Загрузка библиотек

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
```

```
import numpy as np
```

```
pip install -U scikit-learn
```

Показать скрытые выходные данные

```
import sklearn
```

```
pip install scanpy
```

Показать скрытые выходные данные

```
import scanpy as sc
```

```
import scipy as sp
```

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

```
import matplotlib.image as mpimg
```

```
import seaborn as sb
```

```
pip install  vaex
```

Показать скрытые выходные данные

```
pip install loompy
```

Показать скрытые выходные данные

```
import loompy
```

1.Загрузка

```
# загрузка файла формата loom   Вариант 1
adata = sc.read_loom('/content/drive/MyDrive/Cell_2/анализ/РМЖ.loom', sparse=True, cleanup=False,
                     X_name='spliced', obs_names='CellID', var_names='Gene')
```

```
gene_names = pd.read_csv('/content/drive/MyDrive/Cell_2/анализ/РМЖ гены.csv', index_col = 0)
named_list = adata.var.join(gene_names)
```

```
#загрузка файла формата csv    Вариант 2
data_d = pd.read_csv('/content/drive/MyDrive/Cell_2/анализ/PART_1CSV.csv')
data_d
```

```
     Unnamed: 0         PBMC3  PTEN  CDH1  PBMC1  PBMC1.1  BRCA2  BRCA1  ATM  BRCA1.1  ...  PBMC1.35  BRCA2.55  BRCA1.47  ATM.4
0    AU565_AAACCAGTTTGG   0.0   0.0   0.0    0.0      0.0    0.0    0.0  0.0      0.0  ...       0.0       0.0       0.0  0.
1    AU565_AAACGTGCAGCG   0.0   0.0   0.0    0.0      0.0    0.0    0.0  0.0      0.0  ...       0.0       0.0       0.0  0.
2    AU565_AAAGCCACATGC   0.0   0.0   0.0    0.0      0.0    0.0    0.0  0.0      0.0  ...       0.0       0.0       0.0  0.
3    AU565_AAAGTCGGCTGG   0.0   0.0   0.0    3.0      0.0    0.0    0.0  0.0      0.0  ...       0.0       0.0       0.0  1.
4    AU565_AAAGTGCCTAAA   1.0   7.0   0.0    0.0      1.0    0.0    0.0  0.0      0.0  ...       0.0       0.0       0.0  0.
...                 ...   ...   ...   ...    ...      ...    ...    ...  ...      ...  ...       ...       ...       ...  ..
583                 NaN   1.0   NaN   NaN    NaN      NaN    NaN    NaN  NaN      NaN  ...       NaN       NaN       NaN  Na
584                 NaN   NaN   NaN   NaN    NaN      NaN    NaN    NaN  NaN      NaN  ...       NaN       NaN       NaN  Na
585                 NaN   NaN   NaN   NaN    NaN      NaN    NaN    NaN  NaN      NaN  ...       NaN       NaN       NaN  Na
586                 NaN   NaN   NaN   NaN    NaN      NaN    NaN    NaN  NaN      NaN  ...       NaN       NaN       NaN  Na
587                 NaN   NaN   NaN   NaN    NaN      NaN    NaN    NaN  NaN      NaN  ...       NaN       NaN       NaN  Na

588 rows × 256 columns
```

```python
data_m = pd.read_csv('/content/drive/MyDrive/Cell_2/анализ/breast-cancer (1).csv')
data_m
```

```
     Gene             Index_Cell           organism       id         Cell_type    radius_mean  texture_mean  perimeter_mean  area_mean
0    ENSG00000006634  AU565_ATAACAGCCACT   Human Breast   89511501   Basal cells        12.20         15.21           78.01      457.9
1    ENSG00000006652  AU565_ATAAGGGCTGAC   Human Breast   8911800    Basal cells        13.59         17.84           86.24      572.3
2    ENSG00000006695  AU565_ATACCGCACACA   Human Breast   924084     Basal cells        12.77         29.43           81.35      507.9
3    ENSG00000006704  AU565_ATACGCGACACA   Human Breast   869218     Basal cells        11.43         17.31           73.66      398.0
4    ENSG00000006712  AU565_ATAGACTCGCAG   Human Breast   902976     Basal cells        13.88         16.16           88.37      596.6
...  ...              ...                  ...            ...        ...                  ...           ...             ...        ...
565  ENSG00000063438  AU565_TGTGTCTGAGCG   Human Breast   927241     Unknown            20.60         29.33          140.10     1265.0
566  ENSG00000063587  AU565_TGTGTTCTACGT   Human Breast   878796     Unknown            23.29         26.67          158.90     1685.0
567  ENSG00000063601  AU565_TGTTACAATACG   Human Breast   911296202  Unknown            27.42         26.27          186.90     2501.0
568  ENSG00000063660  AU565_TGTTTCACCTGA   Human Breast   8610862    Unknown            20.18         23.97          143.70     1245.0
569  ENSG00000063854  AU565_TTAATGAGGACG   Human Breast   86355      Unknown            22.27         19.67          152.80     1509.0

570 rows × 36 columns
```

2.1 Анализ данных

Для данных только формата csv

Необходимо данные объеденить в один файл

```python
data_d.head()
```

	Unnamed: 0	PBMC3	PTEN	CDH1	PBMC1	PBMC1.1	BRCA2	BRCA1	ATM	BRCA1.1	...	PBMC1.35	BRCA2.55	BRCA1.47	ATM.42
0	AU565_AAACCAGTTTGG	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1	AU565_AAACGTGCAGCG	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
2	AU565_AAAGCCACATGC	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
3	AU565_AAAGTCGGCTGG	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	1.0
4	AU565_AAAGTGCCTAAA	1.0	7.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0

5 rows × 256 columns

Строки в столбцы

df = df.T

df.columns = list("ABCD")

```
#Если необходимо поменять местами столбцы и строки в исходных данных
print(data_d.T)
```

```
                             0                   1                   2   \
Unnamed: 0  AU565_AAACCAGTTTGG  AU565_AAACGTGCAGCG  AU565_AAAGCCACATGC   
PBMC3                      0.0                 0.0                 0.0   
PTEN                       0.0                 0.0                 0.0   
CDH1                       0.0                 0.0                 0.0   
PBMC1                      0.0                 0.0                 0.0   
...                        ...                 ...                 ...   
ATM.43                     0.0                 0.0                 0.0   
ATM.44                     0.0                 0.0                 0.0   
ATM.45                     0.0                 0.0                 0.0   
ATM.46                     0.0                 0.0                 0.0   
ATM.47                     0.0                 0.0                 0.0   

                             3                   4                   5   \
Unnamed: 0  AU565_AAAGTCGGCTGG  AU565_AAAGTGCCTAAA  AU565_AAATCATCTTAT   
PBMC3                      0.0                 1.0                 2.0   
PTEN                       0.0                 7.0                 0.0   
CDH1                       0.0                 0.0                 0.0   
PBMC1                      3.0                 0.0                 0.0   
...                        ...                 ...                 ...   
ATM.43                     0.0                 0.0                 1.0   
ATM.44                     0.0                 0.0                 0.0   
ATM.45                     0.0                 0.0                 0.0   
ATM.46                     0.0                 0.0                 1.0   
ATM.47                     0.0                 0.0                 0.0   

