alvaDesc 3.0

コマンドライン入門

株式会社アフィニティサイエンス

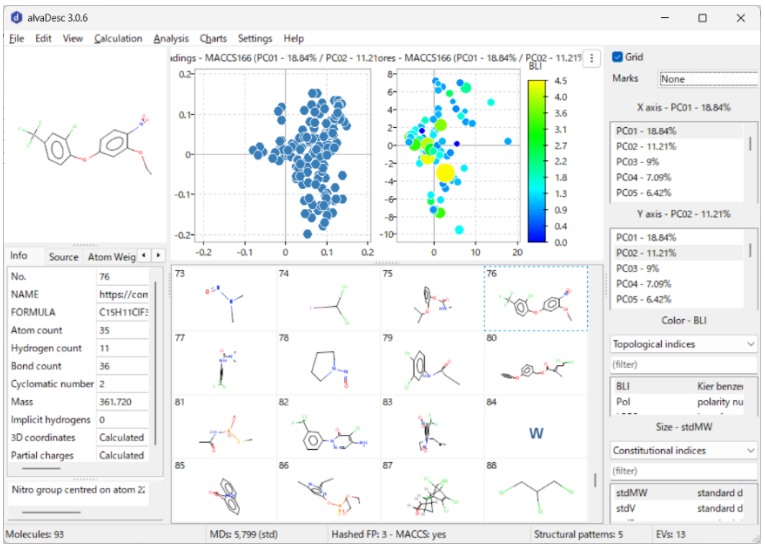
Revision: 2025 年 6 月 27 日 (1.0.0)

株式会社アフィニティサイエンス
〒141-0031 東京都品川区西五反田 1-11-1 アイオス五反田駅前

TEL:　03-6417-3695
FAX:　03-6417-3696
Email:　help@affinity-science.com / sales@affinity-science.com
Web:　https://www.affinity-science.com

目次

1. はじめに




```
alvaDescCLI --script=myscript.adscr
```

alvaDesc は、6000 種近くの分子記述子と 4 種のフィンガープリントを計算・解析することができるソフトウェアです。分子記述子のうち、3 次元座標なしで計算できるものは 4000 種以上、3 次元記述子も約 1600 種実装されています。

塩やイオン液体、金属錯体といった非完全結合型の分子の計算が可能で、相関分析や主成分分析(PCA)、t-SNE 分析といった簡易的な分析機能なども備えています。また、バージョン 2.0 から 3.0 へのメジャーアップデートでは 3 次元座標の計算機能も追加され、2 次元の構造ファイルからでも全ての記述子を計算できるようになりました。また、ユーザ定義による構造パターンの計算機能も追加されました。

alvaDesc のグラフィカルユーザーインターフェース（GUI）では、シンプルで直感的なインターフェースを使用して、これらの計算や解析を簡単に行うことができます。

このチュートリアルは、alvaDesc を GUI で使われている方やコマンドラインインターフェース（CLI）で使い始めた方に対し、CLI からの基本的な操作方法を紹介する内容となっています。alvaDesc の基本機能については、姉妹編チュートリアル「alvaDesc スタートアップガイド」で詳しく紹介していますので、まずそちらをご覧になってからこのチュートリアルをお読みになると、理解が一層進むかと存じます。より深く知りたいという方は、alvaDesc のユーザーマニュアルも併せてご参照ください。

このチュートリアルを、皆様の業務や研究にお役立ていただければ幸いです。

※本資料は alvaDesc バージョン 3.0.6 を使用して作成しています。

（尚、以降、分子記述子を「記述子」、フィンガープリントを「FP」と表記します。）

2. 概要

alvaDesc を使用する際、入力する化合物数がとても多い場合などには、リアルタイムでインタラクティブに計算処理を確認していく GUI からではなく、バックグラウンドのジョブとして実行、または、独立したバッチ処理として実行させる CLI からの使用がとても便利です。Windows コマンドプロンプトや Windows Power Shell、Linux や macOS の Bash や Zsh などのいろいろな CLI からの実行は、alvaDesc を他のアプリ等と組み合わせて使う際のベースともなります。

alvaDesc のインストーラには GUI から実行するための exe ファイル（Windows の場合 alvaDescGUI.exe）と共に、コマンドラインインターフェイス（CLI）から実行するための exe ファイル（Windows の場合 alvaDescCLI.exe）が含まれています。インストールを行うことにより、GUI からだけでなく、CLI からも実行可能になっていますが、CLI からの実行には多少準備が必要となります。その準備については次の第 3 章で説明します。

CLI からの実行方法は、コマンドライン上で必要な条件や入出力等について直接入力し計算させる方法と、それらの条件等をスクリプトファイルという XML（Extensible Markup Language）フォーマットのファイルに記載したものを CLI の exe ファイルに投入する方法があり、前者を 4 章で説明します。スクリプトファイルはテキストエディタで書くこともできますが、GUI のスクリプトウィザードを使って簡単に作成することができますので、その内容については 5 章で説明します。
更に発展的な使い方として、Python を使ってワークフローに組み込むこともできます。7 章では、その概略について説明します。

3. 準備

この章では、alvaDesc をコマンドラインから使うための事前準備について説明します。

3.1 実行ファイル「alvaDescCLI」について

3.1.1 格納ディレクトリ

alvaDescCLI 実行ファイルは、インストール時に alvaDescGUI 実行ファイルと同じディレクトリに格納されています。
以下に OS 毎の標準的な格納場所を示します。

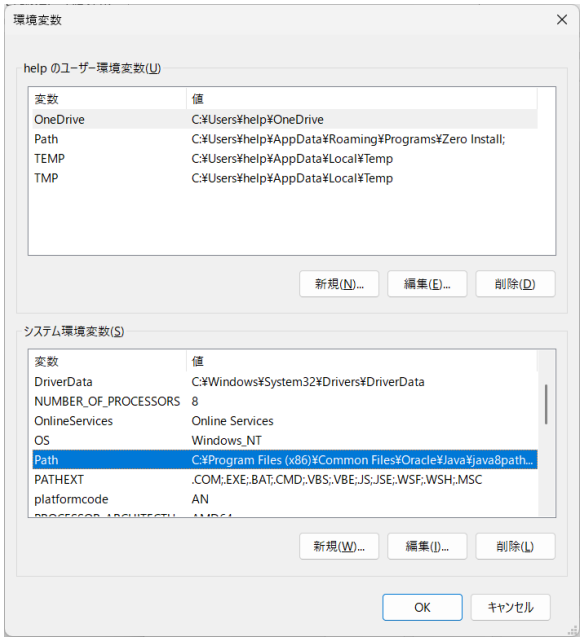
OS	alvaDescCLI のパス（path）
Windows	C:¥Program Files¥Alvascience¥alvaDesc¥alvaDescCLI.exe
Linux	/usr/bin/alvaDescCLI
macOS	/Applications/alvaDesc.app/Contents/MacOS/alvaDescCLI

3.1.2 環境変数 PATH の設定

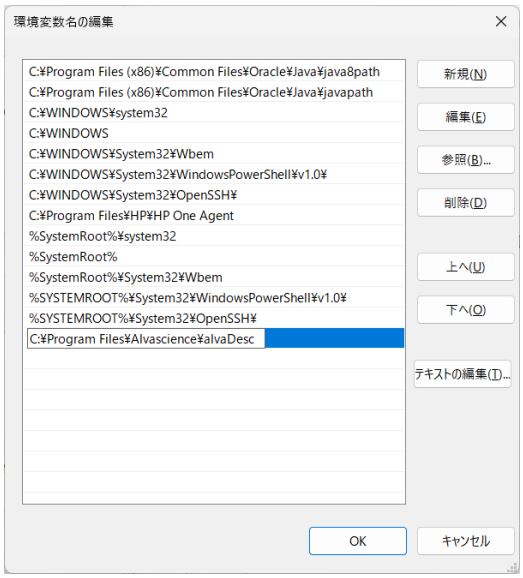
上に示した格納場所にある alvaDescCLI を普段使っているディレクトリから呼び出すことができるよう、環境変数や PATH を設定しておく必要があります。

- Windows の場合（Windows 11）
  - [設定] > [システム] > [システムの詳細設定] > [システムのプロパティ] > [環境変数...] （または、[設定] ホームの[設定の検索] から「環境変数」を検索し 「システム環境変数の編集」を選択）で表示される環境変数ウィンド

ウ下段［システム環境変数］の［Path］を選択し［編集］を押下します。



➢ ［環境変数名の編集］画面で［新規］を選び「C:¥Program Files¥Alvascience¥alvaDesc」を記入し［OK］>［OK］>［OK］。



- Linux の場合

➢ deb や rpm パッケージを管理者権限でインストールした場合、 /usr/bin/以下に実行ファイルが格納されます。/usr/bin/は環境変数 PATH に登録されているので、追加で PATH を設定する必要はありません。

➢ tar ファイルなどでインストールした場合など、/usr/bin/以外の PATH の通っていないディレクトリに実行ファイルがある際は、PATH の設定を行う必要があります。(以下の例では、/home/my_apps/ 以下に実行ファイルが

格納されているものとします。)

✧　シェルが bash の場合

「export PATH=$PATH:/home/my_apps/」をユーザのホームディレクトリに格納された.bashrc の末尾に書き込みます。起動済みの bash で設定したパスを使用する場合は、source コマンドで変更内容を反映させます。

```
echo 'export PATH=$PATH:/home/my_apps/' >> ~/.bashrc
source ~/.bashrc
```

✧　シェルが zsh の場合

bash の場合と同じ内容を.zshrc に書き込みます。

```
echo 'export PATH=$PATH:/home/my_apps/' >> ~/.zshrc
source ~/.zshrc
```

✧　シェルが fish の場合

ユニバーサル変数 fish_user_paths に/home/my_apps /のパスをセットし、set コマンドのオプション -U により設定を永続化します。

```
set -U fish_user_paths /home/my_apps/ $fish_user_paths
```

- macOS の場合
  - /Applications/alvaDesc.app/Contents/MacOS/ を PATH に設定します。

✧　シェルが bash の場合

「export PATH=$PATH: /Applications/alvaDesc.app/Contents/MacOS/」をホームディレクトリに格納された.bash_profile の末尾に書き込みます。起動済みの bash で設定したパスを使用する場合は、source コマンドで変更内容を反映させます(ログインシェルではなく、インタラクティブシェルに設定する場合、.bash_profile を.bashrc へ置き換えて実行してください)。

```
echo 'export PATH=$PATH:/Applications/alvaDesc.app/Contents/MacOS/' >>
~/.bash_profile
source ~/.bash_profile
```

✧　シェルが zsh の場合

bash の場合と同じ内容を.zshrc に書き込みます。

```
echo 'export PATH=$PATH:/Applications/alvaDesc.app/Contents/MacOS/' >>
~/.zshrc
source ~/.zshrc
```

【Note】記号”>>”と“~”の意味について
上記で使われている”>>”は標準出力をファイルに出力させる「リダイレクト」を表わしている記号です。リダイレクトには 2 種類あり、”>”はその直後に提示されるファイル名がない場合には新規作成を行い、既存のファイルがある場合は置き換えを行います。これに対し、”>>”は既存ファイルがある場合にファイルの末尾への書き込みを意味します。
“~”（チルダ）はユーザのホームディレクトリを表わしています。

3.2 入出力形式とコマンドヘルプについて

3.2.1 CLI からの実行方法

alvaDesc を CLI から実行する場合、大きく 2 つの方法があります。

1. コマンドライン上で計算条件（引数）を入力する方法
   4 章で使用法の具体例を見ていきます。
2. 予め作成したスクリプトファイルを用いる方法
   5 章でその詳細を見ていきます。

