

Assignment Goals

- Get familiar with the single-cell analysis tool in Python: Scanpy.
- Compare different dimensionality reduction methods.
- Comparing scRNA-seq with bulk RNA-seq data.
- Get familiar with the workflow of machine-learning modeling.
- Browse online databases such as Gene Expression Omnibus.
- Gain a deep understanding of convolutional neural networks (CNN) for regulatory genomics.

Submission Instructions

- To turn in your assignment, please log in to the server **pluto.biostat.wisc.edu** using your **pluto** server username and password.
- Copy all relevant files to the directory

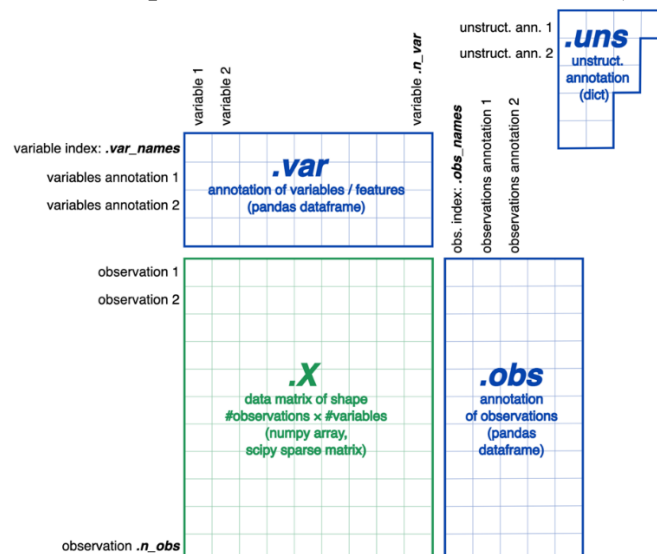
/home/medinfo/bmi776-2022/hw4/<USERNAME>

where **<USERNAME>** is your NetID. Submit all of your Python source code and test that it runs on the pluto server.

- For the rest of the assignment, compile all of your answers in a single file and submit as **solution.pdf**.
- Write the number of late days you used at the top of **solution.pdf**.
- For the written portions of the assignment, show your work for partial credit.

Part 1: scRNA-seq Analysis (30 points)

Scanpy (Wolf et al. 2018) is a popular python tool to process and analyze single-cell gene expression data. It uses AnnData format as follows to store the matrix. In this assignment, we will work on some basic single-cell data preprocessing steps on the processed real word data - Peripheral Blood Mononuclear Cells (PBMC).



To answer the questions below, you are required to install Scanpy on your local computer.

Following the instruction (<https://scanpy.readthedocs.io/en/latest/installation.html>)

You could also access the Scanpy on the pluto server using the following command line.

```
conda activate py3
source activate scanpy
scanpy -h
```

Then, you could load the data using the following code, **pbmc3k.h5ad** is included in the homework folder.

```
import scanpy as sc
adata = sc.read('pbmc3k.h5ad')
```

```
In [1]: import numpy as np
import pandas as pd
import scanpy as sc

In [2]: adata = sc.read('pbmc3k.h5ad')
adata

Out[2]: AnnData object with n_obs × n_vars = 2638 × 208
  obs: 'n_genes', 'percent_mito', 'n_counts', 'louvain', 'cell_type'
  var: 'n_cells'
  uns: 'draw_graph', 'louvain', 'louvain_colors', 'neighbors', 'pca', 'rank_genes_groups', 'umap'
  obsm: 'X_draw_graph_fr', 'X_pca', 'X_tsne', 'X_umap'
  varm: 'PCs'
  obsp: 'connectivities', 'distances'
```

Please submit your Python code (it can be .py or .ipynb) as a separate file with the file name title “part1_scanpy”. When answering these following questions, please save or take a screenshot of the output figures.

