nature protocols

https://doi.org/10.1038/s41596-023-00892-x

Protocol

Check for updates

Inferring cellular and molecular processes in single-cell data with non-negative matrix factorization using Python, R and GenePattern Notebook implementations of CoGAPS

In the format provided by the authors and unedited

Supplementary Note 1

```
Cloning into 'pycogaps'...
remote: Enumerating objects: 1570, done.
remote: Counting objects: 100% (290/290), done.
remote: Compressing objects: 100% (191/191), done.
remote: Total 1570 (delta 160), reused 154 (delta 97), pack-reused
1280
Receiving objects: 100% (1570/1570), 8.97 MiB | 15.06 MiB/s, done.
Resolving deltas: 100% (950/950), done.
Filtering content: 100% (4/4), 2.29 GiB | 21.49 MiB/s, done.
Submodule 'src/CoGAPS' (https://github.com/FertigLab/CoGAPS.git)
registered for path 'src/CoGAPS'
Cloning into '/Users/fertiglab/pycogaps/src/CoGAPS'...
remote: Enumerating objects: 16976, done.
remote: Counting objects: 100% (343/343), done.
remote: Compressing objects: 100% (156/156), done.
remote: Total 16976 (delta 201), reused 312 (delta 187), pack-reused
16633
Receiving objects: 100% (16976/16976), 177.58 MiB | 6.10 MiB/s, done.
Resolving deltas: 100% (10621/10621), done.
Submodule path 'src/CoGAPS': checked out
'e1e002caa009866d41402f3aa5ad3f97b541d962'
```

Supplementary Note 2

```
This is pycogaps version  0.0.1


Running Standard CoGAPS on provided data object ( 25 genes and 20
samples) with parameters:


-- Standard Parameters --
nPatterns:  3
nIterations:  50000
seed:  42
sparseOptimization:  False
-- Sparsity Parameters --
alpha: 0.01
maxGibbsMass:  100.0
Data Model: Dense, Normal
Sampler Type: Sequential
Loading Data...Done! (00:00:00)
-- Equilibration Phase --
1000 of 50000, Atoms: 41(A), 37(P), ChiSq: 3076, Time: 00:00:00 /
00:00:00
2000 of 50000, Atoms: 51(A), 39(P), ChiSq: 1764, Time: 00:00:00 /
00:00:00
3000 of 50000, Atoms: 53(A), 41(P), ChiSq: 892, Time: 00:00:00 /
00:00:00
4000 of 50000, Atoms: 50(A), 42(P), ChiSq: 695, Time: 00:00:00 /
00:00:00
5000 of 50000, Atoms: 52(A), 44(P), ChiSq: 519, Time: 00:00:00 /
00:00:00
[ … ]
45000 of 50000, Atoms: 72(A), 51(P), ChiSq: 107, Time: 00:00:00 /
00:00:00
46000 of 50000, Atoms: 78(A), 52(P), ChiSq: 128, Time: 00:00:00 /
00:00:00
47000 of 50000, Atoms: 79(A), 48(P), ChiSq: 95, Time: 00:00:00 /
00:00:00
48000 of 50000, Atoms: 71(A), 56(P), ChiSq: 136, Time: 00:00:00 /
00:00:00
49000 of 50000, Atoms: 72(A), 47(P), ChiSq: 143, Time: 00:00:00 /
00:00:00
50000 of 50000, Atoms: 69(A), 53(P), ChiSq: 131, Time: 00:00:00 /
00:00:00
-- Sampling Phase --
1000 of 50000, Atoms: 74(A), 55(P), ChiSq: 104, Time: 00:00:00 /
00:00:00
2000 of 50000, Atoms: 75(A), 55(P), ChiSq: 123, Time: 00:00:00 /
```

```
00:00:00
3000 of 50000, Atoms: 87(A), 49(P), ChiSq: 101, Time: 00:00:00 /
00:00:00
4000 of 50000, Atoms: 82(A), 47(P), ChiSq: 102, Time: 00:00:00 /
00:00:00
5000 of 50000, Atoms: 82(A), 44(P), ChiSq: 100, Time: 00:00:00 /
00:00:00
[ … ]
45000 of 50000, Atoms: 71(A), 52(P), ChiSq: 132, Time: 00:00:01 /
00:00:01
46000 of 50000, Atoms: 68(A), 50(P), ChiSq: 130, Time: 00:00:01 /
00:00:01
47000 of 50000, Atoms: 75(A), 51(P), ChiSq: 140, Time: 00:00:01 /
00:00:01
48000 of 50000, Atoms: 74(A), 51(P), ChiSq: 108, Time: 00:00:01 /
00:00:01
49000 of 50000, Atoms: 74(A), 52(P), ChiSq: 108, Time: 00:00:01 /
00:00:01
50000 of 50000, Atoms: 81(A), 47(P), ChiSq: 108, Time: 00:00:01 /
00:00:01

GapsResult result object with 25 features and 20 samples
3 patterns were learned
```

