myPresto 5.0

溶解度予測/アグリゲーター予測

サンプル

- LogS_Predictor -

USER MANUAL

2018/1/14

本ドキュメントについて

本ドキュメントは、「***myPresto* 5.0** USER MANUAL」の別冊です。コピーライト、プログラム使用許諾条件、著者および引用文献については、「***myPresto* 5.0** USER MANUAL」の記述に準じます。

謝辞

本ソフトウェアの研究開発は、新エネルギー・産業技術総合開発機構(NEDO)、及び、経済産業省(METI)の援助によって行われました。ここに感謝の意を記します。

本ソフトウェアは、故・京極好正博士の推進する研究プロジェクトで開発されました。

引用文献について

本ソフトウェアの使用時には、以下の文献を引用してください。

Sptool 溶解度(LogS)及び aggregator (frequent hitter)の予測について

Tadaaki Mashimo, Yoshifumi Fukunishi, Masaya Orita, Naoko Katayama, Shigeo Fujita, Haruki Nakamura, Quantitative analysis of aggregation-solubility relationship by in-silico solubility prediction, International Journal of High Throughput Screening, 2010:1, 99-107.

目次

1. 溶解度予測（アグリゲータ解析）サンプル概要

溶解度予測は、分子記述子に基づくある種の回帰（MLR, PLS, NN ...）で行う事が一般的です。本サンプルでもその方法を踏襲していますが、GB/SA・ASA といった物理記述子の追加および重み付き学習法の利用により計算精度の向上を図っています。
本サンプルでは、データベースに付属の約 1300 分子のデータセット、回帰計算に PLS 回帰を使用して溶解度予測を行います。更に、アグリゲータ分子の溶解度予測結果に対してベイジアン分析を行う事により、アグリゲータ予測を実施します。

2. 溶解度予測ツールのインストールとテストプログラムの実行

LogS_predictorYYMMDD.tar.gz は Linux/Unix 環境計で動作するように設計されています。(YYMMDD には年月日を示す数字が入ります。）ここでは、Linux 環境にインストールする手順を説明します。

2.1. インストール方法

LogS_predictorYYMMDD.tar.gz を、ユーザーが書き込み可能なディレクトリに配置してから、以下のコマンドを実行してください。インストールには、GNU の FORTAN コンパイラー(gfortran)もしくは、Intel の FORTRAN コンパイラー(ifort)が必要です

```
% tar -xzvf LogS_predictorYYMMDD.tar.gz
% cd LogS_predictorYYMMDD

次のコマンドは、どちらか一方を実行します。
% bin/install.sh            (GNU のコンパイラを使用する場合)
% bin/install.sh intel      (Intel のコンパイラを使用する場合)
```

2.2. テストプログラムの実行

次のコマンドで LogS_predictor に含まれるプログラムのテスト計算を実行できます。

```
% bin/test_LogS_predictor.sh
```

このコマンドを実行すると、sample17/に含まれるサンプルデータを使ったテスト計算を開始します。このプログラムの出力先ディレクトリは、test_LogS_Predictor_with_sample17/ です。このコマンドを実行することにより、LogS_predictor のプログラム群が適切に動作することを確認することができます。

3. 記述子計算（sample17/dsc/）

先ず、記述子の計算を溶解度予測の準備として実施します。記述子の計算は、溶解度既知のデータベースおよび溶解度未知の入力分子について行います。ここで計算する記述子は、薬物候補低分子用の拡張 Joback 記述子（構造記述子）および GB/SA・ASA といった物理記述子です（全 60 記述子－下表）。

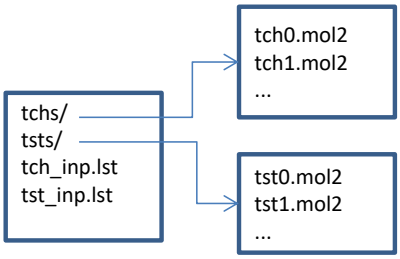
ID	Name	Description
1	MW	Mass weight of molecule
2	Ar	Number of aroamtic atoms
3	DelO_H	Number of dissociated H atoms, which bind O
4	AddN_H	Number of additional H atoms, which bind N
5	Ra	Number of atoms in ring
6	dASA_o	Accessible surface area par unit volume (in octanol)
7	dG_o	Solvation free energy evaluated by the GBSA method (in octanol)
8	dASA_wd	Accessible surface area par unit volume (in water, ion form)
9	dG_wd	Solvation free energy evaluated by the GBSA method (in water, ion form)
10	dASA_w	Accessible surface area par unit volume (in water)
11	dG_w	Solvation free energy evaluated by the GBSA method (in water)
12	ddG_o	dG_o / Accessible surface area (in octanol)
13	ddG_wd	dG_wd / Accessible surface area (in water, ion form)
14	ddG_w	dG_w / Accessible surface area (in water)
15	−CH3, CH4	Number of $-CH_3$ fragments or CH_4s
		… Joback-like sub-structure descriptors (15 – 60)…
60	>N-GinR	>N- group in ring which binds the atom with π electron.