- (A) Use Scanpy to perform PCA, t-SNE, and UMAP. Please plot two-dimension figures on the data. Then, please discuss which method would be more suitable for this dataset. **(10 points)**
- (B) ‘Perplexity’ is an important hyperparameter parameter for t-SNE as it can balance the local and global aspects of the data. The original paper mentions that ‘The performance of SNE is fairly robust to changes in the perplexity, and typical values are between 5 and 50.’ Try different Perplexity (5, 40) on the PBMC data, and discuss what you found. Notes: *We could learn more from the data using t-SNE with different perplexities.* **(5 points)**

(C) For UMAP, **n_neighbors** and **min_dist** are the two most commonly used hyperparameters, which are effectively used to balance the local and global structure of the data.

n_neighbors, the number of approximate nearest neighbors used to construct the initial graph in high-dimension. Low values would help UMAP to focus more on fine details in the local structure. High values would push UMAP to represent the big-picture of the data.

min_dist, the minimum distance between points in low-dimensional space. It controls how tightly UMAP clumped points together.

Try different **n_neighbors** and **min_dist** values, and then discuss what you found. (5 points)

(D) In class, we introduced several ways to identify the cell type marker genes. For the non-parametric test, (1) Wilcoxon rank sum test (2) Student's t-test could be easily applied in the Scanpy. Please perform Wilcoxon rank sum test and Student's t-test to identify the top 25 marker genes for each cell type, then compare these results. (10 points)

Part 2 Machine-Learning Modeling (45 points)

It is important to better understand genetic mechanisms of critical illness (severe outcomes) in Covid-19 patients. In collaboration with biologists and front-line researchers, you as a bioinformatician want to build a predictive machine-learning model for Covid-19 severity based on changes in global gene expression of blood gene expression data of hospitalized human patients. You receive data such as that in this study (<https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE157103>); please note that this dataset is just an example, for your reference, but you do not need to use this dataset to answer this problem. In general, being in the Intensive Care Unit (ICU) tends to mean the patient is undergoing severe health problems (whether by Covid-19 for infected patients or some other health cases for negative controls (groups 3 and 4). The key information is as follows: you have 4 groups of patients in the data:

Group 1: The 50 patients who are hospitalized with severe Covid-19 (in the Intensive Care Unit (ICU)).

Group 2: The 50 patients who are hospitalized with Covid-19 (but it is not severe)

Group 3: The 13 patients who are hospitalized in the Intensive Care Unit for some other severe condition (they do not have Covid-19 and have never been exposed).

Group 4: The 13 patients who are hospitalized for some other condition (they do not have Covid-19 and have never been exposed) that is not severe.

Your collaborators performed RNA-Seq and obtained measurements for 24,000 genes following exposure for these 126 different human samples, with five biological replicates for each gene.

In this data, you thus have access to whether a patient has been exposed to Covid-19 or not, as well as to different levels of severity and can ask a few different questions.

- (A) What are some of the questions you could address using Differential Gene Expression Analysis (DGEA)? That is, how could you compare groups with one another or group them together to derive some meaningful statistical analysis from this multi-class data? Which of these questions would you like to focus on for parts B and C, and why (please pick 1 as your research question). Hint: the questions typically are classification-based. **(5 points)**
- (B) (*Exploratory data analysis*) It is common practice to apply unsupervised learning methods (clustering, dimensionality reduction, etc.) on the measurement data in order to understand the intragroup variability among replicates and the intergroup variability among samples with different outcomes. Based on your research question of interest for part A, please outline *two* unsupervised learning methods that you think would serve this purpose. Please describe what you would expect these methods to uncover. **(15 points)**
- (C) (*Learning a classifier*) Now that you have gained some insight from the data through unsupervised learning, you would like to proceed and build a support vector machine (SVM) to address this same research question you chose from part A (please choose 1). Given a gene expression profile of an individual based on the blood serum levels, the SVM will perform classification. Please describe a workflow for training and evaluating the SVM. Please beware of the high feature dimensionality relative to the sample size and the class imbalance problem **(15 points)**.

