fconv Tutorial Part 1

Gerd Neudert

Introduction to some basic fconv features based on version 1.08

Welcome to the first part of fconv tutorials!
It will cover a general introduction concerning help functions and fconv syntax, followed by an example to demonstrate PDB to MOL2 conversion and basics on atom type perception.

How to get help on fconv features?

```
fconv -h
fconv -h[pmsdc]
fconv -eg
fconv -wiz
```

- This command will show you the complete help on fconv. At first this help might be confusing, but proceeding through the tutorials, you will get familiar with it very quickly.

How to get help on fconv features?

```
fconv -h
fconv -h[pmsdc]
fconv -eg
fconv -wiz
```

- If you are only interested in e.g. help on functions for PDB files you can type `fconv -hp` or if you are interested in valid options for MOL2 files type `fconv -hm`

How to get help on fconv features?

```
fconv -h
fconv -h[pmsdc]
fconv -eg
fconv -wiz
```

- This option will print some very easy examples on the level also covered by this first tutorial.

How to get help on fconv features?

```
fconv -h
fconv -h[pmsdc]
fconv -eg
fconv -wiz
```

- You can type `fconv -wiz` to start a command-line wizard. He will guide you through some menus where you specify a task you have for fconv and finally he will give you the corresponding command-line. ATTENTION: The wizard is based on a very early fconv version (0.34), so only a small part of the available fconv features is covered by the wizard and some functions and explanations of the wizard are even outdated. Nevertheless, for some people it might be a nice tool to explore the character of the fconv syntax.

Syntax introduction

General usage

```
fconv Options1 Files [Options2]
```

I have to start with a little excuse. In the very first versions, the fconv syntax was logic and consistent, but as the scope of features was growing on rapidly and users had more and more "special wishes" I had to make some trade-offs. In the future there will be a completely new implementation of the command-line parser (i already have a nice concept in mind), but until then you have to get familiar with the current implementation.

Syntax introduction

General usage

```
fconv Options1 Files [Options2]
```

Options1 mainly comprises flags that specify what fconv should do, while Options2 is responsible for various parameters correlated with the given task. Options1 must always be given before any filenames and Options2 always after the filenames. This order is mandatory, because there are flags with the same name in Options1 and Options2 (e.g. `--m` is responsible for the atom type mode in Options1, while it is responsible for the maximum number of ring members in Options2).

Syntax introduction

General usage

```
fconv Options1 Files [Options2]
```

If you are familiar with other command-line-based programs you might be used to combine single letter flags, e.g. to write `-abc` instead of `-a -b -c`. Here comes the first trade-off in the fconv syntax; for some flags it is possible to combine them, but for many it isn't. My general suggestion is to always separate the flags to avoid unexpected results. The reason for this is, that I introduced multiple letter flags like `-rmsd` to handle the increasing number of options. For most programs, it is common to use consecutive hyphens for multiple letter flags, but for fconv those consecutive hyphen flags were reserved for other options a long time before I introduced the multiple letter flags...

Syntax introduction

General usage

```
fconv Options1 Files [Options2]
```

... Flags with a user defined string or number always start with a consecutive hyphen, e.g. `--r=5.5` to specify a radius of 5.5 Å or `--t=myname.mol2` to specify a target filename.

Syntax introduction

General usage

```
fconv Options1 Files [Options2]
```

Finally there is a second, more annoying trade-off, concerning the different input file types. Principally, MOL2, SDF and other files are mapped on the same internal object structure, thus allowing for the same features for all file types. Nevertheless, reading the help you will see that there are much more options available for MOL2 files compared to e.g. CIF files. This limitation will also be abolished with the planned reimplementation of the command-line parser.

Syntax introduction

General usage

```
fconv Options1 Files [Options2]
```

Generally fconv can process multiple files, so you can give an arbitrary number of files between Options1 and Options2 (e.g. `fconv -l *.pdb` to extract ligands from all PDB files in your current directory).

Important notes

- Be careful with your files, as there are some options which overwrite them. For example, if you type `fconv -W my.pdb` the original file my.pdb will be overwritten by a version without water molecules. You can always use `--t=newname` to specify a target filename in Options2.
- Make sure to check for new fconv versions from time to time:
  `https://pc1664.pharmazie.uni-marburg.de:433/index.php?topic=download`
- If you have questions or suggestions, contact me:
  neudert@staff.uni-marburg.de
  Please note, that this email may change in the future as I am finishing my PhD thesis. You will get new contact information from the README.txt which is part of each fconv download.

Important notes

- Be careful with your files, as there are some options which overwrite them. For example, if you type `fconv -W my.pdb` the original file my.pdb will be overwritten by a version without water molecules. You can always use `--t=newname` to specify a target filename in Options2.
- Make sure to check for new fconv versions from time to time: `https://pc1664.pharmazie.uni-marburg.de:433/index.php?topic=download`
- If you have questions or suggestions, contact me:
  neudert@staff.uni-marburg.de
  Please note, that this email may change in the future as I am finishing my PhD thesis. You will get new contact information from the README.txt which is part of each fconv download.

