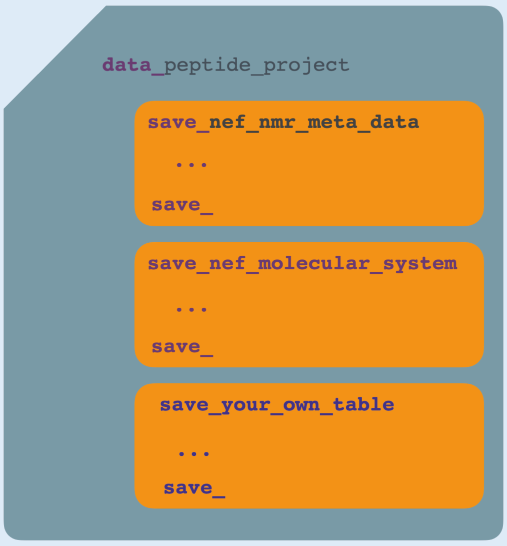
How To
Export Data to MARS and PINE for Auto assignment using NEF-Pipelines

Introduction

This *How To* will show you how to use NEF-Pipelines, a set of tools for manipulating NEF (NMR Exchange Format) files. NEF files are text files used for transporting files between programs used for NMR data analysis. NEF has been developed by a *consortium* of NMR software developers including CCPN. It is *not* a CCPN project, it is a community project.

What's inside a NEF File?

A NEF file is a text file, and specifically a STAR file so it has the same basic syntax as a NMRStar file or a mmCIF/PDBx file. However, it has a different and more simple structure designed for interchanging NMR data, at the most basic it's a series of tables with extra information. Here is the schematic of a NEF file... This file has a entry name `peptide_project`



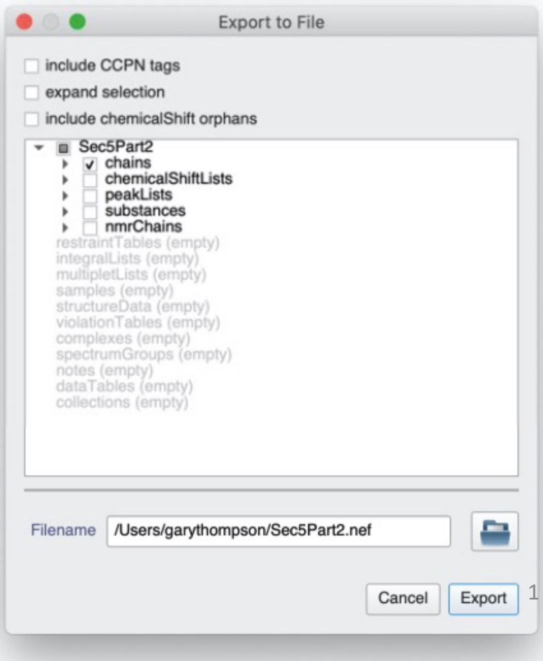
and three containers for tables of data which are called saveframes in the parlance of a STAR file. They are called `nef_nmr_meta_data`, `nef_molecular_system` and `your_own_table`. The first two are defined by the NEF standard and hold information about the particular file (meta data), the definition of the molecules in the file molecular system and a custom frame your_own_data_table (there are ways to save your own data in a NEF file and not clash with with the built in data).

Let's create our first NEF file and have a look at what's in it. To do this we will open the tutorial project **Sec5bPart2.ccpn** in CcpNmr Analysis and go to **File → Export NEF**. This will show the following dialog. We are only going to export the chains in the project and not include CCPN tags (extra data CCPN saves into NEF files using its own extensions). Save it to a file name of your choice with the extension `.nef`

To look inside the file we need to find a text editor on you computer. We will use the following ones:

Windows: `notepad`,
Linux: `kate / gedit`
MacOS `textedit`

Use the File → Open menus


Export to File

- [ ] include CCPN tags
- [ ] expand selection
- [ ] include chemicalShift orphans

Sec5Part2
- [✓] chains
- [ ] chemicalShiftLists
- [ ] peakLists
- [ ] substances
- [ ] nmrChains

