Documentation for the *Chemexpy* package

Package version: v1.0.10
Date: 10/06/2021
Author: Auste Kanapeckaite

The package contains the following functions:
1. data_prep
2. molecule_check
3. scatter_plot
4. correlation_plot
5. feature_plot
6. normality_check
7. feature_check
8. feature_violinplots
9. similarity_search
10. similarity_dendogram
11. similarity_heatmap

Dependencies:

rdkit, pandas, numpy, scipy, seaborn, matplotlib, collections

Introduction.

Chemexpy package provides a user-friendly and organised approach to explore chemical libraries and identify key features. The information generated by the package functions can be easily integrated into other pipelines or downstream processing. The package provides exploratory plots as well as compound similarity assessment allowing to search for similar compounds. Moreover, there are several additional functions helping to easily extract chemical descriptors and evaluate chemical libraries.

Function Description.

1. Function data_prep

Function call example: data_prep(data,*args)

#Function provides a snapshot of the input data as well as returns a processed data file to include information on chemical descriptors, atomic composition, chemical structure features.

#Input values: path to a csv file that contains compound ID 'CID' and smiles 'SMILES' columns. These columns have to be named as described above.
Additional columns can be passed as arguments if, for example, the data file contains other columns of interest.

#Output values: data frame with added chemical descriptors. The output could be integrated into downstream analyses and databases or used to visualise the structures.

2. Function molecule_check

Function call example: molecule_check(data,*args)

#Function allows to visualise molecules of interest as well as returns a data frame that contains information on the selected list of molecules. It is recommended not to select more than 20 molecules at a time to draw the structures.

#Input values: data frame with "CID" (compound ID) and "SMILES" (smiles column). Please note, the IDs for columns need to match the examples.

#Additional input: arguments for "CID", e.g., "2821293". If none is selected first 10 structures will be drawn. Names for compounds have to be in a string format.

#Output values: structure visualisation and a data frame that can be used for further visualisations.

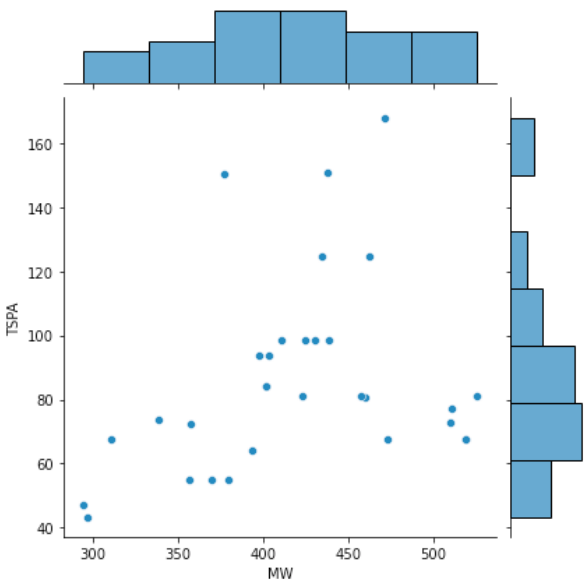
3. Function scatter_plot

Function call example: scatter_plot(data,var1=None,var2=None)

#Function takes the data file provided by the data_prep function and plots analytical scatter plots for selected variables.

#Input values: data frame generated by the data_prep function, as well as variables to plot, e.g. "MW" and "TSPA".

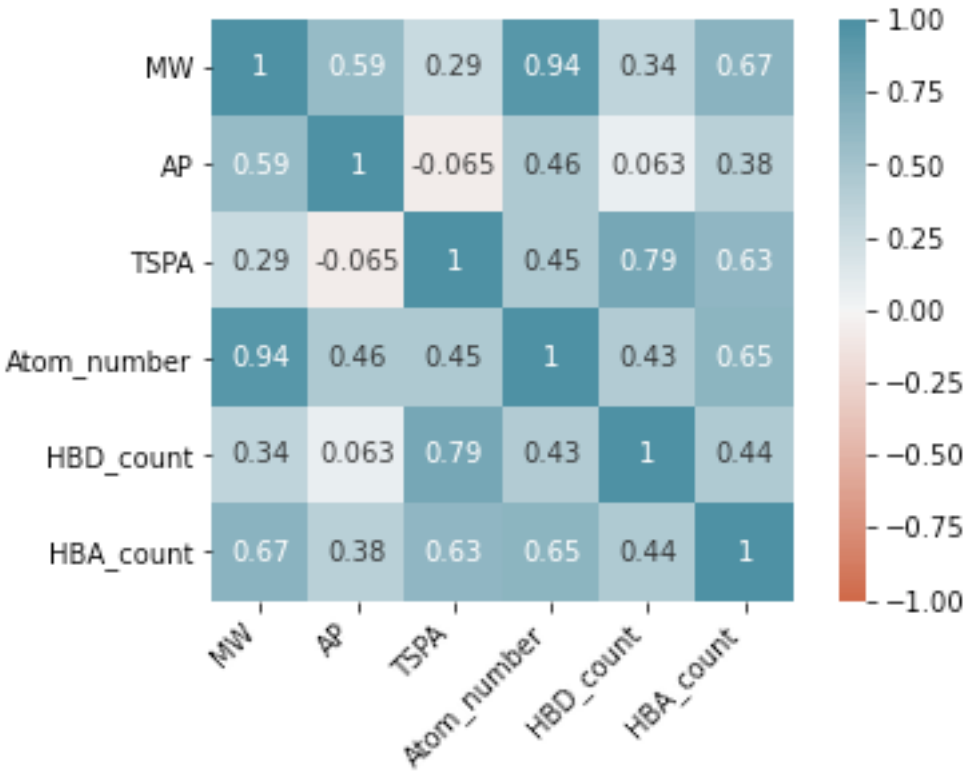
#Output values: scatter plot.



