Description

SNPMeta is a tool that quickly and easily collects metadata about SNPs in a particular organism, and produces a report suitable for dbSNP submission, or a tabular report for other organism-specific databases. It can be used to annotate SNPs from either genotype by sequencing (GBS) approaches or other next generation sequencing approaches, or oligo hybridization-based approaches such as Illumina Infinium hybridization technologies. Because the contextual sequences that flank these SNPs are handled in a variety of formats, we provide utility scripts to convert them into a format appropriate for SNPMeta.

Installation

SNPMeta is a standalone Python application and can be installed in any user accessible directory on a Linux or UNIX-based computer (for compatibility details see below).

Dependencies

SNPMeta depends on the following tools to be installed and accessible to the user:

- Python 2, version $\geq$ 2.6 (`http://python.org/`). Note that if Python 2.6 is being used, the `argparse` must also be installed. BioPython is not extensively tested with Python 3, so SNPMeta is also untested with Python 3. Use caution. For Windows, it is necessary for BioPython that the 32-bit version of Python be installed, regardless of your actual computer architecture.
- (Linux and MacOS only) C compiler, for building and installing BioPython and EMBOSS tools. On Linux, the GNU C Compiler (gcc) should already be available. On MacOS, a C compiler is freely available from the `Apple Developer Tools`.
- BioPython (`http://biopython.org/`). Download the 'source' ZIP installation package, unzip it, and follow the instructions in the included 'README' file. For Windows, the pre-built exe works - there is no need to compile the library.
- EMBOSS (`http://emboss.sourceforge.net/`). Download the 'Stable Release' and follow their compilation instructions. To install the programs, it is necessary to run '`make install`' after completing the compilation instructions. You may need super user privileges to perform this

action. On Windows, an older version (6.5.0, at the time of this writing) is available. It can be found under the '`windows`' directory in the EMBOSS download FTP site, and works in the same way as the current version for UNIX-like operating systems.

- (Optional) NCBI BLAST+ ([Link]). Follow the FTP link, and download the appropriate package for your platform. This is only required if running BLAST against a local database.

SNPMeta has been tested on, and run successfully on the following platforms:

- CentOS 6.3 (GNU/Linux), Python 2.7.2, BioPython 1.58, EMBOSS 6.5.7
- OSX 10.6.8 (Snow Leopard), Python 2.6.1, BioPython 1.61, EMBOSS 6.5.7
- OSX 10.5.8 (Leopard), Python 2.7.3, BioPython 1.61, EMBOSS 6.5.7
- OSX 10.8.3 (Mountain Lion), Python 2.7.3, BioPython 1.60, EMBOSS 6.5.7
- Windows XP SP3, Python 2.7.3, BioPython 1.61, EMBOSS 6.5.0

Running SNPMeta

SNPMeta is a command-line application; it must be used through a terminal emulator (Terminal in Mac OS X, XTerm/RXVT/etc. on Linux or UNIX, and Command Prompt in Windows). To execute SNPMeta, pass the script to the Python interpreter by typing:

```
python SNPMeta.py
```

followed by any desired arguments. On Windows, you must specify the full path to the Python executable.

```
C:\Python27\python.exe SNPMeta.py
```

Alternately, on UNIX-like operating systems, you may give executable permissions to SNPMeta, and it can be placed in the current user's PATH. This is a little more complicated, but makes SNPMeta accessible from anywhere in the system. To do this, type

```
mkdir -p ~/bin
chmod +x SNPMeta.py
cp SNPMeta.py ~/bin
export PATH=${PATH}:${HOME}/bin
```

SNPMeta can then be executed by typing:

```
SNPMeta.py
```

on the command line, followed by any arguments. From now on, the text '`SNPMeta.py`' will be used to refer to the command that needs to be called to execute SNPMeta.

Basic Usage

To see a brief overview of the options and arguments SNPMeta accepts, pass the `-h` switch on the command line:

```
SNPMeta.py -h
```

Minimally, SNPMeta takes three arguments:

```
SNPMeta.py -f [FASTA_FILE] -l [LENGTH] -a [EMAIL]
```

where `[FASTA_FILE]` is the file containing the SNP contextual sequences, `[LENGTH]` is the length of one side of the contextual sequence around each SNP, and `[EMAIL]` is the email address to send to NCBI with batch requests.

The above usage is for annotating a list of SNPs kept in a single FASTA file. SNPMeta can annotate a directory of SNPs (See `this table` for further explanation). For this behavior, the `-f` argument must be replaced with the `-d` argument:

```
SNPMeta.py -d [DIRECTORY] -l [LENGTH] -a [EMAIL]
```

will annotate all SNPs in `[DIRECTORY]`. In this usage, each SNP must be contained in its own FASTA file. We also provide a quick `tutorial` for running SNPMeta on the provided example dataset.

