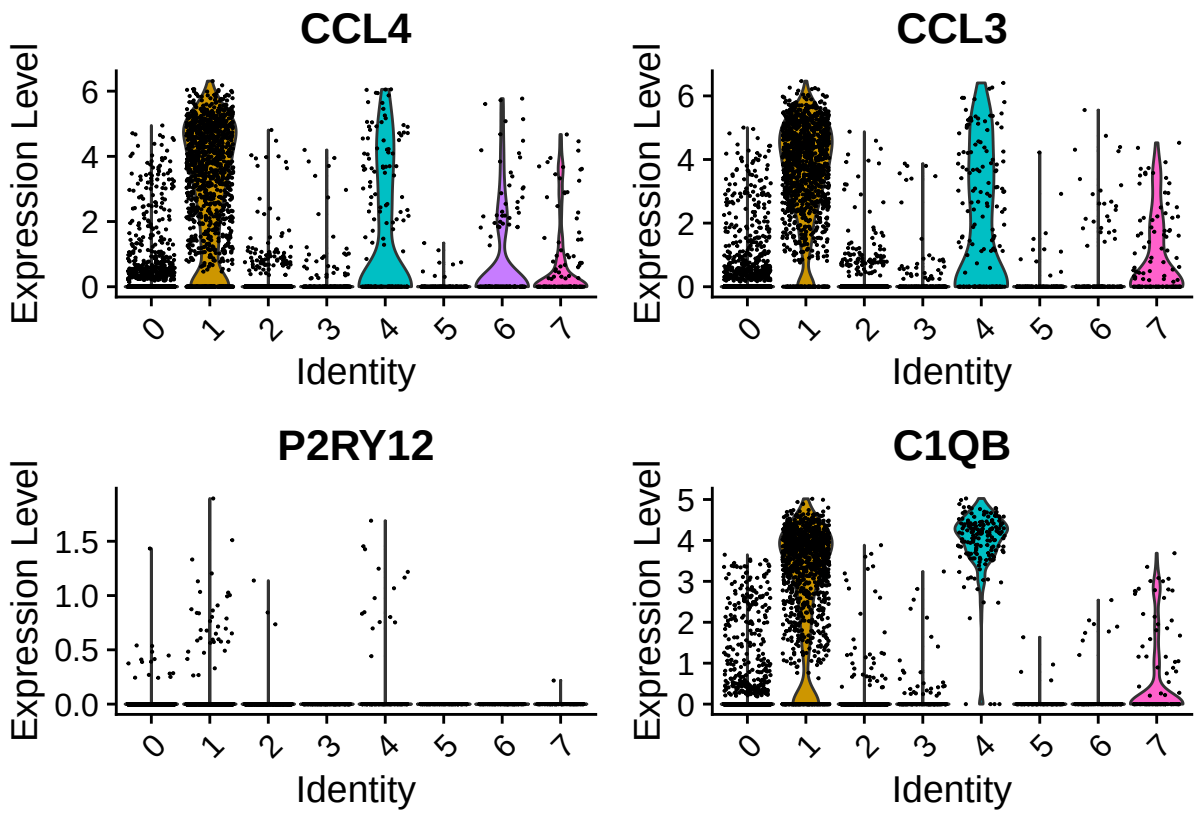

```
Seurat::FeaturePlot(gbm, microglia_genes, ncol=2)
```

```
## Warning in FetchData(object = object, vars = c(dims, "ident", features), : The
## following requested variables were not found: CSF1ER, CY3CR1
```

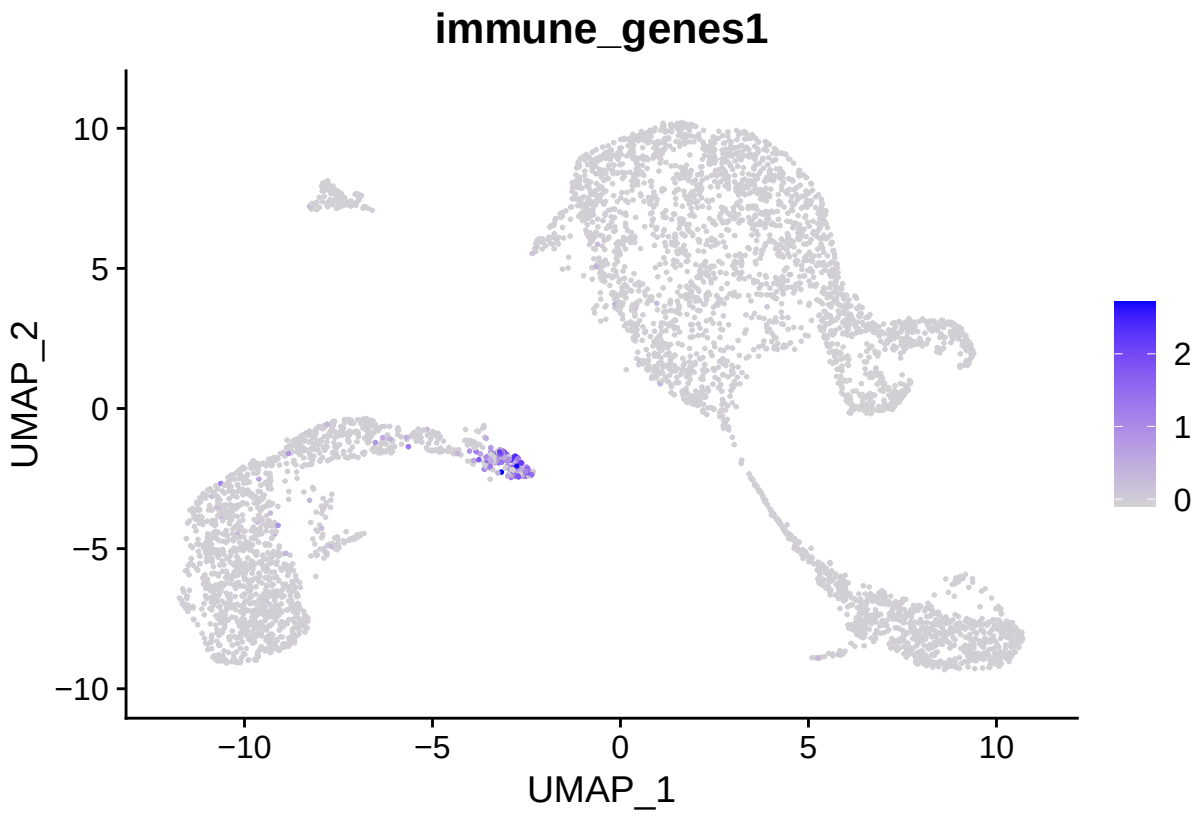



```
# which correspond mainly to cluster 1 and 4
Seurat::VlnPlot(gbm,
                features = microglia_genes,
                ncol = 2)
```

