gseapy Documentation

Release 0.10.0

Zhuoqing Fang

Aug 11, 2020

Contents

CHAPTER 1

GSEAPY: Gene Set Enrichment Analysis in Python.

Release notes : https://github.com/zqfang/GSEApy/releases

CHAPTER 2

GSEAPY is a python wrapper for **GSEA** and **Enrichr**.

GSEAPY has six subcommands: `gsea`, `prerank`, `ssgsea`, `replot` `enrichr`, `biomart`.

1. The `gsea` module produces **GSEA** results. The input requries a txt file(FPKM, Expected Counts, TPM, et.al), a cls file, and gene_sets file in gmt format.

2. The `prerank` module produces **Prerank tool** results. The input expects a pre-ranked gene list dataset with correlation values, which in .rnk format, and gene_sets file in gmt format. `prerank` module is an API to *GSEA* pre-rank tools.

3. The `ssgsea` module performs **single sample GSEA(ssGSEA)** analysis. The input expects a gene list with expression values(same with `.rnk` file, and gene_sets file in gmt format. ssGSEA enrichment score for the gene set as described by D. Barbie et al 2009.

4. The `replot` module reproduces GSEA desktop version results. The only input for GSEAPY is the location to GSEA Desktop output results.

5. The `enrichr` module enables you to perform gene set enrichment analysis using `Enrichr` API. Enrichr is open source and freely available online at: http://amp.pharm.mssm.edu/Enrichr . It runs very fast and generates results in txt format.

6. The `biomart` module helps you convert gene ids using BioMart API.

GSEAPY could be used for **RNA-seq, ChIP-seq, Microarry** data. It's used for convenient GO enrichments and produce **publishable quality figures** in python.

The full `GSEA` is far too extensive to describe here; see GSEA documentation for more information. All files' formats for GSEApy are identical to `GSEA` desktop version.

If you use gseapy, you should cite the original ‘‘GSEA‘‘ and ‘‘Enrichr‘‘ paper.

CHAPTER 3

Why GSEAPY

I would like to use Pandas to explore my data, but I did not find a convenient tool to do gene set enrichment analysis in python. So, here are my reasons:

- **Ability to run inside python interactive console without having to switch to R!!!**
- User friendly for both wet and dry lab users.
- Produce or reproduce publishable figures.
- Perform batch jobs easy.
- Easy to use in bash shell or your data analysis workflow, e.g. snakemake.

3.1 Welcome to GSEAPY's documentation!

3.1.1 GSEAPY: Gene Set Enrichment Analysis in Python.

3.1.2 GSEAPY is a python wrapper for GESA and Enrichr.

It's used for convenient GO enrichments and produce **publication-quality figures** from python.

GSEAPY could be used for **RNA-seq, ChIP-seq, Microarry** data.

Gene Set Enrichment Analysis (GSEA) is a computational method that determines whether an a priori defined set of genes shows statistically significant, concordant differences between two biological states (e.g. phenotypes).

The full `GSEA` is far too extensive to describe here; see GSEA documentation for more information.

Enrichr is open source and freely available online at: http://amp.pharm.mssm.edu/Enrichr .

3.1.3 Why GSEAPY

I would like to use Pandas to explore my data, but I did not find a convenient tool to do gene set enrichment analysis in python. So, here are my reasons:

- **Ability to run inside python interactive console without having to switch to R!!!**
- User-friendly for both wet and dry lab users.
- Produce or reproduce publishable figures.
- Perform batch jobs easy.
- Easy to use in bash shell or your data analysis workflow, e.g. snakemake.

3.1.4 GSEA Java version output:

This is an example of GSEA desktop application output

3.1.5 GSEAPY `Prerank` module output

Using the same data from `GSEA`, GSEAPY reproduces the example above.

Using `Prerank` or `replot` module will reproduce the same figure for GSEA Java desktop outputs

3.1.6 GSEAPY `enrichr` module

A graphical introduction of Enrichr

The only thing you need to prepare is a gene list file in txt format(one gene id per row), or a python list object.

Note: Enrichr uses a list of Entrez gene symbols as input. You should convert all gene names to uppercase.

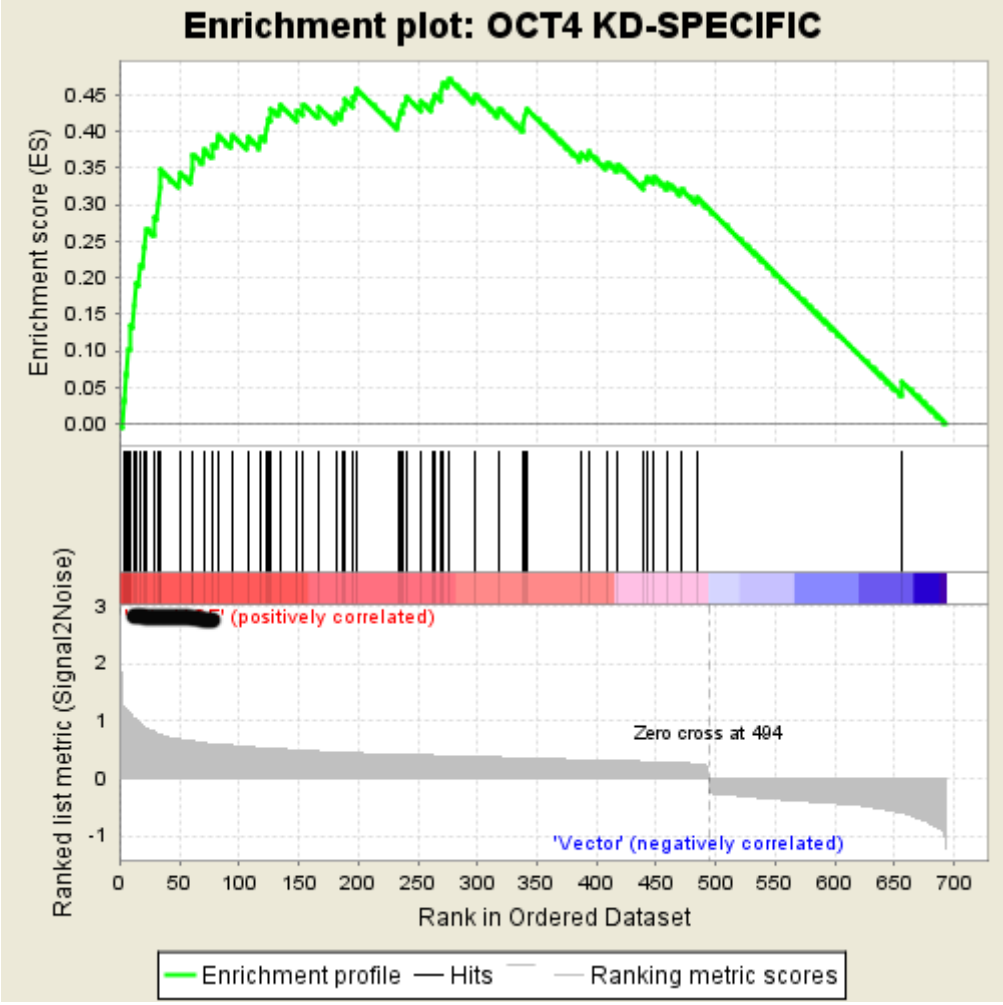
For example, both a list object and txt file are supported for `enrichr` API

```
# if you prefer to run gseapy.enrchr() inside python console, you could assign a list␣
↪object to
# gseapy like this.
gene_list = ['SCARA3', 'LOC100044683', 'CMBL', 'CLIC6', 'IL13RA1', 'TACSTD2', 'DKKL1',
               'CSF1', 'CITED1', 'SYNPO2L']
```

```
# an alternative way is that you could provide a gene list txt file which looks like␣
↪this:
with open('data/gene_list.txt') as genes:
    print(genes.read())


CTLA2B
SCARA3
```

(continues on next page)


Enrichment plot: OCT4 KD-SPECIFIC
Enrichment score (ES)
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
[redacted] (positively correlated)
Zero cross at 494
'Vector' (negatively correlated)
Ranked list metric (Signal2Noise)
3
2
1
0
-1
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
Rank in Ordered Dataset
Enrichment profile
Hits
Ranking metric scores

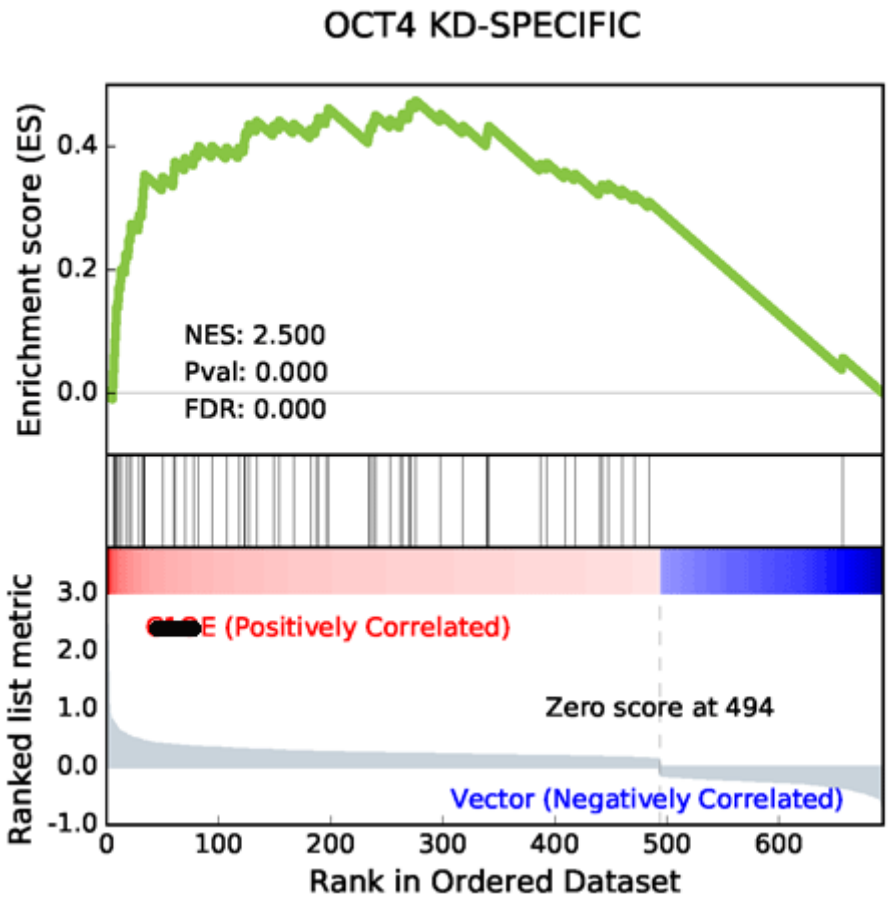


Fig. 1: Generated by GSEAPY

GSEAPY figures are supported by all matplotlib figure formats.

You can modify `GSEA` plots easily in .pdf files. Please Enjoy.

