gffpandas Documentation

Release 1.0.0

Vivian Monzon, Konrad Foerstner

Sep 24, 2023

Contents

CHAPTER 1

Table of content

1.1 What is gffpandas?

gffpandas is a Python library, which can be used for annotation data. It facilitates the work with GFF3 files in regard to process them as to obtain desired annotation entries of them. Thereby, it is an easy to use and time-saving library.

This gffpandas library is an alternative to gffutils or bcbio-gff, but it is inspired by the Python library pandas. This means that the data frame structure is used to work with the annotation data, because gffpandas reads in the GFF3 file and makes a data frame out of it. With gffpandas it is possible to process a GFF3 file by different functions. One big advantage is that several functions can be combined so that the required annotation entries can be selected. Furthermore, the processed annotation data can be safed again as GFF3 file or as csv or tsv file.

To see the single functions see How to use gffpandas.

1.2 Background information

The Python library gffpandas facilitates working on generatic feature format version 3 (GFF3) files.

The GFF3 file contains location and attribute information about features, as e.g. genes, of DNA, RNA or protein sequences. It has one general format. This format includes a header, which is marked with a hash at the begin of the line. The header describes meta-data about the feature. The location and attribute information are described in nine columns, which are the following:

seq_id	source	type	start	end	score	strand	phase	attributes

1. **seq_id:** identification number of the sequence.
2. **source:** it gives information about how the annotation was generated. Normally, it is a database name or software name.
3. **type:** it describes the feature type, as e.g. gene, CDS, tRNA, exon etc.
4. **start:** it gives the start position of the feature [base pair (bp)].

5. **end:** it gives the end position of the feature [bp].
6. **score:** describes the score of the feature. It is written as a floating point number.
7. **strand:** gives the information, whether the feature is coded on the positive (+) or minus (-) strand. Otherwise, the strand can be '.' for features which are not stranded or '?' when the strand of the feature is unknown.
8. **phase:** The phase is required for all coding sequence (CDS)-features and gives the information about, at which position the CDS begins in the reading frame. It can be position 0, 1 or 2.
9. **attributes:** The attribute column is written in a 'tag=value' format and contains information about the following tags[1]: ID, Dbxref, gbkey, genome, genomic, mol_type, serovar, strain, Name, gene, locus_tag, Parent, Genbank, product, protein_id, transl_table

With the gffpandas library a GFF3 file can be processed as a pandas data frame. Different criteria (see How to use gffpandas) can be selected to obtain the desired entries of the data. Optional, the filtered data frame can be given back as GFF3, tsv or csv file.

So far, only GFF3 files containing only one GFF3 file, i.e. one header, can be used.

1.3 How to use gffpandas

The Python library gffpandas facilitates the work with GFF3 files. Thereby, different conditions can be choosen to process the annotation data, as e.g. to retain only the entries of a specific feature. The big advantages are that several functions and thus options, can be combined and that a GFF3 file or even csv or tsv file can be returned.
In this gffpandas version only files, which contain one GFF3 file, i.e. one header, can be used.
The given GFF3 file will be read by the pandas library and a data frame will be returned as instance variable of the class of gffpandas. Additionally, the header will be read and returned as another instance variable of this class. The data frame and header can be printed before or after filtering the annotation data by one or several functions. For printing the data frame or the header or even both, the suffix '.df' or rather '.header' has to be used.
In this tutorial it will be shown how to read in a GFF3 file, how to process the annnotation data and how to return again a GFF3 file by gffpandas. Additionally, all functions of gffpandas will be presented.

1.3.1 Example Tutorial:

The following GFF3 file will be used as example, to show how gffpandas has to be used. It contains a header and eleven annotation entries:

```
##gff-version 3
##sequence-region NC_016810.1 1 20
NC_016810.1   RefSeq  region  1       4000    .       +       .
↪Dbxref=taxon:216597;ID=id0;gbkey=Src;genome=genomic;mol_type=genomic DNA;
↪serovar=Typhimurium;strain=SL1344
NC_016810.1   RefSeq  gene    1       20      .       +       .       ID=gene1;
↪Name=thrL;gbkey=Gene;gene=thrL;locus_tag=SL1344_0001
NC_016810.1   RefSeq  CDS     13      235     .       +       0       Dbxref=UniProtKB
↪%252FTrEMBL:E1W7M4%2CGenbank:YP_005179941.1;ID=cds0;Name=YP_005179941.1;
↪Parent=gene1;gbkey=CDS;product=thr operon leader peptide;protein_id=YP_005179941.1;
↪transl_table=11
```

(continues on next page)

