

## NAME

ExtractFromSDFFiles.pl - Extract specific data from SDFFile(s)

## SYNOPSIS

ExtractFromSDFFiles.pl SDFFile(s)...

```
ExtractFromSDFFiles.pl [-h, --help] [-d, --datafields "fieldlabel,..." | "fieldlabel,value,criteria..." | "fieldlabel,value,value..." ] [
--datafieldsfile filename] [--indelim comma | tab | semicolon] [-m, --mode alldatafields | commondatafields | datafields |
datafieldsbyvalue | datafieldsbyregex | datafieldbylist | datafielduniquebylist | molnames | randomcmpds | recordnum |
recdrange | 2dcmpdrecords | 3dcmpdrecords ] [-n, --numofcmpds number] [--outdelim comma | tab | semicolon] [
--output SD | text | both] [-o, --overwrite] [-q, --quote yes | no] [--record recnum | startrecnum,endrecnum]
--RegexIgnoreCase yes or no [-r, --root rootname] [-s, --seed number] [--StrDataString yes | no] [
--StrDataStringDelimiter text] [--StrDataStringMode StrOnly | StrAndDataFields] [--ValueComparisonMode Numeric |
Alphanumeric] [-v, --violations- number] [-w, --workingdir dirname] SDFFile(s)...
```

## DESCRIPTION

Extract specific data from *SDFFile(s)* and generate appropriate SD or CSV/TSV text file(s). The structure data from SDFFile(s) is not transferred to CSV/TSV text file(s). 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.

## OPTIONS

-h, --help

Print this help message.

-d, --datafields "*fieldlabel,...*" | "*fieldlabel,value,criteria...*" | "*fieldlabel,value,value,...*"

This value is mode specific. In general, it's a list of comma separated data field labels and associated mode specific values.

For *datafields* mode, input value format is: *fieldlabel,....* Examples:

```
Extreg
Extreg,CompoundName,ID
```

For *datafieldsbyvalue* mode, input value format contains these triplets: *fieldlabel,value,criteria....* Possible values for criteria: *le, ge or eq*. The values of --ValueComparisonMode indicates whether values are compared numerical or string comparison operators. Default is to consider data field values as numerical values and use numerical comparison operators. Examples:

```
MolWt,450,le
MolWt,450,le,LogP,5,le,SumNumNO,10,le,SumNHOH,5,le
```

For *datafieldsbyregex* mode, input value format contains these triplets: *fieldlabel,regex,criteria....* *regex* corresponds to any valid regular expression and is used to match the values for specified *fieldlabel*. Possible values for criteria: *eq or ne*. During *eq* and *ne* values, data field label value is matched with regular expression using *=~* and *!~* respectively. --RegexIgnoreCase option value is used to determine whether to ignore letter upper/lower case during regular expression match. Examples:

```
Name,ol,eq
Name,'^pat',ne
```

For *datafieldbylist* and *datafielduniquebylist* mode, input value format is: *fieldlabel,value1,value2....* This is equivalent to *datafieldsbyvalue* mode with this input value format: *fieldlabel,value1,eq,fieldlabel,value2,eq,....* For *datafielduniquebylist* mode, only unique compounds identified by first occurrence of *value* associated with *fieldlabel* in *SDFFile(s)* are kept; any subsequent compounds are simply ignored.

--datafieldsfile *filename*

Filename which contains various mode specific values. This option provides a way to specify mode specific values in a file instead of entering them on the command line using -d --datafields.

For *datafields* mode, input file lines contain comma delimited field labels: *fieldlabel,....* Example:

```
Line 1:MolId
Line 2:"Extreg",CompoundName,ID
```

For *datafieldsbyvalue* mode, input file lines contains these comma separated triplets: *fieldlabel,value,criteria*. Possible values for criteria: *le, ge or eq*. Examples:

```
Line 1:MolWt,450,le

Line 1:"MolWt",450,le,"LogP",5,le,"SumNumNO",10,le,"SumNHOH",5,le
```

```
Line 1:MolWt,450,le
Line 2:"LogP",5,le
Line 3:"SumNumNO",10,le
Line 4:SumNHOH,5,le
```

For *datafieldbylist* and *datafielduniquebylist* mode, input file line format is:

```
Line 1:fieldlabel;
```

Subsequent lines:value1,value2...

For *datafielduniquebylist* mode, only unique compounds identified by first occurrence of *value* associated with *fieldlabel* in *SDFfile(s)* are kept; any subsequent compounds are simply ignored. Example:

```
Line 1: MolID
Subsequent Lines:
907508
832291,4642
"1254", "907303"
```

--indelim *comma | tab | semicolon*

Delimiter used to specify text values for -d --datafields and --datafieldsfile options. Possible values: *comma, tab, or semicolon*. Default value: *comma*.

-m, --mode *alldatafields | commondatafields | datafields | datafieldsbyvalue | datafieldsbyregex | datafieldbylist | datafielduniquebylist | molnames | randomcmpds | recordnum | recordrange | 2dcmpdrecords | 3dcmpdrecords*

Specify what to extract from *SDFfile(s)*. Possible values: *alldatafields, commondatafields, datafields, datafieldsbyvalue, datafieldsbyregex, datafieldbylist, datafielduniquebylist, molnames, randomcmpds, recordnum, recordrange, 2dcmpdrecords, 3dcmpdrecords*. Default value: *alldatafields*.

