PathwayForte Documentation

Release 0.0.1-dev

Sarah Mubeen and Daniel Domingo-Fernández

Mar 02, 2021

COMMAND LINE INTERFACE :

Python package for pathway database benchmarking.

A Python package for benchmarking pathway databases with functional enrichment and prediction methods tasks.

CHAPTER

ONE

COMMAND LINE INTERFACE

PathwayForte commands.

1.1 pathway_forte

Run PathwayForte.

```
pathway_forte [OPTIONS] COMMAND [ARGS]...
```

1.1.1 datasets

List the available cancer datasets.

```
pathway_forte datasets [OPTIONS]
```

1.1.2 export

Generate gene set files using ComPath.

```
pathway_forte export [OPTIONS]
```

1.1.3 fcs

List of FCS Analyses.

```
pathway_forte fcs [OPTIONS] COMMAND [ARGS]...
```

gsea

Run GSEA on TCGA data.

```
pathway_forte fcs gsea [OPTIONS]
```

Options

-d, --data <data>
Required Name of the cancer dataset from TCGA

Options prad | ov | kirc | brca | lihc

-p, --permutations <permutations>
Number of permutations

Default 100

gsea-msig

Run GSEA on TCGA data using MSigDB gene sets.

```
pathway_forte fcs gsea-msig [OPTIONS]
```

Options

-d, --data <data>
Required Name of the cancer dataset from TCGA

Options prad | ov | kirc | brca | lihc

ssgsea

Run ssGSEA on TCGA data.

```
pathway_forte fcs ssgsea [OPTIONS]
```

Options

-d, --data <data>
Required Name of the cancer dataset from TCGA

Options prad | ov | kirc | brca | lihc

1.1.4 ora

Perform ORA analysis.

```
pathway_forte ora [OPTIONS] COMMAND [ARGS]...
```

hypergeometric

Perform one-tailed hypergeometric test enrichment.

```
pathway_forte ora hypergeometric [OPTIONS]
```

Options

-d, --genesets <genesets>
Required Path to GMT file

-s, --fold-changes <fold_changes>
Required Path to fold changes file

--no-threshold
Do not apply threshold

-o, --output <output>
Optional path for output JSON file

1.1.5 prediction

List of Prediction Methods.

```
pathway_forte prediction [OPTIONS] COMMAND [ARGS]...
```

binary

Train elastic net for binary prediction.

```
pathway_forte prediction binary [OPTIONS]
```

Options

-d, --data <data>
Required Name of the cancer dataset from TCGA

Options prad | ov | kirc | brca | lihc

--outer-cv <outer_cv>
Number of splits in outer cross-validation

Default 10

--inner-cv <inner_cv>
Number of splits in inner cross-validation

Default 10

-i, --max_iterations <max_iterations>
Number of max iterations to converge

Default 1000

--turn-off-warnings
Turns off warnings

subtype

Train subtype analysis.

```
pathway_forte prediction subtype [OPTIONS]
```

Options

-d, --ssgsea <ssgsea>
Required Path to ssGSEA file

-s, --subtypes <subtypes>
Required Path to the subtypes file

--outer-cv <outer_cv>
Number of splits in outer cross-validation

Default 10

--inner-cv <inner_cv>
Number of splits in inner cross-validation

Default 10

--chain-pca

--explained-variance <explained_variance>
Explained variance

Default 0.95

--turn-off-warnings
Turns off warnings

survival

Train survival model.

```
pathway_forte prediction survival [OPTIONS]
```

Options

-d, --data <data>
Required Name of dataset

--outer-cv <outer_cv>
Number of splits in outer cross-validation

Default 10

--inner-cv <inner_cv>
Number of splits in inner cross-validation

Default 10

--turn-off-warnings
Turns off warnings

test-stability-prediction

Test stability of prediction.

```
pathway_forte prediction test-stability-prediction [OPTIONS]
```

Options

-s, --ssgsea-scores-path <ssgsea_scores_path>
Required ssGSEA scores file

-p, --phenotypes-path <phenotypes_path>
Required Path to the phenotypes file

--outer-cv <outer_cv>
Number of splits in outer cross-validation

Default 10

--inner-cv <inner_cv>
Number of splits in inner cross-validation

Default 10

-i, --max_iterations <max_iterations>
Number of max iterations to converge

Default 1000

--turn-off-warnings
Turns off warnings

CHAPTER

TWO

PIPELINE

Pipelines from Pathway Forte.