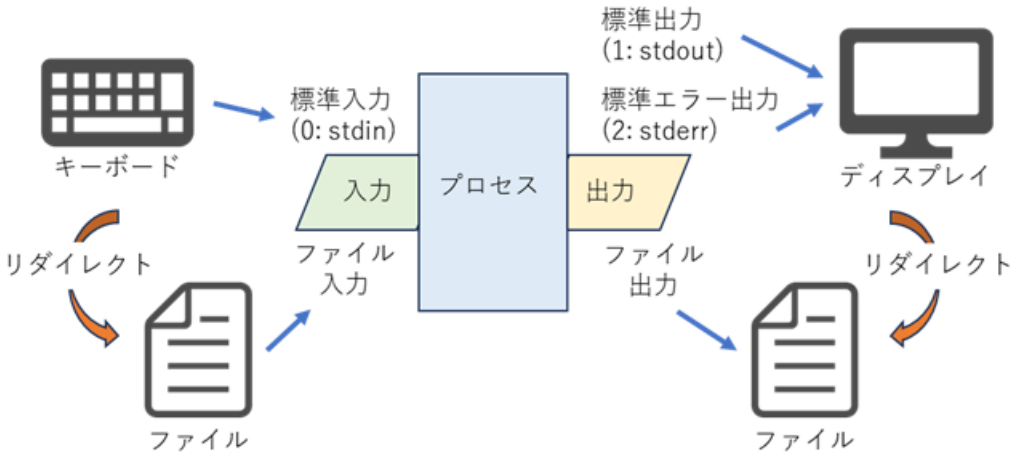
以下では、それらの実行方法に共通する、入出力形式について説明します。

3.2.2 標準入出力とファイル入出力

alvaDesc を CLI から実行する場合、入力する化合物データと出力する計算結果の形式を標準入出力とすることも、ファイル入出力とすることもできます。
通常、標準入出力とはキーボードからの入力になり、標準出力とはターミナル画面（ディスプレイ）上での出力となります。ファイル入出力とは、指定したファイルからデータをインポートしたり、指定したファイルに計算結果をエクスポートしたりすることを言いますが、標準入出力をリダイレクトすることによりファイルと組み合わせることもできます。

- 入力
  - コマンドライン上で alvaDescCLI を実行する際、引数として[--iSMILES=化合物の SMILES 表記] を使うことでキーボード入力により化合物の構造を入力できますが、多くの化合物を入力することには向かないため、通常、ファイルから入力を行います。
  - 引数［--input］を使用し、[< ファイル名］を使うことにより標準入力をリダイレクトし、指定ファイルから化合物をインポートできます。この場合には、［--inputtype=SMILES|MDL|SYBYL|HYPERCHEM］により、構造のフォーマットを指定する必要があります。
  - 引数［--input=ファイル名］によりファイル入力を行うこともできます。この場合には、構造のフォーマットは構造ファイルの拡張子で判断されるため、--inputtype の指定は必要ありません。
- 出力
  - コマンドライン上で alvaDescCLI を実行する際、出力の指定を何もしない場合には標準出力として画面上に結果が表示されます。
  - 引数［--output］を使用し、[> ファイル名］によりファイル名を指定することにより、標準出力をリダイレクトし、ファイルに出力させることもできます。この場合、[--outputtype=TXT|SMILES|MDL］により出力ファイルのフォーマットを指定する必要があります。尚、[--output]、[--outputtype］は［--input］と同時に使用する必要があることに留意が必要です。

- alvaDescCLI の引数として［--output=ファイル名］を指定することにより、計算結果をファイルに出力することができます。この場合、カレントディレクトリ内に指定したファイルがあれば上書き、ない場合にはファイルが新規作成されます。



3.2.3 エラー出力とログファイル

alvaDescCLI では、実行時のエラーやログを出力することもできます（ここで述べるログは、GUI の Help メニュー ＞ Log file から設定するデバッグ用ログファイルとは異なるものです)。

- エラー
  - 何も指定をしていない場合には、標準エラーとして画面表示されます。これを［2>ファイル名］によりリダイレクトし、ファイルに出力できます。
- ログ
  - 実行時のログは通常、標準エラーに出力されます。スクリプトファイルを使用する場合、ログを出力しない、標準エラーに出力、別ファイルに出力のオプションが選択可能です。

3.2.4 コマンドヘルプ

alvaDescCLI の引数として［-h]もしくは［--help]を指定することにより CLI 上で使い方やオプションを確認することができます。

```
C:¥Users¥user1¥temp>alvaDescCLI -h
alvaDescCLI.
Version 3.0 for Win64 (x86_64) - v.3.0.6 - built on: 2025/01/27

Usage:
alvaDescCLI --input[=<molecule_file>] --
inputtype=(SMILES|MDL|SYBYL|HYPERCHEM) --descriptors=<descriptor_list> --
output[=<output_file>] [--outputtype=<output_filetype>] [--labels] [--
threads=<nthreads>] [--verbose]
.....
.....途中省略
alvaDescCLI -h | --help

Options:
--input[=<molecule_file>] : Input file [default: standard input]
--inputtype=(SMILES|MDL|SYBYL|HYPERCHEM) : Input file type. Mandatory if
input file name is omitted (standard input mode)
.....
.....以下省略
```

オプションの詳細については「4.3 使用できる引数」をご参照下さい。

【Note】Windows の PowerShell でコマンドを実行するには

実行したいコマンドの前に「cmd /c」を付けると、直接コマンドを実行することができます。

例、PS C¥Users¥user1＞ cmd /c alvaDescCLI.exe -h

4. 基本の CLI 操作

ここからは、Windows のコマンドプロンプトを使用した具体例を通じて、基本の CLI 操作とその結果を見ていきます。
（ユーザ：user1、実行ディレクトリ：temp とします。）

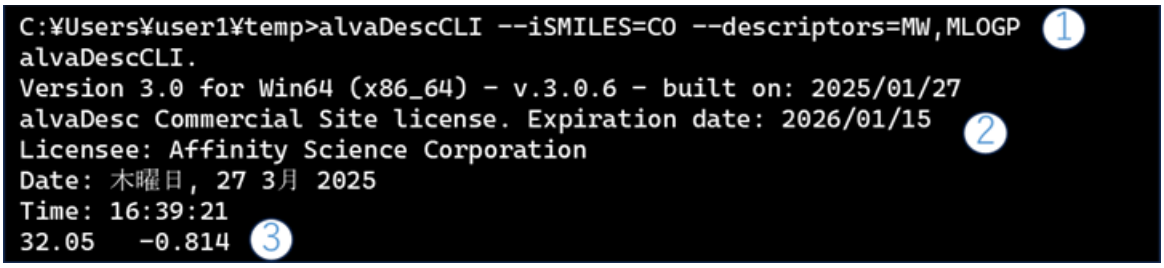
4.1 記述子計算の実行例

4.1.1 1 つの SMILES 文字列を入力し、指定した記述子を計算

1 つの化合物を SMILES で入力し、分子量（MW）と MlogP を計算し、標準出力でアウトプット

入力：

① 引数[--iSMILES]で一酸化炭素（CO）を SMILES 形式で入力、[--descriptors=MW,MLOGP]で計算させる分子記述子（MW と MlogP）を指定。



出力：

② アプリ情報

③ MW と MlogP の計算結果

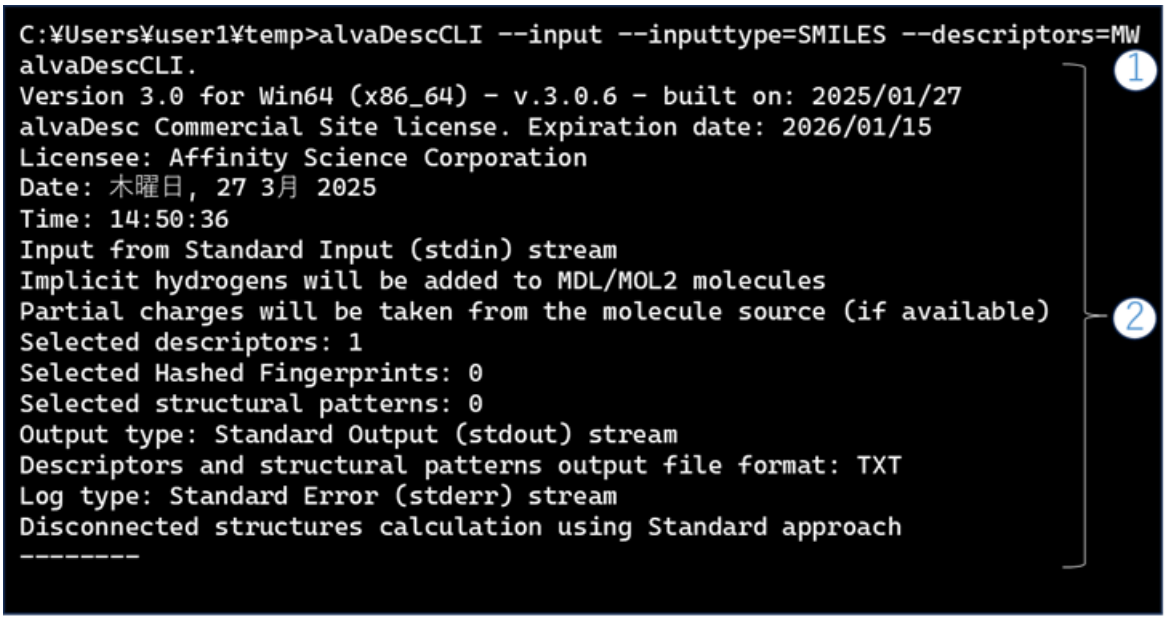
4.1.2 標準入出力を使用し、指定した記述子を計算

化合物を SMILES で標準入力し、分子量（MW）を計算し標準出力でアウトプット

入力 1：

① 引数[--input]で標準入力を指定、[--inputtype=SMILES]で化合物を SMILES 形式で入力することを明示、[--descriptors=MW]で計算させる分子記述子（MW）を指定。

② アプリ情報と計算条件（デフォルトの内容）



入力 2：

③　標準入力によりキーボードから SMILES で入力した化合物（エタノールとエタン）、入力打切りで ctrl+Z+Enter

出力：

④　MW の計算結果

⑤　計算ログ

【Note】Windows/Linux/macOS 環境での標準入力に於ける入力打切り

画面上の標準入力では入力される分子の数が予め決められていないため、1 つの分子構造を入力し Enter を押すと次の分子構造の入力となります。そのため、計算を開始するためには分子の入力が終わったことを明示する必要があります。

Windows 環境（コマンドプロンプト、Windows Power Shell）では、ctrl+Z を入力した後で Enter を押します。画面上では入力した分子構造の次に^Z が表示されます。

Linux 環境の場合、ターミナルから ctrl+D を入力すると計算が始まり、画面上には入力した分子構造のみが表示されます。

macOS 環境では、Linux 環境と同様、ctrl+D を入力し、Enter を押しますが、画面上には入力した分子構造の次に^D が表示されます。

上記のように入力の打切りを明示的に行わず、入力終了時に Enter を 2 回押すと計算が始まりますが、その場合、入力した分子数が 1 つ余計にカウントされてしまうことに留意が必要です。

4.1.3 標準入力を使用し、ファイルに出力

化合物を SMILES で標準入力し、全ての記述子を計算しファイルにアウトプット

入力：

① 引数[--input]で標準入力を指定、[--inputtype=SMILES]で化合物を SMILES 形式で入力することを明示、[--descriptors=ALL2D]で 3D 以外の全ての分子記述子を指定、[--output=output.txt]で出力先のファイル名を指定、[--labels]で出力に記述子名を含めることを指定。

②　【表示省略】アプリ情報と計算条件

```
C:¥Users¥user1¥temp>alvaDescCLI --input --inputtype=SMILES --
descriptors=ALL2D --output=output.txt --labels
```
①

入力 2：

③　標準入力によりキーボードから SMILES で入力した化合物（エタノールとエタン）、入力打切りで ctrl+Z+Enter

```
CCO
CC
^Z
```
③

