GFFUtils Documentation

Release 0.12.0

Peter Briggs

Nov 25, 2020

Contents

CHAPTER 1

Overview

The `GFFUtils` package provides a small set of utility programs for working with GFF and GTF files, specifically:

- `gff_cleaner`: perform "cleaning" operations on a GFF file
- `gff_annotation_extractor`: combine and annotate feature counts (e.g. from `htseq-count`) with data from a GFF file
- `gtf_extract`: extract selected data items from a GTF file
- `gtf2bed`: convert GTF file to BED format

Warning: The old names for the utilities (`GFFcleaner`, `GFF3_Annotation_Extractor` and `GTF_extract`) are still supported, but will be removed in a future release.

CHAPTER 2

Installation

To install the latest version of `GFFUtils`, download the latest release from as a `tar.gz` file from:

https://github.com/fls-bioinformatics-core/GFFUtils/releases

For example for version 0.10.3, download `GFFUtils-0.10.3.tar.gz`.

Unpack the code using:

```
tar xzf GFFUtils-0.10.3.tar.gz
```

which will unpack into a new directory called e.g. `GFFUtils-0.10.3`.

It is recommended to install the code into a Python virtual environment, which you can create by doing:

```
virtualenv venv
source venv/bin/activate
pip install -r ./GFFUtils-0.10.3/requirements.txt
pip install ./GFFUtils-0.10.3/
```

To install the developmental code directly from GitHub:

```
pip install -r https://raw.githubusercontent.com/fls-bioinformatics-core/GFFUtils/
↪devel/requirements.txt
pip install git+https://github.com/fls-bioinformatics-core/GFFUtils.git@devel
```

Note: `GFFUtils` is currently supported under the following Python versions:

- 2.7
- 3.5
- 3.6
- 3.7
- 3.8

but support for Python 2.7 is likely to be dropped in the near future.

CHAPTER 3

Contents

3.1 `gff_cleaner`: clean up GFF files

3.1.1 Overview

The `gff_cleaner` utility performs various manipulations on a GFF file to “clean” it.

3.1.2 Usage and options

General usage syntax:

```
gff_cleaner [OPTIONS] <file>.gff
```

Options:

--version
show program’s version number and exit

-h, --help
show the help message and exit

-o `OUTPUT_GFF`
Name of output GFF file (default is `<file>_clean.gff`)

--prepend`=PREPEND_STR`
String to prepend to seqname in first column

--clean
Perform all the ‘cleaning’ manipulations on the input data (equivalent to specifying all of `--clean-score`, `--clean-replace-attributes`, `--clean-exclude-attributes` and `--clean-group-sgds`)

--clean-score
Replace `Anc_*` and blanks in the `score` field with zeroes

--clean-replace-attributes
Replace `ID`, `Gene`, `Parent` and `Name` attributes with the value of the `SGD` attribute, if present

--clean-exclude-attributes
Remove the `kaks`, `kaks2` and `ncbi` attributes (to remove arbitrary attributes, see the `--remove-attribute=...` option)

--clean-group-sgds
Group features with the same `SGD` by adding unique numbers to the `ID` attributes; `ID``s will have the form ``CDS:<SGD>:<n>` (where `n` is a unique number for a given SGD)

--report-duplicates
Report duplicate `SGD` names and write list to `<file>_duplicates.gff` with line numbers, chromosome, start coordinate and strand.

--resolve-duplicates=MAPPING_FILE
Resolve duplicate `SGD``s by matching against 'best' genes in the supplied mapping file; other non-matching genes are discarded and written to ``<file>_discarded.gff`.

--discard-unresolved
Discard any unresolved duplicates, which are written to `<file>_unresolved.gff`.

--insert-missing=GENE_FILE
Insert genes from gene file with `SGD` names that don't appear in the input GFF. If `GENE_FILE` is blank ('='s must still be present) then the mapping file supplied with the `--resolve-duplicates` option will be used instead.

--add-exon-ids
For exon features without an `ID` attribute, construct and insert an ID of the form `exon:<Parent>:<n>` (where `n` is a unique number).

--add-missing-ids
For features without an `ID` attribute, construct and insert a generated ID of the form `<feature>:<Parent>:<n>` (where `n` is a unique number).

--no-percent-encoding
Convert encoded attributes to the correct characters in the output GFF.

This may result in a non-cannonical GFF that can't be read correctly by this or other programs.

