The Browser Extensible Data (BED) format

Jeffrey Niu, Danielle Denisko, Michael M. Hoffman

5 Jan 2022

The master version of this document can be found at https://github.com/samtools/hts-specs. This printing is version 9ddbc52 from that repository, last modified on the date shown above.

1 Specification

BED is a whitespace-delimited file format, where each **file** consists of zero or more **lines**.[1] Data are in **data lines**, which describe discrete genomic **features** by physical start and end position on a linear **chromosome**. The file extension for the BED format is `.bed`.

1.1 Scope

This specification formalizes reasonable interpretations of the UCSC Genome Browser BED description. This specification also makes clear potential interoperability issues in the current format, which could be addressed in a future specification.

1.2 Typographic conventions

This document uses several typographic conventions (Table 1).

Style	Meaning	Examples
Bold	Terms defined in subsections 1.3–1.4	**chromosome** **file**
Sans serif	Names of **fields**	chrom chromStart chromEnd
Fixed-width	Literals or regexes[2]	`.bed` `grep` `[[:alnum:]]+` `ATCG`

Table 1: **Typographic conventions.**

1.3 Terminology and concepts

0-based, 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`.

[1] "Frequently Asked Questions: Data File Formats." University of California, Santa Cruz (UCSC) Genome Browser FAQ, https://genome.ucsc.edu/FAQ/FAQformat.html

[2] POSIX/IEEE 1003.1–2017 Extended Regular Expressions, for the "C" locale. *IEEE Standard for Information Technology—Portable Operating System Interface (POSIX) Base Specifications*, IEEE 1003.1–2017, 2017

BED field: One of the 12 standard **field**s defined in this specification. The first 3 **BED field**s are mandatory. The remaining 9 **BED field**s are optional.

BED*n*: A **file** with the first *n* **BED field**s. For example, **BED3** means a **file** with only the first 3 **BED field**s; **BED12** means a **file** with all 12 **BED field**s.

BED*n*+: A **file** that has at least the first *n* **BED field**s, followed by zero or more of the remaining **BED field**s and zero or more **custom field**s. A BED*n* **file** also satisfies the definition of a BED*n*+ **file**.

BED*n*+*m*: A **file** that has a custom format starting with the first *n* **field**s of the BED format, followed by *m* **custom field**s. For example, **BED6+4** means a **file** with the first 6 **BED field**s, followed by 4 custom **field**s.

blank line: A **line** consisting entirely of horizontal whitespace.

block: Linear subfeatures within a **feature**. Usually used to designate exons.

chromosome: A sequence of nucleobases with a name. In this specification, "chromosome" may also describe a named scaffold that does not fit the biological definition of a chromosome. Often, **chromosome**s are numbered starting from `1`. There are also often sex **chromosome**s such as `W`, `X`, `Y`, and `Z`, mitochondrial **chromosome**s such as `M`, and possibly scaffolds from an unknown chromosome, often labeled `Un`. The name of each **chromosome** is often prefixed with `chr`. Examples of **chromosome** names include `chr1`, `21`, `chrX`, `chrM`, `chrUn`, `chr19_KI270914v1_alt`, and `chrUn_KI270435v1`.

comment line: A **line** that starts with `#` with no horizontal whitespace beforehand.

custom field: A **field** defined by the **file** creator. **Custom field**s occur in each **line** after any **BED field**s.

data line: A **line** that contains **feature** data.

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

field: Data stored as non-tab text. All **field**s are 7-bit US ASCII printable characters[3].

field separator: One or more horizontal whitespace characters (space or tab). The **field separator** must match the regex `[ \t]+`. The **field separator** can vary throughout the **file**. Some capabilities of the BED format, however, are available only when a single tab is used as the **field separator** throughout the **file**.

file: Sequence of one or more **line**s.

line: String terminated by a **line separator**, in one of the following classes. Either a **data line**, a **comment line**, or a **blank line**. Discussed more fully in subsection 1.4.

line separator: Either carriage return (`\r`, equivalent to `\x0d`), newline (`\n`, equivalent to `\x0a`), or carriage return followed by newline (`\r\n`, equivalent to `\x0d\x0a`). The same **line separator** must be used throughout the **file**.

[3] Characters in the range `\x20` to `\x7e`, therefore not including any control characters

1.4 Lines

1.4.1 Data lines

Data lines contain **feature** data. A **data line** is composed of **field**s separated by **field separator**s.

1.4.2 Comment lines and blank lines

Both **comment line**s and **blank line**s provide no **feature** data.

