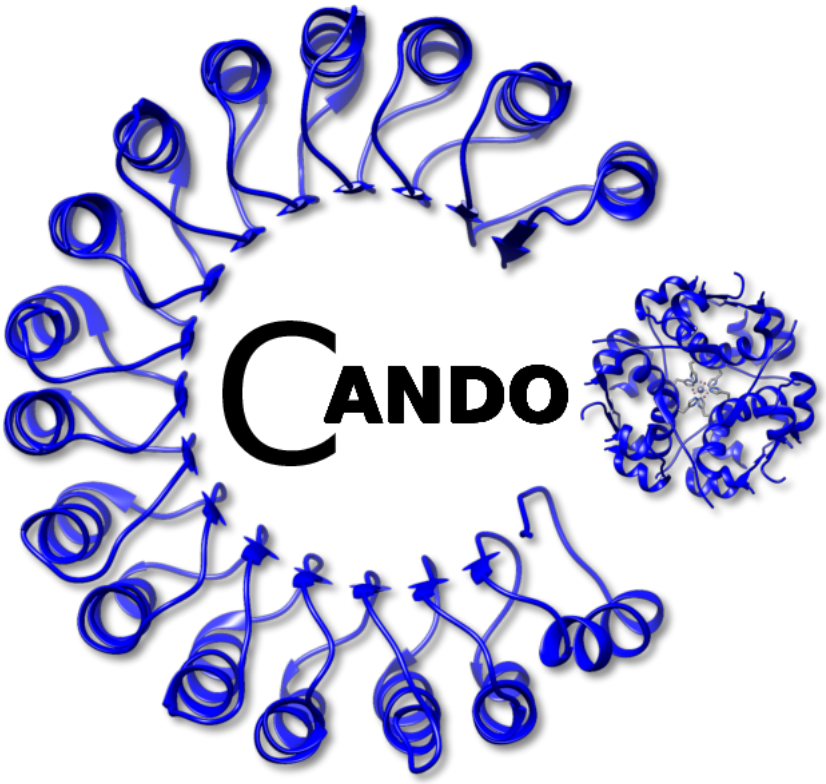
Version:
v2.0.0
Last Modified:
September 28, 2019



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:

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 Namespace Documentation

9.1 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 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 (cmpd_scores='', prot_scores='', matrix_file='cando_interaction_↩ matrix.tsv', ncpus=1)

  Generate a CANDO Matrix.

- def generate_scores (fp="rd_ecfp4", cmpd_pdb='', out_path='.')

  Generate the fingerprint for a new compound and calculate the Tanimoto similarities against all binding site ligands.

- def generate_signature (cmpd_scores='', prot_scores='', matrix_file='')

  Generate signature.

- def get_scores (c, p_scores, c_score)

  Get best score for each Compound-Protein interaction.

- def score_fp (fp, cmpd_file, cmpd_id, bs)

  Generate the scores for a given Compound against all Protein ligands.

- 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_v2_0 ()

  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.

9.1.1 Function Documentation

9.1.1.1 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

9.1.1.2 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

9.1.1.3 generate_matrix()

```
def cando.generate_matrix (
            cmpd_scores = '',
            prot_scores = '',
            matrix_file = 'cando_interaction_matrix.tsv',
            ncpus = 1 )
```

Generate a CANDO Matrix.

Parameters

cmpd_scores	str: File path to tab-separated scores for all Compounds
prot_scores	str: File path to tab-separated scores for all Proteins
matrix_file	str: File path to where the generated Matrix will be written
ncpus	int: Number of cpus to use for parallelization

9.1.1.4 generate_scores()

```
def cando.generate_scores (
            fp = "rd_ecfp4",
            cmpd_pdb = '',
            out_path = '.'  )
```

Generate the fingerprint for a new compound and calculate the Tanimoto similarities against all binding site ligands.

Parameters

fp	str: The fingerprinting software and method used, e.g. 'rd_ecfp4', 'ob_fp2'
cmpd_pdb	str: File path to the PDB
out_path	str: Path to where the scores file will be written

9.1.1.5 generate_signature()

```
def cando.generate_signature (
            cmpd_scores = '',
            prot_scores = '',
            matrix_file = '' )
```

Generate signature.

Parameters

cmpd_scores	str: File path to tab-separated scores for all Compounds
prot_scores	str: File path to tab-separated scores for all Proteins
matrix_file	str: File path to where the generated Compounds signature will be written

9.1.1.6 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

9.1.1.7 get_scores()

```
def cando.get_scores (
            c,
            p_scores,
            c_score )
```

Get best score for each Compound-Protein interaction.

