bedparse Documentation

Release v0.2.0

Tommaso Leonardi

Jan 23, 2020

Contents

Bedparse is a simple python module and CLI tool to perform common operations on BED files.

It offers the following functionality:

- BED format validation
- Filtering of transcripts based on annotations
- Joining of annotation files based on transcript names
- Conversion from GTF to BED format
- Conversion from UCSC to Ensembl chromosome names (and viceversa)
- Conversion from bed12 to bed6
- Promoter reporting
- Intron reporting
- CDS reporting
- UTR reporting

CHAPTER 1

Contents

1.1 Installation

Bedparse is distributed on PyPI. To install, just run:

```
pip install bedparse
```

Alternatively, to install it from the Github repository:

```
pip install git+https://github.com/tleonardi/bedparse.git
```

1.2 Motivation

The BED (Browser Extensible Data) format is a plain text file format commonly used in bioinformatics to represent genomic features (e.g. genes, transcripts, peaks, regulatory regions, etc.). Each line in the file represents a genomic feature and consists of up to 12 tab-separated fields:

1. chromosome name
2. start coordinate in the chromosome
3. end coordinate in the chromosome
4. feature name
5. feature score
6. strand
7. thick start (conventionally the start codon for protein coding transcripts)
8. thick end (conventionally the stop codon for protein coding transcripts)
9. rgb color for visualisation in genome browsers
10. number of connected blocks (conventionally the number of exons)

11. comma separated list of blocks size
12. comma separated list of block starts relative to field 2 (i.e. genomic start of the feature)

One of the major advantages of the BED format over many of its alternatives is that each line includes all the information required to define an individual transcript. This characteristic allows to perform numerous operations on BED a file as part of unix pipes, for example using GNU awk.

For example, the following is a common approach to extract gene promoters (here defined as 500bp around the gene start):

```
awk 'BEGIN{OFS=FS="\t"}{print $1,$2-500,$3+500,$4,$5}' transcritpome.bed > promoters.
↪bed
```

However, these one-liners can quickly get long and hard to read. For example, if we wanted to do the same as before but keeping the strand into considerations:

```
awk 'BEGIN{OFS=FS="\t"}{if($6=="+"){print $1,$2-500,$2+500,$4,$5}else{print $1,$3-500,
↪$3+500,$4,$5}}' transcritpome.bed > promoters_stranded.bed
```

These and other more complex operations quicly get long to type and prone to errors and typos. Bedparse greatly simplifies the process:

```
bedparse promoter transcritpome.bed > promoters_stranded.bed
```

or:

```
bedparse promoter --unstranded transcritpome.bed > promoters.bed
```

Despite the simplicity of most of its operations, all functions in bedparse are thouroughly and rigourously tested through an automated test suit to ensure the accuracy and correctness of the results. Additionally, bedparse performs syntax validation checks on the input BED files and warns the user in case of malformed or unsupported formats.

Additionally, bedparse also provides two format conversion operations:

- gtf2bed allows converting Ensembl/Gencode Gene transfer format (GTF) files into bed format
- convertChr implements an internal dictionary that allows conversion of human and mouse chromosome names between the two most widely used formats, i.e. the Ensembl and the UCSC naming schemes.

1.3 Usage

```
usage: bedparse [-h] [--version]
                {3pUTR,5pUTR,cds,promoter,introns,filter,join,gtf2bed,bed12tobed6,
↪convertChr,validateFormat}
                ...

Perform various simple operations on BED files.

positional arguments:
  {3pUTR,5pUTR,cds,promoter,introns,filter,join,gtf2bed,bed12tobed6,convertChr,
↪validateFormat}
                        sub-command help
    3pUTR               Prints the 3' of coding genes.
    5pUTR               Prints the 5' of coding genes.
    cds                 Prints the CDS of coding genes.
    promoter            Prints the promoters of transcripts.
```

(continues on next page)

(continued from previous page)

```
    introns             Prints BED records corresponding to the introns of
                        each transcript in the original file.
    filter              Filters a BED file based on an annotation.
    join                Joins a BED file with an annotation file using the BED
                        name (col4) as the joining key.
    gtf2bed             Converts a GTF file to BED12 format.
    bed12tobed6         Converts a BED12 file to BED6 format
    convertChr          Convert chromosome names between UCSC and Ensembl
                        formats
    validateFormat      Check whether the BED file adheres to the BED format
                        specifications

optional arguments:
  -h, --help            show this help message and exit
  --version, -v         show program's version number and exit
```

The basic syntax in the form: `bedparse sub-command [parameters]`.

