Version:
v2.2
Last Modified:
May 18, 2021

Samudrala Group
Department of Biomedical Informatics
Jacobs School of Medicine and Biomedical Sciences
University at Buffalo
77 Goodell St., Suite 540
Buffalo, NY 14203

1 CANDO

Computational **A**nalysis of **N**ovel **D**rug **O**pportunities

CANDO is a unique computational drug discovery, design, and repurposing platform.

2 Install

You may download the source code via the releases or cloning the git repository. However, we suggest using anaconda to install the CANDO package, as this is the easiest and quickest way to start using our platform!

The CANDO package relies on multiple "conda-forge" dependencies. Therefore, we require that you add "conda-forge" to your anaconda channels:

```
conda config --add channels conda-forge
```

Then you can install CANDO using the following command:

```
conda install -c ram-compbio cando
```

3 Test

You can test your install by running our script:

run_test.py

4 Authors

- William Mangione
- Zackary Falls
- James Schuler
- Matt Hudson
- Liana Bruggemann
- Ram Samudrala

For general questions, please contact Ram Samudrala (`ram@compbio.org`). For direct questions about source code for cando.py, please contact William Mangione (`wmangion@buffalo.edu`) or Zackary Falls (`zmfalls@buffalo.edu`).

5 LICENSE

6 Namespace Index

6.1 Packages

Here are the packages with brief descriptions (if available):

7 Hierarchical Index

7.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

object

8 Class Index

8.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

9 File Index

9.1 File List

Here is a list of all files with brief descriptions:

10 Namespace Documentation

10.1 cando Namespace Reference

Classes

- class ADR

  An object to represent an adverse reaction.
- class CANDO

  An object to represent all aspects of CANDO (compounds, indications, matrix, etc.)
- class Compound

  An object to represent a compound/drug.
- class Compound_pair

  An object to represent a compound/drug-pair.
- class Indication

  An object to represent an indication (disease)
- class Matrix

  An object to represent a matrix.
- class Pathway

  An object to represent a pathway.
- class Protein

  An object to represent a protein.

Functions

- def generate_matrix (v="v2.2", fp="rd_ecfp4", vect="int", dist="dice", org="nrpdb", bs="coach", c_cutoff=0.0, p_cutoff=0.0, percentile_cutoff=0.0, i_score="P", out_file='', out_path=".", nr_ligs=True, lig_name=False, ncpus=1)
- def calc_scores (c, c_fps, l_fps, p_dict, dist, pscore_cutoff=0.0, cscore_cutoff=0.0, percentile_cutoff=0.0, i_score='P', nr_ligs=[ ], lig_name=False)
- def generate_signature (cmpd_file, fp="rd_ecfp4", vect="int", dist="dice", org="nrpdb", bs="coach", c_cutoff=0.0, p_cutoff=0.0, percentile_cutoff=0.0, i_score="P", out_file='', out_path=".", nr_ligs=True)
- def add_cmpds (cmpd_list, fp="rd_ecfp4", vect="int", cmpd_dir=".", v=None)
- def cosine_dist (A)
- def tanimoto_sparse (str1, str2)

  Calculate the tanimoto coefficient for a pair of sparse vectors.
- def tanimoto_dense (list1, list2)

  Calculate the tanimoto coefficient for a pair of dense vectors.
- def get_fp_lig (fp)

  Download precompiled binding site ligand fingerprints using the given fingerprint method.
- def get_data (v="v2.2", org='nrpdb')

  Download CANDO v2.0 data.
- def get_tutorial ()

  Download data for tutorial.
- def get_test ()

  Download data for test script.
- def dl_dir (url, out, l)

  Function to recursively download a directory.
- def dl_file (url, out_file)

  Function to download a file.

10.1.1 Function Documentation

10.1.1.1 add_cmpds()

```
def cando.add_cmpds (
            cmpd_list,
            fp = "rd_ecfp4",
            vect = "int",
            cmpd_dir = ".",
            v = None )
```

10.1.1.2 calc_scores()

```
def cando.calc_scores (
            c,
            c_fps,
            l_fps,
            p_dict,
            dist,
            pscore_cutoff = 0.0,
            cscore_cutoff = 0.0,
            percentile_cutoff = 0.0,
            i_score = 'P',
            nr_ligs = [],
            lig_name = False )
```

10.1.1.3 cosine_dist()

```
def cando.cosine_dist (
            A )
```

10.1.1.4 dl_dir()

```
def cando.dl_dir (
            url,
            out,
            l )
```

Function to recursively download a directory.

Prints the name of the directory and a progress bar.

Parameters

url	str: URL of the dir to be downloaded
out	str: Path to where the dir will be downloaded
l	list: List of files in dir to be downloaded

10.1.1.5 dl_file()

```
def cando.dl_file (
            url,
            out_file )
```

Function to download a file.

Prints the name of the file and a progress bar.

Parameters

url	str: URL of the file to be downloaded
out_file	str: File path to where the file will be downloaded

10.1.1.6 generate_matrix()

```
def cando.generate_matrix (
            v = "v2.2",
            fp = "rd_ecfp4",
            vect = "int",
            dist = "dice",
            org = "nrpdb",
            bs = "coach",
            c_cutoff = 0.0,
            p_cutoff = 0.0,
            percentile_cutoff = 0.0,
            i_score = "P",
            out_file = '',
            out_path = ".",
            nr_ligs = True,
            lig_name = False,
            ncpus = 1 )
```

10.1.1.7 generate_signature()

```
def cando.generate_signature (
            cmpd_file,
            fp = "rd_ecfp4",
            vect = "int",
            dist = "dice",
            org = "nrpdb",
            bs = "coach",
            c_cutoff = 0.0,
            p_cutoff = 0.0,
            percentile_cutoff = 0.0,
            i_score = "P",
            out_file = '',
            out_path = ".",
            nr_ligs = True )
```

10.1.1.8 get_data()

```
def cando.get_data (
            v = "v2.2",
            org = 'nrpdb' )
```

Download CANDO v2.0 data.

This data includes:

- Compound mapping (approved and all)
- Indication-compound mapping
- Scores file for all approved compounds (fingerprint: rd_ecfp4)
- Matrix file for approved drugs (2,162) and all proteins (14,610) (fingerprint: rd_ecfp4)

10.1.1.9 get_fp_lig()

```
def cando.get_fp_lig (
            fp )
```

Download precompiled binding site ligand fingerprints using the given fingerprint method.

Parameters

fp	str: Fingerprinting method used to compile each binding site ligand fingerprint

10.1.1.10 get_test()

```
def cando.get_test ( )
```

Download data for test script.

This data includes:

- Test Matrix (Approved drugs (2,162) and 64 proteins)
- v2.0 Compound mapping (approved and all)
- v2.0 Indication - Compound mapping
- Compound scores file for all approved compounds (fingerprint: rd_ecfp4)
- Test Protein scores file (64 proteins) for all binding site ligands for each Protein (fingerprint: rd_ecfp4)
- Test Compound in PDB format to generate a new fingerprint and vector in the Matrix
- Directory of test Compounds in PDB format to generate multiple new fingerprints and vectors in the Matrix
- Test Pathways set

10.1.1.11 get_tutorial()

```
def cando.get_tutorial ( )
```

Download data for tutorial.

