

NAME

AtomNeighborhoodsFingerprints.pl - Generate atom neighborhoods fingerprints for SD files

SYNOPSIS

AtomNeighborhoodsFingerprints.pl SDFfile(s)...

```
AtomNeighborhoodsFingerprints.pl [-a, --AtomIdentifierType AtomicInvariantsAtomTypes] [--AtomicInvariantsToUse
"AtomicInvariant,AtomicInvariant..."] [--FunctionalClassesToUse "FunctionalClass1,FunctionalClass2..."] [--CompoundID DataFieldName
or LabelPrefixString] [--CompoundIDLabel text] [--CompoundIDMode] [--DataFields "FieldLabel1,FieldLabel2,..."] [-d,
--DataFieldsMode All | Common | Specify | CompoundID] [-f, --Filter Yes | No] [--FingerprintsLabel text] [-h, --help] [-k,
--KeepLargestComponent Yes | No] [--MinNeighborhoodRadius number] [--MaxNeighborhoodRadius number] [--OutDelim
comma | tab | semicolon] [--output SD | text | both] [-o, --overwrite] [-q, --quote Yes | No] [-r, --root RootName] [-w,
--WorkingDir dirname] SDFfile(s)...
```

DESCRIPTION

Generate atom neighborhoods fingerprints [Ref 53-56, Ref 73] for *SDFfile(s)* and create appropriate SD or CSV/TSV text file(s) containing fingerprints vector strings corresponding to molecular fingerprints.

Multiple SDFfile names are separated by spaces. The valid file extensions are *.sdf* and *.sd*. All other file names are ignored. All the SD files in a current directory can be specified either by **.sdf* or the current directory name.

The current release of MayaChemTools supports generation of atom neighborhoods fingerprints corresponding to following -a, --AtomIdentifierTypes:

```
AtomicInvariantsAtomTypes, DREIDINGAtomTypes, EStateAtomTypes,
FunctionalClassAtomTypes, MMFF94AtomTypes, SLogPAtomTypes,
SYBYLAtomTypes, TPSAAtomTypes, UFFAtomTypes
```

Based on the values specified for -a, --AtomIdentifierType and --AtomicInvariantsToUse, initial atom types are assigned to all non-hydrogen atoms in a molecule. Using atom neighborhoods around each non-hydrogen central atom corresponding to radii between specified values --MinNeighborhoodRadius and --MaxNeighborhoodRadius, unique atom types at each radii level are counted and an atom neighborhood identifier is generated.

The format of an atom neighborhood identifier around a central non-hydrogen atom at a specific radius is:

```
NR<n>-<AtomType>-ATC<n>
```

```
NR: Neighborhood radius
AtomType: Assigned atom type
ATC: Atom type count
```

The atom neighborhood identifier for a non-hydrogen central atom corresponding to all specified radii is generated by concatenating neighborhood identifiers at each radii by colon as a delimiter:

```
NR<n>-<AtomType>-ATC<n>:NR<n>-<AtomType>-ATC<n>:...
```

The atom neighborhood identifiers for all non-hydrogen central atoms at all specified radii are concatenated using space as a delimiter and constitute atom neighborhood fingerprint of the molecule.