出力：

④　【表示なし】記述子計算結果はファイルに出力されるため、画面上に表示なし

⑤　計算ログ

```
--------
Processed molecules: 2
Unrecognized molecules: 0
Rejected molecules: 0
Warned molecules: 2
Elapsed time: 00:00:22.641
--------
```
⑤

※出力ファイル output.txt を Excel 形式で表示した一部

No.	NAME	MW	AMW	Sv	Se	Sp	Si	Mv	Me	Mp	Mi
1	Molecule1	46.08	5.12	4.2952	8.9781	4.7387	10.455	0.477244	0.997567	0.526522	1.161667
2	Molecule2	30.08	3.76	3.5804	7.6508	4.2842	9.2456	0.44755	0.95635	0.535525	1.1557

4.1.4 標準入出力をファイルにリダイレクト

化合物を標準入力から SMILES 形式でファイル入力し、全ての記述子を計算し標準出力からファイルにアウトプット

入力：引数[--input]で標準入力を指定、[--inputtype=SMILES]で化合物を SMILES 形式で入力することを明示、[--descriptors= MW,nAA,ChiralCenter,MLOGP]で 4 つの分子記述子を指定、[--labels]で出力に記述子名を含めることを指定、[< sample.smi]で標準入力から sample.smi という名前の SMILES ファイルを入力指定、[> output.txt] で標準出力から output.txt というテキストファイルに出力指定、[2> log.txt] で標準エラーから計算ログとエラーを log.txt というテキストファイルに出力指定。

（※以降、入力内容のみを表示）

```
alvaDescCLI --input --inputtype=SMILES --
descriptors=MW,nAA,ChiralCenter,MLOGP --labels < sample.smi > output.txt 2>
log.txt
```

※パイプを利用して標準入力からファイルを入力することもできます。

（Linux/macOS の場合は[type]が[cat]となります。）

```
type sample.smi | alvaDescCLI --input --inputtype=SMILES --
descriptors=MW,nAA,ChiralCenter,MLOGP --labels
```

4.1.5 Input/output オプションを使用したファイル入出力

・化合物を SMILES 形式でファイルから入力し、3D 以外の全ての記述子を計算しファイルにアウトプット

入力：引数[--input=sample.smi]で化合物を sample.smi という名前の SMILES 形式のファイルで入力することを指定、[--output=output.txt]で output.txt という名前のテキストファイルに出力することを指定、[--descriptors=ALL2D]で 3D 以外の全ての記述子を指定、[--labels]で出力に記述子名を含めることを指定。

```
alvaDescCLI --input=sample.smi --outpupt=output.txt --descriptors=ALL2D --
labels
```

・化合物を MDL 形式でファイルから入力し、全ての記述子を計算しファイルにアウトプット

入力：引数[--input=sample.sdf]で化合物を sample.sdf という名前のファイルで入力することを指定、[inputtype=MDL]で入力が MDL 形式であることを指定、[--output=output.sdf]で output.sdf という名前のファイルに出力することを指定、[outputtype=MDL]で出力が MDL 形式であることを指定、[--descriptors=ALL]で全ての記述子を指定、[--labels]で出力に記述子名を含めることを指定。

```
alvaDescCLI --input=sample.sdf inputtype=MDL --outpupt=output.sdf -
outputtype=MDL --descriptors=ALL --labels
```

※input/output にサポートされる形式のファイルが指定されている場合、拡張子でファイル形式が判定されるため、inputtype/outputtype は省略可能です。

```
alvaDescCLI --input=sample.sdf --outpupt=output.sdf --descriptors=ALL --
labels
```

4.2 FP 計算の実行例

4.2.1 標準入出力を使用し、MACCS166 FP を計算

1 つの化合物を SMILES 形式で標準入力からパイプで入力し、MACCS166 FP を計算し、標準出力でアウトプット

入力：[echo CCO]で標準入力から入力する構造を指定し、パイプ[｜]で alvaDescCLI に繋げ、引数[--input]で標準入力を[inputtype=SMILES]で入力形式を指定、[--maccsfp]で計算させる FP を指定。

```
echo CCO | alvaDescCLI --input --inputtype=SMILES --maccsfp
```

出力：

```
--------

Molecule1
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000010000000000000000000000000010000100000000000000000000000010000000000000
0010101001000100

--------
```

4.2.2 ファイル入出力を使用し、ECFP を計算

化合物を MDL 形式のファイルから入力し、ECFP を計算し、ファイルにアウトプット

入力:引数[--input=molecules.sdf]と[inputtype=MDL]で molecules.sdf という名前の MDL 形式ファイルを入力として指定、[--output=molecules-ecfp.txt]で FP の出力先ファイルを指定、[--ecfp]で計算させる FP を指定、[--size=2048]で FP の長さを 2048 ビットに指定。

```
alvaDescCLI --input=molecules.sdf --inputtype=MDL -output=molecules-
ecfp.txt --ecfp --size=2048
```

4.3 使用できる引数

4.3.1 基本の引数

・入出力の指定

引数（Argument）	内容
--script=<スクリプトファイル名> または -s <スクリプトファイル名>	スクリプトファイル名の指定（--verbose と --threads 以外の引数指定不可）
--input[=<入力ファイル名>]	入力の指定： ファイル名を指定しない場合には標準入力（デフォルト）またはファイル名を指定
--inputtype=SMILES|MDL|SYBYL|HYPERCHEM	入力形式の指定： SMILES 形式/MDL 形式/Sybyl 形式/HyperChem 形式 （※--input でファイル名を指定しない場合には不可欠）
--output[=<出力ファイル名>]	出力の指定： ファイル名を指定しない場合には標準出力（デフォルト）またはファイル名を指定
--outputtype=TXT|SMILES|MDL	出力形式の指定：テキストファイル、SMILES 形式 MOL Structure Data File 形式

※--output/--outputtype は --input と共に使用しない場合動作不可

・計算させる特徴量の指定（アウトプットオプション）

引数（Argument）	内容
--descriptors=<記述子名>	記述子の計算： 計算させる記述子はカンマ区切りで指定、全記述子は=ALL、3D 記述子以外の全ては=ALL2D
--ecfp	ECFP（extended connectivity fingerprint）の計算

--pfp	PFP（path fingerprint）の計算
--ecfpv3	ECFPV3（extended connectivity fingerprint v3）の計算
--maccsfp	MACCS166 フィンガープリントの計算

※アウトプットオプションはいずれか 1 つのみ指定可能

4.3.2 オプションの引数

引数（Argument）	内容
--labels	記述子と分子のラベルを記載
--size=<SIZE>	ハッシュ化 FP のサイズを指定（デフォルト：1024 ビット）
--min=<MIN LENGTH>	ハッシュ化 FP の最小フラグメント長を指定（デフォルト：0）
--max=<MAX LENGTH>	ハッシュ化 FP の最大フラグメント長を指定（デフォルト：2）
--count=<TRUE|FALSE>	ハッシュ化 FP 計算でフラグメント数をカウント（デフォルト：TRUE）
--bits=<BITS PER PATTERN>	ハッシュ化 FP 計算でパターン毎のビット数を指定（デフォルト：2）
--fpoptions=<OPTIONS>	ハッシュ化 FP 計算での原子オプションを指定（デフォルト：Atom type, Aromaticity, Charge, Connectivity (total), Bond order）
--threads=<NUMBER OF THREADS>	計算に使用するスレッド数を指定（デフォルト：最大使用可能 CPU 数）
--verbose	詳細表示オプションの有効化
-h 若しくは --help	コマンドヘルプの表示

※1 つの分子のみに対して計算させる場合、[--threads=1] の使用を推奨

4.3.3 その他の引数

引数（Argument）	内容
--descriptorlist	全記述子のリストを出力（番号/名称/詳細/ブロック No./サブブロック No./数値のタイプ/3D か否か）
--cite	alvaDesc を論文等で引用する際の表記を出力
--update	アップデート版があるかどうかをチェック
--update=<UPDATE FILE>	アップデート版がある場合にアップデートファイル名を指定してダウンロード
--license	現在使用しているライセンスファイルとその内容を出力
--license=<LICENSE FILE>	指定したライセンスファイルの内容を出力
--request-license=<REQUEST FILE>	ライセンスファイルのリクエストをファイルへ出力
--import-license=<LICENSE FILE>	指定したライセンスファイルのインポート
--deactivate-license=<DEACTIVATION FILE>	ライセンスファイルの無効化

※ライセンスファイルの無効化等、ライセンスファイルに関してはユーザーマニュアル 1.2 License を参照

5. スクリプトファイルの使用

ここからは、スクリプトファイルを用いて alvaDescCLI を使用する方法について説明します。

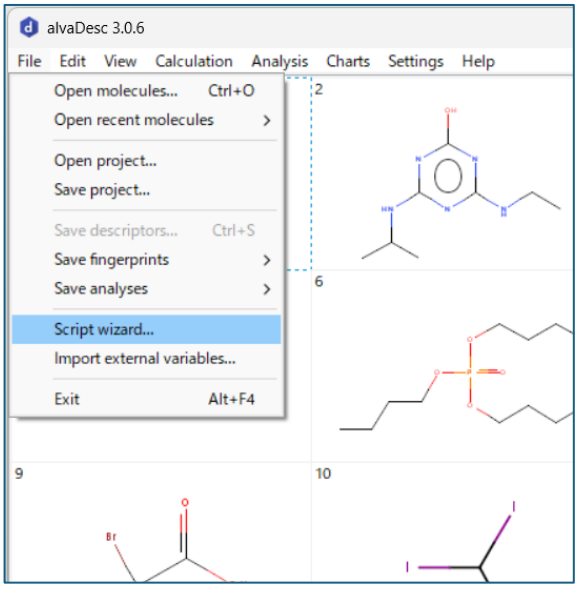
スクリプトファイルは alvaDescCLI の全てのオプションが含まれた、ファイル拡張子が .adscr というファイルです。XML 形式で記述されていますので、その記法と内容について理解していればテキストエディタなどで作成することも可能です。ただし、ゼロからスクリプトファイルを書くより、alvaDesc の GUI 画面からスクリプト生成のウィザードに従って進めていく方が簡単ですので、まず、ウィザードによるスクリプトファイルの作成について説明します。その後で、スクリプトファイルの詳細内容について見ていくこととします。

5.1 ウィザードによるスクリプトファイル作成

5.1.1 ウィザードの操作

・スクリプトウィザードの開始

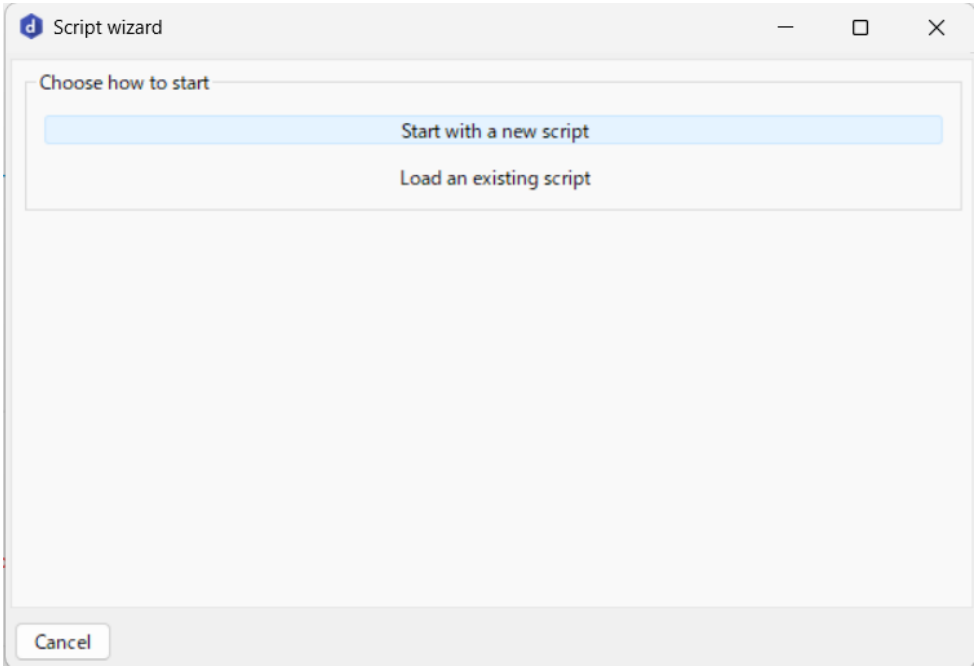
alvaDesc の GUI アプリを立上げ、[File]メニューから[Script wizard...]を選択します。この例では、既に分子セデータがロードされた状態になっていますが、スクリプトファイルを作成するだけの目的であれば、特に分子データをロードする必要はありません。