This data includes:

- Example Matrix (Approved drugs (2,162) and 64 proteins)
- v2.0 Compound mapping (approved and all)
- v2.0 Indication - Compound mapping
- Compound scores file for all approved compounds (fingerprint: rd_ecfp4)
- Example Protein scores file (64 proteins) for all binding site ligands for each Protein (fingerprint: rd_ecfp4)
- Example Compound in PDB format to generate a new fingerprint and vector in the Matrix
- Example Pathways set

10.1.1.12 tanimoto_dense()

```
def cando.tanimoto_dense (
            list1,
            list2 )
```

Calculate the tanimoto coefficient for a pair of dense vectors.

Parameters

list1	list: List of positions that have a 1 in first compound fingerprint
list2	list: List of positions that have a 1 in second compound fingerprint

10.1.1.13 tanimoto_sparse()

```
def cando.tanimoto_sparse (
            str1,
            str2 )
```

Calculate the tanimoto coefficient for a pair of sparse vectors.

Parameters

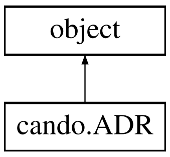
str1	str: String of 1s and 0s representing the first compound fingerprint
str2	str: String of 1s and 0s representing the second compound fingerprint

11 Class Documentation

11.1 cando.ADR Class Reference

An object to represent an adverse reaction.

Inheritance diagram for cando.ADR:



Public Member Functions

- def __init__ (self, id_, name)

Public Attributes

- id_

    str: Identification for the given ADR
- name

    str: Name of the given ADR
- compounds

    list: Compound objects associated with the given ADR
- compound_pairs

11.1.1 Detailed Description

An object to represent an adverse reaction.

11.1.2 Constructor & Destructor Documentation

11.1.2.1 __init__()

```
def cando.ADR.__init__ (
              self,
              id_,
              name )
```

11.1.3 Member Data Documentation

11.1.3.1 compound_pairs

```
cando.ADR.compound_pairs
```

11.1.3.2 compounds

```
cando.ADR.compounds
```

list: Compound objects associated with the given ADR

11.1.3.3 id_

```
cando.ADR.id_
```

str: Identification for the given ADR

11.1.3.4 name

```
cando.ADR.name
```

str: Name of the given ADR

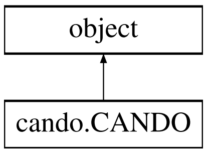
The documentation for this class was generated from the following file:

- cando.py

11.2 cando.CANDO Class Reference

An object to represent all aspects of CANDO (compounds, indications, matrix, etc.)

Inheritance diagram for cando.CANDO:



Public Member Functions

- def __init__ (self, c_map, i_map, matrix='', compute_distance=False, save_dists='', read↩_dists='', pathways='', pathway_quantifier='max', indication_pathways='', indication_↩proteins='', similarity=False, dist_metric='rmsd', protein_set='', rm_zeros=False, rm↩_compounds='', ddi_compounds='', ddi_adrs='', adr_map='', protein_distance=False, protein_map='', ncpus=1)
- def search_compound (self, name, n=5)

  Print closest Compound names/IDs for input search str.
- def get_compound (self, cmpd_id)

  Get Compound object from Compound id or fuzzy match to Compound name.
- def get_compound_pair (self, ids)

  Get Compound object from Compound id.

- def get_protein (self, protein_id)

  Get Protein object from Protein id.
- def get_indication (self, ind_id)

  Get Indication object from Indication id.
- def get_pathway (self, id_)

  Get Pathway object from Pathway id.
- def get_adr (self, id_)

  Get ADR (adverse drug reaction) from ADR id.
- def search_indication (self, name, n=5)

  Print closest MeSH IDs for Indication name.
- def top_targets (self, cmpd, n=10, negative=False, save_file='')

  Get the top scoring protein targets for a given compound.
- def common_targets (self, cmpds_file, n=10, negative=False, save_file='')

  Get the top scoring protein targets for a given compound.
- def virtual_screen (self, protein, n=10, negative=False, compound_set='all', save_file='')

  Get the top scoring protein targets for a given compound.
- def uniprot_set_index (self, prots)

  Gather proteins from input matrix that map to UniProt IDs from 'protein_set=' param.
- def generate_similar_sigs (self, cmpd, sort=False, proteins=[ ], aux=False)

  For a given compound, generate the similar compounds using distance of sigs.
- def generate_similar_sigs_cp (self, cmpd_pair, sort=False, proteins=[ ], aux=False)

  For a given compound pair, generate the similar compound pairs using distance of sigs.
- def generate_some_similar_sigs (self, cmpds, sort=False, proteins=[ ], aux=False)

  For a given list of compounds, generate the similar compounds based on dist of sigs This is pathways/genes for all intents and purposes.
- def quantify_pathways (self, indication=None)

  Uses the pathway quantifier defined in the CANDO instantiation to make a pathway signature for all pathways in the input file (NOTE: does not compute distances)
- def results_analysed (self, f, metrics, effect_type)

  Creates the results analysed named file for the benchmarking and computes final avg indication accuracies.
- def canbenchmark (self, file_name, indications=[ ], continuous=False, bottom=False, ranking='standard', adrs=False)

  Benchmarks the platform based on compound similarity of those approved for the same diseases.
- def canbenchmark_associated (self, file_name, indications=[ ], continuous=False, ranking='standard')

  Benchmark only the compounds in the indication mapping, aka get rid of "noisy" compounds.
- def canbenchmark_bottom (self, file_name, indications=[ ], ranking='standard')

  Benchmark the reverse ranking of similar compounds as a control.
- def canbenchmark_ndcg (self, file_name)
- def canbenchmark_cluster (self, n_clusters=5)

Benchmark using k-means clustering.

- def compounds_analysed (self, f, metrics)
- def canbenchmark_compounds (self, file_name, indications=[ ], continuous=False, bottom=False, ranking='standard', adrs=False)
- def canbenchmark_ddi (self, file_name, indications=[ ], adrs=[ ], continuous=False, bottom=False, ranking='standard')

  Benchmarks the platform based on compound similarity of those approved for the same diseases.
- def ml (self, method='rf', effect=None, benchmark=False, adrs=False, predict=[ ], threshold=0.5, negative='random', seed=42, out='')

  create an ML classifier for a specified indication or all inds (to benchmark) predict (used w/ 'effect=' - indication or ADR) is a list of compounds to classify with the trained ML model out=X saves benchmark SUMMARY->SUMMARY_ml_X; raw results->raw_results/raw_results_ml↩ _X (same for RAN) currently supports random forest ('rf'), support vector machine ('svm'), 1-class SVM ('1csvm'), and logistic regression ('log') models are trained with leave-one-out cross validation during benchmarking
- def raw_results_roc (self, rr_files, labels, save='roc-raw_results.pdf')
- def canpredict_denovo (self, method='count', threshold=0.0, topX=10, ind_id=None, proteins=None, cmpd_set='all', save='')