Arguments

SNPMeta takes many optional arguments in addition to the required ones listed above. The full list of arguments can be seen by giving the `-h` switch on the command-line. These are detailed below.

Required Arguments

`-f [FASTA_FILE]`	A single FASTA-formatted file containing all SNPs to annotate. This argument is mutually exclusive with `-d`.
`-d [DIRECTORY]`	A directory containing FASTA-formatted files. Each SNP is contained within its own FASTA file. This argument is mutually exclusive with `-f`.
`-l [LENGTH]`	The length of the flanking sequence around the query SNP in each sequence. This is the length of only one side, not the combined length on both sides.
`-a [EMAIL]`	The email address that is sent to NCBI with batch queries.

Optional Arguments

`-h / --help`	Print a brief overview of usage and available options.
`-t / --target [TARGET]`	Annotate only from sequences originating from `[TARGET]`, where `[TARGET]` is the organism name defined in the GenBank record. This option may be specified multiple times to allow annotation from multiple specific organisms. For example, to annotate against either *Bos taurus* or *Bos indicus*, supply `-t 'Bos taurus' -t 'Bos indicus'`.
`-o [OUTPUT_FILE]`	Annotation information will be written to `[OUTPUT_FILE]`. It defaults to `stdout`.

`-q / --entrez-query [ENTREZ_QUERY]`	When specified, this argument will send `[ENTREZ_QUERY]` to the BLAST server. This can be used to narrow the BLAST results to certain taxa. For example, to narrow results to hits from grasses, use `-q 'Poaceae'`. This option is not compatible with using a local BLAST database.
`--no-blast`	BLAST has already been run on a collection of SNPs, and annotation is to be performed on pre-built XML reports. Assumes `-d` is given.
`-v / --verbose`	Verbose output. Prints annotation information as a tab-delimited table, with diagnostic messages. The default is to produce a dbSNP-like submission report.
`-p {blastn, tblastx}`	Specify which BLAST program is to be run for each SNP. Defaults to `blastn`.
`-e [EVALUE]`	Sets the maximum e-value (the score of the worst match) to be returned by BLAST. Defaults to `5e-10`.
`-m [MAX_HITS]`	Sets the maximum number of hits to be returned by BLAST. Defaults to `5`.
`-b [DATABASE]`	Run BLAST searches against a local database, `[DATABASE]`. This option cannot be used with `-q`.
`-g / --gbs`	SNP contextual sequences are from GBS sources. Specifying this causes SNPMeta to ignore anything entered for the `-l` option, and instead annotate on the first IUPAC ambiguity encountered in the sequence.
`-i / --illumina`	SNP contextual sequences are FASTA-formatted, but contain the SNP states enclosed by brackets `[]`, and separated by a slash `/`, eg. `[A/G]` for an A→G SNP.

Input Files

Input files must be plain text, and FASTA-formatted. If the `-f` option is given, then all SNP contextual sequences must appear in one multi-record FASTA file. If the `-d` option is given, then the directory must contain FASTA-formatted files, with only one record per file. With `-d`, each file should end in '`.fasta`' (case-sensitive).

For SNPs from an Illumina assay, each sequence in the FASTA file should be symmetrical, in terms of length, around the query SNP. This is not necessary for SNPs from GBS; SNPs can occur at arbitrary positions within the sequence. Finally, the SNP should be encoded as an IUPAC two-base ambiguity code, unless the `-i` switch is provided. For example, an A/G SNP from an Illumina assay with a contextual sequence length of 30 would look like this (query SNP shown in red):

SNP_1.fasta (Illumina IUPAC)

```
>SNP_1
TGACTGCCGAGGACGCAGTGCGCTATAGACRTTACAGATTCCGCCCTCAGATTATCTGAAC
```

The same SNP in the Illumina format would look like this:

SNP_1.fasta (Illumina)

```
>SNP_1
TGACTGCCGAGGACGCAGTGCGCTATAGAC[A/G]TTACAGATTCCGCCCTCAGATTATCTGAAC
```

We provide a script (`Convert_Illumina.py`) to convert the Illumina format into standard FASTA format. An A/G SNP from a GBS study might look like this (SNP shown in red):

SNP_1.fasta (GBS)

```
>SNP_1
ATACCACCCAGCCTCCGTGACCTCGAACAGCTACCTGACTATCGCCTTGGGATTCTGTACCTCRACAATAG
```

In some cases, it is possible that the contextual sequence will also contain an IUPAC ambiguity code. These cases should not pose problems to SNPMeta, as long as the sequence conforms to the above form. In other cases, the contextual sequence could be short on one side, or even short on both. These cases could potentially cause an erroneous annotation, especially when there are other IUPAC ambiguities in the contextual sequence. When the actual sequence length differs from the expected length, SNPMeta scans the sequence from left to right, and annotates from the first ambiguous base it encounters. It writes a message into the output for these SNPs.