The current release of MayaChemTools generates the following types of atom neighborhoods fingerprints vector strings:

```
FingerprintsVector;AtomNeighborhoods:AtomicInvariantsAtomTypes;23;Alpha
NumericalValues;ValuesString;NR0-C.X1.BO1.H3-ATC1:NR1-C.X2.BO2.H2-ATC1:
NR2-C.X3.BO3.H1-ATC1:NR0-C.X1.BO1.H3-ATC1:NR1-C.X2.BO2.H2-ATC1:NR2-C.X3
.BO3.H1-ATC1:NR0-C.X1.BO1.H3-ATC1:NR1-C.X3.BO4-ATC1:NR2-N.X2.BO2.H1-ATC
1:NR2-O.X1.BO2-ATC1:NR0-C.X2.BO2.H2-ATC1:NR1-C.X1.BO1.H3-ATC1:NR1-C.X3
.BO3.H1-ATC1:NR2-C.X2.BO2.H2-ATC1:NR2-C.X3.BO3.H1-ATC1:NR0-C.X2.BO2.H...
```

```
FingerprintsVector;AtomNeighborhoods:DREIDINGAtomTypes;23;AlphaNumerical
Values;ValuesString;NR0-C_2-ATC1:NR1-C_3-ATC1:NR1-N_3-ATC1:NR1-O_2-ATC1
:NR2-C_3-ATC1:NR0-C_2-ATC1:NR1-C_3-ATC1:NR1-O_2-ATC1:NR1-O_3-ATC1:NR2-C
_3-ATC2:NR0-C_2-ATC1:NR1-N_2-ATC1:NR1-N_3-ATC2:NR2-C_3-ATC1:NR0-C_3-ATC
1:NR1-C_2-ATC1:NR1-C_3-ATC2:NR2-C_3-ATC2:NR2-O_2-ATC1:NR2-O_3-ATC2:NR0-
C_3-ATC1:NR1-C_2-ATC1:NR2-N_3-ATC1:NR2-O_2-ATC1:NR0-C_3-ATC1:NR1-C_3...
```

```
FingerprintsVector;AtomNeighborhoods:EStateAtomTypes;23;AlphaNumerical
Values;ValuesString;NR0-dNH-ATC1:NR1-dssC-ATC1:NR2-sNH2-ATC1:NR2-ssNH-A
TC1:NR0-dO-ATC1:NR1-dssC-ATC1:NR2-sCH3-ATC1:NR2-ssNH-ATC1:NR0-dO-ATC1:N
R1-dssC-ATC1:NR2-sOH-ATC1:NR2-sssCH-ATC1:NR0-dssC-ATC1:NR1-dNH-ATC1:NR1
-sNH2-ATC1:NR1-ssNH-ATC1:NR2-sssCH-ATC1:NR0-dssC-ATC1:NR1-dO-ATC1:NR1-s
CH3-ATC1:NR1-ssNH-ATC1:NR2-sssCH-ATC1:NR0-dssC-ATC1:NR1-dO-ATC1:NR1-...
```

```
FingerprintsVector;AtomNeighborhoods:FunctionalClassAtomTypes;23;Alpha
NumericalValues;ValuesString;NR0-HBA-ATC1:NR1-NI-ATC1:NR2-HBA.HBD-ATC1:
NR2-None-ATC1 NR0-HBA-ATC1:NR1-None-ATC1:NR2-HBD-ATC1:NR2-None-ATC1 NR0
-HBA.HBD-ATC1:NR1-NI-ATC1:NR2-HBA-ATC1:NR2-None-ATC1 NR0-HBA.HBD-ATC1:N
R1-None-ATC1:NR2-None-ATC2 NR0-HBD-ATC1:NR1-None-ATC1:NR2-HBD-ATC1:NR2-
HBD.PI-ATC1 NR0-HBD-ATC1:NR1-None-ATC2:NR2-HBA-ATC1:NR2-None-ATC3 NR...
```

```
FingerprintsVector;AtomNeighborhoods:MMFF94AtomTypes;23;AlphaNumerical
Values;ValuesString;NR0-C=ON-ATC1:NR1-CR-ATC1:NR1-NC=O-ATC1:NR1-O=CN-AT