  This function is used for predicting putative therapeutics for an indication of interest by summing/counting the number of interactions above a certain input interaction threshold for all proteins or a specified subset of proteins.
- def canpredict_compounds (self, ind_id, n=10, topX=10, keep_associated=False, cmpd↩ _set='all', save='')

  This function is used for predicting putative therapeutics for an indication of interest using a homology-based approach.
- def canpredict_indications (self, cmpd, n=10, topX=10, save='')

  This function is the inverse of canpredict_compounds.
- def canpredict_adr (self, cmpd, n=10, topX=10, save='')

  This function is the inverse of canpredict_compounds.
- def canpredict_ddi_cmpds (self, cmpd, n=10, topX=10, save='')
- def canpredict_ddi_adrs (self, cmpd_pair, n=10, topX=10, save='')
- def similar_compounds (self, cmpd, n=10)

  Computes and prints the top n most similar compounds to an input Compound object cando↩ _cmpd or input novel signature new_sig.
- def add_cmpd (self, new_sig, new_name='')

  Add a new Compound object to the platform.
- def sigs (self, rm)

  Return a list of all signatures, rm is a list of compound ids you do not want in the list.
- def save_dists_to_file (self, f)

  Write calculated distances of all compounds to all compounds to file.
- def fusion (self, cando_objs, out_file='', method='sum')

  This function re-ranks the compounds according to the desired comparison specified by 'method' -> currently supports 'min', 'avg', 'mult', and 'sum'.

- def normalize (self)

    Normalize the distance scores to between [0,1].
- def __str__ (self)

    Print stats about the CANDO object.

Public Attributes

- c_map

    str: File path to the compound mapping file (relative or absolute)
- i_map

    str: File path to the indication mapping file (relative or absolute)
- matrix

    str: File path to the cando matrix file (relative or absolute)
- protein_set

    str: File path to protein subset file (relative or absolute)
- pathways

    str: File path to pathway file
- accuracies
- compute_distance

    bool: Calculate the distance for each Compound against all other Compounds using chosen distance metric
- clusters
- rm_zeros

    bool: Remove Compounds with all-zero signatures from CANDO object
- rm_compounds

    list: Compounds to remove from the CANDO object
- rm_cmpds
- save_dists

    bool: Write the calculated distances to file after computation (set compute_distances=True)
- read_dists

    str: File path to pre-computed distance matrix
- similarity

    bool: Use similarity instead of distance
- dist_metric

    str: Distance metric to be used for computing Compound-Compound distances
- ncpus

    int: Number of CPUs used for parallelization
- pathway_quantifier

    str: Method used to quantify a all Pathways
- indication_pathways

str: File path to Indication-Pathway association file

- indication_proteins

  str: File path to Indication-Protein association file

- adr_map

  str: File path to ADR mapping file

- protein_map

  str: File path to Protein metadata mapping file

- ddi_compounds
- ddi_adrs
- protein_distance
- proteins
- protein_id_to_index
- compounds
- compound_ids
- compound_pairs
- compound_pair_ids
- indications
- indication_ids
- adrs
- adr_ids
- short_matrix_path
- short_read_dists
- short_protein_set
- cmpd_set
- data_name

11.2.1 Detailed Description

An object to represent all aspects of CANDO (compounds, indications, matrix, etc.)

To instantiate you need the compound mapping (c_map), an indication mapping file (i_map), and typically and a compound-protein matrix (matrix=) or or precomputed compound-compound distance matrix (read_rmsds=), but those are optional.

11.2.2 Constructor & Destructor Documentation

11.2.2.1 __init__()

```
def cando.CANDO.__init__ (
            self,
            c_map,
            i_map,
            matrix = '',
            compute_distance = False,
            save_dists = '',
            read_dists = '',
            pathways = '',
            pathway_quantifier = 'max',
            indication_pathways = '',
            indication_proteins = '',
            similarity = False,
            dist_metric = 'rmsd',
            protein_set = '',
            rm_zeros = False,
            rm_compounds = '',
            ddi_compounds = '',
            ddi_adrs = '',
            adr_map = '',
            protein_distance = False,
            protein_map = '',
            ncpus = 1 )
```

11.2.3 Member Function Documentation

11.2.3.1 __str__()

```
def cando.CANDO.__str__ (
            self )
```

Print stats about the CANDO object.

11.2.3.2 add_cmpd()

```
def cando.CANDO.add_cmpd (
            self,
            new_sig,
            new_name = '' )
```

Add a new Compound object to the platform.

Parameters

new_sig	str: Path to the tab-separated interaction scores
new_name	str: Name for the new Compound

Returns

cmpd Compound: Compound object

11.2.3.3 canbenchmark()

```
def cando.CANDO.canbenchmark (
            self,
            file_name,
            indications = [],
            continuous = False,
            bottom = False,
            ranking = 'standard',
            adrs = False )
```

Benchmarks the platform based on compound similarity of those approved for the same diseases.

Parameters

file_name	str: Name to be used for the various results files (e.g. file_name=test –> summary_test.tsv)
indications	list or str: List of Indication ids to be used for this benchmark, otherwise all will be used.
continuous	bool: Use the percentile of distances from the similarity matrix as the cutoffs for benchmarking
bottom	bool: Reverse the ranking (descending) for the benchmark
ranking	str: What ranking method to use for the compounds. This really only affects ties. (standard, modified, and ordinal)
adrs	bool: ADRs are used as the Compounds' phenotypic effects instead of Indications

11.2.3.4 canbenchmark_associated()

```
def cando.CANDO.canbenchmark_associated (
            self,
            file_name,
            indications = [],
            continuous = False,
            ranking = 'standard' )
```

Benchmark only the compounds in the indication mapping, aka get rid of "noisy" compounds.

This function returns the filtered CANDO object in the event that you want to explore further.

Parameters

file_name	str: Name to be used for the variosu results files (e.g. file_name=test –> summary_test.tsv)
indications	list: List of Indication ids to be used for this benchmark, otherwise all will be used.
continuous	bool: Use the percentile of distances from the similarity matrix as the cutoffs for benchmarking
ranking	str: What ranking method to use for the compounds. This really only affects ties. (standard, modified, and ordinal)

11.2.3.5 canbenchmark_bottom()

```
def cando.CANDO.canbenchmark_bottom (
            self,
            file_name,
            indications = [],
            ranking = 'standard' )
```

Benchmark the reverse ranking of similar compounds as a control.

Parameters

file_name	str: Name to be used for the variosu results files (e.g. file_name=test –> summary_test.tsv)
indications	list: List of Indication ids to be used for this benchmark, otherwise all will be used.
ranking	str: What ranking method to use for the compounds. This really only affects ties. (standard, modified, and ordinal)

11.2.3.6 canbenchmark_cluster()

```
def cando.CANDO.canbenchmark_cluster (
            self,
            n_clusters = 5 )
```

Benchmark using k-means clustering.

Parameters

n_clusters	int: Number of clusters for k-means

11.2.3.7 canbenchmark_compounds()

```
def cando.CANDO.canbenchmark_compounds (
            self,
            file_name,
            indications = [],
            continuous = False,
            bottom = False,
            ranking = 'standard',
            adrs = False )
```

11.2.3.8 canbenchmark_ddi()

```
def cando.CANDO.canbenchmark_ddi (
            self,
            file_name,
            indications = [],
            adrs = [],
            continuous = False,
            bottom = False,
            ranking = 'standard' )
```

Benchmarks the platform based on compound similarity of those approved for the same diseases.

