compound_target_pairs_dataset

Release 0.0.1

Lina Heinzke, Barbara Zdrazil

Feb 13, 2024

CONTENTS:

CHAPTER

ONE

INTRODUCTION

This code extract a dataset of compound-target pairs from the open-source bioactivity database ChEMBL [Zdrazil2023].

The compound-target pairs are known to interact because

- they have at least one corresponding measured activity values in ChEMBL or
- they are part of a set of manually curated known interactions in ChEMBL.

Furthermore, the dataset contains a number of compounds and target annotations to enable future analyses.

Previously, a similar dataset has been curated manually and has been used to investigate target-based differences in drug-like properties and ligand efficiencies [Leeson2021]. This code can generate an extended version of the previous dataset for every ChEMBL version from ChEMBL 26 onwards.

1.1 Dataset Documentation

If you are interested in understanding the fields in the resulting dataset, see *Columns in the Final Dataset*

1.2 User Guide

If you are interested in using the code, see *User Guide*

1.3 Code Documentation

If you are interested in understanding the code, see *src*

CHAPTER

TWO

COLUMNS IN THE FINAL DATASET

This page provides explanations for all columns available in the final dataset.

More information on ChEMBL-based columns can be found in the respective ChEMBL schema documentation. The information on this page mostly corresponds to the ChEMBL 32 schema documentation.

2.1 Initial Query

2.1.1 PChEMBL Values

The pchembl_value is later aggregated into mean, max and median per compound-target pair and dropped.

Column Name	Type	Info Re.	Based On	Description / Notes
pchembl_value	Float	Compound-Target Pair	ChEMBL: activities	Negative log of selected concentration-response activity values (IC50 / EC50 / XC50 / AC50 / Ki / Kd / Potency).

2.1.2 Compound Information

Column Name	Type	Info Re.	Based On	Description / Notes
parent_molregno	Int	Compound	ChEMBL: molecule_dictionary	Internal Primary Key for the molecule
parent_chemblid	String	Compound	"	ChEMBL identifier for this compound (for use on web interface etc)
parent_pref_name	String	Compound	"	Preferred name for the molecule
max_phase	Float	Compound	"	Maximum phase of development reached for the compound across all indications[1]
first_approval	Int	Compound	"	Earliest known approval year for the drug (NULL is the default value)
usan_year	Int	Compound	"	The year in which the application for a USAN/INN name was granted. (NULL is the default value)
black_box_warning	Int	Compound	"	Indicates that the drug has a black box warning (1 = yes, 0 = default value)
prodrug	Int	Compound	"	Indicates that the drug is a pro-drug (1 = yes, 0 = no, -1 = preclinical compound ie not a drug)
oral	Int	Compound	"	Indicates whether the drug is known to be administered orally (1 = yes, 0 = default value)
parenteral	Int	Compound	"	Indicates whether the drug is known to be administered parenterally (1 = yes, 0 = default value)
topical	Int	Compound	"	Indicates whether the drug is known to be administered topically (1 = yes, 0 = default value)

2.1.3 Target Information

Column Name	Type	Info Re.	Based On	Description / Notes
tid	Int	Target	ChEMBL: assays	Unique ID for the target
mutation	String	Target	ChEMBL: variant_sequences	Details of variant(s) used, with residue positions adjusted to match provided sequence.
target_chembl_id	String	Target	ChEMBL: target_dictionary	ChEMBL identifier for this target (for use on web interface etc)
target_pref_name	String	Target	"	Preferred target name: manually curated
target_type	String	Target	"	Describes whether target is a protein, an organism, a tissue etc.
organism	String	Target	"	Source organism of molecuar target or tissue, or the target organism if compound activity is reported in an organism rather than a protein or tissue

[1] There have been changes to the max_phase field in ChEMBL with version 32. See *MAX_PHASE in ChEMBL*.

2.1.4 Helper Columns

These columns are combination of other columns, used for easier processing of the dataset.

Column Name	Type	Info Re.	Based On	Description / Notes
tid_mutation	String	Target	tid + '_' + mutation	Helper column
cpd_target_pair	String	Compound-Target Pair	parent_molregno + '_' + tid	Helper column
cpd_target_pair_mutation	String	Compound-Target Pair	parent_molregno + '_' + tid_mutation	Helper column

2.2 Aggregated Values

Aggregated per compound-target pair using parent_molregno and tid_mutation.

Column Name	Type	Info Re.	Description / Notes
pchembl_value_mean_BF / _B	Float	Compound-Target Pair	Mean pchemb_value for the compound-target pair
pchembl_value_max_BF / _B	Float	Compound-Target Pair	Maximum pchemb_value for the compound-target pair
pchembl_value_median_BF / _B	Float	Compound-Target Pair	Median pchemb_value for the compound-target pair
first_publication_cpd_target_pair_BF / _B	Int	Compound-Target Pair	First publication in ChEMBL with this compound-target pair
first_publication_cpd_target_pair_w_pchembl_BF / _B	Int	Compound-Target Pair	First publication in ChEMBL with this compound-target pair and an associated pchembl value

2.2.1 Naming Convention: B vs. BF

These values are aggregated based on different subsets of the full dataset. The corresponding columns in the final dataset have a suffix that corresponds to the assay types the value is based on:

- _BF: based on binding + functional assays
- _B: based on binding assays

2.3 DTI (Drug-Target Interaction) Annotations

Based on cpd_target_pair, does not include mutation information.

Column Name	Type	Info Re.	Based On	Description / Notes
therapeutic_target	Bool	Target	ChEMBL: drug_mechanism table	Is the target in the drug mechanism table?
DTI	String	Compound-Target Pair	Assigned as below	Drug target interaction (DTI) annotation

2.3.1 Mechanism to Assign DTI

In DM Table?[2]	max_phase?[3]	Th. Target?[4]	DTI	Explanation
Yes	4	–	D_DT	Drug - drug target
Yes	3	–	C3_DT	Clinical candidate in phase 3 - drug target
Yes	2	–	C2_DT	Clinical candidate in phase 2 - drug target
Yes	1	–	C1_DT	Clinical candidate in phase 1 - drug target
Yes	< 1	–	C0_DT	Compound in unknown clinical phase[5] - drug target
No	–	Yes	DT	Drug target
No	–	No	NDT	Not drug target

2.3.2 MAX_PHASE in ChEMBL

Before ChEMBL 32, compounds with a max_phase not between 1 and 4 were assigned a max_phase of 0.

From ChEMBL 32 onwards, compounds with a max_phase not between 1 and 4 can have three possible values:

- 0.5 = early phase 1 clinical trials
- -1 = clinical phase unknown for drug or clinical candidate drug, i.e., where ChEMBL cannot assign a clinical phase
- NULL = preclinical compounds with bioactivity data

2.4 Compound and Target Properties Based on ChEMBL Data

2.4.1 First publication

In contrast to the aggregated time-related fields, this field takes all of ChEMBL and not just the time-related data within the dataset into account.