物理記述子の計算は、水中およびオクタノール中での GB/SA・ASA 等の値を cosgene により計算します。その際、水中では Hgene で発生させた分子の解離型・非解離型について計算します。各条件・型の計算では、それぞれ confgene で発生させた複数配座について計算し、それらを平均したものを採用します。
一方、構造記述子の計算は、trans_code を使用して拡張 Joback 記述子の官能基の存在数をカウントします。

記述子計算の実行手順は以下の通りです。

① 分子の準備（sample17/dsc/）

データベースまたは入力分子は、格納ディレクトリ内に single-mol2 ファイルとして用意され、カレントディレクトリにその分子リストが用意されている必要があります（下図）。



分子が multi-mol2 ファイルで用意されている場合、split_mols 等のツールを使用して、single-mol2 ファイルに分割します（split_mols は tools に含まれています）。

```
$ mkdir tchs
$ split_mols -i tchs.mol2 -o tchs -p tch          ※ データベース
$ mkdir tsts
$ split_mols -i tsts.mol2 -o tsts -p tst          ※ 入力分子
```

分子リストは以下の内容のものを用意します。ID、ファイル名、logS 値を記述しますが、logS 値が無い場合は ID、ファイル名のみとします（なお、各 mol2 ファイルの @<TRIPOS> COMMENT 内に logS 値は記述可能で、その場合も ID、ファイル名のみとします）。

```
$ less tch_inp.lst
0   tch0.mol2          ※ logS 値が分かっているが、各 tch<n>.mol2 に記載済み
1   tch1.mol2             または、logS 値が分かっていない場合
…
$ less tst_inp.lst
0   tst0.mol2   -2.675 ※ logS 値が分かっており、各 tst<n>.mol2 には未記載で
1   tst1.mol2   -1.972    分子リストで指定する場合
…
```

② 計算条件の設定（sptool/Descriptor/）

指定可能な物理記述子の計算条件はそれぞれ指定個所が異なります。デフォルト値については各設定値を参照してください。

計算条件	指定個所
tplgeneL のパラメータ定義ファイル	Descriptor.rb 内の TPLLDBF
GBSAinp のパラメータ定義ファイル	Descriptor.rb 内の GBSADB
cosgene での最小化条件	min_template.erb
cosgene での GB/SA・ASA 計算条件	min_template_o.erb（オクタノール中） min_template_w.erb（水中）
confgene での発生配座数	Descriptor.rb 内の CNF_NUM
confgene での発生配座角度（360/<n>）	Descriptor.rb 内の CNF_ANG

指定可能な構造記述子の計算条件についても同様です。

計算条件	指定個所
trans_code の記述子定義ファイル	Descriptor.rb 内の TRNSDB

③ 計算の実行（sample17/dsc/）

現ディレクトリにおいて以下のコマンドで実行します（Descriptor/にパスが通っていない場合は 2 番目のコマンドとなります)。

1	$ Descriptor.rb -i tchs -l tch_inp.lst -o tch_out.dsc > tch_out.err 2>&1 $ Descriptor.rb -i tsts -l tst_inp.lst -o tst_out.dsc > tst_out.err 2>&1
2	$ ruby -I [path: Descriptor/] [path: Descriptor.rb] -i tchs -l tch_inp.lst -o tch_out.dsc > tch_out.err 2>&1 $ ruby -I [path: Descriptor/] [path: Descriptor.rb] -i tsts -l tst_inp.lst -o tst_out.dsc > tst_out.err 2>&1

※ スイッチ： "-i" = mol2 格納ディレクトリ, "-l" = mol2 リストファイル, "-o" = 記述子ファイル

※ 実行時には次のディレクトリをカレント以下に作成するので注意してください。

分子リストと同名ディレクトリ： テンポラリ、エラー詳細ファイルを格納

出力記述子ファイルと同名ディレクトリ： 物理記述子設定済み mol2 ファイルを格納

4. 溶解度推算（sample17/sol/）

次に 3.で計算した記述子について回帰計算を行います。回帰計算には PLS 回帰を利用します。重み付き学習法を適用して入力分子に対する学習データベースを作成し、入力分子と学習データベースのセットに対して PLS による回帰計算及び入力分子の溶解度推算を行います。その際、各回帰計算および推算結果全体について推算精度（決定係数）が計算されます（推算結果全体は入力分子にも logS 値が指定されている場合）。

溶解度計算の実行手順は以下の通りです。

① 記述子の準備（sample17/sol/）

3.のデータベースおよび入力分子の記述子をコピーします。入力分子の記述子に logS 値が指定されていれば推算精度が計算されます。なお、記述子ファイルの右端列に logS 値を設定する事も可能です（未設定の場合、logS 値列（右端）は 0.000000e+00）。

```
$ cp ../dsc/tch_out.dsc .
$ cp ../dsc/tst_out.dsc .
```

② 計算条件の設定（sptool/Solubility/）

指定可能な物理記述子の計算条件は Solubility.rb 内の当該個所で指定します。

計算条件	指定個所
記述子のサイズ設定	Solubility.rb 内の DSCS
重み付き学習法の bin 数	Solubility.rb 内の BINS
重み付き学習法の decoy 数	Solubility.rb 内の DCYS
補正計算の on/off	Solubility.rb 内の SWT_CRR

③ 計算の実行（sample17/sol/）

現ディレクトリにおいて以下のコマンドで実行します（インストール場所にパスが通っていない場合は 2 番目のコマンドとなります）。

1	$ Solubility.rb -i tst_out.dsc -d tch_out.dsc -o tst_sol.out -m w p > tst_sol.err 2>&1
2	$ ruby -I [path: Solubility/] [path: Solubility.rb] -i tst_out.dsc -d tch_out.dsc -o tst_sol.out -m w p > tst_sol.err 2>&1

