The BED format

Jeffrey Niu, Danielle Denisko, Michael M. Hoffman

August 15, 2019

1 Specification

Browser Extensible Data (BED) is a whitespace-delimited file format, where each **file** consists of one or more **lines**.[1] Each **line** describes discrete genomic **features** by physical start and end position on a linear **chromosome** of a genome in terms of a starting and ending position. The file extension for the BED format is `.bed`.

All **fields** are 7-bit US ASCII. In this document, names of **fields** belonging to the BED format are in sans-serif font with a grey background; technological terms such as file extensions and programs are in typewriter font; and terms from section 1.1 and section 1.2 are in bold font. Regular expressions in this document follow POSIX/IEEE 1003.1 extended syntax.[2]

1.1 Terminology and concepts

0-start, half-open coordinate system: A coordinate system where the first base starts at position 0 and the start of the interval is included but the end is not. For example, for a sequence of bases ACTGCG, the bases given by the interval [2, 4) are TG.

Chromosome: A sequence of nucleic acids that contain genetic information. Chromosomes are numbered starting from 1. There are also sex chromosomes "X" and "Y", mitochondrial DNA labelled as "M", and possibly sequences from an unknown chromosome, typically labelled

[1] "Frequently Asked Questions: Data File Formats." UCSC Genome Browser FAQ, https://genome.ucsc.edu/FAQ/FAQformat.html

[2] *IEEE Standard for Information Technology–Portable Operating System Interface (POSIX(R)) Base Specifications*, IEEE 1003.1-2017, 2017

as "Un". The name of each chromosome is often prefixed with "chr". Some examples of chromosome names would be: chr1, chr2, chrX, chrY, chrM, chrUn.

Field: Data stored as non-whitespace text.

Line: A record by a line terminator. Can either a **data line** made of whitespace-separated **fields**, a **comment line**, or a **blank line**.

File: A plain text file that contains one or more **lines**.

BED*n*: Designates a **file** that has the first n **fields** of the BED format. For example, **BED3** means a **file** with only the first three **fields**; **BED12** means a **file** with all 12 **fields**.

BED*n*+*m*: Designates a custom **file** that has the first n **fields** of the BED format, followed by m **fields** of custom data defined by the user. For example, **BED6+4** means a **file** with the first six **fields** of the BED format, followed by four user-defined **fields**.

Feature: A linear region of a **chromosome** with specifier properties. For example, a **file's features** might all be peaks called from ChIP-seq data, or transcript.

Block: Linear subfeatures within a **feature**. Usually these are exons.

1.2 Lines

1.2.1 Data lines

Data lines contain **feature** information. A data line is composed of **fields** separated by whitespace. The whitespace must match `\s+`. However, a stricter recommendation for whitespacing is given in section 3.4.

1.2.2 Comment lines and blank lines

Both comment lines and blank lines provide no **feature** data.

Comment lines start with `#` with no whitespace beforehand. A `#` appearing anywhere else in a line is treated as **feature** data, not a comment. Blank lines consist entirely of whitespace. Both

comment and blank lines can appear as any line in a **file** (at the beginning, middle, or end) and in any quantity.

1.3 BED fields

Each **data line** contains between three and 12 whitespace-delimited **fields**. The first three **fields** are mandatory, and the last nine **fields** are optional. In optional **fields**, the order is binding; if one **field** is filled, then all previous **fields** must also be filled. However, **BED10** is not allowed.[3]

In a BED **file**, each **data line** must have the same number of **fields**.

The positions in BED **fields** are all described in the **0-based, half-open coordinate system**.

Col	Field	Type	Regexp or range	Brief description
1	chrom	String	\w+[4]	Chromosome or scaffold name
2	chromStart	Int	$[0, 2^{32} - 1]$	Feature start position
3	chromEnd	Int	$[0, 2^{32} - 1]$	Feature end position
4	name	String	[^\s]+	Feature description
5	score	Int	$[0, 1000]$	A numerical value
6	strand	String	[+-.]	Feature strand
7	thickStart	Int	$[0, 2^{32} - 1]$	Thick display start position
8	thickEnd	Int	$[0, 2^{32} - 1]$	Thick display end position
9	itemRgb	Int,Int,Int	$[0, 255], [0, 255], [0, 255]$	Display color
10	blockCount	Int	[0, chromEnd − chromStart][5]	Number of blocks
11	blockSizes	List[Int]	(\d+,){ blockCount -1}\d+,? [6]	List of block sizes
12	blockCount	List[Int]	(\d+,){ blockCount -1}\d+,?	List of block start positions