For a list of all sub-commands and a brief explanation of what they do, use: `bedparse --help`

For a detailed explanation of each subcommand and a list of its parameters, use the `--help` option after the subcommand's name, e.g.: `bedparse promoter --help`

1.3.1 3'/5' UTRs

Usage

```
> bedparse 3pUTR --help
usage: bedparse 3pUTR [-h] [bedfile]
```

Report the 5' or 3' UTRs of each coding transcript in the BED file.

UTRs are defined as the region between transcript start/end and CDS start/end (the CDS is in turn defined as the region between thickStart and thickEnd).

Transcripts with an undefined CDS (i.e. with thickStart and thickEnd set to the same value) are not reported.

Examples

```
> cat transcripts.bed
chr1        167721988        167790819        ENST00000392121.
↪7        0        +        167722151        167787921        0        3        254,
↪167,3000,        0,43594,65831,

> bedparse 3pUTR transcripts.bed
chr1        167787921        167790819        ENST00000392121.
↪7        0        +        167787921        167787921        0        1        2898,
↪        0,
```

1.3.2 CDS

Usage

```
> bedparse cds --help
usage: bedparse cds [-h] [--ignoreCDSonly] [bedfile]

Report the CDS of each coding transcript (i.e. transcripts with distinct
values of thickStart and thickEnd). Transcripts without CDS are not reported.

positional arguments:
  bedfile          Path to the BED file.

optional arguments:
  -h, --help       show this help message and exit
  --ignoreCDSonly  Ignore transcripts that only consist of CDS.
```

Examples

```
> cat transcripts.bed
chr1        167721988        167790819        ENST00000392121.
↪7         0        +        167722151        167787921        0        3        254,
↪167,3000,        0,43594,65831,

> bedparse cds transcripts.bed
chr1        167722151        167787921        ENST00000392121.
↪7         0        +        167722151        167787921        0        3        91,
↪167,102,        0,43431,65668,
```

1.3.3 Promoters

This command reports the promoter of each transcript in the input BED file. The promoter is defined as a fixed interval around the TSS.

Usage

```
> bedparse promoter --help
usage: bedparse promoter [-h] [--up UP] [--down DOWN] [--unstranded] [bedfile]

Report the promoter of each transcript, defined as a fixed interval around its
start.

positional arguments:
  bedfile       Path to the BED file.

  optional arguments:
    -h, --help    show this help message and exit
    --up UP       Get this many nt upstream of each feature.
    --down DOWN   Get this many nt downstream of each feature.
    --unstranded  Do not consider strands.
```

Examples

```
> cat transcripts.bed
chr1        167721988       167790819       ENST00000392121.
↪7         0        +        167722151       167787921       0        3        254,
↪167,3000,        0,43594,65831,

> bedparse promoter transcripts.bed
chr1        167721488       167722488       ENST00000392121.7

> bedparse promoter --up 100 --down 100 transcripts.bed
chr1        167721888       167722088       ENST00000392121.7
```

1.3.4 Introns

Reports BED12 lines corresponding to the introns of each transcript. Unspliced transcripts are not reported.

Usage

```
> bedparse introns --help
usage: bedparse introns [-h] [bedfile]

Report BED12 lines corresponding to the introns of each transcript. Unspliced
transcripts are not reported.

positional arguments:
  bedfile     Path to the BED file.

optional arguments:
  -h, --help  show this help message and exit
```

Examples

```
> cat transcripts.bed
chr1        167721988       167790819       ENST00000392121.
↪7         0        +        167722151       167787921       0        3        254,
↪167,3000,        0,43594,65831,

> bedparse introns transcripts.bed
chr1        167722242       167787819       ENST00000392121.
↪7         0        +        167722242       167722242       0        2        43340,
↪22070,        0,43507,
```

1.3.5 Filter

Filters a BED file based on an annotation file. BED entries with a name (i.e. col4) that appears in the specified column of the annotation are printed to stdout. For efficiency reasons this command doesn't perform BED validation.

