spycone

Release 0

Chit Tong Lio

Nov 01, 2022

CONTENTS

Spycone is a python package that provides systematic analysis of time course transcriptomics data. It uses gene or isoform expression and a biological network as an input. It employs the sum of changes of all isoforms relative abundances (total isoform usage) across time points to detect IS events. Spycone further provides downstream analysis such as clustering by total isoform usage, gene set enrichment analysis, network enrichment, and splicing factors analysis. Below we describe the Spycone workflow in detail. .

CHAPTER

ONE

INSTALLATION

Prerequisite

Spycone is dependent on the pcst_fast library, which is not available through pip install. Please go to the github page or run this command.:

```
pip install https://github.com/fraenkel-lab/pcst_fast/archive/refs/tags/1.0.7.tar.gz
```

To install Spycone:

```
pip install spycone
```

To install the latest development:

```
git clone https://github.com/yollct/spycone.git
cd spycone
pip install .
```

1.1 Contents

[2]:
```
import sys
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import subprocess

import spycone as spy
```

1.1.1 Gene-level workflow

Prepare the dataset

We use a time series dataset of influenza infection with 9 time points.

[ ]:
```
#sample data
subprocess.call("wget https://zenodo.org/record/7228475/files/normticonerhino_wide.csv?
↪download=1 -O normticonerhino_wide.csv", shell=True)
```

(continues on next page)

(continued from previous page)

```
influ = pd.read_csv("normticonerhino_wide.csv", dtype={'entrezid': str})
gene_list = influ['entrezid']
symbs = influ['symbol']

flu_ts = influ.iloc[:,3:] ##filter out the entrez id and gene id column
```

Import expression data with `DataSet` which stores the count matrix, list of gene ID, number of time points, and number of replicates.

`ts` : time series data values with columns as each sample e.g. the order of the columns should be **sample1_rep1**, **sample1_rep2**, **sample2_rep1**, **sample2_rep2** and so on. . . .

`gene_id` : the pandas series or list of gene id (can be entrez gene id or ensembl gene id)

`species` : specify the species ID

`reps1` : Number of replicates

`timepts` : Number of time points

`discreization_steps` : Steps to discretize the data values

[4]:
```
flu_dset = spy.dataset(ts=flu_ts,
                       gene_id = gene_list,
                       symbs=gene_list,
                       species=9606,
                       keytype="entrezgeneid",
                       reps1 = 5,
                       timepts = 9)
```

Import biological network of your choice with `BioNetwork`, Spycone provides Biogrid, IID network in entrez ID as node name. Please specify the keytype if you are using a different ID.

[5]:
```
bionet = spy.BioNetwork("human", data=(('weight',float),))
```

Preprocessing

Filtering out genes that has expression across all time points lower than 1. By giving the biological network, it removes genes from the dataset that are not in the network.

[6]:
```
spy.preprocess(flu_dset)
```

```
Input data dimension: (5, 19463, 9)
Removed 0 with 0 values.
Filtered data: (5, 19463, 9)
```

Clustering

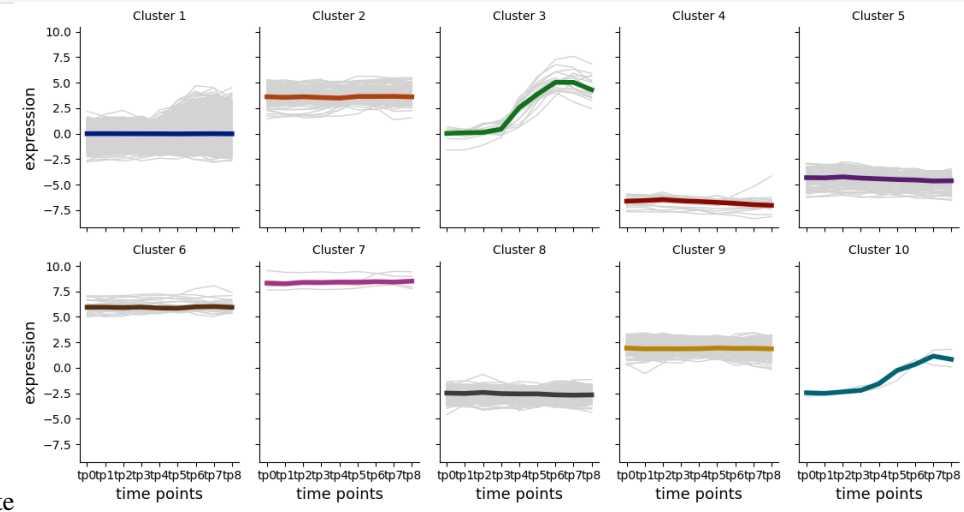
`clustering` create clustering object that provides varies algorithms and result storage.

[7]:
```
asclu = spy.clustering(flu_dset, algorithm='hierarchical', metrics="correlation", input_
↪type="expression", n_clusters=10, composite=False)
c = asclu.find_clusters()
```

```
clustering took 31.098841428756714s.
```

visualizing clustering

[8]:
```
%matplotlib inline
spy.vis_all_clusters(asclu, col_wrap=5)
```



nbsphinx-code-borderwhite

Gene set enrichment analysis

Perform gene set enrichment analysis using `clusters_gsea`. Change the gene_sets parameter into the choice of your knowledge base or gene set database, e.g. Reactome, KEGG, etc. Use `spy.list_genesets` to view the available knowledge base.