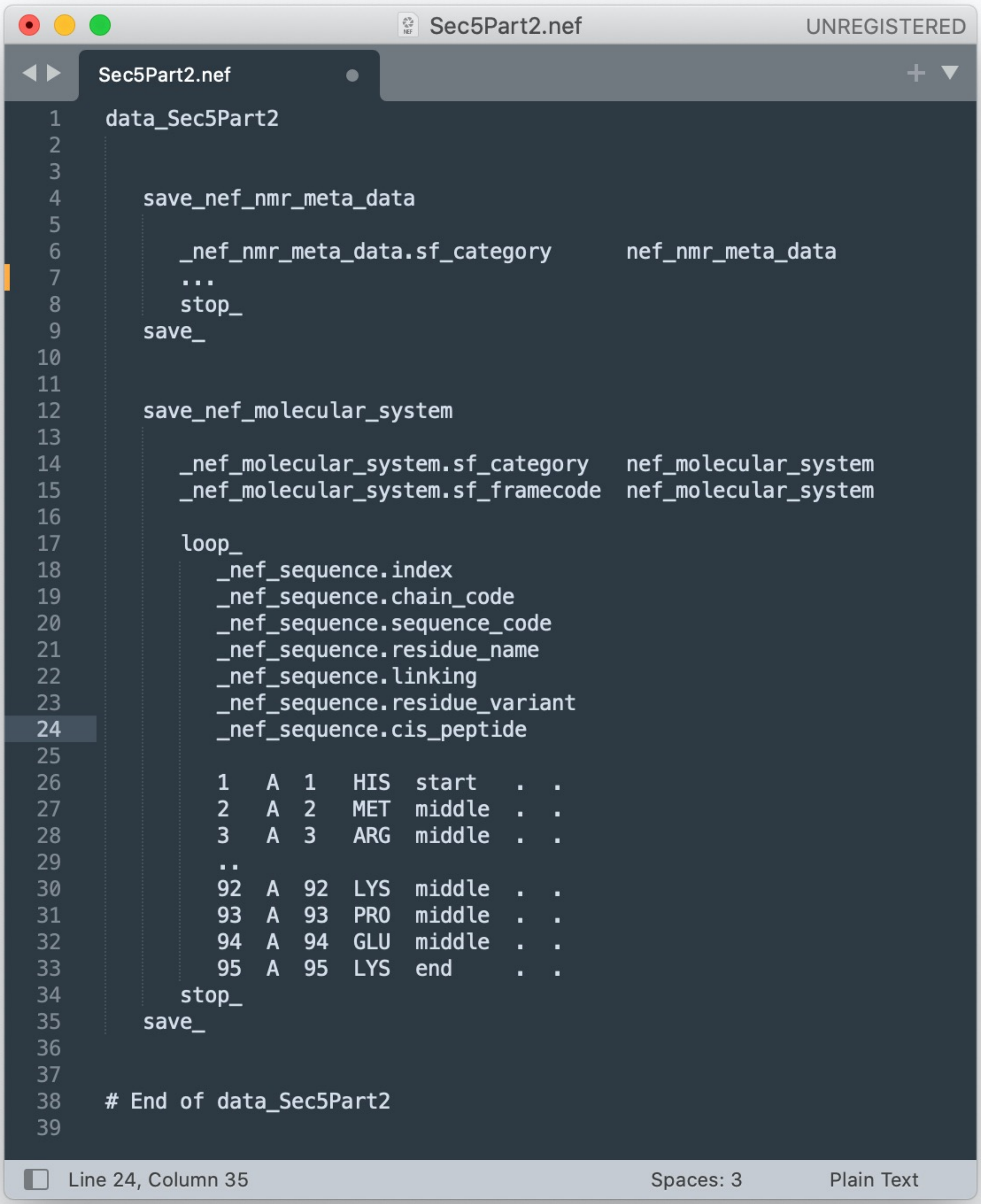
restraintTables (empty)
integralLists (empty)
multipletLists (empty)
samples (empty)
structureData (empty)
violationTables (empty)
complexes (empty)
spectrumGroups (empty)
notes (empty)
dataTables (empty)
collections (empty)

Filename /Users/garythompson/Sec5Part2.nef

Why does my OS think it's a photo?

Unfortunately NEF can also stand for Nikon Exchange Format a format used by cameras...

You should then find you have a text file that looks somewhat like the following:



```
data_Sec5Part2


   save_nef_nmr_meta_data

      _nef_nmr_meta_data.sf_category      nef_nmr_meta_data
      ...
      stop_
   save_


   save_nef_molecular_system

      _nef_molecular_system.sf_category   nef_molecular_system
      _nef_molecular_system.sf_framecode  nef_molecular_system

      loop_
         _nef_sequence.index
         _nef_sequence.chain_code
         _nef_sequence.sequence_code
         _nef_sequence.residue_name
         _nef_sequence.linking
         _nef_sequence.residue_variant
         _nef_sequence.cis_peptide

         1   A  1   HIS  start   .  .
         2   A  2   MET  middle  .  .
         3   A  3   ARG  middle  .  .
         ..
         92  A  92  LYS  middle  .  .
         93  A  93  PRO  middle  .  .
         94  A  94  GLU  middle  .  .
         95  A  95  LYS  end     .  .
      stop_
   save_


# End of data_Sec5Part2
```

There will be more text we have hidden which we have replaced with … to make it fit here. Lets look at the second save frame. This has some meta data which is required `sf_category [nef_molecular_system]` and `sf_framecode` and then a loop which is our table of data. In this case it has six columns `index`, `chain_code`, `sequence_code`, `residue_name`, `linking`, `residue_variant`, and `cis_peptide`. These define a chain of residues in a molecule with chain A starting at HIS 1 and ending at LYS 95. Note the table or loop starts with `loop_` and ends with `stop_`. The saveframe (which can contain multiple loops) ends with a `save_`. The entry (i.e. the contents of the file is is called `Sec5Part2` as the file starts with `data_Sec5Part2`. Note the `# End of data_Sec5Part2` at the end of the file is not required, this is just a helpful comment put in by CcpNmr Analysis to tell you where the end of the file is. In actual fact, generally, any text following a `# is a comment to the end of a line`. The only way to avoid this is to surround the text containing the # with quotes either single `‘# this is not a comment’` or double `”# this is also not a comment”`. Can you see where this may cause problems with how CcpNmr Analysis names chains?

Start A Terminal

- Mac users should start the terminal which is in the Utilities folder in Applications. It can be started by typing ⌘ **spacebar** and typing **terminal**
- Unix users type **Ctrl+Alt+T**
- Windows users open the start menu and type **powershell** in Search and open the Powershell app

Installing of NEF-Pipelines

NEF-Pipelines is a program being written by Gary Thompson (University of Kent), a member of the CCPN Working group. It is still unpublished. To install it you can either

1. Go to the github page https://github.com/varioustoxins/NEF-Pipelines and follow the latest installation instructions
2. If you have CcpNmr Analysis installed run the script `install_nef_pipelines` in the ccpnmr3.1.x bin directory which will either install the latest version or update the current installation to the latest version.