Parameters

c	int: Compound id
p_scores	df: DataFrame of all Protein ligands
c_score	df: DataFrame of all Compound-ligand scores

9.1.1.8 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

9.1.1.9 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

9.1.1.10 get_v2_0()

```
def cando.get_v2_0 ( )
```

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)

9.1.1.11 score_fp()

```
def cando.score_fp (
            fp,
            cmpd_file,
            cmpd_id,
            bs )
```

Generate the scores for a given Compound against all Protein ligands.

Parameters

fp	str: Fingerprinting software and method used, e.g., rd_ecfp4
cmpd_file	str: File path to PDB
cmpd_id	int: Number correspodning to the new Compound id
bs	df: DataFrame of all protein ligand fingerprints for the given fingerprinting method (fp)

9.1.1.12 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

9.1.1.13 tanimoto_sparse()

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

Calculate the tanimoto coefficient for a pair of sparse vectors.

Parameters

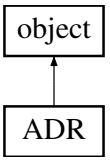
str1	str: String of 1s and 0s representing the first compound fingerprint
str2	str: String of 1s and 0s representing the second compound fingerprint

10 Class Documentation

10.1 ADR Class Reference

An object to represent an adverse reaction.

Inheritance diagram for 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

10.1.1 Detailed Description

An object to represent an adverse reaction.

10.1.2 Constructor & Destructor Documentation

10.1.2.1 __init__()

```
def __init__ (
            self,
            id_,
            name )
```

10.1.3 Member Data Documentation

10.1.3.1 compounds

```
compounds
```

list: Compound objects associated with the given ADR

10.1.3.2 id_

```
id_
```

str: Identification for the given ADR

10.1.3.3 name

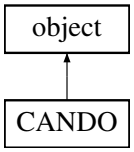
```
name
```

str: Name of the given ADR

10.2 CANDO Class Reference

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

Inheritance diagram for CANDO:



Public Member Functions

- def __init__ (self, c_map, i_map, matrix='', compute_distance=False, save_rmsds='', read_rmsds='', pathways='', pathway_quantifier='max', indication_pathways='', indication_proteins='', similarity=False, dist_metric='rmsd', protein_set='', rm_zeros=False, rm_compounds='', adr_map='', ncpus=1)
- def get_compound (self, id_)

  Get Compound object from Compound 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 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_some_similar_sigs (self, cmpds, sort=False, proteins=[ ], aux=False)

  For a given list of compounds, generate the similar compounds based on rmsd 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_cluster (self, n_clusters=5)

  Benchmark using k-means clustering.

- def ml (self, method='rf', effect=None, benchmark=False, adrs=False, predict=[ ], 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 canpredict_compounds (self, ind_id, n=10, topX=10, sum_scores=False, keep_approved=False)

  This function is used for predicting putative therapeutics for an indication of interest.

- def canpredict_indications (self, new_sig=None, new_name=None, cando_cmpd=None, n=10, topX=10)

  This function is the inverse of canpredict_compounds.

- def similar_compounds (self, new_sig=None, new_name=None, cando_cmpd=None, 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_rmsds_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_rmsds

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

  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: Numebr 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
- proteins

- protein_id_to_index
- compounds
- indications
- indication_ids
- adrs
- short_matrix_path
- short_read_rmsds
- short_protein_set
- cmpd_set
- data_name

10.2.1 Detailed Description

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

To instantiate you need the compound mapping (c_map), indication mapping file (i_map), and a compound-protein matrix (matrix=) or or precomputed compound-compound distance matrix (read_rmsds=)

10.2.2 Constructor & Destructor Documentation

10.2.2.1 __init__()

```
def __init__ (
            self,
            c_map,
            i_map,
            matrix = '',
            compute_distance = False,
            save_rmsds = '',
            read_rmsds = '',
            pathways = '',
            pathway_quantifier = 'max',
            indication_pathways = '',
            indication_proteins = '',
            similarity = False,
            dist_metric = 'rmsd',
            protein_set = '',
            rm_zeros = False,
            rm_compounds = '',
            adr_map = '',
            ncpus = 1 )
```

10.2.3 Member Function Documentation

10.2.3.1 __str__()

```
def __str__ (
            self )
```

Print stats about the CANDO object.

10.2.3.2 add_cmpd()

```
def 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

10.2.3.3 canbenchmark()

```
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.

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

10.2.3.4 canbenchmark_associated()

```
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.

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)

10.2.3.5 canbenchmark_bottom()

```
def 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)

10.2.3.6 canbenchmark_cluster()

```
def canbenchmark_cluster (
            self,
            n_clusters = 5 )
```

Benchmark using k-means clustering.

Parameters

n_clusters	int: Number of clusters for k-means

10.2.3.7 canpredict_compounds()

```
def canpredict_compounds (
            self,
            ind_id,
            n = 10,
            topX = 10,
            sum_scores = False,
            keep_approved = False )
```

This function is used for predicting putative therapeutics for an indication of interest.

