Frequently Asked Questions: Data File Formats

General formats:

ENCODE-specific formats:

Download only formats:

Return to FAQ Table of Contents

BED format

Index ▷

BED (Browser Extensible Data) format provides a flexible way to define the data lines that are displayed in an annotation track. BED lines have three required fields and nine additional optional fields. The number of fields per line must be consistent throughout any single set of data in an annotation track. The order of the optional

fields is binding: lower-numbered fields must always be populated if higher-numbered fields are used.

BED information should not be mixed as explained above (BED3 should not be mixed with BED4), rather additional column information must be filled for consistency, for example with a "." in some circumstances, if the field content is to be empty. BED fields in custom tracks can be whitespace-delimited or tab-delimited. Only some variations of BED types, such as bedDetail, require a tab character delimitation for the detail columns.

Please note that only in custom tracks can the first lines of the file consist of header lines, which begin with the word "browser" or "track" to assist the browser in the display and interpretation of the lines of BED data following the headers. Such annotation track header lines are not permissible in downstream utilities such as bedToBigBed which convert lines of BED text to indexed binary files.

If your data set is BED-like, but it is very large (over 50MB) and you would like to keep it on your own server, you should use the bigBeddata format.

The first three required BED fields are:

1. **chrom** - The name of the chromosome (e.g. chr3, chrY, chr2_random) or scaffold (e.g. scaffold10671).
2. **chromStart** - The starting position of the feature in the chromosome or scaffold. The first base in a chromosome is numbered 0.
3. **chromEnd** - The ending position of the feature in the chromosome or scaffold. The *chromEnd* base is not included in the display of the feature. For example, the first 100 bases of a chromosome are defined as *chromStart=0, chromEnd=100*, and span the bases numbered 0-99.

The 9 additional optional BED fields are:

4. **name** - Defines the name of the BED line. This label is displayed to the left of the BED line in the Genome Browser window when the track is open to full display mode or directly to the left of the item in pack mode.
5. **score** - A score between 0 and 1000. If the track line *useScore* attribute is set to 1 for this annotation data set, the *score* value will determine the level of gray in which this feature is displayed (higher numbers = darker gray). This table shows the Genome Browser's translation of BED score values into shades of gray:

shade								
score in range	≤ 166	167-277	278-388	389-499	500-611	612-722	723-833	834-944	≥ 945

6. **strand** - Defines the strand. Either "." (=no strand) or "+" or "-".
7. **thickStart** - The starting position at which the feature is drawn thickly (for example, the start codon in gene displays). When there is no thick part, thickStart and thickEnd are usually set to the chromStart position.
8. **thickEnd** - The ending position at which the feature is drawn thickly (for example, the stop codon in gene displays).
9. **itemRgb** - An RGB value of the form R,G,B (e.g. 255,0,0). If the track line *itemRgb* attribute is set to "On", this RBG value will determine the display color of the data contained in this BED line. NOTE: It is recommended that a simple color scheme (eight colors or less) be used with this attribute to avoid overwhelming the color resources of the Genome Browser and your Internet browser.
10. **blockCount** - The number of blocks (exons) in the BED line.
11. **blockSizes** - A comma-separated list of the block sizes. The number of items in this list should correspond to *blockCount*.
12. **blockStarts** - A comma-separated list of block starts. All of the *blockStart* positions should be calculated relative to *chromStart*. The number of items in this list should correspond to *blockCount*.

In BED files with block definitions, the first *blockStart* value must be 0, so that the first block begins at *chromStart*. Similarly, the final*blockStart* position plus the final *blockSize* value must equal *chromEnd*. Blocks may not overlap.

Example:
Here's an example of an annotation track, introduced by a header line, that is followed by a complete BED

definition:

```
track name=pairedReads description="Clone Paired Reads" useScore=1
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
```

Example:
This example shows an annotation track that uses the itemRgb attribute to individually color each data line. In this track, the color scheme distinguishes between items named "Pos*" and those named "Neg*". See the usage note in the *itemRgb* description above for color palette restrictions. NOTE: The track and data lines in this example have been reformatted for documentation purposes. Thisexample can be pasted into the browser without editing.

```
browser position chr7:127471196-127495720
browser hide all
track name="ItemRGBDemo" description="Item RGB demonstration" visibility=2
itemRgb="On"
chr7    127471196  127472363  Pos1  0  +  127471196  127472363  255,0,0
chr7    127472363  127473530  Pos2  0  +  127472363  127473530  255,0,0
chr7    127473530  127474697  Pos3  0  +  127473530  127474697  255,0,0
chr7    127474697  127475864  Pos4  0  +  127474697  127475864  255,0,0
chr7    127475864  127477031  Neg1  0  -  127475864  127477031  0,0,255
chr7    127477031  127478198  Neg2  0  -  127477031  127478198  0,0,255
chr7    127478198  127479365  Neg3  0  -  127478198  127479365  0,0,255
chr7    127479365  127480532  Pos5  0  +  127479365  127480532  255,0,0
chr7    127480532  127481699  Neg4  0  -  127480532  127481699  0,0,255
```

Click here to display this track in the Genome Browser.

Example:
It is also possible to color items by strand in a BED track using the *colorByStrand* attribute in the track line as shown below. For BED tracks, this attribute functions only for custom tracks with 6 to 8 fields (i.e. BED6 through BED8). NOTE: The track and data lines in this example have been reformatted for documentation purposes. This example can be pasted into the browser without editing.

```
browser position chr7:127471196-127495720
browser hide all
track name="ColorByStrandDemo" description="Color by strand demonstration"
visibility=2 colorByStrand="255,0,0 0,0,255"
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  -
```

Click here to display this track in the Genome Browser.

bigBed format

Index ▷

The bigBed format stores annotation items that can either be simple, or a linked collection of exons, much as bed files do. BigBed files are created initially from bed type files, using the program `bedToBigBed`. The resulting bigBed files are in an indexed binary format. The main advantage of the bigBed files is that only the portions of the files needed to display a particular region are transferred to UCSC, so for large data sets bigBed is considerably faster than regular bed files. The bigBed file remains on your web accessible server (http, https, or ftp), not on the UCSC server.

