Supporting Information

Machine Learning in a Molecular Modeling Course for Chemistry, Biochemistry, and Biophysics Students

Jacob M. Remington, Jonathon B. Ferrell, Marlo Zorman, Adam Petrucci, Severin T. Schneebeli, Jianing Li*

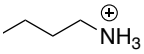
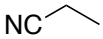
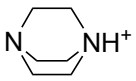
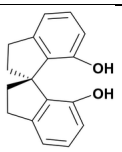
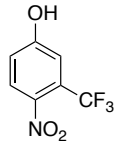
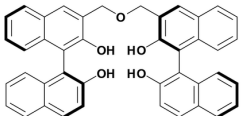
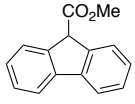
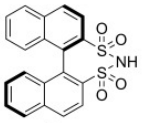
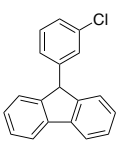
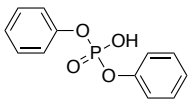
Department of Chemistry, The University of Vermont, Burlington, VT 05403

* Jianing Li (jianing.li@uvm.edu, 82 University Place, Burlington, VT 05403)

1. Design of student competition.

12 students divided into two groups (G1 and G2) were given 3 weeks to build the training data set and prepared the ML algorithm on a laptop to predict the value of pK_a for small organic compounds in the DMSO solution. Simple instructions were provided by the instructor (e.g. SMILES inputs, pK_a value outputs, etc.), and rules of the competition were explained as well. An incentive credit would be given to students in the winning group. In the last class for the ML topic, student spent 5 minutes at the beginning to setup their laptops and prepared for the competition.

Table S1. Performance of student group 1 and 2 (G1 and G2) in the ML-predicted pK_a competition.

Molecule	Exp.	G1	G2	Ref.	Molecule	Exp.	G1	G2	Ref.
	11.1	24.2	17.2	J.Chem. Research (S), 1997, 22-23		32.5	22.7	19.5	CEJ 2011, 17, 8524
	9.1	23.4	19.5	J.Chem. Research (S), 1997, 22-23		16.4	10.2	24.9	CEJ 2011, 17, 8524
	9.3	12.3	22.8	Org. Chem. Front., 2016,3, 1154-1158		11.9	10.2	18.0	CEJ 2011, 17, 8524
	10.4	9.6	16.4	Org. Chem. Front., 2016,3, 1154-1158		1.8	10.1	19.5	Org. Chem. Front., 2016, 3, 1154
	16.9	11.1	19.5	Org. Chem. Front., 2016,3, 1154-1158		3.8	10.2	17.3	Org. Chem. Front., 2016, 3, 1154

After receiving the structures of 10 compounds (Table S1), each group had 15 minutes to convert the structures into SMILES and generate the predictions. Then, each group was asked to report their predictions before the instructor announced the experimental values and the winner group.

2. Student survey analysis.

We analyzed the student responses from the survey given to students in class after the ML-predicted *p*K_a competition. In general, students provided very diverse feedback. The number before each response annotates the number of students that have the same responses.

Q1: What do you learn from Task 3 and related lectures?

Student Responses:

4 Databases to build a 'calibration curve'/predictions from large dataset.
2 Larger databases are more effective.
2 Choice of input (key) affects accuracy.
2 Random forest predictions.
1 Practical applications.
1 Discovery type research.
1 Importance/difficulties of curating large datasets.
1 *p*K_a and machine learning literature is larger than thought.
1 "Choosing the most effective substrings is difficult and important".
1 Recourses for converting between chemical formats.
1 Lots of human based parameterizations. thought it would be more automatic.

Q2: What do you like most when working on Task 3? Are you going to read or study the related topics?

Student Responses:

1 Programming part.
1 Implementing algorithm.
3 Logic behind programing. "try to think like a computer".
3 Applications of ML.
1 Range of ML algorithms.
1 Non-analytical approach to problem solving.