※ スイッチ： "-i" = 入力記述子ファイル, "-d" = DB 記述子ファイル, "-o" = 溶解度推算結果,
"-m" = 重み付き学習法 {w(ln)/n(o)} 及び回帰関数 {p(ls)/m(lr)} 切り替え

※ 実行時には次のディレクトリをカレント以下に作成するので注意してください。
入力記述子ファイルと同名ディレクトリ： テンポラリ、エラー詳細ファイルを格納

④ 計算結果の内容（sample17/sol/）

得られた計算結果ファイル（tst_sol.out）には以下の様な項目が有ります。入力分子の記述子に logS 値が指定されていれば推算精度も計算されます。

推算結果：

項目	内容
ID	入力分子 ID
tchC1	データベースの回帰直線の係数 C1: y = C1 * x + C2
tchC2	データベースの回帰直線の係数 C2: y = C1 * x + C2
tchR2	データベースの決定係数
tchAveErr	データベースの平均エラー
tchMaxErr	データベースの最大エラー（絶対値が最大）
tchWL	重み付き学習時の追加デコイ数
tstExp	入力分子の logS（実験値）
tstPre	入力分子の logS（推算値）
tstErr	入力分子の推算エラー（絶対値）

推算精度：

項目	内容
tstC1	全入力分子の回帰直線の係数 C1: y = C1 * x + C2
tstC2	全入力分子の回帰直線の係数 C2: y = C1 * x + C2
tstR2	全入力分子の決定係数
tchAveErr	全入力分子の平均エラー
tchMaxErr	全入力分子の最大エラー（絶対値が最大）

5. アグリゲータ予測（sample17/agg/, sample17/agg_trgt/）

最後に、アグリゲータ予測の計算は以下の手順で行います。データベースの logS 頻度分布からベイジアン解析によって logS 依存のアグリゲータ存在確率を求め、これをシグモイド関数でフィッティングします。その後、得られたシグモイド関数を使って入力分子のアグリゲータ予測を行います。

アグリゲータ予測の手順は以下の通りです。

① 溶解度の推算（sample17/agg/）

3.の通りに記述子の計算を行い、4.の通りに溶解度推算を行います。

※ 溶解度推算には 3.で計算したデータベースを利用します。

```
$ cp ../dsc/tch_out.dsc sol/                    ※ 3.で計算したデータベースを利用
$ cd dsc
$ Descriptor.rb -i aggs -l agg_inp.lst -o agg_out.dsc > agg_out.err 2>&1
$ Descriptor.rb -i nons -l non_inp.lst -o non_out.dsc > non_out.err 2>&1
$ cp *_out.dsc ../sol
$ cd ../sol
$ Solubility.rb -i agg_out.dsc -d tch_out.dsc -o agg_sol.out -m w p > agg_sol.err 2>&1
$ Solubility.rb -i non_out.dsc -d tch_out.dsc -o non_sol.out -m w p > non_sol.err 2>&1
```

② logS 頻度分布の作成（sample17/agg/sol/）

溶解度の推算結果より、アグリゲータ及び非アグリゲータの logS 頻度分布を別個に作成します。計算は現ディレクトリにおいて以下のコマンドで実行します（インストール場所にパスが通っていない場合は 2 番目のコマンドとなります)。

1	$ FreqMaker.rb -i agg_sol.out -w 0.5 -o agg.frq $ FreqMaker.rb -i non_sol.out -w 0.5 -o non.frq
2	$ ruby -I [path: Aggregator/] [path: FreqMaker.rb] -i agg_sol.out -w 0.5 -o agg.frq $ ruby -I [path: Aggregator/] [path: FreqMaker.rb] -i non_sol.out -w 0.5 -o non.frq

※ スイッチ： "-i" = 溶解度推算ファイル, "-w" = ヒストグラム幅, "-o" = ヒストグラムファイル

その後、アグリゲータ及び非アグリゲータの頻度分布をエディタ等で一つに纏めます（これは手作業で行います)。纏めた logS 頻度分布は、以下の書式に従う様にします。

```
$ less agg_non.frq
-4.5   5   9                    ※ <logS> <アグリゲータ度数> <非アグリゲータ度数>
-4.0   5   11
…
```

③ ベイジアン解析及びフィッティングの実行（sample17/agg/sol/）

logS 頻度分布からベイジアン解析によりアグリゲータの存在確率を計算します。なお、入力に必要なシグモイド関数の係数初期値は、グラフ等を作成して適宜探索する必要があります。計算は現ディレクトリにおいて以下のコマンドで実行します（インストール場所にパスが通っていない場合は 2 番目のコマンドとなります）。

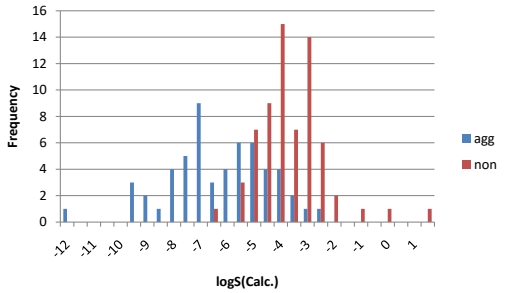
1	$ bys -d agg_non.frq	※ グラフ用データを作成（fitting しない）
	$ bys -d agg_non.frq -i 1.35 -4.7	※ 係数初期値はグラフを見て決定
2	$ [path: bys] -d agg_non.frq	
	$ [path: bys] -d agg_non.frq -i 1.35 -4.7	