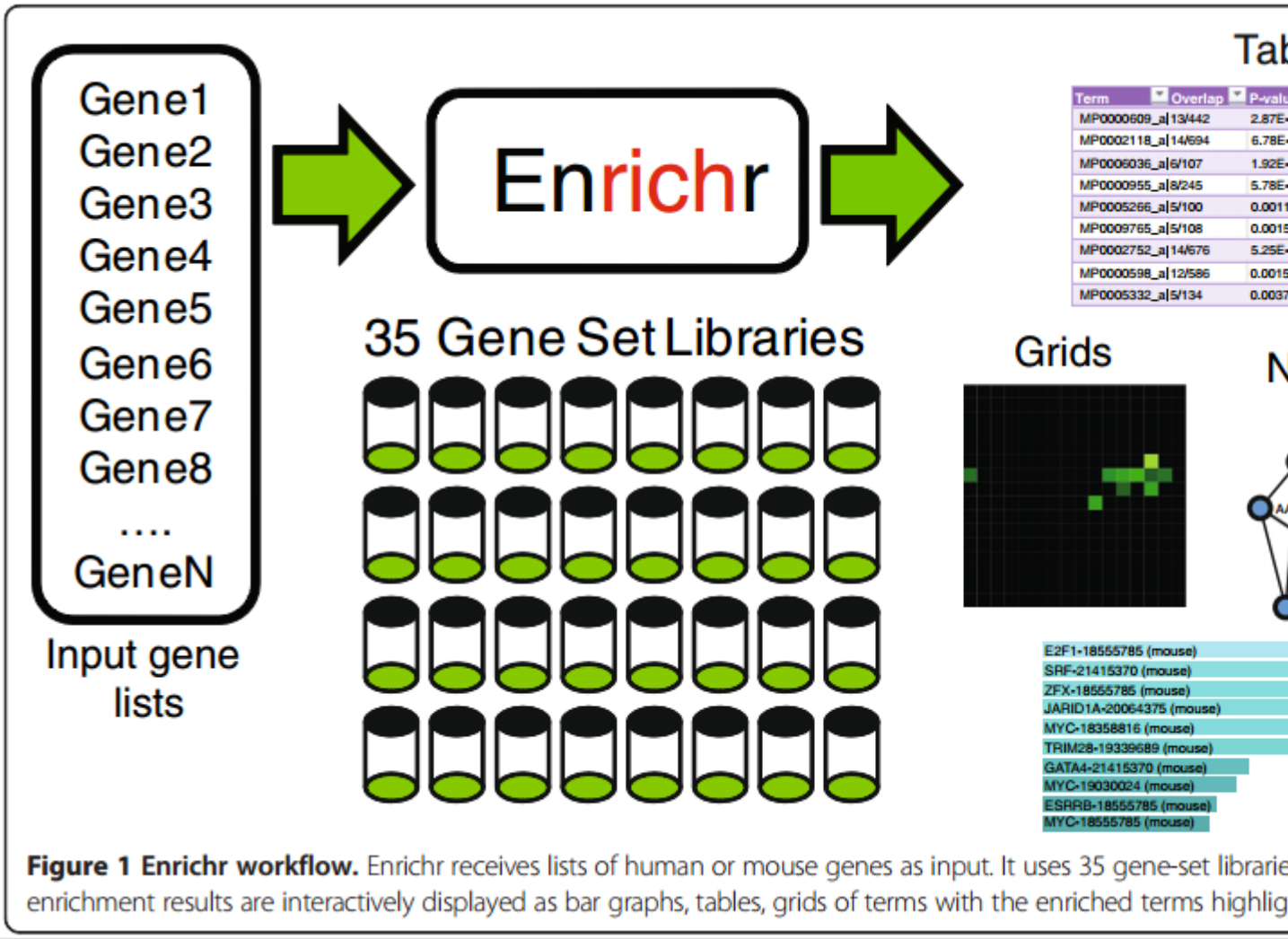


Figure 1 Enrichr workflow. Enrichr receives lists of human or mouse genes as input. It uses 35 gene-set librarie
enrichment results are interactively displayed as bar graphs, tables, grids of terms with the enriched terms highlig

(continued from previous page)

```
LOC100044683
CMBL
CLIC6
IL13RA1
TACSTD2
DKKL1
CSF1
CITED1
SYNPO2L
TINAGL1
PTX3
```

3.1.7 Installation

Install gseapy package from bioconda or pypi.

```
# if you have conda
$ conda install -c conda-forge -c bioconda gseapy

# or use pip to install the latest release
$ pip install gseapy
```

You may instead want to use the development version from Github, by running

```
$ pip install git+git://github.com/BioNinja/gseapy.git#egg=gseapy
```

3.1.8 Dependency

- Python 3+

Mandatory

- Numpy
- Scipy
- Pandas
- Matplotlib
- Requests(for enrichr API)

For API information to use this library, see the *Developmental Guide*.

3.2 Developmental Guide

3.2.1 Module APIs

gseapy.**gsea**()

Run Gene Set Enrichment Analysis.

Parameters

- **data** – Gene expression data table, Pandas DataFrame, gct file.
- **gene_sets** – Enrichr Library name or .gmt gene sets file or dict of gene sets. Same input with GSEA.
- **cls** – A list or a .cls file format required for GSEA.
- **outdir** (*str*) – Results output directory.
- **permutation_num** (*int*) – Number of permutations for significance computation. Default: 1000.
- **permutation_type** (*str*) – Permutation type, "phenotype" for phenotypes, "gene_set" for genes.
- **min_size** (*int*) – Minimum allowed number of genes from gene set also the data set. Default: 15.
- **max_size** (*int*) – Maximum allowed number of genes from gene set also the data set. Default: 500.
- **weighted_score_type** (*float*) – Refer to *algorithm.enrichment_score()*. Default:1.
- **method** – The method used to calculate a correlation or ranking. Default: 'log2_ratio_of_classes'. Others methods are:

  1. 'signal_to_noise'

     You must have at least three samples for each phenotype to use this metric. The larger the signal-to-noise ratio, the larger the differences of the means (scaled by the standard deviations); that is, the more distinct the gene expression is in each phenotype and the more the gene acts as a "class marker."

  2. 't_test'

     Uses the difference of means scaled by the standard deviation and number of samples. Note: You must have at least three samples for each phenotype to use this metric. The larger the tTest ratio, the more distinct the gene expression is in each phenotype and the more the gene acts as a "class marker."

  3. 'ratio_of_classes' (also referred to as fold change).

     Uses the ratio of class means to calculate fold change for natural scale data.

  4. 'diff_of_classes'

     Uses the difference of class means to calculate fold change for nature scale data

  5. 'log2_ratio_of_classes'

     Uses the log2 ratio of class means to calculate fold change for natural scale data. This is the recommended statistic for calculating fold change for log scale data.

- **ascending** (*bool*) – Sorting order of rankings. Default: False.

- **processes** (*int*) – Number of Processes you are going to use. Default: 1.
- **figsize** (*list*) – Matplotlib figsize, accept a tuple or list, e.g. [width,height]. Default: [6.5,6].
- **format** (*str*) – Matplotlib figure format. Default: 'pdf'.
- **graph_num** (*int*) – Plot graphs for top sets of each phenotype.
- **no_plot** (*bool*) – If equals to True, no figure will be drawn. Default: False.
- **seed** – Random seed. expect an integer. Default:None.
- **verbose** (*bool*) – Bool, increase output verbosity, print out progress of your job, Default: False.

Returns

Return a GSEA obj. All results store to a dictionary, obj.results, where contains:

```
| {es: enrichment score,
|  nes: normalized enrichment score,
|  p: P-value,
|  fdr: FDR,
|  size: gene set size,
|  matched_size: genes matched to the data,
|  genes: gene names from the data set
|  ledge_genes: leading edge genes}
```

gseapy.**prerank**()

Run Gene Set Enrichment Analysis with pre-ranked correlation defined by user.

Parameters

- **rnk** – pre-ranked correlation table or pandas DataFrame. Same input with GSEA .rnk file.
- **gene_sets** – Enrichr Library name or .gmt gene sets file or dict of gene sets. Same input with GSEA.
- **outdir** – results output directory.
- **permutation_num** (*int*) – Number of permutations for significance computation. Default: 1000.
- **min_size** (*int*) – Minimum allowed number of genes from gene set also the data set. Default: 15.
- **max_size** (*int*) – Maximum allowed number of genes from gene set also the data set. Defaults: 500.
- **weighted_score_type** (*str*) – Refer to *algorithm.enrichment_score()*. Default:1.
- **ascending** (*bool*) – Sorting order of rankings. Default: False.
- **processes** (*int*) – Number of Processes you are going to use. Default: 1.
- **figsize** (*list*) – Matplotlib figsize, accept a tuple or list, e.g. [width,height]. Default: [6.5,6].
- **format** (*str*) – Matplotlib figure format. Default: 'pdf'.
- **graph_num** (*int*) – Plot graphs for top sets of each phenotype.
- **no_plot** (*bool*) – If equals to True, no figure will be drawn. Default: False.

- **seed** – Random seed. expect an integer. Default:None.
- **verbose** (*bool*) – Bool, increase output verbosity, print out progress of your job, Default: False.

Returns

Return a Prerank obj. All results store to a dictionary, obj.results, where contains:

```
| {es: enrichment score,
|  nes: normalized enrichment score,
|  p: P-value,
|  fdr: FDR,
|  size: gene set size,
|  matched_size: genes matched to the data,
|  genes: gene names from the data set
|  ledge_genes: leading edge genes}
```

gseapy.**ssgsea**()

Run Gene Set Enrichment Analysis with single sample GSEA tool

Parameters

- **data** – Expression table, pd.Series, pd.DataFrame, GCT file, or .rnk file format.
- **gene_sets** – Enrichr Library name or .gmt gene sets file or dict of gene sets. Same input with GSEA.
- **outdir** – Results output directory.
- **sample_norm_method** (*str*) – "Sample normalization method. Choose from {'rank', 'log', 'log_rank'}. Default: rank.

  1. 'rank': Rank your expression data, and transform by 10000*rank_dat/gene_numbers
  2. 'log' : Do not rank, but transform data by log(data + exp(1)), while data = data[data<1] =1.
  3. 'log_rank': Rank your expression data, and transform by log(10000*rank_dat/gene_numbers+ exp(1))
  4. 'custom': Do nothing, and use your own rank value to calculate enrichment score.

see here: https://github.com/GSEA-MSigDB/ssGSEAProjection-gpmodule/blob/master/src/ssGSEAProjection.Library.R, line 86

Parameters

- **min_size** (*int*) – Minimum allowed number of genes from gene set also the data set. Default: 15.
- **max_size** (*int*) – Maximum allowed number of genes from gene set also the data set. Default: 2000.
- **permutation_num** (*int*) – Number of permutations for significance computation. Default: 0.
- **weighted_score_type** (*str*) – Refer to *algorithm.enrichment_score()*. Default:0.25.
- **scale** (*bool*) – If True, normalize the scores by number of genes in the gene sets.
- **ascending** (*bool*) – Sorting order of rankings. Default: False.
- **processes** (*int*) – Number of Processes you are going to use. Default: 1.

- **figsize** (*list*) – Matplotlib figsize, accept a tuple or list, e.g. [width,height]. Default: [7,6].
- **format** (*str*) – Matplotlib figure format. Default: 'pdf'.
- **graph_num** (*int*) – Plot graphs for top sets of each phenotype.
- **no_plot** (*bool*) – If equals to True, no figure will be drawn. Default: False.
- **seed** – Random seed. expect an integer. Default:None.
- **verbose** (*bool*) – Bool, increase output verbosity, print out progress of your job, Default: False.

Returns

Return a ssGSEA obj. All results store to a dictionary, access enrichment score by obj.resultsOnSamples, and normalized enrichment score by obj.res2d. if permutation_num > 0, additional results contain:

```
| {es: enrichment score,
|  nes: normalized enrichment score,
|  p: P-value,
|  fdr: FDR,
|  size: gene set size,
|  matched_size: genes matched to the data,
|  genes: gene names from the data set
|  ledge_genes: leading edge genes, if permutation_num >0}
```

gseapy.**enrichr**()

Enrichr API.

Parameters

- **gene_list** – Flat file with list of genes, one gene id per row, or a python list object
- **gene_sets** – Enrichr Library to query. Required enrichr library name(s). Separate each name by comma.
- **organism** – Enrichr supported organism. Select from (human, mouse, yeast, fly, fish, worm). see here for details: https://amp.pharm.mssm.edu/modEnrichr
- **description** – name of analysis. optional.
- **outdir** – Output file directory
- **cutoff** (*float*) – Show enriched terms which Adjusted P-value < cutoff. Only affects the output figure. Default: 0.05
- **background** (*int*) – BioMart dataset name for retrieving background gene information. This argument only works when gene_sets input is a gmt file or python dict. You could also specify a number by yourself, e.g. total expressed genes number. In this case, you will skip retrieving background infos from biomart.

Use the code below to see valid background dataset names from BioMart. Here are example code:: >>> from gseapy.parser import Biomart >>> bm = Biomart(verbose=False, host="asia.ensembl.org") >>> ## view validated marts >>> marts = bm.get_marts() >>> ## view validated dataset >>> datasets = bm.get_datasets(mart='ENSEMBL_MART_ENSEMBL')

Parameters

- **format** (*str*) – Output figure format supported by matplotlib,('pdf','png','eps'...). Default: 'pdf'.

- **figsize** (*list*) – Matplotlib figsize, accept a tuple or list, e.g. (width,height). Default: (6.5,6).
- **no_plot** (*bool*) – If equals to True, no figure will be drawn. Default: False.
- **verbose** (*bool*) – Increase output verbosity, print out progress of your job, Default: False.

Returns An Enrichr object, which obj.res2d stores your last query, obj.results stores your all queries.

gseapy.**replot**()

The main function to reproduce GSEA desktop outputs.

Parameters

- **indir** – GSEA desktop results directory. In the sub folder, you must contain edb file folder.
- **outdir** – Output directory.
- **weighted_score_type** (*float*) – weighted score type. choose from {0,1,1.5,2}. Default: 1.
- **figsize** (*list*) – Matplotlib output figure figsize. Default: [6.5,6].
- **format** (*str*) – Matplotlib output figure format. Default: 'pdf'.
- **min_size** (*int*) – Min size of input genes presented in Gene Sets. Default: 3.
- **max_size** (*int*) – Max size of input genes presented in Gene Sets. Default: 5000. You are not encouraged to use min_size, or max_size argument in *replot()* function. Because gmt file has already been filtered.
- **verbose** – Bool, increase output verbosity, print out progress of your job, Default: False.