Click here for more information on the bigBed format.

BED detail format

Index ▷

This is an extension of BED format. BED detail uses the first 4 to 12 columns of BED format, plus 2 additional fields that are used to enhance the track details pages. The first additional field is an ID, which can be used in place of the name field for creating links from the details pages. The second additional field is a description of the item, which can be a long description and can consist of html, including tables and lists.

Requirements for BED detail custom tracks are: fields must be tab-separated, "type=bedDetail" must be included in the track line, and the name and position fields should uniquely describe items so that the correct ID and description will be displayed on the details pages.

Example:
This example uses the first 4 columns of BED format, but up to 12 may be used. Click here to view this track in the Genome Browser.

```
track name=HbVar type=bedDetail description="HbVar custom track" db=hg19 visibility=3 url="http://globin.bx.psu.edu/cgi-bin/
chr11   5246919 5246920 Hb_North_York   2619    Hemoglobin variant
chr11   5255660 5255661 HBD c.1 G>A     2659    delta0 thalassemia
chr11   5247945 5247946 Hb Sheffield    2672    Hemoglobin variant
chr11   5255415 5255416 Hb A2-Lyon      2676    Hemoglobin variant
chr11   5248234 5248235 Hb Aix-les-Bains        2677    Hemoglobin variant
```

bedGraph format

Index ▷

The bedGraph format allows display of continuous-valued data in track format. This display type is useful for probability scores and transcriptome data. This track type is similar to the WIG format, but unlike the WIG format, data exported in the bedGraph format are preserved in their original state. This can be seen on export using the table browser. For more information about the bedGraph format, please see the bedGraph details page.

If you have a very large data set and you would like to keep it on your own server, you should use the bigWig format.

PSL format

Index ▷

PSL lines represent alignments, and are typically taken from files generated by BLAT or psLayout. See the BLAT documentation for more details. All of the following fields are required on each data line within a PSL file:

1. **matches** - Number of bases that match that aren't repeats
2. **misMatches** - Number of bases that don't match
3. **repMatches** - Number of bases that match but are part of repeats
4. **nCount** - Number of 'N' bases
5. **qNumInsert** - Number of inserts in query
6. **qBaseInsert** - Number of bases inserted in query
7. **tNumInsert** - Number of inserts in target
8. **tBaseInsert** - Number of bases inserted in target
9. **strand** - '+' or '-' for query strand. For translated alignments, second '+'or '-' is for genomic strand
10. **qName** - Query sequence name
11. **qSize** - Query sequence size
12. **qStart** - Alignment start position in query
13. **qEnd** - Alignment end position in query
14. **tName** - Target sequence name
15. **tSize** - Target sequence size
16. **tStart** - Alignment start position in target
17. **tEnd** - Alignment end position in target
18. **blockCount** - Number of blocks in the alignment (a block contains no gaps)
19. **blockSizes** - Comma-separated list of sizes of each block
20. **qStarts** - Comma-separated list of starting positions of each block in query
21. **tStarts** - Comma-separated list of starting positions of each block in target

Example:
Here is an example of an annotation track in PSL format. Note that line breaks have been inserted into the PSL lines in this example for documentation display purposes. This example can be pasted into the browser without

editing.

```
browser position chr22:13073000-13074000
browser hide all
track name=fishBlats description="Fish BLAT" visibility=2
useScore=1
59 9 0 0 1 823 1 96 +- FS_CONTIG_48080_1 1955 171 1062 chr22
    47748585 13073589 13073753 2 48,20,  171,1042,  34674832,34674976,
59 7 0 0 1 55 1 55 +- FS_CONTIG_26780_1 2825 2456 2577 chr22
    47748585 13073626 13073747 2 21,45,  2456,2532,  34674838,34674914,
59 7 0 0 1 55 1 55 -+ FS_CONTIG_26780_1 2825 2455 2676 chr22
    47748585 13073727 13073848 2 45,21,  249,349,  13073727,13073827,
```

Click here to display this track in the Genome Browser.

Be aware that the coordinates for a negative strand in a PSL line are handled in a special way. In the *qStart* and *qEnd* fields, the coordinates indicate the position where the query matches from the point of view of the forward strand, even when the match is on the reverse strand. However, in the *qStarts* list, the coordinates are reversed.

Example:
Here is a 61-mer containing 2 blocks that align on the minus strand and 2 blocks that align on the plus strand (this sometimes happens due to assembly errors):

```
0         1         2         3         4         5         6 tens position in query
01234567890123456789012345678901234567890123456789012345678901 ones position in query
                     ++++++++++++++                   +++++ plus strand alignment on query
    ------------------             --------------------      minus strand alignment on query
09876543210987654321098765432109876543210987654321098765432109 ones position in query negative strand coordinates
6         5         4         3         2         1         0 tens position in query negative strand coordinates

Plus strand:
     qStart=22
     qEnd=61
     blockSizes=14,5
     qStarts=22,56

Minus strand:
     qStart=4
     qEnd=56
     blockSizes=20,18
     qStarts=5,39
```

Essentially, the minus strand *blockSizes* and *qStarts* are what you would get if you reverse-complemented the query. However, the*qStart* and *qEnd* are not reversed. Use the following formulas to convert one to the other:

```
     Negative-strand-coordinate-qStart = qSize - qEnd   = 61 - 56 =  5
     Negative-strand-coordinate-qEnd   = qSize - qStart = 61 -  4 = 57
```

BLAT this actual sequence against hg19 for a real-world example:

CCCC
GGGTAAAATGAGTTTTTT
GGTCCAATCTTTTA
ATCCACTCCCTACCCTCCTA
GCAAG

Look for the alignment on the negative strand (-) of chr21, which conveniently aligns to the window chr21:10,000,001-10,000,061.

Browser window coordinates are 1-based [start,end] while PSL coordinates are 0-based [start,end), so a start of 10,000,001 in the browser corresponds to a start of 10,000,000 in the PSL. Subtracting 10,000,000 from the target (chromosome) position in PSL gives the query negative strand coordinate above.