C1:NR2-CR-ATC1 NR0-CGD-ATC1:NR1-N=C-ATC1:NR1-NC=N-ATC2:NR2-CR-ATC1 NR0-
COO-ATC1:NR1-CR-ATC1:NR1-O=CO-ATC1:NR1-OC=O-ATC1:NR2-CR-ATC2 NR0-CR-ATC
1:NR1-C=ON-ATC1:NR2-NC=O-ATC1:NR2-O=CN-ATC1 NR0-CR-ATC1:NR1-COO-ATC1:NR
1-CR-ATC2:NR2-CR-ATC2:NR2-O=CO-ATC1:NR2-OC=O-ATC1:NR2-OR-ATC1 NR0-CR...
```

```
FingerprintsVector;AtomNeighborhoods:SLogPAtomTypes;23;AlphaNumerical
Values;ValuesString;NR0-C1-ATC1:NR1-C1-ATC1:NR1-C2-ATC1:NR2-C1-ATC1:NR2
-CS-ATC1 NR0-C1-ATC1:NR1-C1-ATC1:NR1-C2-ATC1:NR2-C1-ATC1:NR2-CS-ATC1 NR
0-C1-ATC1:NR1-C1-ATC1:NR2-C2-ATC1 NR0-C1-ATC1:NR1-C1-ATC1:NR2-C2-ATC1 N
R0-C1-ATC1:NR1-C2-ATC1:NR1-CS-ATC1:NR2-C2-ATC1:NR2-C5-ATC1:NR2-CS-ATC1:
NR2-N2-ATC1 NR0-C1-ATC1:NR1-C5-ATC1:NR2-N2-ATC1:NR2-O9-ATC1 NR0-C2-A...
```

```
FingerprintsVector;AtomNeighborhoods:SYBYLAtomTypes;23;AlphaNumerical
Values;ValuesString;NR0-C.2-ATC1:NR1-C.3-ATC1:NR1-N.am-ATC1:NR1-O.2-ATC
1:NR2-C.3-ATC1 NR0-C.2-ATC1:NR1-C.3-ATC1:NR1-O.co2-ATC2:NR2-C.3-ATC2 NR
0-C.3-ATC1:NR1-C.2-ATC1:NR1-C.3-ATC2:NR2-C.3-ATC2:NR2-O.3-ATC1:NR2-O.co
2-ATC2 NR0-C.3-ATC1:NR1-C.2-ATC1:NR2-N.am-ATC1:NR2-O.2-ATC1 NR0-C.3-ATC
1:NR1-C.3-ATC1:NR2-C.3-ATC1 NR0-C.3-ATC1:NR1-C.3-ATC1:NR2-C.3-ATC1 N...
```

```
FingerprintsVector;AtomNeighborhoods:TPSAAtomTypes;23;AlphaNumerical
Values;ValuesString;NR0-N10-ATC1:NR1-None-ATC1:NR2-N7-ATC1:NR2-N9-ATC1
NR0-N7-ATC1:NR1-None-ATC2:NR2-N10-ATC1:NR2-N9-ATC1:NR2-None-ATC2 NR0-N7
-ATC1:NR1-None-ATC2:NR2-None-ATC3:NR2-O3-ATC1 NR0-N9-ATC1:NR1-None-ATC1
:NR2-N10-ATC1:NR2-N7-ATC1 NR0-None-ATC1:NR1-N10-ATC1:NR1-N7-ATC1:NR1-N9
-ATC1:NR2-None-ATC1 NR0-None-ATC1:NR1-N7-ATC1:NR1-None-ATC1:NR1-O3-A...
```

```
FingerprintsVector;AtomNeighborhoods:UFFAAtomTypes;23;AlphaNumerical
Values;ValuesString;NR0-C_2-ATC1:NR1-C_3-ATC1:NR1-N_3-ATC1:NR1-O_2-ATC1
:NR2-C_3-ATC1 NR0-C_2-ATC1:NR1-C_3-ATC1:NR1-O_2-ATC1:NR1-O_3-ATC1:NR2-C
_3-ATC2 NR0-C_2-ATC1:NR1-N_2-ATC1:NR1-N_3-ATC2:NR2-C_3-ATC1 NR0-C_3-ATC
1:NR1-C_2-ATC1:NR1-C_3-ATC2:NR2-C_3-ATC2:NR2-O_2-ATC1:NR2-O_3-ATC2 NR0-
C_3-ATC1:NR1-C_2-ATC1:NR2-N_3-ATC1:NR2-O_2-ATC1 NR0-C_3-ATC1:NR1-C_3...
```