・新規作成か既存スクリプトファイルの流用かの選択

「Script wizard」というウィンドウが立ち上がります。新しいスクリプトを作成するか（Start with a new script）、既存のスクリプトファイルをロードするか（Load an existing script）のメニューがあります。ここでは、上の新しいスクリプトを作成するメニューをクリックして選択します。

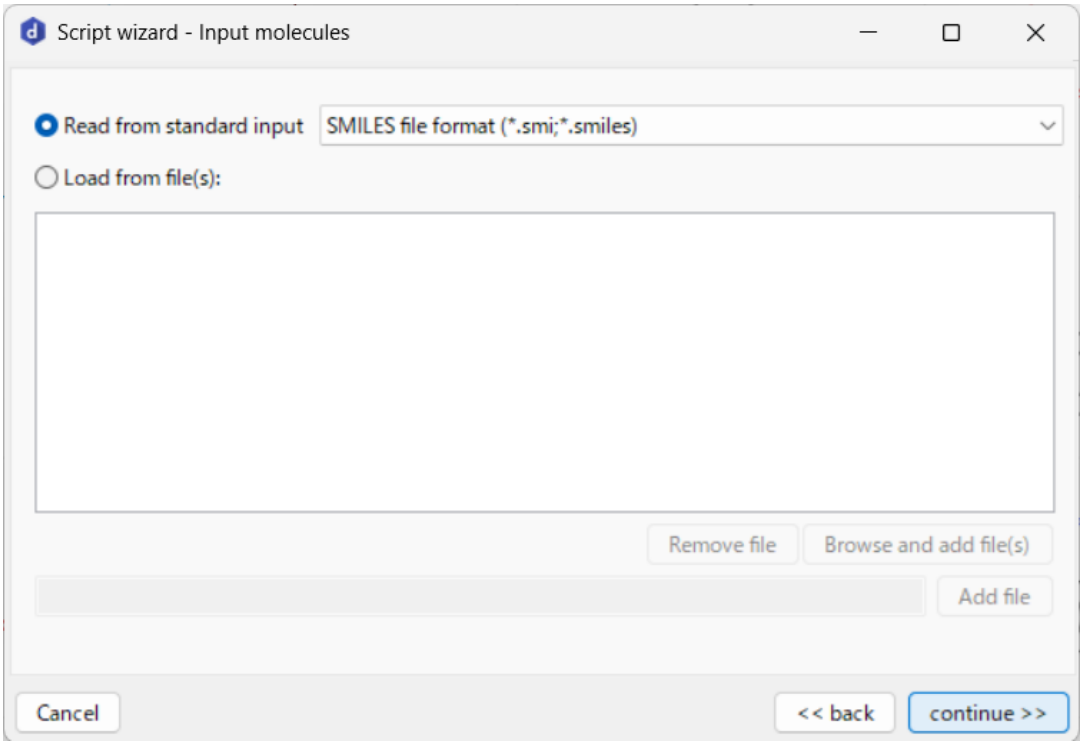
尚、既存のスクリプトファイルを選択し流用する場合、ファイルエクスプローラメニューからロードするスクリプトファイルを選択した後の画面遷移は新規作成と同じになり、保存していたスクリプトの内容を必要に応じて書き換えることになります。



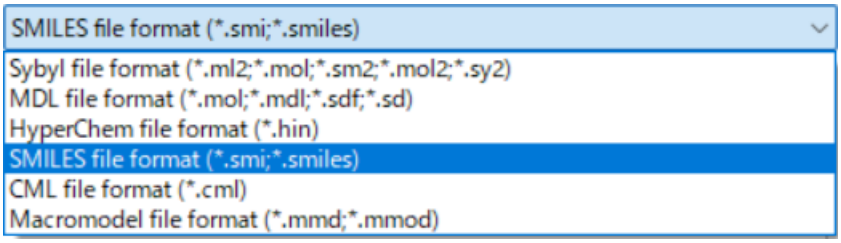
・入力する分子データの指定

「Input molecules」という画面に遷移します。ここで、標準入力にするかファイル入力にするかを二択で選び、[continue >>]で次に進みます。

以下の例では標準入力を選択したものとして進めますが、ファイル入力（Load from file(s)）を選択した場合、白枠の下にある[Remove file][Brows and add file(s)][Add file]のメニューがアクティブになり、ロードするファイルを指定できるようになります。予めファイルのパスがわかっている場合には[Add file]の左にあるボックスにパスを入れ、[Add file]を押下することによりファイルを指定できます。ファイルエクスプローラからファイルを指定したい場合、[Brows and add file(s)]を押下し、エクスプローラでファイルを検索し指定します。尚、入力ファイルは複数指定でき、また、形式が異なるファイルの混在も可能です。



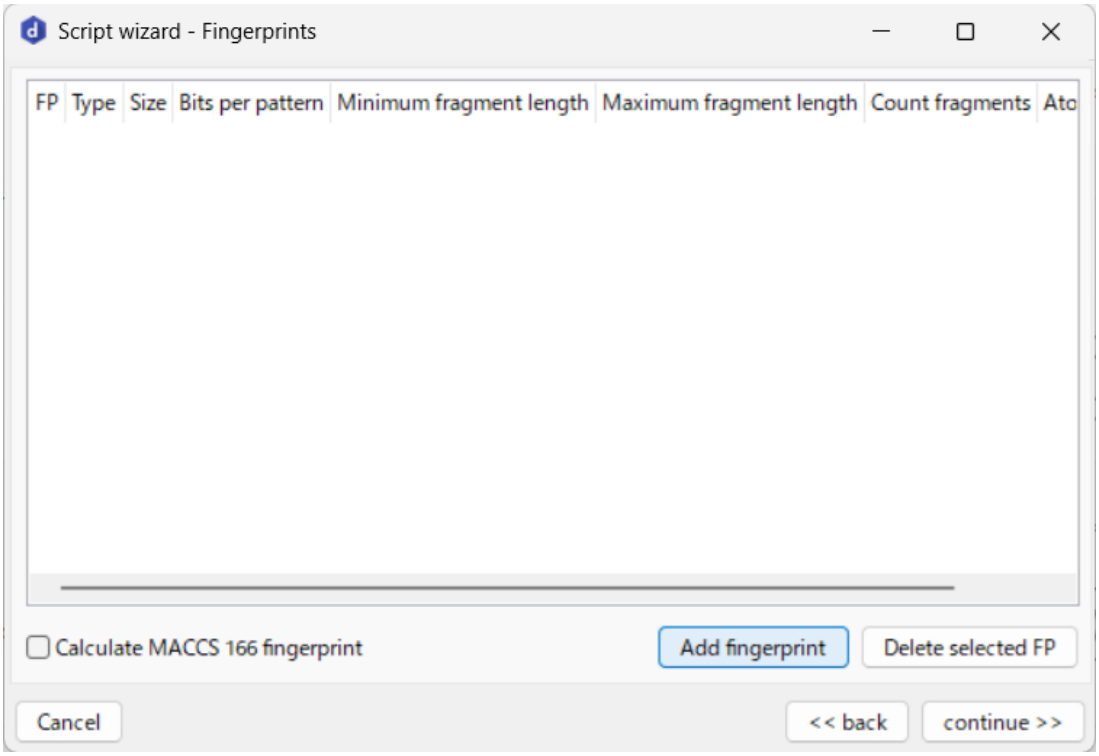
標準入力を選択する場合、プルダウンメニューから入力ファイルの形式を指定する必要があります。