If the `--no-blast` flag is supplied, SNPMeta also assumes that `-d` is specified. SNPMeta will search the given directory for files ending in '`.blast.xml`' (case sensitive). These files must be XML BLAST reports from NCBI's standalone BLAST application. Because the XML report doesn't necessarily store the complete query sequence, both the FASTA containing the SNP contextual sequence and the XML reports should be in the same directory. The SNP sequence and the corresponding XML report should have the same name, except for their extensions. That is, if there is an XML report `SNP_1.blast.xml`, then there must also be a corresponding FASTA file `SNP_1.fasta`.

Output

Default Output

SNPMeta's default output is a format that is almost ready for submission to dbSNP; all that is required is that the submitter-specific information be prepended to the report. The sections that need to be filled in by the submitter are the 'CONT', 'METHOD', 'POPULATION', and the 'SNPASSAY' sections. Instructions for filling in these sections can be found here. SNPMeta outputs for a SNP only report - no information on frequency is shown. The template for SNPMeta's output is given below.

Default SNPMeta Output Template

```
SNP:
GENENAME:
ACCESSION:
COMMENT:
SAMPLESIZE:
LENGTH: ?
5'_FLANK:
OBSERVED:
3'_FLANK:
||
```

Descriptions of the fields are given in the table below:

SNP	Name of the SNP, as recorded in the FASTA file.
GENENAME	Contains the name of the gene in which the SNP occurs, if applicable. This is also filled if the SNP is non-coding, but occurs near a named gene.
ACCESSION	The ID of the GenBank record that contains the SNP.
COMMENT	Contains coding state (non-coding, synonymous, nonsynonymous) and the amino acid states, if applicable. If amino acid states are listed, then the 'reference' state is listed first, followed by the alternate state.
SAMPLESIZE	Number of chromosomes sampled to identify the SNP. Must be filled in after annotation, as it is specific to each study.
LENGTH	Total length of the SNP sequence. NCBI automatically calculates the length, so this field is left as '?', as per the guidelines.

`5'_FLANK`	The contextual sequence that is upstream of the query SNP.
`OBSERVED`	The two alleles observed for the SNP, separated by a '`/`'. The 'reference' state is listed first.
`3'_FLANK`	The contextual sequence that is downstream of the SNP.

Note that the sum of the lengths of the data in the '`5'_FLANK`', '`OBSERVED`', and '`3'_FLANK`' fields must be greater than 100. The whole block of fields is repeated for each SNP. For example, one of the testing SNPs from *Drosophila melanogaster* (GBS) is shown below:

Example SNPMeta Output

```
SNP: RIL01_12001_39
GENENAME: CG4629
ACCESSION: AE014134
COMMENT: nonsynonymous I S
SAMPLESIZE:
LENGTH: ?
5'_FLANK: ATTCACTCCATTTTCCAGAAGAGAGCGGCAGGGACAGCGACGCCGGCTGTAGCGGCGCCA
OBSERVED: T/G
3'_FLANK: TGGCAGCGGCAAAAAGCCCAGTGAGAGAGCCGTCATTTGAGAGGTCGCCGGCGGAGCGCC
||
```

Verbose Output

When the `-v` or `--verbose` switch is given, SNPMeta will write tab-delimited text as output. Each row corresponds to a single SNP, and columns are various pieces of annotation information. Columns will be filled, depending on the amount of annotation information that is available for each SNP. If no data is available for a particular field, SNPMeta will write a '`-`' for that field. Some fields are mutually exclusive; that is, if one is filled, then the other will always be '`-`'. The table outputs 23 fields, which are explained below.

`SNPName`	The name of the SNP, taken from the FASTA record.
`Organism`	Organism from which the annotation was derived.
`GenBankID`	ID of the GenBank record used for annotation.
`ProteinID`	Protein ID of the CDS used for annotation, if applicable.

`GeneShortName`	Short name of the gene used for annotation, if applicable.
`Position`	Position of the SNP in the codon. Can take values of {`1, 2, 3`} if coding, and '`non-coding`' if not coding.
`Downstream`	How many bp downstream of the CDS the SNP lies, if non-coding.
`Upstream`	How many bp upstream of the CDS the SNP lies, if non-coding.
`Silent`	Whether or not the SNP alters amino acid sequence. Synonymous and non-coding SNPs get '`yes`', and nonsynonymous SNPs get '`no`'.
`AA1`	Amino acid state present in the GenBank record.
`AA2`	Alternate amino acid state. May be identical to AA1, if synonymous.
`GranthamScore`	Score of amino acid substitution, as defined by Grantham (1974) in *Science*.
`CDSPosition`	Position of the affected residue in the CDS.
`Codon1`	Codon triplet present in the GenBank sequence.
`Codon2`	Alternate codon triplet.
`AmbiguityCode`	IUPAC ambiguity code that describes the two SNP states.
`ProductName`	Name of the CDS product, if applicable.
`Notes*`	Messages and warnings about SNP files, or the nature of certain SNPs.
`RelatedGene`	Short name of a related gene, found by searching through the list of BLAST hits.
`RelatedOrganism`	Source organism for the `RelatedGene`.
`ContextSequence`	Warnings about malformed SNP sequences.
`AlignScore`	Alignment score reported by `needle`, divided by the query sequence length in base pairs. Useful for flagging SNPs that may be called incorrectly due to low-quality alignments.
`DateTime`	Date and time the SNP was annotated.