--remove-attribute=RM_ATTR
Remove attribute `RM_ATTR` from the list of attributes for all records in the GFF file (can be specified multiple times)

--strict-attributes
Remove attributes that don't conform to the `KEY=VALUE` format

--debug
Print debugging information

3.1.3 Output files

- `<file>_clean.gff`: 'cleaned' version of input
- `<file>_duplicates.txt`: list of duplicated `SGD` names and the lines they appear on in the input file, along with chromosome, start coordinate and strand
- `<file>_discarded.gff`: genes rejected by `--resolve-duplicates`
- `<file>_unresolved.gff`: unresolved duplicates rejected by `--discard-unresolved`

3.1.4 Usage recipe

The following steps outline the procedure for using the program, with each step being run on the output from the previous one:

1. **Clean the chromosome names in the file by adding a prefix (‘–prepend‘ option)**

   Creates a copy of the input file with the chromosome names updated with a specified prefix.

   E.g. `--prepend=chr` will add `chr` to the start of each chromosome name in the file, which is useful if the chromosome is denoted by a number and needs the prefix for consistency with a mapping file.

2. **Clean the GFF score and attribute data (‘–clean‘ options)**

   The “clean” options perform the following operations:

   - `--clean-score`: the data in the score column is cleaned up by replacing `Anc_*` and blanks with ‘0’s.

   The attribute field of the GFF can contain various semicolon-separated key-value pairs:

   - `--clean-replace-attributes`: if one of these is a non-blank `SGD` then the `Gene`, `Parent` and `Name` values are updated to be the same as the `SGD` name.
   - `--clean-exclude-attributes`: attributes called `kaks`, `kaks2` and `ncbi` are removed (n.b. to remove arbitrary attributes, use the more general `--remove-attribute=...` option).

   If multiple features share the same SGD name then `--clean-replace-attributes` can result in them also sharing the same ID; to deal with this:

   - `--clean-group-sgds`: update the ID attribute to group neighbouring lines that have the same `SGD` (see *SGD grouping* below).

     A single *–clean* can be specified which performs all these operations automatically.

3. **Detect duplicate SGDs (‘–report-duplicates‘ option)**

   Report duplicate SGD names found in the input file.

   This option writes a list of the duplicates to a ‘duplicates’ file.

   It also reports the number of ‘trivial’ duplicates, i.e. lines having the same `SGD` because they are part of the same gene.

4. **Resolve duplicate SGDs using a mapping file (‘–resolve-duplicates‘ option)**

   Attempt to resolve duplicates by referring to a list of “best” genes given in a mapping file. For each duplicated name the resolution procedure is:

   - Find mapping gene(s) with the same name
   - For each mapping gene, keep duplicates which match chromosome, strand and which overlap with the start and end of the gene (see *Overlap criteria* below). For `SGD` groups the mapping gene must overlap the whole group for it to match; mapping genes and duplicates which don’t have matches are removed from the process.
   - At the end of the matching procedure the duplication is resolved if there is one `SGD` (or `SGD` group) matched to one mapping gene. Otherwise the duplication remains unresolved.

   When duplicates are resolved, the non-matching duplicates are discarded; otherwise by default all unresolved duplicates are kept. However if the `--discard-unresolved` option is also specified then all unresolved duplicates are removed before output; the `--insert-missing` option can then be used to add them back in.

   Note that the `--discard-unresolved` option cannot get rid of ‘trivial’ duplicates (i.e. lines having the same SGD because they are part of the same gene).

5. **Add missing genes ('–insert-missing' option)**

   Adds genes from a list of "best" genes given in a mapping file which have names not found in the input GFF.

3.1.5 SGD grouping

As part of setting the ID attribute of GFF lines, the "clean" option also attempts to group neighbouring lines which have the same SGD name.

The ID attribute is updated to the form:

```
ID=CDS:<sgd_name>:<i>
```

where <sgd_name> is a gene or transcript name (e.g. YEL0W) and <i> is an integer index which starts from 1. Groupings are indicated by subsequent lines having the same <sgd_name> but monotonically increasing indices, for example:

```
chr1    Test    CDS    34525    35262    0    -    0    ID=CDS:YEL0W:1;SGD=YEL0W
chr1    Test    CDS    35823    37004    0    -    0    ID=CDS:YEL0W:2;SGD=YEL0W
chr1    Test    CDS    38050    38120    0    -    0    ID=CDS:YEL0W:3;SGD=YEL0W
chr1    Test    CDS    39195    39569    0    -    0    ID=CDS:YEL0W:4;SGD=YEL0W
```