Please find another dataset of your choosing from Gene Expression Omnibus dataset browser (<https://www.ncbi.nlm.nih.gov/sites/GDSbrowser/>) and include the following information below. For instance, the screenshot below shows how you can analyze studies related to small cell lung cancer. **(10 points)**

The screenshot shows the NCBI GEO Data Browser interface. At the top, there's a search bar with 'lung cancer' entered. Below the search bar, a table lists 59 datasets. The table has columns for DataSet, Title, Organism(s), Platform, Series, and Samples. The dataset GDS4794 is highlighted. Below the table, there's a detailed view of dataset GDS4794, including its title, summary, organism, platform, citation, reference series, sample count, and value type. On the right side, there's a 'Cluster Analysis' section with a heatmap and a 'Download' section with links to various data files.

DataSet	Title	Organism(s)	Platform	Series	Samples
GDS5410	Non-small cell lung cancer H460-derived cancer stem cell differentiation	<i>Homo sapiens</i>	GPL4133	GSE54712	8
GDS5418	Glucose effect on steroid receptor coactivator 1 deficient-A549 lung cancer epithelial cell line	<i>Homo sapiens</i>	GPL570	GSE54843	8
GDS5468	Chronic high-calorie diet effect on Kras-driven lung tumors	<i>Mus musculus</i>	GPL6887	GSE54260	11
GDS5004	Adult alveolar type 2 and embryonic bipotent progenitor lung cells	<i>Mus musculus</i>	GPL1261	GSE49346	6
GDS5409	Akt inhibitor MK2206 effect on influenza H1N1 infection of non-small cell lung cancer line	<i>Homo sapiens</i>	GPL10558	GSE54293	8
GDS5391	Protein tyrosine kinase PTK7 knockdown effect on lung adenocarcinoma cell lines	<i>Homo sapiens</i>	GPL6244	GSE50138	8
GDS5040	V-ets erythroblastosis virus E26 oncogene homolog 2 knockdown effect on lung cancer cells	<i>Homo sapiens</i>	GPL6244	GSE43459	6
GDS5067	Oligonucleotide effect on hypoxic alveolar adenocarcinoma cell line	<i>Homo sapiens</i>	GPL6244	GSE48134	15
GDS4794	Small cell lung cancers	<i>Homo sapiens</i>	GPL570	GSE43346	65
GDS4767	Pancreatic cancer model: kalouza tumorigenesis	<i>Mus musculus</i>	GDR1361	GSE47811	12

DataSet Record GDS4794: Expression Profiles | Data Analysis Tools | Sample Subsets

Title: Small cell lung cancers

Summary: Analysis of 23 clinical small cell lung cancer (SCLC) samples from patients undergoing pulmonary resection and 42 normal tissue samples including the lung. SCLC is a lung cancer subtype with poor prognosis. Results provide insight into the molecular mechanisms underlying SCLC.

Organism: *Homo sapiens*

Platform: GPL570: [HG-U133_Plus_2] Affymetrix Human Genome U133 Plus 2.0 Array

Citation: Sato T, Kaneda A, Tsuji S, Isagawa T et al. PRIC2 overexpression and PRIC2-target gene repression relating to poorer prognosis in small cell lung cancer. *Sci Rep* 2013;3:1911. PMID: 23714854

Reference Series: GSE43346

Sample count: 65

Value type: count

Series published: 2013/06/12

Download:

- DataSet full SOFT file
- DataSet SOFT file
- Series family SOFT file
- Series family MINML file
- Annotation SOFT file

- Screenshot of the GEO download page
- Link to dataset
- Title of dataset
- Paraphrased summary of dataset
- Organism studied
- # of samples
- Type of data (e.g. bulk RNA-seq, etc.)
- Potential question(s) that could be addressed from this data (such as in (A))
- Other information

Part 3: Deep Regulatory Genomic Neural Networks (DragoNN) (25 points)

We will use the DragoNN Python package to explore convolutional neural networks (CNN) for regulatory genomics. DragoNN can create DeepSEA-like networks but is more user-friendly, which makes it easier to simulate training sequence data for user-specified *cis*-regulatory modules, experiment with different network architectures, and visualize the filters learned by a CNN.