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

Blank lines consist entirely of horizontal whitespace. Both comment and blank **line**s may appear as any **line** in a **file**, at the beginning, middle, or end of the **file**. They may appear in any quantity.

1.5 BED fields

Each **data line** contains between 3 and 12 **BED field**s delimited by a **field separator**. The first 3 **BED field**s are mandatory, and the last 9 **BED field**s are optional (Table 2). In optional **BED field**s, the order is binding—if an optional **BED field** is filled, then all previous **BED field**s must also be filled. Any **BED field** included on any **data line** in the **file** must not be empty on any other **data line**. **BED10** and **BED11** are prohibited.

Col	BED Field	Type	Regex or range	Brief description
1	chrom	String	`[[:alnum:]_]{1,255}`[4]	**Chromosome** name
2	chromStart	Int	$[0, 2^{64} - 1]$	**Feature** start position
3	chromEnd	Int	$[0, 2^{64} - 1]$	**Feature** end position
4	name	String	`[\x20-\x7e]{1,255}`	**Feature** description
5	score	Int	$[0, 1000]$	A numerical value
6	strand	String	`[-+.]`	**Feature** strand
7	thickStart	Int	$[0, 2^{64} - 1]$	Thick start position
8	thickEnd	Int	$[0, 2^{64} - 1]$	Thick end position
9	itemRgb	Int,Int,Int	$([0, 255], [0, 255], [0, 255])$ `| 0`	Display color
10	blockCount	Int	$[0, \mathsf{chromEnd} - \mathsf{chromStart}]$[5]	Number of **block**s
11	blockSizes	List[Int]	`([[:digit:]]+,){blockCount−1}[[:digit:]]+,?`[6]	**Block** sizes
12	blockStarts	List[Int]	`([[:digit:]]+,){blockCount−1}[[:digit:]]+,?`	**Block** start positions

Table 2: **BED Fields.**

In a BED **file**, each **data line** must have the same number of **field**s. The positions in **BED field**s are all described in the **0-based, half-open coordinate system**.

1.6 Coordinates

1. chrom: The name of the **chromosome** where the **feature** is present. Limiting to word characters only, instead of all non-whitespace printable characters, makes BED **file**s more

[4] `[[:alnum:]_]` is equivalent to the regex `[A-Za-z0-9_]`. It is also equivalent to the Perl extension `[[:word:]]`

[5] chromEnd-chromStart is the maximum number of **block**s that may exist without overlaps

[6] For example, if blockCount = 4, then the allowed regex would be `([[:digit:]]+,){3}[[:digit:]]+,?`

portable to varying environments which may make different assumptions about allowed characters. The name must be between 1 and 255 characters long, inclusive.

2. chromStart: Start position of the **feature** on the **chromosome**. chromStart must be an integer greater than or equal to 0 and less than or equal to 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 $2^{64} - 1$, which is the maximum size of an unsigned 64-bit integer.

3. chromEnd: End position of the **feature** on the **chromosome**. 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 chromEnd is equal to chromStart, this indicates a **feature** between chromStart and the preceding base, such as an insertion. When chromStart and chromEnd are both 0, this indicates a feature before the entire **chromosome**. If the size of the **chromosome** is unknown, then chromEnd must be less than or equal to $2^{64} - 1$, the maximum size of an unsigned 64-bit integer.

1.7 Simple attributes

4. name: String that describes the **feature**. name must be 1 to 255 non-tab characters. name must not contain whitespace, unless the only **field separator** is a single tab. Multiple **data line**s may share the same name. In **BED5+ file**s where all **feature**s have uninformative names, dot (`.`) may be used as a name on every **data line**. A visual representation of the BED format may display name next to the **feature**.

5. score: Integer between 0 and 1000, inclusive. In **BED6+ file**s where all **feature**s have uninformative scores, 0 should be used as the score on every **data line**. A visual representation of the BED format may shade **feature**s differently depending on their score.

6. strand: Strand that the **feature** appears on. The strand may either refer to the `+` (sense or coding) strand or the `-` (antisense or complementary) strand. If the **feature** has no strand information or unknown strand, then a dot (`.`) must be used as an uninformative value. strand should be treated as `.` when parsing files that are not **BED6+**.

1.8 Display attributes

7. thickStart: Start position at which the **feature** is visualized with a thicker or accented display. This value must be an integer between chromStart and chromEnd, inclusive. In **BED7+ file**s where all **feature**s have uninformative thickStarts, the value of chromStart should be used as the thickStart on every **data line**.