4. Function correlation_plot

Function call example: correlation_plot(data,*args)

#Function takes the data file provided by the data_prep function and plots a correlation heatmap.
#Input values: data frame generated by the data_prep function, as well as variables to calculate correlation and plot selected values, e.g., "MW" and "TSPA".
If the user does not select args, the default values will be used: "Atom_number","MW","TSPA","HBD_count","HBA_count","Rotatable_bond_count","MolLogP","Ring_number","AP".
#Output values: plot for correlation visualisation and a data frame with correlation values.

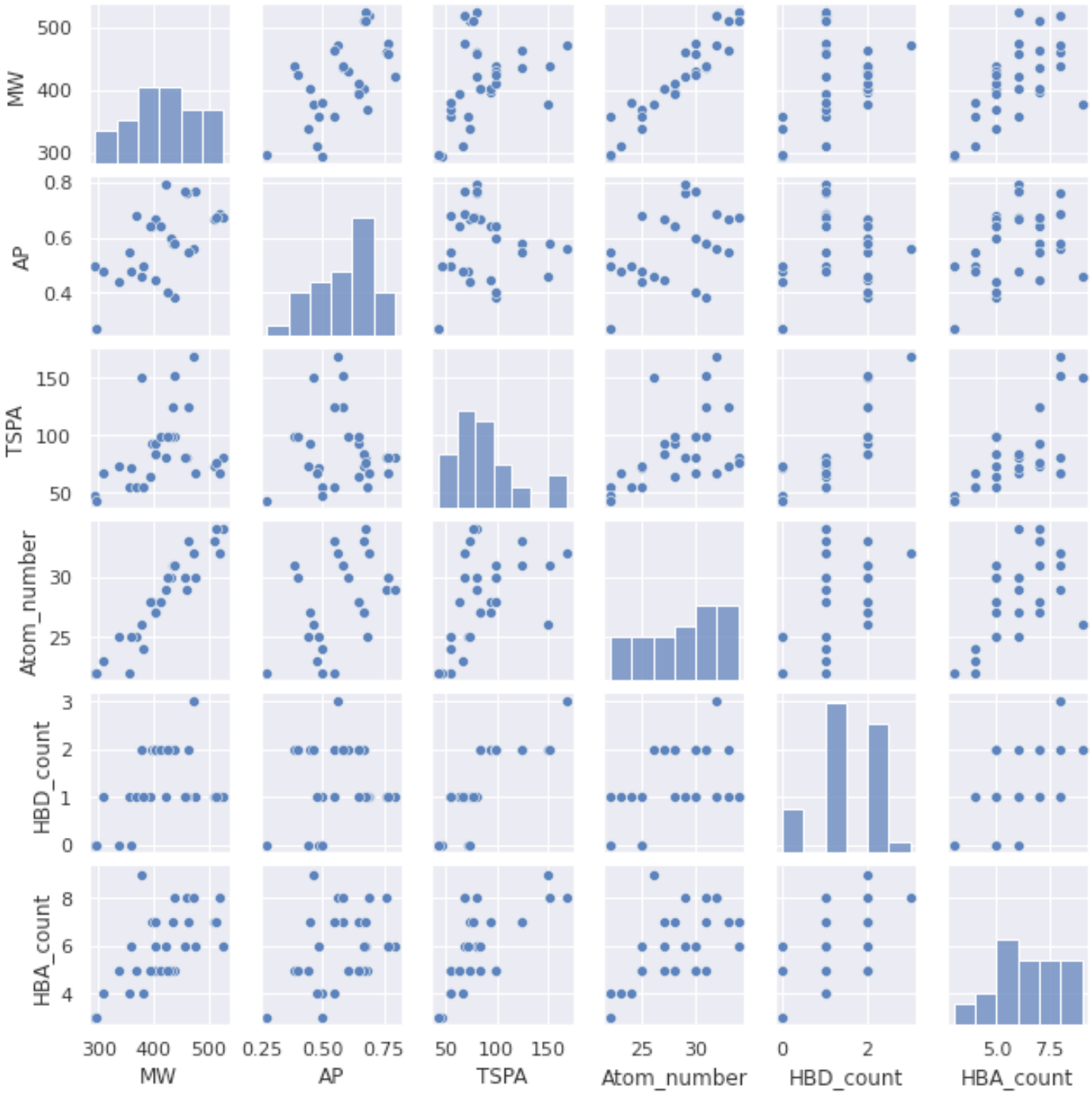


5. Function feature_plot

Function call example: feature_plot(data,*args)

#Function takes the data file provided by the data_prep function and plots analytical scatter plots for multiple features.
#Input: data frame generated by the data_prep function, as well as variables for feature ploting, e.g. "MW" and "TSPA".
#If the user does not select args, the default values will be used: "Atom_number","MW","TSPA","HBD_count","HBA_count", "Rotatable_bond_count","MolLogP","Ring_number","AP".

#Output: scatter plot of multiple feature visualisation.



6. Function normality_check

Function call example: normality_check(data,var=None)

#Function takes the data file provided by the data_prep function and plots a histogram with an estimated probability density function.
#Input: data frame generated by the data_prep function, as well as a single variable to check the normality for, e.g., "MW" and "TSPA".
#Output: bar plot with an estimated normal distribution line plot based on distribution probability.

7. Function feature_check

