I used the same data about stressed mice

Usually i must import apropriate libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import scanpy as sc

Scanpy is my main tool for single cell expression analysis.

I upload my data from the same directory

dat = sc.read\_10x\_mtx(r"C:\Users\Виталик\Documents\RNAproject\10x genomics", var\_names="gene\_symbols")

I have renemed some files: barcodes, genes, y matrix, in my case 10Xread function cannot eat this files with anything else in name: like GCS\_barcodes.tsv, i should have renamed it to barcodes.tsv

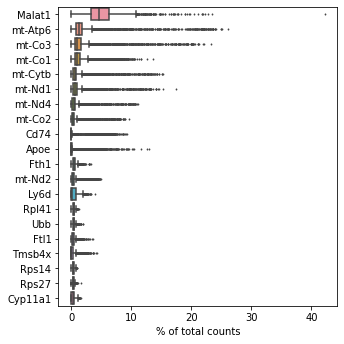
I should make variant names unique, for no count one gene as two or more name.

dat.var\_names\_make\_unique()

After, the pipeline tell me to look at 20 highest expression genes

The function estimates percent of counts exists for each gene and show me a plot

sc.pl.highest\_expr\_genes(dat, n\_top=20)



I could see a lot of genes with prefix “mt-” are mitochondrial genes which usually went from disrupted cells, and the lider is malat1 it is a long noncoding RNA.

Lets filter the set of genes with at least 200 genes

sc.pp.filter\_cells(dat, min\_genes=200)

And at least 3 cells

sc.pp.filter\_genes(dat, min\_cells=3)

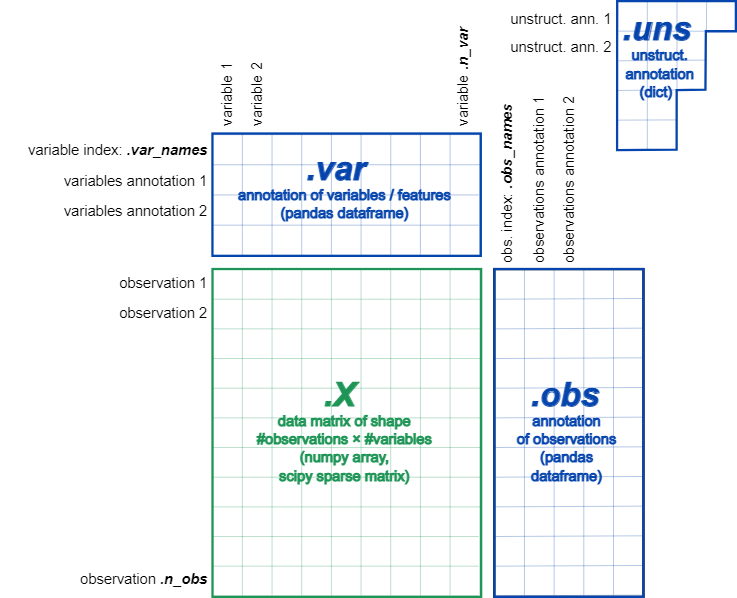
And make a set of mitochondrial genes

dat.var['mt'] = dat.var\_names.str.startswith('mt-')

And calculate quality control metrics

sc.pp.calculate\_qc\_metrics(dat, qc\_vars=['mt'], percent\_top=None, log1p=False, inplace=True)

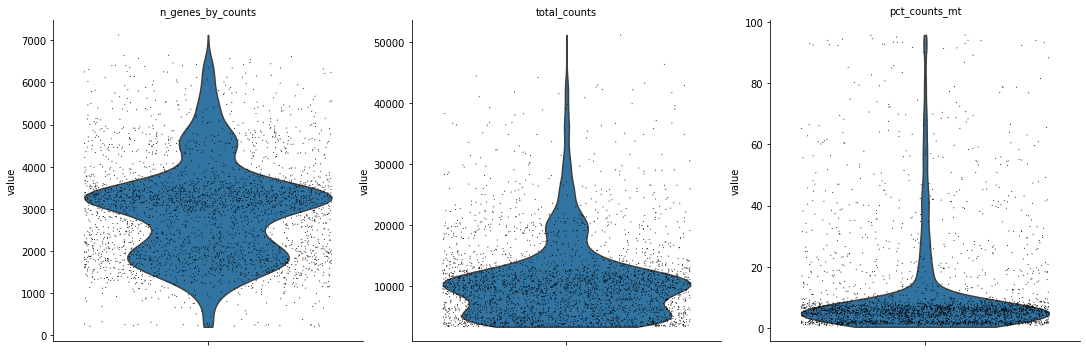
In this package data object is named AnnData, it means annotated data, and look at structure