Parameters

file_name	str: Name to be used for the various results files (e.g. file_name=test –> summary_test.tsv)
indications	list: List of Indication ids to be used for this benchmark, otherwise all will be used.
continuous	bool: Use the percentile of distances from the similarity matrix as the cutoffs for benchmarking
bottom	bool: Reverse the ranking (descending) for the benchmark
ranking	str: What ranking method to use for the compounds. This really only affects ties. (standard, modified, and ordinal)
adrs	bool: ADRs are used as the phenotypic effect instead of Indications

11.2.3.9 canbenchmark_ndcg()

```
def cando.CANDO.canbenchmark_ndcg (
            self,
            file_name )
```

11.2.3.10 canpredict_adr()

```
def cando.CANDO.canpredict_adr (
            self,
            cmpd,
            n = 10,
            topX = 10,
            save = '' )
```

This function is the inverse of canpredict_compounds.

Input a compound of interest cando_cmpd (or a novel protein signature of interest new_sig) and the most similar compounds to it will be computed. The ADRs associated with the top n most similar compounds to the query compound will be examined to see if any are repeatedly enriched.

Parameters

cmpd	Compound: Compound object to be used
n	int: top number of similar Compounds to be used for prediction
topX	int: top number of predicted Indications to be printed

11.2.3.11 canpredict_compounds()

```
def cando.CANDO.canpredict_compounds (
            self,
            ind_id,
            n = 10,
            topX = 10,
            keep_associated = False,
            cmpd_set = 'all',
            save = '' )
```

This function is used for predicting putative therapeutics for an indication of interest using a homology-based approach.

Input an ind_id id and for each of the associated compounds, it will generate the similar compounds (based on distance) and add them to a dictionary with a value of how many times it shows up (enrichment). If a compound not approved for the indication of interest keeps showing up, that means it is similar in signature to the drugs that are ALREADY approved for the indication, so it may be a target for repurposing. Control how many similar compounds to consider with the argument 'n'. In the output, 'score1' refers to the number of times the compound shows up in the top 'n' drugs associated with the indication and 'score2' is the average of the ranks for 'score1' (note: 'score2' <= 'n').

Parameters

ind_id	str: Indication id
n	int: top number of similar Compounds to be used for each Compound associated with the given Indication
topX	int: top number of predicted Compounds to be printed
keep_associated	bool: Print Compounds that are already approved/associated for the Indication
cmpd_set	str: specify the compound set to use ('all', 'approved', or 'other')
save	str: name of a file to save results

11.2.3.12 canpredict_ddi_adrs()

```
def cando.CANDO.canpredict_ddi_adrs (
            self,
            cmpd_pair,
            n = 10,
            topX = 10,
            save = '' )
```

Parameters

cmpd_pair	Compound_pair: Compound_pair object to be used
n	int: top number of similar Compounds to be used for prediction
topX	int: top number of predicted Indications to be printed

11.2.3.13 canpredict_ddi_cmpds()

```
def cando.CANDO.canpredict_ddi_cmpds (
            self,
            cmpd,
            n = 10,
            topX = 10,
            save = '' )
```

Parameters

cmpd	Compound: Compound object to be used
n	int: top number of similar Compounds to be used for prediction
topX	int: top number of predicted Drug-drug Interactions to be printed

11.2.3.14 canpredict_denovo()

```
def cando.CANDO.canpredict_denovo (
            self,
            method = 'count',
            threshold = 0.0,
            topX = 10,
            ind_id = None,
            proteins = None,
            cmpd_set = 'all',
            save = '' )
```

This function is used for predicting putative therapeutics for an indication of interest by summing/-counting the number of interactions above a certain input interaction threshold for all proteins or a specified subset of proteins.

An indication can be specified to mark drugs associated with that indication in the output. The threshold will vary based on the values of the input matrix. Method can either be 'count' (score1), which ranks compounds based on the number of interactions above the threshold, or 'sum' (score2), which ranks the compounds based on the highest total sum for interaction scores above the threshold (these two are highly correlated but can differ for larger sets of proteins or lower thresholds). A third option is 'targets', which inspects and outputs the top protein interactions on an individual basis without summing/counting per drug (the output format differs from the other two options). If indication_proteins flag is used for the CANDO object instantiation, the proteins associated with the input indication will automatically be used. Otherwise, the 'proteins=' input can be used. The output can be saved to a file specified by 'save='. If ind_id is used, compounds associated with the indication will be included and marked in the output for comparison.

Parameters

method	str: 'sum', 'count', or 'targets'
threshold	float: a interaction score cutoff to use (ignores values for sum/count less than threshold)

Parameters

topX	int: top number of predicted Compounds to be printed/saved
ind_id	str: an indication id for marking drug output/ specifying protein set
proteins	List str: list of protein IDs from the matrix to use for the sum/count
cmpd_set	str: specify the compound set to use ('all', 'approved', or 'other')
save	str: name of a file to save results

11.2.3.15 canpredict_indications()

```
def cando.CANDO.canpredict_indications (
            self,
            cmpd,
            n = 10,
            topX = 10,
            save = '' )
```

This function is the inverse of canpredict_compounds.

Input a compound of interest cando_cmpd (or a novel protein signature of interest new_sig) and the most similar compounds to it will be computed. The indications associated with the top n most similar compounds to the query compound will be examined to see if any are repeatedly enriched.

Parameters

cmpd	Compound: Compound object to be used
n	int: top number of similar Compounds to be used for prediction
topX	int: top number of predicted Indications to be printed

11.2.3.16 common_targets()

```
def cando.CANDO.common_targets (
            self,
            cmpds_file,
            n = 10,
            negative = False,
            save_file = '' )
```

Get the top scoring protein targets for a given compound.

Parameters

cmpds_file	str: list of Compound IDs for which to search common targets
n	int: number of top targets to print/return
negative	int: if the interaction scores are negative (stronger) energies
save_file	str: save results to file name

Returns

Returns list: list of tuples (protein id_, score)

11.2.3.17 compounds_analysed()

```
def cando.CANDO.compounds_analysed (
            self,
            f,
            metrics )
```

11.2.3.18 fusion()

```
def cando.CANDO.fusion (
            self,
            cando_objs,
            out_file = '',
            method = 'sum' )
```

This function re-ranks the compounds according to the desired comparison specified by 'method' -> currently supports 'min', 'avg', 'mult', and 'sum'.

Parameters

cando_objs	list: List of CANDO objects
out_file	str: Path to where the result will be written
method	str: Method of fusion to be used (e.g., sum, mult, etc.)

11.2.3.19 generate_similar_sigs()

```
def cando.CANDO.generate_similar_sigs (
            self,
            cmpd,
            sort = False,
            proteins = [],
            aux = False )
```

For a given compound, generate the similar compounds using distance of sigs.

