Subject: Re: Question about R-group occurences estimation
Posted by nbehrnd on Tue, 11 Feb 2020 13:53:18 GMT
View Forum Message <> Reply to Message

There are two possible difficulties to retrieve methyl groups as-such in a list of SMILES. For one, the characteristic of them is the (single) carbon atom and searching just for this is less identifying than the string of c1ccccc1 about benzene, for example. Second, probably there would be a need to add explicit hydrogens on all SMILES before these would be easier to identify (which may be done, e.g., with babel).

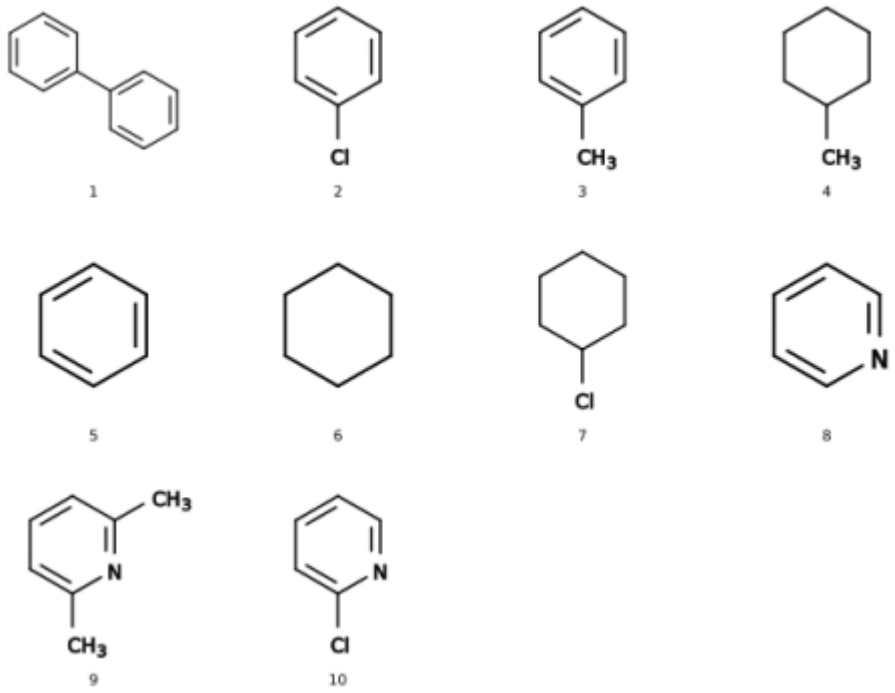
If you have access to Python, then the additional module by rdkit (http://rdkit.org/) may be quite helpful to querry your SMILES with SMARTS. With the test file of smiles_list.smi attached below, the identification of methyl groups (in SMART's convention, expressed as [CH3]) works fine both locally -- per SMILES entry -- as well as in counting the globally:

```
from rdkit import Chem
smiles_source = "smiles_list.smi"
grand_total = 0

# example pattern to identify and count:
functional_group = Chem.MolFromSmarts('[CH3]')  #  methyl group

# alternative examples:
#functional_group = Chem.MolFromSmarts('c1ccccn1')  # for a pyridine
#functional_group = Chem.MolFromSmarts('C1CCCCC1')  # for cyclohexane

with open(smiles_source, mode="r") as source_file:
    for index, line in enumerate(source_file, start=1):
        molecule = Chem.MolFromSmiles(line.strip())
        match = molecule.GetSubstructMatches(functional_group)

        print("{:3} matches in entry {:2}: {}.".format(
            len(match), index, line.strip()))

        grand_total += len(match)

print("\nIn total {} instances were identified.".format(grand_total))
```

As DataWarrior relies on java, the implementation of the «ErtlFunctionalGroupsFinder» described by Fritsch et al. (https://jcheminf.biomedcentral.com/articles/10.1186/s13321-019-0361-8, open access) equally may be of interest for Thomas.

File Attachments

1) example.png, downloaded 1362 times



2) listing.png, downloaded 1280 times

```
$ python example.py
  0 matches in entry  1: c1cc(ccc1)c2ccccc2.
  0 matches in entry  2: Clc1ccccc1.
  1 matches in entry  3: Cc1ccccc1.
  1 matches in entry  4: CC1CCCCC1.
  0 matches in entry  5: c1ccccc1.
  0 matches in entry  6: C1CCCCC1.
  0 matches in entry  7: ClC1CCCCC1.
  0 matches in entry  8: c1ccccn1.
  2 matches in entry  9: Cc1cccc(C)n1.
  0 matches in entry 10: Clc1ccccn1.

In total 4 instances were identified.
```

3) smiles_list.smi, downloaded 667 times
4) example.py, downloaded 652 times