Usage

```
> bedparse filter --help
usage: bedparse filter [-h] --annotation ANNOTATION [--column COLUMN]
                       [--inverse]
                       [bedfile]

Filters a BED file based on an annotation. BED entries with a name (i.e. col4)
that appears in the specified column of the annotation are printed to stdout.
For efficiency reasons this command doesn't perform BED validation.

positional arguments:
  bedfile               Path to the BED file.

optional arguments:
  -h, --help            show this help message and exit
  --annotation ANNOTATION, -a ANNOTATION
                        Path to the annotation file.
  --column COLUMN, -c COLUMN
                        Column of the annotation file (1-based, default=1).
  --inverse, -v         Only report BED entries absent from the annotation
                        file.
```

Examples

```
> cat transcripts.bed
chr1        67092164        67231852        ENST00000371007.6        0        -
chr1        67092175        67127261        ENST00000371006.5        0        -
chr1        67092175        67127261        ENST00000475209.6        0        -
chr1        67092394        67134970        ENST00000371004.6        0        -
chr1        67092396        67127261        ENST00000621590.4        0        -
chr1        67092947        67134977        ENST00000544837.5        0        -
chr1        67093558        67231853        ENST00000448166.6        0        -
chr1        67096295        67134977        ENST00000603691.1        0        -
chr1        201283451        201332993        ENST00000263946.7        0        +
chr1        201283451        201332993        ENST00000367324.7        0        +

> cat filter.txt
GeneX        ENST00000263946.7        Other_field
GeneY        ENST00000367324.7        Another_field

> bedparse filter --annotation filter.txt --column 2 transcripts.bed
chr1        201283451        201332993        ENST00000263946.7        0        +
chr1        201283451        201332993        ENST00000367324.7        0        +
```

1.3.6 Join

Adds the content of an annotation file to a BED file as extra columns. The two files are joined by matching the BED Name field (column 4) with a user-specified field of the annotation file.

Usage

```
> bedparse join --help
usage: bedparse join [-h] --annotation ANNOTATION [--column COLUMN]
                     [--separator SEPARATOR] [--empty EMPTY] [--noUnmatched]
                     [bedfile]

Adds the content of an annotation file to a BED file as extra columns. The two
files are joined by matching the BED Name field (column 4) with a user-
specified field of the annotation file.

positional arguments:
  bedfile               Path to the BED file.

optional arguments:
  -h, --help            show this help message and exit
  --annotation ANNOTATION, -a ANNOTATION
                        Path to the annotation file.
  --column COLUMN, -c COLUMN
                        Column of the annotation file (1-based, default=1).
  --separator SEPARATOR, -s SEPARATOR
                        Field separator for the annotation file (default tab)
  --empty EMPTY, -e EMPTY
                        String to append to empty records (default '.').
  --noUnmatched, -n     Do not print unmatched lines.
```

Examples

```
> cat transcripts.bed
chr1        67092164        67231852        ENST00000371007.6       0       -
chr1        67092175        67127261        ENST00000371006.5       0       -
chr1        67092175        67127261        ENST00000475209.6       0       -
chr1        67092394        67134970        ENST00000371004.6       0       -
chr1        67092396        67127261        ENST00000621590.4       0       -
chr1        67092947        67134977        ENST00000544837.5       0       -
chr1        67093558        67231853        ENST00000448166.6       0       -
chr1        67096295        67134977        ENST00000603691.1       0       -
chr1        201283451        201332993        ENST00000263946.7       0       +
chr1        201283451        201332993        ENST00000367324.7       0       +

> cat annotation.txt
GeneX        ENST00000263946.7        Other_field
GeneY        ENST00000367324.7        Another_field

> bedparse join --column 2 --annotation annotation.txt transcripts.bed
chr1        67092164        67231852        ENST00000371007.6       0       -
↪        .
chr1        67092175        67127261        ENST00000371006.5       0       -
↪        .
chr1        67092175        67127261        ENST00000475209.6       0       -
↪        .
chr1        67092394        67134970        ENST00000371004.6       0       -
↪        .
chr1        67092396        67127261        ENST00000621590.4       0       -
↪        .
chr1        67092947        67134977        ENST00000544837.5       0       -
↪        .
```

(continues on next page)

(continued from previous page)

```
chr1        67093558        67231853        ENST00000448166.6        0         -
↪          .
chr1        67096295        67134977        ENST00000603691.1        0         -
↪          .
chr1        201283451        201332993        ENST00000263946.
↪7         0         +         GeneX        Other_field
chr1        201283451        201332993        ENST00000367324.
↪7         0         +         GeneY        Another_field

> bedparse join --column 2 --annotation annotation.txt --noUnmatched transcripts.bed
chr1        201283451        201332993        ENST00000263946.
↪7         0         +         GeneX        Other_field
chr1        201283451        201332993        ENST00000367324.
↪7         0         +         GeneY        Another_field
```

1.3.7 Convert GTF to BED

Converts a GTF file to BED12 format. This tool supports the Ensembl GTF format. The GTF file must contain 'transcript' and 'exon' features in field 3. If the GTF file also annotates 'CDS' 'start_codon' or 'stop_codon' these are used to annotate the thickStart and thickEnd in the BED file.

