_start_smiles_to_graph

May 29, 2020

1 Read in SMILES molecules

[2]:
```
import pandas as pd
from glob import glob
from rdkit import Chem
from rdkit.Chem import AllChem

%matplotlib inline

import matplotlib.pyplot as plt
```

[ ]:
```
mol_fs = glob('./tox21_dataset-master/compounds/*.tab')
print(mol_fs)
in_df = pd.read_csv(mol_fs[0], sep='\s+',header=0)
in_df.head()
```

2D Depections

[4]:
```
in_df.iloc[2].SMILES
template = Chem.MolFromSmiles(in_df.iloc[2].SMILES)
m = Chem.MolFromSmiles(in_df.iloc[2].SMILES)

AllChem.Compute2DCoords(template)
```

```
      ␣
↪---------------------------------------------------------------------------

       NameError                                 Traceback (most recent call␣
↪last)

       <ipython-input-4-c81d518c9505> in <module>
   ----> 1 in_df.iloc[2].SMILES
         2 template = Chem.MolFromSmiles(in_df.iloc[2].SMILES)
         3 m = Chem.MolFromSmiles(in_df.iloc[2].SMILES)
         4
         5 AllChem.Compute2DCoords(template)
```

```
NameError: name 'in_df' is not defined
```

[ ]:

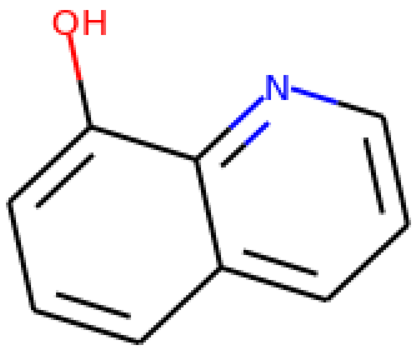
```
[5]: from rdkit.Chem.Draw import IPythonConsole
     from rdkit.Chem.Draw.MolDrawing import MolDrawing, DrawingOptions #Only needed␣
      ↪if modifying defaults
```

```
[6]: DrawingOptions.bondLineWidth=1.5
```

```
[6]: AllChem.GenerateDepictionMatching2DStructure(m, template)
```

```
[7]: m
```

[7]:



```
[8]: m.GetNumAtoms()
```

```
[8]: 11
```

The text representaiton of this molecule

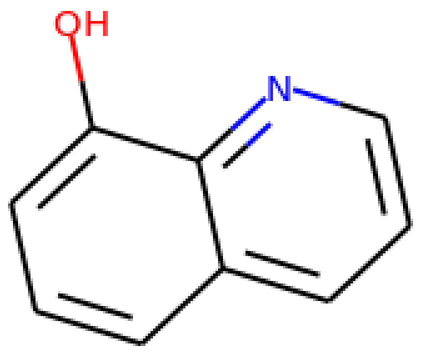
```
[9]: print(Chem.MolToMolBlock(m))
```

```

     RDKit          2D

 11 12  0  0  0  0  0  0  0  0999 V2000
   -1.7578    2.3964    0.0000 O   0  0  0  0  0  0  0  0  0  0  0  0
   -1.4243    0.9340    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
   -2.5240   -0.0861    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
   -2.1905   -1.5485    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
   -0.7572   -1.9909    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    0.3425   -0.9709    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    1.7758   -1.4133    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    2.8756   -0.3932    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    2.5421    1.0692    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    1.1088    1.5116    0.0000 N   0  0  0  0  0  0  0  0  0  0  0  0
    0.0090    0.4916    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
  1  2  1  0
  2  3  2  0
  3  4  1  0
  4  5  2  0
  5  6  1  0
  6  7  2  0
  7  8  1  0
  8  9  2  0
  9 10  1  0
 10 11  2  0
 11  2  1  0
 11  6  1  0
M  END
```

[10]:
```
DrawingOptions.includeAtomNumbers=True
```

[11]:
```
m = Chem.MolFromSmiles(in_df.iloc[2].SMILES)
m
```

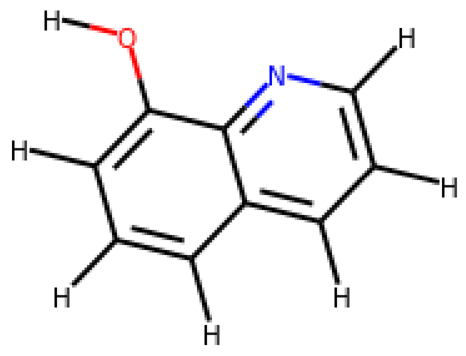
[11]:



[12]:
```
ibuH = AllChem.AddHs(m)
```

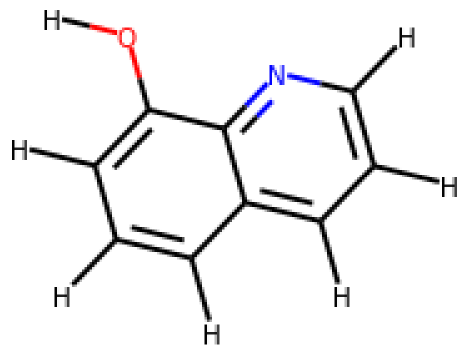
[13]:
```
DrawingOptions.includeAtomNumbers=False
ibuH
```

[13]:



[14]:
```
DrawingOptions.includeAtomNumbers=True
ibuH
```

[14]:



[15]:
```
adj = Chem.GetAdjacencyMatrix(m)
print(adj)
```

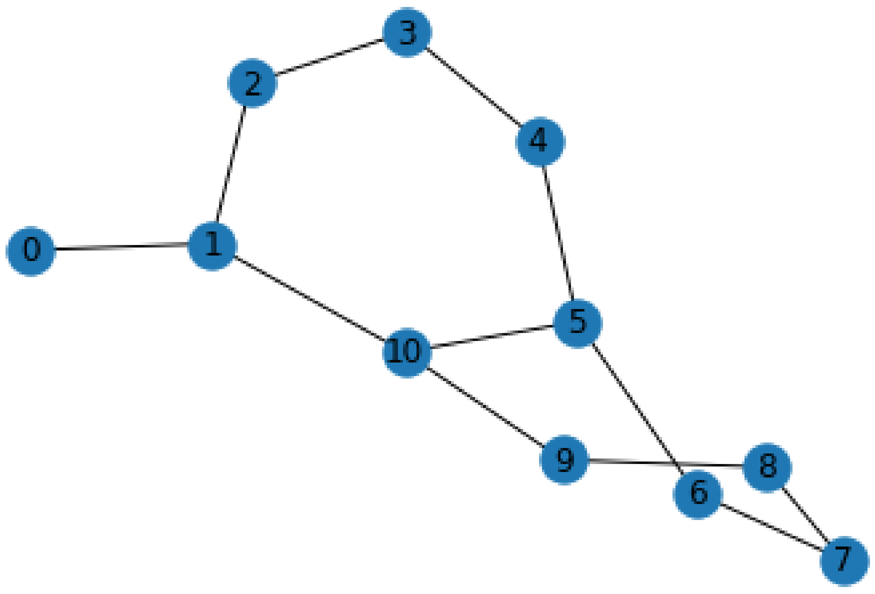
```
[[0 1 0 0 0 0 0 0 0 0 0]
 [1 0 1 0 0 0 0 0 0 0 1]
 [0 1 0 1 0 0 0 0 0 0 0]
 [0 0 1 0 1 0 0 0 0 0 0]
 [0 0 0 1 0 1 0 0 0 0 0]
 [0 0 0 0 1 0 1 0 0 0 1]
 [0 0 0 0 0 1 0 1 0 0 0]
 [0 0 0 0 0 0 1 0 1 0 0]
 [0 0 0 0 0 0 0 1 0 1 0]
 [0 0 0 0 0 0 0 0 1 0 1]
 [0 1 0 0 0 1 0 0 0 1 0]]
```

[16]:
```
def mol_to_nx(mol):
    G = nx.Graph()
    for atom in mol.GetAtoms():
        G.add_node(atom.GetIdx(),
                   atomic_num=atom.GetAtomicNum(),
                   formal_charge=atom.GetFormalCharge(),
                   chiral_tag=atom.GetChiralTag(),
                   hybridization=atom.GetHybridization(),
                   num_explicit_hs=atom.GetNumExplicitHs(),
                   is_aromatic=atom.GetIsAromatic())
    for bond in mol.GetBonds():
        G.add_edge(bond.GetBeginAtomIdx(),
                   bond.GetEndAtomIdx(),
                   bond_type=bond.GetBondType())
    return G
```