[9]:
```
sys.path.insert(0, "../../../")
from spycone_pkg.spycone.go_terms import clusters_gsea

asclu_go, _ = clusters_gsea(flu_dset, "hsapiens", method="gsea")
```

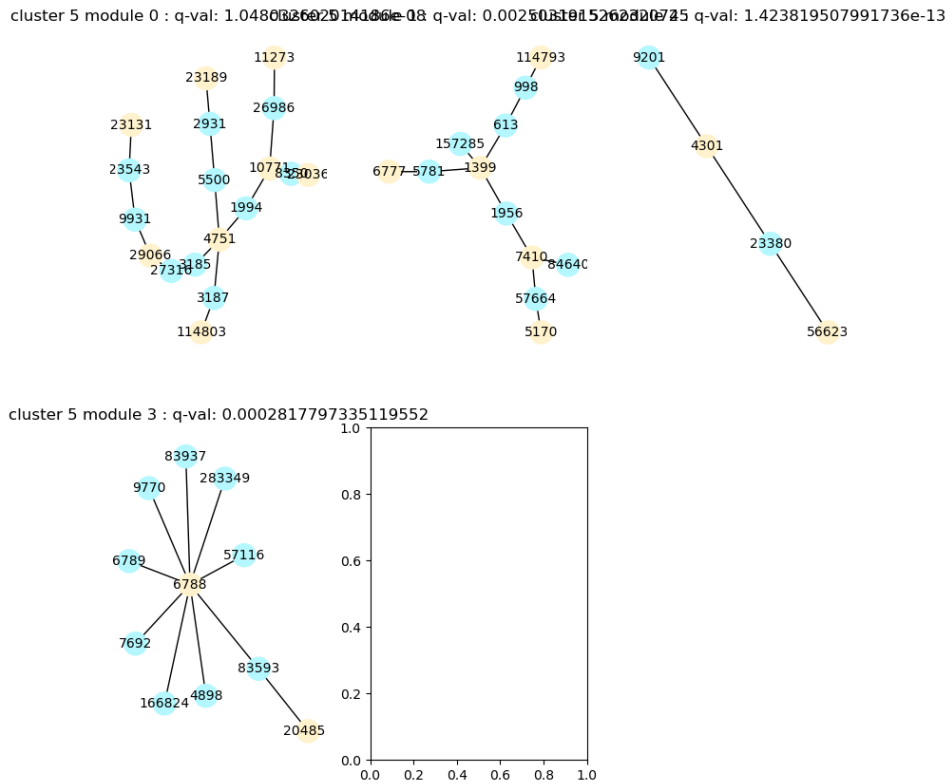
[10]:
```
from spycone_pkg.spycone.visualize import gsea_plot
gsea_plot(asclu_go, cluster=5, nterms=15)
```

Run DOMINO

[11]:
```
mods = spy.run_domino(asclu, network_file=bionet, output_file_path="newslices.txt")
```

To visualize the modules, use `vis_modules`.

[12]:
```
%matplotlib inline
spy.vis_modules(mods, flu_dset, cluster=5, size=0)
```



nbsphinx-code-borderwhite

It is also possible to visualize modules with javascript, use `vis_better_modules` and input a desired directory, the function will generate networks with dot format (Graphviz) (https://github.com/pydot/pydot).

vis_better_modules(flu_dset, mods, cluster=5, dir='/path/to/file')

[23]:
```
import sys
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
sys.path.insert(0, "../../")
import spycone as spy
import subprocess
from gtfparse import read_gtf
```

1.1.2 Transcript-level workflow

Prepare the dataset

The dataset is SARS-Cov-2 infection with 8 time points and 4 replicates.

[ ]:
```
#sample data
subprocess.call("wget https://zenodo.org/record/7228475/files/tutorial_alt_sorted_bc_tpm.
↪csv?download=1 -O alt_sorted_bc_tpm.csv", shell=True)
subprocess.call("wget https://zenodo.org/record/7228475/files/tutorial_alt_genelist.csv?
↪download=1 -O alt_genelist.csv", shell=True)

data = pd.read_csv("alt_sorted_bc_tpm.csv", sep="\t")
genelist = pd.read_csv("alt_genelist.csv", sep="\t")

geneid= list(map(lambda x: str(int(x)) if not np.isnan(x) else x,  genelist['gene'].
↪tolist()))
transcriptid = genelist['isoforms'].to_list()
```

Read the data to the dataset function.

[25]:
```
dset = spy.dataset(ts=data,
        transcript_id=transcriptid,
        gene_id = geneid,
        species=9606,
        keytype='entrezgeneid',
        timepts=4, reps1=3)
```

Import biological network of your choice with `BioNetwork`, Spycone provides Biogrid, IID network in entrez ID as node name. Please specify the keytype if you are using a different ID.

[26]:
```
bionet = spy.BioNetwork(path="human", data=(('weight',float),))
```

Preprocessing

Filtering out genes that has expression across all time points lower than 1. By giving the biological network, it removes genes from the dataset that are not in the network.

[27]:
```
spy.preprocess(dset, bionet, cutoff=1)
```

Detect isoform switch

iso object contains method for isoform switch detect and total isoform usage calculation.

[28]:
```
iso = spy.iso_function(dset)
        #run isoform switch
ascov=iso.detect_isoform_switch(filtering=False, min_diff=0.05, corr_cutoff=0.5, event_
↪im_cutoff=0.1, p_val_cutoff=0.05)
```

`iso.detect_isoform_switch` returns a dataframe containing the metrics of all isoform switch events detected.
`iso_ratio` : ratio of switching time to number of time points