OPTIONS

-a, --AtomIdentifierType *AtomicInvariantsAtomTypes* | *DREIDINGAtomTypes* | *EStateAtomTypes* | *FunctionalClassAtomTypes* | *MMFF94AtomTypes* | *SLogPAtomTypes* | *SYBYLAtomTypes* | *TPSAAtomTypes* | *UFFAAtomTypes*

Specify atom identifier type to use for assignment of initial atom identifier to non-hydrogen atoms during calculation of atom neighborhoods fingerprints. Possible values in the current release are: *AtomicInvariantsAtomTypes*, *DREIDINGAtomTypes*, *EStateAtomTypes*, *FunctionalClassAtomTypes*, *MMFF94AtomTypes*, *SLogPAtomTypes*, *SYBYLAtomTypes*, *TPSAAtomTypes*, *UFFAAtomTypes*. Default value: *AtomicInvariantsAtomTypes*.

--AtomicInvariantsToUse "*AtomicInvariant,AtomicInvariant...*"

This value is used during *AtomicInvariantsAtomTypes* value of a, --AtomIdentifierType option. It's a list of comma separated valid atomic invariant atom types.

Possible values for atomic invariants are: *AS, X, BO, LBO, SB, DB, TB, H, Ar, RA, FC, MN, SM*. Default value: *AS,X,BO,H,FC*.

The atomic invariants abbreviations correspond to:

AS = Atom symbol corresponding to element symbol

X<n> = Number of non-hydrogen atom neighbors or heavy atoms

BO<n> = Sum of bond orders to non-hydrogen atom neighbors or heavy atoms

LBO<n> = Largest bond order of non-hydrogen atom neighbors or heavy atoms

SB<n> = Number of single bonds to non-hydrogen atom neighbors or heavy atoms

DB<n> = Number of double bonds to non-hydrogen atom neighbors or heavy atoms

TB<n> = Number of triple bonds to non-hydrogen atom neighbors or heavy atoms

H<n> = Number of implicit and explicit hydrogens for atom

Ar = Aromatic annotation indicating whether atom is aromatic

RA = Ring atom annotation indicating whether atom is a ring

FC<+n/-n> = Formal charge assigned to atom

MN<n> = Mass number indicating isotope other than most abundant isotope

SM<n> = Spin multiplicity of atom. Possible values: 1 (singlet), 2 (doublet) or 3 (triplet)

Atom type generated by AtomTypes::AtomicInvariantsAtomTypes class corresponds to:

AS.X<n>.BO<n>.LBO<n>.<SB><n>.<DB><n>.<TB><n>.H<n>.Ar.RA.FC<+n/-n>.MN<n>.SM<n>

Except for AS which is a required atomic invariant in atom types, all other atomic invariants are optional. Atom type specification doesn't include atomic invariants with zero or undefined values.

In addition to usage of abbreviations for specifying atomic invariants, the following descriptive words are also allowed:

X : NumOfNonHydrogenAtomNeighbors or NumOfHeavyAtomNeighbors
 BO : SumOfBondOrdersToNonHydrogenAtoms or SumOfBondOrdersToHeavyAtoms
 LBO : LargestBondOrderToNonHydrogenAtoms or LargestBondOrderToHeavyAtoms
 SB : NumOfSingleBondsToNonHydrogenAtoms or NumOfSingleBondsToHeavyAtoms
 DB : NumOfDoubleBondsToNonHydrogenAtoms or NumOfDoubleBondsToHeavyAtoms
 TB : NumOfTripleBondsToNonHydrogenAtoms or NumOfTripleBondsToHeavyAtoms
 H : NumOfImplicitAndExplicitHydrogens
 Ar : Aromatic
 RA : RingAtom
 FC : FormalCharge
 MN : MassNumber
 SM : SpinMultiplicity

AtomTypes::AtomicInvariantsAtomTypes module is used to assign atomic invariant atom types.

--FunctionalClassesToUse "*FunctionalClass1,FunctionalClass2...*"

This value is used during *FunctionalClassAtomTypes* value of a, --AtomIdentifierType option. It's a list of comma separated valid functional classes.

Possible values for atom functional classes are: *Ar, CA, H, HBA, HBD, Hal, NI, PI, RA*. Default value [Ref 24]: *HBD,HBA,PI,NI,Ar,Hal*.

The functional class abbreviations correspond to:

HBD: HydrogenBondDonor
 HBA: HydrogenBondAcceptor
 PI : PositivelyIonizable
 NI : NegativelyIonizable
 Ar : Aromatic
 Hal : Halogen
 H : Hydrophobic
 RA : RingAtom
 CA : ChainAtom

Functional class atom type specification for an atom corresponds to:

Ar.CA.H.HBA.HBD.Hal.NI.PI.RA

AtomTypes::FunctionalClassAtomTypes module is used to assign functional class atom types. It uses following definitions [Ref 60-61, Ref 65-66]:

HydrogenBondDonor: NH, NH2, OH
 HydrogenBondAcceptor: N[!H], O
 PositivelyIonizable: +, NH2
 NegativelyIonizable: -, C(=O)OH, S(=O)OH, P(=O)OH

--CompoundID *DataFieldName* or *LabelPrefixString*

This value is --CompoundIDMode specific and indicates how compound ID is generated.

For *DataField* value of --CompoundIDMode option, it corresponds to datafield label name whose value is used as compound ID; otherwise, it's a prefix string used for generating compound IDs like LabelPrefixString<Number>. Default value, *Cmpd*, generates compound IDs which look like Cmpd<Number>.

