

NAME

PathLengthFingerprints.pl - Generate atom path length based fingerprints for SD files

SYNOPSIS

PathLengthFingerprints.pl SDFfile(s)...

```
PathLengthFingerprints.pl [--AromaticityModel AromaticityModelType] [-a, --AtomIdentifierType
AtomicInvariantsAtomTypes] [--AtomicInvariantsToUse "AtomicInvariant1,AtomicInvariant2..."] [
--FunctionalClassesToUse "FunctionalClass1,FunctionalClass2..."] [--BitsOrder Ascending | Descending] [-b,
--BitStringFormat BinaryString | HexadecimalString] [--CompoundID DataFieldName or LabelPrefixString] [
--CompoundIDLabel text] [--CompoundIDMode DataField | MolName | LabelPrefix | MolNameOrLabelPrefix] [
--DataFields "FieldLabel1,FieldLabel2,..." ] [-d, --DataFieldsMode All | Common | Specify | CompoundID] [
--DetectAromaticity Yes | No] [-f, --Filter Yes | No] [--FingerprintsLabel text] [--fold Yes | No] [
--FoldedSize number] [-h, --help] [-i, --IgnoreHydrogens Yes | No] [-k, --KeepLargestComponent Yes |
No] [-m, --mode PathLengthBits | PathLengthCount] [--MinPathLength number] [--MaxPathLength number] [
-n, --NumOfBitsToSetPerPath number] [--OutDelim comma | tab | semicolon] [--output SD | FP | text | all] [-q,
--quote Yes | No] [-r, --root RootName] [-p, --PathMode AtomPathsWithoutRings | AtomPathsWithRings |
AllAtomPathsWithoutRings | AllAtomPathsWithRings] [-s, --size number] [-u, --UseBondSymbols Yes | No] [
--UsePerCoreRandom Yes | No] [--UseUniquePaths Yes | No] [-q, --quote Yes | No] [-r, --root RootName] [
-v, --VectorStringFormat IDsAndValuesString | IDsAndValuesPairsString | ValuesAndIDsString |
ValuesAndIDsPairsString] [-w, --WorkingDir dirname] SDFfile(s)...
```

DESCRIPTION

Generate atom path length fingerprints for *SDFfile(s)* and create appropriate SD, FP or CSV/TSV text file(s) containing fingerprints bit-vector or 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 path length fingerprints corresponding to following *-a, --AtomIdentifierTypes*:

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

Based on the values specified for *-p, --PathMode, --MinPathLength* and *--MaxPathLength*, all appropriate atom paths are generated for each atom in the molecule and collected in a list and the list is filtered to remove any structurally duplicate paths as indicated by the value of *--UseUniquePaths* option.

For each atom path in the filtered atom paths list, an atom path string is created using value of *-a, --AtomIdentifierType* and specified values to use for a particular atom identifier type. Value of *-u, --UseBondSymbols* controls whether bond order symbols are used during generation of atom path string. For each atom path, only lexicographically smaller atom path strings are kept.

For *PathLengthBits* value of *-m, --mode* option, each atom path is hashed to a 32 bit unsigned integer key using `TextUtil::HashCode` function. Using the hash key as a seed for a random number generator, a random integer value between 0 and *--Size* is used to set corresponding bits in the fingerprint bit-vector string. Value of *--NumOfBitsToSetPerPath* option controls the number of times a random number is generated to set corresponding bits.

For *PathLengthCount* value of *-m, --mode* option, the number of times an atom path appears is tracked and a fingerprints count-string corresponding to count of atom paths is generated.

Example of *SD* file containing path length fingerprints string data:

```
... ..
... ..
$$$$
... ..
... ..
... ..
41 44 0 0 0 0 0 0 0 0 0999 v2000
-3.3652 1.4499 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
... ..
2 3 1 0 0 0 0
... ..
M END
> <CmpdID>
```

```
Cmpd1
```

```
> <PathLengthFingerprints>
FingerprintsBitVector;PathLengthBits:AtomicInvariantsAtomTypes:MinLength
h1:MaxLength8;1024;HexadecimalString;Ascending;9c8460989ec8a49913991a66
03130b0a19e8051c89184414953800cc2151082844a201042800130860308e8204d4028
00831048940e44281c00060449a5000ac80c894114e006321264401600846c050164462
08190410805000304a10205b0100e04c0038ba0fad0209c0ca8b1200012268b61c0026a
aa0660a11014a011d46

$$$$
... ..
... ..
```

Example of *FP* file containing path length fingerprints string data:

```
#
# Package = MayaChemTools 7.4
# ReleaseDate = Oct 21, 2010
#
# TimeStamp = Mon Mar 7 15:14:01 2011
#
# FingerprintsStringType = FingerprintsBitVector
#
# Description = PathLengthBits:AtomicInvariantsAtomTypes:MinLength1:...
# Size = 1024
# BitStringFormat = HexadecimalString
# BitsOrder = Ascending
#
Cmpd1 9c8460989ec8a49913991a6603130b0a19e8051c89184414953800cc21510...
Cmpd2 000000249400840040100042011001001980410c00000001010088001120...
... ..
... ..
```