Usage

```
> bedparse gtf2bed --help
usage: bedparse gtf2bed [-h] [--extraFields EXTRAFIELDS]
                        [--filterKey FILTERKEY] [--filterType FILTERTYPE]
                        [gtf]

Converts a GTF file to BED12 format. This tool supports the Ensembl GTF
format. The GTF file must contain 'transcript' and 'exon' features in field 3.
If the GTF file also annotates 'CDS' 'start_codon' or 'stop_codon' these are
used to annotate the thickStart and thickEnd in the BED file.

positional arguments:
  gtf                   Path to the GTF file.

optional arguments:
  -h, --help            show this help message and exit
  --extraFields EXTRAFIELDS
                        Comma separated list of extra GTF fields to be added
                        after col 12 (e.g. gene_id,gene_name).
  --filterKey FILTERKEY
                        GTF extra field on which to apply the filtering
  --filterType FILTERTYPE
                        Comma separated list of filterKey field values to
                        retain.
```

1.3.8 Convert BED12 to BED6

Convert the BED12 format into BED6 by reporting a separate line for each block of the original record.

Usage

```
> bedparse bed12tobed6 --help
usage: bedparse bed12tobed6 [-h] [--appendExN] [--whichExon {all,first,last}]
                            [--keepIntrons]
                            [bedfile]

Convert the BED12 format into BED6 by reporting a separate line for each block
of the original record.

positional arguments:
  bedfile               Path to the GTF file.

optional arguments:
  -h, --help            show this help message and exit
  --appendExN           Appends the exon number to the transcript name.
  --whichExon {all,first,last}
                        Which exon to return. First and last respectively
                        report the first or last exon relative to the TSS
                        (i.e. taking strand into account).
  --keepIntrons         Add records for introns as well. Only allowed if
                        --whichExon all
```

Examples

```
> cat transcripts.bed
chr1        67092164        67231852        ENST00000371007.6        0        -
↪        67093004        67127240        0        8        1440,187,70,113,158,92,86,
↪7,        0,3070,4087,23187,33587,35001,38977,139681,

> bedparse bed12tobed6 transcripts.bed
chr1        67092164        67093604        ENST00000371007.6        0        -
chr1        67095234        67095421        ENST00000371007.6        0        -
chr1        67096251        67096321        ENST00000371007.6        0        -
chr1        67115351        67115464        ENST00000371007.6        0        -
chr1        67125751        67125909        ENST00000371007.6        0        -
chr1        67127165        67127257        ENST00000371007.6        0        -
chr1        67131141        67131227        ENST00000371007.6        0        -
chr1        67231845        67231852        ENST00000371007.6        0        -
```

1.3.9 Convert chromosome names

Convert chromosome names between UCSC and Ensembl formats. The conversion supports the hg38 assembly up to patch 11 and the mm10 assembly up to patch 4. By default patches are not converted (because the UCSC genome browser does not support them), but can be enabled using the -p flag. When the BED file contains a chromosome that is not recognised, by default the program stops and throws an error. Alternatively, unrecognised chromosomes can be suppressed (-s) or artificially set to 'NA' (-a).

Usage

```
> bedparse convertChr --help
usage: bedparse convertChr [-h] --assembly ASSEMBLY --target TARGET
                           [--allowMissing] [--suppressMissing] [--patches]
                           [bedfile]

Convert chromosome names between UCSC and Ensembl formats. The conversion
supports the hg38 assembly up to patch 11 and the mm10 assembly up to patch 4.
By default patches are not converted (because the UCSC genome browser does not
support them), but can be enabled using the -p flag. When the BED file
contains a chromosome that is not recognised, by default the program stops and
throws an error. Alternatively, unrecognised chromosomes can be suppressed
(-s) or artificially set to 'NA' (-a).

positional arguments:
  bedfile               Path to the BED file.

optional arguments:
  -h, --help            show this help message and exit
  --assembly ASSEMBLY   Assembly of the BED file (either hg38 or mm10).
  --target TARGET       Desidered chromosome name convention (ucsc or ens).
  --allowMissing, -a    When a chromosome name can't be matched between USCS
                        and Ensembl set it to 'NA' (by default thrown as
                        error).
  --suppressMissing, -s
                        When a chromosome name can't be matched between USCS
                        and Ensembl do not report it in the output (by default
                        throws an error).
  --patches, -p         Allows conversion of all patches up to p11 for hg38
                        and p4 for mm10. Without this option, if the BED file
                        contains contigs added by a patch the conversion
                        terminates with an error (unless the -a or -s flags
                        are present).
```