Returns Generate new figures with selected figure format. Default: 'pdf'.

3.2.2 Algorithm

gseapy.algorithm.**enrichment_score**(*gene_list*, *correl_vector*, *gene_set*, *weighted_score_type=1*, *nperm=1000*, *rs=None*, *single=False*, *scale=False*)

This is the most important function of GSEApy. It has the same algorithm with GSEA and ssGSEA.

Parameters

- **gene_list** – The ordered gene list gene_name_list, rank_metric.index.values
- **gene_set** – gene_sets in gmt file, please use gsea_gmt_parser to get gene_set.
- **weighted_score_type** – It's the same with gsea's weighted_score method. Weighting by the correlation is a very reasonable choice that allows significant gene sets with less than perfect coherence. options: 0(classic),1,1.5,2. default:1. if one is interested in penalizing sets for lack of coherence or to discover sets with any type of nonrandom distribution of tags, a value p < 1 might be appropriate. On the other hand, if one uses sets with large number of genes and only a small subset of those is expected to be coherent, then one could consider using p > 1. Our recommendation is to use p = 1 and use other settings only if you are very experienced with the method and its behavior.
- **correl_vector** – A vector with the correlations (e.g. signal to noise scores) corresponding to the genes in the gene list. Or rankings, rank_metric.values
- **nperm** – Only use this parameter when computing esnull for statistical testing. Set the esnull value equal to the permutation number.
- **rs** – Random state for initializing gene list shuffling. Default: seed=None

Returns

ES: Enrichment score (real number between -1 and +1)

ESNULL: Enrichment score calculated from random permutations.

Hits_Indices: Index of a gene in gene_list, if gene is included in gene_set.

RES: Numerical vector containing the running enrichment score for all locations in the gene list .

gseapy.algorithm.**enrichment_score_tensor**(*gene_mat*, *cor_mat*, *gene_sets*, *weighted_score_type*, *nperm=1000*, *rs=None*, *single=False*, *scale=False*)

Next generation algorithm of GSEA and ssGSEA.

Parameters

- **gene_mat** – the ordered gene list(vector) with or without gene indices matrix.
- **cor_mat** – correlation vector or matrix (e.g. signal to noise scores) corresponding to the genes in the gene list or matrix.
- **gene_sets** (*dict*) – gmt file dict.
- **weighted_score_type** (*float*) – weighting by the correlation. options: 0(classic), 1, 1.5, 2. default:1 for GSEA and 0.25 for ssGSEA.
- **nperm** (*int*) – permutation times.
- **scale** (*bool*) – If True, normalize the scores by number of genes_mat.
- **single** (*bool*) – If True, use ssGSEA algorithm, otherwise use GSEA.
- **rs** – Random state for initialize gene list shuffling. Default: seed=None

Returns

a tuple contains:

```
| ES: Enrichment score (real number between -1 and +1), for ssGSEA, set scale eq to True.
| ESNULL: Enrichment score calculated from random permutation.
| Hits_Indices: Indices of genes if genes are included in gene_set.
| RES: The running enrichment score for all locations in the gene list.
```

gseapy.algorithm.**gsea_compute**(*data*, *gmt*, *n*, *weighted_score_type*, *permutation_type*, *method*, *pheno_pos*, *pheno_neg*, *classes*, *ascending*, *processes=1*, *seed=None*, *single=False*, *scale=False*)

compute enrichment scores and enrichment nulls.

Parameters

- **data** – preprocessed expression dataframe or a pre-ranked file if prerank=True.
- **gmt** (*dict*) – all gene sets in .gmt file. need to call load_gmt() to get results.
- **n** (*int*) – permutation number. default: 1000.
- **method** (*str*) – ranking_metric method. see above.
- **pheno_pos** (*str*) – one of labels of phenotype's names.
- **pheno_neg** (*str*) – one of labels of phenotype's names.

- **classes** (*list*) – a list of phenotype labels, to specify which column of dataframe belongs to what category of phenotype.
- **weighted_score_type** (*float*) – default:1
- **ascending** (*bool*) – sorting order of rankings. Default: False.
- **seed** – random seed. Default: np.random.RandomState()
- **scale** (*bool*) – if true, scale es by gene number.

Returns

a tuple contains:

```
| zipped results of es, nes, pval, fdr.
| nested list of hit indices of input gene_list.
| nested list of ranked enrichment score of each input gene_sets.
| list of enriched terms
```

gseapy.algorithm.**gsea_compute_tensor**(*data*, *gmt*, *n*, *weighted_score_type*, *permutation_type*, *method*, *pheno_pos*, *pheno_neg*, *classes*, *ascending*, *processes=1*, *seed=None*, *single=False*, *scale=False*)

compute enrichment scores and enrichment nulls.

Parameters

- **data** – preprocessed expression dataframe or a pre-ranked file if prerank=True.
- **gmt** (*dict*) – all gene sets in .gmt file. need to call load_gmt() to get results.
- **n** (*int*) – permutation number. default: 1000.
- **method** (*str*) – ranking_metric method. see above.
- **pheno_pos** (*str*) – one of labels of phenotype's names.
- **pheno_neg** (*str*) – one of labels of phenotype's names.
- **classes** (*list*) – a list of phenotype labels, to specify which column of dataframe belongs to what category of phenotype.
- **weighted_score_type** (*float*) – default:1
- **ascending** (*bool*) – sorting order of rankings. Default: False.
- **seed** – random seed. Default: np.random.RandomState()
- **scale** (*bool*) – if true, scale es by gene number.

Returns

a tuple contains:

```
| zipped results of es, nes, pval, fdr.
| nested list of hit indices of input gene_list.
| nested list of ranked enrichment score of each input gene_sets.
| list of enriched terms
```

gseapy.algorithm.**gsea_fdr**(*nEnrichmentScores*, *nEnrichmentNulls*)

Create a histogram of all NES(S,pi) over all S and pi. Use this null distribution to compute an FDR q value.

Parameters

- **nEnrichmentScores** – normalized ES

- **nEnrichmentNulls** – normalized ESnulls

Returns FDR

gseapy.algorithm.**gsea_pval**(*es*, *esnull*)
Compute nominal p-value.

From article (PNAS): estimate nominal p-value for S from esnull by using the positive or negative portion of the distribution corresponding to the sign of the observed ES(S).

gseapy.algorithm.**gsea_significance**(*enrichment_scores*, *enrichment_nulls*)
Compute nominal pvals, normalized ES, and FDR q value.

For a given NES(S) = NES* >= 0. The FDR is the ratio of the percentage of all (S,pi) with NES(S,pi) >= 0, whose NES(S,pi) >= NES*, divided by the percentage of observed S wih NES(S) >= 0, whose NES(S) >= NES*, and similarly if NES(S) = NES* <= 0.

gseapy.algorithm.**normalize**(*es*, *esnull*)
normalize the ES(S,pi) and the observed ES(S), separately rescaling the positive and negative scores by dividing the mean of the ES(S,pi).

return: NES, NESnull

gseapy.algorithm.**ranking_metric**(*df*, *method*, *pos*, *neg*, *classes*, *ascending*)
The main function to rank an expression table.

Parameters

- **df** – gene_expression DataFrame.
- **method** – The method used to calculate a correlation or ranking. Default: 'log2_ratio_of_classes'. Others methods are:

  1. 'signal_to_noise' (s2n) or 'abs_signal_to_noise' (abs_s2n)

     You must have at least three samples for each phenotype to use this metric. The larger the signal-to-noise ratio, the larger the differences of the means (scaled by the standard deviations); that is, the more distinct the gene expression is in each phenotype and the more the gene acts as a "class marker."

  2. 't_test'

     Uses the difference of means scaled by the standard deviation and number of samples. Note: You must have at least three samples for each phenotype to use this metric. The larger the tTest ratio, the more distinct the gene expression is in each phenotype and the more the gene acts as a "class marker."

  3. 'ratio_of_classes' (also referred to as fold change).

     Uses the ratio of class means to calculate fold change for natural scale data.

  4. 'diff_of_classes'

     Uses the difference of class means to calculate fold change for natural scale data

  5. 'log2_ratio_of_classes'

     Uses the log2 ratio of class means to calculate fold change for natural scale data. This is the recommended statistic for calculating fold change for log scale data.

- **pos** (*str*) – one of labels of phenotype's names.
- **neg** (*str*) – one of labels of phenotype's names.
- **classes** (*dict*) – column id to group mapping.

- **ascending** (*bool*) – bool or list of bool. Sort ascending vs. descending.

Returns

returns a pd.Series of correlation to class of each variable. Gene_name is index, and value is rankings.

visit here for more docs: http://software.broadinstitute.org/gsea/doc/GSEAUserGuideFrame.html

gseapy.algorithm.**ranking_metric_tensor**(*exprs*, *method*, *permutation_num*, *pos*, *neg*, *classes*, *ascending*, *rs=None*)

Build shuffled ranking matrix when permutation_type eq to phenotype.

Parameters

- **exprs** – gene_expression DataFrame, gene_name indexed.
- **method** (*str*) – calculate correlation or ranking. methods including: 1. 'signal_to_noise' (s2n) or 'abs_signal_to_noise' (abs_s2n). 2. 't_test'. 3. 'ratio_of_classes' (also referred to as fold change). 4. 'diff_of_classes'. 5. 'log2_ratio_of_classes'.
- **permuation_num** (*int*) – how many times of classes is being shuffled
- **pos** (*str*) – one of labels of phenotype's names.
- **neg** (*str*) – one of labels of phenotype's names.
- **classes** (*list*) – a list of phenotype labels, to specify which column of dataframe belongs to what class of phenotype.
- **ascending** (*bool*) – bool. Sort ascending vs. descending.

Returns

returns two 2d ndarray with shape (nperm, gene_num).

cor_mat_indices: the indices of sorted and permutated (exclude last row) ranking matrix.

cor_mat: sorted and permutated (exclude last row) ranking matrix.

3.2.3 Enrichr

class gseapy.enrichr.**Enrichr**(*gene_list*, *gene_sets*, *organism='human'*, *descriptions=''*, *outdir='Enrichr'*, *cutoff=0.05*, *background='hsapiens_gene_ensembl'*, *format='pdf'*, *figsize=(6.5, 6)*, *top_term=10*, *no_plot=False*, *verbose=False*)

Enrichr API

check_genes(*gene_list*, *usr_list_id*)

Compare the genes sent and received to get successfully recognized genes

enrich(*gmt*)

use local mode

p = p-value computed using the Fisher exact test (Hypergeometric test)

Not implemented here:

combine score = log(p)·z

see here: http://amp.pharm.mssm.edu/Enrichr/help#background&q=4

columns contain:

Term Overlap P-value Adjusted_P-value Genes

get_background()
get background gene

get_libraries()
return active enrichr library name. Official API

get_organism()
Select Enrichr organism from below:

Human & Mouse, H. sapiens & M. musculus Fly, D. melanogaster Yeast, S. cerevisiae Worm, C. elegans Fish, D. rerio

get_results(*gene_list*)
Enrichr API

parse_genelists()
parse gene list

parse_genesets()
parse gene_sets input file type

prepare_outdir()
create temp directory.

run()
run enrichr for one sample gene list but multi-libraries

send_genes(*gene_list*, *url*)
send gene list to enrichr server

gseapy.enrichr.**enrichr**(*gene_list*, *gene_sets*, *organism='human'*, *description=''*, *outdir='Enrichr'*, *background='hsapiens_gene_ensembl'*, *cutoff=0.05*, *format='pdf'*, *figsize=(8, 6)*, *top_term=10*, *no_plot=False*, *verbose=False*)

Enrichr API.

Parameters

- **gene_list** – Flat file with list of genes, one gene id per row, or a python list object
- **gene_sets** – Enrichr Library to query. Required enrichr library name(s). Separate each name by comma.
- **organism** – Enrichr supported organism. Select from (human, mouse, yeast, fly, fish, worm). see here for details: https://amp.pharm.mssm.edu/modEnrichr
- **description** – name of analysis. optional.
- **outdir** – Output file directory
- **cutoff** (*float*) – Show enriched terms which Adjusted P-value < cutoff. Only affects the output figure. Default: 0.05
- **background** (*int*) – BioMart dataset name for retrieving background gene information. This argument only works when gene_sets input is a gmt file or python dict. You could also specify a number by yourself, e.g. total expressed genes number. In this case, you will skip retrieving background infos from biomart.

