Developing a Python package

An overview of functionality and tools

Daniel Munro
Postdoc in PejLab at Scripps Research
and Palmer Lab at UC San Diego

Topics

Why make a Python package?

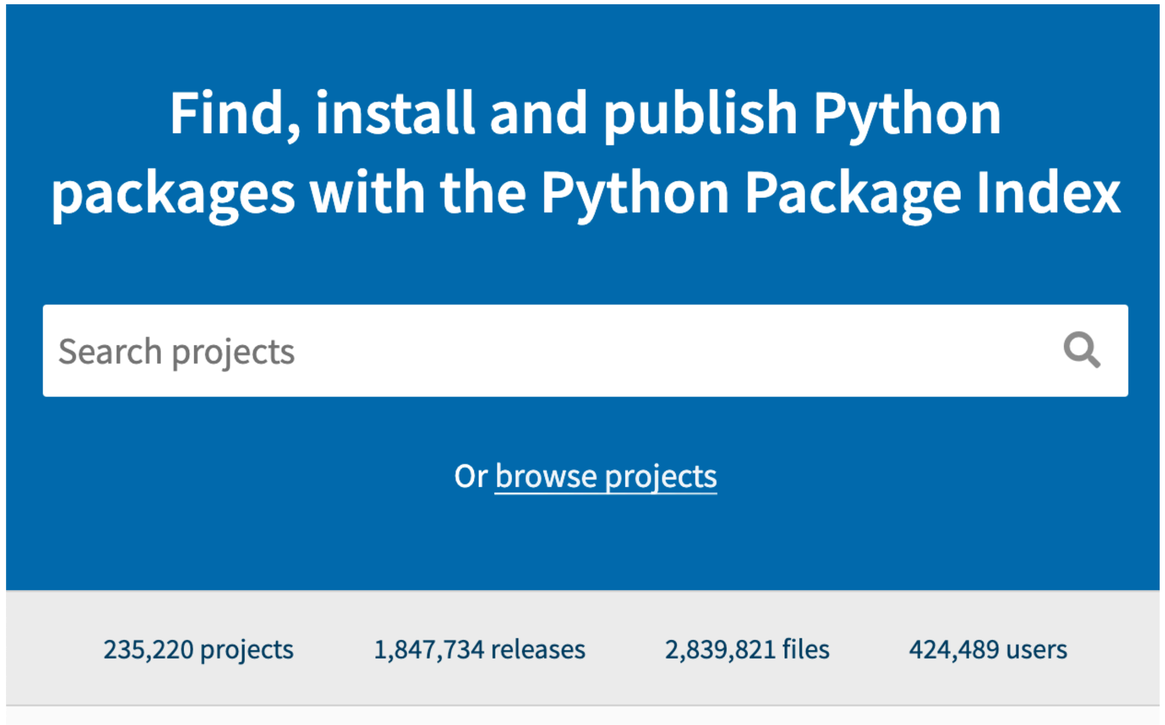
Command line vs. import

Testing

Documentation

Why Python package?

- Multiple options to distribute (GitHub, PyPI, etc.)
- Users can automatically install package and its dependencies
- Standardized structure for important parts of complex software
- Turning existing code into a package: not as much extra work as it may seem!



Your package can be imported...

in demask/homologs.py:

```python
def find_homologs(
    seqfile: str,
    blastp: str,
    db: str,
    threads: int = 1,
    evalue_cutoff: float = 0.01,
    bitscore_cutoff: float = None,
    nseqs: int = 300,
    outfile: str = None,
    outdir: str = None,
):
    """Find protein homologs using blastp.

    For each query sequence, an output file in a2m (FASTA) format will
    be produced containing the query sequence and the most similar
    homologs up to a specified number (default 300).

    Args:
```

user script:

```python
from demask.homologs import find_homologs

find_homologs("seq.fa", "blastp", "/path/to/db", outfile="homologs.a2m")
```

...or it can be run from the command line

in demask/homologs.py:

```
if __name__ == "__main__":
    args = parse_args()
    del args.config
    find_homologs(**vars(args))
```

```
dan@laptop ~> python3 -m demask.homologs -s seq.fa -o homologs.a2m
```

Tailor for scripts vs. command line.

e.g. function inputs and output are Python objects, but it can read/write files instead if run from command line:

```
if __name__ == "__main__":
    args = parse_args()
    outfile = args.outfile
    if outfile is None:
        inbase, inext = os.path.splitext(args.infile)
        outext = ".txt" if inext != ".txt" else "_demask.txt"
        outfile = inbase + outext
    del args.config, args.outfile
    predictions = run_demask(**vars(args))
    with open(outfile, "w") as out:
        out.write("\t".join(["pos", "WT", "var", "score"]) + "\n")
        for pred in predictions:
            out.write("{}\t{}\t{}\t{:.4f}\n".format(*pred))
```