                             6                   7                   8   \
Unnamed: 0  AU565_AAATCATTGTCT  AU565_AAATTACTTCAT  AU565_AAATTATTAGAT   
PBMC3                      0.0                 0.0                 0.0   
PTEN                       0.0                 0.0                 0.0   
CDH1                       0.0                 0.0                 0.0   
PBMC1                      0.0                 0.0                 0.0   
...                        ...                 ...                 ...   
ATM.43                     0.0                 0.0                 0.0   
ATM.44                     0.0                 0.0                 0.0   
ATM.45                     0.0                 0.0                 0.0   
ATM.46                     0.0                 0.0                 0.0   
ATM.47                     0.0                 0.0                 0.0   

                             9   ...  578  579  580  581  582  583  584  585  \
Unnamed: 0  AU565_AAATTGATTTGT  ...  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN   
PBMC3                      0.0  ...  NaN  NaN  NaN  NaN  NaN  1.0  NaN  NaN   
PTEN                       1.0  ...  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN   
CDH1                       0.0  ...  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN   
PBMC1                      2.0  ...  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN   
...                        ...  ...  ...  ...  ...  ...  ...  ...  ...  ...   
ATM.43                     0.0  ...  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN   
ATM.44                     0.0  ...  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN   
ATM.45                     1.0  ...  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN   
ATM.46                     0.0  ...  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN   
ATM.47                     0.0  ...  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN   

            586  587  
Unnamed: 0  NaN  NaN  
PBMC3       NaN  NaN  
PTEN        NaN  NaN  
CDH1        NaN  NaN  
PBMC1       NaN  NaN  
```

```
data_d.columns = ['Index_Cell', *data_d.columns[1:]]
```

```
data_d.T.rename(columns=data_d.T.iloc[0]).drop(data_d.T.index[0])
```

	AU565_AAACCAGTTTGG	AU565_AAACGTGCAGCG	AU565_AAAGCCACATGC	AU565_AAAGTCGGCTGG	AU565_AAAGTGCCTAAA	AU565_AAATCATCTTAT	AU
PBMC3	0.0	0.0	0.0	0.0	1.0	2.0	
PTEN	0.0	0.0	0.0	0.0	7.0	0.0	
CDH1	0.0	0.0	0.0	0.0	0.0	0.0	
PBMC1	0.0	0.0	0.0	3.0	0.0	0.0	
PBMC1.1	0.0	0.0	0.0	0.0	1.0	0.0	
...	...	...	...	...	...	...	
ATM.43	0.0	0.0	0.0	0.0	0.0	1.0	
ATM.44	0.0	0.0	0.0	0.0	0.0	0.0	
ATM.45	0.0	0.0	0.0	0.0	0.0	0.0	
ATM.46	0.0	0.0	0.0	0.0	0.0	1.0	
ATM.47	0.0	0.0	0.0	0.0	0.0	0.0	

255 rows × 588 columns

```
data_d=data_d.T
```

```
data_d
```

	0	1	2	3	4	
Index_Cell	AU565_AAACCAGTTTGG	AU565_AAACGTGCAGCG	AU565_AAAGCCACATGC	AU565_AAAGTCGGCTGG	AU565_AAAGTGCCTAAA	A
PBMC3	0.0	0.0	0.0	0.0	1.0	
PTEN	0.0	0.0	0.0	0.0	7.0	
CDH1	0.0	0.0	0.0	0.0	0.0	
PBMC1	0.0	0.0	0.0	3.0	0.0	
...	...	...	...	...	...	
ATM.43	0.0	0.0	0.0	0.0	0.0	
ATM.44	0.0	0.0	0.0	0.0	0.0	
ATM.45	0.0	0.0	0.0	0.0	0.0	
ATM.46	0.0	0.0	0.0	0.0	0.0	
ATM.47	0.0	0.0	0.0	0.0	0.0	

256 rows × 588 columns

```
metadata = pd.concat([data_m], axis=0).drop_duplicates()
metadata = data_m.drop_duplicates(subset='Index_Cell', keep="last")
metadata
```

	Gene	Index_Cell	organism	id	Cell_type	radius_mean	texture_mean	perimeter_mean	area_mean
0	ENSG00000006634	AU565_ATAACAGCCACT	Human Breast	89511501	Basal cells	12.20	15.21	78.01	457.9
1	ENSG00000006652	AU565_ATAAGGGCTGAC	Human Breast	8911800	Basal cells	13.59	17.84	86.24	572.3
2	ENSG00000006695	AU565_ATACCGCACACA	Human Breast	924084	Basal cells	12.77	29.43	81.35	507.9
3	ENSG00000006704	AU565_ATACGCGACACA	Human Breast	869218	Basal cells	11.43	17.31	73.66	398.0
4	ENSG00000006712	AU565_ATAGACTCGCAG	Human Breast	902976	Basal cells	13.88	16.16	88.37	596.6
...	...	...	...	...	...	...	...	...	...
565	ENSG00000063438	AU565_TGTGTCTGAGCG	Human Breast	927241	Unknown	20.60	29.33	140.10	1265.0
566	ENSG00000063587	AU565_TGTGTTCTACGT	Human Breast	878796	Unknown	23.29	26.67	158.90	1685.0
567	ENSG00000063601	AU565_TGTTACAATACG	Human Breast	911296202	Unknown	27.42	26.27	186.90	2501.0
568	ENSG00000063660	AU565_TGTTTCACCTGA	Human Breast	8610862	Unknown	20.18	23.97	143.70	1245.0
569	ENSG00000063854	AU565_TTAATGAGGACG	Human Breast	86355	Unknown	22.27	19.67	152.80	1509.0

570 rows × 36 columns

```
#Если одинаковые таблицы по наполнению, то их можно объеденить
data_d = sc.AnnData(data_d)
data_m.index = metadata.Index_Cell.map(str)
data_m.obs = metadata
adata
```