CHAPTER

THREE

CONSTANTS

This module contains all the constants used in the PathwayForte repo.

pathway_forte.constants.**BIO2BEL_DATA_DIR = '/home/docs/.bio2bel/pathwayforte'**
Cancer Data Sets

pathway_forte.constants.**make_classifier_results_directory**()
Ensure that the result folder exists.

pathway_forte.constants.**MSIG_GSEA = '/home/docs/checkouts/readthedocs.org/user_builds/pathw**
Output files with results for GSEA

pathway_forte.constants.**make_gsea_export_directories**()
Ensure that gsea export directories exist.

pathway_forte.constants.**MSIG_SSGSEA = '/home/docs/checkouts/readthedocs.org/user_builds/pat**
Pickles with results for ssGSEA

pathway_forte.constants.**make_ssgsea_export_directories**()
Ensure that gsea export directories exist.

pathway_forte.constants.**check_gmt_files**()
Check if GMT files exist and returns GMT files as constant variables.

pathway_forte.constants.**GENESET_COLUMN_NAMES = {'kegg': 'KEGG Geneset', 'reactome': 'Reac**
Columns to read to perform ORA analysis.

CHAPTER

FOUR

OVER REPRESENTATION METHODS

This module contains the functions to run Over Representation Analysis (ORA).

pathway_forte.pathway_enrichment.over_representation.**read_fold_change_df**(*path*)

Read csv with gene names, fold changes and their p-values.

Return type DataFrame

pathway_forte.pathway_enrichment.over_representation.**filter_p_value**(*df*, *p_value=True*, *cut-off=0.05*)

Return significantly differentially expressed genes in fold change df.

pathway_forte.pathway_enrichment.over_representation.**perform_hypergeometric_test**(*genes_to_test*, *path-way_dict*, *gene_universe*, *ap-ply_threshold=*, *thresh-old=0.05*)

Perform hypergeometric tests.

Parameters

- **genes_to_test** (Set[str]) – gene set to test against pathway
- **pathway_dict** (Mapping[str, Set[str]]) – pathway name to gene set
- **gene_universe** (int) – number of HGNC symbols
- **apply_threshold** (bool) – return only significant pathways
- **threshold** (float) – significance threshold (by default 0.05)

Return type DataFrame

CHAPTER

FIVE

FUNCTIONAL CLASS SCORING

This module contain the functional class methods implemented in PathwayForte.

Currently this includes GSEA and ssGSEA.

pathway_forte.pathway_enrichment.functional_class.**create_cls_file**(*gene_expression_file*, *normal_sample_file*, *tumor_sample_file*, *data*)

Create categorical (e.g. tumor vs sample) class file format (i.e., .cls) for input into GSEA.

Parameters

- **gene_expression_file** (*str*) – Text file containing expression values for each gene from each sample.
- **normal_sample_file** (*str*) –
- **tumor_sample_file** (*str*) –
- **data** –

pathway_forte.pathway_enrichment.functional_class.**run_gsea**(*gene_exp*, *gene_set*, *phenotype_class*, *permutations=500*, *output_dir='/home/docs/checkouts/readthedocs.or*

Run GSEA on a given dataset with a given gene set.

Parameters

- **gene_exp** (str) – file with gene expression data
- **gene_set** (str) – gmt files containing pathway gene sets
- **phenotype_class** (str) – cls file containing information on class labels
- **permutations** (int) – number of permutations
- **output_dir** (str) – output directory

Returns

pathway_forte.pathway_enrichment.functional_class.**filter_gsea_results**(*gsea_results_path*, *source*, *kegg_manager=None*, *reactome_manager=None*, *wikipathways_manager=None*, *p_value=None*, *absolute_nes_filter=None*, *geneset_set_filter_minimum_size=*, *geneset_set_filter_maximum_size=*

Get top and bottom rankings from GSEA results.