Making NEF-Pipelines & CcpNmr AnalysisAssign easier to start

To make these commands easier to start it's a good idea to add the ccpn bin directory to your PATH. To do this

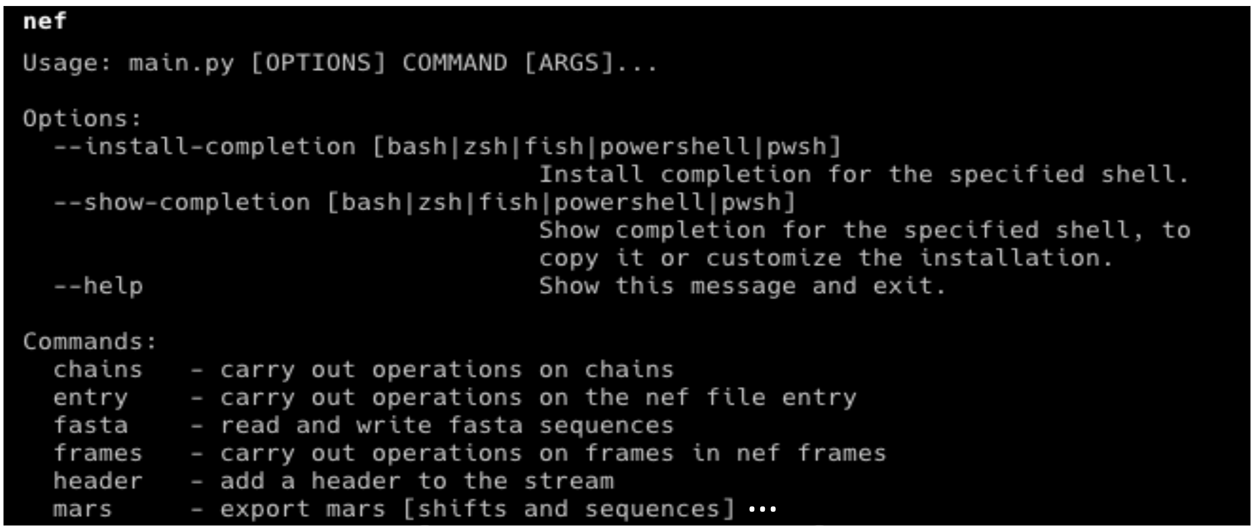
1. Find your ccpn bin directory and copy it down (for me it is **`/Users/garythompson/programs/ccpnmr/3.1.1/bin/assign`**)
2. Type **`echo $SHELL`** and note down the name of your shell (for me its **`zsh`** as this prints **`/bin/zsh`**)
3. Edit the startup file for you shell (for `zsh ~/.zshrc` for bash `~/.bashrc` for `csh ~/.cshrc`)
4. Add a line like the following at the end of you file

```
export PATH=/Users/garythompson/programs/ccpnmr/3.1.1
/bin:$PATH
```

5. This will allow you to start Nef-Pipelines by typing **`nef`**
6. This will allow you to start AnalysisAssign by typing **`assign`**

Navigating NEF-Pipelines

To see if NEF-Pipelines is installed type nef on the command line you should get something like the output shown on the next page. This output shows the available commands in NEF-Pipelines which either provide commands to deal with nef components `chains`, `entry`, `frames` etc. or commands to import and export nef data such `as nmrpipe nmrview mars pales` etc. :

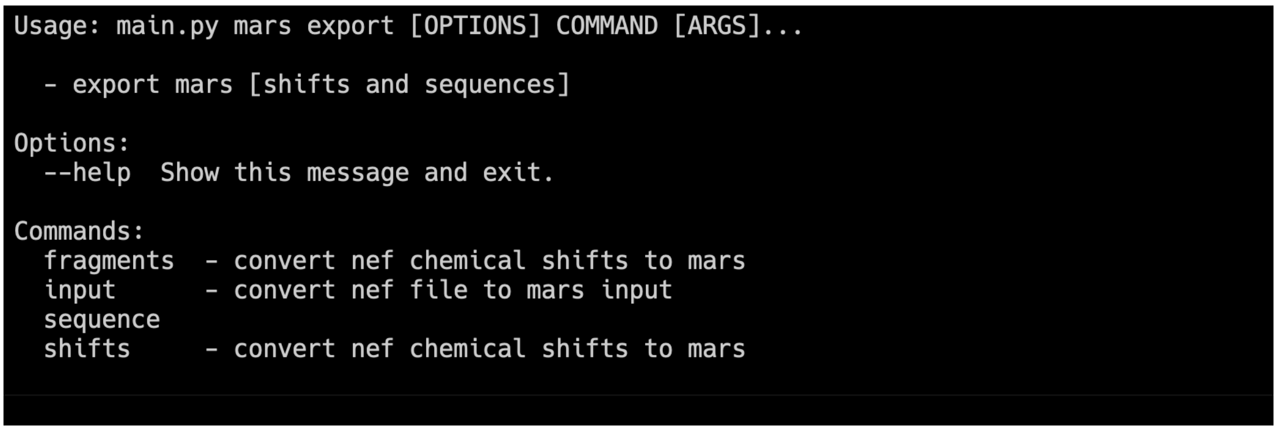


```
nef
Usage: main.py [OPTIONS] COMMAND [ARGS]...

Options:
  --install-completion [bash|zsh|fish|powershell|pwsh]
                                  Install completion for the specified shell.
  --show-completion [bash|zsh|fish|powershell|pwsh]
                                  Show completion for the specified shell, to
                                  copy it or customize the installation.
  --help                          Show this message and exit.

Commands:
  chains    - carry out operations on chains
  entry     - carry out operations on the nef file entry
  fasta     - read and write fasta sequences
  frames    - carry out operations on frames in nef frames
  header    - add a header to the stream
  mars      - export mars [shifts and sequences] ...
```

Generally, commands will show an error and this help screen or a more specific one if they don't understand your input.