1. chrom : The name of the **chromosome** or scaffold where the **feature** is present. The name that the **chromosome** or scaffold is given should align with the associated genome, such as

[3] Knowing only the number of **blocks** has almost no use cases.
[4] Though restrictive, this limitation makes BED **files** portable to varying environments.
[5] chromEnd − chromStart is the maximum number of **blocks** that can exist without overlaps.
[6] For example, if blockCount =4, then the regexp would be (\d+,){3}\d+,?

hg38. Additionally, the naming of **chromosomes** should be consistent within a **file**. For example, "17" and "chr17" should not represent the same **chromosome**.

2. `chromStart`: The starting position of the **feature** in the **chromosome** or scaffold. `chromStart` must be an integer greater than or equal to zero and less than the total number of bases of the **chromosome** to which it belongs. If the size of the **chromosome** is unknown, then `chromStart` must be less than or equal to $2^{32} - 1$, which is the maximum size of an unsigned integer.

3. `chromEnd`: The ending position of the **feature** in the **chromosome** or scaffold. `chromEnd` must be an integer greater than or equal to the value of `chromStart` and less than or equal to the total number of bases in the **chromosome** to which it belongs. If the size of the **chromosome** is unknown, then `chromEnd` must be less than or equal to $2^{32} - 1$, which is the maximum size of an unsigned integer.

4. `name`: A string that describes the **feature**. The name must be one or more non-whitespace characters. The name will be displayed next to the **feature** in the Genome Browser. There is no default value for name. If the **feature** does not need a name, there is a recommendation for the default value in section 3.2.

5. `score`: An integer between 0 and 1000, inclusive. The score is used by the Genome Browser to shade **features** where **features** with higher scores get darker shades. If the **feature** has no score, then "0" should be used.

6. `strand`: The strand that the **feature** appears on. The strand can either refer to the `+` (sense or coding) strand or the `-` (antisense or complementary) strand. If the **feature** has no strand information, then a dot "`.`" must be used.

7. `thickStart`: The starting position at which the **feature** is drawn thickly in the Genome Browser. This value must be an integer between `chromStart` and `chromEnd`, inclusive. There is no specified default value for `thickStart`. Instead, if the **feature** does not need a thick outline, there is a recommendation for the default value in section 3.2.

8. `thickEnd`: The ending position at which the **feature** is drawn thickly in the Genome Browser.

This value must be an integer greater than or equal to `thickStart` and less than or equal to `chromEnd`, inclusive. If this **field** is not specified but `thickStart` is, then the entire **feature** will have thick display. There is no specified default value for `thickEnd`. Instead, if the **feature** does not need a thick outline, there is a recommendation for the default value in section 3.2.

9. `itemRgb`: A RGB value that determines the color that this **feature** will be given when visualized. The RGB value must be in the format of three integers separated by commas, each integer between the values of 0 and 255, inclusive. If the **feature** does not need to be colored, then a single zero "0" should be used, though any non-negative integer can also be used.

10. `blockCount`: The number of **blocks** in the **feature**. In the Genome Browser, the **blocks** will appear thicker than the rest of the **feature**. `blockCount` must be an integer greater than zero. There is no default value for `blockCount` because `blockSizes` and `blockStarts` cannot be defined without `blockCount`.

11. `blockSizes`: A comma-separated list of length `blockCount` that contains the size of each **block**. There must be no spaces before or after the comma(s). There may be a trailing comma after the last element of the list. Each size must be an integer greater than or equal to zero but less than or equal to the length of the **feature**. There is no default value for `blockSizes` because `blockStarts` cannot be verified without `blockSizes`.