Use the code below to see valid background dataset names from BioMart. Here are example code:: >>> from gseapy.parser import Biomart >>> bm = Biomart(verbose=False, host="asia.ensembl.org") >>>

view validated marts >>> marts = bm.get_marts() >>> ## view validated dataset >>> datasets = bm.get_datasets(mart='ENSEMBL_MART_ENSEMBL')

Parameters

- **format** (*str*) – Output figure format supported by matplotlib,('pdf','png','eps'...). Default: 'pdf'.
- **figsize** (*list*) – Matplotlib figsize, accept a tuple or list, e.g. (width,height). Default: (6.5,6).
- **no_plot** (*bool*) – If equals to True, no figure will be drawn. Default: False.
- **verbose** (*bool*) – Increase output verbosity, print out progress of your job, Default: False.

Returns An Enrichr object, which obj.res2d stores your last query, obj.results stores your all queries.

3.2.4 Parser

class gseapy.parser.**Biomart**(*host='www.ensembl.org'*, *verbose=False*)
query from BioMart

get_attributes(*dataset*)
Get available attritbutes from dataset you've selected

get_datasets(*mart='ENSEMBL_MART_ENSEMBL'*)
Get available datasets from mart you've selected

get_filters(*dataset*)
Get available filters from dataset you've selected

get_marts()
Get available marts and their names.

query(*dataset='hsapiens_gene_ensembl'*, *attributes=[]*, *filters={}*, *filename=None*)
mapping ids using BioMart.

Parameters

- **dataset** – str, default: 'hsapiens_gene_ensembl'
- **attributes** – str, list, tuple
- **filters** – dict, {'filter name': list(filter value)}
- **host** – www.ensembl.org, asia.ensembl.org, useast.ensembl.org

Returns a dataframe contains all attributes you selected.

Note: it will take a couple of minutes to get the results. A xml template for querying biomart. (see https://gist.github.com/keithshep/7776579) Example:: >>> import requests >>> exampleTaxonomy = "mmusculus_gene_ensembl" >>> exampleGene = "ENSMUSG00000086981,ENSMUSG00000086982,ENSMUSG00000086983" >>> urlTemplate = ''http://ensembl.org/biomart/martservice?query='' ''<?xml version="1.0" encoding="UTF-8"?>'' ''<!DOCTYPE Query>'' ''<Query virtualSchemaName="default" formatter="CSV" header="0" uniqueRows="0" count="" datasetConfigVersion="0.6">'' ''<Dataset name="%s" interface="default"><Filter name="ensembl_gene_id" value="%s"/>'' ''<Attribute name="ensembl_gene_id"/><Attribute name="ensembl_transcript_id"/>'' ''<Attribute name="transcript_start"/><Attribute name="transcript_end"/>'' ''<Attribute name="exon_chrom_start"/><Attribute name="exon_chrom_end"/>'' ''</Dataset>'' ''</Query>'' >>> exampleURL = urlTemplate % (exampleTaxonomy, exampleGene) >>> req = requests.get(exampleURL, stream=True)

gseapy.parser.**get_library_name**(*database='Human'*)
return enrichr active enrichr library name. :param str database: Select one from { 'Human', 'Mouse', 'Yeast', 'Fly', 'Fish', 'Worm' } :return: a list of enrichr libraries from selected database

gseapy.parser.**gsea_cls_parser**(*cls*)
Extract class(phenotype) name from .cls file.

Parameters **cls** – the a class list instance or .cls file which is identical to GSEA input .

Returns phenotype name and a list of class vector.

gseapy.parser.**gsea_edb_parser**(*results_path*)
Parse results.edb file stored under **edb** file folder.

Parameters **results_path** – the .results file located inside edb folder.

Returns a dict contains enrichment_term, hit_index,nes, pval, fdr.

gseapy.parser.**gsea_gmt_parser**(*gmt*, *min_size=3*, *max_size=1000*, *gene_list=None*)
Parse gene_sets.gmt(gene set database) file or download from enrichr server.

Parameters

- **gmt** – the gene_sets.gmt file of GSEA input or an enrichr library name. checkout full enrichr library name here: http://amp.pharm.mssm.edu/Enrichr/#stats
- **min_size** – Minimum allowed number of genes from gene set also the data set. Default: 3.
- **max_size** – Maximum allowed number of genes from gene set also the data set. Default: 5000.
- **gene_list** – Used for filtering gene set. Only used this argument for `call()` method.

Returns Return a new filtered gene set database dictionary.

DO NOT filter gene sets, when use `replot()`. Because `GSEA` Desktop have already done this for you.

3.2.5 Graph

class gseapy.plot.**MidpointNormalize**(*vmin=None*, *vmax=None*, *midpoint=None*, *clip=False*)

gseapy.plot.**adjust_spines**(*ax*, *spines*)
function for removing spines and ticks.

Parameters

- **ax** – axes object
- **spines** – a list of spines names to keep. e.g [left, right, top, bottom] if spines = []. remove all spines and ticks.

gseapy.plot.**barplot**(*df*, *column='Adjusted P-value'*, *title=''*, *cutoff=0.05*, *top_term=10*, *figsize=(6.5, 6)*, *color='salmon'*, *ofname=None*, ***kwargs*)
Visualize enrichr results.

Parameters

- **df** – GSEApy DataFrame results.
- **column** – which column of DataFrame to show. Default: Adjusted P-value
- **title** – figure title.
- **cutoff** – terms with 'column' value < cut-off are shown.

- **top_term** – number of top enriched terms to show.
- **figsize** – tuple, matplotlib figsize.
- **color** – color for bars.
- **ofname** – output file name. If None, don't save figure

gseapy.plot.**dotplot**(*df*, *column='Adjusted P-value'*, *title=''*, *cutoff=0.05*, *top_term=10*, *sizes=None*, *norm=None*, *legend=True*, *figsize=(6, 5.5)*, *cmap='RdBu_r'*, *ofname=None*, ***kwargs*)

Visualize enrichr results.

Parameters

- **df** – GSEApy DataFrame results.
- **column** – which column of DataFrame to show. Default: Adjusted P-value
- **title** – figure title
- **cutoff** – terms with 'column' value < cut-off are shown.
- **top_term** – number of enriched terms to show.
- **ascending** – bool, the order of y axis.
- **sizes** – tuple, (min, max) scatter size. Not functional for now
- **norm** – maplotlib.colors.Normalize object.
- **legend** – bool, whether to show legend.
- **figsize** – tuple, figure size.
- **cmap** – matplotlib colormap
- **ofname** – output file name. If None, don't save figure

gseapy.plot.**gseaplot**(*rank_metric*, *term*, *hit_indices*, *nes*, *pval*, *fdr*, *RES*, *pheno_pos=''*, *pheno_neg=''*, *figsize=(6, 5.5)*, *cmap='seismic'*, *ofname=None*, ***kwargs*)

This is the main function for reproducing the gsea plot.

Parameters

- **rank_metric** – pd.Series for rankings, rank_metric.values.
- **term** – gene_set name
- **hit_indices** – hits indices of rank_metric.index presented in gene set S.
- **nes** – Normalized enrichment scores.
- **pval** – nominal p-value.
- **fdr** – false discovery rate.
- **RES** – running enrichment scores.
- **pheno_pos** – phenotype label, positive correlated.
- **pheno_neg** – phenotype label, negative correlated.
- **figsize** – matplotlib figsize.
- **ofname** – output file name. If None, don't save figure

gseapy.plot.**heatmap**(*df*, *z_score=None*, *title=''*, *figsize=(5, 5)*, *cmap='RdBu_r'*, *xticklabels=True*, *yticklabels=True*, *ofname=None*, ***kwargs*)

Visualize the dataframe.

Parameters

- **df** – DataFrame from expression table.
- **z_score** – z_score axis{0, 1}. If None, don't normalize data.
- **title** – gene set name.
- **outdir** – path to save heatmap.
- **figsize** – heatmap figsize.
- **cmap** – matplotlib colormap.
- **ofname** – output file name. If None, don't save figure

gseapy.plot.**zscore**(*data2d*, *axis=0*)

Standardize the mean and variance of the data axis Parameters.

Parameters

- **data2d** – DataFrame to normalize.
- **axis** – int, Which axis to normalize across. If 0, normalize across rows, if 1, normalize across columns. If None, don't change data

Returns Normalized DataFrame. Normalized data with a mean of 0 and variance of 1 across the specified axis.

3.3 A Protocol to Prepare files for GSEAPY

As a biological researcher, I like protocols, so as other researchers, too.

Here is an short tutorial to walk you through gseapy.

For file format explanation, please see here

In order to run gseapy successfully, install gseapy use pip.

```
pip install gseapy

# if you have conda
conda install -c bioconda gseapy
```

3.3.1 Use `gsea` command, or `gsea()`

Follow the steps blow.

One thing you should know is that the gseapy input files are totally the same as GSEA desktop required. You can use these files below to run GSEA desktop, too.

1. Prepare an tabular text file of gene expression like this:

RNA-seq,ChIP-seq, Microarry data are all supported.

Here is to see what the structure of expression table looks like, you don't have to run commands below:

```
import pandas as pd
df = pd.read_table('./test/gsea_data.txt')
df.head()

#or assign df to the parameter 'data'
```

2. An cls file is also expected.

This file is used to specify column attributes in step 1, just like GSEA asked.

An example of cls file looks like below.

```
with open('gsea/edb/C1OE.cls') as cls:
    print(cls.read())

# or assign a list object to parameter 'cls' like this
# cls=['C1OE', 'C1OE', 'C1OE', 'Vector', 'Vector', 'Vector']
```

```
6 2 1
# C1OE Vector
C1OE C1OE C1OE Vector Vector Vector
```

The first line specify the total samples and phenotype numbers. Leave number 1 alway be 1.
The second line specify the phenotype class(name).
The third line specify column attributes in setp 1.

3. Gene_sets file in gmt format.

All you need to do is to download gene set database file from GSEA website.

Or you could use enrichr library. In this case, just provide library name to parameter 'gene_sets'

If you would like to use you own gene_sets.gmts files, build such a file use excel, and then rename to gene_sets.gmt.

An example of gmt file looks like below:

```
with open('gsea/edb/gene_sets.gmt') as gmt:
    print(gmt.read())
```

```
ES-SPECIFIC Arid3a_used     ACTA1   CALML4  CORO1A  DHX58   DPYS    EGR1    ESRRB
↪GLI2    GPX2    HCK     INHBB
HDAC-UNIQUE     Arid3a_used 1700017B05RIK   8430427H17RIK   ABCA3   ANKRD44 ARL4A
↪BNC2    CLDN3
XEN-SPECIFIC        Arid3a_used     1110036O03RIK   A130022J15RIK   B2M     B3GALNT1
↪       CBX4    CITED1  CLU     CTSH    CYP26A1
GATA-SPECIFIC       Arid3a_used     1200009I06RIK   5430407P10RIK   BAIAP2L1
↪BMP8B   CITED1  CLDN3   COBLL1  CORO1A  CRYAB   CTDSPL  DKKL1
TS-SPECIFIC Arid3a_used     5430407P10RIK   AFAP1L1 AHNAK   ANXA2   ANXA3   ANXA5
↪B2M     BIK     BMP8B   CAMK1D  CBX4    CLDN3   CSRP1   DKKL1   DSC2
```

3.3.2 Use `enrichr` command, or `enrichr()`

The only thing you need to prepare is a gene list file.

Note: Enrichr uses a list of Entrez gene symbols as input.