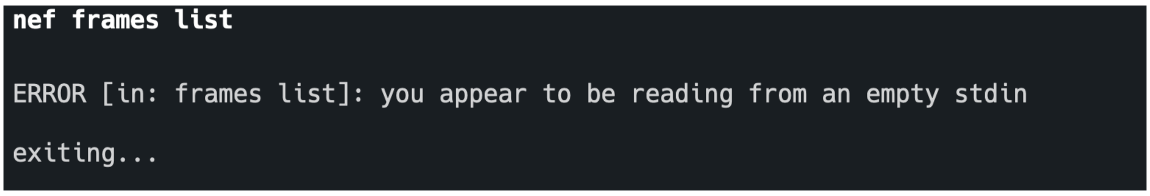
Tools may have sub-commands

Most tools have sub commands, for example the `mars` command has an `export` sub command which then has the further sub commands `sequence`, `shifts` and `input`. So, for example, typing `nef mars export` or `nef mars export --help` shows:



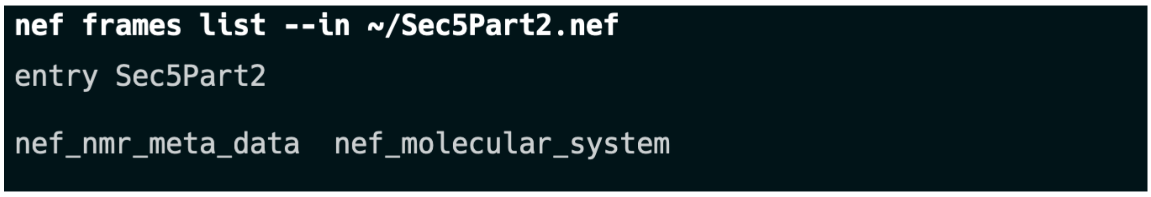
```
Usage: main.py mars export [OPTIONS] COMMAND [ARGS]...

  - export mars [shifts and sequences]

Options:
  --help  Show this message and exit.

Commands:
  fragments  - convert nef chemical shifts to mars
  input      - convert nef file to mars input
  sequence
  shifts     - convert nef chemical shifts to mars
```

Lets use our first Nef-Pipelines tool `nef frame list` if you type `nef frames list` on its own you will get something like:



```
nef frames list

ERROR [in: frames list]: you appear to be reading from an empty stdin

exiting...
```

This is because it requires an input NEF file. Let's give it the Sec5BPart2.nef file we just exported. For now to do this we use the option -in:



```
nef frames list --in ~/Sec5Part2.nef
entry Sec5Part2

nef_nmr_meta_data  nef_molecular_system
```

A we can see this contains the two frames we exported `nef_nmr_meta_data` `nef_molecular_system` and the name of the entry is `Sec5Part2`

We can also use command line options to get more information:

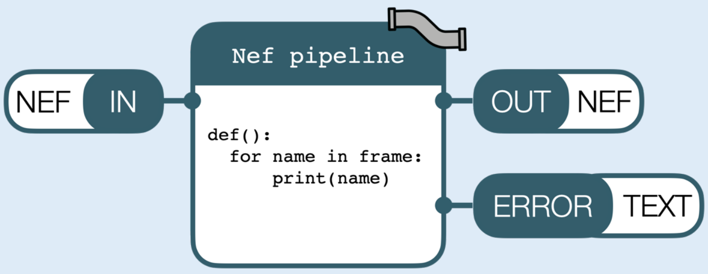
```
nef frames list --in ~/Sec5Part2.nef -v

entry Sec5Part2
    lines: 137 frames: 2 checksum: 8c69bc92ae0dbc995fc3df767a37e34e [md5]

1. nef_nmr_meta_data
    category: nef_nmr_meta_data
    loops: 1 [lengths: 19]
    is nef frame: True

2. nef_molecular_system
    category: nef_molecular_system
    loops: 1 [lengths: 13]
    is nef frame: True
```

As you can see this gives you considerably more detail (which is written to the error output and so allows us to put this in a pipeline, see below…)

Using pipelines

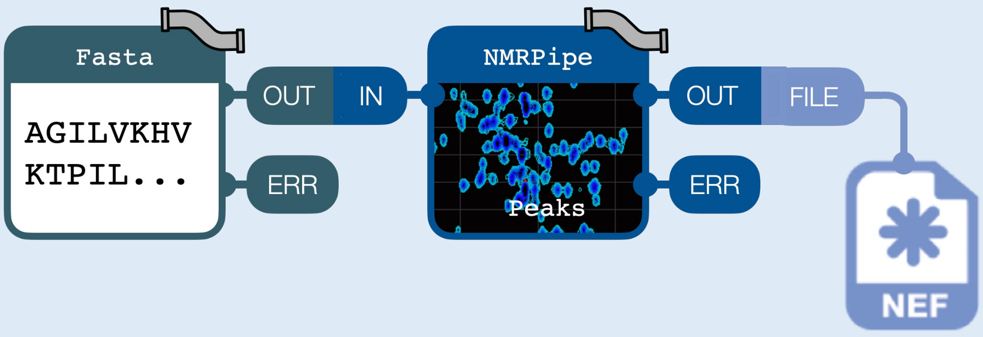
As expected from the name of the program, pipelines are an important part of the program that we haven't really investigated so far. So what's a pipeline and how do we use them?