[1] https://github.com/The-Sequence-Ontology/Specifications/blob/master/gff3.md

(continued from previous page)

```
NC_016810.1   RefSeq  gene    1       20      .       +       .       ID=gene2;
↪Name=thrA;gbkey=Gene;gene=thrA;locus_tag=SL1344_0002
NC_016810.1   RefSeq  CDS     341     523     .       +       0       Dbxref=UniProtKB
↪%252FTrEMBL:E1W7M4%2CGenbank:YP_005179941.1;ID=cds0;Name=YP_005179941.1;
↪Parent=gene2;gbkey=CDS;product=thr operon leader peptide;protein_id=YP_005179941.1;
↪transl_table=11
NC_016810.1   RefSeq  gene    1       600     .       -       .       ID=gene3;
↪Name=thrX;gbkey=Gene;gene=thrX;locus_tag=SL1344_0003
NC_016810.1   RefSeq  CDS     21      345     .       -       0       Dbxref=UniProtKB
↪%252FTrEMBL:E1W7M4%2CGenbank:YP_005179941.1;ID=cds0;Name=YP_005179941.1;
↪Parent=gene3;gbkey=CDS;product=thr operon leader peptide;protein_id=YP_005179941.1;
↪transl_table=11
NC_016810.1   RefSeq  gene    41      255     .       +       .       ID=gene4;
↪Name=thrB;gbkey=Gene;gene=thrB;locus_tag=SL1344_0004
NC_016810.1   RefSeq  CDS     61      195     .       +       0       Dbxref=UniProtKB
↪%252FTrEMBL:E1W7M4%2CGenbank:YP_005179941.1;ID=cds0;Name=YP_005179941.1;
↪Parent=gene4;gbkey=CDS;product=thr operon leader peptide;protein_id=YP_005179941.1;
↪transl_table=11
NC_016810.1   RefSeq  gene    170     546     .       +       .       ID=gene5;
↪Name=thrC;gbkey=Gene;gene=thrC;locus_tag=SL1344_0005
NC_016810.1   RefSeq  CDS     34      335     .       +       0       Dbxref=UniProtKB
↪%252FTrEMBL:E1W7M4%2CGenbank:YP_005179941.1;ID=cds0;Name=YP_005179941.1;
↪Parent=gene5;gbkey=CDS;product=thr operon leader peptide;protein_id=YP_005179941.1;
↪transl_table=11
```

The library can be imported as the following:

```
import gffpandas.gffpandas as gffpd
```

First step is to read in the GFF3 file with the method called 'read_gff3'. Then a dataframe (.df) or rather header (.header) can be returned:

```
>>> annotation = gffpd.read_gff3('annotation.gff')
>>> print(annotation.header)
>>> print(annotation.df)

Out[1]:
##gff-version 3
##sequence-region NC_016810.1 1 20
         seq_id  source    type  start   end score strand phase  \
0   NC_016810.1  RefSeq  region      1  4000     .      +     .
1   NC_016810.1  RefSeq    gene      1    20     .      +     .
2   NC_016810.1  RefSeq     CDS     13   235     .      +     0
3   NC_016810.1  RefSeq    gene      1    20     .      +     .
4   NC_016810.1  RefSeq     CDS    341   523     .      +     0
5   NC_016810.1  RefSeq    gene      1   600     .      -     .
6   NC_016810.1  RefSeq     CDS     21   345     .      -     0
7   NC_016810.1  RefSeq    gene     41   255     .      +     .
8   NC_016810.1  RefSeq     CDS     61   195     .      +     0
9   NC_016810.1  RefSeq    gene    170   546     .      +     .
10  NC_016810.1  RefSeq     CDS     34   335     .      +     0

                                           attributes
0   Dbxref=taxon:216597;ID=id0;gbkey=Src;genome=ge...
1   ID=gene1;Name=thrL;gbkey=Gene;gene=thrL;locus_...
2   Dbxref=UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:Y...
```

(continues on next page)

(continued from previous page)

```
3   ID=gene2;Name=thrA;gbkey=Gene;gene=thrA;locus_...
4   Dbxref=UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:Y...
5   ID=gene3;Name=thrX;gbkey=Gene;gene=thrX;locus_...
6   Dbxref=UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:Y...
7   ID=gene4;Name=thrB;gbkey=Gene;gene=thrB;locus_...
8   Dbxref=UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:Y...
9   ID=gene5;Name=thrC;gbkey=Gene;gene=thrC;locus_...
10  Dbxref=UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:Y...
```

The created data frame contains all eleven annotation entries and can be changed now. Depending on which annotation entries are desired, different options of gffpandas can be used and/or combined.