12. `blockStarts`: A comma-separated list of length `blockCount` containing the positions of where each **block** starts, relative to `chromStart`. There should be no spaces before or after the comma(s). There may be a trailing comma after the last element of the list. Each element in `blockStarts` is paired with the corresponding element in `blockSizes`. Each `blockStart` must be an integer between zero and `chromEnd` − `chromStart`, and `chromStart` + `blockStart` + `blockSize` must be less or equal to `chromEnd`. These conditions enforce that each **block** is contained within the **feature**. The first **block** must start at `chromStart` and the last **block** must end at `chromEnd`. Moreover, the **blocks** must not overlap each other. The list must be sorted in ascending order. There is no default value for `blockStarts`.

2 Examples

An example from the UCSC Genome Browser FAQ[7]:

```
chr7 127471196  127472363  Pos1  0  +
chr7 127472363  127473530  Pos2  0  +
chr7 127473530  127474697  Pos3  0  +
chr7 127474697  127475864  Pos4  0  +
chr7 127475864  127477031  Neg1  0  -
chr7 127477031  127478198  Neg2  0  -
chr7 127478198  127479365  Neg3  0  -
chr7 127479365  127480532  Pos5  0  +
chr7 127480532  127481699  Neg4  0  -
```

Another example from the UCSC Genome Browser FAQ that uses all 12 BED **fields**:

```
chr22 1000 5000 cloneA 960 + 1000 5000 0 2 567,488, 0,3512
chr22 2000 6000 cloneB 900 - 2000 6000 0 2 433,399, 0,3601
```

One can verify that the **blocks** satisfy the constraints. The first **block** starts at `chromStart` since the first `blockStart` is 0. The last **block** ends at `chromEnd` since the last **block** starts at 3512 with size 488, and $3512 + 488 =$ `chromEnd` $-$ `chromStart`.

3 Recommended practice for the BED format

3.1 Mandatory fields

- `chrom`: The name of each **chromosome** should also match the names from a tool or match other files being used such as a chromosome sizes file.

3.2 Optional fields

- `name`: A recommended default value for `name` is a dot ".".

[7] "Frequently Asked Questions: Data File Formats." UCSC Genome Browser FAQ, https://genome.ucsc.edu/FAQ/FAQformat.html

- `itemRgb`: It is recommended that eight or fewer colors should be used as too many colors may slow down some visualization tools.
- `thickStart` and `thickEnd`: It is recommended that if there is no need for **features** to be drawn thickly, then both `thickStart` and `thickEnd` should be set to `chromStart`.

3.3 Sorting

BED **files** should be sorted by `chrom` alphabetically, then by `chromStart` numerically. For example, the order that **chromosomes** should appear should be chr1, chr10, chr11, chr12, ..., chr2, chr20, chr21, ..., chr3, ..., chrM, chrX, chrY.

3.4 Whitespacing

Though any kind of whitespacing is allowed as a delimiter for the BED **fields**, it is recommended that a single TAB (`\t`) is used because almost all tools support TABs while some tools do not support other kinds of whitespace.

3.5 Very large BED files

For **files** over 50 megabytes in size intended for the Genome Browser, it is recommended that the **file** be converted to `bigBed` format, which is an indexed binary format.[8] The conversion is done by a program called `bedToBigBed`.[9]

4 UCSC track files

Track files are files that contain additional information intended for a visualization tool such as the UCSC Genome Browser.[10] Track files contain browser lines and track lines that precede lines from a file format supported by the Genome Browser.[11] Track files are not valid BED files. This means

[8] Kent, W J et al. (2010) "BigWig and BigBed: enabling browsing of large distributed datasets." *Bioinformatics (Oxford, England)* 26(17):2204-2207. https://doi.org/10.1093/bioinformatics/btq351

[9] "bigBed Track Format." UCSC Genome Browser FAQ, https://genome.ucsc.edu/goldenPath/help/bigBed.html

[10] Haeussler, Maximilian et al. (2019) "The UCSC Genome Browser database: 2019 update." *Nucleic Acids Research* 47(D1):D853–D858. https://doi.org/10.1093/nar/gky1095

[11] "Displaying your own annotations in the Genome Browser." UCSC Genome Browser FAQ, https://genome.ucsc.edu/goldenPath/help/customTrack.html#lines

that the BED format must not have any browser or track lines. To distinguish between BED files and track files, it is recommended that track files use the file extension `.track`.