*: Will be described in detail in the following section

The 'Notes' Field

A variety of messages can be written into the Notes field. These will be elaborations on special classes of mutations, or messages about problems when annotating a particular SNP. The range of messages and their meanings are given below.

`Empty File`	The file SNPMeta tried to read contains no data.
`No BLAST Hits`	There were no BLAST hits for the sequence. If many SNPs have this message, it could be that either the list of target organisms is too restrictive, or there is a spelling error in the name of the organism.
`No Annotations`	The GenBank records returned by BLAST did not contain any annotated CDS.
`SNP aligns to a gap in the GenBank sequence`	When the GenBank sequence and the SNP contextual sequence are aligned, the SNP is placed over a gap in the GenBank sequence, making annotation impossible.
`SNP aligns to a gap in the CDS`	When the GenBank sequence and the SNP contextual sequence are aligned, the SNP is placed over a gap in the CDS. This could either be due to an alignment error, or a break in the annotated CDS.
`CDS annotation has a fuzzy start`	The CDS annotation in the GenBank record does not have an exact start position. This could cause issues with inferring reading frame.
`SNP - Sequence Mismatch`	The base in the GenBank sequence is not represented by the IUPAC ambiguity code in the SNP sequence.
`Methionine mutation`	The mutation changes an amino acid to a methionine, or from a methionine.
`Disrupted STOP codon`	The mutation changes a STOP codon to a different codon.
`Premature STOP codon`	The mutation creates a STOP codon before the end of the annotated CDS.

Tutorial

We provide example data from the *Drosophila melanogaster* SNPs used to test SNPMeta. These SNPs were constructed to have 60bp of sequence flanking each query SNP. To annotate the provided SNPs against the best match in GenBank, type

```
SNPMeta.py -f Example_SNPs.fasta -a 'name@domain.com' -l 60
```

, supplying a valid email address for the `-a` argument. The resulting annotations will be written to `stdout`. Supply the `-o` argument to have to write to a file instead:

```
SNPMeta.py -f Example_SNPs.fasta -a 'name@domain.com' -l 60 \
           -o SNPMeta_output.txt
```

To annotate the same SNPs against only *D. simulans*, it is possible to supply only the `-t` argument, but it gives better results to also supply the `-q` argument:

```
SNPMeta.py -f Example_SNPs.fasta -a 'name@domain.com' -l 60 \
           -q 'Drosophila simulans' -t 'Drosophila simulans'
```

`-q` is applied to the results returned by BLAST (it sends an Entrez query to NCBIs servers), while `-t` acts on the GenBank records once they have been downloaded. Supplying only `-t` would require that more GenBank records be downloaded in this example (by using the `-m` argument), as these sequences would primarily match records from *D. melanogaster*. Using only `-q` does not provide complete filtering of the results to *D. simulans.* By using `-q` and `-t`, SNPMeta annotates against *D. simulans* while downloading as few records as possible.

To annotate the SNPs with pre-built BLAST reports, first make a directory for the SNPs and their corresponding BLAST reports.

```
mkdir -p XML/
```

Then, run the companion script '`Split_FASTA.py`' to split the large FASTA into a series of smaller ones. Use '`1`' as the second argument so that the resulting FASTA files only contain one record each.

```
python Split_FASTA.py Example_SNPs.fasta 1
mv Example_SNPs_* XML
```

Now, to run BLAST on each of these SNPs, use the '`BLAST_SNPs.sh`' script provided. It requires that a local copy of the BLAST+ executables be available. Instructions for downloading and installing them are available here. Open the script in a text editor, and adjust the BLAST parameters as desired. Then, run the script:

```
sh BLAST_SNPs.sh XML/
```

This will fill the `XML/` directory with BLAST reports (in XML format) for each SNP. Then, to annotate the SNPs, supply both the `-d` and the `--no-blast` arguments.

```
SNPMeta.py -d XML/ -a 'name@domain.com' -l 60 --no-blast
```

Note that both '`BLAST_SNPs.sh`' and SNPMeta can run BLAST against a local copy of a BLAST database. Instructions for setting up a local copy can be found here. Currently, there is no support for running against a custom-built database, as we require GenBank records for annotation.