When determining a grouping the program looks ahead from each line for subsequent lines (up to five) which have the same SGD value. So groupings can also accommodate "breaks", for example:

```
chr1    Test    CDS    34525    35262    0    -    0    ID=CDS:YEL0W:1;SGD=YEL0W
chr1    Test    CDS    35823    37004    0    -    0    ID=CDS:YEL0W:2;SGD=YEL0W
chr1    Test    CDS    38050    38120    0    -    0    ID=CDS:YEL0X:1;SGD=YEL0X
chr1    Test    CDS    39195    39569    0    -    0    ID=CDS:YEL0W:3;SGD=YEL0W
```

3.1.6 Mapping file format

The mapping file is a tab-delimited text file with lines of the form:

```
name    chr    start    end    strand
```

<name> is used to match against the SGD names in the input GFF file.

3.1.7 Overlap criteria

Aside from matching chromosome and strand, one of the criteria for a mapping gene to match a duplicate from the GFF file is that the two must overlap.

An overlap is counted as the duplicate from the GFF having start/end positions such that it lies inside the start/end positions of the mapping gene extended by 1kb i.e. between `start - 1000` and `end + 1000`.

3.2 `gff_annotation_extractor`: annotate gene feature data

3.2.1 Overview

`gff_annotation_extractor` takes gene feature data (for example the output from one or more runs of the HTSeq-count program) and combines it with data about each feature's parent gene from a GFF file.

By default the program takes feature data from a single tab-delimited input file where the first column contains feature IDs, and outputs an updated copy of the file with data about the feature's parent feature and parent gene appended to each line.

In 'htseq-count' mode, one or more `htseq-count` output files should be provided as input; the program will write out the data about the feature's parent feature and parent gene appended with the counts from each input file.

By default feature IDs from the feature data files are matched to the first record in the input GFF where the `ID` attribute of that record is the same (a different attribute can be specified using the `-i` option). All records are considered regardless of the feature type, unless the `-t` option is used to restrict the records to just those with the specified feature type (this may be required in 'htseq-count' mode).

The parent gene is located by recursively looking up records where the `ID` attribute matches the `Parent` attribute, until a gene record is found.

Note: `gff_annotation_extractor` can also be used with GTF input, in which case the feature IDs are matched using the `gene_id` attribute by default. Only `gene` feature types are considered when using GTF data.

3.2.2 Usage and options

General usage syntax:

```
gff_annotation_extractor OPTIONS <file>.gff FEATURE_DATA
```

Usage in 'htseq-count' mode:

```
gff_annotation_extractor --htseq-count OPTIONS <file>.gff FEATURE_COUNTS [FEATURE_
↪COUNTS2 ...]
```

Options:

`--version`
show program's version number and exit

`-h, --help`
show the help message and exit

`-o` `OUT_FILE`
specify output file name

`-t` `FEATURE_TYPE`, **`--type`**`=FEATURE_TYPE`
restrict feature records to this type when matching features from input count files; if used in conjunction with `--htseq-count` then should be the same as that specified when running htseq-count (default: include all feature records)

`-i` `ID_ATTRIBUTE`, **`--id-attribute`**`=ID_ATTRIBUTE`
explicitly specify the name of the attribute to get the feature IDs from (defaults to `ID` for GFF input, `gene_id` for GTF input)

`--htseq-count`
htseq-count mode: input is one or more output `FEATURE_COUNT` files from the `htseq-count` program

3.2.3 'htseq-count' mode

To generate the feature count files using `htseq-count` do e.g.:

```
htseq-count --type=exon -i Parent <file>.gff <file>.sam
```

which returns counts of each exon against the name of that exon's parent.

`gff_annotation_extractor` should then be run using the same value for the `--type` option:

```
gff_annotation_extractor --htseq-count --type=exon <file>.gff <counts>.out
```

3.2.4 Output files

`gff_annotation_extractor` always produces a copy of the feature data annotated with data for each parent gene. By default this will be called `<basename>_annot.txt`; use the `-o` option to specify a different name.

The annotation consists of the following fields:

- `exon_parent`: ID for the parent feature
- `feature_type_exon_parent`: type for the parent feature
- `gene_ID`: ID for the gene the feature belongs to
- `gene_name`: name of the gene (from the `Name` attribute for GFF, or `gene_name` attribute for GTF)
- `chr`: chromosome of the gene
- `start`: start position of the gene
- `end`: end position of the gene
- `strand`: strand for the gene
- `gene_length`: gene length
- `locus`: string consisting of `<chr>:<start>-<end>`
- `description`: text from the gene's `description` attribute

In the default mode these fields are appended to each line from the input feature file; in 'htseq-count' mode each line in the annotation file consists of these fields, with the counts from each `htseq-count` file appended.