[4]:
```
import networkx as nx
G = nx.from_numpy_matrix(adj)
```

```
      ␣
↪---------------------------------------------------------------------------

    NameError                                 Traceback (most recent call␣
↪last)

    <ipython-input-4-db03cc705068> in <module>
      1 import networkx as nx
  ----> 2 G = nx.from_numpy_matrix(adj)


    NameError: name 'adj' is not defined
```

[22]:
```
plt.axis('off')
nx.draw_networkx(G)
G.edges()
df =pd.DataFrame(data=list(G.edges()))
```



```
[24]: df.to_csv('chem.g',sep='\t',header=False, index=False)
```

1.1 Vertex Replacement Grammars (graph transformation)

```
[19]: !pwd
```

```
/home/sal/gstax/gstx_start
```

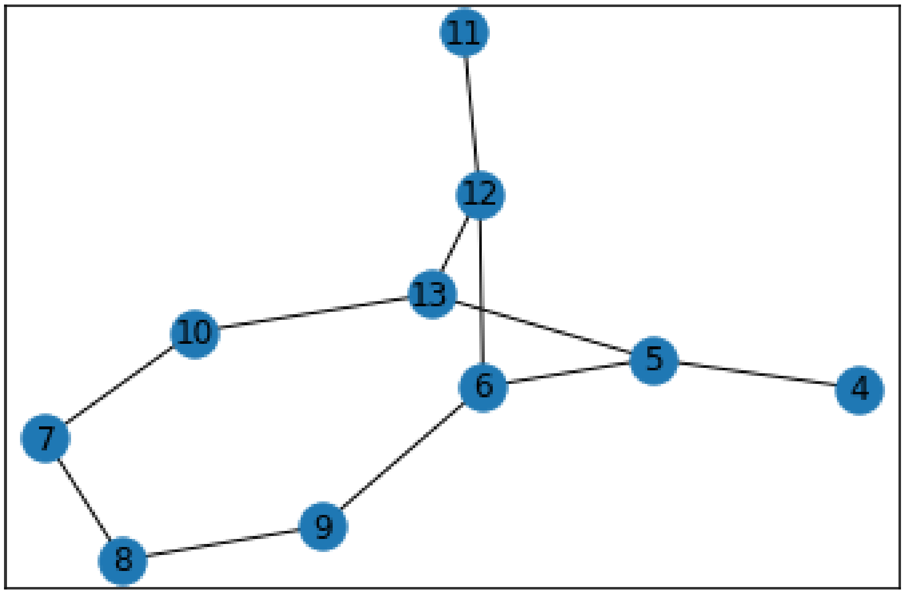
```
[15]: !python runner.py -g chem -o out -n 2
```

```
main
Graph: chem, n = 11, m = 12 read in 0.003s.
Running leiden clustering…
100%|                    |[00:00<00:00]
graph: chem, mu: 4, type: mu_level_dl clustering: leiden rules: 4(4) mdl:
194.624 bits generated in 0.015 secs

name: chem, original: 187.04439411935846, grammar: 194.62406251802892, time:
0.0178
[2 -> (4, 3), 3 -> (4, 3), 3 -> (3, 2), 0 -> (3, 4){2,3,3}]
Generated graph: (11, 12)
[(4, 5), (4, 7), (5, 6), (6, 9), (7, 14), (8, 9), (9, 10), (10, 11), (10, 14),
(11, 12), (12, 13), (13, 14)]
```

```
[7]: elist = [(4, 5), (4, 7), (5, 6), (6, 11), (7, 14), (8, 9), (8, 13), (9, 10),␣
     ↪(10, 11), (11, 14), (12, 13), (13, 14)]
     trnxfd_g = nx.from_edgelist(elist)
```