And i observe Violin plots

sc.pl.violin(dat, ['n\_genes\_by\_counts', 'total\_counts', 'pct\_counts\_mt'],

jitter=0.4, multi\_panel=True)



I could saw the three standart distribution, and i wanna see the data frame

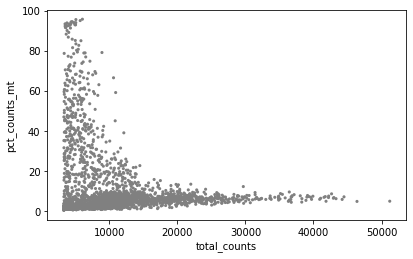
dat.obs

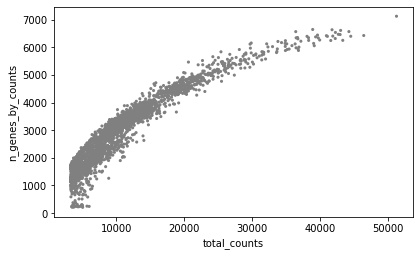
|  | **n\_genes** | **n\_genes\_by\_counts** | **total\_counts** | **total\_counts\_mt** | **pct\_counts\_mt** |
| --- | --- | --- | --- | --- | --- |
| **AAACCTGAGCGATAGC-1** | 1905 | 1905 | 7111.0 | 3809.0 | 53.564899 |
| **AAACCTGCACAGTCGC-1** | 3329 | 3323 | 10489.0 | 576.0 | 5.491467 |
| **AAACCTGGTTAAGGGC-1** | 3533 | 3533 | 11411.0 | 197.0 | 1.726404 |
| **AAACCTGTCCAGATCA-1** | 2238 | 2238 | 8326.0 | 188.0 | 2.257987 |
| **AAACCTGTCGTTGACA-1** | 3499 | 3499 | 11548.0 | 545.0 | 4.719432 |
| **...** | ... | ... | ... | ... | ... |
| **TTTGTCAAGCTTCGCG-1** | 2553 | 2553 | 9524.0 | 3282.0 | 34.460312 |
| **TTTGTCAAGGTGACCA-1** | 1124 | 1123 | 4014.0 | 1501.0 | 37.394123 |
| **TTTGTCACAATGTAAG-1** | 1430 | 1430 | 4532.0 | 1500.0 | 33.097969 |
| **TTTGTCATCCTTGCCA-1** | 3637 | 3636 | 12375.0 | 895.0 | 7.232323 |
| **TTTGTCATCGACCAGC-1** | 3622 | 3622 | 12698.0 | 1169.0 | 9.206174 |

So before the process of cutting some cells i should to observe counts and mitochondrial genes by plots

sc.pl.scatter(dat, x='total\_counts', y='pct\_counts\_mt')

sc.pl.scatter(dat, x='total\_counts', y='n\_genes\_by\_counts')





After i could cut some cells with too many counts and mitochondrial genes

adata = dat[dat.obs.n\_genes\_by\_counts < 4500, :]

adata = dat[dat.obs.pct\_counts\_mt < 25, :]

i wanna make threshhold for mitochondrial genes 25% and 4500 counts for genes,

and it is time for normalization by divided to 10000 reads per cell

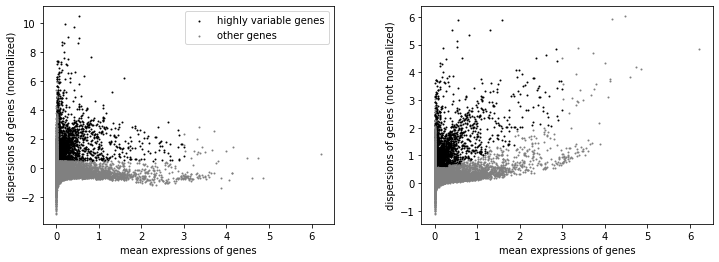
sc.pp.normalize\_total(adata, target\_sum=1e4)

and identify highly-variable genes.

sc.pp.highly\_variable\_genes(adata, min\_mean=0.0125, max\_mean=3, min\_disp=0.5)

I use minimal dispersion and mean by default and plot the result

sc.pl.highly\_variable\_genes(adata)



Regress out effects of total counts per cell and the percentage of mitochondrial genes expressed. Scale the data to unit variance.

sc.pp.regress\_out(adata, ['total\_counts', 'pct\_counts\_mt'])

How this function described in documentation:

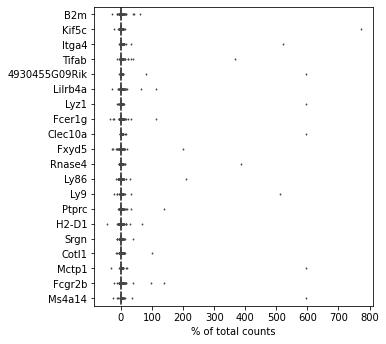
Uses simple linear regression. This is inspired by Seurat’s regressOut function in R [Satija15]. Note that this function tends to overcorrect in certain circumstances as described in [issue 526](https://github.com/scverse/scanpy/issues/526).

and scale each gene to unit variance. Clip values exceeding standard deviation 10.

sc.pp.scale(adata, max\_value=10)

After i wanna take a look at scaled and filtered genes, best 20

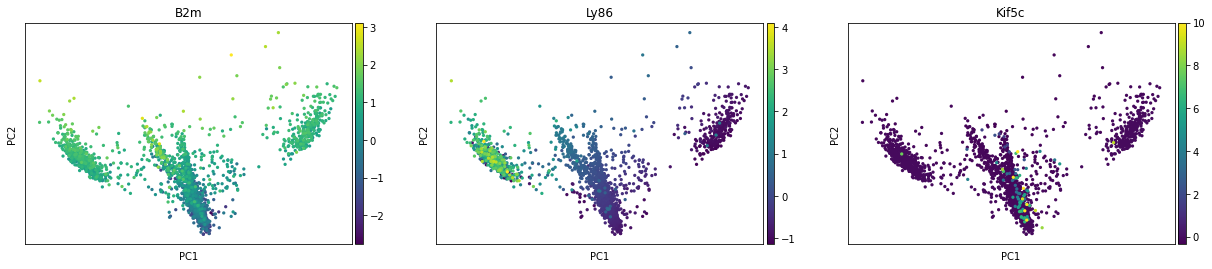
sc.pl.highest\_expr\_genes(adata, n\_top=20)



I make pca and look at three example genes

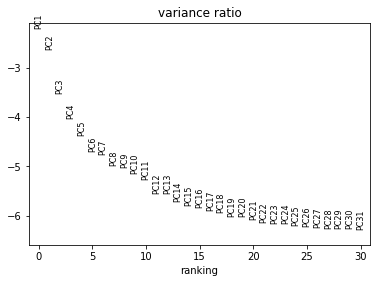
sc.tl.pca(adata, svd\_solver='arpack')

sc.pl.pca(adata, color=['B2m', 'Ly86', 'Kif5c'])



And make one pca elbow plot

sc.pl.pca\_variance\_ratio(adata, log=True)



I use 25 PCA for making neiborhood and umap graphic, and i plot umapplot with three genes, color by genes names.

sc.pp.neighbors(adata, n\_neighbors=10, n\_pcs=25)

and umap

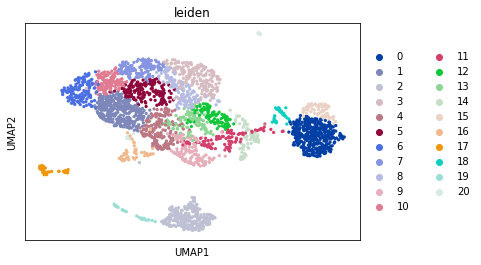
sc.tl.umap(adata)

sc.pl.umap(adata, color= ['B2m', 'Ly86', 'Kif5c'] )



Use leiden algorithm

sc.pl.umap(adata, color='leiden')