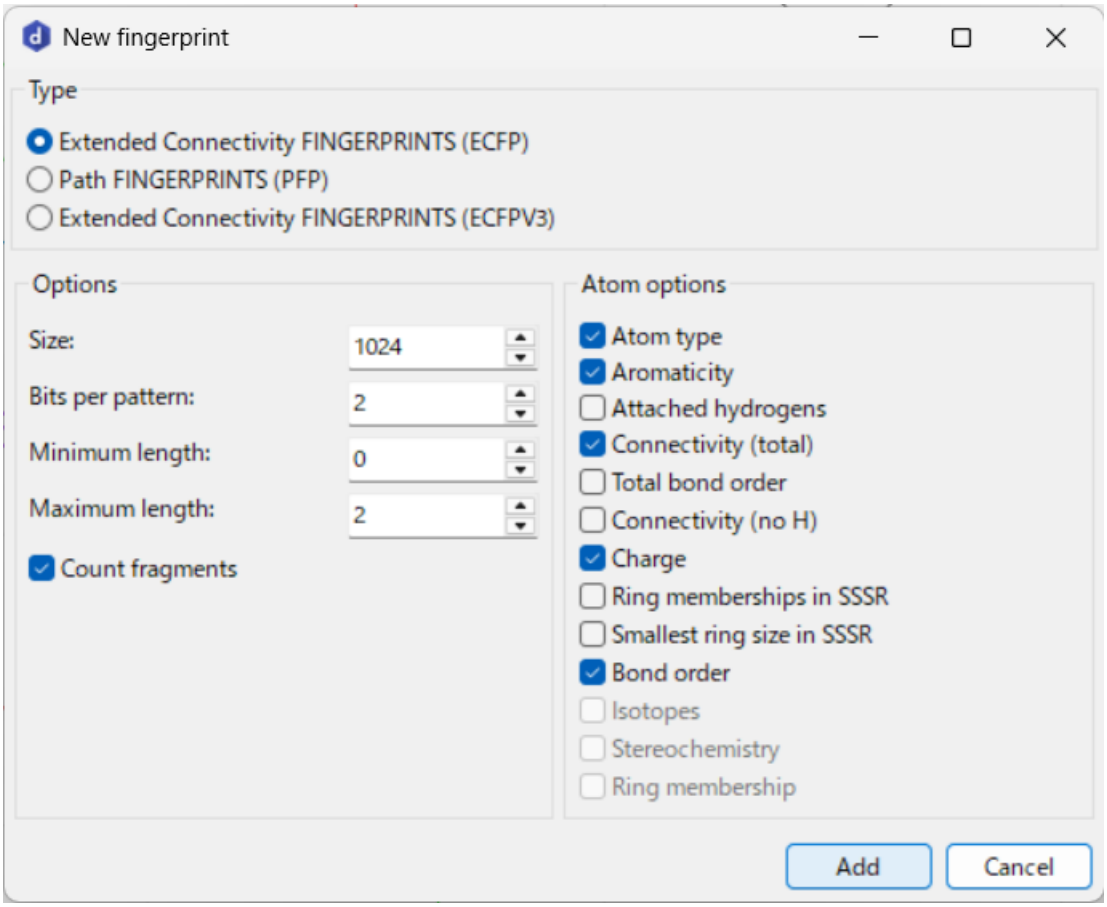
・FP 計算の設定

「Fingerprints」という画面に遷移しますので、ここで計算させる FP の種類や計算オプションを指定します。

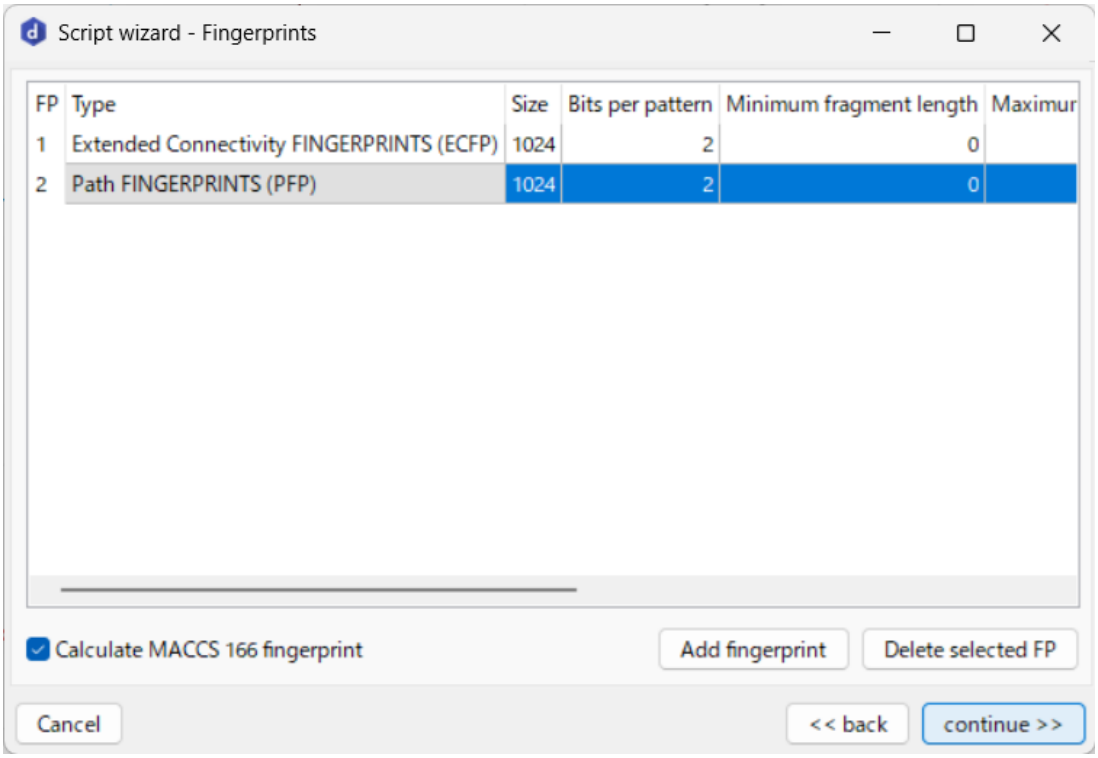
まず、ハッシュ化 FP を指定するため[Add fingerprint]のボタンを押下します。



「New fingerprint」という画面が現れますので、ECFP/PFP/ECFPV3 の中から計算させたい FP を選択し、必要に応じて下にある計算オプションを選択します。デフォルト条件で良ければオプションはそのままにしておきます。[Add]を押すと選択した FP が追加されます。オプションの詳細については、「alvaDesc 3.0 スタートアップガイド」やユーザーマニュアルをご参照下さい。



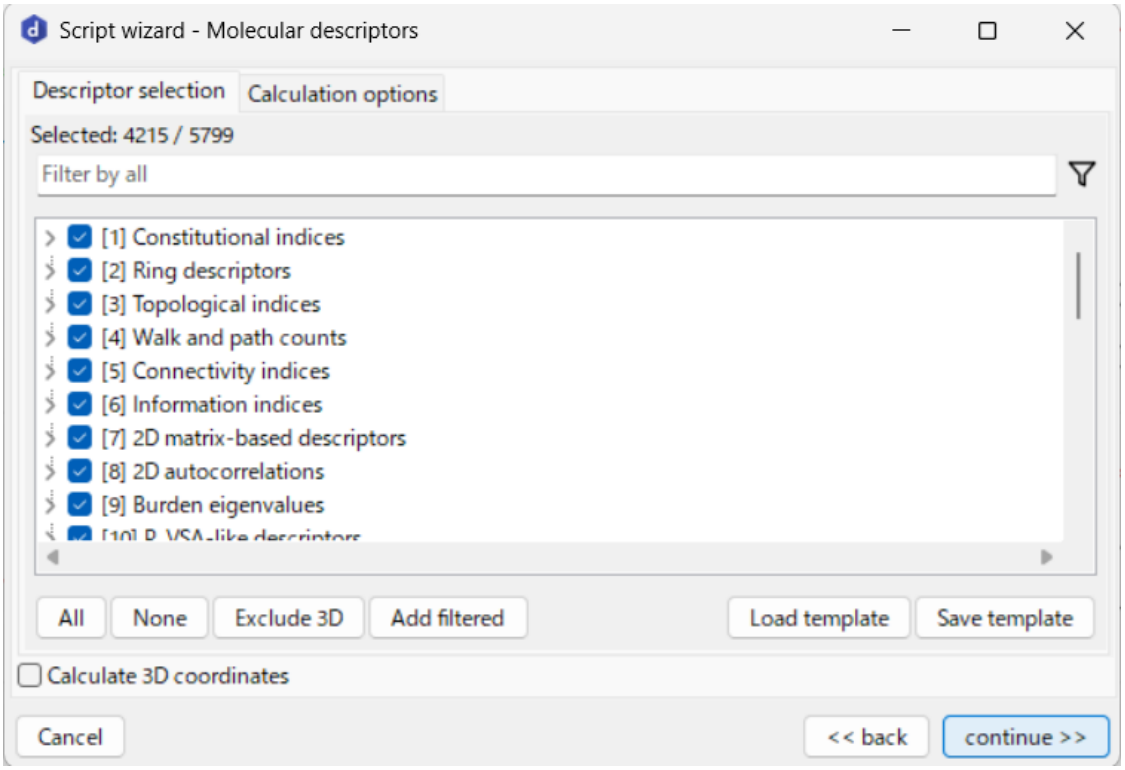
ハッシュ化 FP はいくつでも追加できます。MACCS 166 FP の計算は、左下のチェックボックスにチェックを入れるだけで設定できます。[continue > >]で次に進みます。

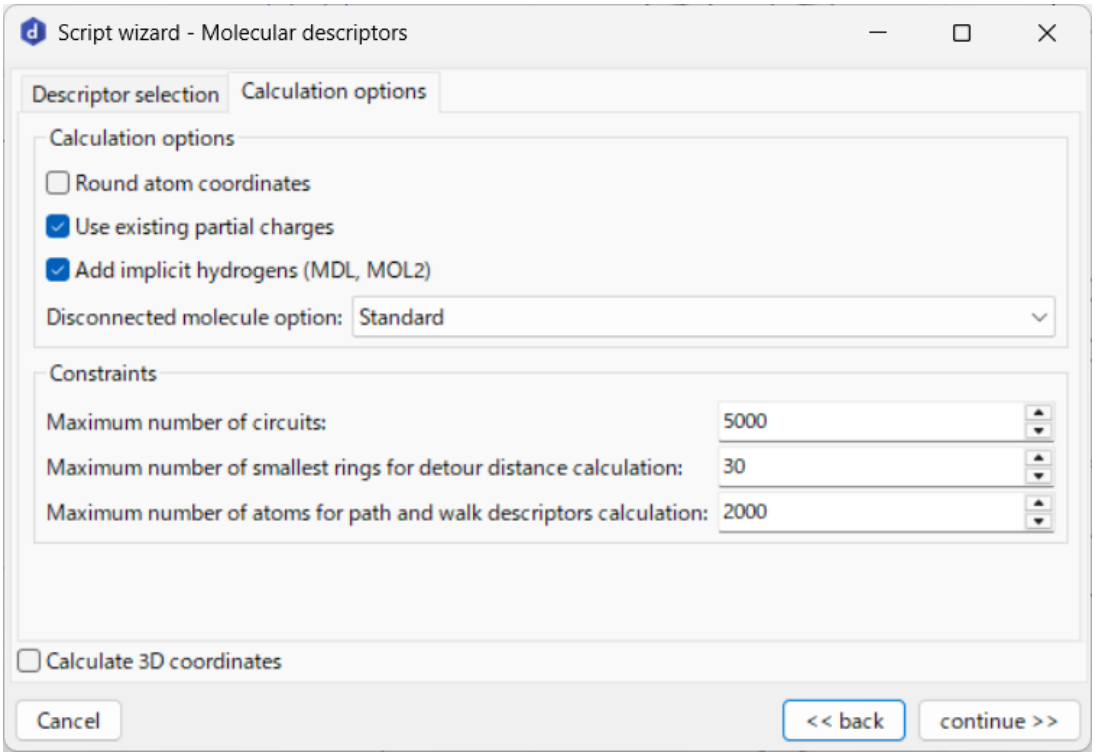


・記述子計算の設定

「Molecular descriptors」という画面に遷移します。タブが 2 つありますが、このうち「Descriptor selection」タブで計算させたい記述子を選択、「Calculation options」タブで計算時のオプションを選択し、[continue > >]で次に進みます。