Column Name	Type	Info Re.	Based On	Description / Notes
first_publication_cpd	Int	Compound	ChEMBL: docs	First appearance of the compound in the literature

[2] Is the compound-target pair in the drug_mechanisms table? = Is it a known relevant compound-target interaction?
[3] What is the max_phase of the compound? = Is it a drug / clinical compound?
[4] Is the target in the drug_mechanisms table? = Is it a therapeutic target?
[5] There have been changes to the max_phase field in ChEMBL with version 32. C0_DT groups together all compounds with a max_phase not between 1 and 4. See *MAX_PHASE in ChEMBL*

2.4.2 Compound Properties

Column Name	Type	Info Re.	Based On	Description / Notes
mw_freebase	Float	Compound	ChEMBL: compound_properties	Molecular weight of parent compound
alogp	Float	Compound	“	Calculated ALogP
hba	Int	Compound	“	Number hydrogen bond acceptors
hbd	Int	Compound	“	Number hydrogen bond donors
psa	Float	Compound	“	Polar surface area
rtb	Int	Compound	“	Number rotatable bonds
ro3_pass	String	Compound	“	Indicates whether the compound passes the rule-of-three (mw < 300, logP < 3 etc)
num_ro5_violations	Int	Compound	“	Number of violations of Lipinski’s rule-of-five, using HBA and HBD definitions
cx_most_apka	Float	Compound	“	The most acidic pKa calculated using ChemAxon
cx_most_bpka	Float	Compound	“	The most basic pKa calculated using ChemAxon
cx_logp	Float	Compound	“	The calculated octanol/water partition coefficient using ChemAxon
cx_logd	Float	Compound	“	The calculated octanol/water distribution coefficient at pH7.4 using ChemAxon
molecular_species	String	Compound	“	Indicates whether the compound is an acid/base/neutral
full_mwt	Float	Compound	“	Molecular weight of the full compound including any salts
aromatic_rings	Int	Compound	“	Number of aromatic rings
heavy_atoms	Int	Compound	“	Number of heavy (non-hydrogen) atoms
qed_weighted	Float	Compound	“	Weighted quantitative estimate of drug likeness (as defined by Bickerton et al., Nature Chem 2012)
mw_monoisotopic	Float	Compound	“	Monoisotopic parent molecular weight
full_molformula	String	Compound	“	Molecular formula for the full compound (including any salt)
hba_lipinski	Int	Compound	“	Number of hydrogen bond acceptors calculated according to Lipinski’s original rules (i.e., N + O count))
hbd_lipinski	Int	Compound	“	Number of hydrogen bond donors calculated according to Lipinski’s original rules (i.e., NH + OH count)
num_lipinski_ro5_violations	Int	Compound	“	Number of violations of Lipinski’s rule of five using HBA_LIPINSKI and HBD_LIPINSKI counts

2.4.3 Compound Structures

Column Name	Type	Info Re.	Based On	Description / Notes
standard_inchi	String	Compound	ChEMBL: compound_structures	IUPAC standard InChI for the compound
standard_inchi_key	String	Compound	“	IUPAC standard InChI key for the compound
canonical_smiles	String	Compound	“	Canonical smiles, generated using RDKit

2.4.4 ATC and Target Class

Column Name	Type	Info Re.	Based On	Description / Notes
atc_level1	String	Compound	ChEMBL: atc_classification, molecule_atc_ classification	Anatomical Therapeutic Chemical (ATC) classification, level 1
target_class_l1	String	Target	ChEMBL: protein_classification, protein_family_ classification	Target class, level 1 (more general)
target_class_l2	String	Target	“	Target class, level 2 (more detailed)

2.5 Ligand Efficiency Metrics

Calculated based on pchembl_value_mean.

Since LE metrics are based on pChEMBL values, they are calculated twice. Once for the pChEMBL values based on binding and functional assays (suffix _BF) and once for the pChEMBL values based on binding assays only (suffix _B).

Column Name	Type	Info Re.	Description / Notes
LE_BF / LE_B	Float	Compound	Ligand efficiency
BEI_BF / BEI_B	Float	Compound	Binding efficiency index
SEI_BF / SEI_B	Float	Compound	Surface efficiency index
LLE_BF / LLE_B	Float	Compound	Lipophilic ligand efficiency

2.5.1 Equations

$$
\begin{aligned}
LE &= \frac{2.303 \cdot 298 \cdot 0.00199 \cdot pchembl\_value}{heavy\_atoms} \\
BEI &= \frac{pchembl\_mean \cdot 1000}{mw\_freebase} \\
SEI &= \frac{pchembl\_mean \cdot 100}{PSA} \\
LLE &= pchembl\_mean - ALogP
\end{aligned}
$$

2.6 RDKit-Based Compound Descriptors

2.6.1 Built-in Methods

These compound descriptors are calculated using built-in RDKit methods from Descriptors and rdMolDescriptors.

Column Name	Type	Info Re.	Based On	Description / Notes
fraction_csp3	Float	Compound	canonical_smiles + built-in RDKit methods	Fraction of C atoms that are SP3 hybridized (rdkit.Chem.Descriptors. FractionCSP3)
ring_count	Int	Compound	“	(rdkit.Chem.Descriptors. RingCount)
num_aliphatic_ rings	Int	Compound	“	Number of aliphatic (containing at least one non-aromatic bond) rings (rdkit.Chem.Descriptors. NumAliphaticRings)
num_aliphatic_ carbocycles	Int	Compound	“	Number of aliphatic (containing at least one non-aromatic bond) carbocycles (rdkit.Chem.Descriptors. NumAliphaticCarbocycles)
num_aliphatic_ heterocycles	Int	Compound	“	Number of aliphatic (containing at least one non-aromatic bond) heterocycles (rdkit.Chem.Descriptors. NumAliphaticHeterocycles)
num_aromatic_ rings	Int	Compound	“	Number of aromatic rings (rdkit.Chem.Descriptors. NumAromaticRings)
num_aromatic_ carbocycles	Int	Compound	“	Number of aromatic carbocycles (rdkit.Chem.Descriptors. NumAromaticCarbocycles)
num_aromatic_ heterocycles	Int	Compound	“	Number of aromatic heterocycles (rdkit.Chem.Descriptors. NumAromaticHeterocycles)
num_saturated_ rings	Int	Compound	“	Number of saturated rings (rdkit.Chem.Descriptors. NumSaturatedRings)
num_saturated_ carbocycles	Int	Compound	“	Number of saturated carbocycles (rdkit.Chem.Descriptors. NumSaturatedCarbocycles)
num_saturated_ heterocycles	Int	Compound	“	Number of saturated heterocycles (rdkit.Chem.Descriptors. NumSaturatedHeterocycles)
num_stereocentres	Int	Compound	“	Number of atomic stereocenters (specified and unspecified) (rdkit.Chem.rdMolDescriptors. CalcNumAtomStereoCenters)
num_heteroatoms	Int	Compound	“	Number of heteroatoms (rdkit.Chem.Descriptors. NumHeteroatoms)

2.6.2 Bespoke Methods

These compound descriptors are calculated using custom RDKit-based methods.