In this example, the user wants to return a GFF3 file, but only its coding sequences ('CDS'), which base pair length (bp) is minimal 10 bp long and maximal 250 bp long. Therefore, the following functions will be combined:

```
>>> combined_df = annotation.filter_feature_of_type(['CDS']).filter_by_length(10,␣
↪250).to_gff3('temp.gff')
>>> gff_content = open('temp.gff').read()
>>> print(gff_content)

Out[2]:
##gff-version 3
##sequence-region NC_016810.1 1 20
NC_016810.1  RefSeq  CDS     13      235     .       +       0       Dbxref=UniProtKB
↪%252FTrEMBL:E1W7M4%2CGenbank:YP_005179941.1;ID=cds0;Name=YP_005179941.1;
↪Parent=gene1;gbkey=CDS;product=thr operon leader peptide;protein_id=YP_005179941.1;
↪transl_table=11
NC_016810.1  RefSeq  CDS     341     523     .       +       0       Dbxref=UniProtKB
↪%252FTrEMBL:E1W7M4%2CGenbank:YP_005179941.1;ID=cds0;Name=YP_005179941.1;
↪Parent=gene2;gbkey=CDS;product=thr operon leader peptide;protein_id=YP_005179941.1;
↪transl_table=11
NC_016810.1  RefSeq  CDS     61      195     .       +       0       Dbxref=UniProtKB
↪%252FTrEMBL:E1W7M4%2CGenbank:YP_005179941.1;ID=cds0;Name=YP_005179941.1;
↪Parent=gene4;gbkey=CDS;product=thr operon leader peptide;protein_id=YP_005179941.1;
↪transl_table=11
```

1.3.2 Methods included in gffpandas:

In this subsection, the possible functions of gffpandas will be presented.

filter_feature_of_type

For this method the requested feature-type has to be given as argument. A processed data frame will then be returned containing only the entries of the given feature-type.

For example:

```
>>> filtered_df = annotation.filter_feature_of_type(['gene'])
>>> print(filtered_df.df)

Out[2]:
        seq_id  source    type  start  end score strand phase  \
1  NC_016810.1  RefSeq    gene      1   20     .      +     .
```

(continues on next page)

(continued from previous page)

```
3  NC_016810.1  RefSeq    gene      1   20     .      +      .
5  NC_016810.1  RefSeq    gene      1  600     .      -      .
7  NC_016810.1  RefSeq    gene     41  255     .      +      .
9  NC_016810.1  RefSeq    gene    170  546     .      +      .

                                        attributes
1  ID=gene1;Name=thrL;gbkey=Gene;gene=thrL;locus_...
3  ID=gene2;Name=thrA;gbkey=Gene;gene=thrA;locus_...
5  ID=gene3;Name=thrX;gbkey=Gene;gene=thrX;locus_...
7  ID=gene4;Name=thrB;gbkey=Gene;gene=thrB;locus_...
9  ID=gene5;Name=thrC;gbkey=Gene;gene=thrC;locus_...
```

filter_by_length

For this method the required minimal and maximal bp-length have to be given. A processed data frame will then be returned with all entries within the given bp-length.

For example:

```
>>> filtered_by_length = annotation.filter_by_length(min_length=10, max_length=300)
>>> print(filtered_by_length.df)

Out[3]:
        seq_id  source    type  start  end score strand phase  \
1  NC_016810.1  RefSeq    gene      1   20     .      +      .
2  NC_016810.1  RefSeq     CDS     13  235     .      +      0
3  NC_016810.1  RefSeq    gene      1   20     .      +      .
4  NC_016810.1  RefSeq     CDS    341  523     .      +      0
7  NC_016810.1  RefSeq    gene     41  255     .      +      .
8  NC_016810.1  RefSeq     CDS     61  195     .      +      0

                                        attributes
1  ID=gene1;Name=thrL;gbkey=Gene;gene=thrL;locus_...
2  Dbxref=UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:Y...
3  ID=gene2;Name=thrA;gbkey=Gene;gene=thrA;locus_...
4  Dbxref=UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:Y...
7  ID=gene4;Name=thrB;gbkey=Gene;gene=thrB;locus_...
8  Dbxref=UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:Y...
```

get_feature_by_attribute

For this method the desired attribute tag as well as the corresponding value(s) have to be given. Therefore, the value name or several value names have to be given as list. A processed data frame will then be returned which contains the regarding attribute tag with the corresponding attribute value(s).

