SCAMP Proteomics Pipeline Guidelines

The SCAMP directory

1. Scamp script
2. Download script
3. SQLite creation script
4. Fetch script
5. MSFragger utilities
6. MSFragger parameter template
7. Maxquant version of pipeline
8. Fixed modification text files
9. Enzyme text files

```
gwips@baileys:/home/DATA2/trips/scamp$ ls
AssayApi.py                __init__.py                philosopher          PXD020451
AssayApi.pyc               merge_proteomics_sqlites.py ProjectApi3.py      __pycache__
closed_fragger.params      models                     ProjectApi.pyc       raw_to_mzml.py
download3.py               msfmods_0.txt              proteomes            scamp_maxquant
downloadfragger.py         msfmods_1.txt              PXD020151_0          scamp.py
downloadfragger.py.save    msfmods_2.txt              PXD020151_0.params   swagger3.py
Fetch.py                   msfmods_3.txt              PXD020151_1          swagger.pyc
FileApi3.py                MSFragger                  PXD020151_sqlites    Trypsin.txt
FileApi.pyc                msfragger_to_sqlite.py     PXD020239            wget-3.2
Finished                   nohup.out                  PXD020256_0          wget.py
finishedprojects.txt       params                     PXD020272            wget.pyc
fragger.params             peptides_to_position2.py   PXD020355            ZJunk
```

scamp.py

The scamp script can be used to automatically run multiple projects through the pipeline consecutively. It does this by feeding each project to the download script in a loop. The script is entitled ‘scamp.py’ and can be passed any number of PRIDE project accessions as arguments:

```
gwips@baileys:/home/DATA2/trips/scamp$ python3 scamp.py PXD000001 PXD000002
```

downloadfragger.py

The download script is the bones of the scamp pipeline, automating file download and analysis, currently with MSFragger. An older version of the pipeline that utilizes Maxquant for the analysis stage can be found in the ‘scamp_maxquant’ folder. The downloadfragger.py retrieves meta information about the project such as species, fixed modification, enzyme, and data type from PRIDE before downloading and processing the raw files in batches. It can be called from the scamp directory as follows, but only takes one PRIDE accession as an argument:

```
gwips@baileys:/home/DATA2/trips/scamp$ python3 downloadfragger.py PXD000001
```

Various errors or warning messages may pop up during the running of the pipeline which will be explained here. For example, if the pipeline is unable to find the fixed modification in either the sample or data processing protocol, the default modification Carbamidomethyl (C) will be used for the analysis and you will be informed accordingly. This warning does not require any further action but is useful to note when looking at the results.

```
gwips@baileys:/home/DATA2/trips/scamp$ python3 downloadfragger.py PXD000001
…
SCAMP could not find the fixed modification in the information retrieved
from PRIDE. The default of Carbamidomethyl (C) will be used.
```

Enzymes in downloadfragger.py

When the pipeline is passed a PRIDE accession, it automatically retrieves the enzyme used in the sample preparation which is essential for the MSFragger analysis. As it stands now, Trypsin is the most common (almost sole) enzyme used, and is the default. If SCAMP cannot find the enzyme used in the sample or data processing protocol, it will print the protocol to the terminal and ask you to manually enter the enzyme. The enzyme must be entered in sentence case, for example 'Chymotrypsin'.

```
gwips@baileys:/home/DATA2/trips/scamp$ python3 scamp.py PXD000001
[Project Title]
[Project Description]
SCAMP could not find the enzyme in the information retrieved from PRIDE.
Please read the processing protocol to see if the enzyme is specified.

[Processing Protocol]

Please manually enter the enzyme for this study, or press enter to
continue with the default (Trypsin)
```

If the enzyme in the study is not one the pipeline has in its dictionary, you will be prompted by the script to add some information about it before continuing. The pipeline will automatically add this enzyme to its dictionary, so you will not be asked to do this again for the same enzyme. You will be asked for the AA residues after which the enzyme will cut, and for residues after which it explicitly will not cut. This information is readily available online for most enzymes, with a convenient location being on the Worthington Biochem entry for a given enzyme, under the 'specificity' heading.

```
gwips@baileys:/home/DATA2/trips/scamp$ python3 scamp.py PXD000001
[Project Title]
[Project Description]
SCAMP could not find the enzyme in the information retrieved from PRIDE.
Please read the processing protocol to see if the enzyme is specified.

[Processing Protocol says Chymotrypsin]

Please manually enter the enzyme for this study, or press enter to
continue with the default (Trypsin)
Chymotrypsin
The enzyme for this study has not been encountered by SCAMP before.
Please manually enter the residues after which this enzyme cleaves.
FWY
Now please enter the residues after which the enzyme will NOT cut.
P
```

In the event that you are prompted to enter details for a new enzyme, it is good practice to add that enzyme to the SCAMP script in order to allow the script to automatically identify that enzyme in future (this will eliminate your manually having to enter the enzyme name for pipeline to proceed). For this, you will need to edit the downloadfragger.py script, and copy the format such as that for Trypsin under the #GET ENZYME comment.

