Extract TAIR gene IDs

Marta Kozlowska

Woodson Lab, University of Arizona, 2020

What is this program for?

gene_ids.py will filter out Arabidopsis ATnGnnnnn gene IDs from files filled with other information that is not necessary. It is useful for when you pull a list of genes from TAIR or AmiGO that is related to your search term. A gene list from TAIR looks something like this:

```
TAIR Accession  Locus   Gene Model  Gene Type   Description Other Name(Type)
Keywords    Is full length cDNA
Gene:1005715766 AT5G67030   AT5G67030.2 protein_coding  Encodes a single copy
zeaxanthin epoxidase gene that functions in first step of the biosynthesis of the
abiotic stress hormone abscisic acid (ABA). Mutants in this gene are unable to express
female sterility in response to beta-aminobutyric acid, as wild type plants do. ABA
DEFICIENT 1; ABA1; ARABIDOPSIS THALIANA ABA DEFICIENT 1; ARABIDOPSIS THALIANA
ZEAXANTHIN EPOXIDASE; ATABA1; ATZEP; IBS3; IMPAIRED IN BABA-INDUCED STERILITY 3; LOS6;
LOW EXPRESSION OF OSMOTIC STRESS-RESPONSIVE GENES 6; NON-PHOTOCHEMICAL QUENCHING 2;
NPQ2; ZEAXANTHIN EPOXIDASE; ZEP chloroplast, chloroplast envelope, FAD binding, LP.02
two leaves visible stage, LP.04 four leaves visible stage, LP.06 six leaves visible
stage, LP.08 eight leaves visible stage, LP.10 ten leaves visible stage, LP.12 twelve
leaves visible stage, abscisic acid biosynthetic process, carpel, cauline leaf,
chloroplast, chloroplast envelope, collective leaf structure, cotyledon, flower,
flower pedicel, flowering stage, guard cell, hypocotyl, inflorescence meristem, leaf
apex, leaf lamina base, mature plant embryo stage, membrane, petal, petal
differentiation and expansion stage, petiole, plant embryo, plant embryo bilateral
stage, plant embryo cotyledonary stage, plant embryo globular stage, plastid, pollen,
response to heat, response to osmotic stress, response to red light, response to water
deprivation, root, rosette leaf, seed, sepal, shoot apex, shoot system, stamen, stem,
sugar mediated signaling pathway, vascular leaf, vascular leaf senescent stage,
xanthophyll biosynthetic process, zeaxanthin epoxidase [overall] activity true
```

And a list of gene IDs you may get from AmiGO when you search for all genes related to a GO term looks like this:

```
AT5G12020|17.6 kDa class II heat shock protein|F14F18.190|F14F18_190
AT5G12020|17.6 kDa class II heat shock protein|F14F18.190|F14F18_190
AT5G41340|ATUBC4|ubiquitin conjugating enzyme 4|MYC6.5|MYC6_5
```

*NB: you will need to select only the "synonyms" aspect when fetching the list of genes from AmiGO.

The Program

`gene_ids.py` will extract TAIR gene IDs from a file and write those IDs to an outfile. The program will take in the following arguments:

- `-f|--file`: One or more files from which to extract the gene IDs (default: None)
- `-o|--outfile`: The output file to write the gene IDs to (default: out.txt)

When run with -h|--help the program gives the following:

```
$ ./gene_ids.py -h
usage: gene_ids.py [-h] -f FILE [FILE ...] [-o outfile]

Find TAIR locus ID

optional arguments:
  -h, --help            show this help message and exit
  -f FILE [FILE ...], --file FILE [FILE ...]
                        File to find TAIR gene IDs (default: None)
  -o outfile, --outfile outfile
                        Output file name (default: out.txt)
```

When run with the convoluted file of genes related to the `response to heat` GO term that I got from the TAIR website, the output will look like this:

```
$ ./gene_ids.py -f tair_heat.txt
  1: tair_heat.txt
Wrote 5 gene IDs from 1 file to file "out.txt"
```

The file will contain a list of gene IDs without all the clutter:

```
$ cat out.txt
AT1G13930
AT5G67030
AT2G22360
AT1G16540
AT3G09440
```

The program also runs on multiple files and gives back a combined list of gene IDs merged from both files:

```
$ ./gene_ids.py -f tair_heat.txt amigo_heat.txt
  1: tair_heat.txt
  2: amigo_heat.txt
Wrote 20 gene IDs from 2 files to file "out.txt"

$ cat out.txt
AT3G24500
AT5G67030
AT1G16540
AT1G13930
AT2G22360
AT3G06400
AT5G41340
AT1G64280
AT3G24520
AT5G03720
AT5G12020
AT2G33590
AT1G54050
AT3G10800
AT3G04120
AT5G12140
AT1G16030
AT3G09440
AT4G14690
AT4G19630
```

And if a file has repeat sequences, the program will only give one iteration of the gene ID:

```
$ ./gene_ids.py -f amigo_repeat.txt
  1: amigo_repeat.txt
Wrote 5 gene IDs from 1 file to file "out.txt"

$ cat out.txt
AT4G14690
AT5G03720
AT5G12020
AT2G22360
AT5G41340
```

You can change the name of the file the gene IDs are written to as well:

```
$ ./gene_ids.py -f tair_heat.txt -o hothothot.txt
  1: tair_heat.txt
Wrote 5 gene IDs from 1 file to file "hothothot.txt"
```

The output file will:

- have gene IDs listed
- have gene IDs each on a new line
- not have any repeat gene IDs

A passing test suite looks like this:

```
$ make test
pytest -xv test.py
============================== test session starts ===============================
...
collected 9 items

test.py::test_exists PASSED
[ 11%]
test.py::test_usage PASSED
[ 22%]
test.py::test_missing_file PASSED
[ 33%]
test.py::test_bad_file PASSED
[ 44%]
test.py::test_amigo PASSED
[ 55%]
test.py::test_tair PASSED
[ 66%]
test.py::test_two_files PASSED
[ 77%]
test.py::test_repeat_seq PASSED
[ 88%]
test.py::test_outfile PASSED
[100%]

============================== 9 passed in 0.61s ==================================
```