RDKit FilterCatalog Lightning Talk

Brian Kelley

NIBR Informatics

Novartis Institutes for BioMedical Research

RDKit UGM Zürich September 2015

NOVARTIS

Outline

- Background Information (What's a Filter? PAINS)
- Code Overview (Python, if you want C++, talk to me over beers)
- Odds and ends

What's a Filter?

- Do we REALLY want this compound?
- Lipinski
- PAINS (Removal of Pan Assay INterference Compounds)
  - NIH/ZINC etc.
- Forcefield atom typing (is the compound compatible with the ForceField)

FilterCatalog

Goals: Make easy use, easy.:

```
from rdkit.Chem.FilterCatalog import (
  FilterCatalogParams, FilterCatalog)

# add the PAINS catalog (can also use PAINS_A/B/C)
params = FilterCatalog.FilterCatalogParams(
   FilterCatalogParams.FilterCatalogs.PAINS)
catalog = FilterCatalog(params)

for mol in mol_supplier:
  matches = catalog.GetMatches(mol)
  if not matches:  … # not hit by pains filter
  else: print ( [entry.GetDescription() for entry in matches] )
```

FilterCatalog

Other things to do with a match:

```
for mol in mol_supplier:
  match = catalog.GetFirstMatch(mol)
  if match:
    # where are the atoms in the matched filter?
    for filtermatch in match.GetFilterMatches(mol):
      for filter_atomidx, atom_idx in filtermatch.atompairs:
          …
```

Why does a single entry have multiple filter matches?

- A catalog entry can be an arbitrarily complicated query.

FilterMatcher

Default is Smarts Pattern with min/max counts:

```
minCount = 1
aromMatcher = FilterCatalog.SmartsMatcher(
   "Aromatic carbon chain”, Chem.MolFromSmarts("c:c:c:c:c”), minCount)
```

Can be combined with arbitrary logic (and/or/not)

```
combined_matcher = FilterMatcherOps.And( aromMatcher, carboxylMatcher)
```

Then add as a catalog entry

```
entry = FilterCatalog.FilterCatalogEntry(”My Filter", combined_matcher)
fc = FilterCatalog.FilterCatalog()
fc.AddEntry(entry)
```

FilterMatcher

Or implement complicated schemes in pure python

```
class MWFilter(FilterCatalog.FilterMatcher):
        def __init__(self, minMw, maxMw):
            super(FilterCatalog, self).__init__("MW violation")
            self.minMw = minMw
            self.maxMw = maxMw

        def IsValid(self): return True

        def HasMatch(self, mol): # no atom idxs needed
            mw = rdMolDescriptors.CalcExactMolWt(mol)
            return not self.minMw <= mw <= self.maxMw

entry = FilterCatalog.FilterCatalogEntry("MW Violation", MWFilter(100,500))
```

Odds and Ends

- Adding a Not filter invalidates matching atom pairs, we are looking at a work-around (internally Novartis uses acceptance not rejection filters)
- FilterCatalogs are serializable (on most systems)
  - Uses the magical boost::serialize
  - FilterCatalog.FilterCatalogCanSerialize() checks for availability (RHEL5 currently not supported)
  - Python catalog entries currently cannot be serialized (we are close, but no cigar at this point)
- For pattern based queries, using a chemfp arena as a screenout filter can be a big win
  - Functional group filtering 3k second -> 500K second.

Acknowledgements

- Greg Landrum For RDKit and spending way too much time validating the PAINS filters themselves.
- boost::serialize (any one want to write a json target?)
- RDKit UGM