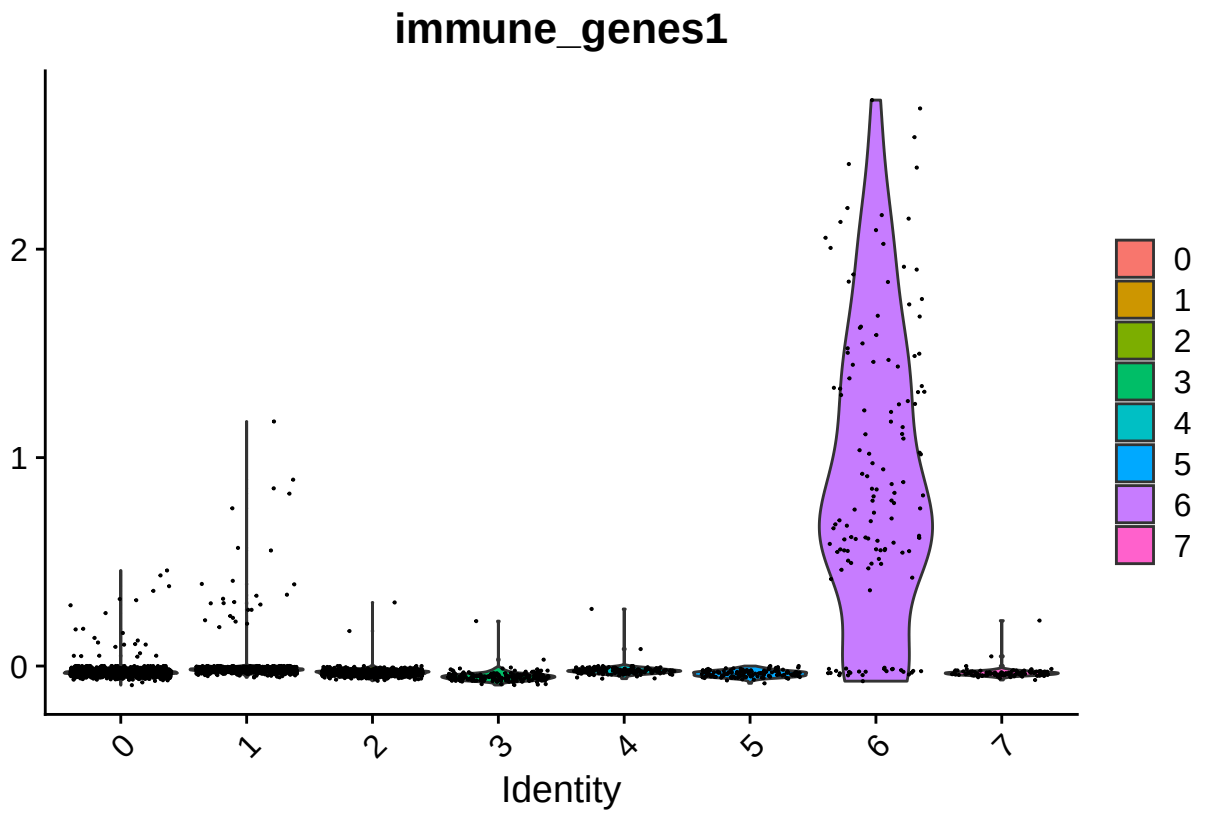
```
## Warning in FetchData(object = object, vars = features, slot = slot): The
## following requested variables were not found: CSF1ER, CY3CR1
```




```
#We can also automate this with the function AddModuleScore. For each cell, an expression score for a g

gbm <- Seurat::AddModuleScore(gbm,
                              features = list(immune_genes),
                              name = "immune_genes")
Seurat::FeaturePlot(gbm, "immune_genes1")
```




```
Seurat::VlnPlot(gbm,
                "immune_genes1") # please note that the column is named immune_genes1
```



Cell type annotation using SingleR

To do a fully automated annoation, we need a reference dataset of primary cells. Any reference could be used. The package scRNAseq in Bioconductor includes several scRNAseq datasets that can be used as reference to SingleR. One could also use a reference made of bulk RNA seq data. Here we are using the Human Primary Cell Atlas dataset from celldex. Check out what's in there:

```
hpca.se <- celldex::HumanPrimaryCellAtlasData()
```

```
## snapshotDate(): 2021-10-19
```

```
## see ?celldex and browseVignettes('celldex') for documentation
```

```
## loading from cache
```

```
## see ?celldex and browseVignettes('celldex') for documentation
```

```
## loading from cache
```

```
class(hpca.se)
```

```
## [1] "SummarizedExperiment"
## attr(,"package")
## [1] "SummarizedExperiment"
```

```
table(hpca.se$label.main)
```

```
##
##            Astrocyte               B_cell                   BM
##                    2                   26                    7
##          BM & Prog.         Chondrocytes                  CMP
##                    1                    8                    2
##                   DC Embryonic_stem_cells    Endothelial_cells
##                   88                   17                   64
##     Epithelial_cells         Erythroblast          Fibroblasts
##                   16                    8                   10
##          Gametocytes                  GMP          Hepatocytes
##                    5                    2                    3
##           HSC_-G-CSF            HSC_CD34+            iPS_cells
##                   10                    6                   42
##        Keratinocytes           Macrophage                  MEP
##                   25                   90                    2
##             Monocyte                  MSC            Myelocyte
##                   60                    9                    2
## Neuroepithelial_cell              Neurons          Neutrophils
##                    1                   16                   21
##              NK_cell          Osteoblasts            Platelets
##                    5                   15                    5
##     Pre-B_cell_CD34-     Pro-B_cell_CD34+        Pro-Myelocyte
##                    2                    2                    2
##  Smooth_muscle_cells              T_cells    Tissue_stem_cells
##                   16                   68                   55
```