Parameters

cmpd	object: Compound object
sort	bool: Sort the list of similar compounds
proteins	list: Protein objects to identify a subset of the Compound signature
aux	bool: Use an auxiliary signature (default: False)

Returns

Returns list: Similar Compounds to the given Compound

11.2.3.20 generate_similar_sigs_cp()

```
def cando.CANDO.generate_similar_sigs_cp (
            self,
            cmpd_pair,
            sort = False,
            proteins = [],
            aux = False )
```

For a given compound pair, generate the similar compound pairs using distance of sigs.

Parameters

cmpd_pair	object: Compound_pair object
sort	bool: Sort the list of similar compounds
proteins	list: Protein objects to identify a subset of the Compound signature
aux	bool: Use an auxiliary signature (default: False)

Returns

Returns list: Similar Compounds to the given Compound

11.2.3.21 generate_some_similar_sigs()

```
def cando.CANDO.generate_some_similar_sigs (
            self,
            cmpds,
            sort = False,
            proteins = [],
            aux = False )
```

For a given list of compounds, generate the similar compounds based on dist of sigs This is pathways/genes for all intents and purposes.

Parameters

cmpds	list: Compound objects
sort	bool: Sort similar compounds for each Compound
proteins	list: Protein objects to identify a subset of the Compound signature
aux	bool: Use an auxiliary signature (default: False)

Returns

Returns list: Similar Compounds to the given Compound

11.2.3.22 get_adr()

```
def cando.CANDO.get_adr (
            self,
            id_ )
```

Get ADR (adverse drug reaction) from ADR id.

Parameters

id↩ _↩	str: ADR id

Returns

Returns object: ADR object

11.2.3.23 get_compound()

```
def cando.CANDO.get_compound (
            self,
            cmpd_id )
```

Get Compound object from Compound id or fuzzy match to Compound name.

Parameters

cmpd↩_id	int or str: Compound id or Compound name

Returns

Returns object: Compound object or None if no exact match is found

11.2.3.24 get_compound_pair()

```
def cando.CANDO.get_compound_pair (
            self,
            ids )
```

Get Compound object from Compound id.

Parameters

id↩_↩	int: Compound id

Returns

Returns object: Compound object

11.2.3.25 get_indication()

```
def cando.CANDO.get_indication (
            self,
            ind_id )
```

Get Indication object from Indication id.

Parameters

ind←↩ _id	str: Indication id

Returns

Returns object: Indication object

11.2.3.26 get_pathway()

```
def cando.CANDO.get_pathway (
            self,
            id_ )
```

Get Pathway object from Pathway id.

Parameters

id←↩ _←↩	str: Pathway id

Returns

Returns object: Pathway object

11.2.3.27 get_protein()

```
def cando.CANDO.get_protein (
            self,
            protein_id )
```

Get Protein object from Protein id.

Parameters

protein←↩ _id	str: Protein name

Returns

Returns object: Protein object

11.2.3.28 ml()

```
def cando.CANDO.ml (
            self,
            method = 'rf',
            effect = None,
            benchmark = False,
            adrs = False,
            predict = [],
            threshold = 0.5,
            negative = 'random',
            seed = 42,
            out = '' )
```

create an ML classifier for a specified indication or all inds (to benchmark) predict (used w/ 'effect=' - indication or ADR) is a list of compounds to classify with the trained ML model out=X saves benchmark SUMMARY->SUMMARY_ml_X; raw results->raw_results/raw_results_ml↩_X (same for RAN) currently supports random forest ('rf'), support vector machine ('svm'), 1-class SVM ('1csvm'), and logistic regression ('log') models are trained with leave-one-out cross validation during benchmarking

Parameters

method	str: type of machine learning algorithm to use ('rf', 'svm', '1csvm', and 'log')
effect	Indication or ADR: provide a specific Indication or ADR object to train a classifer
benchmark	bool: benchmark the ML pipeline by training a classifier with LOOCV for each Indication or ADR
adrs	bool: if the models are trained with ADRs instead of Indications
predict	list: provide a list of Compound objects to classify with the model (only used in combination with effect=Indication/ADR object)
threshold	float: decision threshold for positive vs negative classification
negative	str: choose random negative samples (default) or 'inverse' for most opposite signatures
seed	int: choose a seed for reproducibility
out	str: file name extension for the output of benchmark (note: must have benchmark=True)

11.2.3.29 normalize()

```
def cando.CANDO.normalize (
            self )
```

Normalize the distance scores to between [0,1].

Simply divides all scores by the largest distance between any two compounds.

11.2.3.30 quantify_pathways()

```
def cando.CANDO.quantify_pathways (
            self,
            indication = None )
```

Uses the pathway quantifier defined in the CANDO instantiation to make a pathway signature for all pathways in the input file (NOTE: does not compute distances)

Parameters

indication	object: Indication object

11.2.3.31 raw_results_roc()

```
def cando.CANDO.raw_results_roc (
            self,
            rr_files,
            labels,
            save = 'roc-raw_results.pdf' )
```

11.2.3.32 results_analysed()

```
def cando.CANDO.results_analysed (
            self,
            f,
            metrics,
            effect_type )
```

Creates the results analysed named file for the benchmarking and computes final avg indication accuracies.

Parameters

f	str: File path for results analysed named
metrics	list: Cutoffs used for the benchmarking protocol
effect_type	str: Defines the effect as either an Indication (disease) or ADR (adverse reaction)

11.2.3.33 save_dists_to_file()

```
def cando.CANDO.save_dists_to_file (
            self,
            f )
```

Write calculated distances of all compounds to all compounds to file.

Parameters

f	File name to save distances

11.2.3.34 search_compound()

```
def cando.CANDO.search_compound (
            self,
            name,
            n = 5 )
```

Print closest Compound names/IDs for input search str.

Parameters

name	str: Compound name
n	int: Number of outputted compounds

Returns

Returns None

11.2.3.35 search_indication()

```
def cando.CANDO.search_indication (
            self,
            name,
            n = 5 )
```

Print closest MeSH IDs for Indication name.

Parameters

name	str: Indication name
n	int: Number of outputted indications

Returns

Returns None

11.2.3.36 sigs()

```
def cando.CANDO.sigs (
            self,
            rm )
```

Return a list of all signatures, rm is a list of compound ids you do not want in the list.

Parameters

rm	list: List of compound ids to remove from list of signatures

Returns

list: List of all signatures

11.2.3.37 similar_compounds()

```
def cando.CANDO.similar_compounds (
            self,
            cmpd,
            n = 10 )
```

Computes and prints the top n most similar compounds to an input Compound object cando_↩ cmpd or input novel signature new_sig.

Parameters

cmpd	Compound: Compound object
n	int: top number of similar Compounds to be used for prediction

11.2.3.38 top_targets()

```
def cando.CANDO.top_targets (
            self,
            cmpd,
            n = 10,
            negative = False,
            save_file = '' )
```

Get the top scoring protein targets for a given compound.

Parameters

cmpd	Compound or int: Compound object or int id_ for which to print targets
n	int: number of top targets to print/return
negative	int: if the interaction scores are negative (stronger) energies

Returns

Returns list: list of tuples (protein id_, score)

11.2.3.39 uniprot_set_index()

```
def cando.CANDO.uniprot_set_index (
            self,
            prots )
```

Gather proteins from input matrix that map to UniProt IDs from 'protein_set=' param.