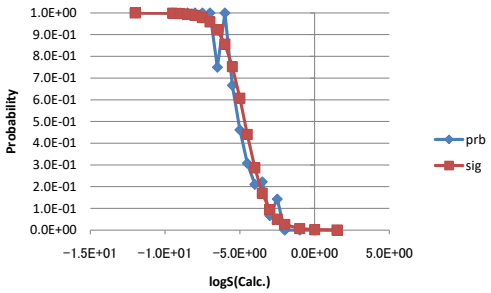
※ スイッチ： "-d" = データベースファイル, "-i" = シグモイド関数の係数初期値 c1 = a, c2 = b

係数値は入力ファイル名と同名の係数ファイル（agg_non.cff）に出力されます。

なお、フィッティングするシグモイド関数の定義は以下の通りです。

$$P_{agg}(\log S) = \frac{1}{1+e^{a(\log S - b)}}$$





④ アグリゲータの予測（sample17/agg_trgt/）

アグリゲータの予測は、フィッティングで得られたシグモイド関数を使って行います。予測対象の logS 実験値が存在しない場合、前述に倣って 3.記述子計算、4.溶解度推算を行います。ここでは LigandBox から抽出した下記条件（分子量および N 原子、O 原子の含有率）の組み合わせ、全 9 セットについて計算を行います。

MW(Da)	200 <= * < 300	300 <= * < 400	400 <= * < 500
% of N,O	0 <= * < 15	15 <= * < 30	30 <= * <= 100

まず、全 9 セットについて 3.の通りに記述子の計算を行い、4.の通りに溶解度推算を行います。

以下は MW={200<=*< 300}かつ%ofN,O={ 0<=*< 15}の例です。

```
$ cp ../dsc/tch_out.dsc dsc/                    ※ 1.2.で計算したデータベースを利用
$ cd dsc
$ Descriptor.rb -i pcts200-300_0-15 -l pcts200-300_0-15.lst -o pcts200-300_0-15_out.dsc
> pcts200-300_0-15_out.err 2>&1                 ※ 他の pcts<mw>_<no%>についても同様
$ cp pcts200-300_0-15_out.dsc ../sol
$ cd ../sol
$ Solubility.rb -i pcts200-300_0-15_out.dsc -d tch_out.dsc -o pcts200-300_0-15_sol.out -m w p
> pcts200-300_0-15_sol.err 2>&1                 ※ 他の pcts<mw>_<no%>についても同様
```

他の条件セットについても同様に計算します。

アグリゲータ予測は、③で得られたシグモイド関数の係数値を利用するので、当該ファイルを現ディレクトリ（sample17/agg_trgt/sol/）にコピーします。

```
$ cp ../../agg/sol/agg_non.cff .                ※ ③で計算した係数値を利用
```

計算は現ディレクトリにおいて以下のコマンドで実行します（インストール場所にパスが通っていない場合は 2 番目のコマンドとなります)。

以下は MW={200<=*< 300}かつ%ofN,O={ 0<=*< 15}の例です。

1	$ bys -c agg_non.cff -t pcts200-300_0-15_sol.out -n 10 9
2	$ [path: bys] -c agg_non.cff -t pcts200-300_0-15_sol.out -n 10 9

※ スイッチ： "-c" = 係数ファイル, "-t" = 入力ファイル, "-n" = 入力ファイル列数, logS 列番号

※ logS 列番号は 1 からカウント

予測結果は入力ファイル名と同名の結果ファイル（例：pcts200-300_0-15_sol.est）に出力されます。他の条件セットについても同様に計算します。

⑤ 予測結果の内容（sample17/agg_trgt/sol/）

得られた計算結果（例：pcts200-300_0-15_sol.est）では以下の様な項目が出力されます。

推算情報：

項目	内容
(c1, c2, err)	シグモイド関数の係数及びエラー
total_no	全入力分子数
avg_est(%)	平均アグリゲータ確率
avg_mkr	平均 logS 値

推算結果：

項目	内容
id	入力分子 ID
maker	入力分子 logS 値

estimation	入力分子アグリゲータ確率

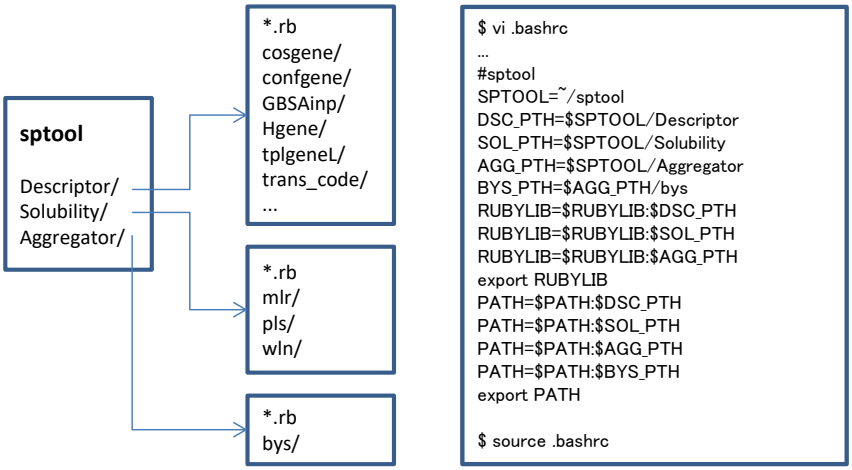
6. 溶解度予測ツール群（LogS_PredictorYYMMDD/sptool/）

sptool は溶解度予測を行う為のツール群です。分子の構造情報をコード化し（記述子計算）、回帰計算により未知の分子の溶解度を推算します。また、推算した溶解度を使用してアグリゲータの予測も行います。