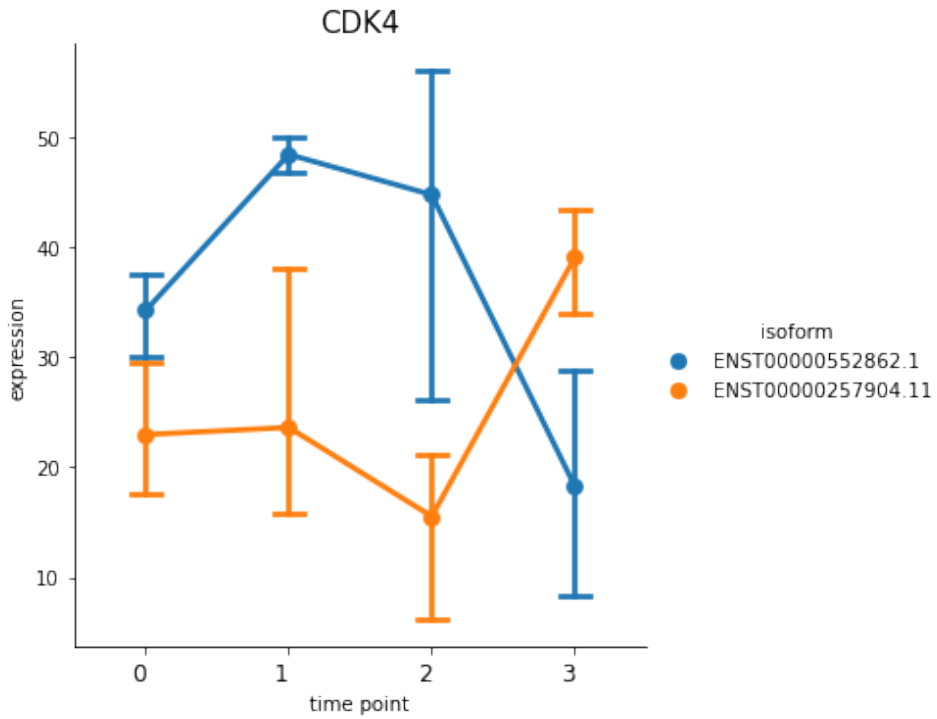
`diff` : difference of relative abundances before and after switch
`p_value` : combined p-value of maj_pval and min_pval (corresponding p-value from t-test of the transcripts)
`corr` : dissimilar correlation (inverted pearson correlation)
`final_sp` : best switch point
`event_importance` : the ratio of the relative abundance of the events to the highest expressed transcripts
`exclusive_domains` : domain loss/gain of the event
`adj_pval` : p-values after multiple test correction

[29]:
```
ascov.head()
```

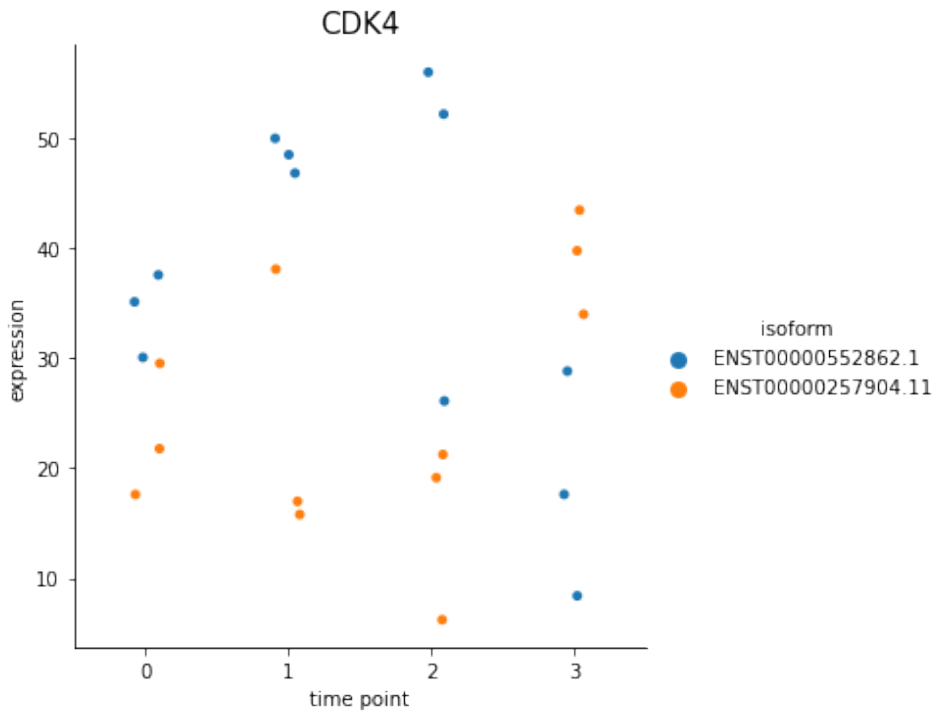
[29]:
```
    gene gene_symb   major_transcript    minor_transcript  switch_prob  \
0   1019      CDK4  ENST00000552862.1  ENST00000257904.11          1.0
1  54882    ANKHD1  ENST00000433049.5   ENST00000394722.7          1.0
2   9140     ATG12  ENST00000379594.7   ENST00000509910.2          1.0
3  64419    MTMR14  ENST00000353332.9   ENST00000431250.1          1.0
4  51184      GPN3  ENST00000228827.8   ENST00000537466.6          1.0

       corr      diff  event_importance   exclusive_domains   p_value  \
0  0.822139  0.066160          0.343667                  []  0.002980
1  0.706529  0.073003          0.891058  [PF12796, PF00013]  0.056052
2  0.613932  0.050557          0.537392           [PF04110]  0.028167
3  0.663200  0.062577          0.564498                  []  0.001605
4  0.913216  0.089762          0.189560                  []  0.001605

   adj_pval
0  0.002838
1  0.039891
2  0.022642
3  0.001549
4  0.001549
```

You can also plot out the genes using `switch_plot`, by provide your dataset object and the result dataframe from detect_isoform_switch.