Examples

```
> cat transcripts.bed
chr1        67092164        67231852        ENST00000371007.6        0        -
chr22_KI270928v1_alt        137191        137686        ENST00000630841.
↪1        0        -
chr1_KI270706v1_random        45985        46062        ENST00000611371.
↪2        0        +
chrM        3229        3304        ENST00000386347.1        0        +

> bedparse convertChr --assembly hg38 --target ens transcripts.bed
1        67092164        67231852        ENST00000371007.6        0        -
CHR_HSCHR22_3_CTG1        137191        137686        ENST00000630841.
↪1        0        -
KI270706.1        45985        46062        ENST00000611371.2        0        +
MT        3229        3304        ENST00000386347.1        0        +
```

1.3.10 Validate Format

Simply performs format validation on the input BED file. If any line doesn't adhere to the BED specifications the program reports an error and terminates. The `--fixSeparators` flag replaces fields separated by spaces into fields separated by a single tab. This is useful when writing a BED file by hand or when copy-pasting from a website.

Usage

```
usage: bedparse validateFormat [-h] [--fixSeparators] [bedfile]

Checks whether the BED file provided adheres to the BED format specifications.
Optionally, it can fix field speration errors.

positional arguments:
  bedfile               Path to the BED file.

optional arguments:
  -h, --help            show this help message and exit
  --fixSeparators, -f  If the fields are separated by multiple spaces (e.g.
                        when copy-pasting BED files), replace them into tabs.
```

Examples

```
> cat example.bed
   chr1  a213941196  213942363
  chr1  213942363  213943530
chr1  213943530         213944697

> bedparse validateFormat -f example.bed
chr1    213941196       213942363
chr1    213942363       213943530
chr1    213943530       213944697
```

1.4 Implementations notes

Internally, bedparse processes a bedfile line by line by instantiating objects of the bedline class. The bedline class implements an init() method that performs several checks on each field in order to ensure the correctness of the format, whereas the other methods of the class implement all the bedparse operations (see functionality).

1.5 bedparse.bedline module

class `bedparse.bedline.`**`bedline`**(*line=None*)

Bases: `object`

The bedline class defines an object that represents a single BED[3,4,6,12] line

Parameters **`line`** (*`list`*) – List where each element corresponds to one field of a BED file

`bed12tobed6`(*appendExN=False*, *whichExon='all'*)

Returns a list of bedlines (bed6) corresponding to the exons.

Parameters

- **appendExN** (*bool*) – Appends the exon number to the transcript name
- **whichExon** (*str*) – Which exon to return. One of [“all”, “first”, “last”]. First and last respectively report the first or last exon relative to the TSS (i.e. taking strand into account).

Returns list of bedline objects, one per exon

Return type list

Examples

```
>>> bl = bedline(["chr1", 100, 420, "Name", 0, "+", 210, 310, ".", 4, "20,20,
↪20,20,", "0,100,200,300,"])
>>> for i in bl.bed12tobed6(appendExN=True): print(i)
...
['chr1', 100, 120, 'Name_Exon001', 0, '+']
['chr1', 200, 220, 'Name_Exon002', 0, '+']
['chr1', 300, 320, 'Name_Exon003', 0, '+']
['chr1', 400, 420, 'Name_Exon004', 0, '+']
```

cds (*ignoreCDSonly=False*)
Return the CDS of a coding transcript. Transcripts without CDS are not reported

Parameters **ignoreCDSonly** (*bool*) – If True return None when the entire transcript is CDS