1 Learn a programing language.
1 Apply ML to biophysics and drug discovery.
1 ML to chemical applications.
1 ML algorithms.
1 Apply ML to my own research.
1 ML using quantum computing.
1 Structure prediction.
1 Take a CS course on ML.

Q3: What improvement can we implement for teaching CL and related topics in the future?

Student Responses:

2	More assignment as ML is abstract and doesn't make sense until you try to program.
2	Connecting the ML and the programming.
1	More topics on ML that could relate to drug discovery.
1	Do a simple task together as a class first before doing it as a project.
1	Example ML scripts/algorithms.
1	Make sure students have coding background, if project is to include coding.
1	Smaller groups so more get to struggle through writing codes.
3	Prebuilt databases, manually drawing molecules is tough.
1	Have students focus on perfecting a ML technique instead of building the database.
2	Difficult not having programing background.
1	Maybe a puzzle game that we solve by observing patterns could help conceptualize the process of machine learning.

3. Example Python scripts.

Example S1:

```
'''
This is an example computer program in Python to (1) create a small data set of chemical structures with SMILES
and (2) perform a simple search function to look for substructures specified with SMARTS.
'''
from rdkit import Chem
from rdkit.Chem.Draw import MolsToGridImage

smiles_list = ["CCCC#C","CCOCC","CCNCC","CCSCC", "CCCC=O", "CC(=O)CC"]
mol_list = [Chem.MolFromSmiles(x) for x in smiles_list]

query = Chem.MolFromSmarts("[O;D2]CC")
match_list = [mol.GetSubstructMatch(query) for mol in mol_list]
im=MolsToGridImage(mols=mol_list, molsPerRow=3,highlightAtomLists=match_list)
im.show()
```

Outcome of the script:

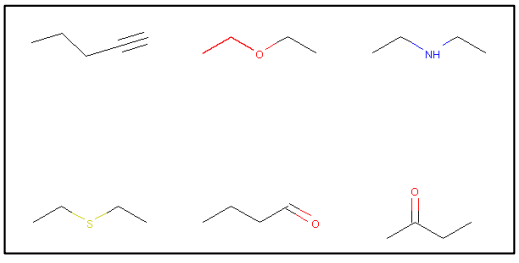


Figure S1. Example script for the students to execute a substructure search. Students are encouraged to modify the script.

Further application to biophysical course materials: This is a simple example to show the usage of SMILES representation/SMARTS search of molecules. It is not limited to small molecules and can be easily modified to show other molecules of biophysical interest.

Example S2:

```
'''
This is an example computer program in Python to compare the structural similarity between heroin, fentanyl, and the peptide dynorphin, which are ligands to the mu opioid receptor.
'''
from rdkit import Chem
from rdkit import DataStructs
from rdkit.Chem.Fingerprints import FingerprintMols
from rdkit.Chem import MACCSkeys

# heroin SMILES
heroin_mol =
Chem.MolFromSmiles("CC(=O)O[C@H]1C=C[C@H]2[C@H]3CC4=C5[C@]2([C@H]1OC5=C(C=C4)OC(=O)C)CCN3C")

# fentanyl SMILES
fentanyl_mol = Chem.MolFromSmiles("CCC(=O)N(C1CCN(CC1)CCC2=CC=CC=C2)C3=CC=CC=C3")

#dynorphin SMILES
dynorphin_mol =
Chem.MolFromSmiles("CCC(C)C(C(=O)NC(CCCNC(=N)N)C(=O)N1CCCC1C(=O)NC(CCCCN)C(=O)NC(CC(C)C)C(=O)NC(CCCCN)C(=O)NC(CC2=CNC3=CC=CC=C32)C(=O)NC(CC(=O)O)C(=O)NC(CC(=O)N)C(=O)NC(CCC(=O)N)C(=O)O)NC(=O)C(CCCNC(=N)N)NC(=O)C(CCCNC(=N)N)NC(=O)C(CC(C)C)NC(=O)C(CC4=CC=CC=C4)NC(=O)CNC(=O)CNC(=O)C(CC5=CC=C(C=C5)O)N")

mol_list = [heroin_mol,fentanyl_mol, dynorphin_mol]

fp_list = [MACCSkeys.GenMACCSKeys(x) for x in mol_list]

print ("\nMACCS Fingerprint Similarity:")
print( "heroin vs fentanyl    %8.3f"%DataStructs.FingerprintSimilarity(fp_list[0],fp_list[1]) )
print( "heroin vs dynorphin   %8.3f"%DataStructs.FingerprintSimilarity(fp_list[0],fp_list[2]) )
print( "fentanyl vs dynorphin %8.3f\n"%DataStructs.FingerprintSimilarity(fp_list[1],fp_list[2]) )
```