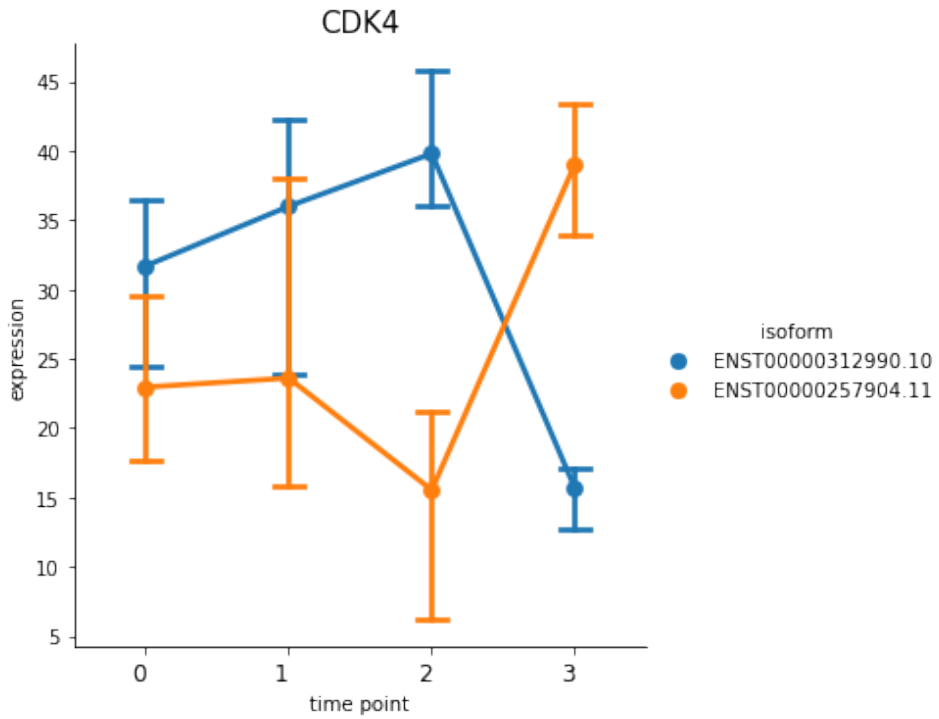
[30]:
```
%matplotlib inline
spy.switch_plot("CDK4", dset, ascov)
```