For example:

```
>>> feature_by_attribute = annotation.get_feature_by_attribute('gbkey', ['CDS'])
>>> print(feature_by_attribute.df)

Out[4]:
```

(continues on next page)

(continued from previous page)

```
         seq_id  source    type  start  end score strand phase  \
2   NC_016810.1  RefSeq     CDS     13  235     .      +     0
4   NC_016810.1  RefSeq     CDS    341  523     .      +     0
6   NC_016810.1  RefSeq     CDS     21  345     .      -     0
8   NC_016810.1  RefSeq     CDS     61  195     .      +     0
10  NC_016810.1  RefSeq     CDS     34  335     .      +     0

                                          attributes
2   Dbxref=UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:Y...
4   Dbxref=UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:Y...
6   Dbxref=UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:Y...
8   Dbxref=UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:Y...
10  Dbxref=UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:Y...
```

attributes_to_columns

This method splits the attribute column by the tags in seperate columns and returns a data frame. This method doesn't give an object file back. Therefore, it is not possible to combine it with other methods.

For example:

```
>>> attr_to_columns = annotation.attributes_to_columns()
>>> print(attr_to_columns)

Out[5]:
         seq_id  source    type  start   end score strand phase  \
0   NC_016810.1  RefSeq  region      1  4000     .      +     .
1   NC_016810.1  RefSeq    gene      1    20     .      +     .
2   NC_016810.1  RefSeq     CDS     13   235     .      +     0
3   NC_016810.1  RefSeq    gene      1    20     .      +     .
4   NC_016810.1  RefSeq     CDS    341   523     .      +     0
5   NC_016810.1  RefSeq    gene      1   600     .      -     .
6   NC_016810.1  RefSeq     CDS     21   345     .      -     0
7   NC_016810.1  RefSeq    gene     41   255     .      +     .
8   NC_016810.1  RefSeq     CDS     61   195     .      +     0
9   NC_016810.1  RefSeq    gene    170   546     .      +     .
10  NC_016810.1  RefSeq     CDS     34   335     .      +     0

                                          attributes  \
0   Dbxref=taxon:216597;ID=id0;gbkey=Src;genome=ge...
1   ID=gene1;Name=thrL;gbkey=Gene;gene=thrL;locus_...
2   Dbxref=UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:Y...
3   ID=gene2;Name=thrA;gbkey=Gene;gene=thrA;locus_...
4   Dbxref=UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:Y...
5   ID=gene3;Name=thrX;gbkey=Gene;gene=thrX;locus_...
6   Dbxref=UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:Y...
7   ID=gene4;Name=thrB;gbkey=Gene;gene=thrB;locus_...
8   Dbxref=UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:Y...
9   ID=gene5;Name=thrC;gbkey=Gene;gene=thrC;locus_...
10  Dbxref=UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:Y...

                                          Dbxref     ...      gbkey  \
0                                    taxon:216597     ...        Src
1                                            None     ...       Gene
```

(continues on next page)

(continued from previous page)

```
2   UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:YP_00517...     ...        CDS
3                                                 None     ...       Gene
4   UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:YP_00517...     ...        CDS
5                                                 None     ...       Gene
6   UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:YP_00517...     ...        CDS
7                                                 None     ...       Gene
8   UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:YP_00517...     ...        CDS
9                                                 None     ...       Gene
10  UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:YP_00517...     ...        CDS

    gene   genome    locus_tag     mol_type                    product  \
0   None  genomic         None  genomic DNA                       None
1   thrL     None  SL1344_0001         None                       None
2   None     None         None         None  thr operon leader peptide
3   thrA     None  SL1344_0002         None                       None
4   None     None         None         None  thr operon leader peptide
5   thrX     None  SL1344_0003         None                       None
6   None     None         None         None  thr operon leader peptide
7   thrB     None  SL1344_0004         None                       None
8   None     None         None         None  thr operon leader peptide
9   thrC     None  SL1344_0005         None                       None
10  None     None         None         None  thr operon leader peptide

        protein_id      serovar  strain transl_table
0             None  Typhimurium  SL1344         None
1             None         None    None         None
2   YP_005179941.1         None    None           11
3             None         None    None         None
4   YP_005179941.1         None    None           11
5             None         None    None         None
6   YP_005179941.1         None    None           11
7             None         None    None         None
8   YP_005179941.1         None    None           11
9             None         None    None         None
10  YP_005179941.1         None    None           11
```

overlaps_with

Here, a to comparable feature will be compared to all entries of the GFF3 file, to find out, with which entries it is overlapping. Therefore, the sequence id of this feature has to be given, as well as start and end position. Optional, its feature-type can be given as well as its strand-type (sense (+) or antisense (-)). By selecting 'complement=True', all the feature, which do not overlap with the to comparable feature will be returned.