The 4, 14, and 5 bases at beginning, middle, and end were chosen to not match with the genome at the corresponding position.

Protein Query:
A protein query consists of amino acids. To align amino acids against a database of nucleic acids, each target chromosome is first translated into amino acids for each of the six different reading frames. The resulting protein PSL is a hybrid; the query fields are all in amino acid coordinates and sizes, while the target database

fields are in nucleic acid chromosome coordinates and sizes. The fields shared by query and target are blockCount and blockSizes. But blockSizes differ between query (AA) and target (NA), so a single field cannot represent both. A choice was therefore made to report the blockSizes field in amino acids since it is a protein query.

To find the size of a target exon in nucleic acids, use the formula **blockSizes[exonNumber]*3**.
Or, to find the end position of a target exon, use the formula **tStarts[exonNumber] + (blockSizes[exonNumber]*3)**.

GFF format

Index ▷

GFF (General Feature Format) lines are based on the Sanger GFF2 specification. GFF lines have nine required fields that *must* be tab-separated. If the fields are separated by spaces instead of tabs, the track will not display correctly. For more information on GFF format, refer to Sanger's GFF page.

Note that there is also a GFF3 specification that is not currently supported by the Browser. All GFF tracks must be formatted according to Sanger's GFF2 specification.

If you would like to obtain browser data in GFF (GTF) format, please refer to Genes in gtf or gff format on the Wiki.

Here is a brief description of the GFF fields:

1. **seqname** - The name of the sequence. Must be a chromosome or scaffold.
2. **source** - The program that generated this feature.
3. **feature** - The name of this type of feature. Some examples of standard feature types are "CDS", "start_codon", "stop_codon", and "exon".
4. **start** - The starting position of the feature in the sequence. The first base is numbered 1.
5. **end** - The ending position of the feature (inclusive).
6. **score** - A score between 0 and 1000. If the track line *useScore* attribute is set to 1 for this annotation data set, the *score* value will determine the level of gray in which this feature is displayed (higher numbers = darker gray). If there is no score value, enter ".".
7. **strand** - Valid entries include '+', '-', or '.' (for don't know/don't care).
8. **frame** - If the feature is a coding exon, *frame* should be a number between 0-2 that represents the reading frame of the first base. If the feature is not a coding exon, the value should be '.'.
9. **group** - All lines with the same group are linked together into a single item.

Example:
Here's an example of a GFF-based track. This example can be pasted into the browser without editing. NOTE: Paste operations on some operating systems will replace tabs with spaces, which will result in an error when the GFF track is uploaded. You can circumvent this problem by pasting the URL of the above example (http://genome.ucsc.edu/goldenPath/help/regulatory.txt) instead of the text itself into the custom annotation track text box. If you encounter an error when loading a GFF track, check that the data lines contain tabs rather than spaces.

```
browser position chr22:10000000-10025000
browser hide all
track name=regulatory description="TeleGene(tm) Regulatory Regions"
visibility=2
chr22  TeleGene enhancer  10000000  10001000  500 +  .  touch1
chr22  TeleGene promoter  10010000  10010100  900 +  .  touch1
chr22  TeleGene promoter  10020000  10025000  800 -  .  touch2
```

Click here to display this track in the Genome Browser.

GTF format

Index ▷

GTF (Gene Transfer Format) is a refinement to GFF that tightens the specification. The first eight GTF fields are the same as GFF. The*group* field has been expanded into a list of *attributes*. Each attribute consists of a type/value pair. Attributes must end in a semi-colon, and be separated from any following attribute by exactly

one space.

The attribute list must begin with the two mandatory attributes:

- **gene_id *value*** - A globally unique identifier for the genomic source of the sequence.
- **transcript_id *value*** - A globally unique identifier for the predicted transcript.

Example:
Here is an example of the ninth field in a GTF data line:

```
gene_id "Em:U62317.C22.6.mRNA"; transcript_id "Em:U62317.C22.6.mRNA"; exon_number 1
```

The Genome Browser groups together GTF lines that have the same *transcript_id* value. It only looks at features of type *exon* and *CDS*.

For more information on this format, see http://mblab.wustl.edu/GTF2.html.

If you would like to obtain browser data in GTF format, please refer to Genes in gtf or gff format on the Wiki.

HAL format

Index ▷

HAL is a graph-based structure to efficiently store and index multiple genome alignments and ancestral reconstructions. HAL files are represented in HDF5 format, an open standard for storing and indexing large, compressed scientific data sets. Genomes within HAL are organized according to the phylogenetic tree that relate them: each genome is segmented into pairwise DNA alignment blocks with respect to its parent and children (if present) in the tree. Note that if the phylogeny is unknown, a star tree can be used. The modularity provided by this tree-based decomposition allows for efficient querying of sub-alignments, as well as the ability to add, remove and update genomes within the alignment with only local modifications to the structure. Another important feature of HAL is reference independence: alignments in this format can be queried with respect to the coordinates of any genome they contain.

HAL files can be created or read with a comprehensive C++ API (click here for source code and manual). A set of command line tools is included to perform basic operations, such as importing and exporting data, identifying mutations, coordinate mapping (liftOver), and comparative assembly hub generation.

HAL is the native output format of the Progressive Cactus alignment pipeline, and is included in the Progressive Cactus installation package.

MAF format

Index ▷

The multiple alignment format stores a series of multiple alignments in a format that is easy to parse and relatively easy to read. This format stores multiple alignments at the DNA level between entire genomes. Previously used formats are suitable for multiple alignments of single proteins or regions of DNA without rearrangements, but would require considerable extension to cope with genomic issues such as forward and reverse strand directions, multiple pieces to the alignment, and so forth.

General Structure

The *.maf* format is line-oriented. Each multiple alignment ends with a blank line. Each sequence in an alignment is on a single line, which can get quite long, but there is no length limit. Words in a line are delimited by any white space. Lines starting with # are considered to be comments. Lines starting with ## can be ignored by most programs, but contain meta-data of one form or another.