ツール群の機能は以下の 3 つ分かれます。

名前	機能
Descriptor	記述子を計算する
Solubility	溶解度を推算する
Aggregator	アグリゲータを予測する

また、sptool は以下のディレクトリ構成となっています（下図左）。



プログラムの構築は、任意に配置した sptool 下の各種ツールのソースディレクトリ（src/等）に移動して適宜 make を実行して下さい。Ruby（ver1.8.5）スクリプトについては、起動ファイル（Descriptor.rb, Solubility.rb の SRC_DIR）にソースディレクトリの絶対パスを設定して下さい。また、環境変数の設定は必要に応じて行って下さい（上図右）。

6.1. Descriptor

分子の構造情報を読み込んで記述子を計算します。このツールは各記述子の計算に cosgene, confgene, Hgene, GBSAinp, tplgeneL, trans_code 等を利用します。インストール後、Descriptor.rb の SRC_DIR にソースディレクトリ（Descriptor/）を設定して下さい。

起動コマンド：

起動コマンドは以下の通りです。

Descriptor.rb -i <input_dir> -l <input_lst> -o <output_file>	
input_dir	single-mol2 ファイルを格納したディレクトリ名
input_lst	格納された single-mol2 ファイルのリスト

output_file	記述子ファイル名

Mol2 ファイルリスト：

入力に必要な mol2 ファイルリストは以下の様に記述子します。

<id> <mol2_file> [<logS>] （例： 4 mol4.mol2 -1.590）	
id	mol2 ファイルの ID
mol2_file	mol2 ファイル名（*.mol2）
logS	logS 値（実験値、任意） logS 値は実験値が分かっている場合に設定（データベースもしくは検証用） mol2 ファイル内@<TRIPOS>COMMENT 以下に記述されている場合には必要ない

物理記述子（Mol2 ファイル）：

mol2 ファイルの@<TRIPOS>COMMENT 以下に物理記述子を記述する事が出来ます。

記述は以下の書式行います。

#<identifier> <value> （例：#LogS -1.590）	
identifier	物理記述子（LogS, DelO_H, AddN_H, …）
value	設定値

記述可能な物理記述子は以下の通りです。

識別子	内容
LogS	logS 値
DelO_H	解離 H 原子数（-OH）
AddN_H	付加 H 原子数（-NH）
dG_o	オクタノール中の GB/SA 値
ddG_o	オクタノール中の GB/SA 値（単位表面積当たり）
dASA_o	オクタノール中の 表面積（単位体積当たり）
dG_w	水中の GB/SA 値
ddG_w	水中の GB/SA 値（単位表面積当たり）
dASA_w	水中の 表面積（単位体積当たり）
dG_wd	水中解離型の GB/SA 値
ddG_wd	水中解離型の GB/SA 値（単位表面積当たり）
dASA_wd	水中解離型の 表面積（単位体積当たり）

計算条件設定：

指定可能な物理記述子の計算条件はそれぞれ指定個所が異なります。デフォルト値については各設定値を参照してください。

計算条件	指定個所
tplgeneL のパラメータ定義ファイル	Descriptor.rb 内の TPLLDBF
GBSAinp のパラメータ定義ファイル	Descriptor.rb 内の GBSADB
cosgene での最小化条件	min_template.erb
cosgene での GB/SA・ASA 計算条件	min_template_o.erb（オクタノール中） min_template_w.erb（水中）
confgene での発生配座数	Descriptor.rb 内の CNF_NUM
confgene での発生配座角度（360/<n>）	Descriptor.rb 内の CNF_ANG

指定可能な構造記述子の計算条件も同様です。

計算条件	指定個所
trans_code の記述子定義ファイル	Descriptor.rb 内の TRNSDB

6.2. Solubility

重み付き学習法による強化学習を伴った回帰計算を実行します。インストール後、Solubility.rb の SRC_DIR にソースディレクトリ（Solubility/）を設定して下さい。

起動コマンド：

起動コマンドは以下の通りです。

Solubility.rb -i <input_file> -o <output_file> -d <db_file> -m <w/n> <p/m>	
input_file	入力分子記述子ファイル名（Descriptor.rb の output_file を想定）
output_file	回帰計算結果ファイル名
db_file	データベース記述子ファイル名（Descriptor.rb の output_file を想定）
w/n	学習選択 w(eighted learning) / n(o learning)
m/p	回帰選択 m(lr) / p(ls)

回帰計算結果の内、推算結果は以下の様な書式になります。

推算結果：

項目	内容
ID	入力分子 ID
tchC1	データベースの回帰直線の係数 C1: y = C1 * x + C2
tchC2	データベースの回帰直線の係数 C2: y = C1 * x + C2
tchR2	データベースの決定係数
tchAveErr	データベースの平均エラー
tchMaxErr	データベースの最大エラー（絶対値が最大）
tchWL	重み付き学習時の追加デコイ数
tstExp	入力分子の logS（実験値）
tstPre	入力分子の logS（推算値）
tstErr	入力分子の推算エラー（絶対値）