If a parent gene cannot be located for a feature then the annotation for that feature will be empty.

In 'htseq-count' mode an additonal file called `<basename>_annot_stats.txt` is also produced with the counts of "ambiguous", "two_low_aQual" etc from each log.

3.2.5 Warnings and errors

The following is a non-exhaustive list of the warnings and errors that `gff_annotation_extractor` can produce, along with a brief description and possible cause:

- `Unable to locate parent data for feature '...'`: indicates IDs in the feature files for which no matching records can be located in the input GFF. In this case the output annotation will be blank. Check that the input feature file consists of tab-delimited data.
- `Multiple parents found on line ...`: indicates that a record matching a feature ID has a `Parent` attribute which contains multiple comma-separated IDs. In this case it may not be possible to locate the parent gene for the feature.
- `No identifier attribute (...) on line ...`: indicates a record from the input GFF with no `ID` attribute (or custom attribute supplied via `-i` option).

- `No '...' attribute found on line ...:` indicates a record from the input GTF with no `gene_id` attribute (or custom attribute supplied via `-i` option).

3.3 `gtf_extract`: extract data items from GTF/GFF

3.3.1 Overview

The `gtf_extract` utility extracts selected data items from a GTF file and output in tab-delimited format.

Note: The program can also operate on GFF files provided the `--gff` option is specified.

3.3.2 Usage and options

General usage syntax:

```
gtf_extract OPTIONS <gft_file>
```

Options:

`--version`
show program's version number and exit

`-h, --help`
show the help message and exit

`-f` `FEATURE_TYPE`, **`--feature`**`=FEATURE_TYPE`
only extract data for lines where feature is FEATURE_TYPE

`--fields``=FIELD_LIST`
comma-separated list of fields to output in tab-delimited format for each line in the GTF, e.g. `chrom,start,end`.

Fields can either be a GTF field name (i.e. `chrom`, `source`, `feature`, `start`, `end`, `score`, `strand` and `frame`), or the name of an attribute (e.g. `gene_name`, `gene_id` etc).

Data items are output in the order they appear in `FIELD_LIST`. If a field doesn't exist for a line then `'.'` will be output as the value.

`-o` `OUTFILE`
write output to OUTFILE (default is to write to stdout)

`--gff`
specify that the input file is GFF rather than GTF format

3.3.3 Output

The program outputs a tab-delimited line of data for each matching line found in the input GTF file; the data items in the line are those specified by the `--fields` option (or else all data items, if no fields were specified).

For example, for `--fields=chrom,start,end,strand`, the GTF line:

```
chr1        HAVANA  gene    11869   14412   .       +       .       gene_id
↪"ENSG00000223972.4" ...
```

will produce the output:

```
chr1        11869   14412   +
```

By default the output of the program is written to stdout; use the `-o` option to direct the output to a named file instead.

3.4 `gtf2bed`: convert GTF contents to BED format

3.4.1 Overview

`gtf2bed` converts the contents of a GTF file to BED format, printing a single line for each `gene` entry in the input GTF.

3.4.2 Usage

General usage syntax:

```
gtf2bed FILE.gtf
```

3.4.3 Output

The program prints the BED file contents directly to stdout, for example:

```
Gnai3       3       108107280       108146146       -       gene
Pbsn        X       77837901        77853623        -       gene
Cdc45       16      18780447        18811987        -       gene
H19 7       142575529       142578143       -       gene
Scml2       X       161117193       161258213       +       gene
```

To sent this to a file use:

```
gtf2bed FILE.gtf > FILE.bed
```

If any problems are encountered then the program will print warning messages to stderr.

3.5 Extra scripts and utilities

3.5.1 run_cleanup.sh

`run_cleanup.sh` is a shell script which automatically runs `GFFcleaner` using all the steps outlined in the _Usage **recipe_**.

Usage:

```
run_cleanup.sh <gff_file> [<mapping_file>]
```

Note that duplicate resolution and missing gene insertion cannot be performed unless a mapping file is supplied.

CHAPTER 4

Version history

See the CHANGELOG.

CHAPTER 5

Additional information

See http://www.sanger.ac.uk/resources/software/gff/spec.html for details of the GFF format.

CHAPTER 6

Credits

These utilities have been developed by Peter Briggs with input from Leo Zeef, to support the activities of the Bioinformatics Core Facility (BCF) in the Faculty of Biology Medicine and Health (FBMH) at the University of Manchester (UoM).

Index