```
AnnData object with n_obs × n_vars = 2544 × 58347
    obs: 'ACCUMULATION_LEVEL', 'ALIGNED_READS', 'AT_DROPOUT', 'Aligned 0 time', 'Aligned 1 time', 'Aligned >1 times', 'Aligned
concordantly 1 time', 'Aligned concordantly >1 times', 'Aligned concordantly or discordantly 0 t', 'Aligned discordantly 1 time',
'BAD_CYCLES.FIRST_OF_PAIR', 'BAD_CYCLES.PAIR', 'BAD_CYCLES.SECOND_OF_PAIR', 'CODING_BASES', 'CORRECT_STRAND_READS',
'ESTIMATED_LIBRARY_SIZE', 'GC_DROPOUT', 'GC_NC_0_19', 'GC_NC_20_39', 'GC_NC_40_59', 'GC_NC_60_79', 'GC_NC_80_100', 'IGNORED_READS',
'INCORRECT_STRAND_READS', 'INTERGENIC_BASES', 'INTRONIC_BASES', 'MAX_INSERT_SIZE', 'MEAN_INSERT_SIZE',
'MEAN_READ_LENGTH.FIRST_OF_PAIR', 'MEAN_READ_LENGTH.PAIR', 'MEAN_READ_LENGTH.SECOND_OF_PAIR', 'MEDIAN_3PRIME_BIAS',
'MEDIAN_5PRIME_BIAS', 'MEDIAN_5PRIME_TO_3PRIME_BIAS', 'MEDIAN_ABSOLUTE_DEVIATION', 'MEDIAN_CV_COVERAGE', 'MEDIAN_INSERT_SIZE',
'MIN_INSERT_SIZE', 'NUM_R1_TRANSCRIPT_STRAND_READS', 'NUM_R2_TRANSCRIPT_STRAND_READS', 'NUM_UNEXPLAINED_READS', 'Overall alignment
rate', 'PAIR_ORIENTATION', 'PCT_ADAPTER.FIRST_OF_PAIR', 'PCT_ADAPTER.PAIR', 'PCT_ADAPTER.SECOND_OF_PAIR',
'PCT_CHIMERAS.FIRST_OF_PAIR', 'PCT_CHIMERAS.PAIR', 'PCT_CHIMERAS.SECOND_OF_PAIR', 'PCT_CODING_BASES', 'PCT_CORRECT_STRAND_READS',
'PCT_INTERGENIC_BASES', 'PCT_INTRONIC_BASES', 'PCT_MRNA_BASES', 'PCT_PF_READS.FIRST_OF_PAIR', 'PCT_PF_READS.PAIR',
'PCT_PF_READS.SECOND_OF_PAIR', 'PCT_PF_READS_ALIGNED.FIRST_OF_PAIR', 'PCT_PF_READS_ALIGNED.PAIR',
'PCT_PF_READS_ALIGNED.SECOND_OF_PAIR', 'PCT_PF_READS_IMPROPER_PAIRS.FIRST_OF_PAI', 'PCT_PF_READS_IMPROPER_PAIRS.PAIR',
'PCT_PF_READS_IMPROPER_PAIRS.SECOND_OF_PA', 'PCT_R1_TRANSCRIPT_STRAND_READS', 'PCT_R2_TRANSCRIPT_STRAND_READS',
'PCT_READS_ALIGNED_IN_PAIRS.FIRST_OF_PAIR', 'PCT_READS_ALIGNED_IN_PAIRS.PAIR', 'PCT_READS_ALIGNED_IN_PAIRS.SECOND_OF_PAI',
'PCT_RIBOSOMAL_BASES', 'PCT_USABLE_BASES', 'PCT_UTR_BASES', 'PERCENT_DUPLICATION', 'PF_ALIGNED_BASES',
'PF_ALIGNED_BASES.FIRST_OF_PAIR', 'PF_ALIGNED_BASES.PAIR', 'PF_ALIGNED_BASES.SECOND_OF_PAIR', 'PF_BASES',
'PF_HQ_ALIGNED_BASES.FIRST_OF_PAIR', 'PF_HQ_ALIGNED_BASES.PAIR', 'PF_HQ_ALIGNED_BASES.SECOND_OF_PAIR',
'PF_HQ_ALIGNED_Q20_BASES.FIRST_OF_PAIR', 'PF_HQ_ALIGNED_Q20_BASES.PAIR', 'PF_HQ_ALIGNED_Q20_BASES.SECOND_OF_PAIR',
'PF_HQ_ALIGNED_READS.FIRST_OF_PAIR', 'PF_HQ_ALIGNED_READS.PAIR', 'PF_HQ_ALIGNED_READS.SECOND_OF_PAIR',
'PF_HQ_ERROR_RATE.FIRST_OF_PAIR', 'PF_HQ_ERROR_RATE.PAIR', 'PF_HQ_ERROR_RATE.SECOND_OF_PAIR',
'PF_HQ_MEDIAN_MISMATCHES.FIRST_OF_PAIR', 'PF_HQ_MEDIAN_MISMATCHES.PAIR', 'PF_HQ_MEDIAN_MISMATCHES.SECOND_OF_PAIR',
'PF_INDEL_RATE.FIRST_OF_PAIR', 'PF_INDEL_RATE.PAIR', 'PF_INDEL_RATE.SECOND_OF_PAIR', 'PF_MISMATCH_RATE.FIRST_OF_PAIR',
'PF_MISMATCH_RATE.PAIR', 'PF_MISMATCH_RATE.SECOND_OF_PAIR', 'PF_NOISE_READS.FIRST_OF_PAIR', 'PF_NOISE_READS.PAIR',
'PF_NOISE_READS.SECOND_OF_PAIR', 'PF_READS.FIRST_OF_PAIR', 'PF_READS.PAIR', 'PF_READS.SECOND_OF_PAIR',
'PF_READS_ALIGNED.FIRST_OF_PAIR', 'PF_READS_ALIGNED.PAIR', 'PF_READS_ALIGNED.SECOND_OF_PAIR',
'PF_READS_IMPROPER_PAIRS.FIRST_OF_PAIR', 'PF_READS_IMPROPER_PAIRS.PAIR', 'PF_READS_IMPROPER_PAIRS.SECOND_OF_PAIR',
'READS_ALIGNED_IN_PAIRS.FIRST_OF_PAIR', 'READS_ALIGNED_IN_PAIRS.PAIR', 'READS_ALIGNED_IN_PAIRS.SECOND_OF_PAIR', 'READS_USED',
'READ_PAIRS', 'READ_PAIRS_EXAMINED', 'READ_PAIR_DUPLICATES', 'READ_PAIR_OPTICAL_DUPLICATES', 'RIBOSOMAL_BASES',
'SECONDARY_OR_SUPPLEMENTARY_RDS', 'STANDARD_DEVIATION', 'STRAND_BALANCE.FIRST_OF_PAIR', 'STRAND_BALANCE.PAIR',
'STRAND_BALANCE.SECOND_OF_PAIR', 'TOTAL_CLUSTERS', 'TOTAL_READS.FIRST_OF_PAIR', 'TOTAL_READS.PAIR', 'TOTAL_READS.SECOND_OF_PAIR',
'Total pairs', 'Total unpaired reads', 'UNMAPPED_READS', 'UNPAIRED_READS_EXAMINED', 'UNPAIRED_READ_DUPLICATES', 'UTR_BASES',
'WIDTH_OF_10_PERCENT', 'WIDTH_OF_20_PERCENT', 'WIDTH_OF_30_PERCENT', 'WIDTH_OF_40_PERCENT', 'WIDTH_OF_50_PERCENT',
'WIDTH_OF_60_PERCENT', 'WIDTH_OF_70_PERCENT', 'WIDTH_OF_80_PERCENT', 'WIDTH_OF_90_PERCENT', 'WIDTH_OF_99_PERCENT', 'WINDOW_SIZE',
'alignable reads', 'filtered reads', 'multiple mapped', 'strand', 'total alignments', 'total reads', 'unalignable reads',
'uncertain reads', 'unique aligned'
```

Удалить NaN

```
data_d3 = data_d.fillna(0)
```

Показать скрытые выходные данные

```
data_m.nunique()
```

```
Gene                       570
Index_Cell                 570
organism                     1
id                         570
Cell_type                    7
radius_mean                455
texture_mean               479
perimeter_mean             521
area_mean                  538
smoothness_mean            473
compactness_mean           536
concavity_mean             538
Cluster                      8
concave points_mean        541
symmetry_mean              431
fractal_dimension_mean     499
radius_se                  539
texture_se                 518
perimeter_se               532
area_se                    527
smoothness_se              546
compactness_se             542
concavity_se               532
concave points_se          506
symmetry_se                497
fractal_dimension_se       544
radius_worst               456
texture_worst              510
perimeter_worst            513
area_worst                 543
smoothness_worst           410
compactness_worst          528
concavity_worst            538
concave points_worst       492
symmetry_worst             499
fractal_dimension_worst    535
dtype: int64
```