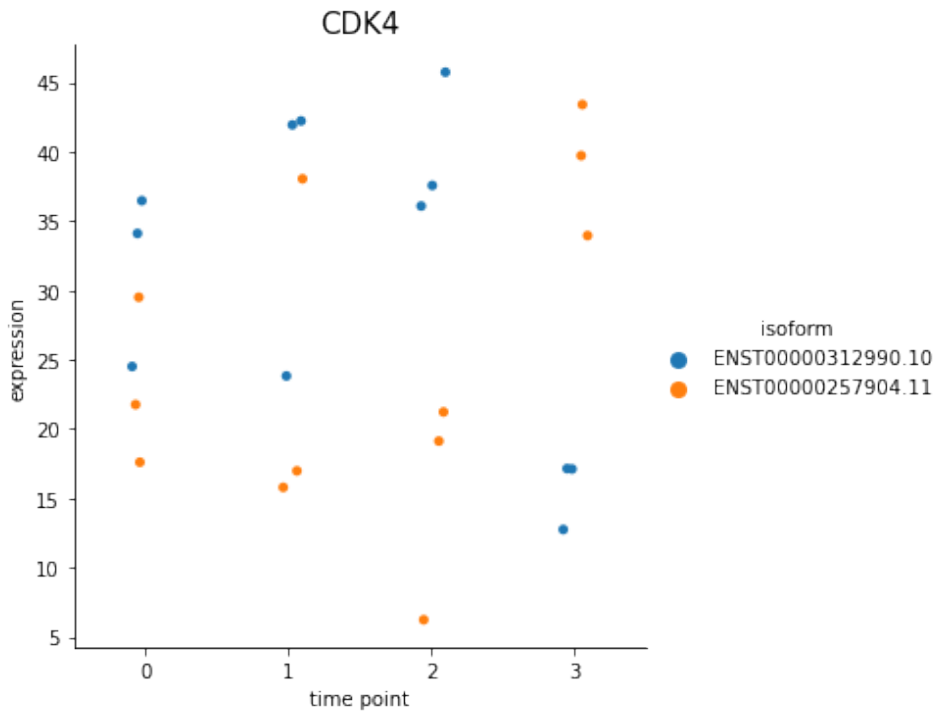
nbsphinx-code-borderwhite



nbsphinx-code-borderwhite



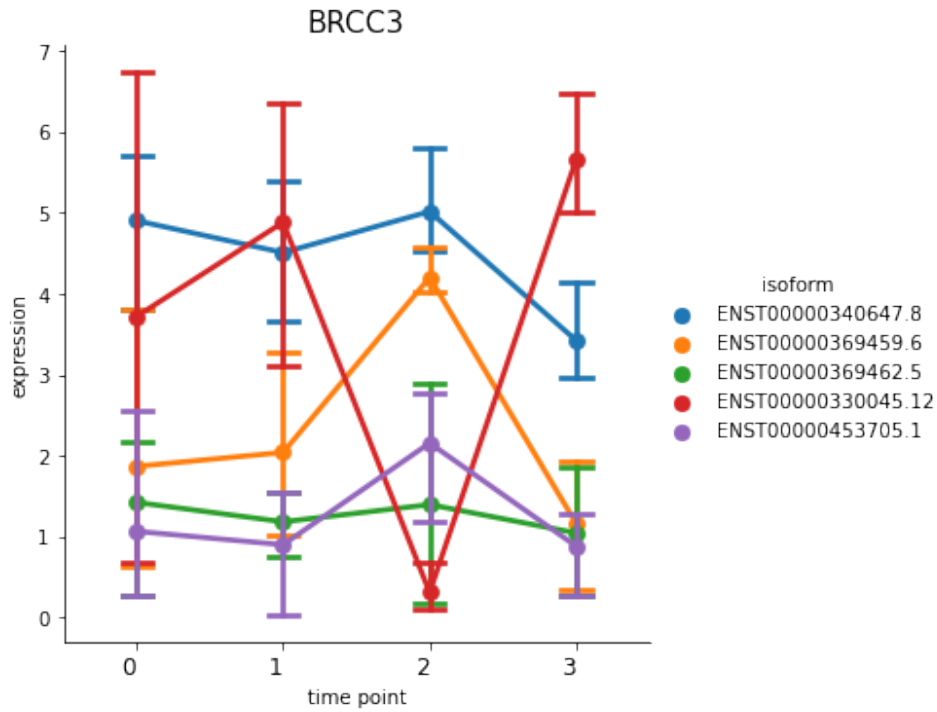
nbsphinx-code-borderwhite



nbsphinx-code-borderwhite

With this function, you can also plot all isoforms in one view, by putting `all_isoforms=True`.

[31]:
```
%matplotlib inline
spy.switch_plot("BRCC3", dset, ascov, all_isoforms=True)
```



nbsphinx-code-borderwhite

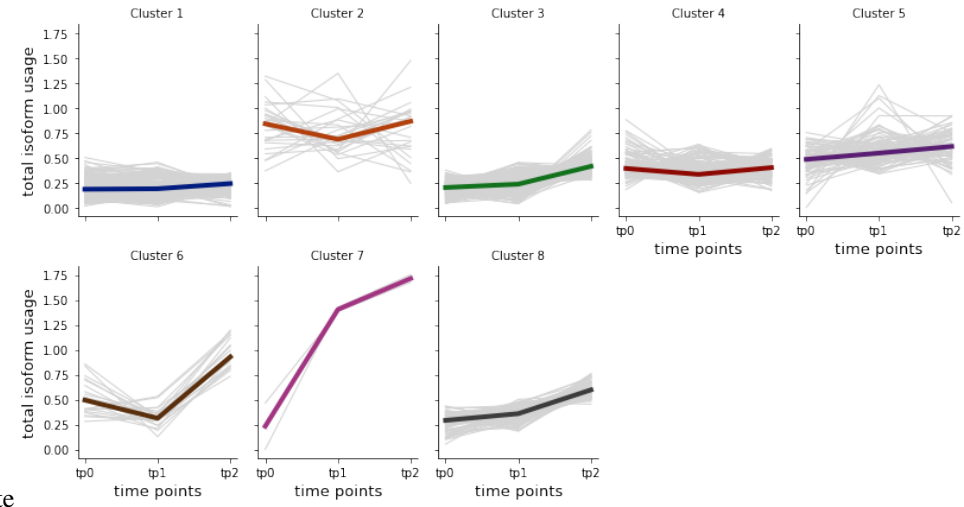
Clustering of total isoform usage

Total isoform usage measures the magnitude of change of isoform relative abundance over time. It indicates the changes due to post-transcriptional regulation (e.g. aternative splicing). You can cluster the genes with total isoform gene, thus getting groups of gene with similar patterns of change of total isoform usage over time.

[32]:
```
dset = iso.total_isoform_usage(ascov, gene_level=True)
```

[33]:
```
asclu = spy.clustering(dset, algorithm='hierarchical', metrics="euclidean", linkage="ward
↪", input_type="isoformusage", n_clusters=8, composite=False)
c = asclu.find_clusters()
```

[34]:
```
%matplotlib inline
spy.vis_all_clusters(asclu, col_wrap=5,y_label="total isoform usage")
```



nbsphinx-code-borderwhite

Gene set enrichment analysis

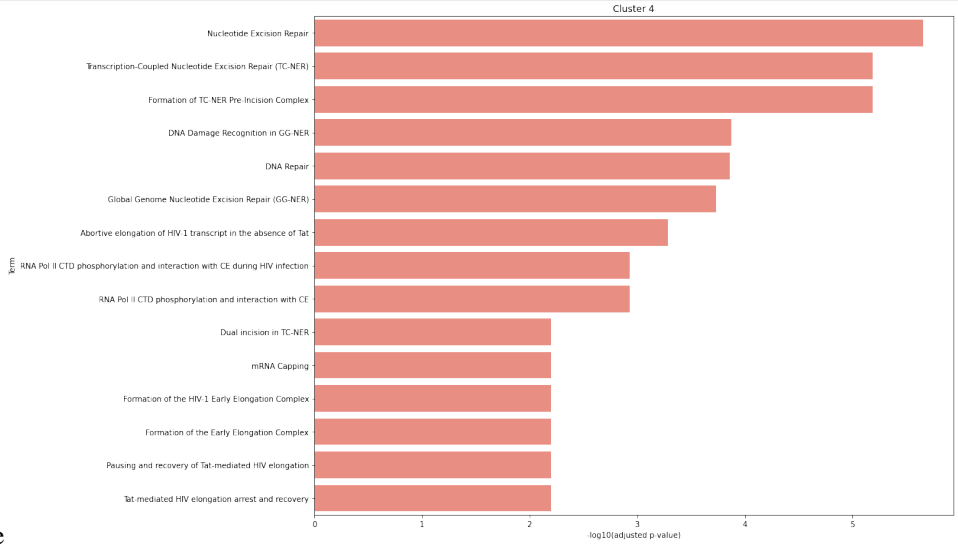
Perform gene set enrichment analysis using `clusters_gsea`. Change the gene_sets parameter into the choice of your knowledge base or gene set database, e.g. Reactome, KEGG, etc. Use `spy.list_genesets` to view the available knowledge base.

With the isoform switch detection, we can enriched the domains affect by the isoform switch events using `Nease`.

Nease only support human at the moment.