Examples for *DataField* value of --CompoundIDMode:

MolID
 ExtReg

Examples for *LabelPrefix* or *MolNameOrLabelPrefix* value of --CompoundIDMode:

Compound

The value specified above generates compound IDs which correspond to Compound<Number> instead of default value of Cmpd<Number>.

--CompoundIDLabel *text*

Specify compound ID column label for CSV/TSV text file(s) used during *CompoundID* value of --DataFieldsMode option.
Default: *CompoundID*.

--CompoundIDMode *DataField | MolName | LabelPrefix | MolNameOrLabelPrefix*

Specify how to generate compound IDs and write to CSV/TSV text file(s) along with generated fingerprints for *text | both* values of --output option: use a *SDFFile(s)* datafield value; use molname line from *SDFFile(s)*; generate a sequential ID with specific prefix; use combination of both MolName and LabelPrefix with usage of LabelPrefix values for empty molname lines.
Possible values: *DataField | MolName | LabelPrefix | MolNameOrLabelPrefix*. Default: *LabelPrefix*.

For *MolNameAndLabelPrefix* value of --CompoundIDMode, molname line in *SDFFile(s)* takes precedence over sequential compound IDs generated using *LabelPrefix* and only empty molname values are replaced with sequential compound IDs.

This is only used for *CompoundID* value of --DataFieldsMode option.

--DataFields "*FieldLabel1,FieldLabel2,...*"

Comma delimited list of *SDFFile(s)* data fields to extract and write to CSV/TSV text file(s) along with generated fingerprints for *text | both* values of --output option.

This is only used for *Specify* value of --DataFieldsMode option.

Examples:

```
Extreg  
MolID,CompoundName
```

-d, --DataFieldsMode *All | Common | Specify | CompoundID*

Specify how data fields in *SDFFile(s)* are transferred to output CSV/TSV text file(s) along with generated fingerprints for *text | both* values of --output option: transfer all SD data field; transfer SD data files common to all compounds; extract specified data fields; generate a compound ID using molname line, a compound prefix, or a combination of both. Possible values: *All | Common | specify | CompoundID*. Default value: *CompoundID*.

-f, --Filter *Yes | No*

Specify whether to check and filter compound data in *SDFFile(s)*. Possible values: *Yes or No*. Default value: *Yes*.

By default, compound data is checked before calculating fingerprints and compounds containing atom data corresponding to non-element symbols or no atom data are ignored.

--FingerprintsLabel *text*

SD data label or text file column label to use for fingerprints string in output SD or CSV/TSV text file(s) specified by --output. Default value: *AtomNeighborhoodsFingerprints*.

-h, --help

Print this help message.

-k, --KeepLargestComponent *Yes | No*

Generate fingerprints for only the largest component in molecule. Possible values: *Yes or No*. Default value: *Yes*.

For molecules containing multiple connected components, fingerprints can be generated in two different ways: use all connected components or just the largest connected component. By default, all atoms except for the largest connected component are deleted before generation of fingerprints.

--MinNeighborhoodRadius *number*

Minimum atom neighborhood radius for generating atom neighborhoods. Default value: *0*. Valid values: positive integers and less than --MaxNeighborhoodRadius. Neighborhood radius of zero corresponds to list of non-hydrogen atoms.

--MaxNeighborhoodRadius *number*

Maximum atom neighborhood radius for generating atom neighborhoods. Default value: *2*. Valid values: positive integers and greater than --MinNeighborhoodRadius.

--OutDelim *comma | tab | semicolon*

Delimiter for output CSV/TSV text file(s). Possible values: *comma, tab, or semicolon* Default value: *comma*.

--output *SD | text | both*

Type of output files to generate. Possible values: *SD, text, or both*. Default value: *text*.

-o, --overwrite

Overwrite existing files.

-q, --quote *Yes | No*

Put quote around column values in output CSV/TSV text file(s). Possible values: *Yes or No*. Default value: *Yes*.

-r, --root *RootName*

New file name is generated using the root: <Root>.<Ext>. Default for new file names:

<SDFileName><AtomNeighborhoodsFP>.<Ext>. The file type determines <Ext> value. The sdf, csv, and tsv <Ext> values are used for SD, comma/semicolon, and tab delimited text files, respectively. This option is ignored for multiple input files.