```
data_m.dtypes
```

```
Gene                        object
Index_Cell                  object
organism                    object
id                           int64
Cell_type                   object
radius_mean                float64
texture_mean               float64
perimeter_mean             float64
area_mean                  float64
smoothness_mean            float64
compactness_mean           float64
concavity_mean             float64
Cluster                      int64
concave points_mean        float64
symmetry_mean              float64
fractal_dimension_mean     float64
radius_se                  float64
texture_se                 float64
perimeter_se               float64
area_se                    float64
smoothness_se              float64
compactness_se             float64
concavity_se               float64
concave points_se          float64
symmetry_se                float64
fractal_dimension_se       float64
radius_worst               float64
texture_worst              float64
perimeter_worst            float64
area_worst                 float64
smoothness_worst           float64
compactness_worst          float64
concavity_worst            float64
concave points_worst       float64
symmetry_worst             float64
fractal_dimension_worst    float64
dtype: object
```

```
adata = sc.AnnData(data_m)
adata.obs = data_m
adata
```

```
AnnData object with n_obs × n_vars = 570 × 36
    obs: 'Gene', 'Index_Cell', 'organism', 'id', 'Cell_type', 'radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean',
'smoothness_mean', 'compactness_mean', 'concavity_mean', 'Cluster', 'concave points_mean', 'symmetry_mean',
'fractal_dimension_mean', 'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se', 'compactness_se', 'concavity_se',
'concave points_se', 'symmetry_se', 'fractal_dimension_se', 'radius_worst', 'texture_worst', 'perimeter_worst', 'area_worst',
'smoothness_worst', 'compactness_worst', 'concavity_worst', 'concave points_worst', 'symmetry_worst', 'fractal_dimension_worst'
```

```
data_m['Cell_type'].replace('PBMC2_1','PBMC2',inplace=True)
data_m['Cell_type'].replace('PBMC1_1','PBMC1',inplace=True)
data_m['Cell_type'].replace('PBMC3_1','PBMC3',inplace=True)
```

```
adata.obs['Cell_type'].value_counts()
```

```
Cell_type
Basal cells      204
Unknown          135
Ductal cells      64
Cholangiocytes    48
malignant         43
Fibroblasts       39
benign            37
Name: count, dtype: int64
```

```
#Выбираем те клетки, которые нам нужнгы для дальнейшего анализа
 adata = adata[adata.obs['Cell_type'].isin(['Basal cells',
'Unknown',
'malignant',
'Ductal cells'
])]
```

Показать скрытые выходные данные

```
adata.obs['Cell_type'].value_counts()
```

```
Cell_type
Basal cells      204
Unknown          135
Ductal cells      64
Cholangiocytes    48
malignant         43
Fibroblasts       39
benign            37
Name: count, dtype: int64
```

```
pip install -U scikit-learn
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.5.0)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.25.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.5.0)
```

```
#Нормализация данных
adata.obs['n_counts'] = adata1.X.sum(1)
adata.obs['log_counts'] = np.log(adata_N.obs['n_counts'])
adata.obs['n_genes'] = (adata_N.X > 0).sum(1)
#с помощью машинного обучения X_normalized = preprocessing.normalize(X, norm='l2')
```

Показать скрытые выходные данные

После данного этапа можно строить различные типа графиков

Фильтруем и нормализцем данные

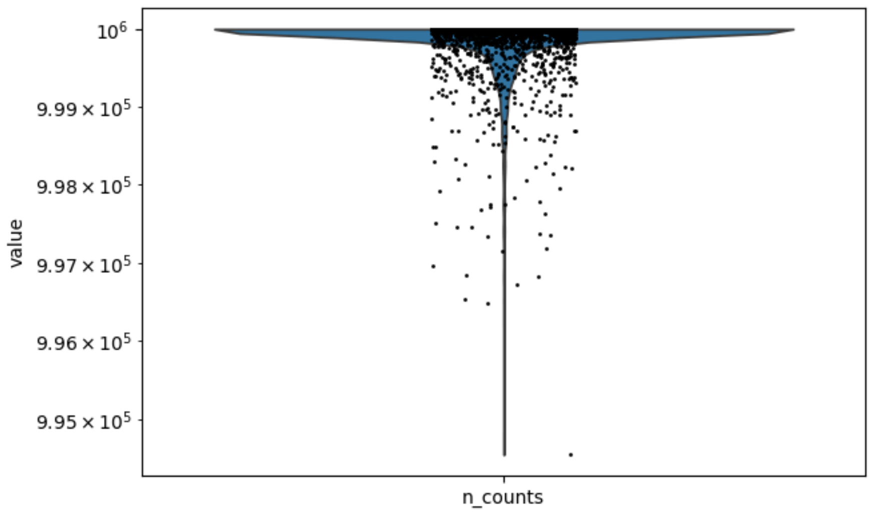
для данных формата csv и loom