Column Name	Type	Info Re.	Based On	Description / Notes
aromatic_atoms	Int	Compound	canonical_smiles + RDKit-based methods	Number of aromatic atoms
aromatic_c	Int	Compound	“	Number of aromatic C
aromatic_n	Int	Compound	“	Number of aromatic N
aromatic_hetero	Int	Compound	“	Number of aromatic hetero atoms
scaffold_ w_stereo	String	Compound	“	Scaffold SMILES, including stereochemistry information
scaffold_ wo_stereo	String	Compound	“	Scaffold SMILES of the molecule after removing stereochemistry information

2.7 Annotations for Filtering

Columns are only available for the full dataset to facilitate the filtering into subsets.

2.7.1 Helper Columns

pair_mutation_in_dm_table and pair_in_dm_table are similar fields. They differ in whether mutation information is taken into account, reflecting that mutation information is only sometimes taken into account when calculating fields and adding rows to the dataset.

- **pair_mutation_in_dm_table:**
  Is the compound-target pair in the drug_mechanism table when taking mutation information into account? Mutation information IS taken into account when adding pairs to the dataset because they appear in the drug_mechanism table. (cpd A, target B without mutation) will be added to the set of existing compound-target pairs with pChEMBL values if there is a pair with a pChEMBL value for (cpd A, target B with mutation C) but there is no pair with a pChEMBL value for (cpd A, target B without mutation). It is used to determine keep_for_binding which in turn is used to determine the B subset of data based on binding assays.
- **pair_in_dm_table:**
  Is the compound-target pair in the drug_mechanism table when ignoring mutation information? Mutation information is NOT taken into account when assigning DTI values.

Column Name	Type	Info Re.	Description / Notes
pair_mutation_in_dm_	Bool	Compound-Target Pair	Is the compound-target pair (taking mutation annotation into account) in the drug mechanism table?
pair_in_dm_table	Bool	Compound-Target Pair	Is the compound-target pair (ignoring mutation annotation) in the drug mechanism table?
keep_for_binding	Bool	Compound-Target Pair	Rows to keep if interested in information based only on binding assays + the drug_mechanism table. True if pchembl_value_mean_B (based on binding assays) exists or if pair_mutation_in_dm_table == True, i.e., the pair (including mutation information) is in the drug mechanism table.

2.7.2 Filtering Columns

Column Name	Type	Info Re.	Assays	#Comparators[Page 12, 6]	Other
BF_100	Bool	Compound-Target Pair	binding + functional	>= 100	
BF_100_c_dt_d_dt	Bool	Compound-Target Pair	binding + functional	>= 100	at least one compound with an annotation of D_DT or C<p>_DT (C0_DT, C1_DT, C2_DT, C3_DT) per target
BF_100_d_dt	Bool	Compound-Target Pair	binding + functional	>= 100	at least one compound with an annotation of D_DT per target
B_100	Bool	Compound-Target Pair	binding	>= 100	
B_100_c_dt_d_dt	Bool	Compound-Target Pair	binding	>= 100	at least one compound with an annotation of D_DT or C<p>_DT (C0_DT, C1_DT, C2_DT, C3_DT) per target
B_100_d_dt	Bool	Compound-Target Pair	binding	>= 100	at least one compound with an annotation of D_DT per target

[6] Comparator compounds in this context are all compounds with a pchembl_value_mean_BF / _B. I.e., this includes compounds with a DTI of D_DT or C<p>_DT.

CHAPTER

THREE

USER GUIDE

The default version of the dataset (the full dataset as a CSV file based on the newest ChEMBL version) can be generated by calling

```
python main.py -o <output_path>
```

with further options explained in *Arguments*.

An overview of the available arguments is also available by calling

```
python main.py --help
```

The output will always contain the full dataset as a CSV file. The arguments only allow for the output of additional files or modify how the full dataset is extracted.

3.1 Arguments

Parameter	Required	Flag	Default	Explanation
--chembl, -c	No	No	None	ChEMBL version. The latest available ChEMBL version is used if this is not set.
--sqlite, -s	No	No	None	Path to SQLite database. If this is not set, ChEMBL is downloaded as an SQLite database and handled using the chembl_downloader package.
--output, -o	Yes	No	None	Path to write the output file(s) to.
--delimiter, -d	No	No	;	Delimiter in output csv-files.
--all_sources	No	Yes	n/a	Include all sources if this is set. By default, this is not set, and the dataset is calculated based on only literature sources.
--rdkit	No	Yes	n/a	Calculate RDKit-based compound properties if this is set.
--excel	No	Yes	n/a	Write the results to excel. Note: this may fail if the output is too large. The results will always be written to csv.
--BF	No	Yes	n/a	Write the subsets based on binding and functional assays.
--B	No	Yes	n/a	Write the subsets based on binding assays.
--debug	No	Yes	n/a	Log additional debugging information.

3.2 Accessing ChEMBL

ChEMBL is accessed either through a given path to an SQLite database download or through the chembl_downloader package. In both cases, SQLite is used to query ChEMBL. Some of the earlier ChEMBL versions are missing tables or fields required to calculate the dataset. Therefore, the earliest ChEMBL version for which the dataset can be calculated is ChEMBL 26.

CHAPTER

FOUR

SRC

4.1 add_chembl_compound_properties module

add_chembl_compound_properties.**add_all_chembl_compound_properties**(*df_combined: DataFrame*, *chembl_con: Connection*, *limit_to_literature: bool*) → tuple[DataFrame, DataFrame, DataFrame]

Add ChEMBL-based compound properties to the given compound-target pairs, specifically:

- the first publication date of a compound (first_publication_cpd)
- ChEMBL compound properties
- InChI, InChI key and canonical smiles
- ligand efficiency metrics
- ATC classifications

Parameters

- **df_combined** (*pd.DataFrame*) – Pandas DataFrame with compound-target pairs
- **chembl_con** (*sqlite3.Connection*) – Sqlite3 connection to ChEMBL database.
- **limit_to_literature** (*bool*) – Base first_publication_cpd on literature sources only if True. Base it on all available sources otherwise.

Returns

- Pandas DataFrame with added compound properties
- Pandas DataFrame with compound properties and structures for all compound ids in ChEMBL
- Pandas DataFrame with ATC annotations in ChEMBL

Return type

(pd.DataFrame, pd.DataFrame, pd.DataFrame)

add_chembl_compound_properties.**add_atc_classification**(*df_combined: DataFrame*, *chembl_con: Connection*) → tuple[DataFrame, DataFrame]

Query and add ATC classifications (level 1) from the atc_classification and molecule_atc_classification tables. ATC level annotations for the same parent_molregno are combined into one description that concatenates all descriptions sorted alphabetically into one string with ‘ | ‘ as a separator.

Parameters

- **`df_combined`** (*pd.DataFrame*) – Pandas DataFrame with compound-target pairs
- **`chembl_con`** (*sqlite3.Connection*) – Sqlite3 connection to ChEMBL database.

Returns

- Pandas DataFrame with added ATC classifications
- Pandas DataFrame with ATC annotations in ChEMBL

Return type

(pd.DataFrame, pd.DataFrame)

`add_chembl_compound_properties.`**`add_chembl_properties_and_structures`**(*df_combined: DataFrame*, *chembl_con: Connection*) → tuple[DataFrame, DataFrame]

Add compound properties from the compound_properties table (e.g., alogp, #hydrogen bond acceptors / donors, etc.). Add InChI, InChI key and canonical smiles.