`-w, --WorkingDir DirName`

Location of working directory. Default: current directory.

EXAMPLES

To generate atom neighborhoods fingerprints corresponding to atom neighborhood radii from 0 to 2 using atomic invariants atom types in vector string format and create a SampleANFP.csv file containing sequential compound IDs along with fingerprints vector strings data, type:

```
% AtomNeighborhoodsFingerprints.pl -r SampleANFP -o Sample.sdf
```

To generate atom neighborhoods fingerprints corresponding to atom neighborhood radii from 0 to 2 using DREIDING atom types in vector string format and create a SampleANFP.csv file containing sequential compound IDs along with fingerprints vector strings data, type:

```
% AtomNeighborhoodsFingerprints.pl -a DREIDINGAtomTypes -r SampleANFP  
-o Sample.sdf
```

To generate atom neighborhoods fingerprints corresponding to atom neighborhood radii from 0 to 2 using EStateAtomTypes types in vector string format and create a SampleANFP.csv file containing sequential compound IDs along with fingerprints vector strings data, type:

```
% AtomNeighborhoodsFingerprints.pl -a EStateAtomTypes -r SampleANFP  
-o Sample.sdf
```

To generate atom neighborhoods fingerprints corresponding to atom neighborhood radii from 0 to 2 using SYBYL atom types in vector string format and create a SampleANFP.csv file containing sequential compound IDs along with fingerprints vector strings data, type:

```
% AtomNeighborhoodsFingerprints.pl -a SYBYLAtomTypes -r SampleANFP  
-o Sample.sdf
```

To generate atom neighborhoods fingerprints corresponding to atom neighborhood radii from 0 to 2 using FunctionalClass atom types in vector string format and create a SampleANFP.csv file containing sequential compound IDs along with fingerprints vector strings data, type:

```
% AtomNeighborhoodsFingerprints.pl -a FunctionalClassAtomTypes  
-r SampleANFP -o Sample.sdf
```

To generate atom neighborhoods fingerprints corresponding to atom neighborhood radii from 0 to 2 using MMFF94 atom types in vector string format and create a SampleANFP.csv file containing sequential compound IDs along with fingerprints vector strings data, type:

```
% AtomNeighborhoodsFingerprints.pl -a MMFF94AtomTypes -r SampleANFP  
-o Sample.sdf
```

To generate atom neighborhoods fingerprints corresponding to atom neighborhood radii from 0 to 2 using SLogP atom types in vector string format and create a SampleANFP.csv file containing sequential compound IDs along with fingerprints vector strings data, type:

```
% AtomNeighborhoodsFingerprints.pl -a SLogPAtomTypes -r SampleANFP  
-o Sample.sdf
```

To generate atom neighborhoods fingerprints corresponding to atom neighborhood radii from 0 to 2 using SYBYL atom types in vector string format and create a SampleANFP.csv file containing sequential compound IDs along with fingerprints vector strings data, type:

```
% AtomNeighborhoodsFingerprints.pl -a SYBYLAtomTypes -r SampleANFP  
-o Sample.sdf
```

To generate atom neighborhoods fingerprints corresponding to atom neighborhood radii from 0 to 2 using TPSA atom types in vector string format and create a SampleANFP.csv file containing sequential compound IDs along with fingerprints vector strings data, type:

```
% AtomNeighborhoodsFingerprints.pl -a TPSAAtomTypes -r SampleANFP  
-o Sample.sdf
```

To generate atom neighborhoods fingerprints corresponding to atom neighborhood radii from 0 to 2 using UFF atom types in vector

string format and create a SampleANFP.csv file containing sequential compound IDs along with fingerprints vector strings data, type:

```
% AtomNeighborhoodsFingerprints.pl -a UFFAtomTypes -r SampleANFP
-o Sample.sdf
```

To generate atom neighborhoods fingerprints corresponding to atom neighborhood radii from 0 to 2 using atomic invariants atom types in vector string format and create both SampleANFP.csv and SampleANFP.sdf files containing sequential compound IDs in CSV file along with fingerprints vector strings data, type:

```
% AtomNeighborhoodsFingerprints.pl --output both -r SampleANFP
-o Sample.sdf
```