To answer the questions below, you are *required* to perform all experiments on the pluto servers. To access the DragoNN package, first confirm that you are using the BMI776 Python environment. Next, type

```
/home/medinfo/miniconda3.7/bin/conda init
```

in the command line.

Log out, and back in.

Then, copy all **.fa.gz** files and **interpret.py** to your handin directory. Run everything in your handin directory and leave the output files there.

- (A) (*CNN Training*) You will first train a CNN on data from a simulated ChIP-Seq experiment. You are provided with a FASTA-formatted file of 5,000 DNA sequences bounded by some regulatory proteins, **positive_train.fa.gz**, and a negative set of 5,000 unbounded sequences, **negative_train.fa.gz**. Use the following command to train a one-layer CNN with five hidden channels (or filters) and a convolutional kernel of width 15:

```
python train.py train \
    --pos-sequences positive_train.fa.gz \
    --neg-sequences negative_train.fa.gz \
    --prefix one_layer \
    --num-filters 5 \
    --conv-width 15
```

This trains the one-layer CNN and saves the model architecture and learned weights to **one_layer.hdf5**. DragonNN splits the input data into training and validation sets, and reports several performance metrics after each epoch of training.

- What are the training and validation auPRC (area under the precision-recall curve) after the last epoch? How does the training auPRC compare with the validation auPRC and is this what you expected? **(3 points)**
- Please try adjusting the number of filters (num-filters) from 5 and/or the convolutional width (conv-width) from 15 and re-run the previous steps. Please include the Python commands you ran. What is the training auPRC and validation auPRC? **(3 points)**

The one-layer CNN is an extremely simple network. We can train a more complex network by adding more layers and filters. Use the following command to train a two-layer CNN with 15 filters per layer and a convolutional kernel of width 15:

```
python train.py train \
    --pos-sequences positive_train.fa.gz \
    --neg-sequences negative_train.fa.gz \
    --prefix two_layer \
    --num-filters 15 15 \
```

```
--conv-width 15 15
```

This trains the two-layer CNN and saves the network architecture and learned weights to **two_layer.hdf5**.

- What are the training and validation auPRC after the last epoch? Why is the two-layer CNN's performance better than the one-layer CNN? **(4 points)**

(B) (*CNN Inference and visualization*) You will inspect and visualize the two-layer CNN you trained in (A) using the following command:

```
python interpret.py \  
    --pos-sequences positive_test.fa.gz \  
    --neg-sequences negative_test.fa.gz \  
    --model-file two_layer.hdf5
```

This will load the trained two-layer CNN from **two_layer.hdf5** and **two_layer.weights.h5**, load positive and negative test sequences from **positive_test.fa.gz** and **negative_test.fa.gz**, predict the probabilities that the test sequences are bounded, and visualize the filters learned by the network.

Examine the output file **two_layer_architecture.png**, which shows a graphical view of the CNN layers and their sizes.

- What do the input and output dimensions of the first Convolution2D layer correspond to (ignore the Nones and 1's)? **(3 points)**
- What do the input and output dimensions of the Dense (i.e., fully connected) layer correspond to? **(3 points)**

Suppose we predict that all sequences with probability ≥ 0.5 are bounded (i.e., *positive*) and all others are unbounded (i.e., *negative*).

- How many true positives, false positives, true negatives and false negatives are predicted? **(3 points)**

The output files **motif1.png** and **motif2.png** visualize the true motifs used for generating the positive training and test data. The output file **two_layer_convolutional_filters.png** visualizes the filters learned in the first layer, i.e., the weights for the hidden channels in that layer.

- Discuss whether or not any of the learned filters resemble the true motifs. **(3 points)**
- In general, what concepts may the filters in the first layer have learned in general. **(3 points)**