<downloadfragger.py>

```
#GET ENZYME
trypsin = ["trypsin", "Trypsin", "Tryptic", "tryptic"]
for x in trypsin:
        if x in processing_protocol:
                enzyme = "Trypsin"
```

Isotope-labelling Data and the SCAMP Pipeline

Isotope-labelled data such as TMT, SILAC or iTRAQ, is currently not recommended for analysis by this pipeline as it is focused towards quantification rather than qualification, which is not the goal of this analysis. The SCAMP pipeline automatically searches for references to isotope labelling in projects that are passed to it and if it suspects a project of having isotope labelling data, prints the processing protocol to the terminal and asks for confirmation before proceeding.

```
gwips@baileys:/home/DATA2/trips/scamp$ python3 scamp.py PXD000001
[Project Title]
[Project Description]

SCAMP has found reference to Isotope-labelled data in the processing
protocol for this project. Such data is not currently recommended for
analysis with this pipeline. Please read the processing protocol and press
enter should you wish to proceed.

[Processing Protocol]
```

Should the objective of the pipeline change in the future, Nesvilab Philosopher has a pipeline feature which will carry out analysis of TMT data that can be easily introduced to the downloadfragger.py script.

Manual Download Feature

Certain projects on the PRIDE archive store raw files in zip files, or in other formats. The pipeline will recognize these projects as having no raw files, so in cases like this you will be given the option to utilize the manual download function of the pipeline.

```
gwips@baileys:/home/DATA2/trips/scamp$ python3 scamp.py PXD000001
[Project Title]
[Project Description]
This project does not have any raw files, would you like to execute the
manual download function? (y/n)
```

Navigate to the project page on PRIDE and see if there are zip or rar files you wish to analyse, and enter ‘y’ if so.

```
gwips@baileys:/home/DATA2/trips/scamp$ python3 scamp.py PXD000001
[Project Title]
[Project Description]
This project does not have any raw files, would you like to execute the
manual download function? (y/n)
y
INSTRUCTIONS: Please manually download the files from PRIDE into the
[accession]_manualdwnld folder that has just been created in the scamp
directory.
When the download is complete, press enter to continue.
```

Download and unzip the files in the folder created for you by the pipeline and press enter. The pipeline will continue the process as usual.

msfragger_to_sqlite.py

The msfragger_to_sqlite.py script takes the output of the downloadfragger.py script and parses the relevant information to create sqlite files interpretable by TripsViz. The information in question includes: Peptide length, the transcript in which it was found, and the position within that transcript. The downloadfragger.py script automatically calls the msfragger_to_sqlite script, so there is not usually a need to use this script manually. However, should the need arise, the script should be passed the batch folder path, the project organism, and the project accession.

```
gwips@baileys:/home/DATA2/trips/scamp$ python3 msfragger_to_sqlite.py
PXD000001_0 human PXD000001
```

As mentioned, among the information this script parses is the transcript in which a given peptide is found. Due to the treatment of the proteome database required for MSFragger action, some peptides are assigned to ‘wildcard’ transcripts that do not meet the standard formatting. During the creation of the SQLite files, these peptides and their transcripts are flagged, and the problem peptides are omitted from the SQLite files.

```
gwips@baileys:/home/DATA2/trips/scamp$ python3 msfragger_to_sqlite.py PXD000001_0
human PXD000001
…
Reading lines for PXD000001_0/final_file1.txt
Unrecognised transcript, skipping read for peptide AAQSLKAK in sp|P00761|TRYP_PIG
Unrecognised transcript, skipping read for peptide AAARGUUQQ in sp|P00584|Keratin
Unrecognised transcript, skipping read for peptide AUKLYSKTW in sp|P00584|Keratin
Unrecognised transcript, skipping read for peptide AAGURQST in sp|P00761|TRYP_PIG
Unrecognised transcript, skipping read for peptide AUKWTSRGQ in sp|P021441|myg_human
Unrecognised transcript, skipping read for peptide AWSGDUQP in sp|P00584|Keratin
Unrecognised transcript, skipping read for peptide AAQSLK in sp|P021441|myg_human
```

Fetch.py

The Fetch.py script can be used to search the entire PRIDE Archive from the terminal to find projects matching a particular keyword, then automatically initiate their running through the pipeline.

```
gwips@baileys:/home/DATA2/trips/scamp$ python3 Fetch.py coronavirus
Searching PRIDE for count of projects containing keyword coronavirus
10 projects on the PRIDE Archive matched this keyword
List of relevant projects has been obtained
['PXD004557', 'PXD017545', 'PXD010494', 'PXD019119', 'PXD009975', 'PXD007107',
'PXD018581', 'PXD002936', 'PXD018117', 'PXD018241']
Checking list obtained from PRIDE for already completed projects
Fetch can recommend 10 of projects from the PRIDE archive based on your criteria.
How many of these would you like to send to the SCAMP Pipeline?
```

For now, this script just provides a list of projects that fit the criteria. However, towards the end of the script there is a space where a subprocess call could be placed to automatically feed a given number of projects to the pipeline from the list found by Fetch.

<Fetch.py>

```
number = len(finallist)
print("Fetch can recommend "+str(number)+" of projects from the PRIDE archive based
on your criteria. How many of these would you like to send to the SCAMP Pipeline?")
scamparg = input()

#could insert a subprocess call here using scamparg to call the first x number of
projects in the list
```

Any questions? Feel free to email Ciara at judge.ciara@gmail.com or ask Stephen! ☺