推算精度：

項目	内容
tstC1	全入力分子の回帰直線の係数 C1: y = C1 * x + C2
tstC2	全入力分子の回帰直線の係数 C2: y = C1 * x + C2
tstR2	全入力分子の決定係数
tchAveErr	全入力分子の平均エラー
tchMaxErr	全入力分子の最大エラー（絶対値が最大）

6.3. FreqMaker

logS の頻度分布を作成します。

起動コマンド：

起動コマンドは以下の通りです。

FreqMaker.rb -i <input_file> -w <width> -o <output_file>	
input_file	入力 logS ファイル名（Solubility.rb の output_file を想定）
width	ヒストグラム幅
output_file	ヒストグラムファイル名

ヒストグラムファイル：

出力するヒストグラムファイルは以下の様な書式になります。

<logS>　<frequency>	（例：　-1.50　24）
logS	logS 値（<width>刻み）
frequency	頻度

6.4. wln

重み付き学習を行います。入力した各分子について強化学習データベースを用意します。

起動コマンド：

起動コマンドは以下の通りです。

wln -t <input_file> -d <db_file> -b <bin> -c <decoy> -n <dsc_num>	
input_file	入力分子記述子ファイル名
db_file	データベース記述子ファイル名
bin	類似度分布の分散の bin 数
decoy	最大強化学習回数（= decoy 数）
dsc_num	記述子列数（拡張 Joback 記述子サイズ = 60 記述子, id 及び logS を除く）
OUTPUT	a) 入力分子記述子ファイル（<input_file>-<id>.dsc, <id>は入力分子の行番号） b) 強化学習データベース（<db_file>-<id>.dsc, <id>は入力分子の行番号） c) 類似度分布ファイル（*.dst） d) 強化学習分子リスト（*.lst）

入力分子記述子ファイル：

出力する入力分子記述子ファイルは以下の様な書式になります。

<id> <dsc1> <dsc2> … <dscn>	（例： 1 0.150000000E+02 … ）
id	番号
dscN	N 番目の記述子（N=1, 2, …, n）

強化学習データベース：

出力する強化学習データベースは以下の様な書式になります。

<id> <dsc1> <dsc2> … <dscn>	（例： 2 0.678432144E+03 … ）
id	番号
dscN	N 番目の記述子（N=1, 2, …, n）

類似度分布ファイル：

出力する強化学習データベースは以下の様な書式になります（wln.f : CALC_CHR=.true.）。

# bin(sd/<bin>), freq.	（例： # bin(sd/30), freq ）
<bid> <freq>	（例： 1 6 ）
bin	bin 数
bid	bin 番号
freq	度数

強化学習分子リスト：

出力する強化学習分子リストは以下の様な書式になります（wln.f：CALC_LST=.true.）。

# similarity avg: <avg> similarity sd: <sd> （例： # similarity avg: 0.4233 similarity sd: 0.2874 ） # id similarity, decys(<dcys>) （例： # id, similarity, decoys(6) ） <id> <similarity> <dcy> （例： 635 -0.1526 6 ）	
avg	similarity 平均
sd	similarity 標準偏差
dcys	decoy 総数
id	番号
similarity	similarity 値
dcy	decoy 数

6.5. pls

PLS 回帰（Partial Least Squares Regression）分析を行います。

起動コマンド：

起動コマンドは以下の通りです（5-1 と 5-2 または 5-3 のみ）。

(5-1) 回帰計算

pls -d <db_file> -n <dsc_num>	
db_file	データベース記述子ファイル名
dsc_num	記述子列数（拡張 Joback 記述子サイズ = 60 記述子, id 及び logS を除く）
OUTPUT	a) データベース回帰結果ファイル（*.rgr） b) 回帰係数ファイル（*.cff）

(5-2) 入力評価

pls -t <input_file> -c <coeff_file> -n <dsc_num>	
input_file	入力分子記述子ファイル名
coeff_file	回帰係数ファイル名
dsc_num	記述子列数（拡張 Joback 記述子サイズ = 60 記述子, id 及び logS を除く）
OUTPUT	a) 評価結果ファイル（*.cal）

(5-3) 回帰計算および入力評価

pls -t <input_file> -d <db_file> -n <dsc_num>	
input_file	入力分子記述子ファイル名
db_file	データベース記述子ファイル名
dsc_num	記述子列数（拡張 Joback 記述子サイズ = 60 記述子, id 及び logS を除く）
OUTPUT	a) データベース回帰結果ファイル（*.rgr） b) 評価結果ファイル（*.cal） c) 回帰係数ファイル（*.cff）

データベース回帰結果ファイル：

出力するデータベース回帰結果ファイルは以下の様な書式になります。

# Regression results: R^2= <R^2> （例： # Regression results: R^2= 0.860515773E+00 ） <id> <exp> <cal> （例： 1 0.158000004E+01 … ）	
R^2	決定係数
id	番号
exp	実験値
cal	推算値

評価結果ファイル：

出力する評価結果ファイルは以下の様な書式になります。

# Regression results: R^2= <R^2> （例： # Regression results: R^2= 0.860515773E+00 ）	
<id> <exp> <cal> （例： 1 0.620433807E+00 … ）	
R^2	決定係数
id	番号
exp	実験値
cal	推算値