Parameters

- **`df_combined`** (*pd.DataFrame*) – Pandas DataFrame with compound-target pairs
- **`chembl_con`** (*sqlite3.Connection*) – Sqlite3 connection to ChEMBL database.

Returns

- Pandas DataFrame with added compound properties and structures.
- Pandas DataFrame with compound properties and structures for all compound ids in ChEMBL.

Return type

(pd.DataFrame, pd.DataFrame)

`add_chembl_compound_properties.`**`add_first_publication_date`**(*df_combined: DataFrame*, *chembl_con: Connection*, *limit_to_literature: bool*) → DataFrame

Query and calculate the first publication of a compound based on ChEMBL data (column name: first_publication_cpd). If limit_to_literature is True, this corresponds to the first appearance of the compound in the literature according to ChEMBL. Otherwise this is the first appearance in any source in ChEMBL.

Parameters

- **`df_combined`** (*pd.DataFrame*) – Pandas DataFrame with compound-target pairs
- **`chembl_con`** (*sqlite3.Connection*) – Sqlite3 connection to ChEMBL database.
- **`limit_to_literature`** (*bool*) – Base first_publication_cpd on literature sources only if True.

Returns

Pandas DataFrame with added first_publication_cpd.

Return type

pd.DataFrame

`add_chembl_compound_properties.`**`add_ligand_efficiency_metrics`**(*df_combined: DataFrame*) → DataFrame

Calculate the ligand efficiency metrics for the compounds based on the mean pchembl values for a compound-target pair and the following ligand efficiency (LE) formulas:

$$
\begin{aligned}
LE &= \frac{\Delta G}{HA} \qquad \text{where } \Delta G = -RT\ln(K_d),\ -RT\ln(K_i), \text{or} -RT\ln(IC_{50}) \\
LE &= \frac{2.303 \cdot 298 \cdot 0.00199 \cdot pchembl\_value}{heavy\_atoms} \\
BEI &= \frac{pchembl\_mean \cdot 1000}{mw\_freebase} \\
SEI &= \frac{pchembl\_mean \cdot 100}{PSA} \\
LLE &= pchembl\_mean - ALOGP
\end{aligned}
$$

Since LE metrics are based on pchembl values, they are calculated twice. Once for the pchembl values based on binding + functional assays (BF) and once for the pchembl values based on binding assays only (B).

Parameters
df_combined (*pd.DataFrame*) – Pandas DataFrame with compound-target pairs

Returns
Pandas DataFrame with added ligand efficiency metrics

Return type
pd.DataFrame

4.2 add_chembl_target_class_annotations module

add_chembl_target_class_annotations.**add_chembl_target_class_annotations**(*df_combined: DataFrame*, *chembl_con: Connection*, *output_path: str*, *write_to_csv: bool*, *write_to_excel: bool*, *delimiter: str*, *chembl_version: str*, *limited_flag: str*) → tuple[DataFrame, DataFrame, DataFrame]

Add level 1 and 2 target class annotations. Assignments for target IDs with more than one target class assignment per level are summarised into one string with '|' as a separator between the different target class annotations.

Targets with more than one level 1 / level 2 target class assignment are written to a file. These could be reassigned by hand if a single target class is preferable.

Parameters

- **df_combined** (*pd.DataFrame*) – Pandas DataFrame with compound-target pairs
- **chembl_con** (*sqlite3.Connection*) – Sqlite3 connection to ChEMBL database.
- **output_path** (*str*) – Path to write the targets with more than one target class assignment to
- **write_to_csv** (*bool*) – True if output should be written to csv

- **`write_to_excel`** (*`bool`*) – True if output should be written to excel
- **`delimiter`** (*`str`*) – Delimiter in csv-output
- **`chembl_version`** (*`str`*) – Version of ChEMBL for output files
- **`limited_flag`** (*`str`*) – Document suffix indicating whether the dataset was limited to literature sources

Returns

- Pandas DataFrame with added target class annotations
- Pandas DataFrame with mapping from target id to level 1 target class
- Pandas DataFrame with mapping from target id to level 2 target class

Return type

(pd.DataFrame, pd.DataFrame, pd.DataFrame)

`add_chembl_target_class_annotations.`**`get_target_class_table`**`(`*chembl_con: Connection*, *current_tids: set[int]*`)` → DataFrame

Get level 1 and level 2 target class annotations in ChEMBL.

Parameters

- **`chembl_con`** (*`sqlite3.Connection`*) – Sqlite3 connection to ChEMBL database.
- **`current_tids`** (*`set[int]`*) – Set of target ids to take into account

Returns

Pandas DataFrame with target class information

Return type

pd.DataFrame

4.3 add_dti_annotations module

`add_dti_annotations.`**`add_dti_annotations`**`(`*df_combined: DataFrame*, *drug_mechanism_pairs_set: set*, *drug_mechanism_targets_set: set*`)` → DataFrame

Every compound-target pair is assigned a DTI (drug target interaction) annotation.

The assignment is based on three questions:

- **Is the compound-target pair in the drug_mechanisms table? =**
  Is it a known relevant compound-target interaction?
- What is the max_phase of the compound? = Is it a drug / clinical compound?
- Is the target in the drug_mechanisms table = Is it a therapeutic target?

The assigments are based on the following table:

in DM table?	max_phase?	th. target?	DTI	explanation
yes	4	–	D_DT[1]	drug - drug target
yes	3	–	C3_DT	clinical candidate in phase 3 - drug target
yes	2	–	C2_DT	clinical candidate in phase 2 - drug target
yes	1	–	C1_DT	clinical candidate in phase 1 - drug target
yes	<1	–	C0_DT	compound in unknown phase[2] - drug target
no	–	yes	DT	drug target
no	–	no	NDT	not drug target

Since ChEMBL32 there are three possible annotations in ChEMBL with a max_phase value not between 1 and 4:

- 0.5 = early phase 1 clinical trials
- **-1 = clinical phase unknown for drug or clinical candidate drug,**
  i.e., where ChEMBL cannot assign a clinical phase
- NULL = preclinical compounds with bioactivity data

All three are grouped together into the annotation C0_DT.

Compound-target pairs that were annotated with NDT, i.e., compound-target pairs that are not in the drug_mechanisms table and for which the target was also not in the drug_mechanisms table (not a comparator compound), are discarded.

Parameters

- **`df_combined`** (*`pd.DataFrame`*) – Pandas DataFrame with compound-target pairs based on activities AND drug_mechanism table
- **`drug_mechanism_pairs_set`** (*`set`*) – set of compound-target pairs in the drug_mechanism table
- **`drug_mechanism_targets_set`** (*`set`*) – set of targets in the drug_mechanism table

Returns

Pandas DataFrame with all compound-target pairs and their DTI annotations.