8. thickEnd: End position at which the **feature** is visualized with a thicker or accented display. This value must be an integer greater than or equal to thickStart and less than or equal to chromEnd, inclusive. In **BED8+ file**s where all **feature**s have uninformative thickEnds, the value of chromEnd should be used as the thickEnd on every **data line**. In BED **file**s that are not **BED7+**, the whole **feature** has thick display. In **BED7+ file**s, to achieve the same effect, set thickStart equal to chromStart and thickEnd equal to chromEnd. If thickEnd is not specified but thickStart is, then the entire **feature** has thick display.

9. itemRgb: A triple of integers that determines the color of this **feature** when visualized. The triple is three integers separated by commas. Each integer is between 0 and 255, inclusive. To

make a **feature** black, itemRgb may be a single 0, which is visualized identically to a **feature** with itemRgb of 0,0,0. An itemRgb of 0 is a special case and no other single-number value is valid. In **BED9+ file**s where all **feature**s have uninformative itemRgbs, 0 should be used as the itemRgb on every **data line**.

1.9 Blocks

10. blockCount: Number of **block**s in the **feature**. blockCount must be an integer greater than 0. blockCount is mandatory in **BED12+ file**s. A visual representation of the BED format may have blocks appear thicker than the rest of the **feature**.

11. blockSizes: Comma-separated list of length blockCount containing the size of each **block**. There must be no spaces before or after commas. There may be a trailing comma after the last element of the list. blockSizes is mandatory in **BED12+ file**s.

12. blockStarts: Comma-separated list of length blockCount containing each **block**'s start position, relative to chromStart. There must not be spaces before or after the commas. 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 blockStarts element must be an integer between 0 and $\mathsf{chromEnd} - \mathsf{chromStart}$, inclusive. For each couple i of $(\mathsf{blockStarts}_i, \mathsf{blockSizes}_i)$, the quantity $\mathsf{chromStart} + \mathsf{blockStarts}_i + \mathsf{blockSizes}_i$ 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 **block**s must not overlap. The list must be sorted in ascending order. blockStarts is mandatory in **BED12+ file**s.

1.10 Custom fields

Custom fields defined by the **file** creator may contain any printable 7-bit US ASCII character (which includes spaces, but excludes tabs, newlines, and other control characters). **Custom field**s may only be empty or contain whitespace when a single tab is used as the **field separator** throughout the **file**. This specification does not contain a means for interchanging custom BED format definitions.

2 Examples

2.1 Example BED6 file 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  -
```

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

2.2 Example BED12 file from the UCSC Genome Browser FAQ