For `enrichr` , you could assign a list object

```
# assign a list object to enrichr
l = ['SCARA3', 'LOC100044683', 'CMBL', 'CLIC6', 'IL13RA1', 'TACSTD2', 'DKKL1', 'CSF1',
     'SYNPO2L', 'TINAGL1', 'PTX3', 'BGN', 'HERC1', 'EFNA1', 'CIB2', 'PMP22', 'TMEM173
↪']

gseapy.enrichr(gene_list=l, description='pathway', gene_sets='KEGG_2016', outfile=
↪'test')
```

or a gene list file in txt format(one gene id per row)

```
gseapy.enrichr(gene_list='gene_list.txt', description='pathway', gene_sets='KEGG_2016
↪', outfile='test')
```

Let's see what the txt file looks like.

```
with open('data/gene_list.txt') as genes:
    print(genes.read())
```

```
CTLA2B
SCARA3
LOC100044683
CMBL
CLIC6
IL13RA1
TACSTD2
DKKL1
CSF1
CITED1
SYNPO2L
TINAGL1
PTX3
```

Select the library you want to do enrichment analysis. For a view all avilable libraries,run

```
#s get_library_name(), it will print out all library names.
import gseapy
names = gseapy.get_library_name()
print(names)
```

```
 ['Genome_Browser_PWMs',
'TRANSFAC_and_JASPAR_PWMs',
'ChEA_2013',
'Drug_Perturbations_from_GEO_2014',
'ENCODE_TF_ChIP-seq_2014',
'BioCarta_2013',
'Reactome_2013',
'WikiPathways_2013',
'Disease_Signatures_from_GEO_up_2014',
'KEGG_2013',
'TF-LOF_Expression_from_GEO',
```

(continues on next page)

(continued from previous page)

```
'TargetScan_microRNA',
'PPI_Hub_Proteins',
'GO_Molecular_Function_2015',
'GeneSigDB',
'Chromosome_Location',
'Human_Gene_Atlas',
'Mouse_Gene_Atlas',
'GO_Cellular_Component_2015',
'GO_Biological_Process_2015',
'Human_Phenotype_Ontology',
'Epigenomics_Roadmap_HM_ChIP-seq',
'KEA_2013',
'NURSA_Human_Endogenous_Complexome',
'CORUM',
'SILAC_Phosphoproteomics',
'MGI_Mammalian_Phenotype_Level_3',
'MGI_Mammalian_Phenotype_Level_4',
'Old_CMAP_up',
'Old_CMAP_down',
'OMIM_Disease',
'OMIM_Expanded',
'VirusMINT',
'MSigDB_Computational',
'MSigDB_Oncogenic_Signatures',
'Disease_Signatures_from_GEO_down_2014',
'Virus_Perturbations_from_GEO_up',
'Virus_Perturbations_from_GEO_down',
'Cancer_Cell_Line_Encyclopedia',
'NCI-60_Cancer_Cell_Lines',
'Tissue_Protein_Expression_from_ProteomicsDB',
'Tissue_Protein_Expression_from_Human_Proteome_Map',
'HMDB_Metabolites',
'Pfam_InterPro_Domains',
'GO_Biological_Process_2013',
'GO_Cellular_Component_2013',
'GO_Molecular_Function_2013',
'Allen_Brain_Atlas_up',
'ENCODE_TF_ChIP-seq_2015',
'ENCODE_Histone_Modifications_2015',
'Phosphatase_Substrates_from_DEPOD',
'Allen_Brain_Atlas_down',
'ENCODE_Histone_Modifications_2013',
'Achilles_fitness_increase',
'Achilles_fitness_decrease',
'MGI_Mammalian_Phenotype_2013',
'BioCarta_2015',
'HumanCyc_2015',
'KEGG_2015',
'NCI-Nature_2015',
'Panther_2015',
'WikiPathways_2015',
'Reactome_2015',
'ESCAPE',
'HomoloGene',
'Disease_Perturbations_from_GEO_down',
'Disease_Perturbations_from_GEO_up',
'Drug_Perturbations_from_GEO_down',
```

(continues on next page)

(continued from previous page)

```
'Genes_Associated_with_NIH_Grants',
'Drug_Perturbations_from_GEO_up',
'KEA_2015',
'Single_Gene_Perturbations_from_GEO_up',
'Single_Gene_Perturbations_from_GEO_down',
'ChEA_2015',
'dbGaP',
'LINCS_L1000_Chem_Pert_up',
'LINCS_L1000_Chem_Pert_down',
'GTEx_Tissue_Sample_Gene_Expression_Profiles_down',
'GTEx_Tissue_Sample_Gene_Expression_Profiles_up',
'Ligand_Perturbations_from_GEO_down',
'Aging_Perturbations_from_GEO_down',
'Aging_Perturbations_from_GEO_up',
'Ligand_Perturbations_from_GEO_up',
'MCF7_Perturbations_from_GEO_down',
'MCF7_Perturbations_from_GEO_up',
'Microbe_Perturbations_from_GEO_down',
'Microbe_Perturbations_from_GEO_up',
'LINCS_L1000_Ligand_Perturbations_down',
'LINCS_L1000_Ligand_Perturbations_up',
'LINCS_L1000_Kinase_Perturbations_down',
'LINCS_L1000_Kinase_Perturbations_up',
'Reactome_2016',
'KEGG_2016',
'WikiPathways_2016',
'ENCODE_and_ChEA_Consensus_TFs_from_ChIP-X',
'Kinase_Perturbations_from_GEO_down',
'Kinase_Perturbations_from_GEO_up',
'BioCarta_2016',
'Humancyc_2016',
'NCI-Nature_2016',
'Panther_2016']
```

for for details, please track the offical links: http://amp.pharm.mssm.edu/Enrichr/

3.3.3 Use `replot` Command, or `replot()`

You may also want to use `replot()` to reproduce `GSEA` desktop plots.

The only input of `replot()` is the directory of `GSEA` desktop output.

The input directory(e.g. gsea), must contained **edb** folder, gseapy need 4 data files inside edb folder.The gsea document tree looks like this:

```
gsea
└─edb
    └─test.cls
    └─gene_sets.gmt
    └─gsea_data.rnk
    └─results.edb
```

After this, you can start to run gseapy.

```
import gseapy
gseapy.replot(indir ='gsea', outdir = 'gseapy_out')
```

If you prefer to run in command line, it's more simple.

```
gseapy replot -i gsea -o gseapy_out
```

For advanced usage of library,see the *Developmental Guide*.

3.4 GSEAPY Example

Examples to use `GSEApy` inside python console

[1]:
```
%matplotlib inline
%config InlineBackend.figure_format='retina' # mac
%load_ext autoreload
%autoreload 2
import pandas as pd
import gseapy as gp
import matplotlib.pyplot as plt
```

Check gseapy version

[2]:
```
gp.__version__
```

[2]:
```
'0.10.0'
```

3.5 1. (Optional) Convert IDs Using Biomart API

Don't use this if you don't know Biomart

```
>>> from gseapy.parser import Biomart
>>> bm = Biomart()
>>> ## view validated marts
>>> marts = bm.get_marts()
>>> ## view validated dataset
>>> datasets = bm.get_datasets(mart='ENSEMBL_MART_ENSEMBL')
>>> ## view validated attributes
>>> attrs = bm.get_attributes(dataset='hsapiens_gene_ensembl')
>>> ## view validated filters
>>> filters = bm.get_filters(dataset='hsapiens_gene_ensembl')
>>> ## query results
>>> queries = ['ENSG00000125285','ENSG00000182968'] # need to be a python list
>>> results = bm.query(dataset='hsapiens_gene_ensembl',
                       attributes=['ensembl_gene_id', 'external_gene_name',
↪'entrezgene_id', 'go_id'],
                       filters={'ensemble_gene_id': queries})
```

3.6 2. Enrichr Example

[3]:
```
# read in an example gene list
gene_list = pd.read_csv("./data/gene_list.txt",header=None, sep="\t")
gene_list.head()
```

[3]:
```
                0
0         IGKV4-1
1            CD55
2            IGKC
3         PPFIBP1
4           ABHD4
```

[4]:
```
# convert dataframe or series to list
glist = gene_list.squeeze().str.strip().tolist()
print(glist[:10])
```

```
['IGKV4-1', 'CD55', 'IGKC', 'PPFIBP1', 'ABHD4', 'PCSK6', 'PGD', 'ARHGDIB', 'ITGB2',
↪'CARD6']
```

See all supported enrichr library names

Select database from **{ 'Human', 'Mouse', 'Yeast', 'Fly', 'Fish', 'Worm' }**

Enrichr library could be used for `gsea`, `ssgsea`, and `prerank`, too

[5]:
```
names = gp.get_library_name() # default: Human
names[:10]
```

[5]:
```
['ARCHS4_Cell-lines',
 'ARCHS4_IDG_Coexp',
 'ARCHS4_Kinases_Coexp',
 'ARCHS4_TFs_Coexp',
 'ARCHS4_Tissues',
 'Achilles_fitness_decrease',
 'Achilles_fitness_increase',
 'Aging_Perturbations_from_GEO_down',
 'Aging_Perturbations_from_GEO_up',
 'Allen_Brain_Atlas_down']
```

[6]:
```
yeast = gp.get_library_name(database='Yeast')
yeast[:10]
```

[6]:
```
['Cellular_Component_AutoRIF',
 'Cellular_Component_AutoRIF_Predicted_zscore',
 'GO_Biological_Process_2018',
 'GO_Biological_Process_AutoRIF',
 'GO_Biological_Process_AutoRIF_Predicted_zscore',
 'GO_Cellular_Component_2018',
 'GO_Cellular_Component_AutoRIF',
 'GO_Cellular_Component_AutoRIF_Predicted_zscore',
 'GO_Molecular_Function_2018',
 'GO_Molecular_Function_AutoRIF']
```

3.6.1 2.1 Assign enrichr with `pd.Series`, `pd.DataFrame`, or `list` object

2.1.1 gene_sets support list, str.

Multi-libraries names supported, separate each name by comma or input a list.

For example:

```
# gene_list
gene_list="./data/gene_list.txt",
gene_list=glist
# gene_sets
gene_sets='KEGG_2016'
gene_sets='KEGG_2016,KEGG_2013'
gene_sets=['KEGG_2016','KEGG_2013']
```

[7]:
```
# run enrichr
# if you are only intrested in dataframe that enrichr returned, please set no_
↪plot=True

# list, dataframe, series inputs are supported
enr = gp.enrichr(gene_list="./data/gene_list.txt",
                 gene_sets=['KEGG_2016','KEGG_2013'],
                 organism='Human', # don't forget to set organism to the one you␣
↪desired! e.g. Yeast
                 description='test_name',
                 outdir='test/enrichr_kegg',
                 # no_plot=True,
                 cutoff=0.5 # test dataset, use lower value from range(0,1)
                )
```

[8]:
```
# obj.results stores all results
enr.results.head(5)
```

[8]:
```
    Gene_set                                             Term Overlap  \
0  KEGG_2016  Osteoclast differentiation Homo sapiens hsa04380  28/132
1  KEGG_2016                Tuberculosis Homo sapiens hsa05152  31/178
2  KEGG_2016                   Phagosome Homo sapiens hsa04145  28/154
3  KEGG_2016        Rheumatoid arthritis Homo sapiens hsa05323   19/90
4  KEGG_2016               Leishmaniasis Homo sapiens hsa05140   17/73

        P-value  Adjusted P-value  Old P-value  Old Adjusted P-value  \
0  3.104504e-13      9.096197e-11            0                     0
1  4.288559e-12      6.282739e-10            0                     0
2  1.614009e-11      1.576349e-09            0                     0
3  2.197884e-09      1.609950e-07            0                     0
4  3.132614e-09      1.835712e-07            0                     0

   Odds Ratio  Combined Score  \
0    5.303030      152.731262
1    4.353933      113.964491
2    4.545455      112.953250
3    5.277778      105.216567
4    5.821918      114.001290

                                              Genes
0  LILRA6;ITGB3;LILRA2;LILRA5;PPP3R1;FCGR3B;SIRPA...
1  RAB5B;ITGB2;PPP3R1;HLA-DMA;FCGR3B;HLA-DMB;CASP...
2  ATP6V1A;RAB5B;ITGB5;ITGB3;ITGB2;HLA-DMA;FCGR3B...
3  ATP6V1A;ATP6V1G1;ATP6V0B;TGFB1;ITGB2;FOS;ITGAL...
4  TGFB1;IFNGR1;PRKCB;IFNGR2;ITGB2;FOS;MAPK14;HLA...
```

2.1.2 Local mode of GO analysis

If input a .gmt file or gene_set dict object, enrichr runs local.
You have to specify the background genes, if local mode used

For example:

```
gene_sets="./data/genes.gmt",
gene_sets={'A':['gene1', 'gene2',...],
           'B':['gene2', 'gene4',...],
           ...}
```

[9]:
```
enr2 = gp.enrichr(gene_list="./data/gene_list.txt",
                  # or gene_list=glist
                  description='test_name',
                  gene_sets="./data/genes.gmt",
                  background='hsapiens_gene_ensembl', # or the number of genes, e.g␣
↪20000
                  outdir='test/enrichr_kegg2',
                  cutoff=0.5, # only used for testing.
                  verbose=True)
```

```
2020-08-11 12:34:31,139 User Defined gene sets is given: ./data/genes.gmt
2020-08-11 12:34:31,145 Connecting to Enrichr Server to get latest library names
2020-08-11 12:34:31,171 using all annotated genes with GO_ID as background genes
2020-08-11 12:34:31,176 Background: found 19041 genes
2020-08-11 12:34:31,181 Save file of enrichment results: Job Id:5030209168
2020-08-11 12:34:31,296 Done.
```