[35]:
```
asclu_go, _ = spy.clusters_gsea(dset, "hsapiens", is_results=ascov, method="nease", gene_
↪sets = ['Reactome'])
```

[36]:
```
%matplotlib inline
spy.gsea_plot(asclu_go, cluster=4, nterms=15)
```



nbsphinx-code-borderwhite

DOMINO on the domain level

[37]:
```
mods = spy.run_domino(asclu, network_file=bionet, output_file_path="newslices.txt")
```

Splicing Factor analysis

Perform co-expression analysis to look for splicing factors with similar expression patterns as isoform relative abundance.

[38]:
```
sf=spy.SF_coexpression(dset, padj_method="fdr_bh",corr_cutoff=0)
sf_df, cluster_sf = sf.coexpression_with_sf()
```

Read gtf annotation file for exons information.

[ ]:
```
# obtain exons gtf, or input customs gtf
subprocess.call("wget https://zenodo.org/record/7229557/files/exons.gtf?download=1 -O␣
↪exons.gtf", shell=True)
gtf = read_gtf("exons.gtf")
```

Co-expression analysis result in dataframe `sf_df` that contains the co-expressed splicing factors and the corresponding cluster. We are specifically interesting in splicing factors that positively and negatively correlated to isoforms in the same gene.

[40]:
```
thiscluster=6 #which cluster
end="5p" #specify 3'p or 5'p
sub = sf_df[sf_df['cluster']==thiscluster].drop_duplicates()
subcount = sub.groupby(['iso_genesymb', 'sf']) #extract list of co-expressed splicing␣
↪factors
subcount = subcount['bin_corr'].agg(sum).reset_index()
subcount=subcount[subcount['bin_corr']==0]
```

To perform motif search, `SF_motifsearch` creates a motif object, taking a list of splicing factors, list of genes (e.g. genes in a cluster), dataset object, and the gtf file). Motif searching (`search_motifs`) will be focusing on `loss exons` or `gain exons` with `exons` parameter and `3p` or `5p` end with `site` parameter. Background are the constants exons which can be done by `search_background_motifs`.

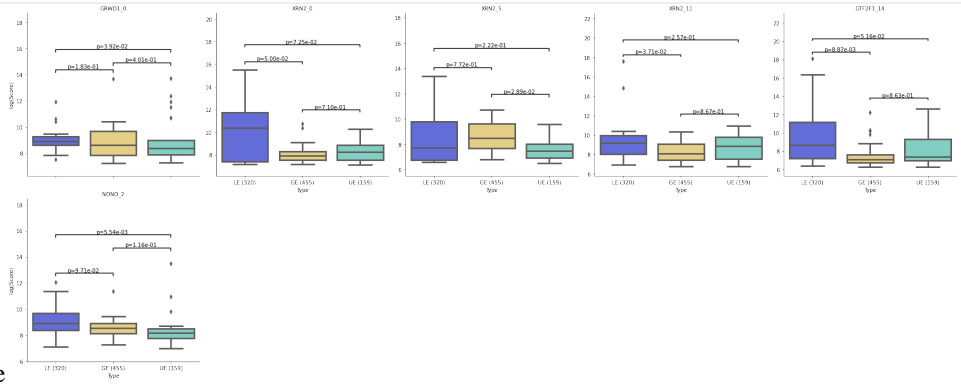
Motif search may take a long time, it is recommended to run in command line environment.

[41]:
```
motif=spy.SF_motifsearch(subcount['sf'].unique(), asclu.symbs_clusters[thiscluster],␣
↪dataset=dset, gtf=gtf)
lossmots = motif.search_motifs(exons="loss", site=end)
gainmots = motif.search_motifs(exons="gain", site=end)
bg = motif.search_background_motifs(site=end)
```

[42]:
```
alldf=motif._df_forvis(lossmots, gainmots, bg=bg)
pvals = motif._get_pvals(alldf)
```

[43]:
```
palette = {"lost exons":"#505BEA",
            "gained exons": "#F5D776",
           "background":"#76DACA"}
motif.vis_motif_density(alldf=alldf, pvals=pvals, pval_cutoff=0.05, palette=palette)
```

```
<Figure size 576x648 with 0 Axes>
```



nbsphinx-code-borderwhite

1.1.3 API

Import:

```
import spycone as spy
```

Import dataset

class spycone.**dataset**(*ts*, *species*, *keytype*, *reps1*, *timepts*, *gtf=None*, *gene_id=None*, *transcript_id=None*, *timeserieslist=None*, *symbs=None*, *discretization_steps=None*)

Input dataset.

Parameters

- **ts** – matrix or dataframe of the time series dataset.
- **gene_id** – list of gene id. (id that matched the biological network, default network : entrez ID). If gene_id is not given, entrez gene id will be mapped.
- **transcript_id** – list of transcript id if the expression matrix is in transcript-level.
- **keytype** – type of ID in gene list ('entrezgeneid', 'ensemblgeneid'). If only transcript ID is given, this serves as the keytype that will be mapped to gene_id.
- **species** – Specifying species ID for annotation (human : 9606, mouse: 10090)
- **reps1** – number of replicates
- **timepts** – number of time points
- **gtf** (*(optional)*) – provide corresponding gtf file for mapping gene names
- **symbs** (*(optional)*) – list of gene symbols or gene names (can be automatically mapped for human and mouse)
- **discretization_steps** (*(optional)*) – discretize the expression matrix according to the number of steps given

timeserieslist

automatically generated 3D-array for analysis

__copy__()

copy function for the dataset object

remove_objects()

remove the given index of the object

Import and store biological network

class spycone.**BioNetwork**(*path='human'*, *keytype='entrezid'*, ***kwargs*)

Storage of biological network.