記述子選択の方法と計算オプションの詳細については、「alvaDesc 3.0 スタートアップガイド」やユーザーマニュアルをご参照下さい。



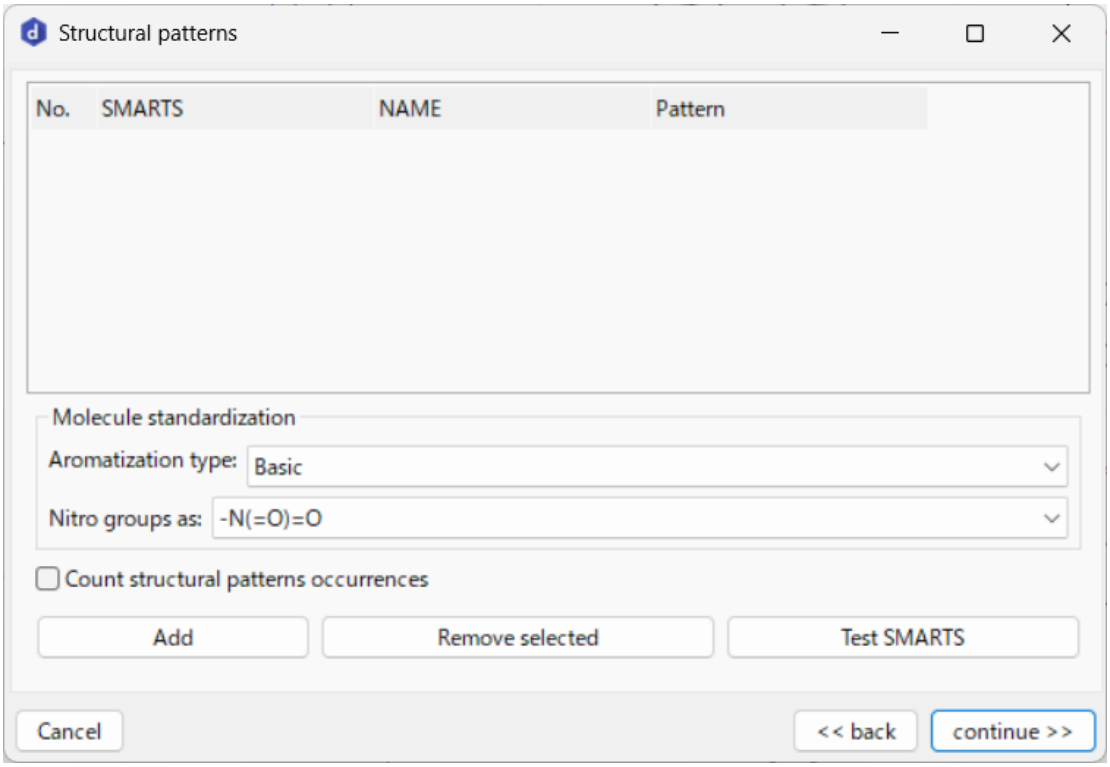


・構造パターン計算

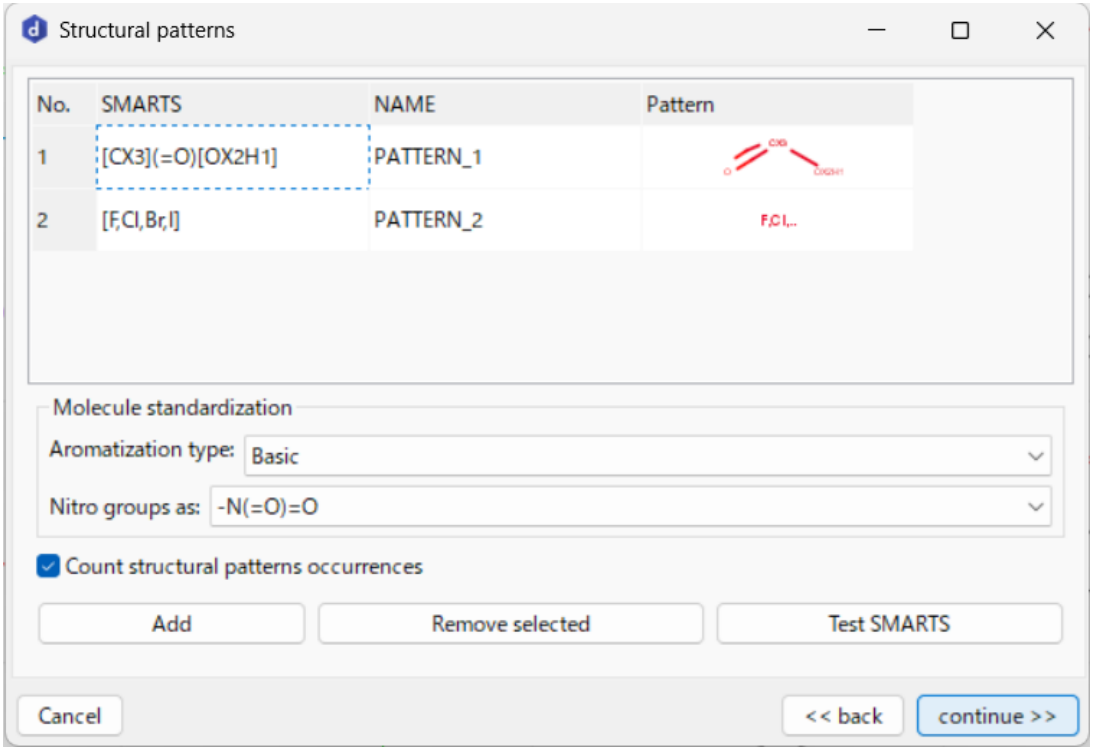
「Structural patterns」画面に遷移します。

指定した部分構造パターンの同定を行う場合、[Add]ボタンを押下し、入力ボックスに SMARTS 表記の文字列を入れていきます。パターンはいくつでも入力が可能で、同定されたパターンが存在する数をカウントする場合は左下の[Count structural patterns occurrences]にチェックを入れます。

構造パターン計算の詳細については、「alvaDesc 3.0 スタートアップガイド」やユーザーマニュアルをご参照下さい。

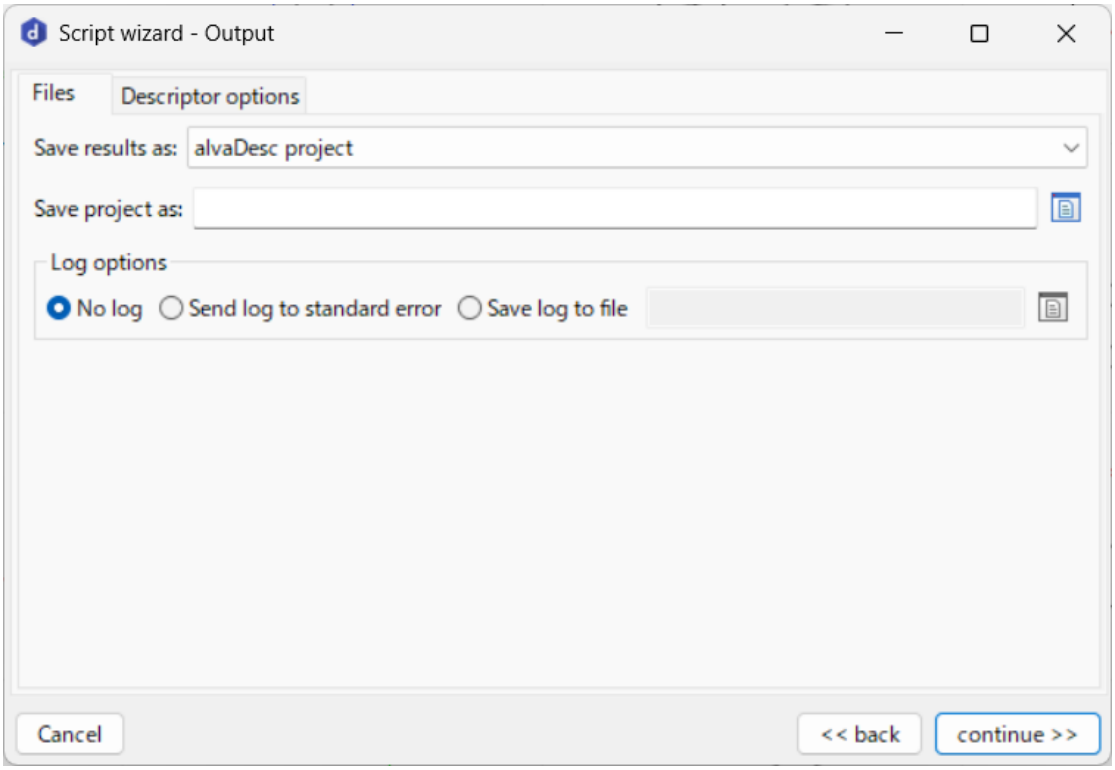


ここでは、同定する部分構造として、カルボン酸（ [CX3](=O)[OX2H1] ）とハロゲン（ [F,Cl,Br,I] ）を SMARTS で入力しました。



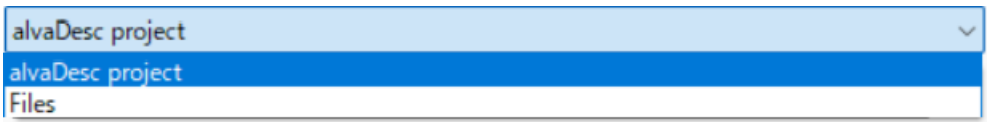
・出力の設定

「Output」画面に遷移します。タブが 2 つありますが、このうち「Files」タブで出力する形式を選択しファイル名を指定、「Descriptor options」タブで出力時のオプションを選択し、[continue > >]で次に進みます。



「Files」タブの一番上にある「Save results as:」はプルダウンメニューになっており、計算結果を alvaDesc のプロジェクトファイルとしてまとめて出力するか(alvaDesc project)、FP と記述子・構造パターンをそれぞれ別のファイルに出力するか(Files)

を選択できます。alvaDesc project を選択した場合は、その下にある「Save project as」のボックスにファイルパスを入力するか、ボックスの右側にあるファイルのボタンをクリックして出てくる「Save project file」ウィンドウで保存するファイル名を指定します。

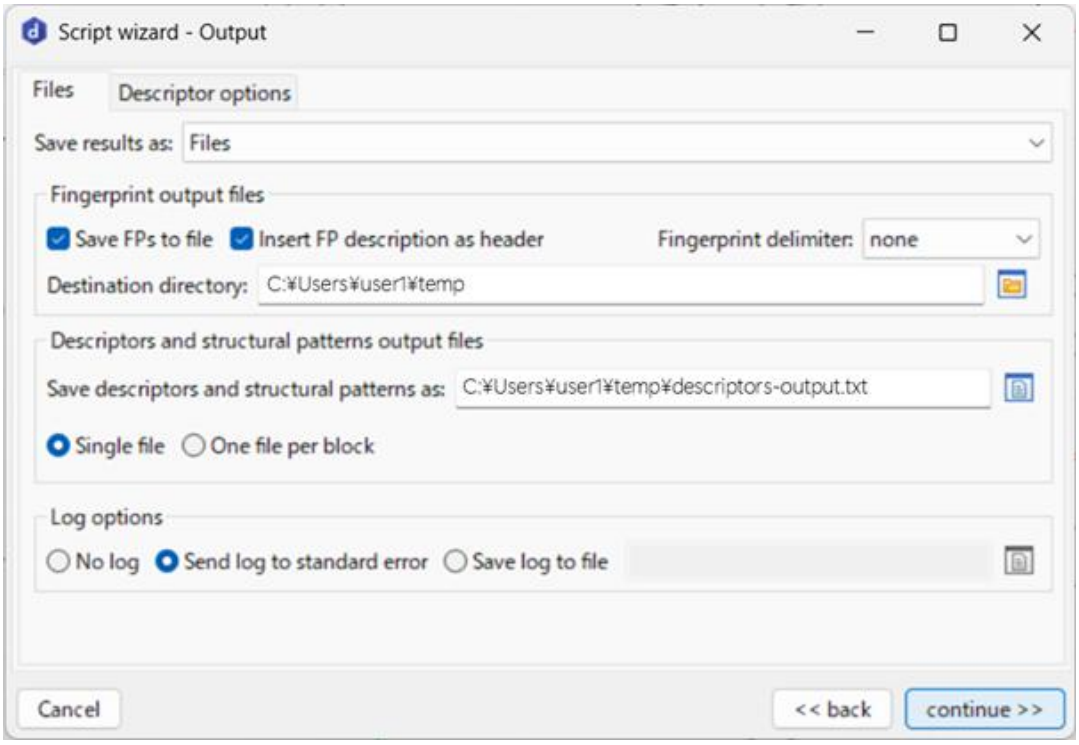


Files を選択した場合には画面構成が変わり、「Fingerprint output files」と「Descriptors and structural patterns output files」の設定が出てきます。

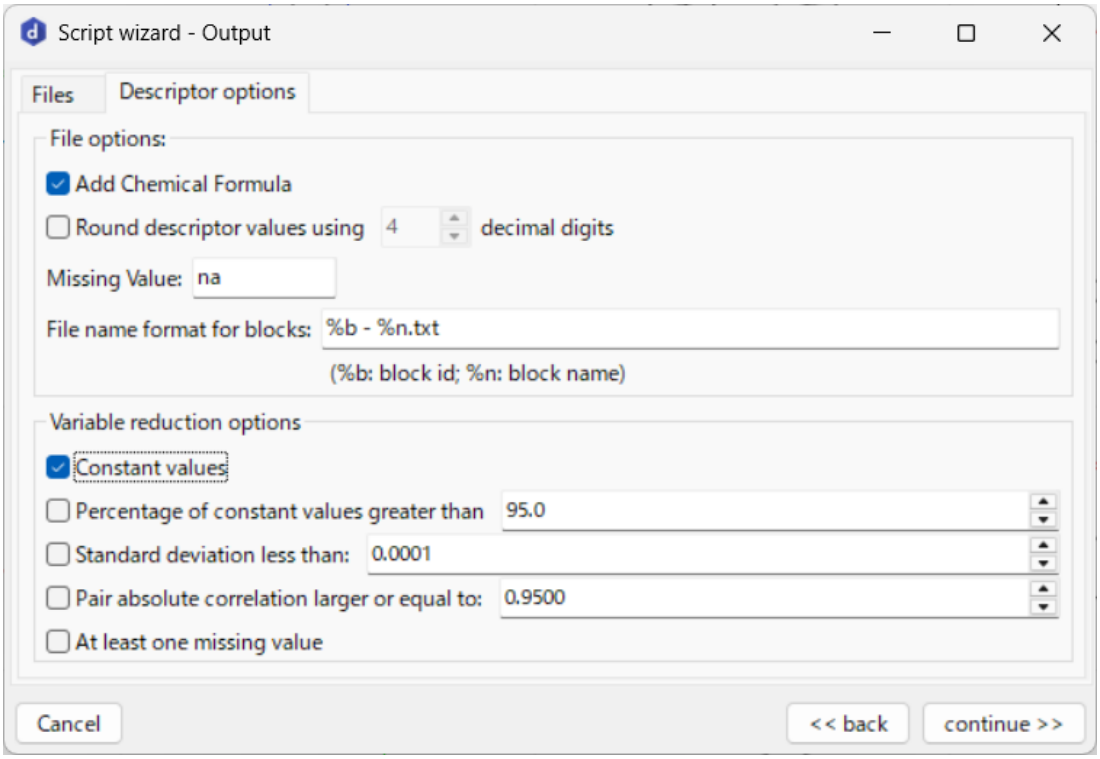
FP については、「Save FPs to file」のチェックボックスをクリックし選択すると、「Insert FP description as header」と「Destination directory:」がアクティブになります。「Insert FP description as header」にチェックを入れると、出力ファイルのヘッダーに FP の詳細情報が書かれるので便利ですが、チェックを入れるかどうかは任意です。また、FP のファイル出力ではファイル名が自動的に付与されますので、指定はファイル名ではなく、出力先のディレクトリとなります。