```
sc.pp.filter_genes(adata, min_cells=2)
```

```
sc.pp.filter_cells(adata, min_genes=100)
```

```
adata.obs['n_counts'] = np.sum(adata.X, axis=1).A1
```

```
sc.pl.violin(adata, 'n_counts', size=2, log=True, cut=0)
```



```
dat = sc.pp.normalize_per_cell(adata, counts_per_cell_after=1e4, copy=True)
dat
```

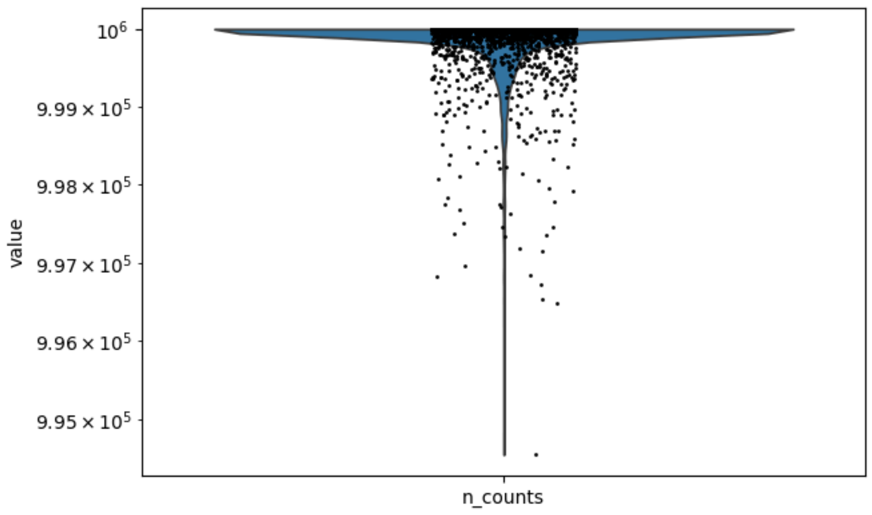
```
AnnData object with n_obs × n_vars = 2544 × 37562
    obs: 'ACCUMULATION_LEVEL', 'ALIGNED_READS', 'AT_DROPOUT', 'Aligned 0 time', 'Aligned 1 time', 'Aligned >1 times', 'Aligned concordantly 1 time', 'Aligned concordantly >1 times', 'Aligned concordantly or discordantly 0 t', 'Aligned discordantly 1 time', 'BAD_CYCLES.FIRST_OF_PAIR', 'BAD_CYCLES.PAIR', 'BAD_CYCLES.SECOND_OF_PAIR', 'CODING_BASES', 'CORRECT_STRAND_READS', 'ESTIMATED_LIBRARY_SIZE', 'GC_DROPOUT', 'GC_NC_0_19', 'GC_NC_20_39', 'GC_NC_40_59', 'GC_NC_60_79', 'GC_NC_80_100', 'IGNORED_READS', 'INCORRECT_STRAND_READS', 'INTERGENIC_BASES', 'INTRONIC_BASES', 'MAX_INSERT_SIZE', 'MEAN_INSERT_SIZE', 'MEAN_READ_LENGTH.FIRST_OF_PAIR', 'MEAN_READ_LENGTH.PAIR', 'MEAN_READ_LENGTH.SECOND_OF_PAIR', 'MEDIAN_3PRIME_BIAS', 'MEDIAN_5PRIME_BIAS', 'MEDIAN_5PRIME_TO_3PRIME_BIAS', 'MEDIAN_ABSOLUTE_DEVIATION', 'MEDIAN_CV_COVERAGE', 'MEDIAN_INSERT_SIZE', 'MIN_INSERT_SIZE', 'NUM_R1_TRANSCRIPT_STRAND_READS', 'NUM_R2_TRANSCRIPT_STRAND_READS', 'NUM_UNEXPLAINED_READS', 'Overall alignment rate', 'PAIR_ORIENTATION', 'PCT_ADAPTER.FIRST_OF_PAIR', 'PCT_ADAPTER.PAIR', 'PCT_ADAPTER.SECOND_OF_PAIR', 'PCT_CHIMERAS.FIRST_OF_PAIR', 'PCT_CHIMERAS.PAIR', 'PCT_CHIMERAS.SECOND_OF_PAIR', 'PCT_CODING_BASES', 'PCT_CORRECT_STRAND_READS', 'PCT_INTERGENIC_BASES', 'PCT_INTRONIC_BASES', 'PCT_MRNA_BASES', 'PCT_PF_READS.FIRST_OF_PAIR', 'PCT_PF_READS.PAIR', 'PCT_PF_READS.SECOND_OF_PAIR', 'PCT_PF_READS_ALIGNED.FIRST_OF_PAIR', 'PCT_PF_READS_ALIGNED.PAIR', 'PCT_PF_READS_ALIGNED.SECOND_OF_PAIR', 'PCT_PF_READS_IMPROPER_PAIRS.FIRST_OF_PAI', 'PCT_PF_READS_IMPROPER_PAIRS.PAIR', 'PCT_PF_READS_IMPROPER_PAIRS.SECOND_OF_PA', 'PCT_R1_TRANSCRIPT_STRAND_READS', 'PCT_R2_TRANSCRIPT_STRAND_READS', 'PCT_READS_ALIGNED_IN_PAIRS.FIRST_OF_PAIR', 'PCT_READS_ALIGNED_IN_PAIRS.PAIR', 'PCT_READS_ALIGNED_IN_PAIRS.SECOND_OF_PAI', 'PCT_RIBOSOMAL_BASES', 'PCT_USABLE_BASES', 'PCT_UTR_BASES', 'PERCENT_DUPLICATION', 'PF_ALIGNED_BASES', 'PF_ALIGNED_BASES.FIRST_OF_PAIR', 'PF_ALIGNED_BASES.PAIR', 'PF_ALIGNED_BASES.SECOND_OF_PAIR', 'PF_BASES', 'PF_HQ_ALIGNED_BASES.FIRST_OF_PAIR', 'PF_HQ_ALIGNED_BASES.PAIR', 'PF_HQ_ALIGNED_BASES.SECOND_OF_PAIR', 'PF_HQ_ALIGNED_Q20_BASES.FIRST_OF_PAIR', 'PF_HQ_ALIGNED_Q20_BASES.PAIR', 'PF_HQ_ALIGNED_Q20_BASES.SECOND_OF_PAIR', 'PF_HQ_ALIGNED_READS.FIRST_OF_PAIR', 'PF_HQ_ALIGNED_READS.PAIR', 'PF_HQ_ALIGNED_READS.SECOND_OF_PAIR', 'PF_HQ_ERROR_RATE.FIRST_OF_PAIR', 'PF_HQ_ERROR_RATE.PAIR', 'PF_HQ_ERROR_RATE.SECOND_OF_PAIR', 'PF_HQ_MEDIAN_MISMATCHES.FIRST_OF_PAIR', 'PF_HQ_MEDIAN_MISMATCHES.PAIR', 'PF_HQ_MEDIAN_MISMATCHES.SECOND_OF_PAIR', 'PF_INDEL_RATE.FIRST_OF_PAIR', 'PF_INDEL_RATE.PAIR', 'PF_INDEL_RATE.SECOND_OF_PAIR', 'PF_MISMATCH_RATE.FIRST_OF_PAIR', 'PF_MISMATCH_RATE.PAIR', 'PF_MISMATCH_RATE.SECOND_OF_PAIR', 'PF_NOISE_READS.FIRST_OF_PAIR', 'PF_NOISE_READS.PAIR', 'PF_NOISE_READS.SECOND_OF_PAIR', 'PF_READS.FIRST_OF_PAIR', 'PF_READS.PAIR', 'PF_READS.SECOND_OF_PAIR', 'PF_READS_ALIGNED.FIRST_OF_PAIR', 'PF_READS_ALIGNED.PAIR', 'PF_READS_ALIGNED.SECOND_OF_PAIR', 'PF_READS_IMPROPER_PAIRS.FIRST_OF_PAIR', 'PF_READS_IMPROPER_PAIRS.PAIR', 'PF_READS_IMPROPER_PAIRS.SECOND_OF_PAIR', 'READS_ALIGNED_IN_PAIRS.FIRST_OF_PAIR', 'READS_ALIGNED_IN_PAIRS.PAIR', 'READS_ALIGNED_IN_PAIRS.SECOND_OF_PAIR', 'READS_USED', 'READ_PAIRS', 'READ_PAIRS_EXAMINED', 'READ_PAIR_DUPLICATES', 'READ_PAIR_OPTICAL_DUPLICATES', 'RIBOSOMAL_BASES', 'SECONDARY_OR_SUPPLEMENTARY_RDS', 'STANDARD_DEVIATION', 'STRAND_BALANCE.FIRST_OF_PAIR', 'STRAND_BALANCE.PAIR', 'STRAND_BALANCE.SECOND_OF_PAIR', 'TOTAL_CLUSTERS', 'TOTAL_READS.FIRST_OF_PAIR', 'TOTAL_READS.PAIR', 'TOTAL_READS.SECOND_OF_PAIR', 'Total pairs', 'Total unpaired reads', 'UNMAPPED_READS', 'UNPAIRED_READS_EXAMINED', 'UNPAIRED_READ_DUPLICATES', 'UTR_BASES', 'WIDTH_OF_10_PERCENT', 'WIDTH_OF_20_PERCENT', 'WIDTH_OF_30_PERCENT', 'WIDTH_OF_40_PERCENT', 'WIDTH_OF_50_PERCENT', 'WIDTH_OF_60_PERCENT', 'WIDTH_OF_70_PERCENT', 'WIDTH_OF_80_PERCENT', 'WIDTH_OF_90_PERCENT', 'WIDTH_OF_99_PERCENT', 'WINDOW_SIZE', 'alignable reads', 'filtered reads', 'multiple mapped', 'strand', 'total alignments', 'total reads', 'unalignable reads', 'uncertain reads', 'unique aligned', 'n_genes', 'n_counts'
    var: 'n_cells'
```

```
#Пример нормализации используя машинное обучение
#X_normalized = preprocessing.normalize(X, norm='l2')
```

```
#Выбираем часть генов в клетке  (первую 500 по экспрессии генов )
data_T = []
for i in range(dat.shape[0]):
    indices = np.asarray(np.argsort(dat.X[i,:].todense())[0,:]).flatten()[::-1][0:500]
    data_T = np.union1d(data_T, indices)
```

```
adata = adata[:, data_T.astype('int')]
```

```
sc.pl.violin(adata, 'n_counts', size=2, log=True, cut=0)
```



Подготовка данных к кластеризации