Returns The CDS as a bedline object

Return type *bedline*

Examples

```
>>> bl = bedline(["chr1", 100, 500, "Tx1", 0, "+", 200, 300, ".", 1, "400,",
↪"0,"])
>>> print(bl.cds())
['chr1', 200, 300, 'Tx1', 0, '+', 200, 300, '.', 1, '100,', '0,']
```

introns ()
Returns a bedline object of the introns of a transcript

Returns The introns of the transcripts as a bedline object

Return type *bedline*

Examples

```
>>> bl = bedline(["chr1", 100, 420, "Name", 0, "+", 210, 310, ".", 4, "20,20,
↪20,20,", "0,100,200,300,"])
>>> print(bl.introns())
['chr1', 120, 400, 'Name', 0, '+', 120, 120, '.', 3, '80,80,80,', '0,100,200,
↪']
>>> bl = bedline(["chr1", 100, 420, "Name", 0, "-", 210, 310, ".", 1, "320,",
↪"0,"])
>>> print(bl.introns())
None
```

pprint()
Prints a bedline object formatted as a python list

print(*end='\n'*)
Prints a bedline object

Parameters **end** – Line terminator character

promoter(*up=500*, *down=500*, *strand=True*)
Returns the promoter of a bedline object

Parameters

- **up** (*int*) – Number of upstream bases
- **down** (*int*) – Number of donwstream bases
- **strand** (*bool*) – If false strandedness is ignored

Returns The promoter as a bedline object

Return type *bedline*

Examples

```
>>> bl = bedline(['chr1', 1000, 2000, 'Tx1', '0', '+'])
>>> print(bl.promoter())
['chr1', 500, 1500, 'Tx1']
```

translateChr(*assembly*, *target*, *suppress=False*, *ignore=False*, *patches=False*)
Convert the chromosome name to Ensembl or UCSC

Parameters

- **assembly** (*str*) – Assembly of the BED file (either hg38 or mm10).
- **target** (*str*) – Desidered chromosome name convention (ucsc or ens).
- **suppress** (*bool*) – When a chromosome name can't be matched between USCS and Ensembl set it to 'NA' (by default throws as error)
- **ignore** (*bool*) – When a chromosome name can't be matched between USCS and Ensembl do not report it in the output (by default throws an error)
- **patches** (*bool*) – Allows conversion of all patches up to p11 for hg38 and p4 for mm10. Without this option, if the BED file contains contigs added by a patch the conversion terminates with an error (unless the -a or -s flags are present

Returns A bedline object with the converted chromosome

Return type *bedline*

Examples

```
>>> bl = bedline(['chr1', 1000, 2000, 'Tx1', '0', '-'])
>>> print(bl.translateChr(assembly="hg38", target="ens"))
['1', 1000, 2000, 'Tx1', '0', '-']
>>> bl = bedline(['chr19_GL000209v2_alt', 1000, 2000, 'Tx1', '0', '-'])
>>> print(bl.translateChr(assembly="hg38", target="ens"))
['CHR_HSCHR19KIR_RP5_B_HAP_CTG3_1', 1000, 2000, 'Tx1', '0', '-']
```

tx2genome(*coord*, *stranded=False*)

Given a position in transcript coordinates returns the equivalent in genome coordinates. The transcript coordinates are considered without regard to strand, i.e. 0 is the leftmost position for both + and - strand transcripts, unless the stranded options is set to True.

Parameters

- **coord** (*int*) – Coordinate to convert from transcript-space to genome space
- **stranded** (*bool*) – If True use the rightmost base of negative strand trascripts as 0

Returns Coordinate in genome-space

Return type int

Examples

```
>>> bl = bedline(['chr1', 1000, 2000, 'Tx1', '0', '-'])
>>> bl.tx2genome(10)
1010
>>> bl.tx2genome(10, stranded=True)
1989
```

utr(*which=None*)

Returns the UTR of coding transcripts (i.e. those with a CDS)

Parameters **which** (*int*) – Which UTR to return: 3 for 3'UTR or 5 for 5' UTR

Returns The UTR as a bedline object

Return type *bedline*

Examples

```
>>> bl = bedline(["chr1", 100, 500, "Tx1", 0, "+", 200, 300, ".", 1, "400,",
↪"0,"])
>>> print(bl.utr(which=5))
['chr1', 100, 200, 'Tx1', 0, '+', 100, 100, '.', 1, '100,', '0,']
```

1.6 Bedparse tutorial

Hi, thanks for your interest in `bedparse`!

The following is a short tutorial that will guide you through the functionality of `bedparse`. You can find the `example.bed` file in this repo under docs/example.bed. This file contains 6 human transcript models from Gencode. The first three are non-coding transcripts (i.e. field 7 and 8 contain the same coordinate), whereas the last three are coding (i.e. fields 7 and 8 indicate the thickStart and thickEnd, i.e. start and end of the CDS).