Parameters

path – dir path to the network file or "human" as default human biogrid network

connected_nodes

total_node_degree_counts

total_undirected_edgecount_nodedegree

max_degree

all_degree

g :

generate network from file path if type is *file* or return itself if type is *network*

lst_g :

return list of node names

adj :

return adjcency matrix

removing_nodes :

remove nodes given the index

Preprocessing

class spycone.**preprocess**(*DataSet*, *BioNetwork=None*, *remove_low_var=False*, *cutoff=0*)

Preprocess data, remove objects without expression along all timepoints.

Parameters

- **DataSet** (*Dataset object.*) –
- **BioNetwork** (*BioNetwork object. If provided, objects that is not in the network will be removed.*) –
- **remove_low_var** (*(boolean) default=False. If true, objects with variance 0 will be removed.*) –
- **cutoff** (*default=0. If given, objects will mean expression across all timepoints lower than the cutoff will be removed.*) –

Return type

None, changes made directly in the DataSet object

Isoform-level function

class spycone.**iso_function**(*dataset*)

Isoform level analysis

Parameters

- **DataSet** – DataSet object
- **Species** – Species ID

detect_isoform_switch :

Parameters

- **combine** (*{'median', 'mean'}*) – aggregation methods for replicates. Default="median"
- **filtering** (*(boolean, default=True)*) – if True, low expression genes will be filtered out.
- **filter_cutoff** (*default=2*) – expression mean cutoff to filter.
- **corr_cutoff** (*default = 0.7*) – minimum correlation of isoform pairs to be included in the output.
- **p_val_cutoff** (*default = 0.05*) – significant p-value cutoff to be included in the output.
- **min_diff** (*default = 0.1*) – minimum differences of relative abundance to be included in the output.
- **event_im_cutoff** (*default = 0.1*) – minimum event importance to be included in the output.
- **adjustp** (*str {'fdr_bh' (default), 'holm_bonf', 'bonf'}*) – Method for multiple testing bonf: Bonferroni method holm_bonf: holm-bonferroni method fdr_bh: Benjamin-hochberg false discovery rate
- **n_permutations** – Number of permutations if permutation test is used.

total_isoform_usage :

Parameters

- **ids_result** – the result dataframe of isoform switch detection
- **norm** (*boolean*, default=True) – if True, it normalizes time series matrix to relative abundance to gene expression.
- **gene_level** (*boolean*, default=True) – if True, it calculates total isoform usage for each gene, otherwise individual isoform usage for each isoform

Return type

Create an instance for isoform switch analysis.

Perfom clustering

class spycone.**clustering**(*DataSet*, *algorithm*, *input_type*, *n_clusters=10*, *composite=False*, *BioNetwork=None*, *metric='euclidean'*, *prototypefunction='median'*, *linkage='average'*, *searchspace=20*, *seed=1234321*, *transform=None*, ***kwargs*)

Clustering object

Parameters

- **DataSet** – DataSet object
- **True)** (BioNetwork *(needed if composite is)* – BioNetwork object
- **input_type** (*str*) – clustering expression data put "expression", and clustering total isoform usage put "isoformusage"
- **algorithm** (*{'kmeans', 'kmedoids', 'dbscan', 'hierarchical', 'optics'}*) – clustering algorithms from sklearn
- **composite** (*boolean*, default=True) – if True, distance metrics is composited with inverse shortest path
- **metric** (*{'euclidean', 'correlation'}*) – metrics from sklearn
- **linkage** (*only for 'hierarchical' clustering {default='average', 'complete', 'ward'}*) –
- **prototypefunction** (*{default='median', 'mean'}*) – aggregation function for cluster prototypes
- **searchspace** (*(default=20)*) – range to search for optimal number of clusters

_prototype

Type
dictionary of prototypes for each cluster (keys)

_lables

The cluster label of each object

Type
array with length of object

genelist_clusters

Key and values pair of clustering with entrez ID (gene_list ID)

Type
dictionary of clusters

index_clusters

Key and values pair of clustering with indices

Type
dictionary of clusters

symbs_clusters

Key and values pair of clustering with gene symbols

Type
dictionary of clusters

_final_n_cluster

number of clusters

_silhouette_index

silhouette index of this clustering

Examples

GO terms enrichment analysis

spycone.**list_gsea**(*genelist*, *species*, *gene_sets=None*, *p_adjust_method='fdr_bh'*, *cutoff=0.05*, *method='gsea'*, *term_source='all'*)

Perform gene set enrichment on a list of gene

Parameters:

genelist species: input taxonomy ID if method is "nease", species name for "gsea" (e.g. hsapiens, mmusculus…) gene_sets: input a valid database name for "nease", ignore for "gsea" p_adjust_method: input one of the following: "fdr_bh", "bonf", "holm_bonf" cutoff: for adjusted p-value

spycone.**clusters_gsea**(*DataSet*, *species*, *gene_sets=None*, *is_results=None*, *cutoff=0.05*, *p_adjust_method='fdr_bh'*, *method='nease'*, *term_source='all'*)

Perform gene set enrichment on clusters (cluster object)

Parameters:

DataSet: Spycone dataset object

species: input taxonomy ID if method is "nease", species name for "gsea" (e.g. hsapiens, mmusculus…)

p_adjust_method: input one of the following: "fdr_bh", "bonf", "holm_bonf"

cutoff: for adjusted p-value

gene_sets : needed when method is "nease", input one of the database : 'PharmGKB','HumanCyc','Wikipathways','Reactome','KEGG','SMPDB','Signalink','NetPath','EHMN','INOH','BioCarta','PID'

method: "nease" or "gsea"

Return:

It returns two objects:

0. Dictionary containing the enrichment dataframes for each cluster.
1. If method is "nease", it returns nease object in the second object. For "gsea", it returns None.

spycone.**modules_gsea**(*X*, *clu*, *species*, *type='PPI'*, *p_adjust_method='fdr_bh'*, *cutoff=0.05*, *method='nease'*, *term_source='all'*)