```
sc.pp.pca(adata, svd_solver='arpack')
sc.pp.neighbors(adata)
```

```
sc.tl.tsne(adata)
sc.tl.umap(adata)
```

```
print(adata)
```

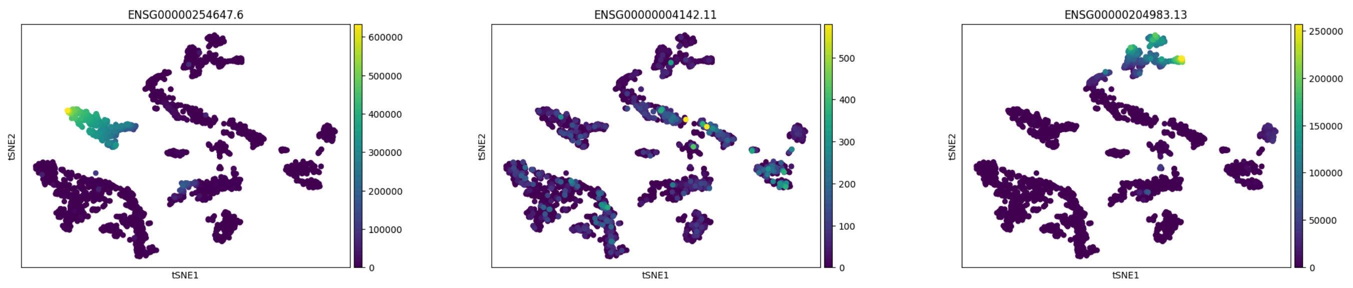
```
AnnData object with n_obs × n_vars = 2544 × 18968
    obs: 'ACCUMULATION_LEVEL', 'ALIGNED_READS', 'AT_DROPOUT', 'Aligned 0 time', 'Aligned 1 time', 'Aligned >1 times', 'Aligned conc
    var: 'n_cells'
    uns: 'pca', 'neighbors', 'tsne', 'umap'
    obsm: 'X_pca', 'X_tsne', 'X_umap'
    varm: 'PCs'
    obsp: 'distances', 'connectivities'
```

Графики

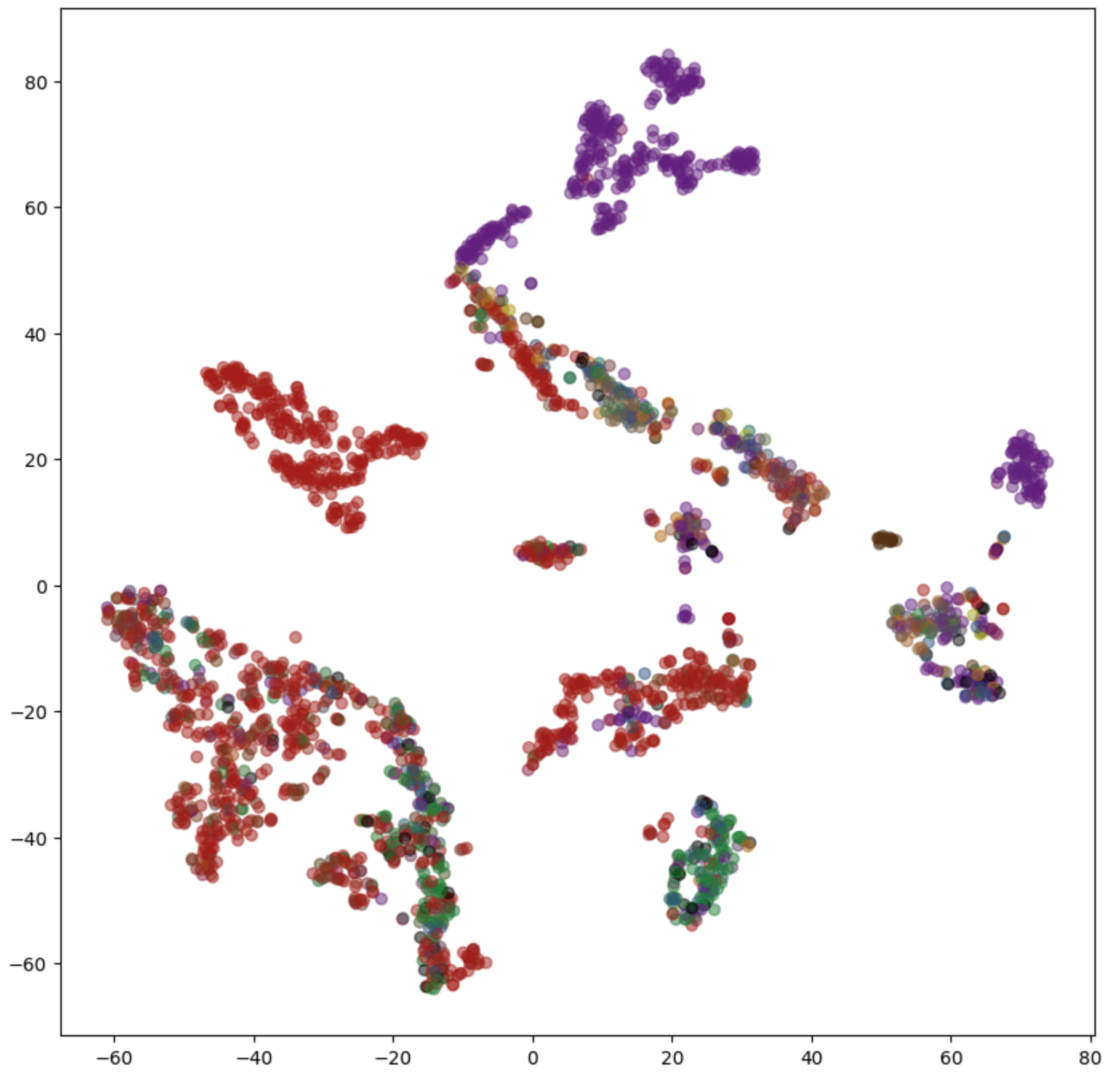
plot UMAP

```
genes = ['ENSG00000254647.6', 'ENSG00000004142.11', 'ENSG00000003147.17', 'ENSG00000204983.13', #TP53 #POLDIP2 #ICA1 #BRCA2
         'ENSG00000007062.11','ENSG00000154096.13', 'ENSG00000108849.7'] #CHEK2 #THY1 #PTEN
colors = np.asarray([[226,43,34], [67,114,175], [48,178,78],
                     [139,47,175], [244,152,46], [249,246,49], [119,73,27]])
weights = adata[:,genes].X.todense()
weights = weights / (weights.sum(axis=1) + 0.0001)
c = np.dot(weights, colors)
```