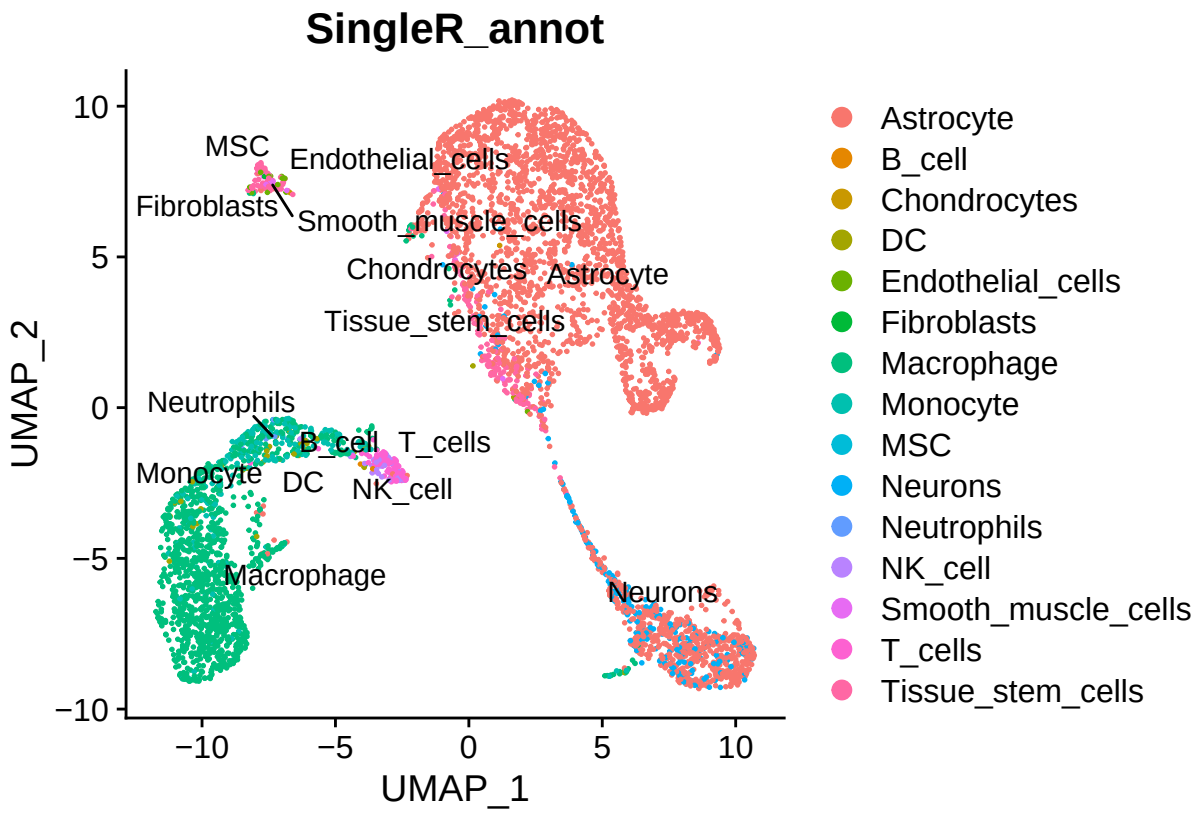
```
#Now SingleR compares our normalized count data to a reference set, and finds the most probable annotio

gbm_SingleR <- SingleR::SingleR(test = Seurat::GetAssayData(gbm, slot = "data"),
                                ref = hpca.se,
                                labels = hpca.se$label.main)
head(gbm_SingleR)
```

```
## DataFrame with 6 rows and 5 columns
##                                           scores first.labels
##                                         <matrix>  <character>
## AAACCCAAGGCGATAC-1 0.132486:0.246913:0.229535:...      T_cells
## AAACCCACAAGTCCCG-1 0.507144:0.204311:0.166601:...    Astrocyte
## AAACCCACAGATGCGA-1 0.147164:0.302800:0.311181:...     Monocyte
## AAACCCACAGGTGAGT-1 0.345649:0.218030:0.178937:...    Astrocyte
## AAACCCAGTCTTGCGG-1 0.400602:0.187095:0.134520:...    Astrocyte
## AAACCCATCGATAACC-1 0.157503:0.281883:0.293659:...     Monocyte
##                         tuning.scores      labels pruned.labels
##                           <DataFrame> <character>   <character>
## AAACCCAAGGCGATAC-1 0.325948:0.227579     T_cells       T_cells
## AAACCCACAAGTCCCG-1 0.507144:0.415796   Astrocyte     Astrocyte
## AAACCCACAGATGCGA-1 0.187794:0.114895  Macrophage    Macrophage
```

```
## AAACCCACAGGTGAGT-1 0.275623:0.219847     Neurons       Neurons
## AAACCCAGTCTTGCGG-1 0.400602:0.338047   Astrocyte     Astrocyte
## AAACCCATCGATAACC-1 0.202418:0.174400    Monocyte      Monocyte
```

In order to visualize it in our UMAP, we have to add the annotation to gbm@meta.data:

```
gbm$SingleR_annot <- gbm_SingleR$labels
# and vizualize UMAP
Seurat::DimPlot(gbm, group.by = "SingleR_annot", label = T, repel = T)
```




```
#We can have a look at the mean module score for each annotation like this:

mean_scores <- tapply(gbm$immune_genes1, gbm$SingleR_annot, mean)
mean_scores[order(mean_scores, decreasing = TRUE)[1:6]]
```

```
##      T_cells      NK_cell           DC     Monocyte   Macrophage Chondrocytes
##  1.015605822  0.679592135  0.017018871 -0.005414908 -0.011926460 -0.011975024
```

Showing that T-cells and NK-cells have a high module score based on our set of immune genes, which makes sense.

Of course, it was also already clear from the UMAP plots that cluster 6 (the cluster with the high module score for the immune genes) contained the T-cells and NK-cells.

! A recent preprint suggest that t-SNE and UMAP do not preserve local or global structure & are misleading. They're also arbitrary

link to bioRxiv : https://t.co/XkAOTKlOcs?amp=1 link to twitter Thread : https://twitter.com/lpachter/status/1431325969411821572

you can't make claims about datasets being the same or different based on a t-SNE or UMAP alone.

Picasso tool could be a solution to overcome this. showing how cell cluster but making any nice plot, like an elephant o a gorilla!

Integration

`This chapter uses the pancreas dataset, which is described in this paper : https://www.biorxiv.org/conte`

Single cell transcriptomics (scRNA-seq) has transformed our ability to discover and annotate cell types and states, but deep biological understanding requires more than a taxonomic listing of clusters. As new methods arise to measure distinct cellular modalities, including high-dimensional immunophenotypes, chromatin accessibility, and spatial positioning, a key analytical challenge is to integrate these datasets into a harmonized atlas that can be used to better understand cellular identity and function.

The gbm dataset does not contain any samples, treatments or methods to integrate. Therefore for these exercises we will use a different dataset that is described in Comprehensive Integration of Single CellData. It is a dataset comprising of four different single cell experiment performed by using four different methods.

```
pancreas.data <- readRDS(file = "C://Users/imateusg/Documents/WW_courses/02_SingleCELL_SIB2021/single_c
metadata <- readRDS(file = "C://Users/imateusg/Documents/WW_courses/02_SingleCELL_SIB2021/single_cell_c

pancreas <- Seurat::CreateSeuratObject(pancreas.data, meta.data = metadata)
head(pancreas@meta.data)
```

```
##               orig.ident nCount_RNA nFeature_RNA   tech celltype
## D101_5     SeuratProject   4615.810         1986 celseq    gamma
## D101_43    SeuratProject  11711.506         3942 celseq    gamma
## D101_93    SeuratProject   5567.659         2418 celseq    gamma
## D102_4     SeuratProject   6804.533         2846 celseq    gamma
## D172444_23 SeuratProject   5541.101         2436 celseq    gamma
## D172444_68 SeuratProject   4301.892         2015 celseq    gamma
```

```
table(pancreas@meta.data$celltype)
```

```
##
##             acinar activated_stellate              alpha               beta
##                711                180               2281               1172
##              delta             ductal        endothelial            epsilon
##                405               1065                 61                 14
##              gamma         macrophage               mast quiescent_stellate
##                359                 24                 17                 20
##            schwann
##                 12
```

The dataset contain ifo about tech (sequencing platform) and celltype (13 different types). The cell types where previously defined, with the use of antibodies???