Outcome of the script:

```
MACCS Fingerprint Similarity:
heroin vs fentanyl      0.391
heroin vs dynorphin     0.488
fentanyl vs dynorphin   0.429
```

Figure S2. Example script for the students to investigate molecular similarity. Students are encouraged to modify the script. Notably, using the MACCS fingerprint similarity, each of heroin and fentanyl has a higher similarity to the neuropeptide dynorphin than the similarity between heroin and fentanyl.

Further application to biophysical course materials: This example can be readily generalized to other biophysical molecules, e.g. neurotransmitters, lipids, etc. (1) Chemical databases like PubChem (https://pubchem.ncbi.nlm.nih.gov) provides the SMILES strings of millions of compounds. (2) The RDKit contains a variety of built-in functionality for generating molecular fingerprints and using them to calculate molecular similarity.

Example S3:

```
'''
This is an example computer program in Python to visualize the maximum common structure between a group of molecules.
'''

from rdkit import Chem
from rdkit.Chem.Draw import MolsToGridImage
from rdkit.Chem import rdFMCS

# morphine, heroin, and fentanyl
mol1 = Chem.MolFromSmiles("CN1CC[C@]23[C@@H]4[C@H]1CC5=C2C(=C(C=C5)O)O[C@H]3[C@H](C=C4)O")
mol2 =
Chem.MolFromSmiles("CC(=O)O[C@H]1C=C[C@H]2[C@H]3CC4=C5[C@]2([C@H]1OC5=C(C=C4)OC(=O)C)CCN3C")
mol3 = Chem.MolFromSmiles("CCC(=O)N(C1CCN(CC1)CCC2=CC=CC=C2)C3=CC=CC=C3")

mols = [mol1,mol2,mol3]

res=rdFMCS.FindMCS(mols)

patt=Chem.MolFromSmarts(res.smartsString)
mols.append(patt)

im=MolsToGridImage(mols=mols, molsPerRow=4)
im.show()
```

Outcome of the script:

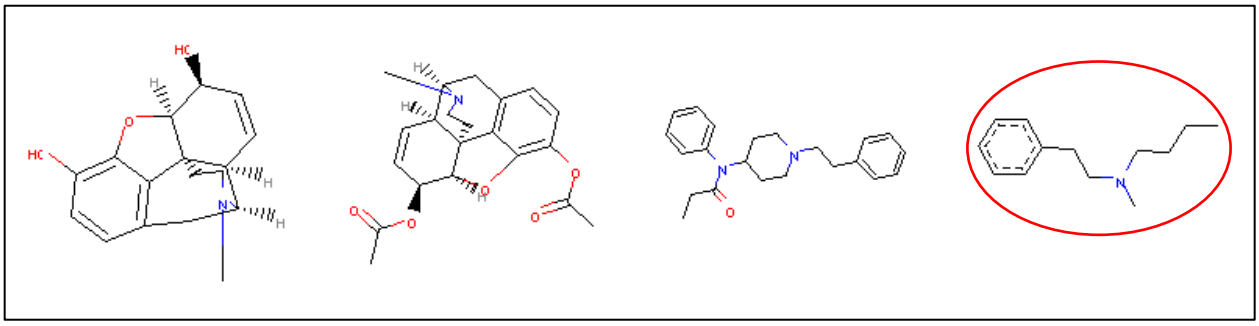


Figure S3. Example script to show the maximum common substructure (MCS, in the red circle) between morphine, heroin, and fentanyl (from left to right). Interestingly, the MCS pattern indicates some similarity (56% of heavy atoms in fentanyl are in similar substructure with morphine/heroin).