Supplementary Note 3

```
>> modsimresult
AnnData object with n_obs × n_vars = 25 × 20
    obs: 'Pattern1', 'Pattern2', 'Pattern3'
    var: 'Pattern1', 'Pattern2', 'Pattern3'
    uns: 'asd', 'psd', 'atomhistoryA', 'atomhistoryP',
'averageQueueLengthA', 'averageQueueLengthP', 'chisqHistory',
'equilibrationSnapshotsA', 'equilibrationSnapshotsP', 'meanChiSq',
'meanPatternAssignment', 'pumpMatrix', 'samplingSnapshotsA',
'samplingSnapshotsP', 'seed', 'totalRunningTime', 'totalUpdates'

>> modsimresult.obs.head()

    Pattern1  Pattern2  Pattern3
0  11.975077  0.101262  0.419482
1  10.741890  0.125388  1.356528
2   8.950110  0.035884  2.792770
3   7.521177  0.019195  3.895204
4   6.137812  0.014312  4.937158

>> modsimresult.var.head()

   Pattern1  Pattern2  Pattern3
0  0.001125  0.000760  0.013232
1  0.075658  0.000514  0.135343
2  0.341718  0.002400  0.604787
3  0.567491  0.004287  1.000000
4  0.391151  0.002598  0.640158
```

Supplementary Note 4

```
scanpy==1.9.1 anndata==0.7.8 umap==0.5.3 numpy==1.23.5 scipy==1.10.0
pandas==1.2.4 scikit-learn==1.0.1 statsmodels==0.13.5
pynndescent==0.5.8
extracting highly variable genes
    finished (0:00:01)
--> added
    'highly_variable', boolean vector (adata.var)
    'means', float vector (adata.var)
    'dispersions', float vector (adata.var)
    'dispersions_norm', float vector (adata.var)
/Users/fertiglab/pycogaps/venv/lib/python3.10/site-packages/scanpy/prep
rocessing/_simple.py:843: UserWarning: Received a view of an AnnData.
Making a copy.
  view_to_actual(adata)
computing PCA
    on highly variable genes
    with  n_comps=50
    finished (0:00:15)
computing neighbors
    using 'X_pca' with n_pcs = 50
    finished: added to `.uns['neighbors']`
    `.obsp['distances']`, distances for each pair of neighbors
    `.obsp['connectivities']`, weighted adjacency matrix (0:00:00)
computing UMAP
    finished: added
    'X_umap', UMAP coordinates (adata.obsm) (0:00:07)
    WARNING: saving figure to file figures/umapepithelial_pycogaps_UMAP.png
```

Supplementary Note 5

```
 _____      _____       _____   ___  _______  _____
| ___ \    /  __ \     |  __ \ / _ \ | ___ \/  ___|
| |_/ /   _| /  \/ ___ | |  \// /_\ \| |_/ /\ `--.
|  __/ | | | |    / _ \| | __ |  _  ||  __/  `--. |
| |  | |_| | \__/\ (_) | |_\ \| | | || |    /\__/ /
\_|   \__, |\____/\___/ \____/\_| |_/\_|    \____/
       __/ |
      |___/
```

```
pycogaps version  0.0.1
This vignette was built using pycogaps version 0.0.1

This is pycogaps version  0.0.1
Running Standard CoGAPS on ModSimData.txt ( 25 genes and 20 samples) with parameters:

-- Standard Parameters --
nPatterns:  3
nIterations:  1000
seed:  0
sparseOptimization:  False


-- Sparsity Parameters --
alpha: 0.01
maxGibbsMass:  100.0


Data Model: Dense, Normal
Sampler Type: Sequential
Loading Data...Done! (00:00:00)
-- Equilibration Phase --
1000 of 1000, Atoms: 70(A), 54(P), ChiSq: 1351, Time: 00:00:00 / 00:00:00
-- Sampling Phase --
1000 of 1000, Atoms: 78(A), 48(P), ChiSq: 1185, Time: 00:00:00 / 00:00:00

GapsResult result object with 25 features and 20 samples
3 patterns were learned

Pickling complete!
```