Important notes

- Be careful with your files, as there are some options which overwrite them. For example, if you type `fconv -W my.pdb` the original file my.pdb will be overwritten by a version without water molecules. You can always use `--t=newname` to specify a target filename in Options2.
- Make sure to check for new fconv versions from time to time: https://pc1664.pharmazie.uni-marburg.de:433/index.php?topic=download
- If you have questions or suggestions, contact me:
  neudert@staff.uni-marburg.de
  Please note, that this email may change in the future as I am finishing my PhD thesis. You will get new contact information from the README.txt which is part of each fconv download.

Extracting ligands

```
fconv -l 1CIL.pdb
```

Let's start with a very common task: extracting ligands from a PDB file as MOL2 file.
Please download 1CIL.pdb from the PDB, go to the directory where it resides and type the command shown in the box. fconv should tell you '`ETS_262 written to 1CIL_ETS_262.mol2`'.
Whatever you do with fconv you can specify your own target filename.

Extracting ligands

```
fconv -l 1CIL.pdb --t=test.mol2
```

Type in the new command shown in the box and fconv will write the ligand with the specified name 'test.mol2'

Extracting ligands

```
fconv -l 1CIL.pdb --t=test.mol2
```

Type in the new command shown in the box and fconv will write the ligand with the specified name 'test.mol2'
Now start a visualization software (e.g. Pymol) and have a look at 'test.mol2'
As you can see, the S=O bonds from the sulfonamide are shown as delocalized bonds. This is due to the assumed standard protonation states of fconv. You can adjust this behavior with the flag --p from Options1. So please have a look at `fconv -h` in the section Options1 and find out how to change protonation states.

Extracting ligands

```
fconv -l --p=101110 1CIL.pdb --t=test.mol2
```

Now you should know how to adjust the protonation states by a 6-digit binary. Now we will try to change the sulfonamide to a not deprotonated state. Digit two is responsible for `SO2NH`, so we will switch it from 1 to 0.
Type in the command shown in the box and have look at the new version of '`test.mol2`'.
As you can see, the S=O bonds are no longer delocalized, so we are finished with this part.

Extracting ligands

```
fconv -l --m=0 1CIL.pdb --t=test.mol2
```

Next, we should have a look at the assigned atom types. The carbons from the thiophene ring are labeled `C.2` which is Sybyl conform. Nevertheless, you might want to label them `C.ar`
Therefore we have to learn some basics about the fconv atom type perception. It has an internal set of different atom types which are always assigned in a first cycle. Afterwards, those internal types are mapped onto Sybyl types by default. If you have a look at the help under Options1 you will find the flag `--m` for changing the default mapping. Type the command shown in the box to get the internal fconv types.

Extracting ligands

```
fconv --M=1
```

As you can see, the internal type of the thiophene carbons is `C.arx`
Let's have a closer look at those internal types. Please type the command shown in the box and fconv will write out a file called '`Atom_types_mode1.def`', which is a definition file for the default mapping, hence for the mapping onto Sybyl atom types.
Please open this file in a text editor. It should be self explaining due to the introduction in each definition file. After the introduction follows a part where all internal atom types are explained. At the end there is a section which is introduced by <DEF> followed by a number of lines with 3 columns. The second column stands for an internal type, while the third holds the type it should be mapped on.
Now find the line with `C.arx` in the second column and change the third column to `C.ar`
Afterwards, please save your modified definition file as '`mydef.def`'

Extracting ligands

```
fconv -l 1CIL.pdb --t=test.mol2 --d=mydef.def
```

Please have a look at the help in Options2 and read the information about the `--d` flag.
As you can see, it is possible to use modified definition files for the atom type assignment. Type in the command shown in the box and have a look at the resulting file. The thiophene carbons should now have the desired type `C.ar`

Converting proteins

```
fconv -f 1CIL.pdb --t=1cil_full.mol2
```

There are two completely different modes for converting proteins from PDB to MOL2. First you can use option `-f`, which stands for full conversion and uses the same routines for atom type perception for the protein that are used for the ligands. In other words, this mode makes no difference between small molecules and protein. Thus, if you apply the command shown in the box, the resulting MOL2 file (1cil_full.mol2) contains a MOLECULE entry for the protein, one for the ligand and one for the zinc ion.

Converting proteins

```
fconv -c 1CIL.pdb --t=1cil_fast.mol2
```

The other option you can use is -c, which uses predefined templates to determine protein atom types. Currently there are only templates for amino acids available, so you can't use this for DNA or RNA. The resulting MOL2 (1cil_fast.mol2) only contains the protein part of 1CIL. It will be a full Sybyl conform protein-MOL2 containing BACKBONE, INTERRES, DICT and complete SUBSTRUCTURE entries.
While this mode is much faster, there are of course differences in the atom typing. If you have a look at the two files, you will see that e.g. LYS18 obtains atom type N.4 for the amine in the template-based conversion, while it is set to N.3 in the full conversion (you should already know how to change this behavior for the full conversion by changing the default protonation states).

- Part 2 will cover RMSD calculations and structural superpositions

Thanks go out to...

- Prof. Dr. Gerhard Klebe
- fconv bug reporters