Example of CSV *Text* file containing pathlength fingerprints string data:

```
"CompoundID", "PathLengthFingerprints"
"Cmpd1", "FingerprintsBitVector;PathLengthBits:AtomicInvariantsAtomTypes
:MinLength1:MaxLength8;1024;HexadecimalString;Ascending;9c8460989ec8a4
9913991a6603130b0a19e8051c89184414953800cc2151082844a20104280013086030
8e8204d402800831048940e44281c00060449a5000ac80c894114e006321264401..."
... ..
... ..
```

The current release of MayaChemTools generates the following types of path length fingerprints bit-vector and vector strings:

```
FingerprintsBitVector;PathLengthBits:AtomicInvariantsAtomTypes:MinLength
h1:MaxLength8;1024;BinaryString;Ascending;001000010011010101011000110
0100010101011000101001011100110001000010001001101000001001001001000
001011010000011100100100000100101010010010000000011000000101001011100
0010000001000101010100000100111100110111011011011000000010110111001101
01011000110000000100010000110000101000111011000010000100000000...
```

```
FingerprintsBitVector;PathLengthBits:AtomicInvariantsAtomTypes:MinLength
h1:MaxLength8;1024;HexadecimalString;Ascending;48caa1315d82d91122b029
42861c9409a4208182d12015509767bd0867653604481a8b1288000056090583603078
9cedae54e26596889ab121309800900490515224208421502120a0dd9200509723ae89
00024181b86c0122821d4e4880c38620dab280824b455404009f082003d52c212b4e6d
6ea05280140069c780290c43
```

```
FingerprintsVector;PathLengthCount:AtomicInvariantsAtomTypes:MinLength
1:MaxLength8;432;NumericalValues;IDsAndValuesPairsString;C.X1.B01.H3 2
C.X2.B02.H2 4 C.X2.B03.H1 14 C.X3.B03.H1 3 C.X3.B04 10 F.X1.B01 1 N.X
2.B02.H1 1 N.X3.B03 1 O.X1.B01.H1 3 O.X1.B02 2 C.X1.B01.H3C.X3.B03.H1
2 C.X2.B02.H2C.X2.B02.H2 1 C.X2.B02.H2C.X3.B03.H1 4 C.X2.B02.H2C.X3.B0
```

4 1 C.X2.BO2.H2N.X3.BO3 1 C.X2.BO3.H1:C.X2.BO3.H1 10 C.X2.BO3.H1:C...

FingerprintsVector;PathLengthCount:DREIDINGAtomTypes:MinLength1:MaxLength8;410;NumericalValues;IDsAndValuesPairsString;C_2 2 C_3 9 C_R 22 F_1 N_3 1 N_R 1 O_2 2 O_3 3 C_2=O_2 2 C_2C_3 1 C_2C_R 1 C_2N_3 1 C_2O_3 1 C_3C_3 7 C_3C_R 1 C_3N_R 1 C_3O_3 2 C_R:C_R 21 C_R:N_R 2 C_RC_R 2 C_RF_1 C_RN_3 1 C_2C_3C_3 1 C_2C_R:C_R 2 C_2N_3C_R 1 C_3C_2=O_2 1 C_3C_2O_3 1 C_3C_3C_3 5 C_3C_3C_R 2 C_3C_3N_R 1 C_3C_3O_3 4 C_3C_R:C_R ...

FingerprintsVector;PathLengthCount:EStateAtomTypes:MinLength1:MaxLength8;454;NumericalValues;IDsAndValuesPairsString;aaCH 14 aasC 8 aasN 1 dO 2 dssC 2 sCH3 2 sF 1 sOH 3 ssCH2 4 ssNH 1 sssCH 3 aaCH:aaCH 10 aaCH:aasC 8 aasC:aasC 3 aasC:aasN 2 aasCaasC 2 aasCdssC 1 aasCsF 1 aasCssNH 1 aasCsssCH 1 aasNssCH2 1 dO=dssC 2 dssCsOH 1 dssCssCH2 1 dssCssNH 1 sCH3sssCH 2 sOHsssCH 2 ssCH2ssCH2 1 ssCH2sssCH 4 aaCH:aaCH:aaCH 6 a...