Each NEF-Pipeline command is an individual program with an input `<IN>` and two outputs `<OUT>` and `<ERROR>; note <IN> and <OUT> read and write NEF text.`



```
def():
  for name in frame:
     print(name)
```

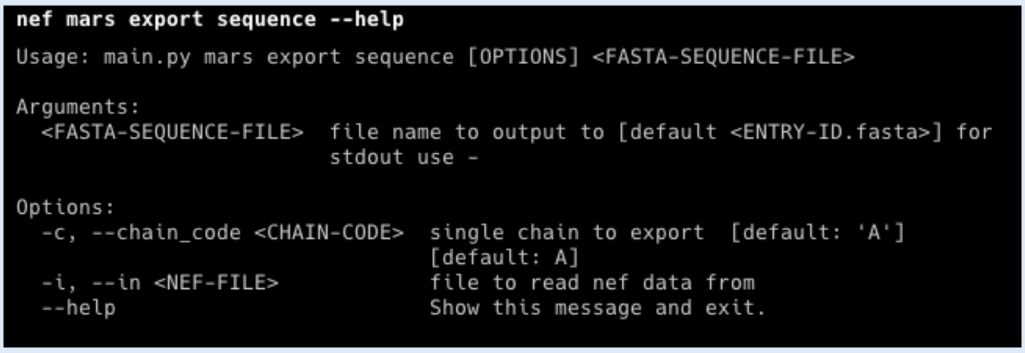
However, the real power comes from combining NEF-Pipelines commands together to build and modify a the text NEF 'file' as it is passed through the pipeline and then output to the screen or a file.



In this case, a sequence is read from a FASTA file and assigned peaks are read from NMRPipe and all the data are written as a NEF file to disc.

Using pipelines

Typing **`nef mars export sequence --help`** shows

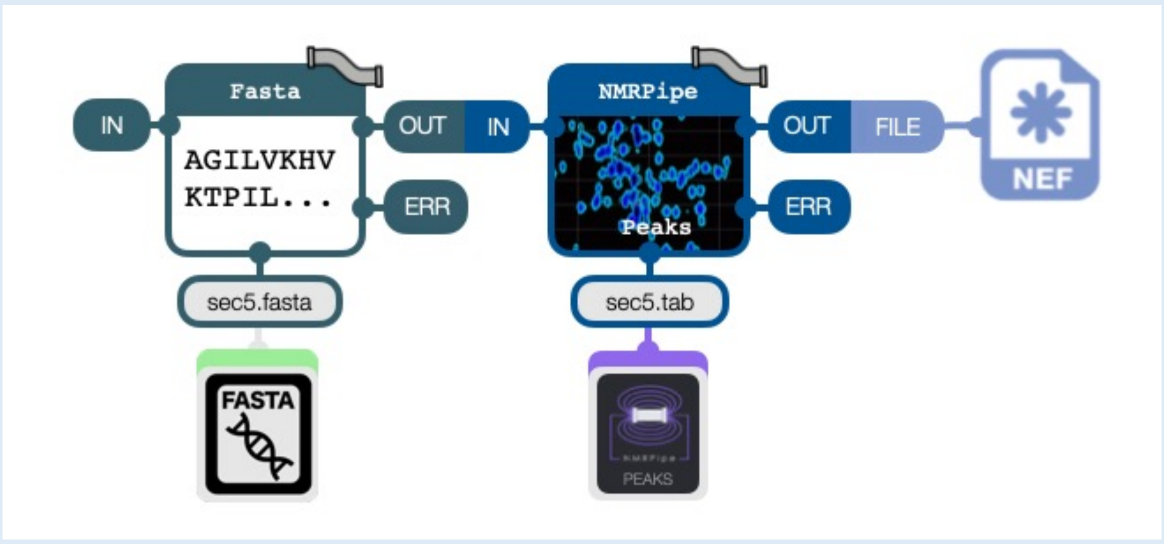


```
nef mars export sequence --help
Usage: main.py mars export sequence [OPTIONS] <FASTA-SEQUENCE-FILE>

Arguments:
  <FASTA-SEQUENCE-FILE>  file name to output to [default <ENTRY-ID.fasta>] for
                         stdout use -

Options:
  -c, --chain_code <CHAIN-CODE>  single chain to export  [default: 'A']
                                 [default: A]
  -i, --in <NEF-FILE>            file to read nef data from
  --help                         Show this message and exit.
```

This shows input arguments to feed the command with data and options that can be used to change how the command works. For example in this case **MARS** only works with a single molecular chain and we can use **`—c`** or **`--chain`** to select which chain to export from a NEF file if there is more than one. We won't use arguments or options today but they are an important part of NEF-pipelines.

Commands may be combined using the pipe symbol

Multiple commands can be combined using the pipe symbol: | under Windows, Linux and OSX. This feeds the output of one command to the next command after the pipe symbol. The > character can be used to write to a file. Below is shown symbolically a pipeline that reads a FASTA file produces a stream which contains a NEF file that is fed to the nmrpipe command that adds nmrpipe peaks to the stream and then writes out to a NEF file containing both sets of data. This is followed by the command line you would use to run these commands.