The file is divided into paragraphs that terminate in a blank line. Within a paragraph, the first word of a line indicates its type. Each multiple alignment is in a separate paragraph that begins with an "a" line and contains an "s" line for each sequence in the multiple alignment. Some MAF files may contain other optional line types:

- an "i" line containing information about what is in the aligned species DNA before and after the immediately preceding "s" line
- an "e" line containing information about the size of the gap between the alignments that span the current block

- a "q" line indicating the quality of each aligned base for the species

Parsers may ignore any other types of paragraphs and other types of lines within an alignment paragraph.

Custom Tracks

The first line of a custom MAF track must be a "track" line that contains a name=value pair specifying the track name. Here is an example of a minimal track line:

```
track name=sample
```

The following variables can be specified in the track line of a custom MAF:

- **name=sample** - Required. Name the track
- **description="Sample Track"** - Optional. Gives a long name for the track
- **frames=multiz28wayFrames** - Optional. Tells the browser which table to grab the gene frames from. This is usually associated with an N-way alignment where the name ends in the string "Frames"
- **mafDot=on** - Optional. Use dots instead of bases when bases are identical
- **visibility=dense|pack|full** - Optional. Sets the default visibility mode for this track.
- **speciesOrder="hg18 panTro2"** - Optional. White-space separated list specifying the order in which the sequences in the maf should be displayed.

The second line of a custom MAF track must be a header line as described below.

Header Line

The first line of a *.maf* file begins with ##maf. This word is followed by white-space-separated variable=value pairs. There should be *no*white space surrounding the "=".

```
##maf version=1 scoring=tba.v8
```

The currently defined variables are:

- **version** - Required. Currently set to one.
- **scoring** - Optional. A name for the scoring scheme used for the alignments. The current scoring schemes are:
  - *bit* - roughly corresponds to blast bit values (roughly 2 points per aligning base minus penalties for mismatches and inserts).
  - *blastz* - blastz scoring scheme -- roughly 100 points per aligning base.
  - *probability* - some score normalized between 0 and 1.
- **program** - Optional. Name of the program generating the alignment.

Undefined variables are ignored by the parser.

The header line is usually followed by a comment line (it begins with a #) that describes the parameters that were used to run the alignment program.

```
# tba.v8 (((human chimp) baboon) (mouse rat))
```

Alignment Block Lines (lines starting with 'a' -- parameters for a new alignment block)

```
a score=23262.0
```

Each alignment begins with an 'a' line that set variables for the entire alignment block. The 'a' is followed by name=value pairs. There are no required name=value pairs. The currently defined variables are:

- score -- Optional. Floating point score. If this is present, it is good practice to also define scoring in the first line.
- pass -- Optional. Positive integer value. For programs that do multiple pass alignments such as blastz, this shows which pass this alignment came from. Typically, pass 1 will find the strongest alignments genome-wide, and pass 2 will find weaker alignments between two first-pass alignments.

Lines starting with 's' -- a sequence within an alignment block

```
s hg16.chr7    27707221 13 + 158545518 gcagctgaaaaca
s panTro1.chr6 28869787 13 + 161576975 gcagctgaaaaca
s baboon         249182 13 +   4622798 gcagctgaaaaca
s mm4.chr6     53310102 13 + 151104725 ACAGCTGAAAATA
```

The 's' lines together with the 'a' lines define a multiple alignment. The 's' lines have the following fields which are defined by position rather than name=value pairs.

- src -- The name of one of the source sequences for the alignment. For sequences that are resident in a browser assembly, the form 'database.chromosome' allows automatic creation of links to other assemblies. Non-browser sequences are typically reference by the species name alone.
- start -- The start of the aligning region in the source sequence. This is a zero-based number. If the strand field is '-' then this is the start relative to the reverse-complemented source sequence (see Coordinate Transforms).
- size -- The size of the aligning region in the source sequence. This number is equal to the number of non-dash characters in the alignment text field below.
- strand -- Either '+' or '-'. If '-', then the alignment is to the reverse-complemented source.
- srcSize -- The size of the entire source sequence, not just the parts involved in the alignment.
- text -- The nucleotides (or amino acids) in the alignment and any insertions (dashes) as well.

Lines starting with 'i' -- information about what's happening before and after this block in the aligning species

```
s hg16.chr7    27707221 13 + 158545518 gcagctgaaaaca
s panTro1.chr6 28869787 13 + 161576975 gcagctgaaaaca
i panTro1.chr6 N 0 C 0
s baboon         249182 13 +   4622798 gcagctgaaaaca
i baboon       I 234 n 19
```

The 'i' lines contain information about the context of the sequence lines immediately preceeding them. The following fields are defined by position rather than name=value pairs:

- src -- The name of the source sequence for the alignment. Should be the same as the 's' line immediately above this line.
- leftStatus -- A character that specifies the relationship between the sequence in this block and the sequence that appears in the previous block.
- leftCount -- Usually the number of bases in the aligning species between the start of this alignment and the end of the previous one.
- rightStatus -- A character that specifies the relationship between the sequence in this block and the sequence that appears in the subsequent block.
- rightCount -- Usually the number of bases in the aligning species between the end of this alignment and the start of the next one.

The status characters can be one of the following values:

- C -- the sequence before or after is contiguous with this block.
- I -- there are bases between the bases in this block and the one before or after it.
- N -- this is the first sequence from this src chrom or scaffold.
- n -- this is the first sequence from this src chrom or scaffold but it is bridged by another alignment from a different chrom or scaffold.
- M -- there is missing data before or after this block (Ns in the sequence).
- T -- the sequence in this block has been used before in a previous block (likely a tandem duplication)

Lines starting with 'e' -- information about empty parts of the alignment block

```
s hg16.chr7    27707221 13 + 158545518 gcagctgaaaaca
e mm4.chr6     53310102 13 + 151104725 I
```

The 'e' lines indicate that there isn't aligning DNA for a species but that the current block is bridged by a chain that connects blocks before and after this block. The following fields are defined by position rather than name=value pairs.