記述子と構造パターンの出力については、結果の全部を 1 つのファイルに出力するか（Single file）、ブロック毎に分けたファイルに出力するか（One file per block）をラジオボタンで選択し、Single file を選択した場合は、右側にある青いファイルのボタンをクリックして出てくる「Save as...」ウィンドウで保存するファイル名を指定します。One file per block を選択した場合、右側のファイルボタンが赤色に変わりますので、FP と同様に出力先のディレクトリを指定します。

File タブ画面の一番下には「Log options」があり、計算ログやエラーメッセージをどのように出力するかをラジオボタンで指定します。「No log」はログの出力なし、「Send log to standard error」はログを標準エラーに出力、「Save log to file」はログをファイル出力する設定となります。ファイル出力を選択した場合は、その右にある黒いファイルのボタンをクリックし、ファイル名を指定します。「Log options」は出力としてプロジェクトファイルを選択した場合も同じ設定内容となります。



「Output」画面の「Descriptor options」タブには、「File options」と「Variable reduction options」の設定があります。File options は出力に化学式を追記するか、記述子値の小数点以下の桁数設定、欠損値（missing value）をどのように表示させるかの設定などがあります。Variable reduction options は計算された多数の記述子から、解析用途に適さない記述子を削減するための設定となります。どの項目にもチェックを入れなければ、計算された全ての記述子が出力されます。

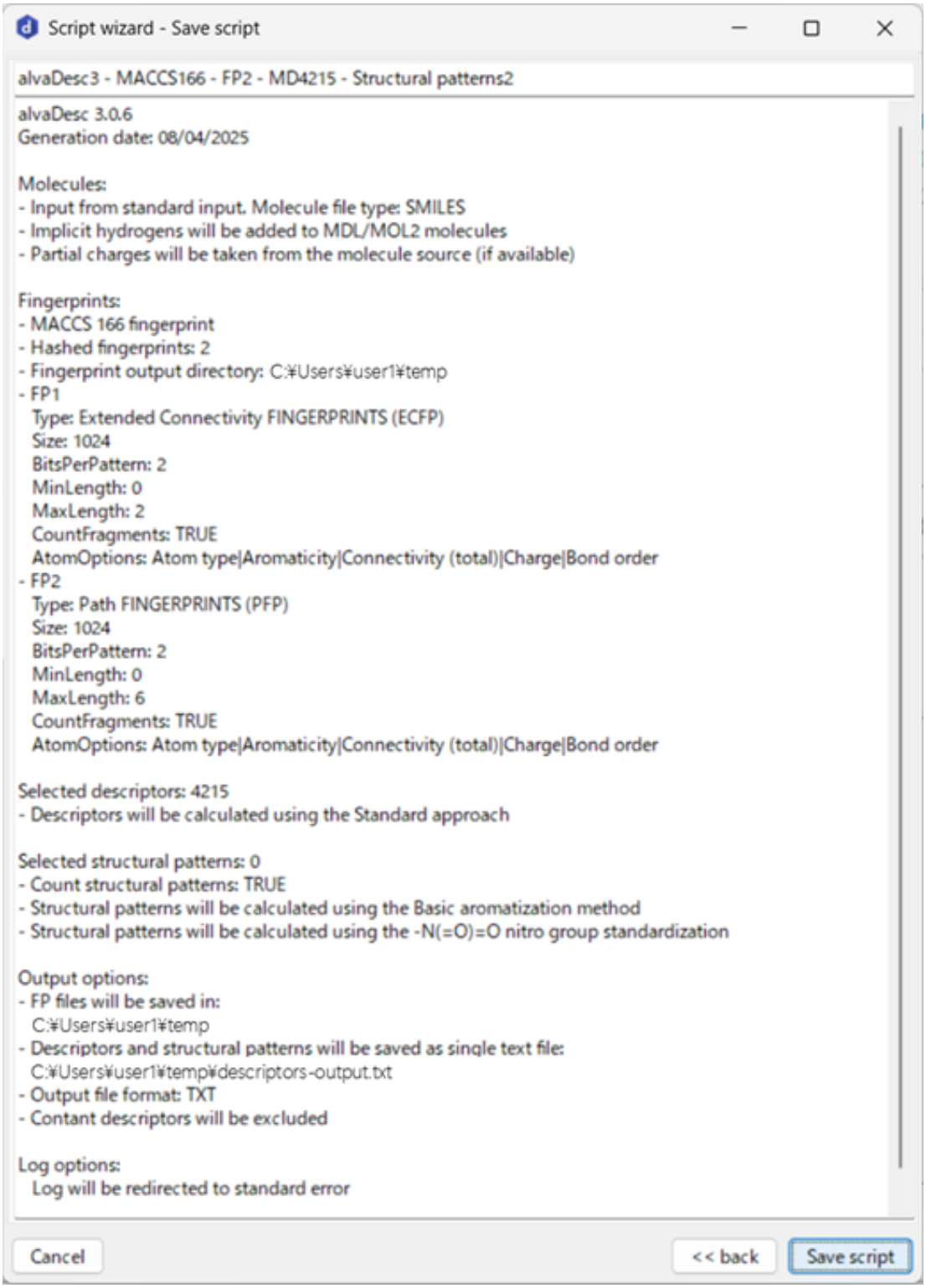


尚、変数選択の詳細については、「alvaDesc 3.0 スタートアップガイド」をご参照下さい。

・生成されたスクリプトの確認と保存

「Save script」画面に遷移します。この画面で、今までスクリプトウィザードで設定して来た内容を確認することができます。一番上には、計算させる項目の概要が「alvaDesc3 - MACCS166 - FP2 - MD4215 - Structural patterns2」のように表示されます。この例では、alvaDesc 3.0 を使用し、MACCS166 FP、2 種類のハッシュ化 FP、4215 種の記述子ならびに、2 つの構造パターンについて計算を行うという内容となります。その下に詳細なスクリプトが表示されていますので、修正したい場合には [< < back] ボタンをクリックすることにより、前の画面に戻っていくことができます。

内容が良ければ、[Save script]ボタンをクリックし、「Save script file」ウィンドウでファイル名を指定し保存します。



5.1.2 スクリプトファイルを用いたコマンド実行方法

作成し保存したスクリプトファイルを実行する際は、alvaDescCLI の引数[--script=]を使います。

入力：--script=tutorial-script.adscr で実行するスクリプトファイルを指定、--verbose で詳細表示オプションを有効化

```
alvaDescCLI --script=tutorial-script.adscr --verbose
```

尚、--script=<script file>を使用する場合、--verbose と—threads 以外の引数を一緒に使用することはできません。

5.2 スクリプトファイルの詳細

スクリプトファイルは alvaDescCLI を実行する際に使用可能な全てのオプションを含んだ XML 形式ファイルで、ファイル拡張子は .adscr となっています。前述のスクリプトウィザードで最後に表示されるものはウィザードで設定した内容だけでしたが、.adscr ファイルには全てのオプションが記載されています。従って、スクリプトファイルの中身はとても長くなるため、ここには実際の例の全てを掲載しておりません。スクリプトウィザードで作成したファイルをメモ帳などのテキストエディタで開いて参照しながら以下の詳細説明をご覧になることをお勧めします。

5.2.1 スクリプトファイルの構造

.adscr ファイルのルートノードは ALVADESC で、その下に以下の 6 つのサブノードがあります。

- OPTIONS
- MOLFILES
- FINGERPRINT
- DESCRIPTORS
- STRUCTURAL_PATTERNS
- OUTPUT

全体の構造を簡単に表すと次のようになります。

```
<?xml version="1.0" encoding="utf-8"?>
<ALVADESC version="3.0.6" description="alvaDesc3 - MACCS166 - FP2 - MD4215 - Structural
patterns2" generation_date="08/04/2025">
  <OPTIONS>
    要素"OPTIONS"の内容設定
  </OPTIONS>
  <MOLFILES>
    要素"MOLFILES"の内容設定
  </MOLFILES>
  <FINGERPRINT>
    要素"FINGERPRINT"の内容設定
  </FINGERPRINT>
  <DESCRIPTORS>
    要素"DESCRIPTORS"の内容設定
  </DESCRIPTORS>
  <STRUCTURAL_PATTERNS>
    要素"STRUCTURAL_PATTERNS"の内容設定
  </STRUCTURAL_PATTERNS>
  <OUTPUT>
    要素"OUTPUT"の内容設定
  </OUTPUT>
</ALVADESC>
```

最初の行で XML 宣言をし、2 行目でルートノード ALVADESC とその概要を表わしています。<OPTIONS>以下は 6 つのサブノード毎にその内容を設定します。

以下、サブノード毎に、含まれるタグとその値、内容について見ていきます。

5.2.2 OPTIONS サブノード

XMLタグ	値	内容
AddImplicitHydrogens	true / false	true: MDL/MOL2形式で入力された分子構造に暗示的水素を付加
UseExistingPartialCharges	true / false	true: 原子の部分電荷情報が入力ファイルに存在すればロードし使用、無ければalvaDescCLIが原子の部分電荷を計算
DisconnectedCalculationOption	0~5の整数	【非連結構造の分子構造に対する計算アプローチの選択】0: Standard / 1: Maximum descriptor value / 2: Minimum descriptor value / 3: Average descriptor value / 4: Sum of descriptor values / 5: Retain the biggest fragment
Calculate3DCoordinates	true / false	true: 3D座標の計算を実行
AromatizationType	0 /1	【芳香族タイプの指定】 0: Basic / 1: General
NitroGroupStd	0 /1	【ニトロ基の標準化】0: -N(=O)=O / 1: -[N+](=O)[O-]
MaxCircuits	正の整数	【縮合環のような複雑な構造に対する計算上の制限付与】環の最大数
MaxSRDetour	正の整数	【縮合環のような複雑な構造に対する計算上の制限付与】最短環経路の計算に使用される環の最大数
MaxAtomWalkPath	正の整数	【縮合環のような複雑な構造に対する計算上の制限付与】パスとウォーク記述子の計算に使用される原子の最大数
RoundCoordinates	true / false	true: 原子の座標を小数点以下4桁で丸め
Missing_String	文字列	【欠損値に対する表示設定】表示する文字列（デフォルトでは"na"）
SaveOnlyData	true / false	true: ラベルを付与せず生データのみを保存
SaveLabelsOnSeparateFile	true / false	true: ラベルと分子名を2つの別々のファイルに保存（SaveFileタグが保存オプションの中で唯一"true"となっており。かつ、SaveTypeタグが"singlefile"となっていることが必要）
SaveFormatBlock	文字列	【記述子をブロック毎に保存する際のファイル名設定】"%b"（ブロック名）と"%n"（ブロックナンバー）を使用し名前を設定（デフォルトでは"%b - %n.txt"）
SaveExcludeMisVal	true / false	【記述子保存時の変数選択設定】true: 欠損値が1つでもある記述子を保存しない
SaveExcludeAllMisVal	true / false	【記述子保存時の変数選択設定】true: 全てが欠損値である記述子を保存しない
SaveExcludeConst	true / false	【記述子保存時の変数選択設定】true: 全てが定数である記述子を保存しない
SaveExcludeNearConstPercentage	true / false	【記述子保存時の変数選択設定】true: SaveNearConstPercentageThresholdタグで指定された割合を超える数が定数である記述子を保存しない
SaveNearConstPercentageThreshold	0〜100の実数	【記述子保存時の変数選択設定】SaveExcludeNearConstPercentageタグに与える割合（%）を指定
SaveExcludeStdDev	true / false	【記述子保存時の変数選択設定】true: SaveStdDevThresholdタグで指定された値を下回る標準偏差の記述子を保存しない
SaveStdDevThreshold	正の実数	【記述子保存時の変数選択設定】SaveExcludeStdDevタグに与える閾値を指定
SaveExcludeCorrelated	true / false	【記述子保存時の変数選択設定】true: ペア相関がSaveCorrThresholdタグで指定された閾値を上回る記述子をV-WSPアルゴリズムにより削減
SaveCorrThreshold	1以下の正の実数	【記述子保存時の変数選択設定】SaveExcludeCorrelatedタグに与える閾値を指定
RoundDescriptorValues	true / false	true: DecimalDigitsタグで指定する小数点以下の桁で記述子値を丸め
DecimalDigits	正の整数	RoundDescriptorValuesタグに与える丸めの桁数を指定
AddFormula	true / false	true: ラベルに化学式を追加