Visualize the data, so first: normalize, find features as , scale data, run PCA and UMAP.

```
pancreas <- Seurat::NormalizeData(pancreas)
pancreas <- Seurat::FindVariableFeatures(pancreas, selection.method = "vst", nfeatures = 2000)
pancreas <- Seurat::ScaleData(pancreas)
```

```
## Centering and scaling data matrix
```

```
pancreas <- Seurat::RunPCA(pancreas, npcs = 30)
```

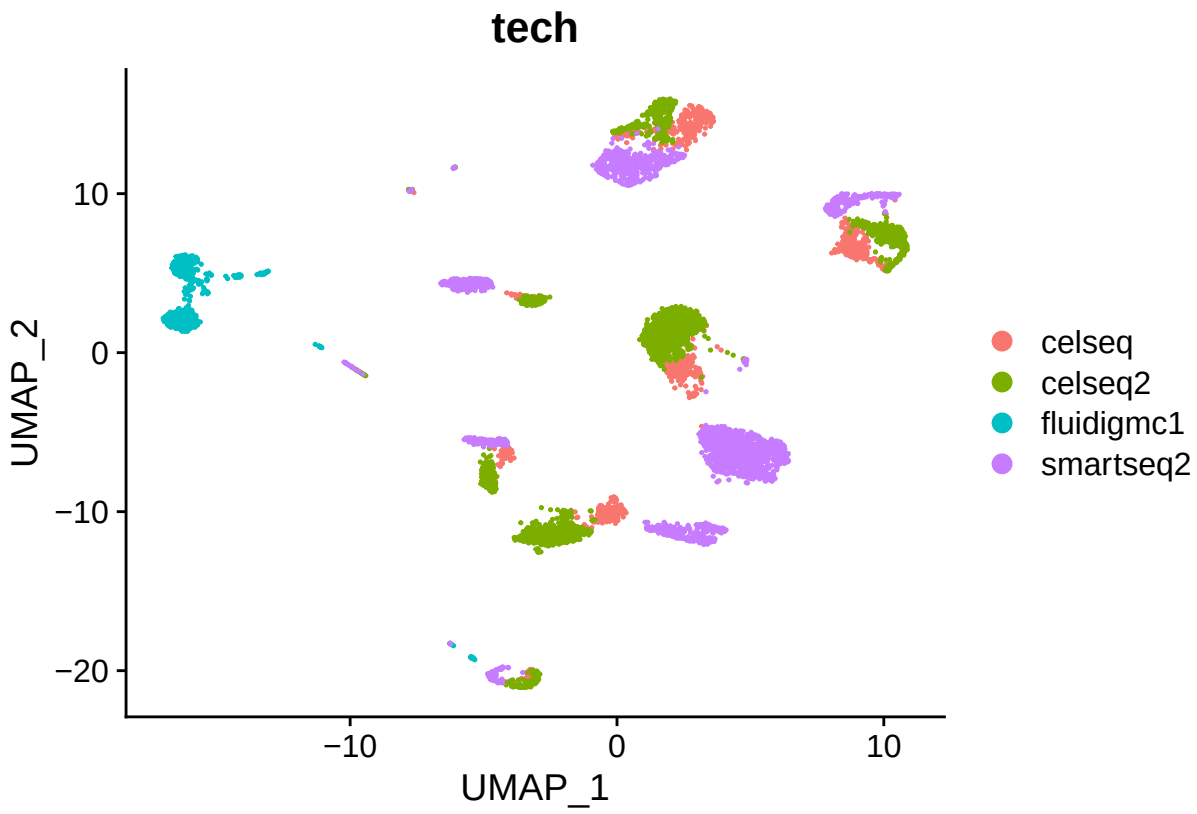
```
## PC_ 1
## Positive:  SPARC, COL4A1, COL1A2, NID1, COL4A2, PDGFRB, COL3A1, COL15A1, PXDN, BGN
##     SERPINH1, COL1A1, COL5A1, COL6A2, MRC2, SFRP2, CDH11, LUM, IGFBP4, COL5A2
##     FN1, THBS2, LAMA4, VCAN, CYGB, LGALS1, LTBP2, LOXL2, DCN, EDNRA
## Negative:  CRYBA2, VGF, PRUNE2, PLCB4, CFC1, LOXL4, C10orf10, PLCE1, MMRN1, PFKFB2
##     PAPPA2, IAPP, EDN3, ADCYAP1, CDKN1C, RBP4, WNK3, KLHL41, ZNF91, PDK4
##     NT5DC3, BTG2, BMP5, FOXP2, ENTPD3, MAST3, ZNF331, ANO5, WSCD2, GPM6A
## PC_ 2
## Positive:  COL6A3, PRUNE2, TIMP1, IGFBP5, PLCB4, ERO1B, TCF4, PFKFB2, ARID5B, ITGA1
##     ATP5F1B, AL354740.1, ZEB1, IAPP, PLAT, SPARC, ADCYAP1, ERO1A, TENT5C, SLC25A6
##     COL1A2, COL5A2, TENT5A, ARFGEF3, PDGFRB, SFRP2, LUM, GPNMB, BGN, COL3A1
## Negative:  SERPINA3, TACSTD2, CFB, SDC4, TM4SF1, IL32, GATM, ANXA4, AKR1C3, LCN2
##     KRT7, PDZK1IP1, MUC1, KRT18, KRT8, REG1A, CLDN4, ANPEP, CD44, SAT1
##     C3, GSTA1, SPINK1, GDF15, ZFP36L1, PRSS1, OLFM4, SERINC2, SOD2, DUOX2
## PC_ 3
## Positive:  ATP5F1B, ERO1A, ERO1B, AL354740.1, SLC25A6, SUSD6, TENT5A, TENT5C, ARFGEF3, SRPRA
##     LNPK, ANKRD36C, HNRNPH2, ECPAS, ARRDC3, BSCL2, FAM234B, AC118549.1, ELOA, PPP4R3A
##     TXNDC5, ERBIN, VPS35L, FXYD2, SYNC, GUCY1B1, DEPP1, TTC39C, NECTIN3, TRIM69
## Negative:  VGF, C10orf10, CRYBA2, SST, LOXL4, PTP4A3, KLHL41, PPY, PDK4, PLCE1
##     IGFBP2, TIMP1, COL1A1, CYGB, COL6A2, MUC13, COL5A1, COL1A2, COL3A1, HSPB1
##     SPON2, MRC2, PDGFRB, AQP3, COL5A2, GREM1, ABCG2, AEBP1, LOXL2, SFRP2
## PC_ 4
## Positive:  CFTR, KRT19, TINAGL1, VTCN1, MMP7, AQP1, SPP1, ALDH1A3, TSPAN8, SERPING1
##     KRT23, SERPINA5, ANXA3, HSD17B2, DEFB1, LGALS4, PMEPA1, CEACAM7, TFPI2, SLC3A1
##     APCDD1, SERPINA1, VCAM1, NRP1, SLPI, TNFAIP2, LAMC2, ADAMTS9, PDLIM3, CD74
## Negative:  CTRC, CPA2, PNLIP, CTRB1, CPA1, PLA2G1B, CTRB2, CELA3A, CELA2A, CPB1
##     PRSS1, PRSS3P2, KLK1, CEL, REG1B, PRSS3, PNLIPRP1, PNLIPRP2, CELA3B, BCAT1
##     ALB, GSTA2, CLPS, GP2, SPINK1, CTRL, CELA2B, REG3G, ALDOB, MGST1
## PC_ 5
## Positive:  FLT1, PODXL, PECAM1, KDR, ESAM, CD93, PLVAP, ECSCR, ERG, MYCT1
##     RGCC, ACVRL1, ELTD1, GPR4, PTPRB, EMCN, S1PR1, CALCRL, MMRN2, CLEC14A
##     CDH5, ROBO4, ABI3, PRDM1, ESM1, PASK, F2RL3, GIMAP4, CXCR4, GPR116
## Negative:  C1S, SERPING1, CFTR, KRT19, SFRP2, ALDH1A3, SPP1, LUM, MMP7, TSPAN8
##     COL5A1, COL3A1, THBS2, VTCN1, SERPINA5, PDGFRB, COL1A2, COL1A1, FN1, LTBP2
##     TNFAIP6, CDH11, ANXA3, COL6A3, DCN, COL5A2, LAMA2, LGALS4, AQP1, TFPI2
```

```
pancreas <- Seurat::RunUMAP(pancreas, reduction = "pca", dims = 1:30)
```

```
## 16:29:15 UMAP embedding parameters a = 0.9922 b = 1.112
```

```
## 16:29:15 Read 6321 rows and found 30 numeric columns

## 16:29:15 Using Annoy for neighbor search, n_neighbors = 30

## 16:29:15 Building Annoy index with metric = cosine, n_trees = 50

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

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

## **************************************************|
## 16:29:16 Writing NN index file to temp file C:\Users\imateusg\AppData\Local\Temp\RtmpYhrpl1\file1100
## 16:29:16 Searching Annoy index using 1 thread, search_k = 3000
## 16:29:19 Annoy recall = 100%
## 16:29:20 Commencing smooth kNN distance calibration using 1 thread
## 16:29:23 Initializing from normalized Laplacian + noise
## 16:29:23 Commencing optimization for 500 epochs, with 268732 positive edges
## 16:30:07 Optimization finished
```

```
#and plot the UMAP based on technology
Seurat::DimPlot(pancreas, reduction = "umap", group.by = "tech")
```