```
sc.pl.tsne(adata, color=['ENSG00000254647.6', 'ENSG00000004142.11', 'ENSG00000204983.13'],
           wspace=0.3, size=150, linewidths=1, edgecolors='black');
```

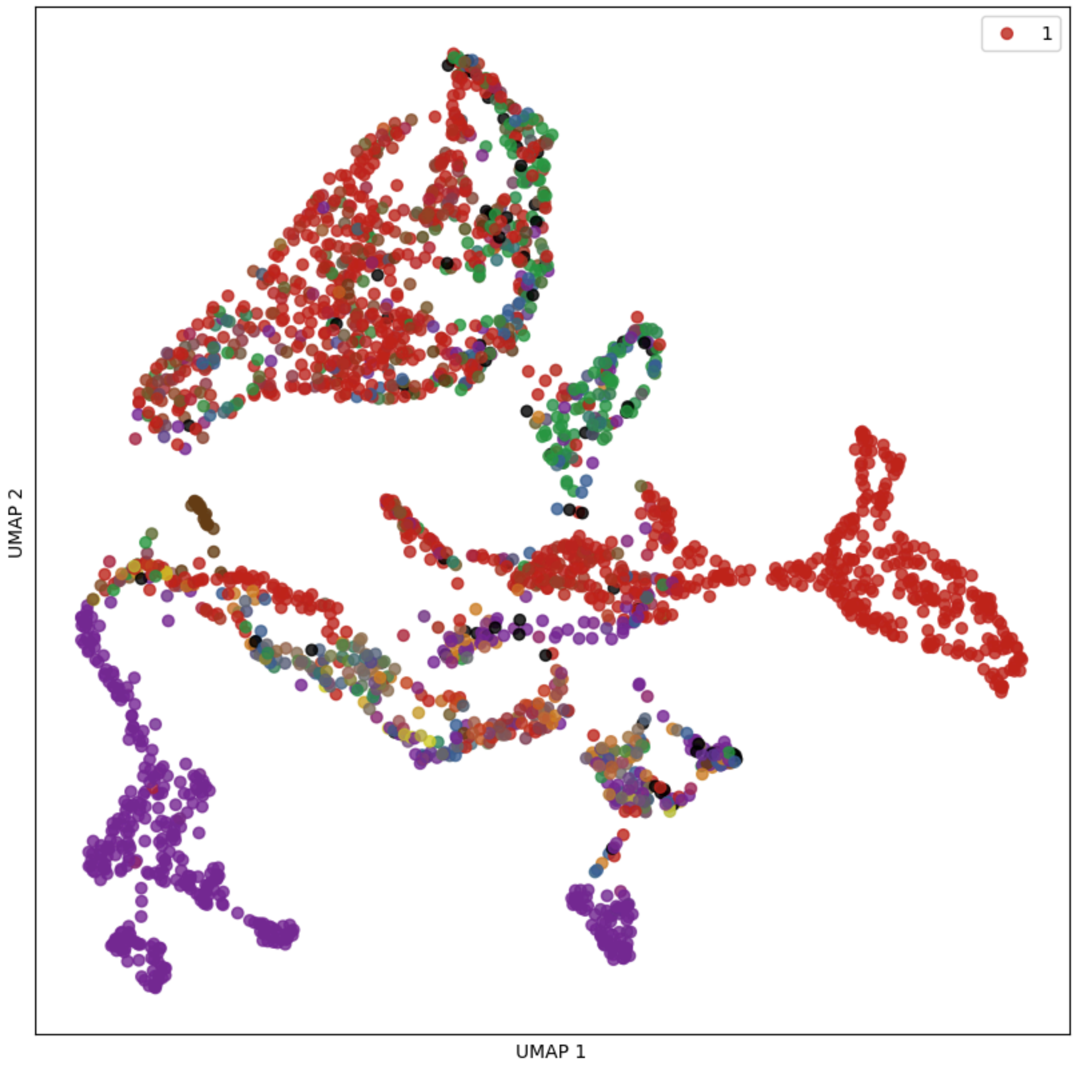


```
plt.figure(figsize=[10, 10])
plt.scatter(adata.obsm['X_tsne'][:,0], adata.obsm['X_tsne'][:,1], c=c/350, alpha=0.5);
#Клетки окрашены за счет экспрессии избранных генов
```



```
plt.figure(figsize=[10, 10])
plt.scatter(adata.obsm['X_umap'][:,0], adata.obsm['X_umap'][:,1], c=c/300, alpha=0.8);
plt.grid(False)
plt.yticks([])
plt.xticks([])
plt.xlabel('UMAP 1')
plt.ylabel('UMAP 2');

plt.legend(['CHEK2'])
plt.show()
```



UMAP Вариант 2

```
warnings.filterwarnings("ignore", message=r"Passing", category=FutureWarning)
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

```
X, y = data_d.drop("Index_Cell", axis=1), data_d[["Index_Cell"]].values.flatten()

X.head().sample(10, axis=1)
```

	CDH1.6	PBMC3.14	BRCA1.47	BRCA2.20	CDH1.4	CDH1.14	BRCA2.21	BRCA1.20	BRCA1.27	BRCA2.24
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	7.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0

```
X.shape
```

```
(588, 255)
```

```
y= le.fit_transform(y.astype(str))
```

```
y
```

Показать скрытые выходные данные

```
np.unique(y)
```

Показать скрытые выходные данные

```
import umap
from sklearn.preprocessing import PowerTransformer
from sklearn.pipeline import make_pipeline
```

```
# Scale
#pipe = make_pipeline(PowerTransformer())
#X = pipe.fit_transform(X.copy())
```

```
# Encode the target to numeric
y_encoded = pd.factorize(y)[0]
```

```
X = np.nan_to_num(X)
```

```
y_encoded = np.nan_to_num(y_encoded)
```

```
manifold = umap.UMAP().fit(X, y_encoded)
```