Unit testing with unittest

```python
class TestProfiles(unittest.TestCase):
    def test_get_weights(self):
        seqs = read_fasta("test/P46937.a2m", as_dict=False)
        aln = seq_matrix(seqs)[:, 1:]
        wts = get_weights(aln)
        self.assertTrue(np.all(wts > 0) and np.all(wts < 1))

    def test_get_aligned_profile(self):
        # Ensure sum of each row of output array is > 0 and <= 1.
        seqs = read_fasta("test/P46937.a2m", as_dict=False)
        aln = seq_matrix(seqs)[:, 1:]
        wts = get_weights(aln)
        profile = get_aligned_profile(aln, wts)
        sums = np.sum(profile, axis=1)
        self.assertTrue(np.all(sums > 0) and np.all(sums <= 1.00))

    def test_gaps_in_query(self):
        # demask should accept gaps in query, e.g. for true MSA input.
        seqs1 = read_fasta("test/test_aln_ungapped.a2m", as_dict=False)
        aln1 = seq_matrix(seqs1)
        seqs2 = read_fasta("test/test_aln_gapped.a2m", as_dict=False)
        aln2 = seq_matrix(seqs2)
        self.assertTrue(aln1.shape == aln2.shape and np.all(aln1 == aln2))

    def test_no_homologs(self):
        seqs = read_fasta("test/random.a2m", as_dict=False)
        aln = seq_matrix(seqs)[:, 1:]
        wts = get_weights(aln)
        profile = get_aligned_profile(aln, wts)
        self.assertTrue(np.all(profile == 1/20))


class TestPredict(unittest.TestCase):
    def setUp(self):
        self.seqs = read_fasta("test/P46937.a2m", as_dict=False)
```

```
dan@laptop ~/d/DeMaSk (master)> python3 test.py
...........F.F..
======================================================================
FAIL: test_compute_scores_string (__main__.TestPredict)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test.py", line 129, in test_compute_scores_string
    self.assertEqual(len(predictions), len(self.seqs[0]) * 18)
AssertionError: 9576 != 9072

======================================================================
FAIL: test_get_aligned_profile (__main__.TestProfiles)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test.py", line 98, in test_get_aligned_profile
    self.assertTrue(np.all(sums > 0) and np.all(sums <= 1.00))
AssertionError: False is not true

----------------------------------------------------------------------
Ran 16 tests in 0.872s

FAILED (failures=2)
dan@laptop ~/d/DeMaSk (master) [1]>
```

```
dan@laptop ~/d/DeMaSk (master)> python3 test.py
................
----------------------------------------------------------------------
Ran 16 tests in 1.022s

OK
dan@laptop ~/d/DeMaSk (master)>
```

Command line help with argparse or configargparse

```python
def parse_args():
    p = configargparse.ArgParser(
        description=(
            "Predict fitness impact of all possible substitutions in "
            "a query protein."
        )
    )
    p.add(
        "-i",
        "--infile",
        required=True,
        help=(
            "Name of the file containing a sequence alignment in A2M (FASTA)
            "format, with the query protein as the first sequence."
        ),
    )
    p.add("-o", "--outfile", default=None, help="Name of new file to write sc
    dirname = os.path.dirname(__file__)
    config = os.path.join(dirname, "..", "config.ini")
    p.add(
        "-c",
        "--config",
        is_config_file=True,
        metavar="FILE",
        default=config,
        help=("Configuration file.  Defaults to 'config.ini' in the demask di
    )
    matrix = os.path.join(dirname, "..", "data", "matrix.txt")
    p.add(
        "-m",
        "--matrix",
        metavar="FILE",
        default=matrix,
        help=(
            "File containing the directional substitution matrix.  "
            "Defaults to the file at 'demask/data/matrix.txt'."
        ),
```

```
[dan@laptop ~> python3 -m demask.predict -h
usage: predict.py [-h] -i INFILE [-o OUTFILE] [-c FILE] [-m FILE]
                  [--coefs FILE] [-n NSEQS] [-w WEIGHT_THRESHOLD]

Predict fitness impact of all possible substitutions in a query protein. Args
that start with '--' (eg. -i) can also be set in a config file (specified via
-c). Config file syntax allows: key=value, flag=true, stuff=[a,b,c] (for
details, see syntax at https://goo.gl/R74nmi). If an arg is specified in more
than one place, then commandline values override config file values which
override defaults.

optional arguments:
  -h, --help            show this help message and exit
  -i INFILE, --infile INFILE
                        Name of the file containing a sequence alignment in
                        A2M (FASTA) format, with the query protein as the
                        first sequence.
  -o OUTFILE, --outfile OUTFILE
                        Name of new file to write scores to.
  -c FILE, --config FILE
                        Configuration file. Defaults to 'config.ini' in the
                        demask directory.
  -m FILE, --matrix FILE
                        File containing the directional substitution matrix.
                        Defaults to the file at 'demask/data/matrix.txt'.
  --coefs FILE          File containing the intercept, entropy, and identity
                        coefficients. Defaults to the file at
                        'demask/data/coefficients.txt'.
  -n NSEQS, --nseqs NSEQS
                        Maximum number of supporting sequences per query
                        sequence. Defaults to 300.
  -w WEIGHT_THRESHOLD, --weight_threshold WEIGHT_THRESHOLD
                        Sequence identity threshold used for sequence
                        weighting
```