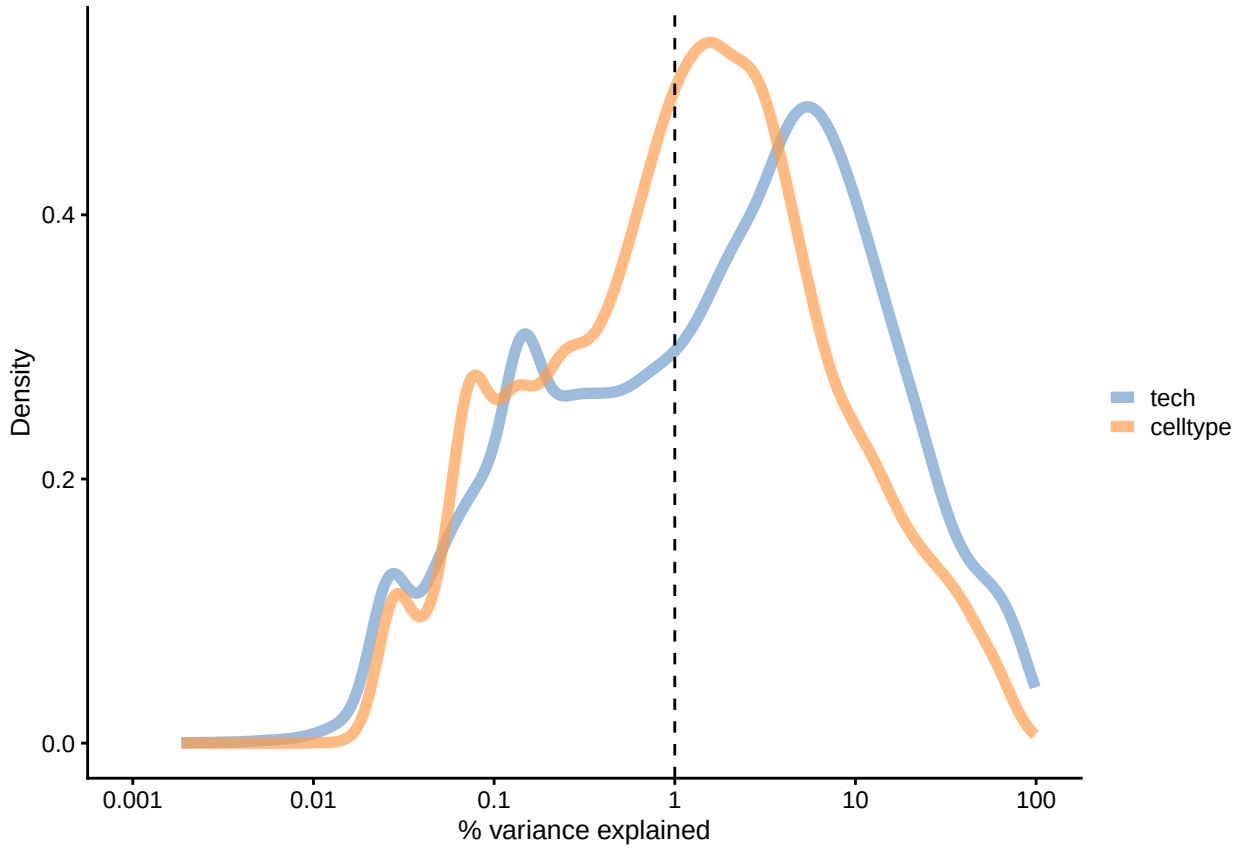


```
# part of variance explained by the 2 variables

ctsP <- Seurat::GetAssayData(pancreas, slot = "counts")
pancreas_sce <- SingleCellExperiment::SingleCellExperiment(
  assays = list(counts = ctsP),
  colData = pancreas@meta.data,
  rowData = rownames(pancreas) )

pancreas_sce <- scater::logNormCounts(pancreas_sce)  # alternative to Seurat's normalization here using

vars2 <- scater::getVarianceExplained(pancreas_sce,
                              variables = c("tech","celltype") )
head(vars2) # show how each variable is  explained for each gene
```

```
##                tech   celltype
## A1BG-AS1  4.1036122  1.2981405
## A1BG     15.3667955 13.7030644
## A1CF      8.2437156 32.0125077
## A2M-AS1   0.6586314  0.7641035
## A2ML1     5.8304207  0.3804157
## A2MP1     0.1415865  0.6000728
```

```
scater::plotExplanatoryVariables(vars2)
```