FingerprintsVector;PathLengthCount:FunctionalClassAtomTypes:MinLength1:MaxLength8;404;NumericalValues;IDsAndValuesPairsString;Ar 22 Ar.HBA 1 HBA 2 HBA.HBD 3 HBD 1 Hal 1 NI 1 None 10 Ar.HBA:Ar 2 Ar.HBANone 1 Ar:Ar 21 ArAr 2 ArHBD 1 ArHal 1 ArNone 2 HBA.HBDNI 1 HBA.HBDNone 2 HBA=NI 1 HBA=None 1 HBDNone 1 NINone 1 NoneNone 7 Ar.HBA:Ar:Ar 2 Ar.HBA:ArAr 1 Ar.HBA:ArNone 1 Ar.HBANoneNone 1 Ar:Ar.HBA:Ar 1 Ar:Ar.HBANone 2 ...

FingerprintsVector;PathLengthCount:MMFF94AtomTypes:MinLength1:MaxLength8;463;NumericalValues;IDsAndValuesPairsString;C5A 2 C5B 2 C=ON 1 CB 1 8 COO 1 CR 9 F 1 N5 1 NC=O 1 O=CN 1 O=CO 1 OC=O 1 OR 2 C5A:C5B 2 C5A:N 5 2 C5ACB 1 C5ACR 1 C5B:C5B 1 C5BC=ON 1 C5BCB 1 C=ON=O=CN 1 C=ONNC=O 1 CB:CB 18 CBF 1 CBNC=O 1 COO=O=CO 1 COOCR 1 COOOC=O 1 CRCR 7 CRN5 1 CROR 2 C5A:C5B:C5B 2 C5A:C5BC=ON 1 C5A:C5BCB 1 C5A:N5:C5A 1 C5A:N5CR ...

FingerprintsVector;PathLengthCount:SLogPAtomTypes:MinLength1:MaxLength8;518;NumericalValues;IDsAndValuesPairsString;C1 5 C10 1 C11 1 C14 1 C18 14 C20 4 C21 2 C22 1 C5 2 C5 2 F 1 N11 1 N4 1 O10 1 O2 3 O9 1 C10C1 1 C10N11 1 C11C1 2 C11C21 1 C14:C18 2 C14F 1 C18:C18 10 C18:C20 4 C18:C22 2 C1C5 1 C1CS 4 C20:C20 1 C20:C21 1 C20:N11 1 C20C20 2 C21:C21 1 C21:N11 1 C21C5 1 C22N4 1 C5=O10 1 C5=O9 1 C5N4 1 C5O2 1 CSO2 2 C10...

FingerprintsVector;PathLengthCount:SYBYLAtomTypes:MinLength1:MaxLength8;412;NumericalValues;IDsAndValuesPairsString;C.2 2 C.3 9 C.ar 22 F 1 N.am 1 N.ar 1 O.2 1 O.3 2 O.co2 2 C.2=O.2 1 C.2=O.co2 1 C.2C.3 1 C.2C.ar 1 C.2N.am 1 C.2O.co2 1 C.3C.3 7 C.3C.ar 1 C.3N.ar 1 C.3O.3 2 C.ar:C.ar 21 C.ar:N.ar 2 C.arC.ar 2 C.arF 1 C.arN.am 1 C.2C.3C.3 1 C.2C.ar:C.ar 2 C.2N.amC.ar 1 C.3C.2=O.co2 1 C.3C.2O.co2 1 C.3C.3C.3 5 C.3C.3...

FingerprintsVector;PathLengthCount:TPSAAtomTypes:MinLength1:MaxLength8;331;NumericalValues;IDsAndValuesPairsString;N21 1 N7 1 None 34 O3 2 O4 3 N21:None 2 N21None 1 N7None 2 None:None 21 None=O3 2 NoneNone 13 N oneO4 3 N21:None:None 2 N21:NoneNone 2 N21NoneNone 1 N7None:None 2 N7N one=O3 1 N7NoneNone 1 None:N21:None 1 None:N21None 2 None:None:None 20 None:NoneNone 12 NoneN7None 1 NoneNone=O3 2 NoneNoneNone 8 NoneNon...

FingerprintsVector;PathLengthCount:UFFAtomTypes:MinLength1:MaxLength8;410;NumericalValues;IDsAndValuesPairsString;C_2 2 C_3 9 C_R 22 F_1 N_3 1 N_R 1 O_2 2 O_3 3 C_2=O_2 2 C_2C_3 1 C_2C_R 1 C_2N_3 1 C_2O_3 1 C_3C_3 7 C_3C_R 1 C_3N_R 1 C_3O_3 2 C_R:C_R 21 C_R:N_R 2 C_RC_R 2 C_RF_1 C_RN_3 1 C_2C_3C_3 1 C_2C_R:C_R 2 C_2N_3C_R 1 C_3C_2=O_2 1 C_3C_2O_3 1 C_3C_3C_3 5 C_3C_3C_R 2 C_3C_3N_R 1 C_3C_3O_3 4 C_3C_R:C_R 1 C_3...