For example:

```
>>> overlapings = annotation.overlaps_with(seq_id='NC_016811.1', type='gene',
                                           start=40, end=300, strand='+')
>>> no_overlap = annotation.overlaps_with(seq_id='NC_016811.1', start=1, end=4000,
                                          strand='+', complement=True)
>>> print(overlapings.df)
>>> print(no_overlap.df)

Out[6]:
         seq_id  source    type  start   end score strand phase  \
```

(continues on next page)

(continued from previous page)

```
0   NC_016810.1  RefSeq  region      1  4000     .      +     .
2   NC_016810.1  RefSeq     CDS     13   235     .      +     0
7   NC_016810.1  RefSeq    gene     41   255     .      +     .
8   NC_016810.1  RefSeq     CDS     61   195     .      +     0
9   NC_016810.1  RefSeq    gene    170   546     .      +     .
10  NC_016810.1  RefSeq     CDS     34   335     .      +     0

                                          attributes
0   Dbxref=taxon:216597;ID=id0;gbkey=Src;genome=ge...
2   Dbxref=UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:Y...
7   ID=gene4;Name=thrB;gbkey=Gene;gene=thrB;locus_...
8   Dbxref=UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:Y...
9   ID=gene5;Name=thrC;gbkey=Gene;gene=thrC;locus_...
10  Dbxref=UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:Y...

Out[7]:
Empty DataFrame
Columns: [seq_id, source, type, start, end, score, strand, phase, attributes]
Index: []
```

find_duplicated_entries

For this method the sequence id as well as the feature-type have to be given. Then all entries which are redundant according to start- and end-position as well as strand-type will be returned.

For example:

```
>>> redundant_entries = annotation.find_duplicated_entries(seq_id='NC_016811.1', type=
↪'gene')
>>> print(redundant_entries.df)

Out[8]:
        seq_id  source    type  start  end score strand phase  \
3  NC_016810.1  RefSeq    gene      1   20     .      +     .

                                       attributes
3  ID=gene2;Name=thrA;gbkey=Gene;gene=thrA;locus_...
```

The following methods of the library won't return a data frame:

to_gff3

With this method the header and the data frame will be safed as GFF3 file. This GFF3 file will be the original file, unless it was changed by other methods of gffpandas. The desired name of the outcome GFF3 file has to be given as argument.

For example:

```
>>> annotation.to_gff3('temp.gff')
>>> gff3_file = open('temp.gff').read()
>>> print(gff3_file)
```

(continues on next page)

(continued from previous page)

```
Out[9]:
##gff-version 3
##sequence-region NC_016810.1 1 20
NC_016810.1  RefSeq  region  1       4000    .       +       .
↪Dbxref=taxon:216597;ID=id0;gbkey=Src;genome=genomic;mol_type=genomic DNA;
↪serovar=Typhimurium;strain=SL1344
NC_016810.1  RefSeq  gene    1       20      .       +       .       ID=gene1;
↪Name=thrL;gbkey=Gene;gene=thrL;locus_tag=SL1344_0001
NC_016810.1  RefSeq  CDS     13      235     .       +       0       Dbxref=UniProtKB
↪%252FTrEMBL:E1W7M4%2CGenbank:YP_005179941.1;ID=cds0;Name=YP_005179941.1;
↪Parent=gene1;gbkey=CDS;product=thr operon leader peptide;protein_id=YP_005179941.1;
↪transl_table=11
NC_016810.1  RefSeq  gene    1       20      .       +       .       ID=gene2;
↪Name=thrA;gbkey=Gene;gene=thrA;locus_tag=SL1344_0002
NC_016810.1  RefSeq  CDS     341     523     .       +       0       Dbxref=UniProtKB
↪%252FTrEMBL:E1W7M4%2CGenbank:YP_005179941.1;ID=cds0;Name=YP_005179941.1;
↪Parent=gene2;gbkey=CDS;product=thr operon leader peptide;protein_id=YP_005179941.1;
↪transl_table=11
NC_016810.1  RefSeq  gene    1       600     .       -       .       ID=gene3;
↪Name=thrX;gbkey=Gene;gene=thrX;locus_tag=SL1344_0003
NC_016810.1  RefSeq  CDS     21      345     .       -       0       Dbxref=UniProtKB
↪%252FTrEMBL:E1W7M4%2CGenbank:YP_005179941.1;ID=cds0;Name=YP_005179941.1;
↪Parent=gene3;gbkey=CDS;product=thr operon leader peptide;protein_id=YP_005179941.1;
↪transl_table=11
NC_016810.1  RefSeq  gene    41      255     .       +       .       ID=gene4;
↪Name=thrB;gbkey=Gene;gene=thrB;locus_tag=SL1344_0004
NC_016810.1  RefSeq  CDS     61      195     .       +       0       Dbxref=UniProtKB
↪%252FTrEMBL:E1W7M4%2CGenbank:YP_005179941.1;ID=cds0;Name=YP_005179941.1;
↪Parent=gene4;gbkey=CDS;product=thr operon leader peptide;protein_id=YP_005179941.1;
↪transl_table=11
NC_016810.1  RefSeq  gene    170     546     .       +       .       ID=gene5;
↪Name=thrC;gbkey=Gene;gene=thrC;locus_tag=SL1344_0005
NC_016810.1  RefSeq  CDS     34      335     .       +       0       Dbxref=UniProtKB
↪%252FTrEMBL:E1W7M4%2CGenbank:YP_005179941.1;ID=cds0;Name=YP_005179941.1;
↪Parent=gene5;gbkey=CDS;product=thr operon leader peptide;protein_id=YP_005179941.1;
↪transl_table=11
```