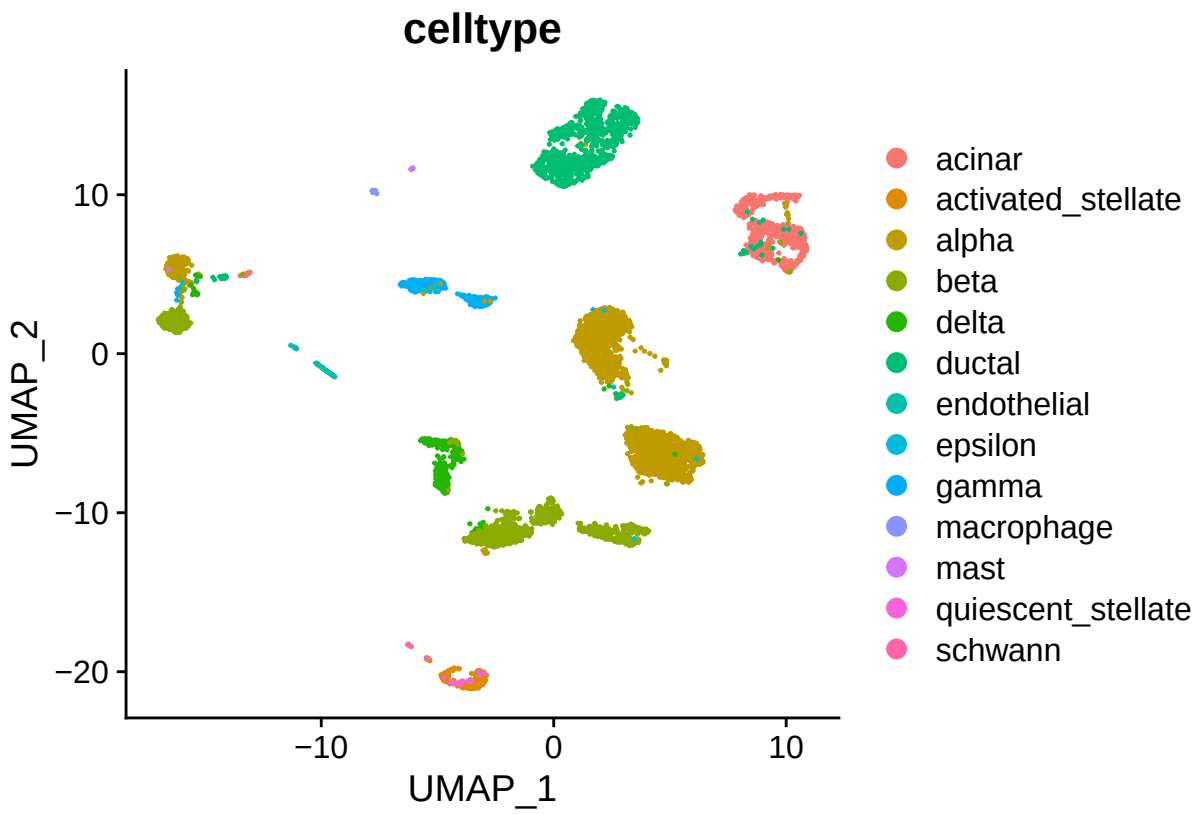
```
## Warning: Removed 10516 rows containing non-finite values (stat_density).
```



We observe that there are bias related to thech and celltype variable.

IF we plot the umap in fonction of the celltype

```
Seurat::DimPlot(pancreas, reduction = "umap", group.by = "celltype")
```



We observe that the celltype describes well the dataset. So we need to integrate he tech type into account as an explanatory variable

To perform the integration, we split the combined object into a list, with each dataset as an element. We perform standard preprocessing (log-normalization), and identify variable features individually for each dataset based on a variance stabilizing transformation ("vst").

So we split the tech dataset and normalize it independantly for each tech type

```
pancreas.list <- Seurat::SplitObject(pancreas, split.by = "tech")

for (i in 1:length(pancreas.list)) {
    pancreas.list[[i]] <- Seurat::NormalizeData(pancreas.list[[i]])
    pancreas.list[[i]] <- Seurat::FindVariableFeatures(pancreas.list[[i]], selection.method = "vst", nf
        verbose = FALSE)
}
```

And lets proceed with the integration that is made in two steps

```r
pancreas.anchors <- Seurat::FindIntegrationAnchors(object.list = pancreas.list, dims = 1:30)
```

```
## Computing 2000 integration features

## Scaling features for provided objects

## Finding all pairwise anchors

## Running CCA

## Merging objects

## Finding neighborhoods

## Finding anchors

##  Found 3498 anchors

## Filtering anchors

##  Retained 2813 anchors

## Running CCA

## Merging objects

## Finding neighborhoods

## Finding anchors

##  Found 2176 anchors

## Filtering anchors

##  Retained 1794 anchors

## Running CCA

## Merging objects

## Finding neighborhoods

## Finding anchors

##  Found 2774 anchors

## Filtering anchors
```

```
##  Retained 2457 anchors
## Running CCA
## Merging objects
## Finding neighborhoods
## Finding anchors
##  Found 3515 anchors
## Filtering anchors
##  Retained 2701 anchors
## Running CCA
## Merging objects
## Finding neighborhoods
## Finding anchors
##  Found 6171 anchors
## Filtering anchors
##  Retained 4631 anchors
## Running CCA
## Merging objects
## Finding neighborhoods
## Finding anchors
##  Found 2721 anchors
## Filtering anchors
##  Retained 1748 anchors
```

```
pancreas.integrated <- Seurat::IntegrateData(anchorset = pancreas.anchors, dims = 1:30)
```

```
## Merging dataset 3 into 2
```

```
## Extracting anchors for merged samples

## Finding integration vectors

## Finding integration vector weights

## Integrating data

## Merging dataset 1 into 2 3

## Extracting anchors for merged samples

## Finding integration vectors

## Finding integration vector weights

## Integrating data

## Merging dataset 4 into 2 3 1

## Extracting anchors for merged samples

## Finding integration vectors

## Finding integration vector weights

## Integrating data
```

After running IntegrateData, the Seurat object will contain an additional element of class Assay with the integrated (or 'batch-corrected') expression matrix. This new Assay is called integrated, and exists next to the already existing RNA element with class Assay.