OPTIONS

--AromaticityModel *MDLAromaticityModel* | *TriposAromaticityModel* | *MMFFAromaticityModel* | *ChemAxonBasicAromaticityModel* | *ChemAxonGeneralAromaticityModel* | *DaylightAromaticityModel* | *MayaChemToolsAromaticityModel*

Specify aromaticity model to use during detection of aromaticity. Possible values in the current release are: *MDLAromaticityModel*, *TriposAromaticityModel*, *MMFFAromaticityModel*, *ChemAxonBasicAromaticityModel*,

ChemAxonGeneralAromaticityModel, *DaylightAromaticityModel* or *MayaChemToolsAromaticityModel*. Default value: *MayaChemToolsAromaticityModel*.

The supported aromaticity model names along with model specific control parameters are defined in *AromaticityModelsData.csv*, which is distributed with the current release and is available under *lib/data* directory. *Molecule.pm* module retrieves data from this file during class instantiation and makes it available to method *DetectAromaticity* for detecting aromaticity corresponding to a specific model.

This option is ignored during *No* value of *--DetectAromaticity* option.

-a, --AtomIdentifierType AtomicInvariantsAtomTypes | DREIDINGAtomTypes | EStateAtomTypes | FunctionalClassAtomTypes | MMFF94AtomTypes | SLogPAtomTypes | SYBYLAtomTypes | TPSAAAtomTypes | UFFAtomTypes

Specify atom identifier type to use for assignment of atom types to hydrogen and/or non-hydrogen atoms during calculation of atom types fingerprints. Possible values in the current release are:

AtomicInvariantsAtomTypes, *DREIDINGAtomTypes*, *EStateAtomTypes*, *FunctionalClassAtomTypes*, *MMFF94AtomTypes*, *SLogPAtomTypes*, *SYBYLAtomTypes*, *TPSAAAtomTypes*, *UFFAtomTypes*. Default value: *AtomicInvariantsAtomTypes*.

-a, --AtomIdentifierType AtomicInvariantsAtomTypes | DREIDINGAtomTypes | EStateAtomTypes | FunctionalClassAtomTypes | MMFF94AtomTypes | SLogPAtomTypes | SYBYLAtomTypes | TPSAAAtomTypes | UFFAtomTypes

Specify atom identifier type to use during generation of atom path strings corresponding to path length fingerprints. Possible values in the current release are: *AtomicInvariantsAtomTypes*, *DREIDINGAtomTypes*, *EStateAtomTypes*, *FunctionalClassAtomTypes*, *MMFF94AtomTypes*, *SLogPAtomTypes*, *SYBYLAtomTypes*, *TPSAAAtomTypes*, *UFFAtomTypes*. Default value: *AtomicInvariantsAtomTypes*.

--AtomicInvariantsToUse "AtomicInvariant1,AtomicInvariant2..."

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*.

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*

Examples:

Benzene: Using value of *AS* for --AtomI nvariantsToUse, *Yes* for UseBondSymbols, and *AllAtomPathsWithRings* for -p, --PathMode, atom path strings generated are:

```
C C:C C:C:C C:C:C:C C:C:C:C:C C:C:C:C:C:C C:C:C:C:C:C:C
```

And using *AS,X,BO* for --AtomI nvariantsToUse generates following atom path strings:

```
C.X2.B03 C.X2.B03:C.X2.B03 C.X2.B03:C.X2.B03:C.X2.B03
C.X2.B03:C.X2.B03:C.X2.B03:C.X2.B03
C.X2.B03:C.X2.B03:C.X2.B03:C.X2.B03:C.X2.B03
C.X2.B03:C.X2.B03:C.X2.B03:C.X2.B03:C.X2.B03:C.X2.B03
C.X2.B03:C.X2.B03:C.X2.B03:C.X2.B03:C.X2.B03:C.X2.B03:C.X2.B03
```

Urea: Using value of *AS* for --AtomI nvariantsToUse, *Yes* for UseBondSymbols, and *AllAtomPathsWithRings* for -p, --PathMode, atom path strings are:

```
C N O C=O CN NC=O NCN
```

And using *AS,X,BO* for --AtomI nvariantsToUse generates following atom path strings:

```
C.X3.B04 N.X1.B01 O.X1.B02 C.X3.B04=O.X1.B02
C.X3.B04N.X1.B01 N.X1.B01C.X3.B04=O.X1.B02
N.X1.B01C.X3.B04N.X1.B01
```