Function call example: feature_check(data,var1=None, var2=None, type=None)

#Function takes the data file provided by the data_prep function and plots analytical contour plots to assess chemical feature distribution when considering a specific chemical entity category. That is, a categorical type data needs to be provided, such as active or inactive, etc.
#Input: data frame generated by the data_prep function, two variables for distribution check, e.g., "MW" and "TSPA", and a column name to select categorical data from.
#Output: contour plot with feature distribution.

8. Function feature_violinplots

Function call example: feature_violinplots(data,var1=None,type=None)

#Function takes the data file provided by data_prep function and plots analytical violin plots to assess the type distribution for selected fetaures.
#Note categorical type data needs to be provided, such as active or inactive, etc.
#Input: data frame generated by the data_prep function as well as a variable name for the distribution check, e.g., "MW" and “TSPA"; also a column name is required to select categorical data specified through the "type" designation.
#Output: contour plot with feature distribution.

9. Function similarity_search

Function call example: similarity_search(data, target=None)

#Function takes the data file provided by the data_prep function and searches for similar structures based on the target molecule.
#Fingerprinting is based on Morgan fingerprints and the similarity search is based on Tanimoto similarity.
#Input: data frame generated by the data_prep function as well as a SMILE string (e.g., the "target" variable) for a molecule to search in the database.
#Output: data frame of similar structures.