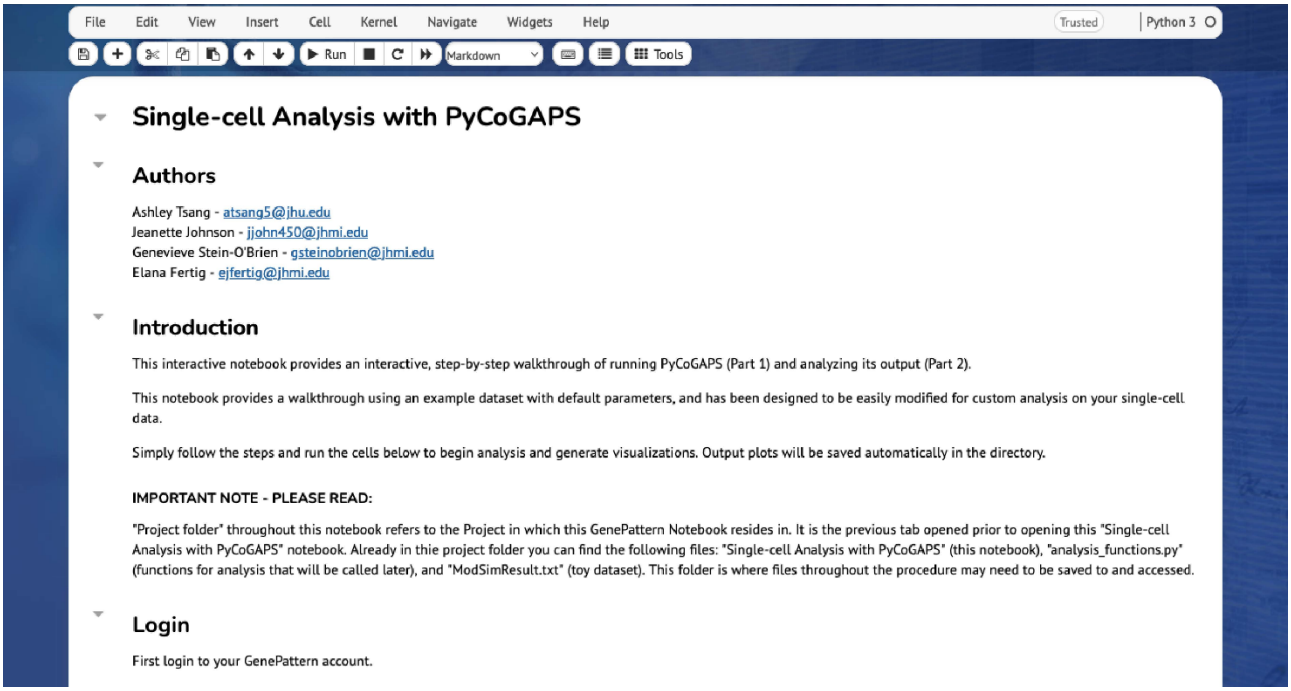
Supplementary Note 6

```
This is CoGAPS version 3.19.1
Running Standard CoGAPS on modsimdata (25 genes and 20 samples) with
parameters:

-- Standard Parameters --
nPatterns             3
nIterations           50000
seed                  622
sparseOptimization    FALSE

-- Sparsity Parameters --
alpha          0.01
maxGibbsMass   100

Data Model: Dense, Normal
Sampler Type: Sequential
Loading Data...Done! (00:00:00)
-- Equilibration Phase --
10000 of 50000, Atoms: 59(A), 49(P), ChiSq: 245, Time: 00:00:00 /
00:00:00
20000 of 50000, Atoms: 68(A), 46(P), ChiSq: 188, Time: 00:00:00 /
00:00:00
30000 of 50000, Atoms: 80(A), 47(P), ChiSq: 134, Time: 00:00:00 /
00:00:00
40000 of 50000, Atoms: 69(A), 46(P), ChiSq: 101, Time: 00:00:00 /
00:00:00
50000 of 50000, Atoms: 76(A), 53(P), ChiSq: 132, Time: 00:00:00 /
00:00:00
-- Sampling Phase --
10000 of 50000, Atoms: 82(A), 52(P), ChiSq: 94, Time: 00:00:00 /
00:00:00
20000 of 50000, Atoms: 74(A), 54(P), ChiSq: 144, Time: 00:00:01 /
00:00:01
30000 of 50000, Atoms: 79(A), 47(P), ChiSq: 116, Time: 00:00:01 /
00:00:01
40000 of 50000, Atoms: 79(A), 46(P), ChiSq: 132, Time: 00:00:01 /
00:00:01
50000 of 50000, Atoms: 76(A), 48(P), ChiSq: 124, Time: 00:00:01 /
00:00:01
```