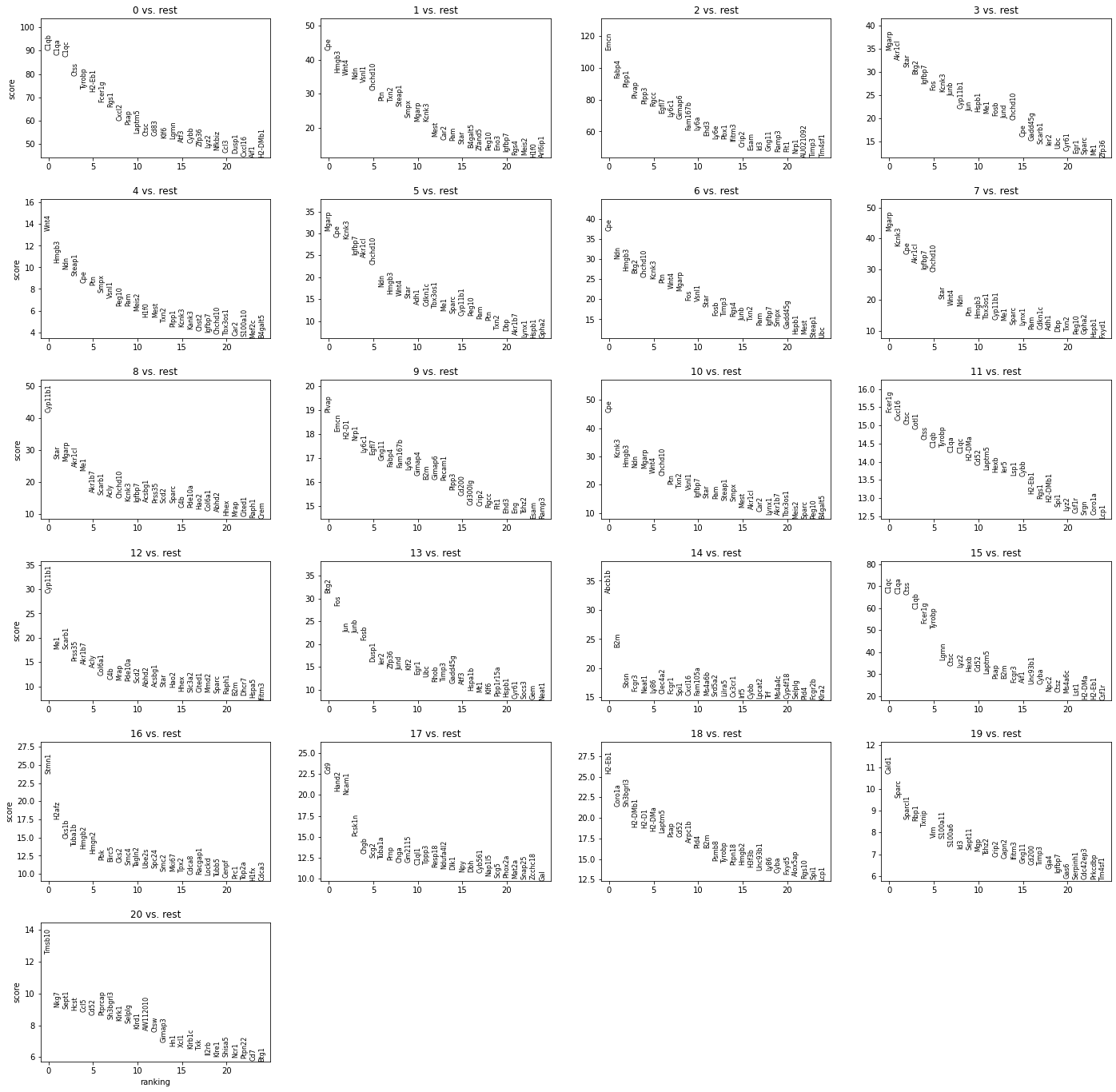


It is show us 20 classes, and

let us compute a ranking for the highly differential genes in each cluster.

sc.tl.rank\_genes\_groups(adata, 'leiden', method='t-test')

sc.pl.rank\_genes\_groups(adata, n\_genes=25, sharey=False)



So, trying to understand what it means, i could take a look at pandas dataframe

plots and table show genes with large expression of genes in each cluster that represents us cells, and for understanding which type of cell is each cluster we could use some genes as marker and annotate their manually.

sc.settings.verbosity = 2 # reduce the verbosity

as an alternative, let us rank genes using logistic regression

sc.tl.rank\_genes\_groups(adata, 'leiden', method='logreg')

sc.pl.rank\_genes\_groups(adata, n\_genes=25, sharey=False)

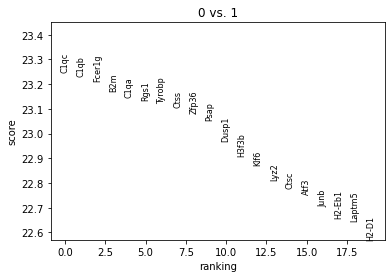
Show the 10 top ranked genes per cluster 0, 1, …, 7 in a dataframe.

pd.DataFrame(adata.uns['rank\_genes\_groups']['names']).head(5)

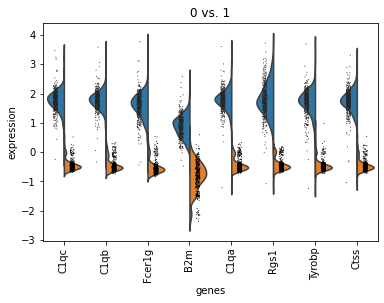
Compare to a single cluster:

sc.tl.rank\_genes\_groups(adata, 'leiden', groups=['0'], reference='1', method='wilcoxon')

sc.pl.rank\_genes\_groups(adata, groups=['0'], n\_genes=20)



sc.pl.rank\_genes\_groups\_violin(adata, groups='0', n\_genes=8)



If you want to compare a certain gene across groups, use the following.

sc.pl.violin(adata, ['Lyz2', 'B2m'], groupby='leiden')

sc.pl.umap(adata, color='leiden', legend\_loc='on data', title='', frameon=False, save='.pdf')