Return type

pd.DataFrame

4.4 add_rdkit_compound_descriptors module

`add_rdkit_compound_descriptors.`**`add_aromaticity_descriptors`**(*df_combined: DataFrame*) → DataFrame

Add number of aromatic atoms in a compounds, specifically:

- total # aromatics atoms (aromatic_atoms)
- # aromatic carbon atoms (aromatic_c)
- # aromatic nitrogen atoms (aromatic_n)
- # aromatic hetero atoms (aromatic_hetero)

[1] The annotation D_DT instead of C4_DT was chosen to be consistent with the annotations in a previous version of the dataset. For the same reason the column is named DTI (drug-target interaction) instead of CTI (compound-target interaction) despite having specific annotations for clinical canidates.

[2] C0_DT groups together all compounds with a max_phase not between 1 and 4.

Parameters
df_combined (*pd.DataFrame*) – Pandas DataFrame with compound-target pairs

Returns
Pandas DataFrame with added counts of aromatic atoms

Return type
pd.DataFrame

`add_rdkit_compound_descriptors.`**`add_built_in_descriptors`**`(`*df_combined: DataFrame*`)` → DataFrame

Add RDKit built-in compound descriptors.

Parameters
df_combined (*pd.DataFrame*) – Pandas DataFrame with compound-target pairs

Returns
Pandas DataFrame with added built-in RDKit compound descriptors

Return type
pd.DataFrame

`add_rdkit_compound_descriptors.`**`add_rdkit_compound_descriptors`**`(`*df_combined: DataFrame*`)` → DataFrame

Add RDKit-based compound descriptors (built-in and numbers of aromatic atoms).

Parameters
df_combined (*pd.DataFrame*) – Pandas DataFrame with compound-target pairs

Returns
Pandas DataFrame with added built-in RDKit compound descriptors and numbers of aromatic atoms

Return type
pd.DataFrame

`add_rdkit_compound_descriptors.`**`calculate_aromatic_atoms`**`(`*smiles_set: set[str]*`)` → tuple[dict[str, int], dict[str, int], dict[str, int], dict[str, int]]

Get dictionaries with number of aromatic atoms for each smiles.

Parameters
smiles_set (*set[str]*) – Set of smiles to calculate the number of aromatic atoms for

Returns

Dictionaries with:

- SMILES -> # aromatics atoms
- SMILES -> # aromatic carbon atoms
- SMILES -> # aromatic nitrogen atoms
- SMILES -> # aromatic hetero atoms

Return type
(dict[str, int], dict[str, int], dict[str, int], dict[str, int])

4.5 clean_dataset module

`clean_dataset.`**`clean_dataset`**(*df_combined: DataFrame*, *calculate_rdkit: bool*) → DataFrame

Clean the dataset by

- changing nan values and empty strings to None
- setting the type of relevant columns to Int64
- rounding floats to 4 decimal places (with the exception of max_phase which is not rounded)
- reordering columns
- sorting rows by cpd_target_pair_mutation

Parameters

- **`df_combined`** (*`pd.DataFrame`*) – Pandas DataFrame with compound-target pairs
- **`calculate_rdkit`** (*`bool`*) – True if the DataFrame contains RDKit-based compound properties

Returns
Cleaned pandas DataFrame with compound-target pairs

Return type
pd.DataFrame

`clean_dataset.`**`clean_none_values`**(*df_combined*)

Change nan values and empty strings to None for consistency.

`clean_dataset.`**`remove_compounds_without_smiles_and_mixtures`**(*df_combined: DataFrame*, *chembl_con: Connection*) → DataFrame

Remove

- compounds without a smiles
- compounds with smiles containing a dot (mixtures and salts).

Since compound information is aggregated for the parents of salts, the number of smiles with a dot is relatively low.

Parameters

- **`df_combined`** (*`pd.DataFrame`*) – Pandas DataFrame with compound-target pairs
- **`chembl_con`** (*`sqlite3.Connection`*) – Sqlite3 connection to ChEMBL database.

Returns
Pandas DataFrame with compound-target pairs with a smiles that does not contain a '.'

Return type
pd.DataFrame

`clean_dataset.`**`reorder_columns`**(*df_combined*, *calculate_rdkit*)

Reorder the columns in the DataFrame.

`clean_dataset.`**`round_floats`**(*df_combined*, *decimal_places=4*)

Round float columns to <decimal_places> decimal places. This does not apply to max_phase.

clean_dataset.**set_types_to_int**(*df_combined*, *calculate_rdkit*)

Set the type of relevant columns to Int64.

4.6 get_activity_ct_pairs module

get_activity_ct_pairs.**get_aggregated_activity_ct_pairs**(*chembl_con: Connection*, *limit_to_literature: bool*, *df_sizes: list[list[int], list[int]]*) → DataFrame

Get dataset of compound target-pairs with an associated pchembl value with pchembl and publication dates aggregated into one entry per pair.

Values are aggregated for

- a subset of the initial dataset based on binding and functional assays (suffix '_BF') and
- a subset of the initial dataset set on only binding assays (suffix '_B').

Therefore, there are two columns for pchembl_value_mean, _max, _median, first_publication_cpd_target_pair and first_publication_cpd_target_pair_w_pchembl, one with the suffix '_BF' based on binding + functional data and one with the suffix '_B' based on only binding data.

Parameters

- **chembl_con** (*sqlite3.Connection*) – Sqlite3 connection to ChEMBL database.
- **limit_to_literature** (*bool*) – Include only literature sources if True. Include all available sources otherwise.
- **df_sizes** (*list[list[int], list[int]]*) – List of intermediate sized of the dataset used for debugging.

Returns

Pandas Dataframe with compound-target pairs based on ChEMBL activity data aggregated into one entry per compound-target pair.

Return type

pd.DataFrame

get_activity_ct_pairs.**get_average_info**(*df: DataFrame*, *suffix: str*) → DataFrame

Aggregate the information about compound-target pairs for which there is more than one entry into one entry. Compound-target pairs are considered equal if parent_molregno (internal compound ID) and tid_mutation (target ID + mutation annotations) are equal.

The following values are aggregated:

pchembl_value_mean	mean pchembl value for a compound-target pair
pchembl_value_max	maximum pchembl value for a compound-target pair
pchembl_value_median	median pchembl value for a compound-target pair
first_publication_cpd_target_pair	first publication in ChEMBL with this compound-target pair
first_publication_cpd_target_pair_w	first publication in ChEMBL with this compound-target pair and an associated pchembl value

Parameters

- **df** (*pd.DataFrame*) – Pandas DataFrame with compound-target pairs for which the information should be aggregated.

- **suffix** (*str*) – Suffix indicating the type of the given DataFrame, e.g., _B for binding assays, _BF for binding+functional assays.

Returns
Pandas DataFrame with 'parent_molregno', 'tid_mutation', and the aggregated columns.

Return type
pd.DataFrame

get_activity_ct_pairs.**get_compound_target_pairs_with_pchembl**(*chembl_con: Connection*, *limit_to_literature: bool*, *df_sizes: list[list[int], list[int]]*) → DataFrame

Query ChEMBL activities and related assay for compound-target pairs with an associated pchembl value. Compound-target pairs are required to have a pchembl value. Salt forms of compounds are mapped to their parent form. If limit_to_literature is true, only literature sources will be considered. Otherwise, all sources are included. Includes information about targets, mutations and year of publication (based on docs).