Perform gene set enrichment on network modules after domino

Run DOMINO

spycone.**run_domino**(*target*, *name=None*, *is_results=None*, *scores=None*, *network_file='/home/docs/checkouts/readthedocs.org/user_builds/spycone/checkouts/stable/spycone/data/network*
output_file_path='./slices/slices.txt', *run_cluster=None*, *slice_threshold=0.3*, *module_threshold=0.05*, *prize_factor=0*, *n_steps=20*)

Parameters

- **target** – clustering object from spycone or gene list in entrez ID
- **is_results** (*DataFrame*) – Data Frame of isoform switch detection result
- **scores** (*None*) – activity scores of the genes (e.g. p-values from differential expression analysis)
- **run_cluster** – Specify the cluster name if you only want to run a specific cluster
- **file** (*Network*) – default: "data/network/network_human_PPIDDI.tab"
- **path** (*output file*) – default: output slices file for DOMINO.
- **slice_threshold** (*float*) –
- **module_threshold** (*float*) –
- **prize_factor** (*float*) –
- **n_steps** (*int*) –

spycone.**run_domain_domino**(*target*, *is_results*, *name=None*, *scores=None*, *network_file='/home/docs/checkouts/readthedocs.org/user_builds/spycone/checkouts/stable/spycone/data*
output_file_path='slices.txt', *run_cluster=None*, *slice_threshold=0.3*, *module_threshold=0.05*, *prize_factor=0*, *n_steps=20*)

Parameters

- **target** – clustering object from spycone or gene list in entrez ID
- **is_results** (*DataFrame*) – Data Frame of isoform switch detection result
- **scores** (*None*) – activity scores of the genes (e.g. p-values from differential expression analysis)
- **run_cluster** – Specify the cluster name if you only want to run a specific cluster
- **file** (*Network*) – default: "data/network/network_human_PPIDDI.tab"
- **path** (*output file*) – default: output slices file for DOMINO.
- **slice_threshold** (*float*) –
- **module_threshold** (*float*) –
- **prize_factor** (*float*) –
- **n_steps** (*int*) –

Visualization

spycone.**vis_all_clusters**(*clusterObj*, *x_label='time points'*, *y_label='expression'*, *Titles='Cluster {col_name}'*, *xtickslabels=None*, ***kwargs*)

Visualize all the clusters with cluster prototype

Parameters

- **clusterObj** – input clustering object with results
- **x_label** – x-axis label of the plot
- **y_label** – y-axis label of the plot
- **Titles** (*"Cluster {col_name}"*) – titles for each cluster

spycone.**switch_plot**(*gene*, *DataSet*, *ascov*, *xaxis_label=None*, *all_isoforms=False*, *relative_abundance=False*)

Switching plot for isoforms / Expression plot for non-switched genes if the input gene is not isoform switched, the expression plot will be plotted.

Parameters

- **gene** (*str*) – Input gene ID / symbs you would like to plot
- **DataSet** (*DataSet obj*) –
- **ascov** (*DataFrame*) – the result dataframe of your isoform switch detection
- **xaxis_label** (*list*) – x axis label for the plots

spycone.**gsea_plot**(*gsea_result*, *cluster*, *modules=None*, *nterms=None*)

Visualizing the functional enrichment

Parameters

- **gsea_result** (*dict*) – the results of gsea
- **cluster** (*str*) – the cluster number you would like to visualize
- **nterms** (*int*) – (optional) if you would like to visualize only subset of terms e.g. the top 10 terms

spycone.**vis_modules**(*mods*, *dataset*, *cluster*, *size=5*, *outputpng=None*)

Visualize all modules from one cluster in one figure.

Parameters:

mods: modules result

dataset: spycone dataset object

cluster: cluster number to visualize

size: minimum number of nodes in one module

outputpng: file path to save the figure (png)

spycone.**vis_better_modules**(*dataset*, *mod*, *cluster*, *dir*, *related_genes={}*, *module=None*)

Visualize modules with pydot in SVG format.

Parameters

- **dataset** – dataset obj

- **mod** – DOMINO results
- **cluster** – Cluster number to visualize
- **dir** – Local directory to save the images
- **related_genes** (*(Optional)*) – Set of genes to change the color of the nodes (color of the node outer border)

Splicing factor analysis

spycone.**SF_coexpression**(*dataset*, *padj_method='bonf'*, *corr_cutoff=0.7*, *padj_cutoff=0.05*, *method='pearson'*)

Return the coexpression between Splicing factor and isoforms

Parameters

- **dataset** (*input dataset object*) –
- **padj_method** (*Multiple testing method. Default=Bonferonni*) –
- **corr_cutoff** (*Correlation coefficient cutoff. Dafault=0.7*) –
- **padj_cutoff** (*Adjusted p-value cutoff. Default=0.05*) –
- **method** (*Method to calculate the correlation value.*) –

Return type

Create an instance for co-expression analysis of splicing factors and transcript abundance.

spycone.**SF_motifsearch**(*list_SF*, *list_genes*, *dataset*, *gtf*, *gc_ratio=0.6*, *flanking=400*)

Return the PSSM score of target SF and exons binding.

Parameters

- **list_SF** (*list*) – List of splicing factors / RBPs. E.g. the SF that is co-expressed to the isoform abundance.
- **list_genes** (*list*) – List of genes you want to check for SF binding sites. E.g. the cluster of genes that the input SF is co-expressed.
- **gtf_df** (*str or dataframe*) – gtf file path / dataframe
- **gc_ratio** (*GC content that makes up the background for PSSM score calculation. Default=0.6.*) –
- **flanking** (*Flanking region size*) –

Return type

Create an instance for SF motif enrichment analysis.

- search

INDEX