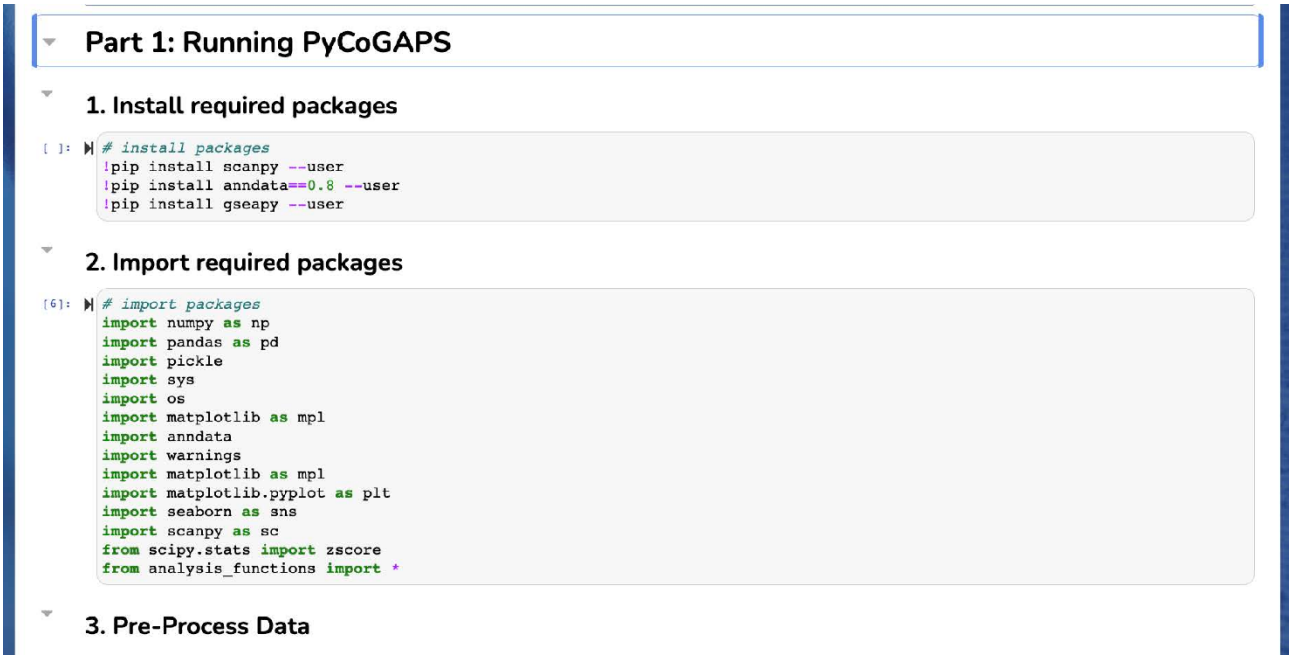
Supplementary Note 7

```
> cogapsresult
[1] "CogapsResult object with 25 features and 20 samples"
[1] "3 patterns were learned"
> cogapsresult@sampleFactors
       Pattern_1    Pattern_2    Pattern_3
V1  8.388746e-03 0.0062945839 0.0007746686
V2  8.932892e-02 0.1355465055 0.0005277477
V3  3.992884e-01 0.6105546355 0.0019834286
V4  6.626218e-01 0.9999989867 0.0034138400
V5  4.499646e-01 0.6090151072 0.0021458515
V6  3.541442e-01 0.1354408264 0.0014731999
V7  7.266073e-01 0.0057070116 0.0048238179
V8  1.000000e+00 0.0058223484 0.0046370891
V9  7.166106e-01 0.0046964670 0.0032584930
V10 2.641986e-01 0.0020137667 0.0012293814
V11 5.295977e-02 0.0005939197 0.0074665351
V12 1.486760e-03 0.0009508047 0.1134860739
V13 2.749619e-04 0.0005655847 0.3701201081
V14 3.304514e-04 0.0020806976 0.7812449932
V15 4.917108e-04 0.0038944422 1.0000000000
V16 3.356318e-04 0.0017714981 0.7815062404
V17 2.180223e-04 0.0006926730 0.3694530129
V18 1.647724e-04 0.0002741633 0.1056895107
V19 7.622870e-04 0.0013716018 0.0139522608
V20 5.930163e-05 0.0004655297 0.0001998204

> cogapsresult@featureLoadings
        Pattern_1   Pattern_2 Pattern_3
Gene_1  11.541452 0.003959592 0.1725101
Gene_2  11.458824 0.011860452 0.3962909
Gene_3  11.200215 0.143655315 0.5942768
Gene_4  10.484943 0.919701695 0.7919101
Gene_5   9.622066 1.971993566 0.9882591
Gene_6   8.901559 2.780912161 1.1860518
Gene_7   8.183671 3.564562798 1.3833313
Gene_8   7.455255 4.367302418 1.5811858
Gene_9   6.721270 5.166158199 1.7781062
Gene_10  5.993835 5.958409309 1.9737232
Gene_11  6.339371 0.005076513 6.1401019
Gene_12  7.336598 0.066603549 5.9148746
Gene_13  8.051244 0.530350804 5.6913567
Gene_14  8.588865 1.326142669 5.4673815
Gene_15  9.184795 2.006909609 5.2382684
Gene_16  9.830432 2.598231554 5.0155287
Gene_17 10.490241 3.181848764 4.7880602
```