Parameters

- **gsea_results_path** (str) – path to GSEA results in .tsv file format
- **source** –
- **kegg_manager** (Optional[Manager]) – KEGG manager
- **reactome_manager** (Optional[Manager]) – Reactome manager
- **wikipathways_manager** (Optional[Manager]) – WikiPathways manager
- **p_value** (Optional[float]) – maximum p value allowed
- **absolute_nes_filter** (Optional[float]) – filter by magnitude of normalized enrichment scores
- **geneset_set_filter_minimum_size** (Optional[int]) – filter to include a minimum number of genes in a gene set
- **geneset_set_filter_maximum_size** (Optional[int]) – filter to include a maximum number of genes in a gene set

Return type DataFrame

Returns list of pathways ranked as having the highest and lowest significant enrichment scores

pathway_forte.pathway_enrichment.functional_class.**merge_statistics**(*merged_pathways_df*, *dataset*)

Get statistics for pathways included in the merged gene sets dataFrame.

These include the proportion of pathways from each of the other databases and the proportion of pathways deriving from 2 or more primary resources

Parameters **merged_pathways_df** (DataFrame) – dataFrame containing pathways from multiple databases

Returns statistics of contents in merged dataset

pathway_forte.pathway_enrichment.functional_class.**rearrange_df_columns**(*df*)

Rearrange order of columns.

Return type DataFrame

pathway_forte.pathway_enrichment.functional_class.**get_pathway_names**(*database*, *pathway_df*, *kegg_manager=None*, *reactome_manager=None*, *wikipathways_manager=None*)

Get pathway names from database specific pathway IDs.

Parameters

- **database** (str) –
- **pathway_df** (DataFrame) –
- **kegg_manager** (Optional[Manager]) –
- **reactome_manager** (Optional[Manager]) –
- **wikipathways_manager** (Optional[Manager]) –

Returns

pathway_forte.pathway_enrichment.functional_class.**pathway_names_to_df**(*filtered_gsea_results_df*, *all_pathway_ids*, *source*, *kegg_manager=None*, *reactome_manager=None*, *wikipathways_manager=None*)

Get pathway names.

Parameters

- **filtered_gsea_results_df** –
- **all_pathway_ids** – list of pathway IDs
- **source** – pathway source (i.e., database name or 'MPath')
- **kegg_manager** (Optional[Manager]) – KEGG manager
- **reactome_manager** (Optional[Manager]) – Reactome manager
- **wikipathways_manager** (Optional[Manager]) – WikiPathways manager

Return type DataFrame

pathway_forte.pathway_enrichment.functional_class.**gsea_results_to_filtered_df**(*dataset*, *kegg_manager=No* *reactome_manager=N* *wikipathways_manager=N* *p_value=None*, *absolute_nes_filter=No* *gene_set_set_filter_mini* *gene_set_set_filter_max*

Get filtered GSEA results dataFrames.

pathway_forte.pathway_enrichment.functional_class.**get_pathways_by_resource**(*pathways*, *resource*)

Return pathways by resource.

Return type list

pathway_forte.pathway_enrichment.functional_class.**get_analogs_comparison_numbers**(*kegg_reactom* *reactome_wikipath* *wikipathways_kegg_pa* ***, *pathway_column=*

Get number of existing versus expected pairwise mappings.

pathway_forte.pathway_enrichment.functional_class.**get_pairwise_mapping_numbers**(*kegg_pathway_d* *reactome_pathway_d* *wikipathways_pathway_d*

Get number of existing versus expected pairwise mappings.

pathway_forte.pathway_enrichment.functional_class.**get_pairwise_mappings**(*kegg_pathway_df*, *reactome_pathway_df*, *wikipathways_pathway_df*)

Get pairwise mappings.

pathway_forte.pathway_enrichment.functional_class.**compare_database_results**(*df_1*, *resource_1*, *df_2*, *resource_2*, *mapping_dict*, *check_contradiction=F*

Compare pathways in the dataframe from enrichment results to evaluate the concordance in similar pathways.

pathway_forte.pathway_enrichment.functional_class.**get_matching_pairs**(*df_1*, *resource_1*, *df_2*, *resource_2*, *equivalent_mappings_dict*)

Get equivalent pathways and their direction of change.

pathway_forte.pathway_enrichment.functional_class.**run_ssgsea**(*filtered_expression_data*, *gene_set*, *output_dir='/home/docs/checkouts/readthedocs processes=1*, *max_size=3000*, *min_size=15*)

Run single sample GSEA (ssGSEA) on filtered gene expression data set.