- src -- The name of one of the source sequences for the alignment.
- start -- The start of the non-aligning region in the source sequence. This is a zero-based number. If the strand field is '-' then this is the start relative to the reverse-complemented source sequence (see Coordinate Transforms).
- size -- The size in base pairs of the non-aligning region in the source sequence.
- strand -- Either '+' or '-'. If '-', then the alignment is to the reverse-complemented source.
- srcSize -- The size of the entire source sequence, not just the parts involved in the alignment. alignment and any insertions (dashes) as well.
- status -- A character that specifies the relationship between the non-aligning sequence in this block and the sequence that appears in the previous and subsequent blocks.

The status character can be one of the following values:

- C -- the sequence before and after is contiguous implying that this region was either deleted in the source or inserted in the reference sequence. The browser draws a single line or a '-' in base mode in these blocks.
- I -- there are non-aligning bases in the source species between chained alignment blocks before and after this block. The browser shows a double line or '=' in base mode.
- M -- there are non-aligning bases in the source and more than 90% of them are Ns in the source. The browser shows a pale yellow bar.
- n -- there are non-aligning bases in the source and the next aligning block starts in a new chromosome or scaffold that is bridged by a chain between still other blocks. The browser shows either a single line or a double line based on how many bases are in the gap between the bridging alignments.

Lines starting with 'q' -- information about the quality of each aligned base for the species

```
s hg18.chr1                  32741  26 + 247249719 TTTTTGAAAAACAAACAACAAGTTGG
s panTro2.chrUn            9697231  26 +  58616431 TTTTTGAAAAACAAACAACAAGTTGG
q panTro2.chrUn                                   99999999999999999999999999
s dasNov1.scaffold_179265     1474   7 +      4584 TT----------AAGCA---------
q dasNov1.scaffold_179265                         99----------32239---------
```

The 'q' lines contain a compressed version of the actual raw quality data, representing the quality of each aligned base for the species with a single character of 0-9 or F. The following fields are defined by position rather than name=value pairs:

- src -- The name of the source sequence for the alignment. Should be the same as the 's' line immediately preceding this line.
- value -- A MAF quality value corresponding to the aligning nucleotide acid in the preceding 's' line. Insertions (dashes) in the preceding 's' line are represented by dashes in the 'q' line as well. The quality value can be 'F' (finished sequence) or a number derived from the actual quality scores (which range from 0-97) or the manually assigned score of 98. These numeric values are calculated as:

  ```
  MAF quality value = min( floor(actual quality value/5), 9 )
  ```

  This results in the following mapping:

MAF quality value	Raw quality score range	Quality level
0-8	0-44	Low
9	45-97	High
0	98	Manually assigned
F	99	Finished

A Simple Example

Here is a simple example of a three alignment blocks derived from five starting sequences. The first **track** line is necessary for custom tracks, but should be removed otherwise. Repeats are shown as lowercase, and each block may have a subset of the input sequences. All sequence columns and rows must contain at least one nucleotide (no columns or rows that contain only insertions).

```
track name=euArc visibility=pack
##maf version=1 scoring=tba.v8
# tba.v8 (((human chimp) baboon) (mouse rat))

a score=23262.0
s hg18.chr7     27578828 38 + 158545518 AAA-GGGAATGTTAACCAAATGA---ATTGTCTCTTACGGTG
s panTro1.chr6 28741140 38 + 161576975 AAA-GGGAATGTTAACCAAATGA---ATTGTCTCTTACGGTG
s baboon          116834 38 +   4622798 AAA-GGGAATGTTAACCAAATGA---GTTGTCTCTTATGGTG
s mm4.chr6      53215344 38 + 151104725 -AATGGGAATGTTAAGCAAACGA---ATTGTCTCTCAGTGTG
s rn3.chr4      81344243 40 + 187371129 -AA-GGGGATGCTAAGCCAATGAGTTGTTGTCTCTCAATGTG

a score=5062.0
s hg18.chr7     27699739 6 + 158545518 TAAAGA
s panTro1.chr6 28862317 6 + 161576975 TAAAGA
s baboon          241163 6 +   4622798 TAAAGA
s mm4.chr6      53303881 6 + 151104725 TAAAGA
s rn3.chr4      81444246 6 + 187371129 taagga

a score=6636.0
s hg18.chr7     27707221 13 + 158545518 gcagctgaaaaca
s panTro1.chr6 28869787 13 + 161576975 gcagctgaaaaca
s baboon          249182 13 +   4622798 gcagctgaaaaca
s mm4.chr6      53310102 13 + 151104725 ACAGCTGAAAATA
```

BAM format

Index ▷

BAM is the compressed binary version of the Sequence Alignment/Map (SAM) format, a compact and index-able representation of nucleotide sequence alignments. Many next-generation sequencing and analysis tools work with SAM/BAM. For custom track display, the main advantage of indexed BAM over PSL and other human-readable alignment formats is that only the portions of the files needed to display a particular region are transferred to UCSC. This makes it possible to display alignments from files that are so large that the connection to UCSC would time out when attempting to upload the whole file to UCSC. Both the BAM file and its associated index file remain on your web-accessible server (http or ftp), not on the UCSC server. UCSC temporarily caches the accessed portions of the files to speed up interactive display.

Click here for more information about BAM custom tracks.

sam – Sequence Alignment/Map file format

1	QNAME	Query template/pair NAME
2	FLAG	bitwise FLAG
3	RNAME	Reference sequence NAME
4	POS	1-based leftmost POSition/coordinate of clipped sequence
5	MAPQ	MAPping Quality (Phred-scaled)
6	CIGAR	extended CIGAR string
7	MRNM	Mate Reference sequence NaMe (`=' if same as RNAME)
8	MPOS	1-based Mate POSistion
9	TLEN	inferred Template LENgth (insert size)
10	SEQ	query SEQuence on the same strand as the reference
11	QUAL	query QUALity (ASCII-33 gives the Phred base quality)
12+	OPT	variable OPTional fields in the format TAG:VTYPE:VALUE

Each bit in the FLAG field is defined as:

0x0001	p	the read is paired in sequencing
0x0002	P	the read is mapped in a proper pair
0x0004	u	the query sequence itself is unmapped
0x0008	U	the mate is unmapped
0x0010	r	strand of the query (1 for reverse)
0x0020	R	strand of the mate
0x0040	1	the read is the first read in a pair
0x0080	2	the read is the second read in a pair
0x0100	s	the alignment is not primary
0x0200	f	the read fails platform/vendor quality checks
0x0400	d	the read is either a PCR or an optical duplicate
0x0800	S	the alignment is supplementary

where the second column gives the string representation of the FLAG field.