Gene_18 11.132268 3.768204927 4.5618424
Gene_19 11.772452 4.362468243 4.3355570
Gene_20 12.417335 4.943210602 4.1082892
Gene_21 13.053541 5.535136700 3.8799365
Gene_22 13.705083 6.107748985 3.6509578
Gene_23 14.347120 6.697602272 3.4237635
Gene_24 14.986680 7.265126705 3.1958113
Gene_25 15.624031 7.858362675 2.9668522



Supplementary figure 1. GenePattern Notebook screenshot 1. Login page for the PyCoGAPS GenePattern Notebook module.



Supplementary figure 2. GenePattern Notebook screenshot 2. Installing required packages.

1. Modify PyCoGAPS parameters

We recommend starting out with a sample run to ensure you are able to run PyCoGAPS and output the expected result file. We have provided the following toy dataset in the project folder, called "ModSimData.txt". It will take about 5 minutes for the run to complete.

Otherwise, load your desired data:

- If running a file saved to the project notebook (eg. the file created in the log-normalization cell), do not use the "Upload File..." button. Instead, go to the project folder, click on the data file (eg. "{filename}_log.h5ad") and copy the web link. Then paste this link into the "input file" box.
- Otherwise, use the "Upload File..." button to upload a file saved on your local computer.

Next, modify any parameters as desired. Descriptions of each parameter can be found below. In particular, we recommend playing around with the parameters 'num patterns' and 'num iterations'.

After modifying the parameters, we are now ready to run PyCoGAPS. We will take advantage of GenePatterns' cloud computing for faster runtimes. Please note that timing will depend on factors such as the dimensionality of your data, the number of patterns to be learned, the number of iterations, and whether sparse optimization is enabled.

GenePattern Not Authenticated

You must be authenticated before the analysis can be displayed. After you authenticate it may take a few seconds for the information to appear.

Not Authenticated

You must be authenticated before the job information can be displayed. After you authenticate it may take a few seconds for the job information to appear.

2. Save the Result File

After the job has successfully, completed you will see "Completed" with three outputs in blue. We want to make sure to save the ".h5ad" output which is our result file. To do so, click the (i) icon next to the file, and select "Download File". This will save the result file locally to your computer.

In order to perform the next analysis steps, we need to upload this result file to the project folder.

Supplementary figure 3. GenePattern Notebook screenshot 3. Modifying parameters and starting the CoGAPS run. .

Part 2: Analysis & Visualization

Once PyCoGAPS has completed running and the result file has been saved, we are now ready for analysis of the result.