Documentation with Sphinx

```python
def run_demask(
    infile: str,
    matrix: str,
    coefs: str,
    nseqs: int = 300,
    weight_threshold: float = 0.8,
) -> list:
    """Predict fitness for all substitutions in a query sequence.

    Args:
        infile: Name of the file containing a sequence alignment in
          A2M (FASTA) format, with the query protein as the first
          sequence.
        matrix: Name of the file containing the directional
          substitution matrix.
        coefs: Name of the file containing the intercept, entropy, and
          identity coefficients, with one name and value, separated by
          a tab, per line.
        nseqs: Maximum number of sequences in the alignment to use.
          Sequence weights can take a long time to compute for huge
          sequence counts.
        weight_threshold: Sequence identity threshold used for
          sequence weighting.  Sequences are weighted by the inverse
          of the number of sequences within this percent identity.

    Returns:
        A list of tuples, each containing the position, WT AA, variant
        AA, and score for each prediction.

    """
    seqs = read_fasta(infile, as_dict=False)
    seqs = seqs[: min(len(seqs), nseqs + 1)]  # +1 for query.
    aligned = seq_matrix(seqs)
    query = aligned[:, 0].T  # Query seq minus any gaps
```

demask.predict.run_demask(*infile, matrix, coefs, nseqs=300, weight_threshold=0.8*)

Predict fitness for all substitutions in a query sequence.

Parameters

- **infile** (`str`) – Name of the file containing a sequence alignment in A2M (FASTA) format, with the query protein as the first sequence.
- **matrix** (`str`) – Name of the file containing the directional substitution matrix.
- **coefs** (`str`) – Name of the file containing the intercept, entropy, and identity coefficients, with one name and value, separated by a tab, per line.
- **nseqs** (`int`) – Maximum number of sequences in the alignment to use. Sequence weights can take a long time to compute for huge sequence counts.
- **weight_threshold** (`float`) – Sequence identity threshold used for sequence weighting. Sequences are weighted by the inverse of the number of sequences within this percent identity.

Return type

`list`

Returns

A list of tuples, each containing the position, WT AA, variant AA, and score for each prediction.

Publishing documentation with Read the Docs

Branch: master ▾ algographics / docs /

Dan and Dan Moved some functions into extras subp

..	
_build	Moved some function
_static/png	Moved some function
API.rst	Moved some function
Makefile	Moved some function
conf.py	Moved some function
glossary.rst	Moved some function
guide.rst	Moved some function
index.rst	Moved some function

algographics.readthedocs.io/en/latest/guide.html

algographics

latest

Search docs

CONTENTS:

- User Guide
  - Components
  - Extras
  - Fills
  - Effects
- API
- Extras API
- Glossary

Support Read the Docs!

Please help keep us sustainable by allowing our Ethical Ads in your ad

Colors

Colors are represented as objects of the
HSL (hue, saturation, lightness) color sp

```
outline = ag.Circle(c=(200, 200), r=
color = ag.Color(
    hue=ag.Uniform(min=0.6, max=0.8)
)
x = ex.fill_spots(outline)
ag.set_styles(x, "fill", color)
```

Resources

- Testing Python code: https://docs.python-guide.org/writing/tests/
- Sphinx: https://www.sphinx-doc.org/
- Read the Docs: https://readthedocs.org/
- Documentation guide: https://docs.python-guide.org/writing/documentation/
- Python Package Index: https://pypi.org/