CRAM format

Index ▷

The CRAM file format is a more dense form of BAM files with the benefit of saving much disk space. While BAM files contain all sequence data within a file, CRAM files are smaller by taking advantage of an additional external "reference sequence" file. This file is needed to both compress and decompress the read information.

Click here for more information on the CRAM format.

WIG format

Index ▷

Wiggle format (WIG) allows the display of continuous-valued data in a track format. Click here for more information.

bigWig format

Index ▷

The bigWig format is for display of dense, continuous data that will be displayed in the Genome Browser as a graph. BigWig files are created initially from wiggle (wig) type files, using the program `wigToBigWig`. Alternatively, bigWig files can be created from bedGraph files, using the program `bedGraphToBigWig`. In either case, the resulting

bigWig files are in an indexed binary format. The main advantage of the bigWig files is that only the portions of the files needed to display a particular region are transferred to UCSC, so for large data sets bigWig is considerably faster than regular wiggle files. The bigWig file remains on your web accessible server (http, https, or ftp), not on the UCSC server. Only the portion that is needed for the chromosomal position you are currently viewing is locally cached as a "sparse file".

Click here for more information on the bigWig format.

Microarray format

Index ▷

The datasets for the built-in microarray tracks in the Genome Browser are stored in BED15 format, an extension of BED format that includes three additional fields: expCount, expIds, and expScores. To display correctly in the Genome Browser, microarray tracks require the setting of several attributes in the trackDb file associated with the track's genome assembly. Each microarray track set must also have an associated microarrayGroups.ra configuration file that contains additional information about the data in each of the arrays.

User-created microarray custom tracks are similar in format to BED custom tracks with the addition of three required track line parameters in the header--expNames, expScale, and expStep--that mimic the trackDb and microarrayGroups.ra settings of built-in microarray tracks.

For a complete description of the microarray track format and an explanation of how to construct a microarray custom track, see theGenome Browser Wiki.

.2bit format

Index ▷

A .2bit file stores multiple DNA sequences (up to 4 Gb total) in a compact randomly-accessible format. The file contains masking information as well as the DNA itself.

The file begins with a 16-byte header containing the following fields:

- signature - the number 0x1A412743 in the architecture of the machine that created the file
- version - zero for now. Readers should abort if they see a version number higher than 0.
- sequenceCount - the number of sequences in the file.
- reserved - always zero for now

All fields are 32 bits unless noted. If the signature value is not as given, the reader program should byte-swap the signature and check if the swapped version matches. If so, all multiple-byte entities in the file will have to be byte-swapped. This enables these binary files to be used unchanged on different architectures.

The header is followed by a file index, which contains one entry for each sequence. Each index entry contains three fields:

- nameSize - a byte containing the length of the name field
- name - the sequence name itself, of variable length depending on nameSize
- offset - the 32-bit offset of the sequence data relative to the start of the file

The index is followed by the sequence records, which contain nine fields:

- dnaSize - number of bases of DNA in the sequence
- nBlockCount - the number of blocks of Ns in the file (representing unknown sequence)
- nBlockStarts - an array of length nBlockCount of 32 bit integers indicating the starting position of a block of Ns
- nBlockSizes - an array of length nBlockCount of 32 bit integers indicating the length of a block of Ns
- maskBlockCount - the number of masked (lower-case) blocks
- maskBlockStarts - an array of length maskBlockCount of 32 bit integers indicating the starting position of a masked block
- maskBlockSizes - an array of length maskBlockCount of 32 bit integers indicating the length of a masked block

- reserved - always zero for now
- packedDna - the DNA packed to two bits per base, represented as so: T - 00, C - 01, A - 10, G - 11. The first base is in the most significant 2-bit byte; the last base is in the least significant 2 bits. For example, the sequence TCAG is represented as 00011011.

For a complete definition of all fields in the twoBit format, see this description in the source code.

.nib format

Index ▷

The .nib format pre-dates the .2bit format and is less compact. It describes a DNA sequence by packing two bases into each byte. Each .nib file contains only a single sequence. The file begins with a 32-bit signature that is 0x6BE93D3A in the architecture of the machine that created the file (or possibly a byte-swapped version of the same number on another machine). This is followed by a 32-bit number in the same format that describes the number of bases in the file. Next, the bases themselves are listed, packed two bases to the byte. The first base is packed in the high-order 4 bits (nibble); the second base is packed in the low-order four bits:

```
byte = (base1<<4) + base2
```

The numerical representations for the bases are:

- 0 - T
- 1 - C
- 2 - A
- 3 - G
- 4 - N (unknown)

The most significant bit in a nibble is set if the base is masked.

GenePred table format

Index ▷

genePred is a table format commonly used for gene prediction tracks in the Genome Browser. Variations of the genePred format are listed below.

If you would like to obtain browser data in GFF (GTF) format, please refer to Genes in gtf or gff format on the Wiki.

Gene Predictions

The following definition is used for gene prediction tables. In alternative-splicing situations, each transcript has a row in this table.

```
table genePred
"A gene prediction."
    (
    string  name;               "Name of gene"
    string  chrom;              "Chromosome name"
    char[1] strand;             "+ or - for strand"
    uint    txStart;            "Transcription start position"
    uint    txEnd;              "Transcription end position"
    uint    cdsStart;           "Coding region start"
    uint    cdsEnd;             "Coding region end"
    uint    exonCount;          "Number of exons"
    uint[exonCount] exonStarts; "Exon start positions"
    uint[exonCount] exonEnds;   "Exon end positions"
    )
```

Gene Predictions (Extended)