to_csv

By this method, the data frame will be safed as csv file. The csv file can contain the entries of the original data frame or if it was changed, then the filtered entries. The desired name of the outcome csv file has to be given as argument.

For example:

```
>>> annotation.to_csv('temp.csv')
>>> csv_file = open('temp.csv').read()
>>> print(csv_file)

Out[9]:
seq_id,source,type,start,end,score,strand,phase,attributes
NC_016810.1,RefSeq,region,1,4000,.,+,.,Dbxref=taxon:216597;ID=id0;gbkey=Src;
↪genome=genomic;mol_type=genomic DNA;serovar=Typhimurium;strain=SL1344
NC_016810.1,RefSeq,gene,1,20,.,+,.,ID=gene1;Name=thrL;gbkey=Gene;gene=thrL;locus_
↪tag=SL1344_0001
```

(continues on next page)

(continued from previous page)

```
NC_016810.1,RefSeq,CDS,13,235,.,+,0,Dbxref=UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:YP_
↪005179941.1;ID=cds0;Name=YP_005179941.1;Parent=gene1;gbkey=CDS;product=thr operon␣
↪leader peptide;protein_id=YP_005179941.1;transl_table=11
NC_016810.1,RefSeq,gene,1,20,.,+,.,ID=gene2;Name=thrA;gbkey=Gene;gene=thrA;locus_
↪tag=SL1344_0002
NC_016810.1,RefSeq,CDS,341,523,.,+,0,Dbxref=UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:YP_
↪005179941.1;ID=cds0;Name=YP_005179941.1;Parent=gene2;gbkey=CDS;product=thr operon␣
↪leader peptide;protein_id=YP_005179941.1;transl_table=11
NC_016810.1,RefSeq,gene,1,600,.,-,.,ID=gene3;Name=thrX;gbkey=Gene;gene=thrX;locus_
↪tag=SL1344_0003
NC_016810.1,RefSeq,CDS,21,345,.,-,0,Dbxref=UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:YP_
↪005179941.1;ID=cds0;Name=YP_005179941.1;Parent=gene3;gbkey=CDS;product=thr operon␣
↪leader peptide;protein_id=YP_005179941.1;transl_table=11
NC_016810.1,RefSeq,gene,41,255,.,+,.,ID=gene4;Name=thrB;gbkey=Gene;gene=thrB;locus_
↪tag=SL1344_0004
NC_016810.1,RefSeq,CDS,61,195,.,+,0,Dbxref=UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:YP_
↪005179941.1;ID=cds0;Name=YP_005179941.1;Parent=gene4;gbkey=CDS;product=thr operon␣
↪leader peptide;protein_id=YP_005179941.1;transl_table=11
NC_016810.1,RefSeq,gene,170,546,.,+,.,ID=gene5;Name=thrC;gbkey=Gene;gene=thrC;locus_
↪tag=SL1344_0005
NC_016810.1,RefSeq,CDS,34,335,.,+,0,Dbxref=UniProtKB%252FTrEMBL:E1W7M4%2CGenbank:YP_
↪005179941.1;ID=cds0;Name=YP_005179941.1;Parent=gene5;gbkey=CDS;product=thr operon␣
↪leader peptide;protein_id=YP_005179941.1;transl_table=11
```

to_tsv

By this method, the data frame will be safed as tsv file. The tsv file can contain the entries of the original data frame or if it was changed, then the filtered entries. The desired name of the outcome tsv file has to be given as argument.