[10]:
```
enr2.results.head(5)
```

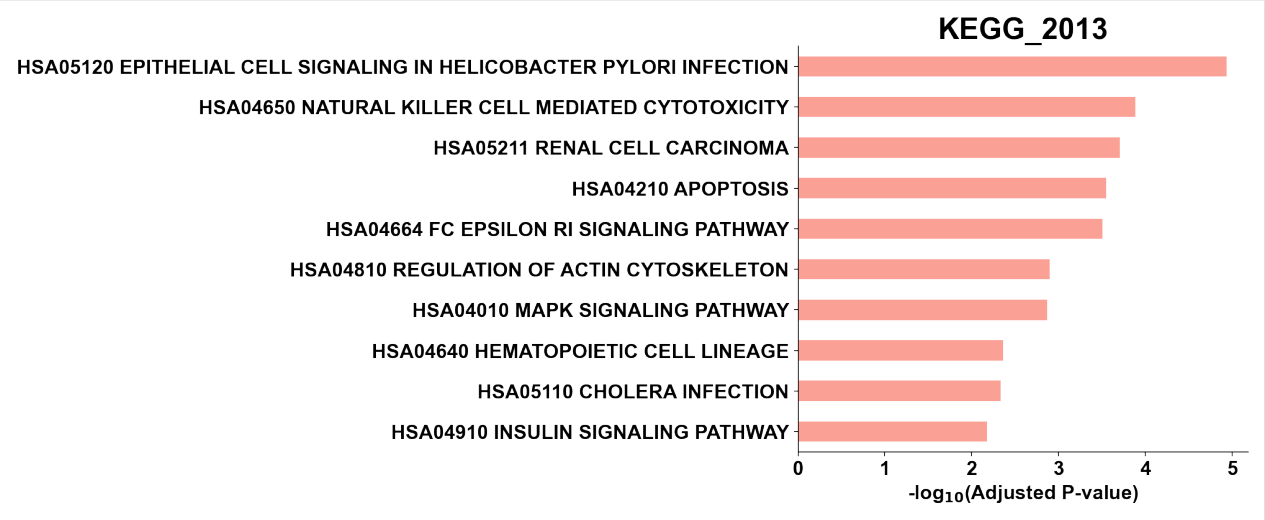
[10]:

	Gene_set	Term	Overlap	P-value	Adjusted P-value	Genes
0	CUSTOM5030209168	BvA_UpIN_A	8/139	0.287984	0.581624	MBOAT2;IL1R1;MAP3K5;PCSK6;IQGAP2;MSRB2;HAL;PADI2
1	CUSTOM5030209168	BvA_UpIN_B	11/130	0.032532	0.227727	SUOX;IL1RAP;GPX8;DYSF;ARHGDIB;KCTD12;LPAR1;SYK...
2	CUSTOM5030209168	CvA_UpIN_A	1/12	0.424279	0.581624	MBOAT2
3	CUSTOM5030209168	DvA_UpIN_A	16/284	0.210293	0.581624	MBOAT2;KIF1B;BCL3;IL1R1;PTGS1;NMNAT1;ATP6V1B2;...
4	CUSTOM5030209168	DvA_UpIN_D	12/236	0.372799	0.581624	IL1RAP;GLIPR2;GNB4;TXNDC5;DYSF;GPX8;SIRPA;LPAR...

2.1.3 Plotting

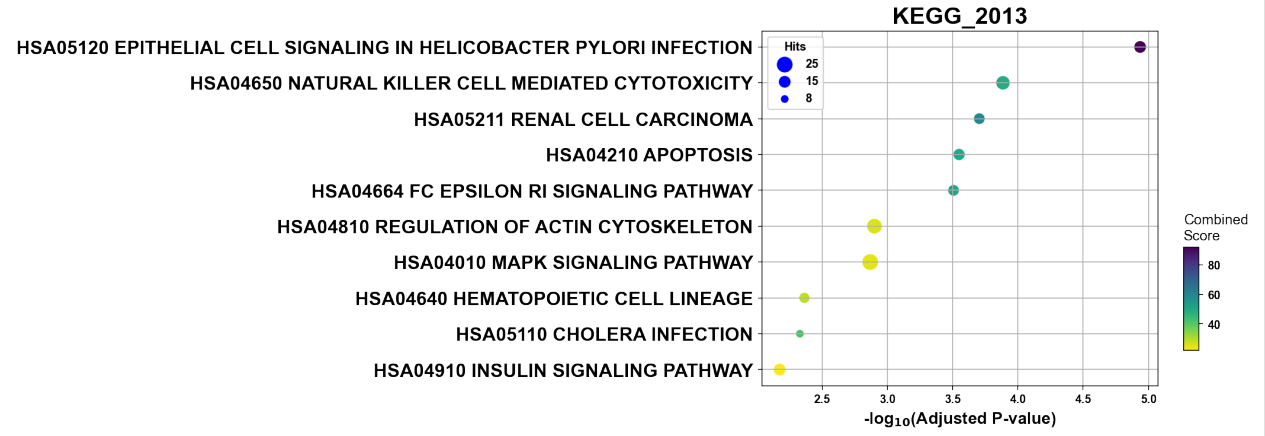
[11]:
```
# simple plotting function
from gseapy.plot import barplot, dotplot

# to save your figure, make sure that ``ofname`` is not None
barplot(enr.res2d,title='KEGG_2013',)
```



[12]:
```
# to save your figure, make sure that ``ofname`` is not None
dotplot(enr.res2d, title='KEGG_2013',cmap='viridis_r')
```

[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x12cb196d0>
```



3.6.2 2.2 Command line usage

You may also want to use enrichr in command line

the option **-v** will print out the progress of your job

[13]:
```
# !gseapy enrichr -i ./data/gene_list.txt \
#                 --ds BP2017 \
#                 -g GO_Biological_Process_2017 \
#                 -v -o test/enrichr_BP
```

3.7 3. Prerank example

3.7.1 3.1 Assign prerank() with a pd.DataFrame, pd.Series , or a txt file

Do not include header in your gene list !
GSEApy will skip any data after "#".
Only contains two columns, or one cloumn with gene_name indexed when assign a `DataFrame` to prerank

[14]:
```
rnk = pd.read_csv("./data/edb/gsea_data.gsea_data.rnk", header=None, sep="\t")
rnk.head()
```

[14]:
```
              0         1
0        CTLA2B  2.502482
1        SCARA3  2.095578
2  LOC100044683  1.116398
3          CMBL  0.877640
4         CLIC6  0.822181
```

[15]:
```
# run prerank
# enrichr libraries are supported by prerank module. Just provide the name
# use 4 process to acceralate the permutation speed

# note: multiprocessing may not work on windows
pre_res = gp.prerank(rnk=rnk, gene_sets='KEGG_2016',
                     processes=4,
                     permutation_num=100, # reduce number to speed up testing
                     outdir='test/prerank_report_kegg', format='png', seed=6)
```

Leading edge genes save to the final output results

[16]:
```
#access results through obj.res2d attribute or obj.results
pre_res.res2d.sort_index().head()
```

[16]:
```
                                                    es       nes  \
Term
Cytokine-cytokine receptor interaction Homo sap...  0.418234  1.526671
Focal adhesion Homo sapiens hsa04510                0.259225  0.859895
HTLV-I infection Homo sapiens hsa05166              0.338286  1.344137
MAPK signaling pathway Homo sapiens hsa04010        0.179667  0.686976
Metabolic pathways Homo sapiens hsa01100            0.194868  0.902211

                                                      pval       fdr  \
Term
Cytokine-cytokine receptor interaction Homo sap...  0.064935  0.591133
Focal adhesion Homo sapiens hsa04510                0.629032  0.993103
HTLV-I infection Homo sapiens hsa05166              0.137931  0.643678
MAPK signaling pathway Homo sapiens hsa04010        0.847222  0.811166
Metabolic pathways Homo sapiens hsa01100            0.617647  1.000000

                                                    geneset_size  \
Term
Cytokine-cytokine receptor interaction Homo sap...           265
Focal adhesion Homo sapiens hsa04510                         202
HTLV-I infection Homo sapiens hsa05166                       258
MAPK signaling pathway Homo sapiens hsa04010                 255
```

(continues on next page)

(continued from previous page)

```
Metabolic pathways Homo sapiens hsa01100                    1239

                                                    matched_size  \
Term
Cytokine-cytokine receptor interaction Homo sap...            18
Focal adhesion Homo sapiens hsa04510                          15
HTLV-I infection Homo sapiens hsa05166                        19
MAPK signaling pathway Homo sapiens hsa04010                  18
Metabolic pathways Homo sapiens hsa01100                      36


↪          genes  \
Term
Cytokine-cytokine receptor interaction Homo sap...  IL13RA1;CSF1;CCL2;TGFBR2;CD40;
↪IL10RB;CXCL10;CX...
Focal adhesion Homo sapiens hsa04510                COL6A1;PARVA;FLNC;THBS4;LAMB3;
↪PDGFRB;FLT4;ILK;...
HTLV-I infection Homo sapiens hsa05166              CRTC3;TGFBR2;CD40;PDGFRB;ADCY6;
↪PPP3CC;ETS1;WNT...
MAPK signaling pathway Homo sapiens hsa04010        CACNA1H;TGFBR2;FLNC;MAP3K5;PDGFRB;
↪PPP3CC;NFATC...
Metabolic pathways Homo sapiens hsa01100            CMBL;CDA;ST3GAL1;PLD2;CYP26A1;
↪ENO2;GALNT4;PYGL...


↪    ledge_genes
Term
Cytokine-cytokine receptor interaction Homo sap...        IL13RA1;CSF1;CCL2;TGFBR2;
↪CD40;IL10RB;CXCL10
Focal adhesion Homo sapiens hsa04510                           COL6A1;PARVA;FLNC;
↪THBS4;LAMB3;PDGFRB
HTLV-I infection Homo sapiens hsa05166              CRTC3;TGFBR2;CD40;PDGFRB;ADCY6;
↪PPP3CC;ETS1;WNT...
MAPK signaling pathway Homo sapiens hsa04010                CACNA1H;TGFBR2;FLNC;
↪MAP3K5;PDGFRB;PPP3CC
Metabolic pathways Homo sapiens hsa01100               CMBL;CDA;ST3GAL1;PLD2;CYP26A1;
↪ENO2;GALNT4;PYGL
```

3.7.2 3.2 How to generate your GSEA plot inside python console

Visualize it using `gseaplot`

Make sure that `ofname` is not None, if you want to save your figure to the disk

[17]:
```
# extract geneset terms in res2d
terms = pre_res.res2d.index
terms
```

[17]:
```
Index(['Pathways in cancer Homo sapiens hsa05200',
       'Cytokine-cytokine receptor interaction Homo sapiens hsa04060',
       'HTLV-I infection Homo sapiens hsa05166',
       'MAPK signaling pathway Homo sapiens hsa04010',
       'Rap1 signaling pathway Homo sapiens hsa04015',
       'PI3K-Akt signaling pathway Homo sapiens hsa04151',
       'Focal adhesion Homo sapiens hsa04510',
       'Ras signaling pathway Homo sapiens hsa04014',
```

(continues on next page)

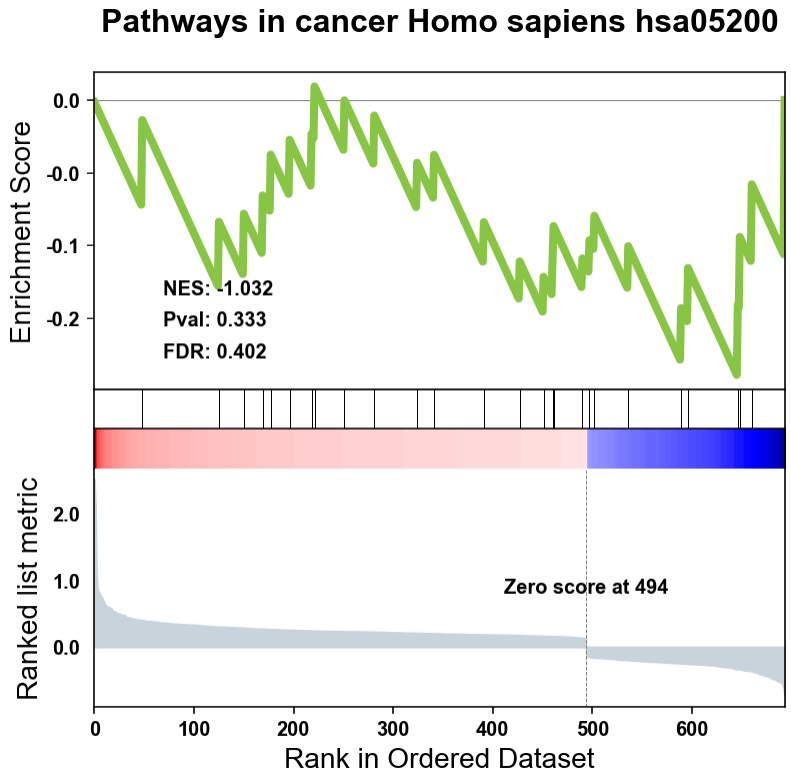
(continued from previous page)