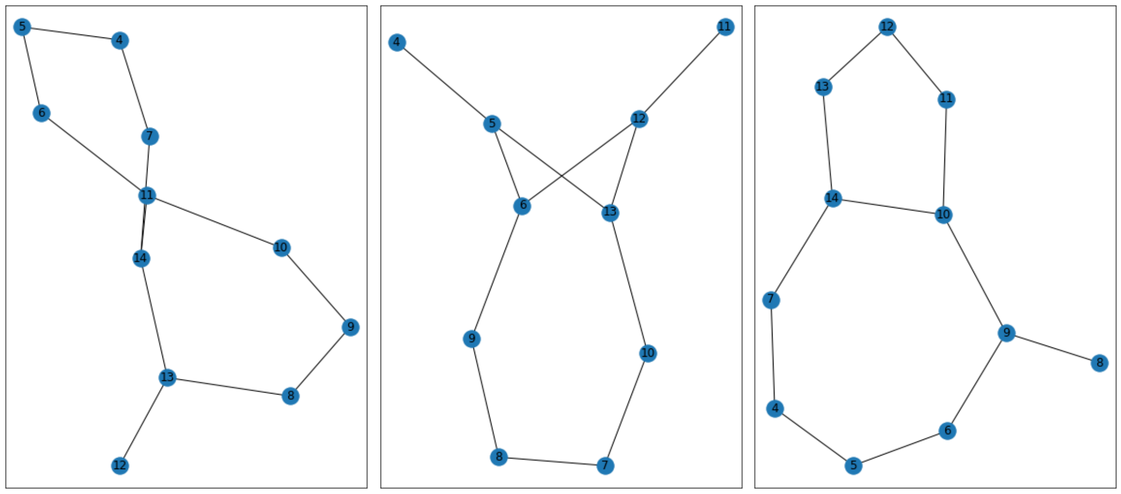
[8]:

```

```



[16]:
```
fig1, f1_axes = plt.subplots(ncols=3, nrows=1, constrained_layout=True,␣
 ↪figsize=(16, 7))
elist = [(4, 5), (4, 7), (5, 6), (6, 11), (7, 14), (8, 9), (8, 13), (9, 10),␣
 ↪(10, 11), (11, 14), (12, 13), (13, 14)]
trnxfd_g = nx.from_edgelist(elist)
nx.draw_networkx(trnxfd_g, ax=f1_axes[0])
f1_axes[0].set_xticks([])
f1_axes[0].set_yticks([])
elist = [(4, 5), (5, 6), (5, 13), (6, 9), (6, 12), (7, 8), (7, 10), (8, 9),␣
 ↪(10, 13), (11, 12), (12, 13)]
trnxfd_g = nx.from_edgelist(elist)
nx.draw_networkx(trnxfd_g, ax=f1_axes[1])
f1_axes[1].set_xticks([])
f1_axes[1].set_yticks([])
elist = [(4, 5), (4, 7), (5, 6), (6, 9), (7, 14), (8, 9), (9, 10), (10, 11),␣
 ↪(10, 14), (11, 12), (12, 13), (13, 14)]
trnxfd_g = nx.from_edgelist(elist)
nx.draw_networkx(trnxfd_g, ax=f1_axes[2])
f1_axes[2].set_xticks([])
f1_axes[2].set_yticks([])
```

[16]:
```
[]
```



1.2 Signals from Graphs

```
[3]: # 3,4 dimethilhexane CCC(C)C(C)CC
     template = Chem.MolFromSmiles("CCC(C)C(C)CC")
     m = Chem.MolFromSmiles("CCC(C)C(C)CC")
```

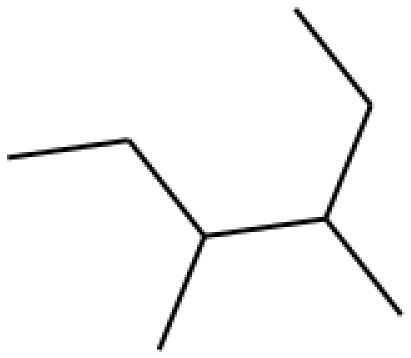
```
[7]: AllChem.Compute2DCoords(template)
```

```
[7]: 0
```

```
[8]: AllChem.GenerateDepictionMatching2DStructure(m, template)
```

```
[10]: DrawingOptions.includeAtomNumbers=True
      m
```

[10]:



For an example of derived signals go to the MathChem notebook