For example:

```
>>> annotation.to_tsv('temp.tsv')
>>> tsv_file = open('temp.tsv').read()
>>> print(tsv_file)

Out[10]:
seq_id       source  type    start   end     score   strand  phase   attributes
NC_016810.1  RefSeq  region  1       4000    .       +       .       ␣
↪Dbxref=taxon:216597;ID=id0;gbkey=Src;genome=genomic;mol_type=genomic DNA;
↪serovar=Typhimurium;strain=SL1344
NC_016810.1  RefSeq  gene    1       20      .       +       .       ID=gene1;
↪Name=thrL;gbkey=Gene;gene=thrL;locus_tag=SL1344_0001
NC_016810.1  RefSeq  CDS     13      235     .       +       0       Dbxref=UniProtKB
↪%252FTrEMBL:E1W7M4%2CGenbank:YP_005179941.1;ID=cds0;Name=YP_005179941.1;
↪Parent=gene1;gbkey=CDS;product=thr operon leader peptide;protein_id=YP_005179941.1;
↪transl_table=11
NC_016810.1  RefSeq  gene    1       20      .       +       .       ID=gene2;
↪Name=thrA;gbkey=Gene;gene=thrA;locus_tag=SL1344_0002
NC_016810.1  RefSeq  CDS     341     523     .       +       0       Dbxref=UniProtKB
↪%252FTrEMBL:E1W7M4%2CGenbank:YP_005179941.1;ID=cds0;Name=YP_005179941.1;
↪Parent=gene2;gbkey=CDS;product=thr operon leader peptide;protein_id=YP_005179941.1;
↪transl_table=11
NC_016810.1  RefSeq  gene    1       600     .       -       .       ID=gene3;
↪Name=thrX;gbkey=Gene;gene=thrX;locus_tag=SL1344_0003
```

(continues on next page)

(continued from previous page)

```
NC_016810.1  RefSeq  CDS     21      345     .       -       0       Dbxref=UniProtKB
↪%252FTrEMBL:E1W7M4%2CGenbank:YP_005179941.1;ID=cds0;Name=YP_005179941.1;
↪Parent=gene3;gbkey=CDS;product=thr operon leader peptide;protein_id=YP_005179941.1;
↪transl_table=11
NC_016810.1  RefSeq  gene    41      255     .       +       .       ID=gene4;
↪Name=thrB;gbkey=Gene;gene=thrB;locus_tag=SL1344_0004
NC_016810.1  RefSeq  CDS     61      195     .       +       0       Dbxref=UniProtKB
↪%252FTrEMBL:E1W7M4%2CGenbank:YP_005179941.1;ID=cds0;Name=YP_005179941.1;
↪Parent=gene4;gbkey=CDS;product=thr operon leader peptide;protein_id=YP_005179941.1;
↪transl_table=11
NC_016810.1  RefSeq  gene    170     546     .       +       .       ID=gene5;
↪Name=thrC;gbkey=Gene;gene=thrC;locus_tag=SL1344_0005
NC_016810.1  RefSeq  CDS     34      335     .       +       0       Dbxref=UniProtKB
↪%252FTrEMBL:E1W7M4%2CGenbank:YP_005179941.1;ID=cds0;Name=YP_005179941.1;
↪Parent=gene5;gbkey=CDS;product=thr operon leader peptide;protein_id=YP_005179941.1;
↪transl_table=11
```

stats_dic

Gives the following statistics for the entries of the original or changed data frame: The maximal and minimal bp-length, the number of sense (+) and antisense (-) strands as well as the number of each available feature-type.

For example:

```
>>> statistics = annotation.stats_dic()
>>> print(statistics.df)

Out[11]:
{'Maximal_bp_length': 599, 'Minimal_bp_length': 19, 'Counted_strands': +    9
-    2
Name: strand, dtype: int64, 'Counted_feature_types': gene      5
CDS       5
region    1
Name: type, dtype: int64}
```

1.4 gffpandas API

1.4.1 gffpandas.gffpandas module

class gffpandas.gffpandas.**Gff3DataFrame**(*input_gff_file=None*, *input_df=None*, *input_header=None*)

Bases: object

This class contains header information in the header attribute and a actual annotation data in the pandas dataframe in the df attribute.

attributes_to_columns() → pandas.core.frame.DataFrame

Saving each attribute-tag to a single column.

Attribute column will be split by the tags in the single columns. For this method only a pandas DataFrame and not a Gff3DataFrame will be returned. Therefore, this data frame can not be saved as gff3 file.

Returns pandas dataframe, whereby the attribute column of the gff3 file are splitted into the different attribute tags

Return type pandas DataFrame

filter_by_length(*min_length=None*, *max_length=None*) → gffpandas.gffpandas.Gff3DataFrame
Filtering the pandas dataframe by the gene_length.

For this method the desired minimal and maximal bp length have to be given.

Parameters

- **min_length** (*int*) – minimal bp length of the feature
- **max_length** (*int*) – maximal bp length of the feature

Returns original header and dataframe with features, whose lengths fits the set parameters, saved as object of the class Gff3DataFrame

Return type class 'gffpandas.gffpandas.Gff3DataFrame'

filter_feature_of_type(*feature_type_list*) → gffpandas.gffpandas.Gff3DataFrame
Filtering the pandas dataframe by feature_type.