Please follow the following steps to analyze your result.

1. Load the Result File

Next, we'll load in the result file generated from PyCoGAPS.

Make sure you have uploaded your result file into the project folder.

Replace the default result_file with your result file name.

[15]:
```
## EDIT HERE: REPLACE THIS LINE ONLY with the path to your result file -- make sure to upload your file in the folder
result_file = 'cogapsresult.h5ad'

# this reads the result object for use
result = anndata.read_h5ad(result_file)
print(result)

# this gets the filename of your result file, it is used to name the saved plots
filename = os.path.basename(result_file).split('.')[0]
```

```
AnnData object with n_obs × n_vars = 15176 × 25442
    obs: 'Pattern_1', 'Pattern_2', 'Pattern_3', 'Pattern_4', 'Pattern_5', 'Pattern_6', 'Pattern_7', 'Pattern_8'
    var: 'Pattern_1', 'Pattern_2', 'Pattern_3', 'Pattern_4', 'Pattern_5', 'Pattern_6', 'Pattern_7', 'Pattern_8', 'cell_type'
```

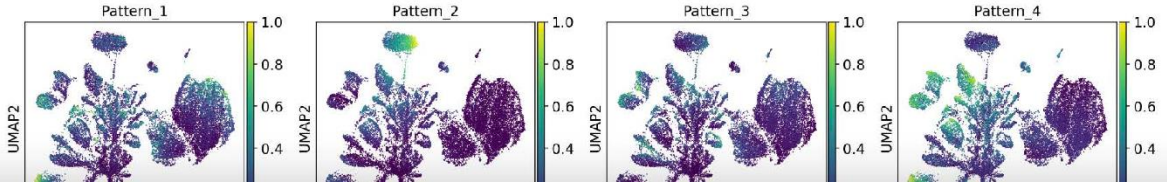
Supplementary figure 4. GenePattern Notebook screenshot 4. Loading CoGAPS result from a file.

2. Plot pattern UMAP

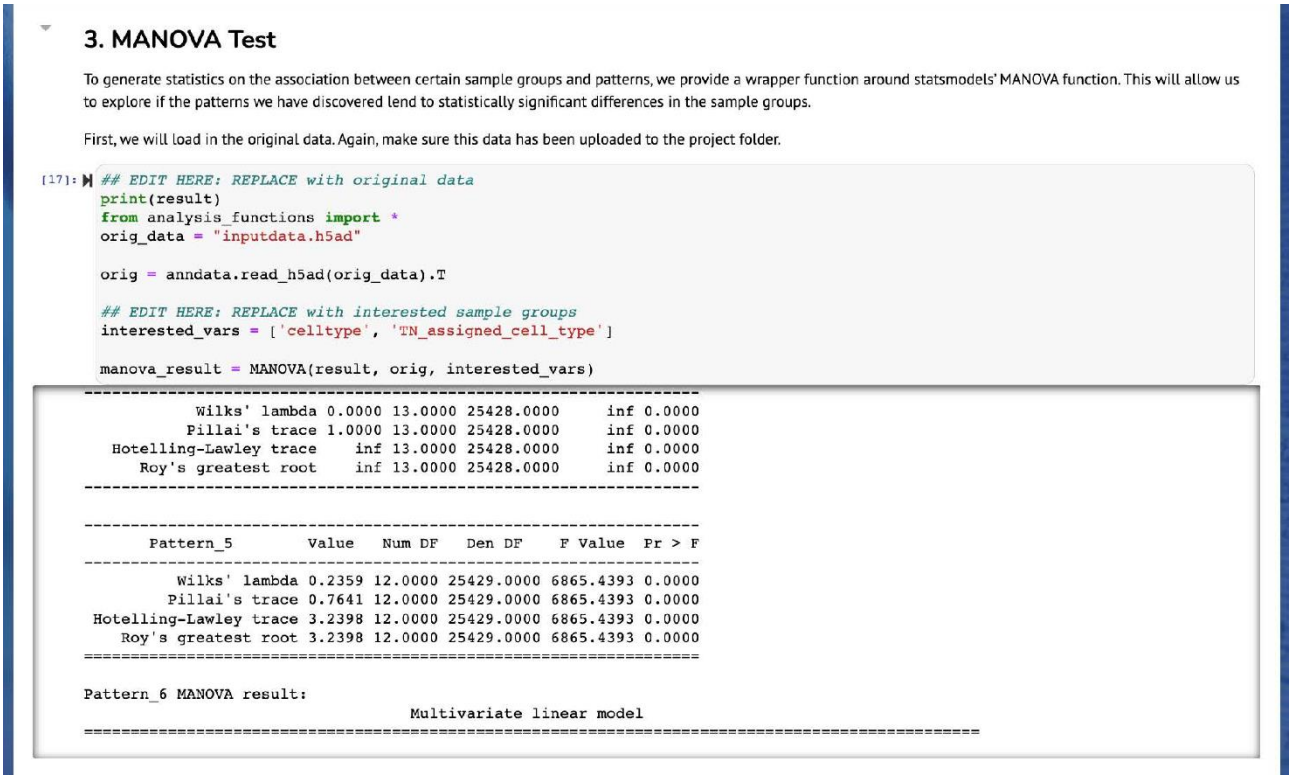
We will visualize patterns and compare it with clusters and annotations using UMAP and the scanpy package. https://scanpy-tutorials.readthedocs.io/en/latest/pbmc3k.html We provide a wrapper function to perform basic clustering workflow in scanpy (all default parameters) and produce a plot of each pattern's intensity displayed on the data's UMAP embedding.

```
[16]: plotPatternUMAP(result, fn="epithelial_pycogaps")
```