! Attention.Use the Assay integrated only for clustering and visualisation. It will give unexpected results during e.g. differential gene expression analysis. Therefore, use the RNA element for other analyses

We dont need to re-run FindVariableFeatures, these were automatically set by calling IntegrateData

We switch the default assay to the integrated one (instead of RNA)

```
Seurat::DefaultAssay(pancreas.integrated) <- "integrated"
```

In order to redo the clustering, scale the integrated data, run the PCA and the UMAP again (using the function ScaleData, RunPCA and RunUMAP). After that, generate the same two UMAP plots (grouped by tech and by celltype). Did the integration perform well?

Performing the scaling, PCA and UMAP:

```
pancreas.integrated <- Seurat::ScaleData(pancreas.integrated)
```

```
## Centering and scaling data matrix
```

```
pancreas.integrated <- Seurat::RunPCA(pancreas.integrated, npcs = 30)
```

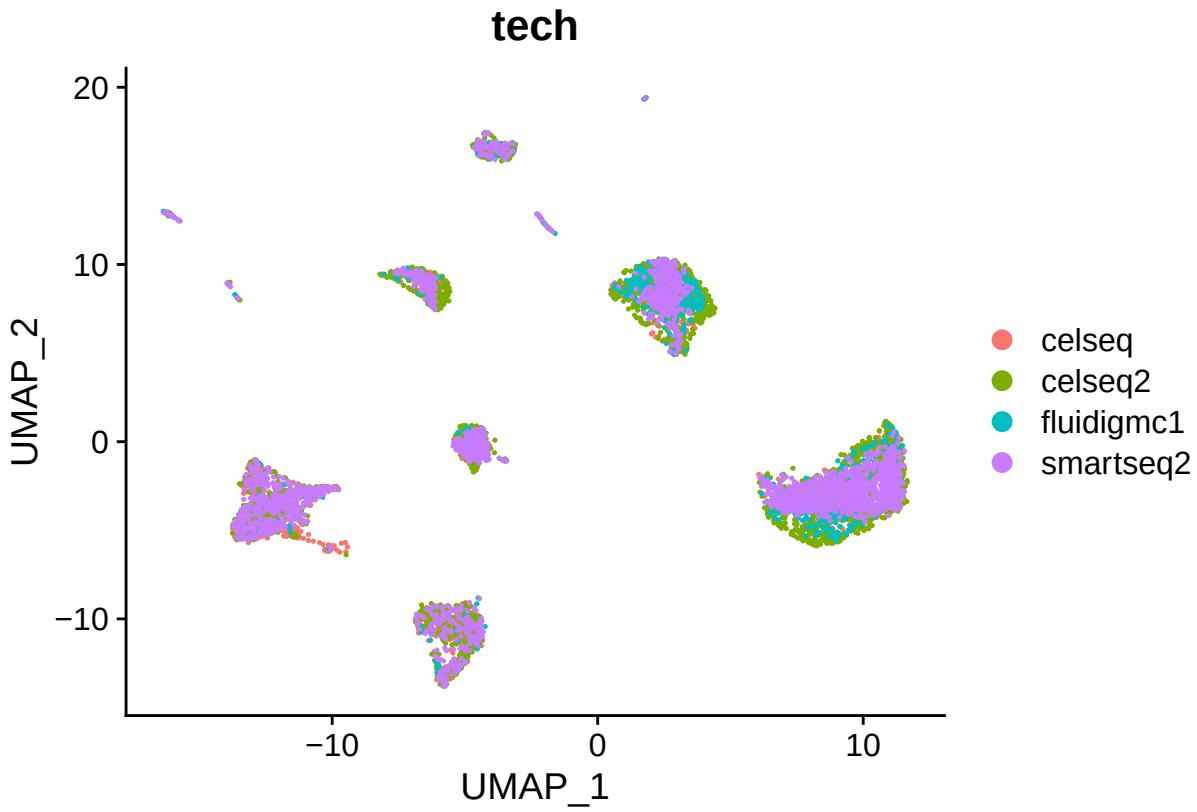
```
## PC_ 1
## Positive:  CHGA, VGF, PAM, MIR7-3HG, GAD2, ABCC8, PLCXD3, LOC100216479, CFC1, PEG10
##     G6PC2, RGS4, SLC38A4, RASD1, TM4SF4, GC, FEV, IRX2, CRYBA2, KCTD12
##     PCP4, ADRBK2, FAM105A, KCNK17, KCNK16, SGSM1, PCSK1, C2CD4A, KLHL41, PEMT
## Negative:  IFITM3, ZFP36L1, S100A11, TMSB4X, MYH9, SOX4, NFIB, TACSTD2, SH3BP4, TPM1
##     RBPMS, AHNAK, PMEPA1, LGALS3, CD44, CDC42EP1, LITAF, ANXA2, ANXA4, IFITM2
##     SPTBN1, ANXA2P2, SAT1, KRT7, OSMR, CLDN1, TM4SF1, BACE2, NOTCH2, SLC16A7
## PC_ 2
## Positive:  SPARC, COL15A1, MRC2, NID1, COL5A3, COL5A1, COL1A2, PDGFRB, COL5A2, LAMA4
##     COL3A1, COL6A2, COL6A3, BGN, LOXL2, FBN1, CYGB, CDH11, SFRP2, PRRX1
##     THBS2, COL1A1, WNT5A, LUM, COL4A1, COL12A1, LTBP2, ITGA11, IGFBP4, LHFP
## Negative:  CD24, ELF3, KRT8, TACSTD2, KIAA1522, CFB, ANXA4, KRT18, SERPINA3, TMC5
##     TC2N, KRT7, CXADR, SDC4, ABCC3, CLDN4, ATP10B, CLDN1, LAD1, PDZK1IP1
##     ATP1A1, ERBB3, GATM, CLDN10, C4orf19, TM4SF1, MUC20, SOD2, B3GNT7, LGR4
## PC_ 3
## Positive:  CPA2, PRSS3P2, PLA2G1B, PNLIP, KLK1, PRSS1, CTRB2, CPA1, CTRB1, CTRC
##     CEL, BCAT1, PNLIPRP2, GSTA2, REG1B, CPB1, CELA2A, ALB, PRSS3, CELA3A
##     PNLIPRP1, ALDOB, FAM129A, SPINK1, CBS, AOX1, REG1A, CELA2B, MGST1, SERPINI2
## Negative:  CFTR, IGFBP7, ONECUT2, ALDH1A3, VTCN1, AQP1, KRT19, SPP1, PPARGC1A, TINAGL1
##     DEFB1, TFPI2, SLC4A4, PKHD1, APCDD1, MMP7, THSD4, FGFR2, PMEPA1, SCNN1A
##     BICC1, RHPN2, SERPINA5, KRT23, PAQR5, S100A14, AKAP7, TGFB2, SERPING1, WWTR1
## PC_ 4
## Positive:  COL5A1, COL1A1, COL6A1, COL6A3, SFRP2, COL3A1, COL1A2, COL5A2, LTBP2, NOTCH3
##     PRRX1, COL5A3, PDGFRB, THBS2, ITGA11, CYGB, LUM, LAMA2, CDH11, COL12A1
##     TNFAIP6, EDNRA, CRLF1, WNT5A, GFPT2, FN1, COL6A2, FOXF2, LAMC3, SPON1
## Negative:  PECAM1, FLT1, KDR, SOX18, ELTD1, ESAM, CD93, LOC100505495, PLVAP, EXOC3L2
##     CALCRL, ACVRL1, COL13A1, RGCC, MMRN2, ECSCR, PODXL, GPR4, ERG, TIE1
##     NOTCH4, MYCT1, EMCN, CLEC14A, GIMAP8, PTPRB, GIMAP4, GPR116, S1PR1, GIMAP7
## PC_ 5
## Positive:  GC, TM4SF4, KCTD12, IRX2, FAP, FABP5, FEV, LOXL4, MUC13, KLHL41
##     RGS4, CRYBA2, PLCE1, SERPINE2, ARRDC4, SLC38A4, PPP2R2B, PYROXD2, PDK4, NPNT
##     DPP4, FAM84A, SPTSSB, SPC25, CD36, LDHA, EGFEM1P, GPC6, RAB27A, PAPPA2
## Negative:  HADH, PDX1, MAFA, SCD5, TGFBR3, NPTX2, INS, CASR, ADCYAP1, PCSK1
##     IAPP, PFKFB2, ENTPD3, RBP4, FFAR4, PCDH7, MEG3, ITPR3, PRSS23, VAT1L
##     DLK1, CABP7, FAM105A, BHLHE41, WSCD2, DHRS2, SRXN1, SMAD9, CYYR1, EXPH5
```

```
pancreas.integrated <- Seurat::RunUMAP(pancreas.integrated, reduction = "pca", dims = 1:30)
```