Further application to biophysical course materials: The FindMCS function in rdkit finds an MCS of two or more molecules (in this example, three molecules). According to rdkit, this function returns an MCSResult instance with information about the the SMARTS string which matches the identified MCS. If no MCS is found then the SMARTS is ''.

Students can visualize the MCS to gain insight into the molecular details of many biophysical phenomena. This example can be adopted to other teaching materials (e.g. to illustrate structure-activity relationship of biophysical molecules, to introduce the principle of host-guest binding in biology, etc.)

Example S4:

The script is available in the supporting materials for download.

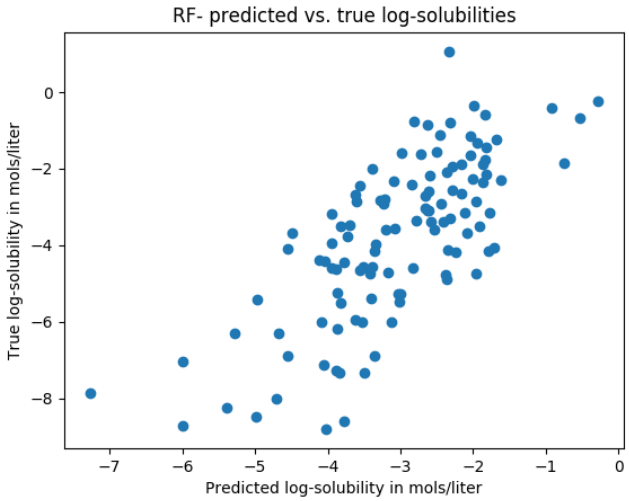


Figure S4. Comparison between the predicted and experimental aqueous solubility in our validation set, using the random forest ML method.

4. An additional example.

We provide an additional example to illustrate molecular fingerprints and applications related to biophysics. This example allows expansion to amino acids, nucleic acid bases, lipids, etc. (1) It can be formatted as a group activity, which ask students to work in groups to design molecular fingerprint to distinguish 20 natural amino acids. (2) The learning goal of this activity can be associated with the learning goal toward protein structures and functions in biophysics.

Table S2. An example fingerprint for 6 amino acids.

mol. \ FP keys	cccccc	NCCCC	c(c)(c)c	[CH3X4]
Alanine	0	0	0	1
Lysine	0	1	0	0
Phenylalanine	1	0	0	0
Serine	0	0	0	0
Tyrosine	1	0	0	0
Tryptophan	1	0	1	0

Note: For more unique SMARTS patterns for the fingerprint (FP) keys, a list is provided in the next page.

List of SMARTS patterns for amino acid side chains (Link to more SMARTS Examples)

Alanine side chain.

[CH3X4]

Arginine side chain.

[CH2X4][CH2X4][CH2X4][NHX3][CH0X3](=[NH2X3+,NHX2+0])[NH2X3]

Hits acid and conjugate base.

Aspargine side chain.

[CH2X4][CX3](=[OX1])[NX3H2]

Also hits Gln side chain when used alone.

Aspartate (or Aspartic acid) side chain.

[CH2X4][CX3](=[OX1])[OH0-,OH]

Hits acid and conjugate base. Also hits Glu side chain when used alone.

Cysteine side chain.

[CH2X4][SX2H,SX1H0-]

Hits acid and conjugate base

Glutamate (or Glutamic acid) side chain.

[CH2X4][CH2X4][CX3](=[OX1])[OH0-,OH]

Hits acid and conjugate base.

Glycine.