```
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
```

The **block**s in this example satisfy the required constraints. The first **block** starts at chromStart since the first blockStarts element is 0. The last **block** ends at chromEnd since the last **block** starts at position 4512 (1000+3512) with size 488, and therefore ends at position 5000 (4512+488).

3 Recommended practice for the BED format

3.1 Mandatory BED fields

- chrom: The name of each **chromosome** should also match the names from a reference genome, if applicable. For example, in the human genome, the **chromosome**s may be named `chr1` to `chr22`, `chrX`, `chrY`, and `chrM`. Names should be consistent within a **file**. For example, one should not use both `17` and `chr17` to represent the same **chromosome** in the same **file**.

3.2 Optional BED fields

- name: Names should avoid using the space character even if the only **field separator** is a single tab character, because parsers may interpret a space as a **field separator**.

- itemRgb: Eight or fewer colors should be used as too many colors may slow down visualizations and are difficult for humans to distinguish.[8] Color schemes should be colorblind-friendly. Red-green color schemes should be avoided.

3.3 Custom fields

Definitions of a custom BED format should restrict the type of each **custom field** to the extent possible. Each **custom field** should contain either one of several specified data types (Table 3) or a comma-separated list of Integer, Unsigned, or Float.

Type	Definition
Integer	Decimal string representation of 64-bit signed integer
Unsigned	Decimal string representation of 64-bit unsigned integer
Float	Decimal string representation of 64-bit floating point number[9]
Character	One printable character
String	One or more printable characters

Table 3: **Custom field data types.**

The AutoSQL format[10] provides one method for defining custom BED formats in a separate file.

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

[9] *IEEE Standard for Binary Floating-Point Arithmetic.* IEEE 754–1985, 1985

[10] Kent, W. James. (2000) "AutoSQL." https://hgwdev.gi.ucsc.edu/~kent/exe/doc/autoSql.doc

3.4 Sorting

BED **file**s should be sorted by chrom, then by chromStart numerically, and finally by chromEnd numerically. chrom may be sorted using any scheme (such as lexicographic or numeric order), but all **data line**s with the same chrom value should occur consecutively. For example, the lexicographic order of `chr1`, `chr10`, `chr11`, `chr12`, . . . , `chr2`, `chr20`, `chr21`, . . . , `chr3`, . . . , `chrX`, `chrY`, `chrM` is an acceptable sorting. This ordering is equivalent to sorting the **file** using the command `LC_ALL=C sort -k 1,1 -k 2,2n -k 3,3n`. The numeric order of `chr1`, `chr2`, . . . , `chr21`, `chr22`, `chrM`, `chrX`, `chrY` is also acceptable. Arbitrary orderings of chrom are allowed, but regardless of the **chromosome** sorting scheme, **data line**s for two **feature**s on the same **chromosome** should not have any **data line**s for **feature**s on other **chromosome**s between them. Multiple **feature**s that have the same chrom, chromStart, and chromEnd can appear in any order. **Comment line**s and **blank line**s do not have to be sorted according to the schemes mentioned.

Sorting is recommended because the implementation of downstream operations is easier if features of one chromosome are all grouped together and chromStart is non-decreasing within a chromosome.

For **BED4+** files, a sorting scheme may also order by optional **BED field**s and any **custom field**s. A recommendation for how to do this is outside the scope of this version of the specification. Total deterministic sorting of BED **file**s can prevent downstream analyses from producing different results depending on sort order.

3.5 Whitespace

We recommend that only a single tab (`\t`) be used as **field separator**. This is because almost all tools support tabs while some tools do not support other kinds of whitespace. Also, spaces within the name **BED field** may be used only if the **field separator** is tab throughout the **file**.

It would be sensible for future major versions of this specification or overlay formats built on top of this specification to require that only a single tab be used as **field separator**.

3.6 Large BED files

If a **file** intended for visualization is over 50 MiB in size, the **file** should be converted to `bigBed` format, which is an indexed binary format.[11] The `bedToBigBed` program may perform this conversion.[12]

Tabix is another option for storing larger BED **files**.[13] Tabix works only on **file**s using a single tab as the **field separator**.

4 Information supplied out-of-band

Some information about a BED **file** can only be supplied unambiguously separately from the **data line**s of the BED **file**. This specification does not contain a means for interchanging this information. Information that must be supplied out-of-band include:

- Which of the first 4 to 12 **field**s are standard **BED field**s and which are **custom field**s.
- The genome assembly that defines chrom, chromStart, and chromEnd.

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

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

[13] Li H. (2011) "Tabix: fast retrieval of sequence features from generic TAB-delimited files." *Bioinformatics* 27(5):718–719. https://doi.org/10.1093/bioinformatics/btq671

- The semantics of **field**s such as score, itemRgb, thick vs. thin positions, and block vs. non-block positions.
- The definitions of **custom field**s.
- Whether the **field separator** is a single tab character.

5 UCSC track files

Track files are files that contain additional information intended for a visualization tool such as the UCSC Genome Browser.[14] Track files contain browser lines and track lines that precede lines from a file format supported by the Genome Browser.[15] Track files are not valid BED **file**s—valid BED **file**s must not have any browser or track lines. To distinguish between BED **file**s and track files, track files should use the file extension `.track`.

6 Acronyms

ASCII	American Standard Code for Information Interchange
BED	Browser Extensible Data
GA4GH	Global Alliance for Genomics and Health
regex	regular expression
UCSC	University of California, Santa Cruz

7 Acknowledgments

We thank W. James Kent and the UCSC Genome Browser team for creating the BED format. We thank W. James Kent and Hiram Clawson (UCSC); Eric Roberts (University Health Network); John Marshall (University of Glasgow); Aaron R. Quinlan and Brent S. Pedersen (University of Utah); Ting Wang (Washington University in St. Louis); Daniel Perrett and Simon Brent (Wellcome Sanger Institute); Jasper Saris (Erasmus Medical Center); Zhenyu Zhang (University of Chicago); Andrew Yates (EMBL—European Bioinformatics Institute); Michael Schatz (Johns Hopkins University); Igor Dolgalev (New York University); Colin Diesh (University of California, Berkeley); Alex Reynolds (Altius Institute for Biomedical Sciences); Junjun Zhang (Ontario Institute for Cancer Research); and the Global Alliance for Genomics and Health (GA4GH) File Formats Task Team for comments on this specification.

[14] 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

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