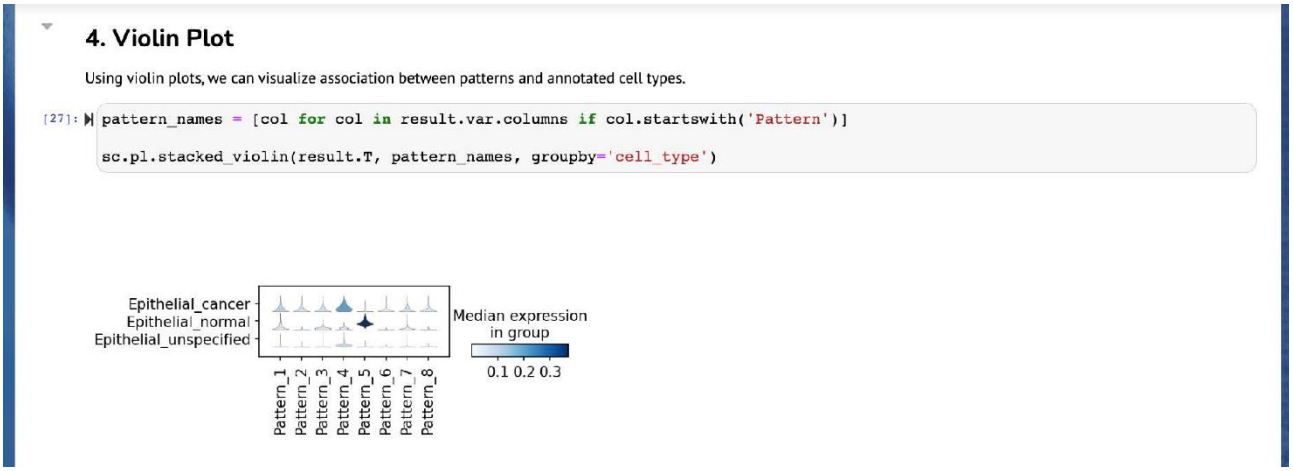
```
scanpy==1.8.2 anndata==0.8.0 umap==0.5.2 numpy==1.20.3 scipy==1.6.3 pandas==1.2.4 scikit-learn==0.24.2 statsmodels==0.12.2 py
nndescent==0.5.6
extracting highly variable genes
    finished (0:00:04)
--> added
    'highly_variable', boolean vector (adata.var)
    'means', float vector (adata.var)
    'dispersions', float vector (adata.var)
    'dispersions_norm', float vector (adata.var)
/home/jovyan/.local/lib/python3.9/site-packages/scanpy/preprocessing/_simple.py:843: UserWarning: Revieved a view of an AnnDa
ta. Making a copy.
  view_to_actual(adata)
computing PCA
    on highly variable genes
    with n_comps=50
    finished (0:00:03)
computing neighbors
    using 'X_pca' with n_pcs = 50
    finished: added to `.uns['neighbors']`
    `.obsp['distances']`, distances for each pair of neighbors
    `.obsp['connectivities']`, weighted adjacency matrix (0:00:16)
computing UMAP
    finished: added
    'X_umap', UMAP coordinates (adata.obsm) (0:00:17)
```