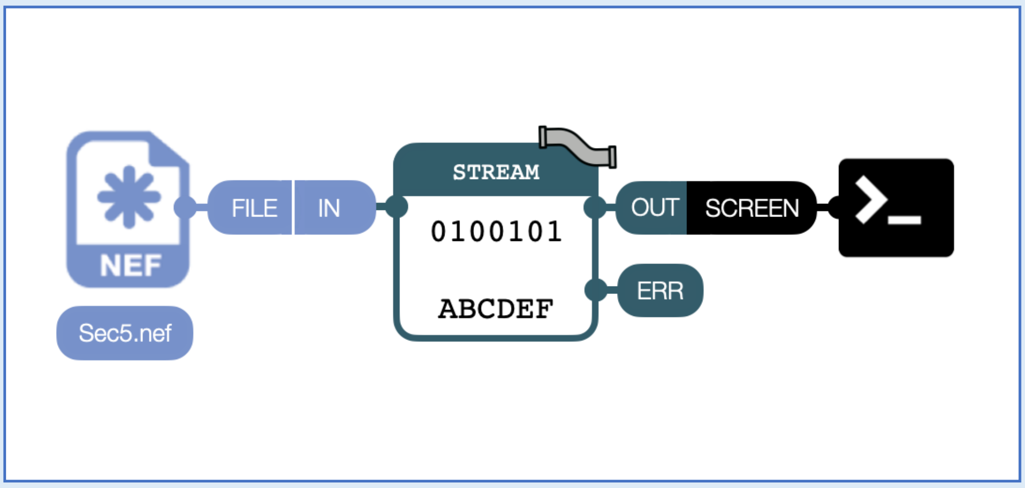


```
nef fasta import sequence sec5.fasta | nef nmrpipe import peaks sec5.tab > sec5.nef
```

1

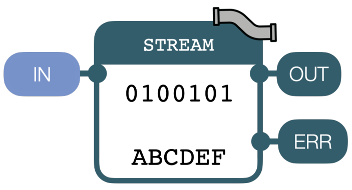
Streaming a NEF File to screen

First of all we are going to stream a provided NEF file, **sec5.nef**, to screen, the diagram below shows conceptually what we are doing:



Note how much data this NEF file contains, it is the information about a complete NMR assignment project with peaks, shifts, a molecular system and other data its quite complex. NEF-Pipelines is designed to manage this complexity…

As discussed above above we shall show a NEF pipeline tool. It is represented by the symbol on the right it has a name, an input (IN) and and 2 outputs (OUT and ERR).



We shall use these symbols throughout the tutorial.

1A Streaming the file to screen

To do this type the following in the terminal

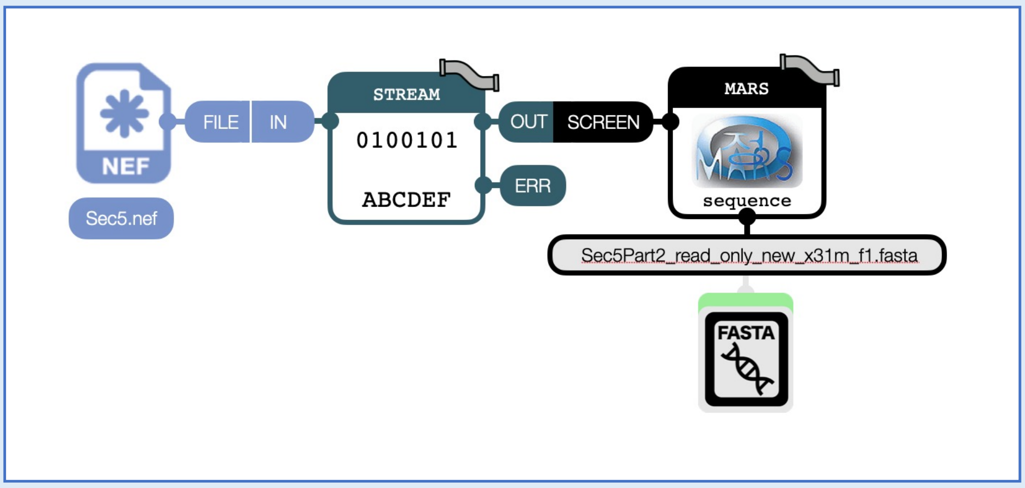
```
nef stream Sec5Part2.nef
```

Note that our data is stored **Sec5Part2.nef** which by default you saved to your home directory. You will need to correct the path to the file if you stored it elsewhere. Note we have not shown all the text out there are lost of lines …

```
data_Sec5Part2


   save_nef_nmr_meta_data

      _nef_nmr_meta_data.sf_category      nef_nmr_meta_data
      _nef_nmr_meta_data.sf_framecode     nef_nmr_meta_data
      _nef_nmr_meta_data.format_name      nmr_exchange_format
      _nef_nmr_meta_data.format_version   1.1
      _nef_nmr_meta_data.program_name     AnalysisAssign
      _nef_nmr_meta_data.program_version  3.1.1
      _nef_nmr_meta_data.creation_date    2023-05-15T08:50:09.105711
      _nef_nmr_meta_data.uuid             AnalysisAssign-2023-05-15T08:50:09.105711-1722851096
      _nef_nmr_meta_data.coordinate_file_name  .

      loop_
         _nef_program_script.program_name
         _nef_program_script.script_name
         _nef_program_script.script

         CcpNmr  exportProject  .
      stop_
   save_
                                                                                                              …
```

Molecular Sequence to Fasta

Now we will not connect the stream to the screen but to a tool that will write out a sequence for MARS. Conceptually we have



1B Outputting the stream to a MARS sequence file

First lets go back to CcpNmr AnalysisAssign and load the tutorial project **Sec5Part4.ccpn** and output all the frames in the project to a file called **Sec5Part4.nef**.

```
nef stream ~/Sec5Part4.nef | nef mars export sequence
```

Note you may see text like the following, this is a bug from one of libraries NEF-Pipelines uses, it will disappear in a future version and and can be ignored!

```
2022-11-29 23:37:13,469,469 WARNING  [parser.py:161] Loop with no data on line: 2993
2022-11-29 23:37:13,469,469 WARNING  [parser.py:161] Loop with no data on line: 3020
```

If you type `ls` in your terminal you will now find a new file called Sec5Part4.fasta this is a rather ugly name that comes from the name of the contents of the NEF file, we will make it a nicer name a bit later.