For *alldatafields* and *molnames* mode, only a CSV/TSV text file is generated; for all other modes, however, a SD file is generated by default - you can change the behavior to generate text file using --output option.

For *3DCmpdRecords* mode, only those compounds with at least one non-zero value for Z atomic coordinates are retrieved; however, during retrieval of compounds in *2DCmpdRecords* mode, all Z atomic coordinates must be zero.

-n, --numofcmpds *number*

Number of compounds to extract during *randomcmpds* mode.

--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: *SD*. For *alldatafields* and *molnames* mode, this option is ignored and only a CSV/TSV text file is generated.

-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*.

--record *recnum | startrecnum,endrecnum*

Record number or range of records to extract during *recordnum* and *recordrange* mode. input value format is: <num> and <startnum, endnum> for *recordnum* and *recordrange* modes respectively. Default value: none.

--RegexIgnoreCase *yes or no*

Specify whether to ignore case during *datafieldsbyregex* value of -m, --mode option. 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: <SDFFileName><mode>.<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.

-s, --seed *number*

Random number seed used for *randomcmpds* mode. Default: 123456789.

--StrDataString *yes | no*

Specify whether to write out structure data string to CSV/TSV text file(s). Possible values: *yes or no*. Default value: *no*.

The value of StrDataStringDelimiter option is used as a delimiter to join structure data lines into a structure data string.

This option is ignored during generation of SD file(s).

--StrDataStringDelimiter *text*

Delimiter for joining multiple structure data lines into a string before writing to CSV/TSV text file(s). Possible values: *any alphanumeric text*. Default value: *.*

This option is ignored during generation of SD file(s).

--StrDataStringMode *StrOnly | StrAndDataFields*

Specify whether to include SD data fields and values along with the structure data into structure data string before writing it out to CSV/TSV text file(s). Possible values: *StrOnly or StrAndDataFields*. Default value: *StrOnly*.

The value of StrDataStringDelimiter option is used as a delimiter to join structure data lines into a structure data

string. This option is ignored during generation of SD file(s).

--ValueComparisonMode *Numeric | Alphanumeric*

Specify how to compare data field values during *datafieldsbyvalue* mode: Compare values using either numeric or string ((eq, le, ge) comparison operators. Possible values: *Numeric or Alphanumeric*. Default value: *Alphanumeric*.

-v, --violations *number*

Number of criterion violations allowed for values specified during *datafieldsbyvalue* and *datafieldsbyregex* mode. Default value: 0.

-w, --workingdir *dirname*

Location of working directory. Default: current directory.

## EXAMPLES

To retrieve all data fields from SD files and generate CSV text files, type:

```
% ExtractFromSDFfiles.pl -o Sample.sdf
% ExtractFromSDFfiles.pl -o *.sdf
```

To retrieve all data fields from SD file and generate CSV text files containing a column with structure data as a string with | as line delimiter, type:

```
% ExtractFromSDFfiles.pl --StrDataString Yes -o Sample.sdf
```

To retrieve MOL\_ID data field from SD file and generate CSV text files containing a column with structure data along with all data fields as a string with | as line delimiter, type:

```
% ExtractFromSDFfiles.pl -m datafields -d "Mol_ID" --StrDataString Yes
--StrDataStringMode StrAndDataFields --StrDataStringDelimiter "|"
--output text -o Sample.sdf
```

To retrieve common data fields which exists for all the compounds in a SD file and generate a TSV text file NewSample.tsv, type:

```
% ExtractFromSDFfiles.pl -m commondatafields --outdelim tab -r NewSample
--output Text -o Sample.sdf
```

To retrieve MolID, ExtReg, and CompoundName data field from a SD file and generate a CSV text file NewSample.csv, type:

```
% ExtractFromSDFfiles.pl -m datafields -d "Mol_ID,MolWeight,
CompoundName" -r NewSample --output Text -o Sample.sdf
```

To retrieve compounds from a SD which meet a specific set of criteria - MolWt <= 450, LogP <= 5 and SumNO < 10 - from a SD file and generate a new SD file NewSample.sdf, type:

```
% ExtractFromSDFfiles.pl -m datafieldsbyvalue -d "MolWt,450,le,LogP
,5,le,SumNO,10" -r NewSample -o Sample.sdf
```

To retrieve compounds from a SD file with a specific set of values for MolID and generate a new SD file NewSample.sdf, type:

```
% ExtractFromSDFfiles.pl -m datafieldbylist -d "Mol_ID,159,4509,4619"
-r NewSample -o Sample.sdf
```

To retrieve 10 random compounds from a SD file and generate a new SD file RandomSample.sdf, type:

```
% ExtractFromSDFfiles.pl -m randomcmpds -n 10 -r RandomSample
-o Sample.sdf
```

To retrieve compound record number 10 from a SD file and generate a new SD file NewSample.sdf, type:

```
% ExtractFromSDFfiles.pl -m recordnum --record 10 -r NewSample
-o Sample.sdf
```

To retrieve compound records between 10 to 20 from SD file and generate a new SD file NewSample.sdf, type:

```
% ExtractFromSDFfiles.pl -m recordrange --record 10,20 -r NewSample
-o Sample.sdf
```

## AUTHOR

Manish Sud <msud@san.rr.com>

## SEE ALSO

FilterSDFfiles.pl, InfoSDFfiles.pl, SplitSDFfiles.pl, MergeTextFilesWithSD.pl

---

**COPYRIGHT**

Copyright (C) 2004-2012 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.