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

This value is used during *FunctionalClassAtomTypes* value of a, --AtomI dentifierType 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
```

--BitsOrder *Ascending | Descending*

Bits order to use during generation of fingerprints bit-vector string for *PathLengthBits* value of -m, --mode option. Possible values: *Ascending, Descending*. Default: *Ascending*.

Ascending bit order which corresponds to first bit in each byte as the lowest bit as opposed to the highest bit.

Internally, bits are stored in *Ascending* order using Perl vec function. Regardless of machine order, big-endian or little-endian, vec function always considers first string byte as the lowest byte and first bit within each byte as the lowest bit.

-b, --BitStringFormat *BinaryString | HexadecimalString*

Format of fingerprints bit-vector string data in output SD, FP or CSV/TSV text file(s) specified by --output used during *PathLengthBits* value of -m, --mode option. Possible values: *BinaryString, HexadecimalString*. Default value: *HexadecimalString*.

BinaryString corresponds to an ASCII string containing 1s and 0s. *HexadecimalString* contains bit values in

ASCII hexadecimal format.

Examples:

```
FingerprintsBitVector;PathLengthBits:AtomicInvariantsAtomTypes:MinLength:MaxLength8;1024;BinaryString;Ascending;00100001001101010101000110010001010110001010010111001100010000100010011010100000100100100100000000110000001010010111000010010101010000010011110011011011011011011000000010110111001101010101000001001111001101101101101101100000001011011100110101010000001000100001100001010001110110000100001000100000000...
```