Parameters

prots	list: UniProt IDs (str)

Returns

Returns list: Protein chains (str) matching input UniProt IDs

11.2.3.40 virtual_screen()

```
def cando.CANDO.virtual_screen (
            self,
            protein,
            n = 10,
            negative = False,
            compound_set = 'all',
            save_file = '' )
```

Get the top scoring protein targets for a given compound.

Parameters

protein	Protein int or str: Protein (object, int index, or str id_) of which to screen for top scores
n	int: number of top compounds to print/return
negative	int: if the interaction scores are negative (stronger) energies
compound_set	str: use all Compounds ('all') or only approved Compounds ('approved')
save_file	str: save results to file name

Returns

Returns list: list of tuples (compound id_, score)

11.2.4 Member Data Documentation

11.2.4.1 accuracies

```
cando.CANDO.accuracies
```

11.2.4.2 adr_ids

```
cando.CANDO.adr_ids
```

11.2.4.3 adr_map

`cando.CANDO.adr_map`

str: File path to ADR mapping file

11.2.4.4 adrs

`cando.CANDO.adrs`

11.2.4.5 c_map

`cando.CANDO.c_map`

str: File path to the compound mapping file (relative or absolute)

11.2.4.6 clusters

`cando.CANDO.clusters`

11.2.4.7 cmpd_set

`cando.CANDO.cmpd_set`

11.2.4.8 compound_ids

`cando.CANDO.compound_ids`

11.2.4.9 compound_pair_ids

`cando.CANDO.compound_pair_ids`

11.2.4.10 compound_pairs

`cando.CANDO.compound_pairs`

11.2.4.11 compounds

`cando.CANDO.compounds`

11.2.4.12 compute_distance

`cando.CANDO.compute_distance`

bool: Calculate the distance for each Compound against all other Compounds using chosen distance metric

11.2.4.13 data_name

`cando.CANDO.data_name`

11.2.4.14 ddi_adrs

`cando.CANDO.ddi_adrs`

11.2.4.15 ddi_compounds

`cando.CANDO.ddi_compounds`

11.2.4.16 dist_metric

`cando.CANDO.dist_metric`

str: Distance metric to be used for computing Compound-Compound distances

11.2.4.17 i_map

`cando.CANDO.i_map`

str: File path to the indication mapping file (relative or absolute)

11.2.4.18 indication_ids

`cando.CANDO.indication_ids`

11.2.4.19 indication_pathways

`cando.CANDO.indication_pathways`

str: File path to Indication-Pathway association file

11.2.4.20 indication_proteins

`cando.CANDO.indication_proteins`

str: File path to Indication-Protein association file

11.2.4.21 indications

`cando.CANDO.indications`

11.2.4.22 matrix

`cando.CANDO.matrix`

str: File path to the cando matrix file (relative or absolute)

11.2.4.23 ncpus

```
cando.CANDO.ncpus
```

int: Number of CPUs used for parallelization

11.2.4.24 pathway_quantifier

```
cando.CANDO.pathway_quantifier
```

str: Method used to quantify a all Pathways

11.2.4.25 pathways

```
cando.CANDO.pathways
```

str: File path to pathway file

11.2.4.26 protein_distance

```
cando.CANDO.protein_distance
```

11.2.4.27 protein_id_to_index

```
cando.CANDO.protein_id_to_index
```

11.2.4.28 protein_map

```
cando.CANDO.protein_map
```

str: File path to Protein metadata mapping file

11.2.4.29 protein_set

`cando.CANDO.protein_set`

str: File path to protein subset file (relative or absolute)

11.2.4.30 proteins

`cando.CANDO.proteins`

11.2.4.31 read_dists

`cando.CANDO.read_dists`

str: File path to pre-computed distance matrix

11.2.4.32 rm_cmpds

`cando.CANDO.rm_cmpds`

11.2.4.33 rm_compounds

`cando.CANDO.rm_compounds`

list: Compounds to remove from the CANDO object

11.2.4.34 rm_zeros

`cando.CANDO.rm_zeros`

bool: Remove Compounds with all-zero signatures from CANDO object

11.2.4.35 save_dists

```
cando.CANDO.save_dists
```

bool: Write the calculated distances to file after computation (set compute_distances=True)

11.2.4.36 short_matrix_path

```
cando.CANDO.short_matrix_path
```

11.2.4.37 short_protein_set

```
cando.CANDO.short_protein_set
```

11.2.4.38 short_read_dists

```
cando.CANDO.short_read_dists
```

11.2.4.39 similarity

```
cando.CANDO.similarity
```

bool: Use similarity instead of distance

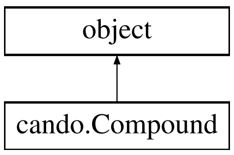
The documentation for this class was generated from the following file:

- cando.py

11.3 cando.Compound Class Reference

An object to represent a compound/drug.

Inheritance diagram for cando.Compound:



Public Member Functions

- def __init__ (self, name, id_, index, status='N/A')
- def add_indication (self, ind)

  Add an Indication to the list of Indications associated to this Compound.

Public Attributes

- name

  str: Name of the Compound (e.g., 'caffeine')
- id_

  int: CANDO id from mapping file (e.g., 1, 10, 100, ...)
- index

  int: The order in which the Compound appears in the mapping file (e.g, 1, 2, 3, ...)
- status

  str: The clinical trial status of the compound from DrugBank ('approved' or 'other')
- sig

  list: Signature is essentially a column of the Matrix
- aux_sig

  list: Potentially temporary signature for things like pathways, where "c.sig" needs to be preserved
- indications

  list: This is every indication the Compound is associated with from the mapping file
- similar

  list: This is the ranked list of compounds with the most similar interaction signatures
- similar_computed

  bool: Have the distances of all Compounds to the given Compound been computed?
- similar_sorted

  bool: Have the most similar Compounds to the given Compound been sorted?
- cluster_id

  int: The cluster id this Compound was assigned from clustering method
- adrs

  list: List of ADRs associated with this Compound
- alt_ids

  dict: dict of other ids inputted with compound mapping
- metabolites

  list: List of all metabolites from the compound
- is_metabolite

  bool: bool if the drug is a metabolite itself
- parent

  Compound: Compound object to which this compound is a metabolite.
- compounds

11.3.1 Detailed Description

An object to represent a compound/drug.

11.3.2 Constructor & Destructor Documentation

11.3.2.1 __init__()

```
def cando.Compound.__init__ (
            self,
            name,
            id_,
            index,
            status = 'N/A' )
```

11.3.3 Member Function Documentation

11.3.3.1 add_indication()

```
def cando.Compound.add_indication (
            self,
            ind )
```

Add an Indication to the list of Indications associated to this Compound.