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'. Use ind_id=None to find greatest score sum across all proteins (sum_scores must be True)

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
sum_scores	bool: Sum all ascores across all proteins
keep_approved	bool: Print Compounds that are already approved for the Indication

10.2.3.8 canpredict_indications()

```
def canpredict_indications (
            self,
            new_sig = None,
            new_name = None,
            cando_cmpd = None,
            n = 10,
            topX = 10 )
```

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

new_sig	str: Path to the new Compound signature
new_name	str: Name to be used for the new Compound
cando_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

10.2.3.9 fusion()

```
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'.

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.)

10.2.3.10 generate_similar_sigs()

```
def 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

10.2.3.11 generate_some_similar_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 rmsd 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

10.2.3.12 get_adr()

```
def get_adr (
            self,
            id_ )
```

Get ADR (adverse drug reaction) from ADR id.

Parameters

*id_*	str: ADR id

Returns

Returns object: ADR object

10.2.3.13 get_compound()

```
def get_compound (
            self,
            id_ )
```

Get Compound object from Compound id.

Parameters

*id_*	int: Compound id

Returns

Returns object: Compound object

10.2.3.14 get_indication()

```
def get_indication (
            self,
            ind_id )
```

Get Indication object from Indication id.

Parameters

ind_id	str: Indication id

Returns

Returns object: Indication object

10.2.3.15 get_pathway()

```
def get_pathway (
            self,
            id_ )
```

Get Pathway object from Pathway id.

Parameters

id↩ _↩	str: Pathway id

Returns

Returns object: Pathway object

10.2.3.16 ml()

```
def ml (
            self,
            method = 'rf',
            effect = None,
            benchmark = False,
            adrs = False,
            predict = [],
            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	list: 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)
seed	int: choose a seed for reproducibility
out	str: file name extension for the output of benchmark (note: must have benchmark=True)

10.2.3.17 normalize()

```
def normalize (
            self )
```

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

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

10.2.3.18 quantify_pathways()

```
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)

Parameters

indication	object: Indication object

10.2.3.19 results_analysed()

```
def 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)

10.2.3.20 save_rmsds_to_file()

```
def save_rmsds_to_file (
            self,
            f )
```

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

Parameters

f	File name to save distances

10.2.3.21 sigs()

```
def 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

10.2.3.22 similar_compounds()

```
def similar_compounds (
            self,
            new_sig = None,
            new_name = None,
            cando_cmpd = None,
            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

new_sig	list: List float of novel compound protein interaction signature
new_name	str: Drug name
cando_cmpd	Compound: Compound object
n	int: top number of similar Compounds to be used for prediction

10.2.3.23 uniprot_set_index()

```
def 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

10.2.4 Member Data Documentation

10.2.4.1 accuracies

```
accuracies
```

10.2.4.2 adr_map

```
adr_map
```

str: File path to ADR mapping file

10.2.4.3 adrs

`adrs`

10.2.4.4 c_map

`c_map`

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

10.2.4.5 clusters

`clusters`

10.2.4.6 cmpd_set

`cmpd_set`

10.2.4.7 compounds

`compounds`

10.2.4.8 compute_distance

`compute_distance`

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

10.2.4.9 data_name

`data_name`

10.2.4.10 dist_metric

`dist_metric`

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

10.2.4.11 i_map

`i_map`

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

10.2.4.12 indication_ids

`indication_ids`

10.2.4.13 indication_pathways

`indication_pathways`

str: File path to Indication-Pathway association file

10.2.4.14 indication_proteins

`indication_proteins`

str: File path to Indication-Protein association file

10.2.4.15 indications

`indications`

10.2.4.16 matrix

`matrix`

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

10.2.4.17 ncpus

`ncpus`

int: Numebr of CPUs used for parallelization

10.2.4.18 pathway_quantifier

`pathway_quantifier`

str: Method used to quantify a all Pathways

10.2.4.19 pathways

`pathways`

str: File path to pathway file

10.2.4.20 protein_id_to_index

`protein_id_to_index`

10.2.4.21 protein_set

`protein_set`

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

10.2.4.22 proteins

`proteins`

10.2.4.23 read_rmsds

`read_rmsds`

str: File path to pre-computed distance matrix

10.2.4.24 rm_cmpds

`rm_cmpds`

10.2.4.25 rm_compounds

`rm_compounds`

list: Compounds to remove from the CANDO object

10.2.4.26 rm_zeros

`rm_zeros`

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

10.2.4.27 save_rmsds

`save_rmsds`

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

10.2.4.28 short_matrix_path

`short_matrix_path`

10.2.4.29 short_protein_set

`short_protein_set`

10.2.4.30 short_read_rmsds

`short_read_rmsds`

10.2.4.31 similarity

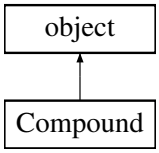
`similarity`

bool: Use similarity instead of distance

10.3 Compound Class Reference

An object to represent a compound/drug.