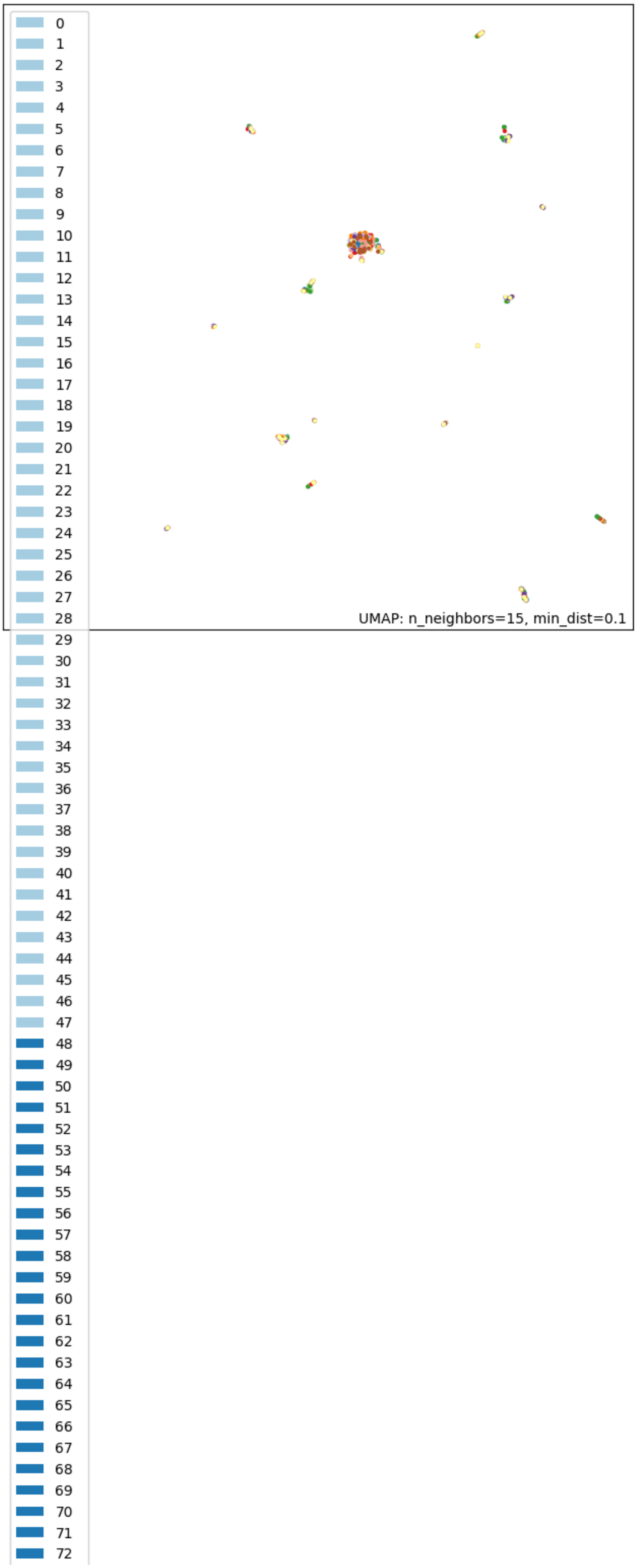
```
pip install umap-learn[plot]
```

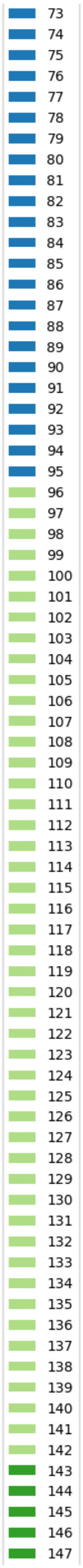
Показать скрытые выходные данные

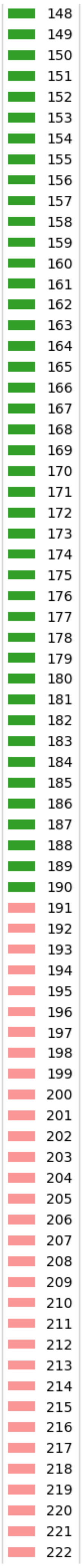
```
import umap.plot
```

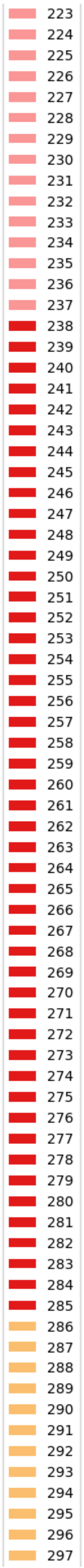
```
umap.plot.points(manifold, labels=y,color_key_cmap='Paired', background='white')
```

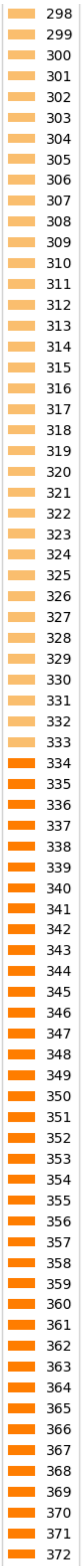
```
<Axes: >
```

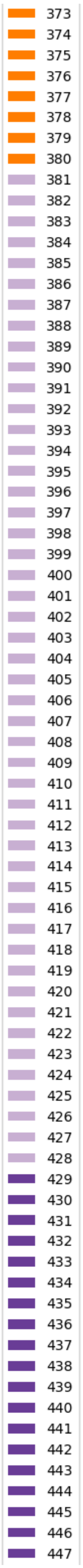

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
UMAP: n_neighbors=15, min_dist=0.1

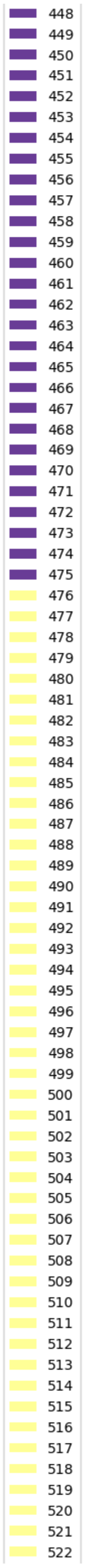

🟦 73
🟦 74
🟦 75
🟦 76
🟦 77
🟦 78
🟦 79
🟦 80
🟦 81
🟦 82
🟦 83
🟦 84
🟦 85
🟦 86
🟦 87
🟦 88
🟦 89
🟦 90
🟦 91
🟦 92
🟦 93
🟦 94
🟦 95
🟩 96
🟩 97
🟩 98
🟩 99
🟩 100
🟩 101
🟩 102
🟩 103
🟩 104
🟩 105
🟩 106
🟩 107
🟩 108
🟩 109
🟩 110
🟩 111
🟩 112
🟩 113
🟩 114
🟩 115
🟩 116
🟩 117
🟩 118
🟩 119
🟩 120
🟩 121
🟩 122
🟩 123
🟩 124
🟩 125
🟩 126
🟩 127
🟩 128
🟩 129
🟩 130
🟩 131
🟩 132
🟩 133
🟩 134
🟩 135
🟩 136
🟩 137
🟩 138
🟩 139
🟩 140
🟩 141
🟩 142
🟩 143
🟩 144
🟩 145
🟩 146
🟩 147

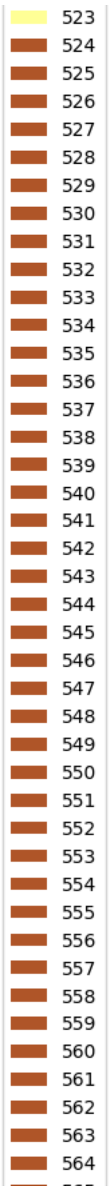

🟩 148
🟩 149
🟩 150
🟩 151
🟩 152
🟩 153
🟩 154
🟩 155
🟩 156
🟩 157
🟩 158
🟩 159
🟩 160
🟩 161
🟩 162
🟩 163
🟩 164
🟩 165
🟩 166
🟩 167
🟩 168
🟩 169
🟩 170
🟩 171
🟩 172
🟩 173
🟩 174
🟩 175
🟩 176
🟩 177
🟩 178
🟩 179
🟩 180
🟩 181
🟩 182
🟩 183
🟩 184
🟩 185
🟩 186
🟩 187
🟩 188
🟩 189
🟩 190
🟥 191
🟥 192
🟥 193
🟥 194
🟥 195
🟥 196
🟥 197
🟥 198
🟥 199
🟥 200
🟥 201
🟥 202
🟥 203
🟥 204
🟥 205
🟥 206
🟥 207
🟥 208
🟥 209
🟥 210
🟥 211
🟥 212
🟥 213
🟥 214
🟥 215
🟥 216
🟥 217
🟥 218
🟥 219
🟥 220
🟥 221
🟥 222


223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297


298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372


373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447


🟪 448
🟪 449
🟪 450
🟪 451
🟪 452
🟪 453
🟪 454
🟪 455
🟪 456
🟪 457
🟪 458
🟪 459
🟪 460
🟪 461
🟪 462
🟪 463
🟪 464
🟪 465
🟪 466
🟪 467
🟪 468
🟪 469
🟪 470
🟪 471
🟪 472
🟪 473
🟪 474
🟪 475
🟨 476
🟨 477
🟨 478
🟨 479
🟨 480
🟨 481
🟨 482
🟨 483
🟨 484
🟨 485
🟨 486
🟨 487
🟨 488
🟨 489
🟨 490
🟨 491
🟨 492
🟨 493
🟨 494
🟨 495
🟨 496
🟨 497
🟨 498
🟨 499
🟨 500
🟨 501
🟨 502
🟨 503
🟨 504
🟨 505
🟨 506
🟨 507
🟨 508
🟨 509
🟨 510
🟨 511
🟨 512
🟨 513
🟨 514
🟨 515
🟨 516
🟨 517
🟨 518
🟨 519
🟨 520
🟨 521
🟨 522


523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564