1.6.1 Extracting the promoters

The `bedparse promoter` command reports the promoter of each transcript, defined as a user specified interval around the TSS. For example, we can extract promoters consisting of 1000bp upstream and 500bp downstream of the CDS:

```
$ bedparse promoter --up 1000 --down 500 example.bed
chr1    10868   12368   ENST00000456328.2
chr1    11009   12509   ENST00000450305.2
chr1    29070   30570   ENST00000488147.1
chr1    922927  924427  ENST00000420190.6
chr1    924149  925649  ENST00000437963.5
chr1    924737  926237  ENST00000342066.7
```

Note how the TSS (and as a consequence the promoter) depends on the strand: for transcripts on the negative strand the TSS is the end coordinate, i.e. column 3. The `--unstranded` option allows you to override this behaviour and report promoters as an interval around column 2, thus disregarding the strand.

1.6.2 Extracting 5' or 3' UTRs

The UTRs are defined in a BED file as the region between the start (column 2) and the thickStart (column 7) for the 5' and between the thickEnd (column 8) and the end (column 3) for the 3'. These rules are reversed for transcripts on the - strand, and `bedparse` automatically takes care of this. Additionally, `bedparse` also handles correctly UTRs that span multiple exons: in these cases `bedparse` recomputes all exon starts and exon lengths as sets coulmns 11 and 12 accordingly.

```
$ bedparse 5pUTR example.bed
chr1    923927  924431  ENST00000420190.6       0       +       923927  923927  0     ␣
↪  1       504,    0,
chr1    925149  925941  ENST00000437963.5       0       +       925149  925149  0     ␣
↪  2       40,20,  0,772,
chr1    925737  925941  ENST00000342066.7       0       +       925737  925737  0     ␣
↪  2       63,20,  0,184,
```

Clearly, as you can see from the output above, UTRs are only reported for coding transcripts.

1.6.3 Extracting the CDS

To extract the CDS of the coding transcripts in the BED file use the `bedparse cds` command:

```
$ bedparse cds example.bed
chr1    924431  939291  ENST00000420190.6       0       +       924431  939291  0     ␣
↪  7       517,92,182,51,125,90,17,        0,1490,5723,6607,11340,14608,14843,
chr1    925941  935793  ENST00000437963.5       0       +       925941  935793  0     ␣
↪  4       72,182,51,22,   0,4213,5097,9830,
chr1    925941  944153  ENST00000342066.7       0       +       925941  944153  0     ␣
↪  13      72,182,51,125,90,186,163,116,79,500,125,111,246,        0,4213,5097,9830,
↪13098,13333,15202,16194,16468,16617,17311,17756,17966,
```

Note how non-coding transcripts are not reported (because by definition they don't have a CDS). Also, note how the number of exons (column 10) and exon lengths and starts (columns 11 and 12) have been readjusted to reflect the fact that the transcripts have "lost" the UTRs. To visualise this operation you can save the output of the command above to a new text file and upload it as a custom track in the UCSC genome browser: you'll see that the new transcripts only correspond to the thick portion of the original Gencode transcripts.

1.6.4 Extracting introns

In a BED file introns are implicitly defined as the genomic regions between exons. The `bedparse introns` command creates new "artificial" transcripts that correspond to the introns of the original transcripts:

```
$ bedparse introns example.bed
chr1    12227   13220   ENST00000456328.2       0       +       12227   12227   0       ␣
↪  2       385,499,        0,494,
chr1    12057   13452   ENST00000450305.2       0       +       12057   12057   0       ␣
↪  5       121,385,277,168,78,     0,170,640,995,1317,
chr1    14501   29533   ENST00000488147.1       0       -       14501   14501   0       ␣
↪  10      503,757,659,92,177,237,172,206,6371,4642,       0,537,1446,2264,2554,2867,
↪3241,3560,3865,10390,
chr1    924948  939274  ENST00000420190.6       0       +       924948  924948  0       ␣
↪  6       973,4141,702,4682,3143,145,     0,1065,5388,6141,10948,14181,
chr1    925189  935771  ENST00000437963.5       0       +       925189  925189  0       ␣
↪  4       732,4141,702,4682,      0,824,5147,5900,
chr1    925800  943907  ENST00000342066.7       0       +       925800  925800  0       ␣
↪  13      121,4141,702,4682,3143,145,1683,829,158,70,194,320,99,  0,213,4536,5289,
↪10096,13329,13660,15506,16451,16688,17258,17577,18008,
```