10. Function similarity_dendogram

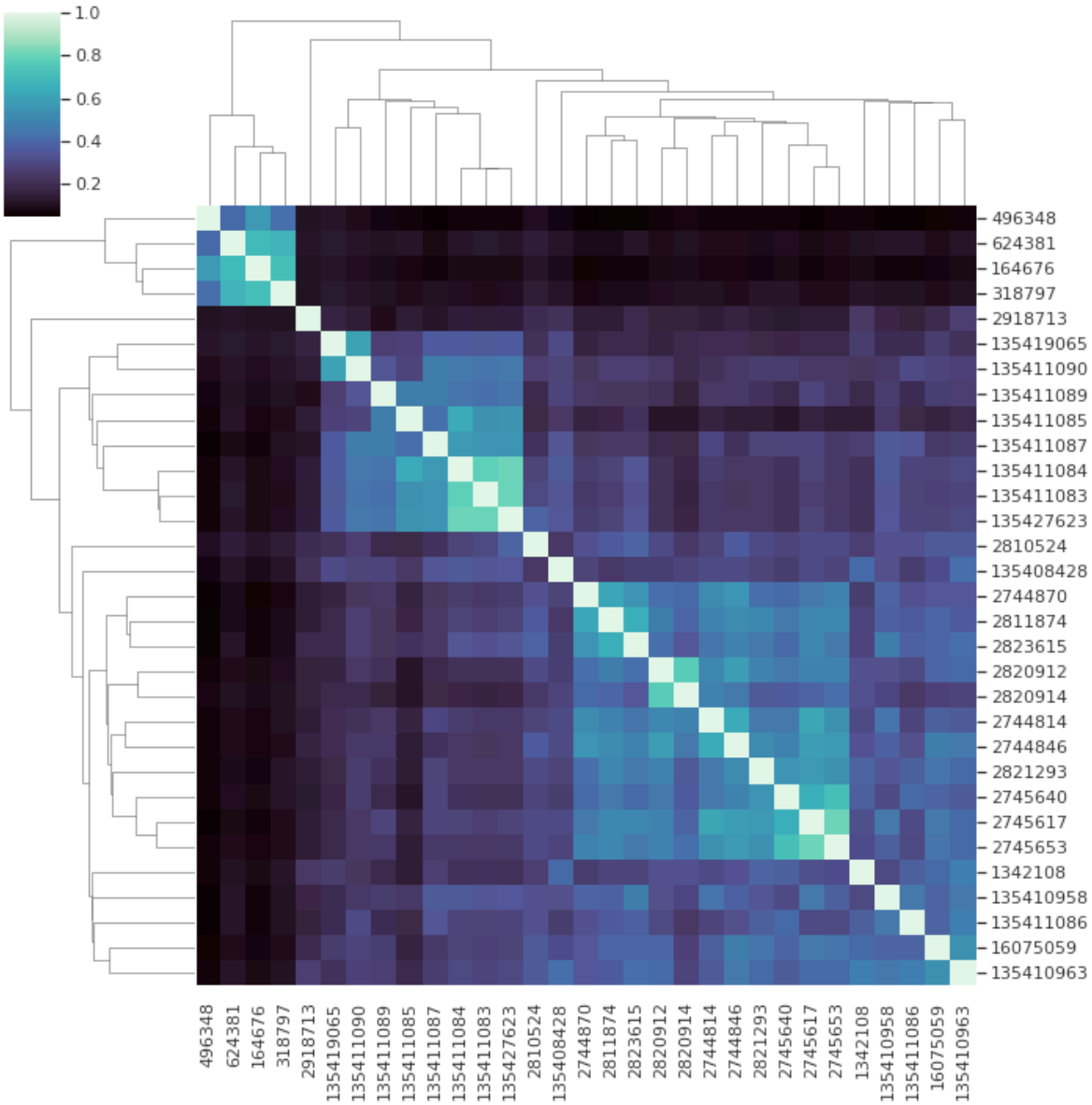
Function call example: similarity_dendogram(data)

#Function takes the data file provided by the data_prep function and plots a dendogram based on compound similarity.
#Fingerprinting is based on Morgan fingerprints and the similarity search is based on Tanimoto similarity.
#Input: data frame generated by the data_prep function.
#Output: dendogram and a data frame with similarity values.

11. Function similarity_heatmap

Function call example: similarity_heatmap(data)

#Function takes the data file provided by the data_prep function and plots a heatmap based on compound similarity.
#Fingerprinting is based on Morgan fingerprints and the similarity search is based on Tanimoto similarity.
#Input: data frame generated by the data_prep function.
#Output: heatmap and a data frame with similarity values.



Running tests and example use cases.

The working directory should contain example data sets (provided with packages PATH="./tests"). There are several datasets to choose from, namely data_1.csv and data_2.csv.

#initiative variables to data
data="./test/data_1.csv"

#prepare the data for subsequent use
#we are selecting an additional column to assess the activity based on a categorical value
data=data_prep(data, "Type")

```
#assess a selected set of molecules and retrieve a data frame that contains information about these molecules
data_eval=molecule_check(data,"2821293")


#evaluate exploratory plots

scatter_plot(data,"MW","TSPA")

corr=correlation_plot(data,"MW","TSPA","AP","HBD_count","HBA_count")

#perform multiple feature assessment

feature_plot(data,"MW","TSPA","AP","HBD_count","HBA_count")


#check the normality of the data distribution

normality_check(data,"MW")

#since we have one categorical value we can perform a feature check

feature_check(data,var1="MW", var2="AP", type="Type")


#similarity assessment
target='COC(=O)c1c[nH]c2cc(OC(C)C)c(OC(C)C)cc2c1=O'

target_matches=similarity_search(data, target)

#similarity value generation for all pairwise comparisons

#dendogram ploting and a data frame preparation with similarity values
#both functions produce the same data frame

similarity_data=similarity_dendogram(data)

similarity_datasimilarity_heatmap(data)
```