Inheritance diagram for Compound:



Public Member Functions

- def __init__ (self, name, id_, index)
- 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, ...)
- 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

10.3.1 Detailed Description

An object to represent a compound/drug.

10.3.2 Constructor & Destructor Documentation

10.3.2.1 __init__()

```
def __init__ (
            self,
            name,
            id_,
            index )
```

10.3.3 Member Function Documentation

10.3.3.1 add_indication()

```
def add_indication (
            self,
            ind )
```

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

Parameters

ind	object: Indication object to add

10.3.4 Member Data Documentation

10.3.4.1 adrs

```
adrs
```

list: List of ADRs associated with this Compound

10.3.4.2 aux_sig

```
aux_sig
```

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

10.3.4.3 cluster_id

`cluster_id`

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

10.3.4.4 id_

`id_`

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

10.3.4.5 index

`index`

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

10.3.4.6 indications

`indications`

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

10.3.4.7 name

`name`

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

10.3.4.8 sig

`sig`

list: Signature is essentially a column of the Matrix

10.3.4.9 similar

`similar`

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

10.3.4.10 similar_computed

`similar_computed`

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

10.3.4.11 similar_sorted

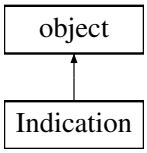
`similar_sorted`

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

10.4 Indication Class Reference

An object to represent an indication (disease)

Inheritance diagram for 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

10.4.1 Detailed Description

An object to represent an indication (disease)

10.4.2 Constructor & Destructor Documentation

10.4.2.1 __init__()

```
def __init__ (
            self,
            ind_id,
            name )
```

10.4.3 Member Data Documentation

10.4.3.1 compounds

```
compounds
```

list: Every associated compound object from the mapping file

10.4.3.2 id_

`id_`

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

10.4.3.3 name

`name`

str: Name for the indication from the mapping file

10.4.3.4 pathways

`pathways`

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

10.4.3.5 proteins

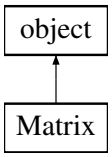
`proteins`

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

10.5 Matrix Class Reference

An object to represent a matrix.

Inheritance diagram for Matrix:



Public Member Functions

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

  Convert similarity matrix to distance matrix or vice versa.

Public Attributes

- matrix_file

  str: Path to file with interaction scores
- rmsd

  bool: if the matrix_file is an rmsd file
- convert_to_tsv

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

  list: Proteins in the Matrix
- values

  list: Values in the Matrix

10.5.1 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)

10.5.2 Constructor & Destructor Documentation

10.5.2.1 __init__()

```
def __init__ (
            self,
            matrix_file,
            rmsd = False,
            convert_to_tsv = False )
```

10.5.3 Member Function Documentation

10.5.3.1 convert()

```
def 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.

10.5.4 Member Data Documentation

10.5.4.1 convert_to_tsv

```
convert_to_tsv
```

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

10.5.4.2 matrix_file

```
matrix_file
```

str: Path to file with interaction scores

10.5.4.3 proteins

```
proteins
```

list: Proteins in the Matrix

10.5.4.4 rmsd

`rmsd`

bool: if the matrix_file is an rmsd file

10.5.4.5 values

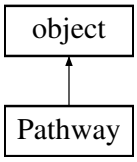
`values`

list: Values in the Matrix

10.6 Pathway Class Reference

An object to represent a pathway.

Inheritance diagram for 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

10.6.1 Detailed Description

An object to represent a pathway.

10.6.2 Constructor & Destructor Documentation

10.6.2.1 __init__()

```
def __init__ (
            self,
            id_ )
```

10.6.3 Member Data Documentation

10.6.3.1 id_

`id_`

str: Identification for the given Pathway

10.6.3.2 indications

`indications`

list: Indication objects associated with the given Pathway

10.6.3.3 proteins

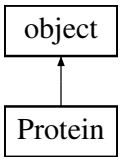
`proteins`

list: Protein objects associated with the given Pathway

10.7 Protein Class Reference

An object to represent a protein.

Inheritance diagram for Protein:



Public Member Functions

- def __init__ (self, id_, sig)

Public Attributes

- id_

  PDB or UniProt ID for the given protein.
- sig

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

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

10.7.1 Detailed Description

An object to represent a protein.

10.7.2 Constructor & Destructor Documentation

10.7.2.1 __init__()

```
def __init__ (
            self,
            id_,
            sig )
```

10.7.3 Member Data Documentation

10.7.3.1 id_

`id_`

PDB or UniProt ID for the given protein.

10.7.3.2 pathways

`pathways`

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

10.7.3.3 sig

`sig`

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