```
       'Metabolic pathways Homo sapiens hsa01100'],
      dtype='object', name='Term')
```

[18]:
```
## easy way
from gseapy.plot import gseaplot

# to save your figure, make sure that ofname is not None
gseaplot(rank_metric=pre_res.ranking, term=terms[0], **pre_res.results[terms[0]])

# save figure
# gseaplot(rank_metric=pre_res.ranking, term=terms[0], ofname='your.plot.pdf', **pre_
↪res.results[terms[0]])
```



3.7.3 3) Command line usage

You may also want to use prerank in command line

[19]:
```
# ! gseapy prerank -r temp.rnk -g temp.gmt -o prerank_report_temp
```

3.8 4. GSEA Example

3.8.1 4.1 Assign gsea() with a pandas DataFrame, .gct format file, or a text file

and cls with a list object or just .cls format file

[20]:
```
phenoA, phenoB, class_vector =  gp.parser.gsea_cls_parser("./data/P53.cls")
```

[21]:
```
#class_vector used to indicate group attributes for each sample
print(class_vector)
```

```
['MUT', 'MUT', 'MUT', 'MUT', 'MUT', 'MUT', 'MUT', 'MUT', 'MUT', 'MUT', 'MUT', 'MUT',
↪'MUT', 'MUT', 'MUT', 'MUT', 'MUT', 'MUT', 'MUT', 'MUT', 'MUT', 'MUT', 'MUT', 'MUT',
↪'MUT', 'MUT', 'MUT', 'MUT', 'MUT', 'MUT', 'MUT', 'MUT', 'MUT', 'WT', 'WT', 'WT', 'WT
↪', 'WT', 'WT', 'WT', 'WT', 'WT', 'WT', 'WT', 'WT', 'WT', 'WT', 'WT', 'WT', 'WT']
```

[22]:
```
gene_exp = pd.read_csv("./data/P53.txt", sep="\t")
gene_exp.head()
```

[22]:

	NAME	DESCRIPTION	786-0	BT-549	CCRF-CEM	COLO 205	EKVX	\
0	TACC2	na	46.05	82.17	16.87	98.60	141.02	
1	C14orf132	na	108.34	59.04	25.61	33.11	42.53	
2	AGER	na	42.20	25.75	76.01	40.41	32.17	
3	32385_at	na	7.43	13.94	8.55	21.13	15.09	
4	RBM17	na	11.40	3.00	3.16	2.34	4.43	

	HCC-2998	HCT-15	HOP-62	...	MCF7	MOLT-4	NCI-H460	OVCAR-4	SF-539	\
0	114.32	134.34	44.95	...	68.14	32.21	105.89	64.99	53.52	
1	9.12	9.36	310.96	...	159.32	10.71	13.59	53.78	57.57	
2	48.28	58.27	42.40	...	51.50	61.48	44.44	45.68	54.17	
3	19.05	16.47	7.60	...	30.77	21.27	13.36	16.19	12.07	
4	1.56	6.04	6.16	...	1.62	2.77	4.42	8.91	12.28	

	SK-MEL-5	SR	UACC-257	UACC-62	UO-31
0	85.47	18.69	32.16	45.70	48.13
1	86.80	17.30	102.66	62.16	73.44
2	62.53	83.18	56.57	50.40	36.75
3	17.62	22.60	4.50	14.59	11.33
4	3.04	10.13	8.32	8.23	3.91

[5 rows x 52 columns]

[23]:
```
print("positively correlated: ", phenoA)
```

```
positively correlated:  MUT
```

[24]:
```
print("negtively correlated: ", phenoB)
```

```
negtively correlated:  WT
```

[25]:
```
# run gsea
# enrichr libraries are supported by gsea module. Just provide the name

gs_res = gp.gsea(data=gene_exp, # or data='./P53_resampling_data.txt'
                 gene_sets='KEGG_2016', # enrichr library names
                 cls= './data/P53.cls', # cls=class_vector
                 # set permutation_type to phenotype if samples >=15
                 permutation_type='phenotype',
                 permutation_num=100, # reduce number to speed up test
                 outdir=None,  # do not write output to disk
                 no_plot=True, # Skip plotting
                 method='signal_to_noise',
                 processes=4, seed= 7,
                 format='png')
```

```
/Users/zqfang/Miniconda/lib/python3.7/site-packages/joblib/externals/loky/process_
↪executor.py:691: UserWarning: A worker stopped while some jobs were given to the␣
↪executor. This can be caused by a too short worker timeout or by a memory leak.
  "timeout or by a memory leak.", UserWarning
```

[26]:
```
#access the dataframe results throught res2d attribute
gs_res.res2d.sort_index().head()
```

[26]:
```
                                                    es       nes  \
Term
ABC transporters Homo sapiens hsa02010          -0.328540 -1.174290
AGE-RAGE signaling pathway in diabetic complica... -0.261859 -1.046512
AMPK signaling pathway Homo sapiens hsa04152      0.198892  0.899468
Acute myeloid leukemia Homo sapiens hsa05221      0.196716  0.710123
Adherens junction Homo sapiens hsa04520           0.246805  0.984227

                                                      pval       fdr  \
Term
ABC transporters Homo sapiens hsa02010            0.252174  0.594993
AGE-RAGE signaling pathway in diabetic complica...  0.386555  0.739592
AMPK signaling pathway Homo sapiens hsa04152      0.714286  1.000000
Acute myeloid leukemia Homo sapiens hsa05221      0.900000  1.000000
Adherens junction Homo sapiens hsa04520           0.477941  1.000000

                                                  geneset_size  \
Term
ABC transporters Homo sapiens hsa02010                      44
AGE-RAGE signaling pathway in diabetic complica...         101
AMPK signaling pathway Homo sapiens hsa04152               124
Acute myeloid leukemia Homo sapiens hsa05221                57
Adherens junction Homo sapiens hsa04520                     74

                                                  matched_size  \
Term
ABC transporters Homo sapiens hsa02010                      33
AGE-RAGE signaling pathway in diabetic complica...          90
AMPK signaling pathway Homo sapiens hsa04152                89
Acute myeloid leukemia Homo sapiens hsa05221                50
Adherens junction Homo sapiens hsa04520                     66

                                                                                 ␣
↪          genes  \
Term
ABC transporters Homo sapiens hsa02010            ABCD3;ABCD4;ABCA2;ABCB7;ABCD1;
↪ABCA3;ABCC9;ABCB...
AGE-RAGE signaling pathway in diabetic complica...  F3;PIK3CA;PLCE1;NRAS;RELA;RAC1;
↪PLCB3;MAPK13;MA...
AMPK signaling pathway Homo sapiens hsa04152      PPP2R5B;PIK3CA;PPP2R5C;CREB3L1;
↪PRKAA1;CREB3;PP...
Acute myeloid leukemia Homo sapiens hsa05221      MAP2K1;PIK3CA;NRAS;RELA;RPS6KB2;
↪TCF7L2;JUP;RAF...
Adherens junction Homo sapiens hsa04520           EP300;YES1;CTNND1;RAC1;WASF1;
↪ERBB2;ACTN1;PTPRF...

                                                                                 ␣
↪    ledge_genes
Term
```

(continues on next page)

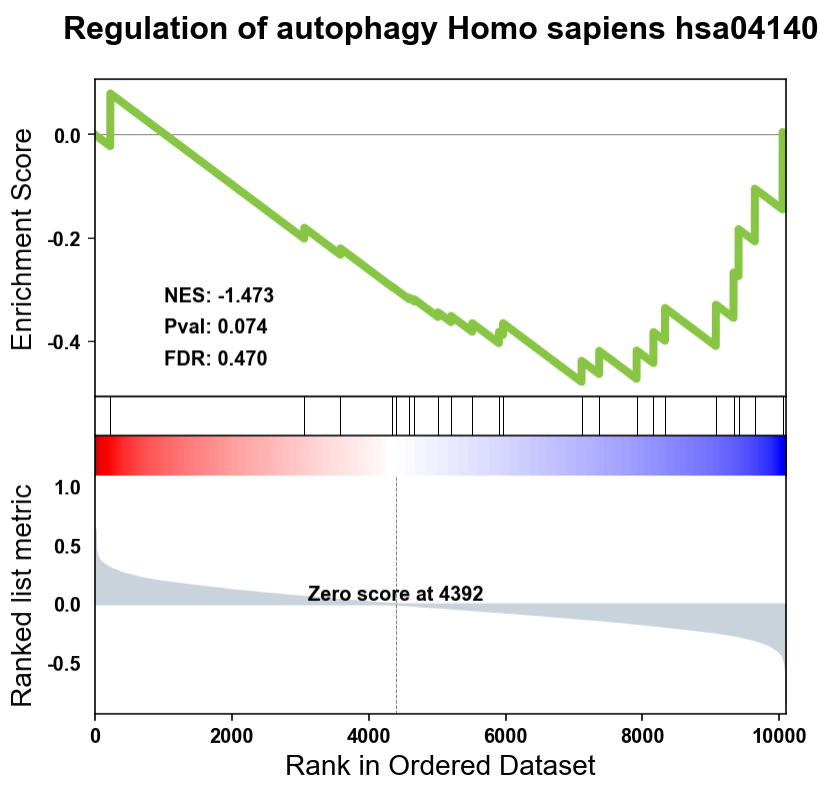
(continued from previous page)

```
ABC transporters Homo sapiens hsa02010                ABCG1;ABCC5;ABCB4;TAP2;CFTR;
 ↪ABCC10;ABCB11;ABCC...
AGE-RAGE signaling pathway in diabetic complica...    COL4A1;AKT3;VCAM1;PIK3R3;SMAD3;
 ↪STAT1;THBD;SELE...
AMPK signaling pathway Homo sapiens hsa04152          PPP2R5B;PIK3CA;PPP2R5C;CREB3L1;
 ↪PRKAA1;CREB3;PP...
Acute myeloid leukemia Homo sapiens hsa05221          MAP2K1;PIK3CA;NRAS;RELA;RPS6KB2;
 ↪TCF7L2;JUP;RAF...
Adherens junction Homo sapiens hsa04520               EP300;YES1;CTNND1;RAC1;WASF1;
 ↪ERBB2;ACTN1;PTPRF...
```

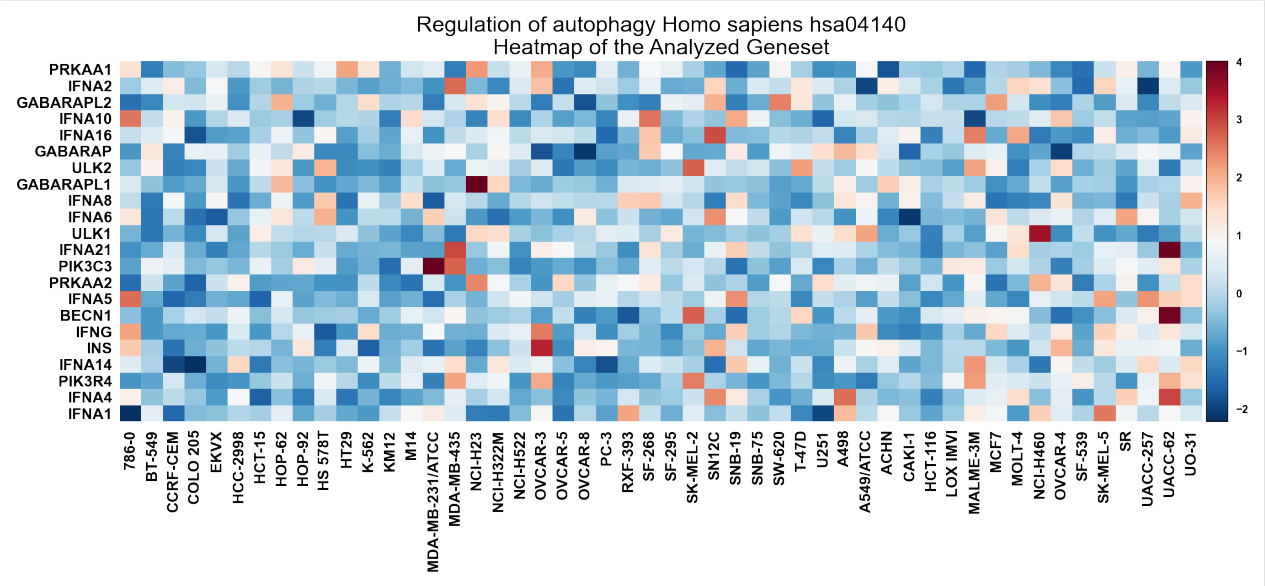
3.8.2 4.2 Show the gsea plots

The **gsea** module will generate heatmap for genes in each gene sets in the backgroud.
But if you need to do it yourself, use the code below

[27]:
```
from gseapy.plot import gseaplot, heatmap
terms = gs_res.res2d.index
# Make sure that ``ofname`` is not None, if you want to save your figure to disk
gseaplot(gs_res.ranking, term=terms[0], **gs_res.results[terms[0]])
```



[28]:
```
# plotting heatmap
genes = gs_res.res2d.genes[0].split(";")
# Make sure that ``ofname`` is not None, if you want to save your figure to disk
heatmap(df = gs_res.heatmat.loc[genes], z_score=0, title=terms[0], figsize=(18,6))
```



3.8.3 4.3 Command line usage

You may also want to use gsea in command line

[30]:
```
# !gseapy gsea -d ./data/P53_resampling_data.txt \
#              -g KEGG_2016 -c ./data/P53.cls \
#              -o test/gsea_reprot_2 \
#              -v --no-plot \
#              -t phenotype
```

3.9 5. Single Sample GSEA example

Note: When you run ssGSEA, all genes names in your gene_sets file should be found in your expression table

What's ssGSEA? Which one should I use? Prerank or ssGSEA

see FAQ here

3.9.1 5.1 Input format

Assign ssgsea() with a txt file, gct file, pd.DataFrame, or pd.Seires(gene name as index)

[29]:
```
# txt, gct file input
ss = gp.ssgsea(data="./data/testSet_rand1200.gct",
               gene_sets="./data/randomSets.gmt",
               outdir='test/ssgsea_report',
               sample_norm_method='rank', # choose 'custom' for your own rank list
               permutation_num=0, # skip permutation procedure, because you don't␣
↪need it
               no_plot=True, # skip plotting, because you don't need these figures
               processes=4, format='png', seed=9)
```

[30]:
```
ss.res2d.sort_index().head()
```

[30]:
```
                AA488_A1.2  AA489_A2.2  AA490_A3  AA491_B1  AA492_B2  \
Term|NES
level10_RAND      0.409675    0.412178  0.402321  0.420735  0.423723
level12_random    0.623442    0.637662  0.624727  0.636428  0.644896
level2_rand      -0.271914   -0.281744 -0.271358 -0.277807 -0.279207
level4_rand      -0.061550   -0.090115 -0.063876 -0.094134 -0.094397
level6_rand      -0.061075   -0.054566 -0.061283 -0.052871 -0.054915