Parameters

- **filtered_expression_data** (DataFrame) – filtered gene expression values for samples
- **gene_set** (str) – .gmt file containing gene sets
- **output_dir** (str) – output directory

Return type SingleSampleGSEA

Returns ssGSEA results in respective directory

pathway_forte.pathway_enrichment.functional_class.**filter_gene_exp_data**(*expression_data*, *gmt_file*)

Filter gene expression data file to include only gene names which are found in the gene set files.

Parameters

- **expression_data** (DataFrame) – gene expression values for samples
- **gmt_file** (str) – .gmt file containing gene sets

Returns Filtered gene expression data with genes with no correspondences in gene sets removed

Return type pandas.core.frame.DataFrame kegg_xml_parser.py

CHAPTER

SIX

PATHWAY TOPOLOGY METHODS

This module contain the topology-based topology methods implemented in PathwayForte used R wrappers and are located outside the main Python package in its corresponding R folder at https://github.com/pathwayforte/results/tree/master/R.

CHAPTER

SEVEN

BINARY PREDICTION

Prediction of binary classes such as tumor vs. normal patients.

Elastic Net regression with nested cross validation module.

This workflow trains an elastic net model for a binary classification task (e.g., tumor vs. normal patients). The training is conducted using a nested cross validation approach (the number of cross validation in both loops can be selected). The model used can be easily changed since most of the models in scikit-learn (the machine learning library used by this package) required the same input.

pathway_forte.prediction.binary.**ssgsea_nes_to_df**(*ssgsea_scores_csv*, *classes_file*, *removed_random=None*)

Create dataFrame of Normalized Enrichment Scores (NES) from ssGSEA of TCGA expression data.

Parameters

- **ssgsea_scores_csv** – Text file containing normalized ES for pathways from each sample
- **test_size** – Default test size is 0.25
- **removed_random** (Optional[int]) – Remove percentage of df

pathway_forte.prediction.binary.**get_l1_ratios**()

Return a list of values that are used by the elastic net as hyperparameters.

pathway_forte.prediction.binary.**train_elastic_net_model**(*x*, *y*, *outer_cv_splits*, *inner_cv_splits*, *l1_ratio*, *model_name*, *max_iter=None*, *export=True*)

Train elastic net model via a nested cross validation given expression data.

Uses a defined hyperparameter space for l1_ratio.

Parameters

- **x** (*numpy.array*) – 2D matrix of pathway scores and samples
- **y** (*list*) – class labels of samples
- **outer_cv_splits** (int) – number of folds for cross validation split in outer loop
- **inner_cv_splits** (int) – number of folds for cross validation split in inner loop
- **l1_ratio** (List[float]) – list of hyper-parameters for l1 and l2 priors
- **model_name** (str) – name of the model
- **max_iter** (Optional[int]) – default to 1000 to ensure convergence
- **export** (bool) – Export the models using joblib

Return type Tuple[List[float], List[float]]

Returns A list of AUC-ROC scores

CHAPTER

EIGHT

MULTI-CLASS PREDICTION

Prediction of multi-class labels such as tumor subtypes.

pathway_forte.prediction.**multiclass**

CHAPTER

NINE

SURVIVAL PREDICTION

Prediction of survival based on clinical and pathway patient data.

pathway_forte.prediction.**survival**

CHAPTER

TEN

UTILS

Complementary methods for prediction analysis.

Utilities for prediction.

pathway_forte.prediction.utils.**pca_chaining**(*train*, *test*, *n_components*)

Chain PCA with logistic regression.

Parameters

- **train** (*pandas.core.frame.DataFrame*) – Training set to apply dimensionality reduction to
- **test** (*pandas.core.series.Series*) – Test set to apply dimensionality reduction to
- **n_components** – Amount of variance retained

Return type Tuple

Returns array-like, shape (n_samples, n_components)

CHAPTER

ELEVEN

MAPPINGS METHODS

Methods related to ComPath mappings.