1.6.5 Convert BED12 to BED6

It’s often convenient to convert a BED12 file into BED6, where each exon appears on its own line. This is easily done with `bedparse bed12tobed6`:

```
$ bedparse bed12tobed6 --appendExN example.bed
chr1    11868   12227   ENST00000456328.2_Exon001       0       +
chr1    12612   12721   ENST00000456328.2_Exon002       0       +
chr1    13220   14409   ENST00000456328.2_Exon003       0       +
chr1    12009   12057   ENST00000450305.2_Exon001       0       +
chr1    12178   12227   ENST00000450305.2_Exon002       0       +
chr1    12612   12697   ENST00000450305.2_Exon003       0       +
chr1    12974   13052   ENST00000450305.2_Exon004       0       +
chr1    13220   13374   ENST00000450305.2_Exon005       0       +
chr1    13452   13670   ENST00000450305.2_Exon006       0       +
chr1    14403   14501   ENST00000488147.1_Exon001       0       -
chr1    15004   15038   ENST00000488147.1_Exon002       0       -
chr1    15795   15947   ENST00000488147.1_Exon003       0       -
chr1    16606   16765   ENST00000488147.1_Exon004       0       -
chr1    16857   17055   ENST00000488147.1_Exon005       0       -
chr1    17232   17368   ENST00000488147.1_Exon006       0       -
chr1    17605   17742   ENST00000488147.1_Exon007       0       -
chr1    17914   18061   ENST00000488147.1_Exon008       0       -
chr1    18267   18366   ENST00000488147.1_Exon009       0       -
chr1    24737   24891   ENST00000488147.1_Exon010       0       -
chr1    29533   29570   ENST00000488147.1_Exon011       0       -
chr1    923927  924948  ENST00000420190.6_Exon001       0       +
chr1    925921  926013  ENST00000420190.6_Exon002       0       +
chr1    930154  930336  ENST00000420190.6_Exon003       0       +
chr1    931038  931089  ENST00000420190.6_Exon004       0       +
chr1    935771  935896  ENST00000420190.6_Exon005       0       +
chr1    939039  939129  ENST00000420190.6_Exon006       0       +
chr1    939274  939291  ENST00000420190.6_Exon007       0       +
chr1    925149  925189  ENST00000437963.5_Exon001       0       +
chr1    925921  926013  ENST00000437963.5_Exon002       0       +
chr1    930154  930336  ENST00000437963.5_Exon003       0       +
chr1    931038  931089  ENST00000437963.5_Exon004       0       +
chr1    935771  935793  ENST00000437963.5_Exon005       0       +
chr1    925737  925800  ENST00000342066.7_Exon001       0       +
chr1    925921  926013  ENST00000342066.7_Exon002       0       +
```

(continues on next page)

(continued from previous page)

```
chr1    930154  930336  ENST00000342066.7_Exon003       0       +
chr1    931038  931089  ENST00000342066.7_Exon004       0       +
chr1    935771  935896  ENST00000342066.7_Exon005       0       +
chr1    939039  939129  ENST00000342066.7_Exon006       0       +
chr1    939274  939460  ENST00000342066.7_Exon007       0       +
chr1    941143  941306  ENST00000342066.7_Exon008       0       +
chr1    942135  942251  ENST00000342066.7_Exon009       0       +
chr1    942409  942488  ENST00000342066.7_Exon010       0       +
chr1    942558  943058  ENST00000342066.7_Exon011       0       +
chr1    943252  943377  ENST00000342066.7_Exon012       0       +
chr1    943697  943808  ENST00000342066.7_Exon013       0       +
chr1    943907  944575  ENST00000342066.7_Exon014       0       +
```

The optional flag –appendExN adds ExonNNN to the end of each transcript name.

1.6.6 APIs

Bedparse can also be imported as a python module. The API documentation contains detailed information of the bedline class and its methods. The following is simple example of how to use it:

```
In [1]: from bedparse import bedline

In [2]: l = bedline(['chr1', 1000, 2000, 'Tx1', '0', '+'])

In [3]: prom = l.promoter()

In [4]: prom.print()
chr1    500     1500    Tx1

In [5]: prom.pprint()
['chr1', 500, 1500, 'Tx1']

In [6]: ens_prom = prom.translateChr(assembly="hg38", target="ens")

In [7]: ens_prom.print()
1       500     1500    Tx1
```

Python Module Index

Index