Parameters

ind	object: Indication object to add

11.3.4 Member Data Documentation

11.3.4.1 adrs

```
cando.Compound.adrs
```

list: List of ADRs associated with this Compound

11.3.4.2 alt_ids

`cando.Compound.alt_ids`

dict: dict of other ids inputted with compound mapping

11.3.4.3 aux_sig

`cando.Compound.aux_sig`

list: Potentially temporary signature for things like pathways, where "c.sig" needs to be preserved

11.3.4.4 cluster_id

`cando.Compound.cluster_id`

int: The cluster id this Compound was assigned from clustering method

11.3.4.5 compounds

`cando.Compound.compounds`

11.3.4.6 id_

`cando.Compound.id_`

int: CANDO id from mapping file (e.g., 1, 10, 100, ...)

11.3.4.7 index

`cando.Compound.index`

int: The order in which the Compound appears in the mapping file (e.g, 1, 2, 3, ...)

11.3.4.8 indications

`cando.Compound.indications`

list: This is every indication the Compound is associated with from the mapping file

11.3.4.9 is_metabolite

`cando.Compound.is_metabolite`

bool: bool if the drug is a metabolite itself

11.3.4.10 metabolites

`cando.Compound.metabolites`

list: List of all metabolites from the compound

11.3.4.11 name

`cando.Compound.name`

str: Name of the Compound (e.g., 'caffeine')

11.3.4.12 parent

`cando.Compound.parent`

Compound: Compound object to which this compound is a metabolite.

11.3.4.13 sig

```
cando.Compound.sig
```

list: Signature is essentially a column of the Matrix

11.3.4.14 similar

```
cando.Compound.similar
```

list: This is the ranked list of compounds with the most similar interaction signatures

11.3.4.15 similar_computed

```
cando.Compound.similar_computed
```

bool: Have the distances of all Compounds to the given Compound been computed?

11.3.4.16 similar_sorted

```
cando.Compound.similar_sorted
```

bool: Have the most similar Compounds to the given Compound been sorted?

11.3.4.17 status

```
cando.Compound.status
```

str: The clinical trial status of the compound from DrugBank ('approved' or 'other')

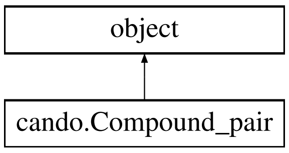
The documentation for this class was generated from the following file:

- cando.py

11.4 cando.Compound_pair Class Reference

An object to represent a compound/drug-pair.

Inheritance diagram for cando.Compound_pair:



Public Member Functions

- def __init__ (self, name, id_, index)
- def add_adr (self, adr)

  Add an ADR to the list of Indications associated to this Compound.

Public Attributes

- name

  str: Name of the Compound (e.g., 'caffeine')
- id_

  int: CANDO id from mapping file (e.g., 1, 10, 100, ...)
- index

  int: The order in which the Compound appears in the mapping file (e.g, 1, 2, 3, ...)
- sig

  list: Signature is essentially a column of the Matrix
- aux_sig

  list: Potentially temporary signature for things like pathways, where "c.sig" needs to be preserved
- similar

  list: This is the ranked list of compounds with the most similar interaction signatures
- similar_computed

  bool: Have the distances of all Compounds to the given Compound been computed?
- similar_sorted

  bool: Have the most similar Compounds to the given Compound been sorted?
- adrs

  list: List of ADRs associated with this Compound

11.4.1 Detailed Description

An object to represent a compound/drug-pair.

11.4.2 Constructor & Destructor Documentation

11.4.2.1 __init__()

```
def cando.Compound_pair.__init__ (
            self,
            name,
            id_,
            index )
```

11.4.3 Member Function Documentation

11.4.3.1 add_adr()

```
def cando.Compound_pair.add_adr (
            self,
            adr )
```

Add an ADR to the list of Indications associated to this Compound.

Parameters

ind	object: Indication object to add

11.4.4 Member Data Documentation

11.4.4.1 adrs

```
cando.Compound_pair.adrs
```

list: List of ADRs associated with this Compound

11.4.4.2 aux_sig

`cando.Compound_pair.aux_sig`

list: Potentially temporary signature for things like pathways, where "c.sig" needs to be preserved

11.4.4.3 id_

`cando.Compound_pair.id_`

int: CANDO id from mapping file (e.g., 1, 10, 100, ...)

11.4.4.4 index

`cando.Compound_pair.index`

int: The order in which the Compound appears in the mapping file (e.g, 1, 2, 3, ...)

11.4.4.5 name

`cando.Compound_pair.name`

str: Name of the Compound (e.g., 'caffeine')

11.4.4.6 sig

`cando.Compound_pair.sig`

list: Signature is essentially a column of the Matrix

11.4.4.7 similar

```
cando.Compound_pair.similar
```

list: This is the ranked list of compounds with the most similar interaction signatures

11.4.4.8 similar_computed

```
cando.Compound_pair.similar_computed
```

bool: Have the distances of all Compounds to the given Compound been computed?

11.4.4.9 similar_sorted

```
cando.Compound_pair.similar_sorted
```

bool: Have the most similar Compounds to the given Compound been sorted?

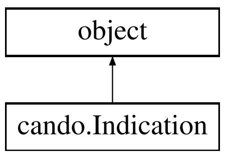
The documentation for this class was generated from the following file:

- cando.py

11.5 cando.Indication Class Reference

An object to represent an indication (disease)

Inheritance diagram for cando.Indication:



Public Member Functions

- def __init__ (self, ind_id, name)

Public Attributes

- id_

    str: MeSH or OMIM ID for the indication from the mapping file
- name

    str: Name for the indication from the mapping file
- compounds

    list: Every associated compound object from the mapping file
- pathways

    list: Every pathway associated to the indication from the mapping file
- proteins

    list: Every protein associated to the indication form the mapping file
- pathogen

    bool: Whether or not this indication is caused by a pathogen

11.5.1 Detailed Description

An object to represent an indication (disease)

11.5.2 Constructor & Destructor Documentation

11.5.2.1 __init__()

```
def cando.Indication.__init__ (
            self,
            ind_id,
            name )
```

11.5.3 Member Data Documentation

11.5.3.1 compounds

```
cando.Indication.compounds
```

list: Every associated compound object from the mapping file

11.5.3.2 id_

`cando.Indication.id_`

str: MeSH or OMIM ID for the indication from the mapping file

11.5.3.3 name

`cando.Indication.name`

str: Name for the indication from the mapping file

11.5.3.4 pathogen

`cando.Indication.pathogen`

bool: Whether or not this indication is caused by a pathogen

11.5.3.5 pathways

`cando.Indication.pathways`

list: Every pathway associated to the indication from the mapping file

11.5.3.6 proteins

`cando.Indication.proteins`

list: Every protein associated to the indication form the mapping file

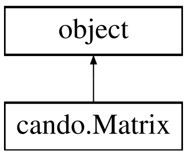
The documentation for this class was generated from the following file:

- cando.py

11.6 cando.Matrix Class Reference

An object to represent a matrix.

Inheritance diagram for cando.Matrix:



Public Member Functions

- def __init__ (self, matrix_file, dist=False, convert_to_tsv=False)
- def convert (self, out_file)

  Convert similarity matrix to distance matrix or vice versa.
- def normalize (self, outfile, dimension='drugs', method='avg')

  Normalize the interaction scores across drugs (default) or proteins (not implemented yet).