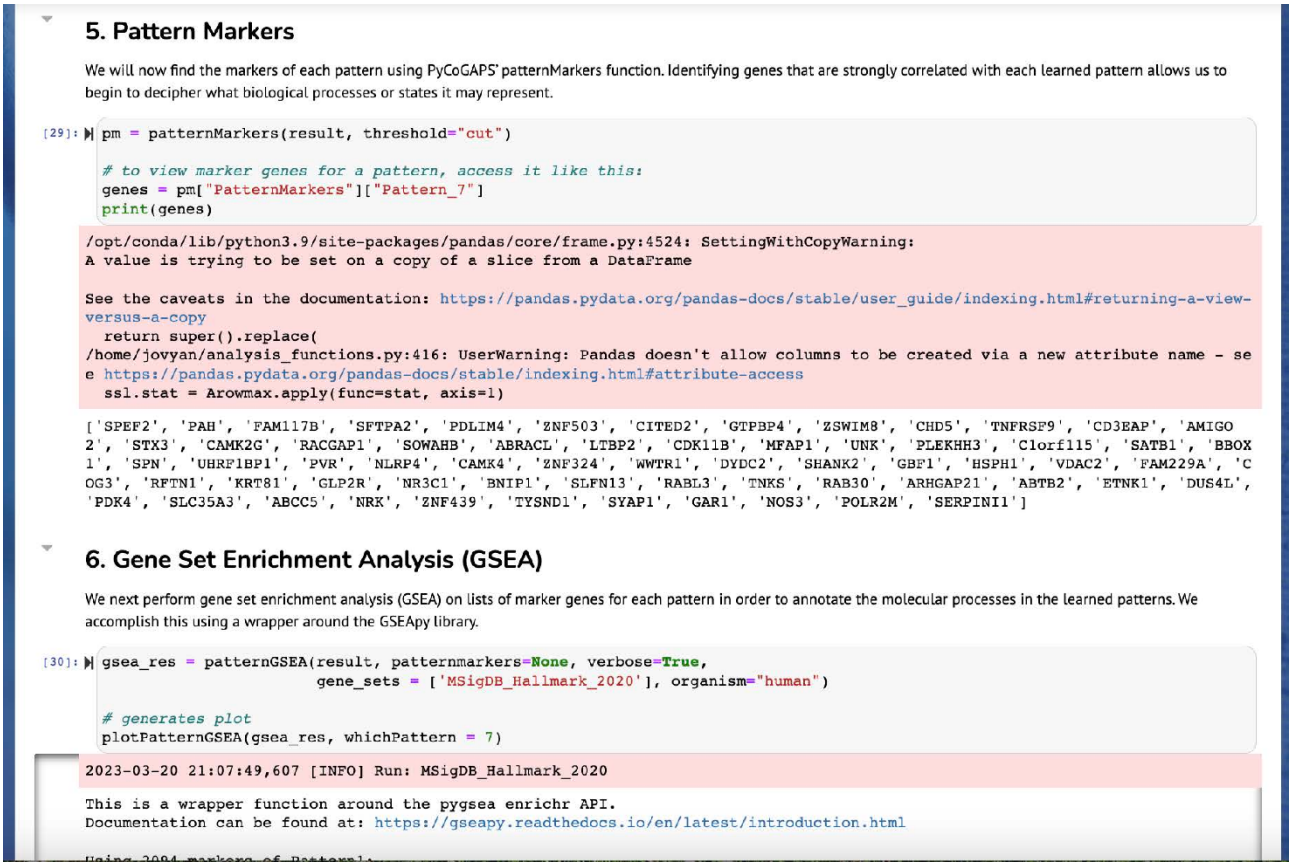
Supplementary figure 5. GenePattern Notebook screenshot 5. Plotting pattern UMAPs.



Supplementary figure 6. GenePattern Notebook screenshot 6. MANOVA Test.



Supplementary figure 7. GenePattern Notebook screenshot 7. Violin plots to compare patterns between groups.



Supplementary figure 8. GenePattern Notebook screenshot 8. Computing Pattern Markers statistic and running preranked Gene Set Enrichment analysis.