Parameters

- **chembl_con** (*sqlite3.Connection*) – Sqlite3 connection to ChEMBL database.
- **limit_to_literature** (*bool*) – Include only literature sources if True. Include all available sources otherwise.
- **df_sizes** (*list[list[int], list[int]]*) – List of intermediate sized of the dataset used for debugging.

Returns
Pandas DataFrame with compound-target pairs with a pchembl value.

Return type
pd.DataFrame

4.7 get_dataset module

get_dataset.**get_ct_pair_dataset**(*chembl_con: Connection*, *chembl_version: str*, *output_path: str*, *limit_to_literature: bool*, *calculate_rdkit: bool*, *write_to_csv: bool*, *write_to_excel: bool*, *delimiter: str*, *write_full_dataset: bool*, *write_bf: bool*, *write_b: bool*)

Calculate and output the compound-target pair dataset.

Parameters

- **chembl_con** (*sqlite3.Connection*) – Sqlite3 connection to ChEMBL database
- **chembl_version** (*str*) – Version of ChEMBL for output file names
- **output_path** (*str*) – Path to write output files to
- **limit_to_literature** (*bool*) – Include only literature sources if True. Include all available sources otherwise.
- **calculate_rdkit** (*bool*) – True if RDKit-based compound properties should be calculated
- **write_to_csv** (*bool*) – True if output should be written to csv
- **write_to_excel** (*bool*) – True if output should be written to excel

- **delimiter** (*str*) – Delimiter in csv-output
- **write_full_dataset** (*bool*) – True if the full dataset should be written to output
- **write_bf** (*bool*) – True if subsets based on binding+functional data should be written to output
- **write_b** (*bool*) – True if subsets based on binding data only should be written to output

4.8 get_drug_mechanism_ct_pairs module

get_drug_mechanism_ct_pairs.**add_annotations_to_drug_mechanisms_cti**(*chembl_con: Connection*, *cpd_target_pairs: DataFrame*) → DataFrame

Add additional information to the compound-target pairs from the drug_mechanisms table to match the information that is present in the compound-target pairs table based on activities.

Parameters

- **chembl_con** (*sqlite3.Connection*) – Sqlite3 connection to ChEMBL database.
- **cpd_target_pairs** (*pd.DataFrame*) – Pandas DataFrame with compound-target pairs from the drug_mechanism table.

Returns

Updated pandas DataFrame with the additional annotations.

Return type

pd.DataFrame

get_drug_mechanism_ct_pairs.**add_drug_mechanism_ct_pairs**(*df_combined: DataFrame*, *chembl_con: Connection*) → tuple[DataFrame, set, set]

Add compound-target pairs from the drug_mechanism table that are not in the dataset based on the initial ChEMBL query. These are compound-target pairs for which there is no associated pchembl value data. Since the pairs are known interactions, they are added to the dataset despite not having a pchembl value.

Parameters

- **df_combined** (*pd.DataFrame*) – Pandas Dataframe with compound-target pairs based on ChEMBL activity data
- **chembl_con** (*sqlite3.Connection*) – Sqlite3 connection to ChEMBL database.

Returns

- **Pandas DataFrame with compound-target pairs**
  based on activities AND drug_mechanism table
- set of compound-target pairs in the drug_mechanism table
- set of targets in the drug_mechanism table

Return type

(pd.DataFrame, set, set)

get_drug_mechanism_ct_pairs.**get_drug_mechanism_ct_pairs**(*chembl_con: Connection*) → DataFrame

Get compound-target pairs from the drug_mechanism table with all the columns that are present in the compound-target pairs based on activities. Relevant mappings of target ids to related target ids are taken into account.

Parameters

chembl_con (*sqlite3.Connection*) – Sqlite3 connection to ChEMBL database.

Returns
Pandas DataFrame with compound-target interactions from the drug_mechanism table.

Return type
pd.DataFrame

get_drug_mechanism_ct_pairs.**get_drug_mechanisms_interactions**(*chembl_con: Connection*) → DataFrame

Extract the known compound-target interactions from the ChEMBL drug_mechanisms table. Note: While the interactions are mostly between drugs and targets, the table also includes some known interactions between compounds with a max_phase < 4 and their targets.

Only entries with a disease_efficacy of 1 are taken into account, i.e., the target is believed to play a role in the efficacy of the drug.

disease_efficacy: Flag to show whether the target assigned is believed to play a role in the efficacy of the drug in the indication(s) for which it is approved (1 = yes, 0 = no).

Parameters
chembl_con (*sqlite3.Connection*) – Sqlite3 connection to ChEMBL database.

Returns
Pandas DataFrame with compound-target pairs from the drug_mechanism table with disease relevance.

Return type
pd.DataFrame

get_drug_mechanism_ct_pairs.**get_relevant_tid_mappings**(*chembl_con: Connection*) → DataFrame

Get DataFrame with mappings from target id to their related target ids based on the target_relations table. The following mappings are considered:

protein family	-[superset of]->	single protein
protein complex	-[superset of]->	single protein
protein complex group	-[superset of]->	single protein
single protein	-[equivalent to]->	single protein
chimeric protein	-[superset of]->	single protein
protein-protein interaction	-[superset of]->	single protein

These mappings can be used to increase the number of target ids for which there is data in the drug_mechanisms table. For example, for *protein family -[superset of]-> single protein* this means: If there is a known relevant interaction between a compound and a protein family, interactions between the compound and single proteins of that protein family are considered to be known interactions as well.

Parameters
chembl_con (*sqlite3.Connection*) – Sqlite3 connection to ChEMBL database.

Returns
Pandas DataFrame with mappings from tid to related tid for the defined subset of target relations.

Return type
pd.DataFrame

4.9 get_stats module

`get_stats.`**`add_dataset_sizes`**(*df: DataFrame*, *label: str*, *df_sizes: list[list[int], list[int]]*)

Count and add representative counts of df to the list df_sizes used for debugging.

Parameters

- **`df`** (*`pd.DataFrame`*) – Pandas DataFrame with current compound-target pairs
- **`label`** (*`str`*) – Description of pipeline step (e.g., initial query).
- **`df_sizes`** (*`list[list[int], list[int]]`*) – List of intermediate sized of the dataset used for debugging.

`get_stats.`**`calculate_dataset_sizes`**(*df: DataFrame*) → list[int]

Calculate the number of unique compounds, targets and pairs for df and df limited to drugs.

Parameters
`df` (*`pd.DataFrame`*) – Pandas DataFrame for which the dataset sizes should be calculated.

Returns
List of calculated unique counts.

Return type
list[int]

`get_stats.`**`get_stats_for_column`**(*df: DataFrame*, *column: str*, *columns_desc: str*) → list[list[str, str, int]]

Calculate the number of unique values in df[column] and various subsets of df.