If you open the file Sec5Part4.fasta with a text editor you will find it contains the text

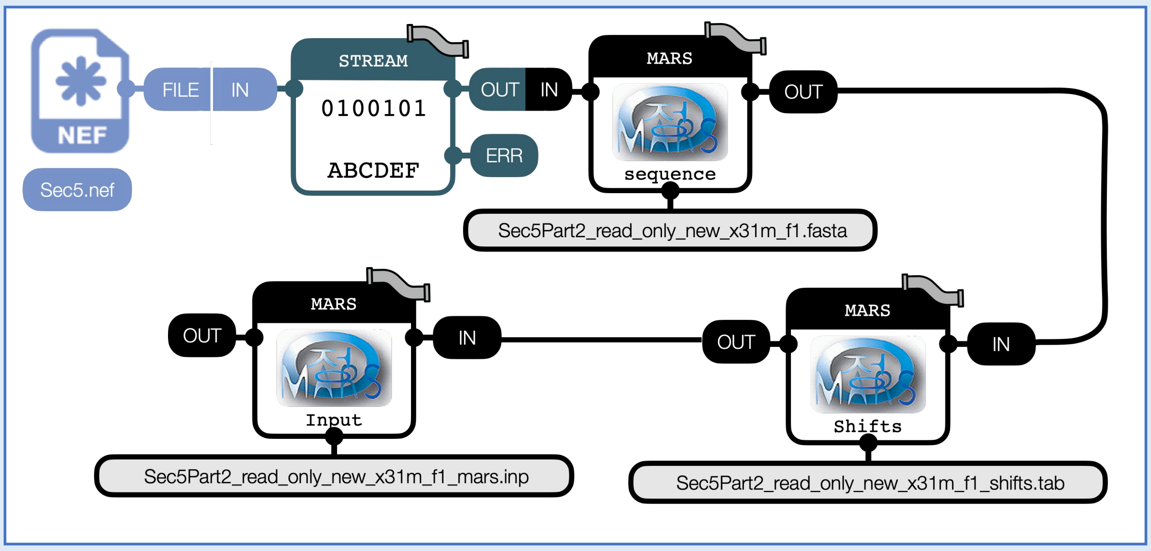
```
>CHAIN: A | START RESIDUE: 3
HMRQPPLVTG ISPNEGIPWT KVTIRGENLG TGPTDLIGLT ICGHNCLLTA EWMSASKIVC RVGQAKNDKG DIIVTTKSGG
```

This is the sequence of the protein sec5 in the format used by MARS (fasta). This is quite different to how the sequence is stored in the NEF file. If there was more than one chain in the NEF file NEF-Pipelines would ask you to select which chain to use as MARS can only run on a single chain…

Note how you no longer see lots of NEF text written to screen. The pipeline assumes you don't want lots of screen output and so absorbs it at the end of the pipeline to make your experience easier. There is a way to display it again using the **cat** command on OSX and Linux and type / Get-Content under Windows PowerShell but that is beyond the scope of this tutorial.

Writing the rest of the files

MARS takes several files which it needs to run. We will now add the extra tools that write these files (shifts and most of a project input file).


NEF
Sec5.nef
FILE
IN
STREAM
0100101
ABCDEF
OUT
ERR
IN
MARS
sequence
OUT
Sec5Part2_read_only_new_x31m_f1.fasta
MARS
Input
OUT
IN
Sec5Part2_read_only_new_x31m_f1_mars.inp
MARS
Shifts
OUT
IN
Sec5Part2_read_only_new_x31m_f1_shifts.tab

1c Outputting the stream to the rest of the MARS files

To do this, type the following in the terminal. Please write all the text on one line (here it is spread out over more than one line to make it fit on the page!). Note that you can use the up and down arrows to show previous commands you ran which can be edited and rerun by typing return again

```
  nef stream ~/Sec5Part4.nef | nef mars export sequence | nef mars export shifts
| nef mars export input
```

If you type `ls Sec5*` in your terminal you will now find you have quite a lot more files

```
Sec5Part4.fasta
Sec5Part4_mars.inp
Sec5Part4_mars_fix_ass.tab
Sec5Part4_mars_pred.tab
Sec5Part4_shifts.tab
```

…

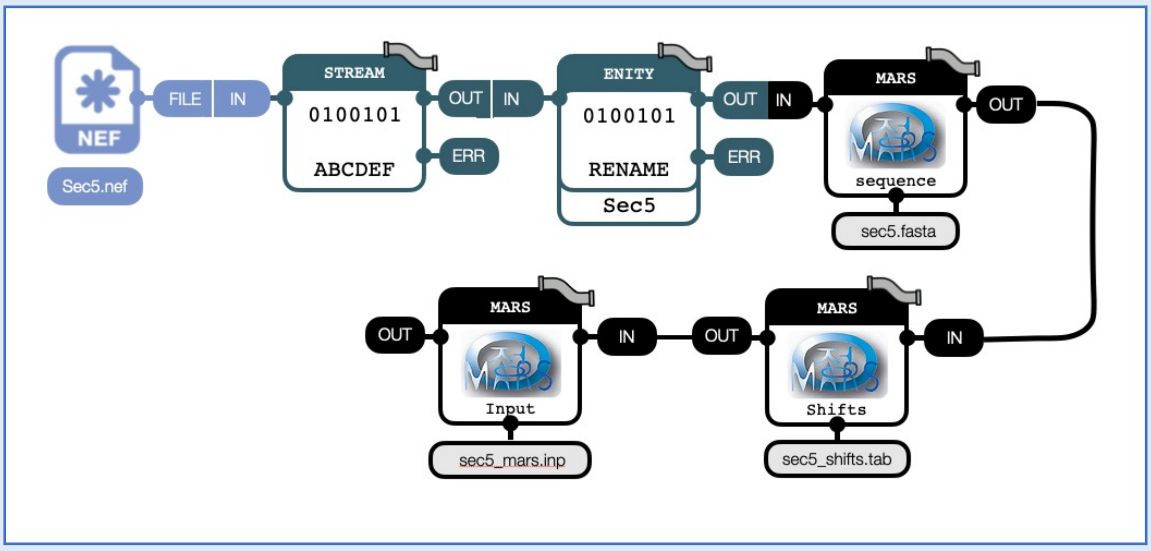
These are the rest of the files you need to run MARS: a sequence file, a list of chemical shifts and an input file. If you investigate the input tool you will find options to write in the input file if it is data for an unfolded protein and if it is a deuterated protein.