Function to deal with ComPath mappings.

pathway_forte.mappings.**get_mapping_dict**(*df*, *mapping_type*)
Create a dictionary with ComPath mappings for each pathway.

Return type Mapping[Tuple[str, str], List[Tuple[str, str]]]

pathway_forte.mappings.**get_equivalent_pairs**(*df*)
Get equivalent pairs of pathways from 2 databases.

Parameters **df** (DataFrame) – pairwise mappings dataframe

Returns equivalent pathway pairs dictionary {(SOURCE_RESOURCE,SOURCE_ID):[(TARGET_RESOURCE,TARGET_

Return type dict[list]

pathway_forte.mappings.**load_compath_mapping_dfs**()
Load ComPath mappings data frames.

Return type Tuple[DataFrame, DataFrame, DataFrame, DataFrame]

pathway_forte.mappings.**get_equivalent_mappings_dict**()
Get mapping dictionary of all equivalent pairs of pathways.

Special mappings are not included in the overall mappings as some of the WP pathways possess identical IDs.

Return type Mapping[Tuple[str, str], List[Tuple[str, str]]]

CHAPTER

TWELVE

INSTALLATION

`pathway_forte` can be installed from PyPI with the following command in your terminal:

```
$ python3 -m pip install pathway_forte
```

The latest code can be installed from GitHub with:

```
$ python3 -m pip install git+https://github.com/pathwayforte/pathway-forte.git
```

For developers, the code can be installed with:

```
$ git clone https://github.com/pathwayforte/pathway-forte.git
$ cd pathway-forte
$ python3 -m pip install -e .
```

CHAPTER

THIRTEEN

MAIN COMMANDS

The table below lists the main commands of PathwayForte.

Command	Action
datasets	Lists of Cancer Datasets
export	Export Gene Sets using ComPath
ora	List of ORA Analyses
fcs	List of FCS Analyses
prediction	List of Prediction Methods

CHAPTER

FOURTEEN

FUNCTIONAL ENRICHMENT METHODS

- **ora**. Lists Over-Representation Analyses (e.g., one-tailed hyper-geometric test).
- **fcs**. Lists Functional Class Score Analyses such as GSEA and ssGSEA using GSEAPy.

CHAPTER

FIFTEEN

PREDICTION METHODS

`pathway_forte` enables three classification methods (i.e., binary classification, training SVMs for multi-classification tasks, or survival analysis) using individualized pathway activity scores. The scores can be calculated from any pathway with a variety of tools (see[1]) using any pathway database that enables to export its gene sets.

- **binary**. Trains an elastic net model for a binary classification task (e.g., tumor vs. normal patients). The training is conducted using a nested cross validation approach (the number of cross validation in both loops can be selected). The model used can be easily changed since most of the models in scikit-learn (the machine learning library used by this package) required the same input.
- **subtype**. Trains a SVM model for a multi-class classification task (e.g., predict tumor subtypes). The training is conducted using a nested cross validation approach (the number of cross validation in both loops can be selected). Similarly as the previous classification task, other models can quickly be implemented.
- **survival**. Trains a Cox's proportional hazard's model with elastic net penalty. The training is conducted using a nested cross validation approach with a grid search in the inner loop. This analysis requires pathway activity scores, patient classes and lifetime patient information.

[1] Lim, S., *et al.* (2018). Comprehensive and critical evaluation of individualized pathway activity measurement tools on pan-cancer data. *Briefings in bioinformatics*, bby125.

CHAPTER

SIXTEEN

OTHER

- **export**. Export GMT files with current gene sets for the pathway databases included in ComPath[2].
- **datasets**. Lists the TCGA data sets[3] that are ready to run in `pathway_forte`.

[2] Domingo-Fernández, D., *et al.* (2018). ComPath: An ecosystem for exploring, analyzing, and curating mappings across pathway databases. *npj Syst Biol Appl.*, 4(1):43.

[3] Weinstein, J. N., *et al.* (2013). The cancer genome atlas pan-cancer analysis project. *Nature genetics*, 45(10), 1113.

CHAPTER

SEVENTEEN

REFERENCES

CHAPTER

EIGHTEEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX

Symbols

B

C

F

G

L

M

P

R

S

T