To generate atom neighborhoods fingerprints corresponding to atom neighborhood radii from 1 to 3 using atomic invariants atom types in vector string format and create a SampleANFP.csv file containing sequential compound IDs along with fingerprints vector strings data, type:

```
% AtomNeighborhoodsFingerprints.pl -a AtomicInvariantsAtomTypes
--MinNeighborhoodRadius 1 --MaxNeighborhoodRadius 3 -r SampleANFP
-o Sample.sdf
```

To generate atom neighborhoods fingerprints corresponding to atom neighborhood radii from 0 to 2 using only AS,X atomic invariants atom types in vector string format and create a SampleANFP.csv file containing sequential compound IDs along with fingerprints vector strings data, type:

```
% AtomNeighborhoodsFingerprints.pl -a AtomicInvariantsAtomTypes
--AtomicInvariantsToUse "AS,X" --MinNeighborhoodRadius 0
--MaxNeighborhoodRadius 3 -r SampleANFP -o Sample.sdf
```

To generate atom neighborhoods fingerprints corresponding to atom neighborhood radii from 0 to 2 using atomic invariants atom types in vector string format and create a SampleANFP.csv file containing compound ID from molecule name line along with fingerprints vector strings data, type:

```
% AtomNeighborhoodsFingerprints.pl -a AtomicInvariantsAtomTypes
--DataFieldsMode CompoundID --CompoundIDMode MolName
-r SampleANFP -o Sample.sdf
```

To generate atom neighborhoods fingerprints corresponding to atom neighborhood radii from 0 to 2 using atomic invariants atom types in vector string format and create a SampleANFP.csv file containing compound IDs using specified data field along with fingerprints vector strings data, type:

```
% AtomNeighborhoodsFingerprints.pl -a AtomicInvariantsAtomTypes
--DataFieldsMode CompoundID --CompoundIDMode DataField --CompoundID
Mol_ID -r SampleANFP -o Sample.sdf
```

To generate atom neighborhoods fingerprints corresponding to atom neighborhood radii from 0 to 2 using atomic invariants atom types in vector string format and create a SampleANFP.csv file containing compound ID using combination of molecule name line and an explicit compound prefix along with fingerprints vector strings data, type:

```
% AtomNeighborhoodsFingerprints.pl -a AtomicInvariantsAtomTypes
--DataFieldsMode CompoundID --CompoundIDMode MolnameOrLabelPrefix
--CompoundID Cmpd --CompoundIDLabel MolID -r SampleANFP -o Sample.sdf
```

To generate atom neighborhoods fingerprints corresponding to atom neighborhood radii from 0 to 2 using atomic invariants atom types in vector string format and create a SampleANFP.csv file containing specific data fields columns along with fingerprints vector strings data, type:

```
% AtomNeighborhoodsFingerprints.pl -a AtomicInvariantsAtomTypes
--DataFieldsMode Specify --DataFields Mol_ID -r SampleANFP
-o Sample.sdf
```

To generate atom neighborhoods fingerprints corresponding to atom neighborhood radii from 0 to 2 using atomic invariants atom types in vector string format and create a SampleANFP.csv file containing common data fields columns along with fingerprints vector strings data, type:

```
% AtomNeighborhoodsFingerprints.pl -a AtomicInvariantsAtomTypes
--DataFieldsMode Common -r SampleANFP -o Sample.sdf
```

To generate atom neighborhoods fingerprints corresponding to atom neighborhood radii from 0 to 2 using atomic invariants atom types in vector string format and create both SampleANFP.csv and SampleANFP.sdf files containing all data fields columns in CSV file along with fingerprints data, type:

```
% AtomNeighborhoodsFingerprints.pl -a AtomicInvariantsAtomTypes
--DataFieldsMode All --output both -r SampleANFP
-o Sample.sdf
```

AUTHOR

Manish Sud <msud@san.rr.com>

SEE ALSO

InfoFingerprintsSDFiles.pl, InfoFingerprintsTextFiles.pl, SimilarityMatrixSDFiles.pl, SimilarityMatrixTextFiles.pl, ExtendedConnectivityFingerprints.pl, MACCSKeysFingerprints.pl, PathLengthFingerprints.pl, TopologicalAtomPairsFingerprints.pl, TopologicalAtomTorsionsFingerprints.pl, TopologicalPharmacophoreAtomPairsFingerprints.pl, TopologicalPharmacophoreAtomTripletsFingerprints.pl

COPYRIGHT

Copyright (C) 2004-2010 Manish Sud. All rights reserved.

This file is part of MayaChemTools.

MayaChemTools is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.