Parameters

- **`df`** (*`pd.DataFrame`*) – Pandas Dataframe for which the number of unique values should be calculated
- **`column`** (*`str`*) – Column of df that the values should be calculated for
- **`columns_desc`** (*`str`*) – Description of the column

Returns
List of results in the format [column_name, subset_type, size]

Return type
list[list[str, str, int]]

4.10 main module

`main.`**`main`**()

Call get_ct_pair_dataset to get the compound-target dataset using the given arguments.

`main.`**`parse_args`**() → Namespace

Get arguments with argparse.

Returns
Populated argparse.Namespace

Return type
argparse.Namespace

4.11 sanity_checks module

`sanity_checks.`**`check_atc_and_target_classes`**(*df_combined: DataFrame*, *atc_levels: DataFrame*, *target_classes_level1: DataFrame*, *target_classes_level2: DataFrame*)

Check that atc_level1 and target class information is only null if the parent_molregno / target id is not in the respective table.

`sanity_checks.`**`check_compound_props`**(*df_combined: DataFrame*, *df_cpd_props: DataFrame*)

Check that compound props are only null if

- the property in the parent_molregno is not in df_cpd_props
- or if the value in the compound props table is null.

`sanity_checks.`**`check_for_mixed_types`**(*df_combined: DataFrame*)

Check that there are no mixed types in columns with dtype=object.

`sanity_checks.`**`check_ligand_efficiency_metrics`**(*df_combined: DataFrame*)

Check that ligand efficiency metrics are only null when at least one of the values used to calculate them is null. Ligand efficiency metrics are only null when at least one of the values used to calculate them is null.

`sanity_checks.`**`check_null_values`**(*df_combined: DataFrame*)

Check if any columns contain nan or null which aren't recognised as null values.

`sanity_checks.`**`check_pairs_without_pchembl_are_in_drug_mechanisms`**(*df_combined: DataFrame*)

Check that rows without a pchembl value based on binding+functional assays (pchembl_x_BF) are in the drug_mechanism table. Note that this is not true for the pchembl_x_B columns which are based on binding data only. They may be in the table because there is data based on functional assays but no data based on binding assays. All pchembl_value_x_BF columns without a pchembl should be in the dm table.

`sanity_checks.`**`check_rdkit_props`**(*df_combined: DataFrame*)

Check that columns set by the RDKit are only null if there is no canonical SMILES for the molecule. Scaffolds are excluded from this test because they can be None if the molecule is acyclic.

`sanity_checks.`**`sanity_checks`**(*df_combined: DataFrame*, *df_cpd_props: DataFrame*, *atc_levels: DataFrame*, *target_classes_level1: DataFrame*, *target_classes_level2: DataFrame*, *calculate_rdkit: bool*)

Check basic assumptions about the finished dataset, specifically:

- no columns contain nan or null values which aren't recognised as null values
- there are no mixed types in columns with dtype=object
- **rows without a pchembl value based on binding+functional assays (pchembl_x_BF)**
  are in the drug_mechanism table
- **ligand efficiency metrics are only null when at least one of the values**
  used to calculate them is null
- **compound props are only null if the compound is not in df_cpd_props**
  or the value in that table is null
- **atc_level1 and target class information is only null if**
  the parent_molregno / target id is not in the respective table
- **columns set by the RDKit are only null if there is no canonical SMILES**
  for the molecule (excluding scaffolds)

Parameters

- **df_combined** (*pd.DataFrame*) – Pandas DataFrame with compound-target pairs
- **df_cpd_props** (*pd.DataFrame*) – Pandas DataFrame with compound properties and structures for all compound ids in ChEMBL.
- **atc_levels** (*pd.DataFrame*) – Pandas DataFrame with ATC annotations in ChEMBL
- **target_classes_level1** (*pd.DataFrame*) – Pandas DataFrame with mapping from target id to level 1 target class
- **target_classes_level2** (*pd.DataFrame*) – Pandas DataFrame with mapping from target id to level 2 target class
- **calculate_rdkit** (*bool*) – True if the DataFrame contains RDKit-based compound properties

`sanity_checks.`**`test_equality`**(*current_df: DataFrame*, *read_file_name: str*, *assay_type: str*, *file_type_list: list[str]*, *calculate_rdkit: bool*)

Check that the file that was written to <read_file_name> is identical to the DataFrame <current_df> it was based on.

Parameters

- **current_df** (*pd.DataFrame*) – Pandas DataFrame that was written to read_file_name
- **read_file_name** (*str*) – Name of the file current_df was written to
- **assay_type** (*str*) – Types of assays current_df contains information about. Options: “BF” (binding+functional), “B” (binding), “all” (contains both BF and B information)
- **file_type_list** (*list[str]*) – List of file extensions used with read_file_name. Options: csv, xlsx
- **calculate_rdkit** (*bool*) – If True, current_df contains RDKit-based columns

4.12 write_subsets module

`write_subsets.`**`get_data_subsets`**(*data: DataFrame*, *min_nof_cpds: int*, *desc: str*) → tuple[DataFrame, DataFrame, DataFrame, DataFrame]

Calculate and return the different subsets of interest.

Parameters

- **data** (*pd.DataFrame*) – Pandas DataFrame with compound-target pairs
- **min_nof_cpds** (*int*) – Miminum number of compounds per target
- **desc** (*str*) – Types of assays current_df contains information about. Options: “BF” (binding+functional), “B” (binding)

Returns

- **data: Pandas DataFrame with compound-target pairs**
  without the annotations for the opposite desc, e.g. if desc = “BF”, the average pchembl value based on binding data only is dropped
- **df_enough_cpds: Pandas DataFrame with targets**
  with at least <min_nof_cpds> compounds with a pchembl value,

- **df_c_dt_d_dt: As df_enough_cpds but with at least one compound-target pair labelled as**
  'D_DT', 'C3_DT', 'C2_DT', 'C1_DT' or 'C0_DT' (i.e., known interaction),
- **df_d_dt: As df_enough_cpds but with at least one compound-target pair labelled as**
  'D_DT' (i.e., known drug-target interaction)

Return type
(pd.DataFrame, pd.DataFrame, pd.DataFrame, pd.DataFrame)

`write_subsets.`**`output_debug_sizes`**(*df_sizes: list[list[int], list[int]]*, *output_path: str*, *write_to_csv: bool*, *write_to_excel: bool*, *delimiter: str*)

Output counts at various points during calculating the final dataset for debugging.

Parameters

- **`df_sizes`** (*`list[list[int], list[int]]`*) – List of intermediate sized of the dataset used for debugging.
- **`output_path`** (*`str`*) – Path to write the dataset counts to
- **`write_to_csv`** (*`bool`*) – True if counts should be written to csv
- **`write_to_excel`** (*`bool`*) – True if counts should be written to excel
- **`delimiter`** (*`str`*) – Delimiter in csv-output

`write_subsets.`**`output_stats`**(*df: DataFrame*, *output_file: str*, *write_to_csv: bool*, *write_to_excel: bool*, *delimiter: str*)

Summarise and output the number of unique values in the following columns:

- parent_molregno (compound ID)
- tid (target ID)
- tid_mutation (target ID + mutation annotations)
- cpd_target_pair (compound-target pairs)
- cpd_target_pair_mutation (compound-target pairs including mutation annotations)

Parameters

- **`df`** (*`pd.DataFrame`*) – Pandas Dataframe for which the stats should be calculated
- **`output_file`** (*`str`*) – Path and filename to write the dataset stats to
- **`write_to_csv`** (*`bool`*) – True if stats should be written to csv
- **`write_to_excel`** (*`bool`*) – True if stats should be written to excel
- **`delimiter`** (*`str`*) – Delimiter in csv-output

`write_subsets.`**`write_and_check_output`**(*df: DataFrame*, *filename: str*, *write_to_csv: bool*, *write_to_excel: bool*, *delimiter: str*, *assay_type: str*, *calculate_rdkit: bool*)

Write df to file and check that writing was successful.