The files **…_ass.tab** and …**_pred.tab** tell MARS about residue types and strings of residues that have already been combined together by previous assignments. These are currently empty as the MARS tool is not sophisticated enough to write them yet, you would have to write them yourself!

Please also note that you would need to obtain a psipred formatted prediction of the protein's secondary structure to run MARS, though we may be able to derive this from alphafold more quickly in the future….
Psipred can be found at http://bioinf.cs.ucl.ac.uk/psipred/.

Making it less ugly

The names of the files the MARS tools output are taken from the name of the NEF file Entry. If we change this at the start of the stream it will give better file names but won't affect the original file on disk (unless you write it first).



1D **Renaming the NEF file Entry**

If we rename the entry inside the NEF file (the structure that contains all the data) we will change the names of all the output files when they are exported as their names are based on the entry name…

```
  nef stream ~/Sec5Part4.nef | nef entry rename sec5
| nef mars export sequence | nef mars export shifts
| nef mars export input
```

If you type **`ls sec5*`** in your terminal you will now find you have files with much nicer names

```
sec5.fasta
sec5_mars.inp
sec5_mars_ass.tab
sec5_mars_pred.tab
sec5_shifts.tab
```

Making input files for PINE

To show how these commands can be generalised and saved to a file, let's also export shifts for assignment using iPine. iPine http://pine.nmrfam.wisc.edu/index_submitform.html is another automated assignment program. However, rather than a shift list in the input format used it requires a set of (Sparky format) peak lists specifically without assignments. To build these files we will read a chemical shift list from a NEF file and use NEF-Pipeline commands to build the NEF peak tables for triple resonance spectra which we will then export to Sparky files for input into iPINE.

```
nef stream ~/Sec5Part4.nef
| nef entry rename sec5
| nef frames delete spectrum --category
| nef shifts make-peaks --spectra triple,n_hsqc
| nef sparky export peaks --discard-assignments  --add-data --suppress-column height
```

We will now show you how we can build a tool that can be used more than once. To do this we will make a script we will call **nef_to_pine.sh**. You should enter the lines below into this file using your favourite text editor. Note: there must be **<u>no</u>** spaces after the \'s at the ends of the lines

```
#!/bin/bash
  nef stream $1                                                 \
| nef entry rename sec5                                         \
| nef frames delete –category spectrum                          \
| nef shifts make-peaks --spectra triple,n_hsqc                 \
| nef sparky export peaks --discard-assignments --add-data      \
                          --suppress-column height
```

If you type

```
bash nef_to_pine.sh ~/Sec5Part4.nef
```

This will carry out the commands to convert a NEF file to iPine input files for any NEF file that contains the correct data.

To make this easier to run you can make the script executable:

```
chmod u+x nef_to_pine.sh
```

And then you can just type the following without the bash in front to run it:

```
nef_to_pine.sh ~/Sec5Part4.nef
```

A summary of the commands for this example are given on the next page

```
nef stream <FILE-NAME>
```

Stream a nef file into a pipeline (this is equivalent to the unix command cat)

```
nef entry rename sec5
```

Rename the NEF entry to sec5. By default `sparky export peaks` uses the entry as the base for its file names, but you can give it a filename template instead

```
nef frames delete spectrum --category
```

Delete all frames that have a category spectrum. Each saveframe has a category, we delete all spectra as we are going to reconstruct our own triple resonance peaks from average shifts. This is so the data that goes into iPine is the same as that that goes into MARS. You could, however, just use the raw peak lists if this doesn't worry you and you don't want to make comparisons.

```
nef shifts make-peaks --spectra triple,n_hsqc
```

Make triple resonance peak lists from the default chemical shift list (you can specify another chemical shift list if you want, you can also specify individual triple resonance spectra, hnca etc. rather than the catchall triple if you want)

```
nef sparky export peaks --discard-assignments --add-data
                        --suppress-column height
```

Export Sparky peak lists. There will be one peak list for each nef_spectrum saveframe in the NEF file steam running through the pipeline at this point, unless you specify the spectrum saveframes to use. We also discard assignments* as iPine wants files without assignments. The **–add-data** adds the text '`data`' to the Sparky headers as iPine wants this but not all programs that read Sparky files want or need this in their header. **–suppress-column height**: iPine wants to have a volume column but doesn't take a height column. Note the volumes are filled in with a dummy height of the correct sign…

* Notes

1. The ability to delete assignments including some more sophisticated capabilities will appear soon as a separate command at some point. At that point this option may disappear…
2. The defaults used for exporting iPine peaks may get moved into a separate `ipine export peaks` command at some point.

FAQs

Q. Can I use NEF-Pipelines as a python library?

A. Yes, partially now, but it isn't implemented for all commands, watch this space!

Q. Is there a GUI?

A. Not currently, but there is a plan to add a GUI based on the CCPN pipeline GUI.