回帰係数ファイル：

出力する回帰係数ファイルは以下の様な書式になります。

# Regression coefficients: r, c(r,1…dscN), beta(r,0…1) → このタイトル以降に出力	
<r> <c(r,dsc1)> <c(r,dsc2)> …<c(r,dscn)> <beta(r,0)> <beta(r,1)> （例： 1 -0.392615944E+00 … ）	
# Database statistics: d=0,y_avg,ysig | d>0,x_avg(d),x_sig(d) → このタイトル以降に出力	
<d=0> <y_avg> <y_sig> （例： 0 -0.305323744E+01 … ）	
<d>0> <x_avg(d)> <x_sig(d)> （例： 1 0.211661744E+01 … ）	
r	潜在変数 T(r)の番号
c(r,dsc)	潜在変数 T(r)の第 dsc 項の重み（T(r) = Sum[dsc=1,n]{ c(r,dsc) * x(dat,dsc) }）
beta(r,typ)	潜在変数 T(r)の回帰係数（y(dat) = Sum[k=1,r]{ beta(k,0) + beta(k,1) * T(k) }）
d	記述子番号
y_avg	データベースの y 平均値（d=0 のみ）
y_sig	データベースの y 標準偏差（d=0 のみ）
x_avg(d)	データベースの x(d)平均値（d>0）
x_sig(d)	データベースの x(d)標準偏差（d>0）

PLS 分析ファイル：

出力する PLS 分析ファイルは以下の様な書式になります（pls.f：CALC_STS=.true.）。

<r> <tR^2> <tAcR^2> <yR^2> <yAcR^2> <PRESS> <Q^2> （例： 1 0.02143 … ）	
r	潜在変数 T(r)の番号
tR^2	因子寄与率
tAcR^2	因子累積寄与率
yR^2	応答寄与率
yAcR^2	応答累積寄与率
PRESS	PRESS (predicted residual sum of squares)
Q^2	Q^2

mlr

重回帰（multiple liner regression）分析を行います。

起動コマンド：

起動コマンドは以下の通りです（6-1 と 6-2 または 6-3 のみ）。

(6-1) 回帰計算

mlr -d <db_file> -n <dsc_num>	
db_file	データベース記述子ファイル名
dsc_num	記述子列数（拡張 Joback 記述子サイズ = 60 記述子, id 及び logS を除く）
OUTPUT	a) データベース回帰結果ファイル（*.rgr） b) 回帰係数ファイル（*.cff）

(6-2) 入力評価

mlr -t <input_file> -c <coeff_file> -n <dsc_num>	
input_file	入力分子記述子ファイル名
coeff_file	回帰係数ファイル名
dsc_num	記述子列数（拡張 Joback 記述子サイズ = 60 記述子, id 及び logS を除く）
OUTPUT	a) 評価結果ファイル（*.cal）

(6-3) 回帰計算および入力評価

mlr -t <input_file> -d <db_file> -n <dsc_num>	
input_file	入力分子記述子ファイル名
db_file	データベース記述子ファイル名
dsc_num	記述子列数（拡張 Joback 記述子サイズ = 60 記述子, id 及び logS を除く）
OUTPUT	a) データベース回帰結果ファイル（*.rgr） b) 評価結果ファイル（*.cal） c) 回帰係数ファイル（*.cff）

データベース回帰結果ファイル：

出力するデータベース回帰結果ファイルは以下の様な書式になります。

# Regression results: R^2= <R^2> （例： # Regression results: R^2= 0.860515773E+00 ） <id> <exp> <cal> （例： 1 0.158000004E+01 … ）	
R^2	決定係数
id	番号
exp	実験値
cal	推算値

評価結果ファイル：

出力する評価結果ファイルは以下の様な書式になります。

# Regression results: R^2= <R^2> （例： # Regression results: R^2= 0.860515773E+00 ） <id> <exp> <cal> （例： 1 0.620433807E+00 … ）	
R^2	決定係数
id	番号
exp	実験値
cal	推算値

回帰係数ファイル：

出力する回帰係数ファイルは以下の様な書式になります。

<id> <a(dsc1)> <a(dsc2)> …<a(dscn)> <b> （例： 1 -0.392615944E+00 … ）	
id	番号
a(dsc)	回帰係数 a （y(dat) = Sum[dsc=1,n]{ a(dsc) * x(dat,dsc) } + b）
b	回帰係数 b

6.6. bys

ベイジアン解析を行います。ベイズ推定を行った後、シグモイド関数でフィッティングします。フィッティングするシグモイド関数の定義は以下の通りです（c1=a, c2=b）。

$$P_{agg}(\log S) = \frac{1}{1+e^{a(\log S - b)}}$$

起動コマンド：

起動コマンドは以下の通りです（7-1 と 7-2 または 7-3 のみ）。

(7-1) ベイジアン解析

bys -d <db_file> -i <c1_value> <c2_value>	
db_file	頻度分布ファイル名（データベース）
c1_value	シグモイド関数の係数 C1 初期値
c2_value	シグモイド関数の係数 C2 初期値
OUTPUT	a) データベース解析結果ファイル（*.prb） b) シグモイド係数ファイル（*.cff）

(7-2) 入力評価

bys -t <input_file> -n <dsc_num> <dsc_idx> -c <coeff_file>	
input_file	入力ファイル名（回帰結果）
dsc_num	列サイズ（回帰結果列数 = 10 列）
dsc_idx	列番号 （logS の列番号 = 9、列番号 1～10）
coeff_file	シグモイド係数ファイル名
OUTPUT	a) 評価結果ファイル（*.est）