For this method a list of feature-type(s) has to be given, as e.g. ['CDS', 'ncRNA'].

Parameters **feature_type_list** (*list*) – List of name(s) of the desired feature(s)

Returns original header and dataframe of the selected features saved as object of the class Gff3DataFrame

Return type class 'gffpandas.gffpandas.Gff3DataFrame'

find_duplicated_entries(*seq_id=None*, *type=None*) → gffpandas.gffpandas.Gff3DataFrame
Find entries which are redundant.

For this method the chromosom accession number (seq_id) as well as the feature-type have to be given. Then all entries which are redundant according to start- and end-position as well as strand-type will be found.

Parameters

- **seq_id** (*str*) – corresponding accession number
- **type** (*str*) – feature type

Returns original header and dataframe containing the duplicated entries, both saved as object of the class Gff3DataFrame

Return type class 'gffpandas.gffpandas.Gff3DataFrame'

get_feature_by_attribute(*attr_tag*, *attr_value_list*) → gffpandas.gffpandas.Gff3DataFrame
Filtering the pandas dataframe by a attribute.

The 9th column of a gff3-file contains the list of feature attributes in a tag=value format. For this method the desired attribute tag as well as the corresponding value have to be given. If the value is not available an empty dataframe would be returned.

Parameters

- **attr_tag** (*str*) – Name of attribute tag, by which the df will be filtered
- **attr_value_list** (*list*) – List of value name or several value names, which has/have to be associated with the attribute tag. If an entry includes the value with the corresponding tag it is selected

Returns original header and dataframe with the entries, which contain the desired attribute values, both saved as object of the class Gff3DataFrame

Return type class 'gffpandas.gffpandas.Gff3DataFrame'

overlaps_with(*seq_id=None*, *start=None*, *end=None*, *type=None*, *strand=None*, *complement=False*) → gffpandas.gffpandas.Gff3DataFrame

To see which entries overlap with a comparable feature.

For this method the chromosom accession number has to be given. The start and end bp position for the to comparable feature have to be given, as well as optional the feature-type of it and if it is on the sense (+) or antisense (-) strand.

Possible overlaps (see code):

```
-----==================----------
---------=====================-----

-----+++++++++++++++++++----------
-----+++++++++++++++++++----------

----++++++++++++++++++++++++-------
----++++++++++++----------------

------++++++++++++++++++++++-------
----------++++++++++++-------
```

By selecting 'complement=True', all the feature, which do not overlap with the to comparable feature will be returned.

Parameters

- **seq_id** (*str*) – accession number of the feature
- **start** (*int*) – start position of the feature
- **end** (*int*) – end position of the feature
- **type** (*str*) – type of the feature
- **strand** (*str*) – minus (-) for antisense and plus (+) for sense strand

Returns original header and dataframe, containing the entries which overlap or do not overlap (complement=True) with the given parameters, both saved as object of the class Gff3DataFrame

Return type class 'gffpandas.gffpandas.Gff3DataFrame'

stats_dic() → dict

Gives the following statistics for the data:

The maximal bp-length, minimal bp-length, the count of sense (+) and antisense (-) strands as well as the count of each available feature.

Returns information about the given dataframe, which are the length of the longest and shortest feature entry (in bp), the number of feature on the sense and antisense strand and the number of different feature types.

Return type dictionary

to_csv(*output_file=None*) → None
Create a csv file.

The pandas data frame is saved as a csv file.

Parameters **output_file** (*str*) – Desired name of the output csv file

Returns csv file with the content of the dataframe

Return type data file in csv format

to_gff3(*gff_file*) → None
Create a gff3 file.

The pandas dataframe is saved as a gff3 file.

Parameters **gff_file** (*str*) – Desired name of the output gff file

Returns gff3 file with the content of the dataframe

Return type data file in gff3 format

to_tsv(*output_file=None*) → None
Create a tsv file.

The pandas data frame is saved as a tsv file.

Parameters **output_file** (*str*) – Desired name of the output tsv file

Returns tsv file with the content of the dataframe

Return type data file in tsv format

gffpandas.gffpandas.**read_gff3**(*input_file*)

1.5 Requirements and installation

1.5.1 Requirements:

The Python library gffpandas was developed with Python 3. Thus, the user is advised to run gffpandas on Python 3.4 or a higher version. For an easy installation pip and setuptools should be installed. gffpandas is dependent on the Python library pandas, which needs to be installed, when using gffpandas.

1.5.2 Installation:

gffpandas is hosted on the PyPI server and can thus be installed by pip3:

```
$ pip3 install gffpandas
```

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

g

Index