Public Attributes

- matrix_file

  str: Path to file with interaction scores
- dist

  bool: if the matrix_file is an dist file
- convert_to_tsv

  bool: Convert old matrix format (.fpt) to .tsv
- proteins

  list: Proteins in the Matrix
- values

  list: Values in the Matrix

11.6.1 Detailed Description

An object to represent a matrix.

Intended for easier handling of matrices. Convert between fpt and tsv, as well as distance to similarity (and vice versa)

11.6.2 Constructor & Destructor Documentation

11.6.2.1 __init__()

```
def cando.Matrix.__init__ (
            self,
            matrix_file,
            dist = False,
            convert_to_tsv = False )
```

11.6.3 Member Function Documentation

11.6.3.1 convert()

```
def cando.Matrix.convert (
            self,
            out_file )
```

Convert similarity matrix to distance matrix or vice versa.

The first value in the matrix will determine the type of conversion (0.0 means distance to similarity, 1.0 means similarity to distance).

Parameters

out_file	str: File path to which write the converted matrix.

11.6.3.2 normalize()

```
def cando.Matrix.normalize (
            self,
            outfile,
            dimension = 'drugs',
            method = 'avg' )
```

Normalize the interaction scores across drugs (default) or proteins (not implemented yet).

Parameters

outfile	str: File path to which is written the converted matrix.
dimension	str: which vector to normalize - either 'drugs' to normalize all scores within the proteomic vector or 'proteins' to normalize for a protein against all drug scores.
method	str: normalize by the average or max within the vectors

11.6.4 Member Data Documentation

11.6.4.1 convert_to_tsv

```
cando.Matrix.convert_to_tsv
```

bool: Convert old matrix format (.fpt) to .tsv

11.6.4.2 dist

```
cando.Matrix.dist
```

bool: if the matrix_file is an dist file

11.6.4.3 matrix_file

```
cando.Matrix.matrix_file
```

str: Path to file with interaction scores

11.6.4.4 proteins

```
cando.Matrix.proteins
```

list: Proteins in the Matrix

11.6.4.5 values

```
cando.Matrix.values
```

list: Values in the Matrix

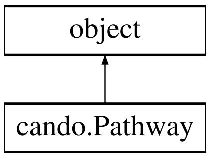
The documentation for this class was generated from the following file:

- cando.py

11.7 cando.Pathway Class Reference

An object to represent a pathway.

Inheritance diagram for cando.Pathway:



Public Member Functions

- def __init__ (self, id_)

Public Attributes

- proteins

  list: Protein objects associated with the given Pathway
- id_

  str: Identification for the given Pathway
- indications

  list: Indication objects associated with the given Pathway

11.7.1 Detailed Description

An object to represent a pathway.

11.7.2 Constructor & Destructor Documentation

11.7.2.1 __init__()

```
def cando.Pathway.__init__ (
            self,
            id_ )
```

11.7.3 Member Data Documentation

11.7.3.1 id_

```
cando.Pathway.id_
```

str: Identification for the given Pathway

11.7.3.2 indications

```
cando.Pathway.indications
```

list: Indication objects associated with the given Pathway

11.7.3.3 proteins

```
cando.Pathway.proteins
```

list: Protein objects associated with the given Pathway

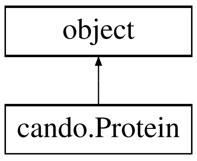
The documentation for this class was generated from the following file:

- cando.py

11.8 cando.Protein Class Reference

An object to represent a protein.

Inheritance diagram for cando.Protein:



Public Member Functions

- def __init__ (self, id_, sig)

Public Attributes

- id_

  PDB or UniProt ID for the given protein.
- alt_id

  Used when a second identifier mapping is available (such as SIFTs project)
- sig

  List of scores representing each drug interaction with the given protein.
- pathways

  List of Pathway objects in which the given protein is involved.
- indications
- name
- gene

11.8.1 Detailed Description

An object to represent a protein.

11.8.2 Constructor & Destructor Documentation

11.8.2.1 __init__()

```
def cando.Protein.__init__ (
            self,
            id_,
            sig )
```

11.8.3 Member Data Documentation

11.8.3.1 alt_id

```
cando.Protein.alt_id
```

Used when a second identifier mapping is available (such as SIFTs project)

11.8.3.2 gene

```
cando.Protein.gene
```

11.8.3.3 id_

```
cando.Protein.id_
```

PDB or UniProt ID for the given protein.

11.8.3.4 indications

```
cando.Protein.indications
```

11.8.3.5 name

```
cando.Protein.name
```

11.8.3.6 pathways

`cando.Protein.pathways`

List of Pathway objects in which the given protein is involved.

11.8.3.7 sig

`cando.Protein.sig`

List of scores representing each drug interaction with the given protein.

The documentation for this class was generated from the following file:

- cando.py

12 File Documentation

12.1 AUTHORS.md File Reference

12.2 cando.py File Reference

Classes

- class cando.Protein

  An object to represent a protein.
- class cando.Compound

  An object to represent a compound/drug.
- class cando.Compound_pair

  An object to represent a compound/drug-pair.
- class cando.Indication

  An object to represent an indication (disease)
- class cando.Pathway

  An object to represent a pathway.
- class cando.ADR

  An object to represent an adverse reaction.
- class cando.CANDO

  An object to represent all aspects of CANDO (compounds, indications, matrix, etc.)
- class cando.Matrix

  An object to represent a matrix.

Namespaces

- cando

Functions

- def cando.generate_matrix (v="v2.2", fp="rd_ecfp4", vect="int", dist="dice", org="nrpdb", bs="coach", c_cutoff=0.0, p_cutoff=0.0, percentile_cutoff=0.0, i_score="P", out_file='', out_path=".", nr_ligs=True, lig_name=False, ncpus=1)
- def cando.calc_scores (c, c_fps, l_fps, p_dict, dist, pscore_cutoff=0.0, cscore_cutoff=0.0, percentile_cutoff=0.0, i_score='P', nr_ligs=[ ], lig_name=False)
- def cando.generate_signature (cmpd_file, fp="rd_ecfp4", vect="int", dist="dice", org="nrpdb", bs="coach", c_cutoff=0.0, p_cutoff=0.0, percentile_cutoff=0.0, i_score="P", out_file='', out_path=".", nr_ligs=True)
- def cando.add_cmpds (cmpd_list, fp="rd_ecfp4", vect="int", cmpd_dir=".", v=None)
- def cando.cosine_dist (A)
- def cando.tanimoto_sparse (str1, str2)

  Calculate the tanimoto coefficient for a pair of sparse vectors.
- def cando.tanimoto_dense (list1, list2)

  Calculate the tanimoto coefficient for a pair of dense vectors.
- def cando.get_fp_lig (fp)

  Download precompiled binding site ligand fingerprints using the given fingerprint method.
- def cando.get_data (v="v2.2", org='nrpdb')

  Download CANDO v2.0 data.
- def cando.get_tutorial ()

  Download data for tutorial.
- def cando.get_test ()

  Download data for test script.
- def cando.dl_dir (url, out, l)

  Function to recursively download a directory.
- def cando.dl_file (url, out_file)

  Function to download a file.

12.3 LICENSE.md File Reference

Index