(7-3) ベイジアン解析および入力評価

bys -t <input_file> -n <dsc_num> <dsc_idx> -d <db_file> -i <c1_value> <c2_value>	
input_file	入力ファイル名（回帰結果）
dsc_num	列サイズ（回帰結果列数 = 10 列）
dsc_idx	列番号 （logS の列番号 = 9、列番号 1～10）
db_file	頻度分布ファイル名（データベース）
c1_value	シグモイド関数の係数 C1 初期値
c2_value	シグモイド関数の係数 C2 初期値
OUTPUT	a) 評価結果ファイル（*.est） b) データベース解析結果ファイル（*.prb） c) シグモイド係数ファイル（*.cff）

頻度分布ファイル（データベース）：

入力する頻度分布ファイル（データベース）は以下の様な書式になります。

<marker> <state_frq > <non_state_frq>	（例： -1.50　24　 59）
marker	logS 値
state_frq	アグリゲータ度数
non_state_frq	非アグリゲータ度数

評価結果ファイル：

出力する評価結果ファイルは以下の様な書式になります。

# Sigmoid function: (c1,c2,err) = (<c1>,<c2>,<err>)	（例： (c1,c2,err) = (0.2143　0.2143　0.1822) ）
# total_no=<total_no>	（例： total_no = 58）
# avg_est(%)=<avg_est>　avg_mkr=<avg_mkr>	（例： avg_est(%) = 41.230　avg_mkr = -4.132）
<id> <maker> <estimation>	（例： 0　-0.392615944E+01 … ）
c1	シグモイド関数係数 C1
c2	シグモイド関数係数 C1
err	シグモイド関数フィッティングエラー
total_no	評価対象数
avg_est	平均評価値（%）
avg_mkr	平均 maker 値
id	番号
maker	maker 値
estimation	評価値

データベース解析結果ファイル：

出力するデータベース解析結果ファイルは以下の様な書式になります。

# Sigmoid function: (c1,c2,err) = (<c1>,<c2>,<err>)	（例： (c1,c2,err) = (0.2143　0.2143　0.1822) ）
<id> <maker> <estimation>	（例： 0　0.479493350E+00 … ）
c1	シグモイド関数係数 C1
c2	シグモイド関数係数 C1
err	シグモイド関数フィッティングエラー
id	番号
maker	maker 値
estimation	評価値

シグモイド係数ファイル：

出力するシグモイド係数ファイルは以下の様な書式になります。

<id> <c1> <c2> <err>	（例： 1　1.355000002E+01 … ）
id	番号
c1	シグモイド関数係数 C1
c2	シグモイド関数係数 C1
err	シグモイド関数フィッティングエラー

6.7. trans_code

Ullmann の定理を利用して入力分子に含まれる官能基数をカウントします。官能基の定義は薬物候補低分子用の拡張 Joback 記述子です（定義ファイルは DB/に配置）。

起動コマンド：

起動コマンドは以下の通りです。

(8-1) 分子構造から記述子に変換（mol2 形式 DB）

trans_code -i <input_file> -o <output_file> -d <db_file>	
input_file	分子構造ファイル名（mol2 ファイル、multi-mol2 形式も可）
output_file	記述子ファイル名
db_file	官能基定義ファイル名（multi-mol2 形式、DB 用）

(8-2) 分子構造から記述子に変換（binary 形式 DB）

trans_code -i <input_file> -o <output_file> -d <db_file> -b	
input_file	分子構造ファイル名（mol2 ファイル、multi-mol2 形式も可）
output_file	記述子ファイル名
db_file	官能基定義ファイル名（binary 形式、DB 用）
“-b”	binary 利用スイッチ

(8-3) mol2 形式 DB から binary 形式 DB に変換

trans_code -i <input_file> -o <output_file> -c	
input_file	官能基定義ファイル（multi-mol2 形式、DB 用）
output_file	官能基定義ファイル名（binary 形式、DB 用）
“-c”	binary 生成スイッチ

物理記述子（Mol2 ファイル）：

mol2 ファイルの@<TRIPOS>COMMENT に以下の記述があれば、trans_code より出力される記述子ファイルに以下の物理記述子の項目が含まれる様になります。

記述は以下の様に行います。

#<identifier> <value>	（例：#LogS　-1.590）
identifier	物理記述子（LogS, DelO_H, AddN_H, …）
value	設定値

記述可能な物理記述子は以下の通りです。

識別子	内容
LogS	logS 値
DelO_H	解離 H 原子数（-OH）
AddN_H	付加 H 原子数（-NH）
dG_o	オクタノール中の GB/SA 値
ddG_o	オクタノール中の GB/SA 値（単位表面積当たり）
dASA_o	オクタノール中の 表面積（単位体積当たり）
dG_w	水中の GB/SA 値
ddG_w	水中の GB/SA 値（単位表面積当たり）
dASA_w	水中の 表面積（単位体積当たり）
dG_wd	水中解離型の GB/SA 値
ddG_wd	水中解離型の GB/SA 値（単位表面積当たり）
dASA_wd	水中解離型の 表面積（単位体積当たり）

記述子ファイル：

出力する記述子ファイルは以下の様な書式になります。

<id>　<dsc1>　<dsc2>　…　<dscn>	（例：　0　0.678432144E+03 … ）
id	番号
dscN	記述子（N=1, 2, …, n）

以上