Parameters

- **`df`** (*`pd.DataFrame`*) – Pandas Dataframe to write to output file.
- **`filename`** (*`bool`*) – Filename to write the output to
- **`write_to_csv`** (*`bool`*) – True if output should be written to csv
- **`write_to_excel`** (*`bool`*) – True if output should be written to excel

- **delimiter** (*str*) – Delimiter in csv-output
- **assay_type** (*str*) – Types of assays current_df contains information about. Options: “BF” (binding+functional), “B” (binding), “all” (contains both BF and B information)
- **calculate_rdkit** (*bool*) – If True, current_df contains RDKit-based columns

write_subsets.**write_b_to_file**(*df_combined: DataFrame*, *df_combined_annotated: DataFrame*, *chembl_version: str*, *min_nof_cpds_b: int*, *output_path: str*, *write_b: bool*, *write_to_csv: bool*, *write_to_excel: bool*, *delimiter: str*, *limited_flag: str*, *calculate_rdkit: bool*, *df_sizes: list[list[int], list[int]]*) → DataFrame

Calculate relevant subsets for the portion of df_combined that is based on binding data. If write_b the subsets are written to output_path. Independent of write_b, filtering columns for B are added to df_combined_annotated.

Parameters

- **df_combined** (*pd.DataFrame*) – Pandas DataFrame with compound-target pairs
- **df_combined_annotated** (*pd.DataFrame*) – Pandas DataFrame with additional filtering columns
- **chembl_version** (*str*) – Version of ChEMBL for output files
- **min_nof_cpds_b** (*int*) – Miminum number of compounds per target
- **output_path** (*str*) – Path to write the output to
- **write_b** (*bool*) – Should the subsets be written to files?
- **write_to_csv** (*bool*) – Should the subsets be written to csv?
- **write_to_excel** (*bool*) – Should the subsets be written to excel?
- **delimiter** (*str*) – Delimiter for csv output
- **limited_flag** (*str*) – Document suffix indicating whether the dataset was limited to literature sources
- **calculate_rdkit** (*bool*) – Does df_combined include RDKit-based columns?
- **df_sizes** (*list[list[int], list[int]]*) – List of intermediate sized of the dataset used for debugging.

Returns
Pandas DataFrame with additional filtering columns for B subsets

Return type
pd.Dataframe

write_subsets.**write_bf_to_file**(*df_combined: DataFrame*, *chembl_version: str*, *min_nof_cpds_bf: int*, *output_path: str*, *write_bf: bool*, *write_to_csv: bool*, *write_to_excel: bool*, *delimiter: str*, *limited_flag: str*, *calculate_rdkit: bool*, *df_sizes: list[list[int], list[int]]*) → DataFrame

Calculate relevant subsets for the portion of df_combined that is based on binding+functional data. If write_bf the subsets are written to output_path. Independent of write_bf, filtering columns for BF are added to df_combined and returned.

Parameters

- **df_combined** (*pd.DataFrame*) – Pandas DataFrame with compound-target pairs
- **chembl_version** (*str*) – Version of ChEMBL for output files
- **min_nof_cpds_bf** (*int*) – Miminum number of compounds per target

- **`output_path`** (*`str`*) – Path to write the output to
- **`write_bf`** (*`bool`*) – Should the subsets be written to files?
- **`write_to_csv`** (*`bool`*) – Should the subsets be written to csv?
- **`write_to_excel`** (*`bool`*) – Should the subsets be written to excel?
- **`delimiter`** (*`str`*) – Delimiter for csv output
- **`limited_flag`** (*`str`*) – Document suffix indicating whether the dataset was limited to literature sources
- **`calculate_rdkit`** (*`bool`*) – Does df_combined include RDKit-based columns?
- **`df_sizes`** (*`list[list[int], list[int]]`*) – List of intermediate sized of the dataset used for debugging.

Returns
Pandas DataFrame with additional filtering columns for BF subsets

Return type
pd.Dataframe

`write_subsets.`**`write_full_dataset_to_file`**(*df_combined: DataFrame*, *chembl_version: str*, *output_path: str*, *write_full_dataset: bool*, *write_to_csv: bool*, *write_to_excel: bool*, *delimiter: str*, *limited_flag: str*, *calculate_rdkit: bool*)

If write_full_dataset, write df_combined with filtering columns to output_path.

Parameters

- **`df_combined`** (*`pd.DataFrame`*) – Pandas DataFrame with compound-target pairs and filtering columns
- **`chembl_version`** (*`str`*) – Version of ChEMBL for output files
- **`output_path`** (*`str`*) – Path to write the output to
- **`write_full_dataset`** (*`bool`*) – Should the subsets be written to files?
- **`write_to_csv`** (*`bool`*) – Should the subsets be written to csv?
- **`write_to_excel`** (*`bool`*) – Should the subsets be written to excel?
- **`delimiter`** (*`str`*) – Delimiter for csv output
- **`limited_flag`** (*`str`*) – Document suffix indicating whether the dataset was limited to literature sources
- **`calculate_rdkit`** (*`bool`*) – Does df_combined include RDKit-based columns?

`write_subsets.`**`write_output`**(*df: DataFrame*, *filename: str*, *write_to_csv: bool*, *write_to_excel: bool*, *delimiter: str*) → list[str]

Write DataFrame df to output file named <filename>.

Parameters

- **`df`** (*`pd.DataFrame`*) – Pandas Dataframe to write to output file.
- **`filename`** (*`bool`*) – Filename to write the output to
- **`write_to_csv`** (*`bool`*) – True if output should be written to csv
- **`write_to_excel`** (*`bool`*) – True if output should be written to excel
- **`delimiter`** (*`str`*) – Delimiter in csv-output

Returns
Returns list of types of files that was written to (csv and/or xlsx)

Return type
list[str]

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[Zdrazil2023] Zdrazil et al., “The ChEMBL Database in 2023: a drug discovery platform spanning multiple bioactivity data types and time periods”, Nucleic Acids Research, gkad1004, 2023, https://doi.org/10.1093/nar/gkad1004

[Leeson2021] Leeson et al., “Target-Based Evaluation of “Drug-Like” Properties and Ligand Efficiencies”, Journal of Medicinal Chemistry, 64(11), 7210-7230, 2021, https://doi.org/10.1021/acs.jmedchem.1c00416

PYTHON MODULE INDEX

INDEX