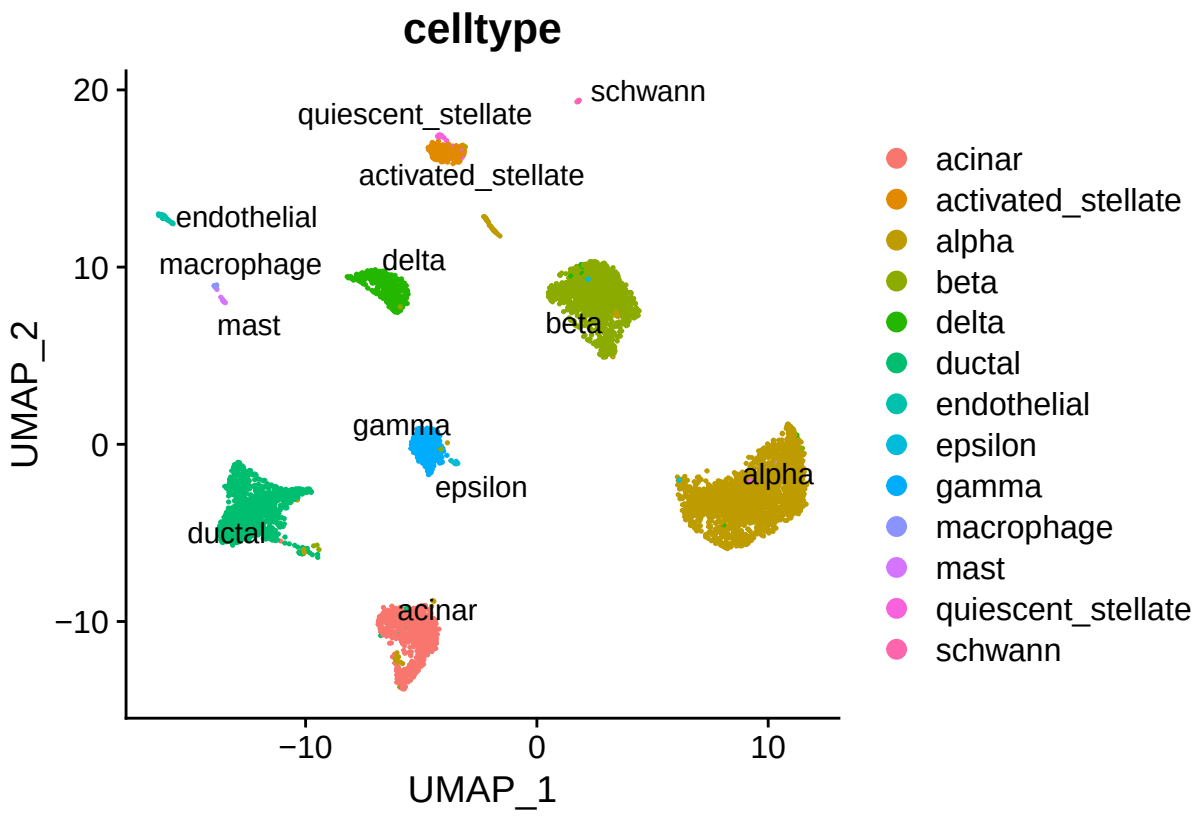
```
## 16:39:10 UMAP embedding parameters a = 0.9922 b = 1.112

## 16:39:10 Read 6321 rows and found 30 numeric columns

## 16:39:10 Using Annoy for neighbor search, n_neighbors = 30

## 16:39:10 Building Annoy index with metric = cosine, n_trees = 50

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

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

```
## **************************************************|
## 16:39:14 Writing NN index file to temp file C:\Users\imateusg\AppData\Local\Temp\RtmpYhrpl1\file1100
## 16:39:14 Searching Annoy index using 1 thread, search_k = 3000
## 16:39:22 Annoy recall = 100%
## 16:39:25 Commencing smooth kNN distance calibration using 1 thread
## 16:39:30 Found 2 connected components, falling back to 'spca' initialization with init_sdev = 1
## 16:39:30 Initializing from PCA
## 16:39:30 PCA: 2 components explained 45.1% variance
## 16:39:30 Commencing optimization for 500 epochs, with 282676 positive edges
## 16:40:54 Optimization finished
```

```
#Plotting the UMAP:

Seurat::DimPlot(pancreas.integrated, reduction = "umap", group.by = "tech")
```




```
Seurat::DimPlot(pancreas.integrated, reduction = "umap", group.by = "celltype", label = TRUE, repel = T
```


celltype
schwann
quiescent_stellate
activated_stellate
endothelial
macrophage
delta
mast
beta
gamma
epsilon
alpha
ductal
acinar
20
10
0
−10
UMAP_2
−10
0
10
UMAP_1
acinar
activated_stellate
alpha
beta
delta
ductal
endothelial
epsilon
gamma
macrophage
mast
quiescent_stellate
schwann