[$([$([NX3H2,NX4H3+]),$([NX3H]©©)][CX4H2][CX3](=[OX1])[OX2H,OX1-,N])]

Histidine side chain.

[CH2X4][#6X3]1:[$([#7X3H+,#7X2H0+0]:[#6X3H]:[#7X3H]),$([#7X3H])]:[#6X3H]:[$([#7X3H+,#7X2H0+0]:[#6X3H]:[#7X3H]),$([#7X3H])]:[#6X3H]1

Hits acid & conjugate base for either Nitrogen. Note that the Ns can be either ([(Cationic 3-connected with one H) or (Neutral 2-connected without any Hs)] where there is a second-neighbor who is [3-connected with one H]) or (3-connected with one H).

Isoleucine side chain.

[CHX4]([CH3X4])[CH2X4][CH3X4]

Leucine side chain.

[CH2X4][CHX4]([CH3X4])[CH3X4]

Lysine side chain.

[CH2X4][CH2X4][CH2X4][CH2X4][NX4+,NX3+0]

Acid and conjugate base

Methionine side chain.

[CH2X4][CH2X4][SX2][CH3X4]

Phenylalanine side chain.

[CH2X4][cX3]1[cX3H][cX3H][cX3H][cX3H][cX3H]1

Proline.

[$([NX3H,NX4H2+]),$([NX3](C)(C)(C))]1[CX4H]([CH2][CH2][CH2]1)[CX3](=[OX1])[OX2H,OX1-,N]

Serine side chain.

[CH2X4][OX2H]

Threonine side chain.

[CHX4]([CH3X4])[OX2H]

Tryptophan side chain.

[CH2X4][cX3]1[cX3H][nX3H][cX3]2[cX3H][cX3H][cX3H][cX3H][cX3]12

Tyrosine side chain.

[CH2X4][cX3]1[cX3H][cX3H][cX3]([OHX2,OH0X1-])[cX3H][cX3H]1

Acid and conjugate base

Valine side chain.

[CHX4]([CH3X4])[CH3X4]

5. Machine Learning Resources and Tutorials

SMILES: https://www.daylight.com/dayhtml/doc/theory/theory.smiles.html
Simplified molecular-input line-entry system, or SMILES, is a system for encoding molecular structures into one-dimensional strings. SMILES strings are useful because they are more easily parsed than 3D or 2D structures, and because motifs can be easily extracted.

SMARTS: https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html
SMILES arbitrary target specification, or SMARTS, specifies molecular substructures and motifs through a set of clear rules. Using this language, it is possible to quickly screen large databases of compounds (in the form of SMILES strings) for specific patterns.

RDKit: https://www.rdkit.org/docs/GettingStartedInPython.html
RDKit is an open-source cheminformatics software that allows users to construct, search, and manipulate SMILES strings.

Scikit-learn
Scikit-learn is a machine learning library for Python. It is easy to use and extremely well documented, making it particularly well-suited for those new to both machine learning and coding.

DeepChem: https://deepchem.io/docs/notebooks/index.html
DeepChem is a combination of chemical datasets and machine learning methods that enables users to quickly make predictions. Datasets fall under a variety of categories; biophysics datasets include PubChem BioAssay (PCBA), PDBBind, and binding results for a set of inhibitors of human β-secretase 1(BACE-1).

Chainer Chemistry: https://chainer-chemistry.readthedocs.io/en/latest/index.html
Chainer Chemistry is a machine learning library for Python that is geared towards biological and chemical problems. Like DeepChem, it includes deep-learning models as well as access to a number of datasets.

Rosalind: http://rosalind.info/problems/locations/
Rosalind provides tools, tutorials, and exercises that teach the user how to apply machine learning and other algorithms to bioinformatics through the use of Python.

Jupyter: https://jupyter.org/
Jupyter is a coding environment for users that prefer a traditional lab-notebook experience. It allows for easy organization of code, plots, and notes. It is highly modular, customizable, and was created with machine learning in mind.