```
FingerprintsBitVector;PathLengthBits:AtomicInvariantsAtomTypes:MinLength:MaxLength8;1024;HexadecimalString;Ascending;48caa1315d82d91122b02942861c9409a4208182d12015509767bd0867653604481a8b12880000560905836030789cedae54e26596889ab121309800900490515224208421502120a0dd9200509723ae8900024181b86c0122821d4e4880c38620dab280824b455404009f082003d52c212b4e6d6ea05280140069c780290c43
```

--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 FP or 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 FP or CSV/TSV text file(s) along with generated fingerprints for *FP* | *text* | *all* 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* | *all* 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* | *all* 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*.

- DetectAromaticity *Yes / No*
- Detect aromaticity before generating fingerprints. Possible values: *Yes or No*. Default value: *Yes*.
- No* --DetectAromaticity forces usage of atom and bond aromaticity values from *SDFFile(s)* and skips the step which detects and assigns aromaticity.
- No* --DetectAromaticity value is only allowed using *AtomicInvariantsAtomTypes* value of *-a*, --AtomIdentifierType options; for all possible values *-a*, --AtomIdentifierType values, it must be *Yes*.
- 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: *PathLengthFingerprints*.
- fold *Yes / No*
- Fold fingerprints to increase bit density during *PathLengthBits* value of *-m*, --mode option. Possible values: *Yes or No*. Default value: *No*.
- FoldedSize *number*
- Size of folded fingerprint during *PathLengthBits* value of *-m*, --mode option. Default value: *256*. Valid values correspond to any positive integer which is less than *-s*, --size and meets the criteria for its value.
- Examples:
- 128
 - 512
- h, --help
- Print this help message
- i, --IgnoreHydrogens *Yes / No*
- Ignore hydrogens during fingerprints generation. Possible values: *Yes or No*. Default value: *Yes*.
- For *yes* value of *-i*, --IgnoreHydrogens, any explicit hydrogens are also used for generation of atoms path lengths and fingerprints; implicit hydrogens are still ignored.
- 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.
- m, --mode *PathLengthBits | PathLengthCount*
- Specify type of path length fingerprints to generate for molecules in *SDFFile(s)*. Possible values: *PathLengthBits, PathLengthCount*. Default value: *PathLengthBits*.
- For *PathLengthBits* value of *-m*, --mode option, a fingerprint bit-vector string containing zeros and ones is generated and for *PathLengthCount* value, a fingerprint vector string corresponding to number of atom paths is generated.
- MinPathLength *number*
- Minimum atom path length to include in fingerprints. Default value: *1*. Valid values: positive integers and less than --MaxPathLength. Path length of 1 correspond to a path containing only one atom.
- MaxPathLength *number*
- Maximum atom path length to include in fingerprints. Default value: *8*. Valid values: positive integers and greater than --MinPathLength.
- n, --NumOfBitsToSetPerPath *number*
- Number of bits to set per path during generation of fingerprints bit-vector string for *PathLengthBits* value

of -m, --mode option. Default value: 1. Valid values: positive integers.

--OutDelim *comma | tab | semicolon*

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

--output *SD | FP | text | all*

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

-o, --overwrite

Overwrite existing files.

-p, --PathMode *AtomPathsWithoutRings | AtomPathsWithRings | AllAtomPathsWithoutRings | AllAtomPathsWithRings*

Specify type of atom paths to use for generating pathlength fingerprints for molecules in *SDF*file(s).

Possible values: *AtomPathsWithoutRings, AtomPathsWithRings, AllAtomPathsWithoutRings, AllAtomPathsWithRings*. Default value: *AllAtomPathsWithRings*.

For molecules with no rings, first two and last two options are equivalent and generate same set of atom paths starting from each atom with length between --MinPathLength and --MaxPathLength. However, all these four options can result in the same set of final atom paths for molecules containing fused, bridged or spiro rings.

For molecules containing rings, atom paths starting from each atom can be traversed in four different ways:

AtomPathsWithoutRings - Atom paths containing no rings and without sharing of bonds in traversed paths.

AtomPathsWithRings - Atom paths containing rings and without any sharing of bonds in traversed paths.

AllAtomPathsWithoutRings - All possible atom paths containing no rings and without any sharing of bonds in traversed paths.

AllAtomPathsWithRings - All possible atom paths containing rings and with sharing of bonds in traversed paths.

Atom path traversal is terminated at the ring atom.

Based on values specified for for -p, --PathMode, --MinPathLength and --MaxPathLength, all appropriate atom paths are generated for each atom in the molecule and collected in a list.

For each atom path in the filtered atom paths list, an atom path string is created using value of -a, --AtomIdentifierType and specified values to use for a particular atom identifier type. Value of -u, --UseBondSymbols controls whether bond order symbols are used during generation of atom path string. Atom symbol corresponds to element symbol and characters used to represent bond order are: 1 - None; 2 - '='; 3 - '#'; 1.5 or aromatic - ':'; others: bond order value. By default, bond symbols are included in atom path strings. Exclusion of bond symbols in atom path strings results in fingerprints which correspond purely to atom paths without considering bonds.

UseUniquePaths controls the removal of structurally duplicate atom path strings are removed from the list.

For *PathLengthBits* value of -m, --mode option, each atom path is hashed to a 32 bit unsigned integer key using TextUtil::HashCode function. Using the hash key as a seed for a random number generator, a random integer value between 0 and --Size is used to set corresponding bits in the fingerprint bit-vector string. Value of --NumOfBitsToSetPerPaths option controls the number of time a random number is generated to set corresponding bits.

For *PathLengthCount* value of -m, --mode option, the number of times an atom path appears is tracked and a fingerprints count-string corresponding to count of atom paths is generated.

For molecule containing rings, combination of -p, --PathMode and --UseBondSymbols allows generation of up to 8 different types of atom path length strings:

| AllowSharedBonds | AllowRings | UseBondSymbols | |
|------------------|------------|----------------|---|
| 0 | 0 | 1 | - AtomPathsNoCyclesWithBondSymbols |
| 0 | 1 | 1 | - AtomPathsWithCyclesWithBondSymbols |
| 1 | 0 | 1 | - AllAtomPathsNoCyclesWithBondSymbols |
| 1 | 1 | 1 | - AllAtomPathsWithCyclesWithBondSymbols |
| | | | [DEFAULT] |
| 0 | 0 | 0 | - AtomPathsNoCyclesNoBondSymbols |
| 0 | 1 | 0 | - AtomPathsWithCyclesNoBondSymbols |
| 1 | 0 | 0 | - AllAtomPathsNoCyclesNoBondSymbols |

1 1 0 - AllAtomPathsWithCyclesNoWithBondSymbols

Default atom path length fingerprints generation for molecules containing rings with *AllAtomPathsWithRings* value for *-p, --PathMode*, *Yes* value for *--UseBondSymbols*, *2* value for *--MinPathLength* and *8* value for *--MaxPathLength* is the most time consuming. Combinations of other options can substantially speed up fingerprint generation for molecules containing complex ring systems.

Additionally, value for option *-a, --AtomIdentifierType* in conjunction with corresponding specified values for atom types changes the nature of atom path length strings and the fingerprints.

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

-s, --size number

Size of fingerprints. Default value: *1024*. Valid values correspond to any positive integer which satisfies the following criteria: power of 2, ≥ 32 and $\leq 2^{**} 32$.

Examples:

256
512
2048

-u, --UseBondSymbols Yes / No

Specify whether to use bond symbols for atom paths during generation of atom path strings. Possible values: *Yes or No*. Default value: *Yes*.

No value option for *-u, --UseBondSymbols* allows the generation of fingerprints corresponding purely to atoms disregarding all bonds.

--UsePerlCoreRandom Yes / No

Specify whether to use Perl `CORE::rand` or MayaChemTools `MathUtil::random` function during random number generation for setting bits in fingerprints bit-vector strings. Possible values: *Yes or No*. Default value: *Yes*.

No value option for *--UsePerlCoreRandom* allows the generation of fingerprints bit-vector strings which are same across different platforms.

The random number generator implemented in MayaChemTools is a variant of linear congruential generator (LCG) as described by Miller et al. [Ref 120]. It is also referred to as Lehmer random number generator or Park-Miller random number generator.

Unlike Perl's core random number generator function `rand`, the random number generator implemented in MayaChemTools, `MathUtil::random`, generates consistent random values across different platforms for a specific random seed and leads to generation of portable fingerprints bit-vector strings.

--UseUniquePaths Yes / No

Specify whether to use structurally unique atom paths during generation of atom path strings. Possible values: *Yes or No*. Default value: *Yes*.

No value option for *--UseUniquePaths* allows usage of all atom paths generated by *-p, --PathMode* option value for generation of atom path strings leading to duplicate path count during *PathLengthCount* value of *-m, --mode* option. It doesn't change fingerprint string generated during *PathLengthBits* value of *-m, --mode*.

For example, during *AllAtomPathsWithRings* value of *-p, --PathMode* option, benzene has 12 linear paths of length 2 and 12 cyclic paths length of 7, but only 6 linear paths of length 2 and 1 cyclic path of length 7 are structurally unique.

-v, --VectorStringFormat IDsAndValuesString | IDsAndValuesPairsString | ValuesAndIDsString | ValuesAndIDsPairsString

Format of fingerprints vector string data in output SD, FP or CSV/TSV text file(s) specified by *--output* used during *PathLengthCount* value of *-m, --mode* option. Possible values: *IDsAndValuesString | IDsAndValuesPairsString | ValuesAndIDsString | ValuesAndIDsPairsString*. Default value: *IDsAndValuesString*.

Examples:

```
FingerprintsVector;PathLengthCount:AtomicInvariantsAtomTypes:MinLength
1:MaxLength8;432;NumericalValues;IDsAndValuesPairsString;C.X1.BO1.H3 2
C.X2.BO2.H2 4 C.X2.BO3.H1 14 C.X3.BO3.H1 3 C.X3.BO4 10 F.X1.BO1 1 N.X
2.BO2.H1 1 N.X3.BO3 1 O.X1.BO1.H1 3 O.X1.BO2 2 C.X1.BO1.H3C.X3.BO3.H1
2 C.X2.BO2.H2C.X2.BO2.H2 1 C.X2.BO2.H2C.X3.BO3.H1 4 C.X2.BO2.H2C.X3.BO
4 1 C.X2.BO2.H2N.X3.BO3 1 C.X2.BO3.H1:C.X2.BO3.H1 10 C.X2.BO3.H1:C...
```

```
FingerprintsVector;PathLengthCount:EStateAtomTypes:MinLength1:MaxLengt
h8;454;NumericalValues;IDsAndValuesPairsString;aaCH 14 aasC 8 aasN 1 d
O 2 dssC 2 sCH3 2 sF 1 sOH 3 ssCH2 4 ssNH 1 sssCH 3 aaCH:aaCH 10 aaCH:
aasC 8 aasC:aasC 3 aasC:aasN 2 aasCaasC 2 aasCdssC 1 aasCsF 1 aasCssNH
1 aasCsssCH 1 aasNssCH2 1 dO=dssC 2 dssCsOH 1 dssCssCH2 1 dssCssNH 1
sCH3sssCH 2 sOHsssCH 2 ssCH2ssCH2 1 ssCH2sssCH 4 aaCH:aaCH:aaCH 6 a...
```

`-w, --WorkingDir DirName`

Location of working directory. Default: current directory.

EXAMPLES

To generate path length fingerprints corresponding to all unique paths from length 1 through 8 in hexadecimal bit-vector string format of size 1024 and create a SamplePLFPHex.csv file containing sequential compound IDs along with fingerprints bit-vector strings data, type:

```
% PathLengthFingerprints.pl -o -r SamplePLFPHex Sample.sdf
```

To generate path length fingerprints corresponding to all unique paths from length 1 through 8 in hexadecimal bit-vector string format of size 1024 and create SamplePLFPHex.sdf, SamplePLFPHex.fpf, and SamplePLFPHex.csv files containing sequential compound IDs in CSV file along with fingerprints bit-vector strings data, type:

```
% PathLengthFingerprints.pl --output all -o -r SamplePLFPHex Sample.sdf
```

To generate path length fingerprints corresponding to all unique paths from length 1 through 8 in binary bit-vector string format of size 1024 and create a SamplePLFPBin.csv file containing sequential compound IDs along with fingerprints bit-vector strings data, type:

```
% PathLengthFingerprints.pl --BitStringFormat BinaryString --size 2048
-o -r SamplePLFPBin Sample.sdf
```

To generate path length fingerprints corresponding to count of all unique paths from length 1 through 8 in IDsAndValuesString format and create a SamplePLFPCount.csv file containing sequential compound IDs along with fingerprints vector strings data, type:

```
% PathLengthFingerprints.pl -m PathLengthCount -o -r SamplePLFPCount
Sample.sdf
```

To generate path length fingerprints corresponding to count of all unique paths from length 1 through 8 in IDsAndValuesString format using E-state atom types and create a SamplePLFPCount.csv file containing sequential compound IDs along with fingerprints vector strings data, type:

```
% PathLengthFingerprints.pl -m PathLengthCount --AtomIdentifierType
EStateAtomTypes -o -r SamplePLFPCount Sample.sdf
```

To generate path length fingerprints corresponding to count of all unique paths from length 1 through 8 in IDsAndValuesString format using SLogP atom types and create a SamplePLFPCount.csv file containing sequential compound IDs along with fingerprints vector strings data, type:

```
% PathLengthFingerprints.pl -m PathLengthCount --AtomIdentifierType
SLogPAtomTypes -o -r SamplePLFPCount Sample.sdf
```

To generate path length fingerprints corresponding to count of all unique paths from length 1 through 8 in IDsAndValuesString format and create a SamplePLFPCount.csv file containing sequential compound IDs along with fingerprints vector strings data, type:

```
% PathLengthFingerprints.pl -m PathLengthCount --VectorStringFormat
ValuesAndIDsPairsString -o -r SamplePLFPCount Sample.sdf
```

To generate path length fingerprints corresponding to count of all unique paths from length 1 through 8 in IDsAndValuesString format using AS,X,BO as atomic invariants and create a SamplePLFPCount.csv file containing sequential compound IDs along with fingerprints vector strings data, type:

```
% PathLengthFingerprints.pl -m PathLengthCount --AtomIdentifierType
AtomicInvariantsAtomTypes --AtomicInvariantsToUse "AS,X,BO" -o
-r SamplePLFPCount Sample.sdf
```

To generate path length fingerprints corresponding to count of all paths from length 1 through 8 in IDsAndValuesString format and create a SamplePLFPCount.csv file containing compound IDs from MolName line along with fingerprints vector strings data, type:

```
% PathLengthFingerprints.pl -m PathLengthCount --UseUniquePaths No
-o --CompoundIDMode MolName -r SamplePLFPCount --UseUniquePaths No
Sample.sdf
```

To generate path length fingerprints corresponding to all unique paths from length 1 through 8 in hexadecimal bit-vector string format of size 512 after folding and create SamplePLFPHex.sdf, SamplePLFPHex.fpf, and SamplePLFPHex.csv files containing sequential compound IDs along with fingerprints bit-vector strings data, type:

```
% PathLengthFingerprints.pl --output all --Fold Yes --FoldedSize 512
-o -r SamplePLFPHex Sample.sdf
```

To generate path length fingerprints corresponding to all unique paths from length 1 through 8 containing no rings and without sharing of bonds in hexadecimal bit-vector string format of size 1024 and create a SamplePLFPHex.csv file containing sequential compound IDs along with fingerprints bit-vector strings data and all data fields, type:

```
% PathLengthFingerprints.pl -p AtomPathsWithoutRings --DataFieldsMode All
-o -r SamplePLFPHex Sample.sdf
```

To generate path length fingerprints corresponding to all unique paths from length 1 through 8 containing rings and without sharing of bonds in hexadecimal bit-vector string format of size 1024 and create a SamplePLFPHex.tsv file containing compound IDs derived from combination of molecule name line and an explicit compound prefix along with fingerprints bit-vector strings data and all data fields, type:

```
% PathLengthFingerprints.pl -p AtomPathsWithRings --DataFieldsMode
CompoundID --CompoundIDMode MolnameOrLabelPrefix --CompoundID Cmpd
--CompoundIDLabel MolID --FingerprintsLabel PathLengthFP --OutDelim Tab
-r SamplePLFPHex -o Sample.sdf
```

To generate path length fingerprints corresponding to count of all unique paths from length 1 through 8 in IDsAndValuesString format and create a SamplePLFPCount.csv file containing sequential compound IDs along with fingerprints vector strings data using aromaticity specified in SD file, type:

```
% PathLengthFingerprints.pl -m PathLengthCount --DetectAromaticity No
-o -r SamplePLFPCount Sample.sdf
```

To generate path length fingerprints corresponding to all unique paths from length 2 through 6 in hexadecimal bit-vector string format of size 1024 and create a SamplePLFPHex.csv file containing sequential compound IDs along with fingerprints bit-vector strings data, type:

```
% PathLengthFingerprints.pl --MinPathLength 2 --MaxPathLength 6
-o -r SamplePLFPHex Sample.sdf
```

AUTHOR

Manish Sud <msud@san.rr.com>

SEE ALSO

InfoFingerprintsFiles.pl, SimilarityMatricesFingerprints.pl, AtomNeighborhoodsFingerprints.pl, ExtendedConnectivityFingerprints.pl, MACCSKeysFingerprints.pl, TopologicalAtomPairsFingerprints.pl, TopologicalAtomTorsionsFingerprints.pl, TopologicalPharmacophoreAtomPairsFingerprints.pl, TopologicalPharmacophoreAtomTripletsFingerprints.pl

COPYRIGHT

Copyright (C) 2024 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.