The following definition is used for extended gene prediction tables. In alternative-splicing situations, each transcript has a row in this table. The refGene table is an example of the genePredExt format.

```
table genePredExt
"A gene prediction with some additional info."
    (
```

```
    string name;                "Name of gene (usually transcript_id from GTF)"
    string chrom;               "Chromosome name"
    char[1] strand;             "+ or - for strand"
    uint txStart;               "Transcription start position"
    uint txEnd;                 "Transcription end position"
    uint cdsStart;              "Coding region start"
    uint cdsEnd;                "Coding region end"
    uint exonCount;             "Number of exons"
    uint[exonCount] exonStarts; "Exon start positions"
    uint[exonCount] exonEnds;   "Exon end positions"
    int score;                  "Score"
    string name2;               "Alternate name (e.g. gene_id from GTF)"
    string cdsStartStat;        "enum('none','unk','incmpl','cmpl')"
    string cdsEndStat;          "enum('none','unk','incmpl','cmpl')"
    lstring exonFrames;         "Exon frame offsets {0,1,2}"
    )
```

Gene Predictions and RefSeq Genes with Gene Names

A version of genePred that associates the gene name with the gene prediction information. In alternative-splicing situations, each transcript has a row in this table.

```
table refFlat
"A gene prediction with additional geneName field."
    (
    string  geneName;           "Name of gene as it appears in Genome Browser."
    string  name;               "Name of gene"
    string  chrom;              "Chromosome name"
    char[1] strand;             "+ or - for strand"
    uint    txStart;            "Transcription start position"
    uint    txEnd;              "Transcription end position"
    uint    cdsStart;           "Coding region start"
    uint    cdsEnd;             "Coding region end"
    uint    exonCount;          "Number of exons"
    uint[exonCount] exonStarts; "Exon start positions"
    uint[exonCount] exonEnds;   "Exon end positions"
    )
```

bigGenePred table format

Index ▷

bigGenePred is a table format commonly used for gene prediction tracks in the Genome Browser. bigGenePred format is a superset of the genePred text-based format supported using the bigBed format, so it can be efficiently accessed over a network.

Click here for more information on the bigGenePred format.

bigPsl table format

Index ▷

bigPsl is a table format commonly used to store alignments in the Genome Browser. bigPsl format is a superset of the PSL text-based format supported using the bigBed format, so it can be efficiently accessed over a network.

Click here for more information on the bigPsl format.

bigMaf table format

Index ▷

bigMaf is a table format commonly used to store multiple alignments in the Genome Browser. bigMaf format is a superset of the MAFtext-based format supported using the bigBed format, so it can be efficiently accessed over a network.

Click here for more information on the bigMaf format.

bigChain table format

Index ▷

bigChain is a table format commonly used to store pairwise alignments in the Genome Browser. bigChain format is a superset of thechain text-based format supported using the bigBed format, so it can be efficiently

accessed over a network.

Click here for more information on the bigChain format.

Personal Genome SNP format

Index ▷

This format is for displaying SNPs from personal genomes. It is the same as is used for the Genome Variants and Population Variants tracks.

1. **chrom** - The name of the chromosome (e.g. chr3, chrY, chr2_random) or scaffold (e.g. scaffold10671).
2. **chromStart** - The starting position of the feature in the chromosome or scaffold. The first base in a chromosome is numbered 0.
3. **chromEnd** - The ending position of the feature in the chromosome or scaffold. The *chromEnd* base is not included in the display of the feature. For example, the first 100 bases of a chromosome are defined as *chromStart=0, chromEnd=100*, and span the bases numbered 0-99.
4. **name** - The allele or alleles, consisting of one or more A, C, T, or G, optionally followed by one or more '/' and another allele (there can be more than 2 alleles). A '-' can be used in place of a base to denote an insertion or deletion; if the position given is zero bases wide, it is an insertion. The alleles are expected to be for the plus strand.
5. **alleleCount** - The number of alleles listed in the name field.
6. **alleleFreq** - A comma-separated list of the frequency of each allele, given in the same order as the name field. If unknown, a list of zeroes (matching the alleleCount) should be used.
7. **alleleScores** - A comma-separated list of the quality score of each allele, given in the same order as the name field. If unknown, a list of zeroes (matching the alleleCount) should be used.

In the Genome Browser, when viewing the forward strand of the reference genome (the normal case), the displayed alleles are relative to the forward strand. When viewing the reverse strand of the reference genome (via the "<--" or "reverse" button), the displayed alleles are reverse-complemented to match the reverse strand. If the allele frequencies are given, the coloring of the box will reflect the frequency for each allele.

The details pages for this track type will automatically compute amino acid changes for coding SNPs as well as give a chart of amino acid properties if there is a non-synonymous change. (The Sift and PolyPhen predictions that are in some of the Genome Variants subtracks are not available.)

Example:
Here is an example of an annotation track in Personal Genome SNP format. The first SNP using a '-' is an insertion; the second is a deletion. The last 4 SNPs are in a coding region.

```
track type=pgSnp visibility=3 db=hg19 name="pgSnp" description="Personal Genome SNP example"
browser position chr21:31811924-31812937
chr21   31812007        31812008        T/G     2       21,70   90,70
chr21   31812031        31812032        T/G/A   3       9,60,7  80,80,30
chr21   31812035        31812035        -/CGG   2       20,80   0,0
chr21   31812088        31812093        -/CTCGG 2       30,70   0,0
chr21   31812277        31812278        T       1       15      90
chr21   31812771        31812772        A       1       36      80
chr21   31812827        31812828        A/T     2       15,5    0,0
chr21   31812879        31812880        C       1       0       0
chr21   31812915        31812916        -       1       0       0
```

VCF format

Index ▷

Variant Call Format (VCF) is a flexible and extendable format (now maintained by thega4gh) for variation data such as single nucleotide variants, insertions/deletions, copy number variants and structural variants. When a VCF file is compressed and indexed usingtabix, and made web-accesible, the Genome Browser can fetch only the portions of the file necessary to display items in the viewed region. This makes it possible to display alignments from files that are so large that the connection to UCSC would time out when attempting to upload the whole file to UCSC. Both the compressed VCF file and its tabix index file remain on your web-accessible server (http or ftp), not on the UCSC server. UCSC temporarily caches the accessed portions of the files to

speed up interactive display.

Go to the VCF Track Format page for more information about VCF custom tracks.