                AA493_B3  AA494_C1.2  AA495_C2  AA496_C3  AA497_D1.2  \
Term|NES
level10_RAND    0.399593    0.417449  0.426562  0.414945    0.427697
level12_random  0.628848    0.645327  0.633330  0.637862    0.647586
level2_rand    -0.260003   -0.294843 -0.289941 -0.286516   -0.294068
level4_rand    -0.053017   -0.096070 -0.081046 -0.080501   -0.092273
level6_rand    -0.047231   -0.059001 -0.069959 -0.061611   -0.055635

                AA498_D3.2  AA499_D2  AA500_x2  AA501_X3  AA502_X2.2  \
Term|NES
level10_RAND      0.428861  0.439795  0.439169  0.429886    0.421071
level12_random    0.648293  0.638572  0.652058  0.650088    0.653153
level2_rand      -0.285716 -0.286194 -0.335032 -0.331248   -0.337686
level4_rand      -0.092036 -0.078247 -0.119548 -0.120003   -0.129959
level6_rand      -0.058775 -0.062907 -0.020677 -0.017325   -0.011440

                AA503_Y1  AA504_Y2  AA505_Y3
Term|NES
level10_RAND    0.433796  0.436511  0.427593
level12_random  0.660680  0.659886  0.648956
level2_rand    -0.336926 -0.334273 -0.339320
level4_rand    -0.143212 -0.138202 -0.126784
level6_rand    -0.018986 -0.017288 -0.009031
```

[31]:
```
# or assign a dataframe, or Series to ssgsea()
ssdf = pd.read_csv("./data/temp.txt", header=None, sep="\t")
ssdf.head()
```

[31]:
```
        0          1
0    ATXN1  16.456753
1  UBQLN4  13.989493
2    CALM1  13.745533
3     DLG4  12.796588
4  MRE11A  12.787631
```

[32]:
```
# dataframe with one column is also supported by ssGSEA or Prerank
# But you have to set gene_names as index
ssdf2 = ssdf.set_index(0)
ssdf2.head()
```

[32]:
```
                1
0
ATXN1   16.456753
UBQLN4  13.989493
CALM1   13.745533
DLG4    12.796588
MRE11A  12.787631
```

[33]:
```
type(ssdf2)
```

[33]:
```
pandas.core.frame.DataFrame
```

[34]:
```
ssSeries = ssdf2.squeeze()
type(ssSeries)
```

[34]:
```
pandas.core.series.Series
```

[35]:
```
# reuse data
df = pd.read_csv("./data/P53_resampling_data.txt", sep="\t")
df.head()
```

[35]:

	NAME	786-0	BT-549	CCRF-CEM	COLO 205	EKVX	HCC-2998	HCT-15	\
0	CTLA2B	111.19	86.22	121.85	75.19	208.62	130.59	124.72	
1	SCARA3	460.30	558.34	183.55	37.29	158.00	43.61	80.83	
2	LOC100044683	97.25	118.94	81.17	119.51	119.88	107.73	165.57	
3	CMBL	33.45	55.10	221.67	50.30	35.12	75.70	84.01	
4	CLIC6	35.75	41.26	63.04	219.86	42.53	54.19	86.98	

	HOP-62	HOP-92	...	MCF7	MOLT-4	NCI-H460	OVCAR-4	SF-539	SK-MEL-5	\
0	324.09	242.71	...	163.76	59.50	134.12	152.09	197.46	137.79	
1	300.08	1250.25	...	109.91	120.42	73.06	115.03	95.12	37.56	
2	203.97	135.43	...	222.84	124.98	114.75	141.66	170.19	147.70	
3	44.12	79.96	...	51.32	117.11	59.46	78.46	45.55	49.07	
4	71.20	53.89	...	154.05	31.62	37.66	32.64	63.35	27.95	

	SR	UACC-257	UACC-62	UO-31
0	81.53	123.37	81.41	180.78
1	76.16	41.10	77.51	519.17
2	157.48	152.18	98.89	118.06
3	96.69	33.09	10.38	52.89
4	70.99	36.25	17.50	49.41

[5 rows x 51 columns]

[36]:
```
# Series, DataFrame Example
# supports dataframe and series
ssgs = []
for i, dat in enumerate([ssdf, ssdf2, ssSeries, df]):
    sstemp = gp.ssgsea(data=dat,
                       gene_sets="./data/genes.gmt",
                       outdir='test/ssgsea_report_'+str(i),
                       scale=False, # set scale to False to get real original ES
                       permutation_num=0, # skip permutation procedure, because you␣
↪don't need it
                       no_plot=True, # skip plotting, because you don't need these␣
↪figures
                       processes=4, seed=10,
                       format='png')
    ssgs.append(sstemp)
```

```
2020-08-11 12:38:38,029 Warning: dropping duplicated gene names, only keep the first␣
↪values
```

3.9.2 5.2 Access Enrichment Score (ES) and NES

results save to two attribute:

1. obj.resultsOnSamples: ES
2. obj.res2d: NES

[37]:
```
# normalized es save to res2d attri
# one sample input
# NES
ssgs[0].res2d.sort_index().head()
```

[37]:
```
                   1
Term|NES
BvA_UpIN_A  2.150114
BvA_UpIN_B  2.953848
DvA_UpIN_A  1.985451
DvA_UpIN_D  2.457489
YvX_UpIN_X  2.148816
```

Note: If you want to obtain the real original enrichment score,
you have to set `scale=False`

[38]:
```
# ES
# convert dict to DataFrame
es = pd.DataFrame(ssgs[-1].resultsOnSamples)
es.sort_index().head()
```

[38]:
```
                786-0     BT-549   CCRF-CEM   COLO 205       EKVX   HCC-2998  \
Term|ES
DvA_UpIN_A  45.703475   6.724266  11.881146  20.639710  36.753558   3.530987
DvA_UpIN_D  82.960021  86.151980  88.176462  65.077923  80.856467  63.085467

                HCT-15     HOP-62     HOP-92    HS 578T  ...       MCF7  \
Term|ES                                                   ...
DvA_UpIN_A   5.257504  33.003838  29.227462  41.404387  ...   4.304996
DvA_UpIN_D  53.584047  73.531016  85.803567  87.688120  ...  72.525357

                MOLT-4   NCI-H460    OVCAR-4     SF-539   SK-MEL-5         SR  \
Term|ES
DvA_UpIN_A  17.789549  19.172561  37.144472  40.135942  18.082717  13.901976
DvA_UpIN_D  85.020685  76.225849  90.948093  97.684104  62.334470  68.252995

              UACC-257    UACC-62      UO-31
Term|ES
DvA_UpIN_A  44.562272  52.021549  51.156682
DvA_UpIN_D  73.484066  68.122566  86.657296

[2 rows x 50 columns]
```

[39]:
```
# if set scale to True, then
# Scaled ES equal to es/gene_numbers
ses = es/df.shape[0]
ses
```

[39]:
```
                786-0     BT-549   CCRF-CEM   COLO 205       EKVX   HCC-2998  \
Term|ES
DvA_UpIN_A   0.065855   0.009689   0.017120   0.029740   0.052959   0.005088
DvA_UpIN_D   0.119539   0.124138   0.127055   0.093772   0.116508   0.090901

               HCT-15     HOP-62     HOP-92    HS 578T  ...       MCF7    MOLT-4  \
Term|ES                                                 ...
DvA_UpIN_A   0.007576   0.047556   0.042114   0.059660  ...   0.006203   0.025633
DvA_UpIN_D   0.077210   0.105952   0.123636   0.126352  ...   0.104503   0.122508

             NCI-H460    OVCAR-4     SF-539   SK-MEL-5         SR   UACC-257  \
Term|ES
DvA_UpIN_A   0.027626   0.053522   0.057833   0.026056   0.020032   0.064211
DvA_UpIN_D   0.109836   0.131049   0.140755   0.089819   0.098347   0.105885

              UACC-62      UO-31
Term|ES
DvA_UpIN_A   0.074959   0.073713
DvA_UpIN_D   0.098159   0.124866

[2 rows x 50 columns]
```

[40]:
```
# NES
# scale or no have no affects on final nes value
nes = ssgs[-1].res2d
nes.sort_index().head()
```

[40]:
```
                786-0     BT-549   CCRF-CEM   COLO 205       EKVX   HCC-2998  \
Term|NES
DvA_UpIN_A   0.402250   0.059182   0.104570   0.181656   0.323479   0.031077
DvA_UpIN_D   0.730157   0.758250   0.776068   0.572771   0.711643   0.555235

               HCT-15     HOP-62     HOP-92    HS 578T  ...       MCF7    MOLT-4  \
Term|NES                                                ...
DvA_UpIN_A   0.046273   0.290477   0.257240   0.364413  ...   0.037890   0.156571
DvA_UpIN_D   0.471610   0.647169   0.755184   0.771770  ...   0.638318   0.748293

             NCI-H460    OVCAR-4     SF-539   SK-MEL-5         SR   UACC-257  \
Term|NES
DvA_UpIN_A   0.168744   0.326920   0.353249   0.159152   0.122356   0.392206
DvA_UpIN_D   0.670887   0.800462   0.859748   0.548625   0.600716   0.646756

              UACC-62      UO-31
Term|NES
DvA_UpIN_A   0.457858   0.450246
DvA_UpIN_D   0.599568   0.762698

[2 rows x 50 columns]
```

3.9.3 3) command line usage of single sample gsea

[41]:
```
# set --no-scale to obtain the real original enrichment score
# !gseapy ssgsea -d ./data/testSet_rand1200.gct \
#                -g data/temp.gmt \
#                -o test/ssgsea_report2  \
#                -p 4 --no-plot --no-scale
```

3.10 6. Replot Example

3.10.1 6.1 locate your directory

notes: `replot` module need to find edb folder to work properly. keep the file tree like this:

```
data
 |--- edb
 |     |--- C1OE.cls
 |     |--- gene_sets.gmt
 |     |--- gsea_data.gsea_data.rnk
 |     |--- results.edb
```

[42]:
```
# run command inside python console
rep = gp.replot(indir="./data", outdir="test/replot_test")
```

3.10.2 6.2 command line usage of replot

[43]:
```
# !gseapy replot -i data -o test/replot_test
```

[ ]:
```

```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

g

Index

S

Z