5.2.3 MOLFILES サブノード

XMLタグ	値	内容
molInput	file / stdin	分子データの入力方法を指定
molFile	文字列	molInputがfileの場合に分子データをファイルから入力する場合のファイル名を指定（molFileを繰り返すことにより複数ファイルの指定可）
molInputFormat	SYBYL / MDL / HYPERCHEM / SMILES	分子データのファイルフォーマットを指定

5.2.4 FINGERPRINTS サブノード

XMLタグ	値	内容
MACCS166	true / false	true: MACCS 166 FPを計算
FP		ハッシュ化FPを計算
FPの属性: id	0からの正の整数	ハッシュ化FPのIDナンバーを設定（id=0, 1, ... で複数のFPを設定可能）
SaveFPToFile	true / false	true: 計算したFPをファイルに保存
SaveOnlyFP	true / false	true: 計算したFPにラベルを付加せずFPのみを保存
FPheader	true / false	true: 計算したFPに詳細情報を付加してファイルに保存
SaveFPpath	文字列	FPを保存するフォルダのパスを指定
		※保存されるFPのファイル名は自動設定される

・ハッシュ化 FP の設定（FP id=x に対応）

XMLタグ	値	内容
Type	ECFP / PFP / ECFPV3	ハッシュ化FPのタイプを指定
Size	正の整数	FPの長さを指定（ビット数）
BitsPerPattern	正の整数	1つのフラグメントに対し割り当てるビット数の指定
MinLength	正の整数	検出されるフラグメントの最小サイズを指定
MaxLength	正の整数	検出されるフラグメントの最大サイズを指定
CountFragments	true / false	true: フラグメントの出現回数を考慮
AtomOptions	原子オプションを表わすキーワード	フラグメント検出の際に考慮する原子オプションを指定（Atom type, Aromaticity, Attached hydrogens, Connectivity (total), Total bond order, Connectivity (no H), Charge, Ring memberships in SSSR, Smallest ring size in SSSR, Bond order）

・実際のハッシュ化 FP の設定例

```
<FP id="0">
  <Type value="ECFP"/>
  <Size value="1024"/>
  <BitsPerPattern value="2"/>
  <MinLength value="0"/>
  <MaxLength value="2"/>
  <CountFragments value="true"/>
  <AtomOptions value="Atom type|Aromaticity|Connectivity (total)|Charge|Bond order"/>
</FP>
```

この例では、ハッシュ化 FP のタイプを ECFP、長さを 1024 ビット、1 つのフラグメントに割り当てるビット数を 2、フラグメントの最小サイズを 0、最大サイズを 2、フラグメントの出現回数を考慮し、原子オプションでは Atom type、Aromaticity、Connectivity (total)、Charge、Bond order をオンにする設定となっています。複数の原子オプションを記述するため、パイプ（｜）で区切っています。

5.2.5 DESCRIPTORS サブノード

XMLタグ	値	内容
block		計算する記述子ブロックの指定
blockの属性: id	1～34の整数	計算する記述子ブロックをIDナンバーで指定
blockの属性: SelectAll	true	true: 指定した記述子ブロックに含まれる記述子を全て計算
blockの属性: descriptor name	文字列	（SelectAll="true"を指定しない場合）計算させる記述子名をシンボルで指定

・実際の記述子設定例

```
<block id="1" SelectAll="true"/>
<block id="2" SelectAll="true"/>
<block id="3" SelectAll="true"/>
<block id="21">
  <descriptor name="nCp"/>
  <descriptor name="nCs"/>
  <descriptor name="nCt"/>
</block>
```

この例では、ID が 1 のブロックと 2 のブロック、3 のブロックにある記述子を全て計算し、ID が 21 のブロックでは、nCp と nCs と nCt という記述子を計算させる設定となっています。

5.2.6 STRUCTURAL_PATTERNS サブノード

XMLタグ	値	内容
CountPatterns	true / false	true: 検出された構造パターンの生起数をカウント
pattern		検出する構造パターンの設定
patternの属性: name	文字列	検出する構造パターン名を設定
patternの属性: SMARTS	文字列	検出する構造パターンをSMARTSで指定

・実際の構造パターン設定例

```
<CountPatterns value="false"/>
<pattern name="aromatic_carbon" SMARTS="c"/>
<pattern name="any_nitrogen" SMARTS="[#7]"/>
<pattern name="carboxylic_acid" SMARTS="="[CX3](=O)[OX2H1]"/>
```

この例では、構造パターンの生起数はカウントせず、検出する構造を、芳香族炭素（aromatic_carbon）c、全ての窒素（any_nitrogen）[#7]、カルボン酸（carboxylic_acid）[CX3](=O)[OX2H1]としています。
尚、XML では”&”がエスケープ文字となっているため、SMARTS 表記で”&”を使用する場合は”&”とする必要があることに留意して下さい。

5.2.1 OUTPUT サブノード

XMLタグ	値	内容
SaveStdOut	true / false	true: 結果を標準出力に保存
SaveProject	true / false	true: 結果をalvaDescプロジェクトファイルに保存
SaveProjectFile	文字列	SaveProjectがtrueの場合に保存するプロジェクトファイル名を指定
SaveFile	true / false	true: 結果をファイルに保存
SaveType	singlefile / block	SaveFileがtrueの場合に記述子の保存方式を指定（singlefile: 1つのファイルに保存/block: ブロック毎にファイルを分けて保存）
SaveOutputType	TXT / SMILES / MDL	結果の出力フォーマットを指定
SaveFilePath	文字列	SaveFileがtrueの場合に保存するファイルのパスを指定（SaveType=singlefileの場合: 出力ファイルのパス、SaveType=blockの場合: ファイルを出力するディレクトリのパス）
logMode	none / stderr / file	ログの出力方法の設定（出力しない/標準エラー/ファイル）
logFile	文字列	logModeがfileの場合に保存するログファイル名を指定

6. おわりに

alvaDesc コマンドライン入門をご覧いただき、誠にありがとうございました。

ここでは、alvaDesc をコマンドラインインターフェースから使う方法について、入出力の画面例やオプションの詳細をまとめた表などで説明してきました。オプションの中には更に発展的な的な使い方の為に用意されている機能もありますが、できるだけ分かり易くご紹介いたしました。それらの機能を活用される場合、ぜひ GUI 版を解説した alvaDesc スタートアップガイドも併せてご覧になり、理解を深めて頂きたいと思います。

また、alvaDesc は分子の特徴量を計算するソフトですが、開発元である Alvascience 社のソフトウェアスイートには、alvaDesc で計算された特徴量に基づいて QSAR/QSPR のモデルを自動生成する alvaModel/alvaRunner や、alvaDesc の一部の記述子や alvaModel で作られたモデルを使ってゼロから新しい分子をデノボ生成する alvaBuilder、分子データセットのキュレーションを行う alvaMolecule というソフトウェア群があります。いずれも無償の評価ライセンスをご提供していますので、ご興味があれば営業担当までご連絡ください。技術的な質問やご不明点がございましたら、弊社ウェブサイトの FAQ をご覧いただくかテクニカルサポート係までご連絡ください。

営業担当
sales@affinity-science.com

テクニカルサポート係
help@affinity-science.com

alvaDesc よくある質問（FAQ）
https://www.affinity-science.com/alvadesc-faq/

なお、開発元のホームページには alvaDesc をはじめとする Alvascience 社ソフトが使用され、引用されている文献のリストも掲載されています。alvaDesc などがどのように使われているか、ご参考にしていただければと思います。

Citations - Alvascience
https://www.alvascience.com/citations/

ぜひ本チュートリアルの内容を実践の中で活用し、ご研究に役立てていただければ幸いです。

7. 参考情報（ワークフローへの組込み）

ここまでは、alvaDescCLI をコマンドラインインターフェース上で実行する方法について説明して来ましたが、alvaDescCLI は Python から実行させることによりワークフローに組み込むこともできます。その概略について説明します。

7.1 alvaDescCLIWrapper

alvaDescCLI を Python のモジュールとして使うために alvaDescCLIWrapper をダウンロードし、インストールする必要があります。

・ダウンロードページ（alvaDesc 本体と同じ弊社 DL ページ）より ZIP ファイルをダウンロードし、任意のディレクトリに展開します。

・ alvaDescCLIWrapper ディレクトリの中に入っているセットアッププログラムをコマンドプロンプト上で実行します。

```
python setup.py install
```

尚、alvaDescCLIWrapper の実行要件は次の通りです。

・ライセンスが付与されたバージョン 1.0.14 以上の alvaDesc がインストールされていること。

・Python 3.5 以上がインストールされていること。

7.2 サンプルプログラム（Windows）

SMILES 表記されたブタンとアスピリンの molecular weight (MW)と average molecular weight (AMW)を計算させます。

① 入力

Python で AlvaDesc を使用することを宣言しそのパスを設定、クラス”set_input_SMILES”で分子を入力、クラス”calculate_descriptors”で計算させる記述子を指定、クラス”get_error”でエラーの場合の内容を出力指定、クラス”get_output”で計算結果を出力指定。

```
from alvadesccliwrapper.alvadesc import AlvaDesc

aDesc = AlvaDesc("C:/Progra~1/Alvascience/alvaDesc/alvaDescCLI.exe")

aDesc.set_input_SMILES(['CCCC', 'CC(=O)OC1=CC=CC=C1C(=O)O'])
if not aDesc.calculate_descriptors(['MW', 'AMW']):
    print('Error: ' + str(aDesc.get_error()))
else:
    print('Results: ' + str(aDesc.get_output()))
```

② 計算結果

```
Result:
[[58.14, 4.1529],[180.17, 8.5795]]
```

7.3 alvaDescCLIWrapper の詳細

alvaDescCLIWrapper を使用して Python から alvaDesc を利用する方法の詳細については、以下をご参照下さい。

- 弊社ホームページ内 “【alvaDesc】機能に関する FAQ” https://www.affinity-science.com/alvadesc-faq-cat2/
  - ✧ Python からの使用方法
  - ✧ Ubuntu 環境での alvaDescCLIWrapper のセットアップと使用方法

- Alvascience 社ホームページ内 ”alvaDesc – Python” https://www.alvascience.com/python-alvadesc/
- alvaDescCLIWrapper ディレクトリの”doc”内にある “alvadesc.html”
  （alvaDescCLIWrapper で使用可能なクラス・メソッドの説明ドキュメント）