ENCODE RNA elements: BED6 + 3 scores format

Index ▷

1. **chrom** - Name of the chromosome (or contig, scaffold, etc.).
2. **chromStart** - The starting position of the feature in the chromosome or scaffold. The first base in a chromosome is numbered 0.
3. **chromEnd** - The ending position of the feature in the chromosome or scaffold. The *chromEnd* base is not included in the display of the feature. For example, the first 100 bases of a chromosome are defined as *chromStart=0, chromEnd=100*, and span the bases numbered 0-99.
4. **name** - Name given to a region (preferably unique). Use '.' if no name is assigned.
5. **score** - Indicates how dark the peak will be displayed in the browser (0-1000). If all scores were '0' when the data were submitted to the DCC, the DCC assigned scores 1-1000 based on signal value. Ideally the average signalValue per base spread is between 100-1000.
6. **strand** - +/- to denote strand or orientation (whenever applicable). Use '.' if no orientation is assigned.
7. **level** - Expression level such as RPKM or FPKM.
8. **signif** - Statistical significance such as IDR.
9. **score2** - Additional measurement/count e.g. number of reads.

ENCODE narrowPeak: Narrow (or Point-Source) Peaks format

Index ▷

This format is used to provide called peaks of signal enrichment based on pooled, normalized (interpreted) data. It is a BED6+4 format.

1. **chrom** - Name of the chromosome (or contig, scaffold, etc.).
2. **chromStart** - The starting position of the feature in the chromosome or scaffold. The first base in a chromosome is numbered 0.
3. **chromEnd** - The ending position of the feature in the chromosome or scaffold. The *chromEnd* base is not included in the display of the feature. For example, the first 100 bases of a chromosome are defined as *chromStart=0, chromEnd=100*, and span the bases numbered 0-99.
4. **name** - Name given to a region (preferably unique). Use '.' if no name is assigned.
5. **score** - Indicates how dark the peak will be displayed in the browser (0-1000). If all scores were '0' when the data were submitted to the DCC, the DCC assigned scores 1-1000 based on signal value. Ideally the average signalValue per base spread is between 100-1000.
6. **strand** - +/- to denote strand or orientation (whenever applicable). Use '.' if no orientation is assigned.
7. **signalValue** - Measurement of overall (usually, average) enrichment for the region.
8. **pValue** - Measurement of statistical significance (-log10). Use -1 if no pValue is assigned.
9. **qValue** - Measurement of statistical significance using false discovery rate (-log10). Use -1 if no qValue is assigned.
10. **peak** - Point-source called for this peak; 0-based offset from chromStart. Use -1 if no point-source called.

Here is an example of narrowPeak format:

```
track type=narrowPeak visibility=3 db=hg19 name="nPk" description="ENCODE narrowPeak Example"
browser position chr1:9356000-9365000
chr1    9356548 9356648 .       0       .       182     5.0945  -1  50
chr1    9358722 9358822 .       0       .       91      4.6052  -1  40
chr1    9361082 9361182 .       0       .       182     9.2103  -1  75
```

ENCODE broadPeak: Broad Peaks (or Regions) format

Index ▷

This format is used to provide called regions of signal enrichment based on pooled, normalized (interpreted) data. It is a BED 6+3 format.

1. **chrom** - Name of the chromosome (or contig, scaffold, etc.).
2. **chromStart** - The starting position of the feature in the chromosome or scaffold. The first base in a chromosome is numbered 0.
3. **chromEnd** - The ending position of the feature in the chromosome or scaffold. The *chromEnd* base is not included in the display of the feature. For example, the first 100 bases of a chromosome are defined as *chromStart=0, chromEnd=100*, and span the bases numbered 0-99. If all scores were '0' when the data were submitted to the DCC, the DCC assigned scores 1-1000 based on signal value. Ideally the average signalValue per base spread is between 100-1000.
4. **name** - Name given to a region (preferably unique). Use '.' if no name is assigned.
5. **score** - Indicates how dark the peak will be displayed in the browser (0-1000).
6. **strand** - +/- to denote strand or orientation (whenever applicable). Use '.' if no orientation is assigned.
7. **signalValue** - Measurement of overall (usually, average) enrichment for the region.
8. **pValue** - Measurement of statistical significance (-log10). Use -1 if no pValue is assigned.
9. **qValue** - Measurement of statistical significance using false discovery rate (-log10). Use -1 if no qValue is assigned.

Here is an example of broadPeak format:

```
track type=broadPeak visibility=3 db=hg19 name="bPk" description="ENCODE broadPeak Example"
browser position chr1:798200-800700
chr1     798256 798454 .       116     .       4.89716 3.70716 -1
chr1     799435 799507 .       103     .       2.46426 1.54117 -1
chr1     800141 800596 .       107     .       3.22803 2.12614 -1
```

ENCODE gappedPeak: Gapped Peaks (or Regions) format

Index ▷

This format is used to provide called regions of signal enrichment based on pooled, normalized (interpreted) data where the regions may be spliced or incorporate gaps in the genomic sequence. It is a BED12+3 format.

1. **chrom** - Name of the chromosome (or contig, scaffold, etc.).
2. **chromStart** - The starting position of the feature in the chromosome or scaffold. The first base in a chromosome is numbered 0.
3. **chromEnd** - The ending position of the feature in the chromosome or scaffold. The *chromEnd* base is not included in the display of the feature. For example, the first 100 bases of a chromosome are defined as *chromStart=0, chromEnd=100*, and span the bases numbered 0-99.
4. **name** - Name given to a region (preferably unique). Use '.' if no name is assigned.
5. **score** - Indicates how dark the peak will be displayed in the browser (0-1000). If all scores were '0' when the data were submitted to the DCC, the DCC assigned scores 1-1000 based on signal value. Ideally the average signalValue per base spread is between 100-1000.
6. **strand** - +/- to denote strand or orientation (whenever applicable). Use '.' if no orientation is assigned.
7. **thickStart** - The starting position at which the feature is drawn thickly. Not used in gappedPeak type, set to 0.
8. **thickEnd** - The ending position at which the feature is drawn thickly. Not used in gappedPeak type, set to 0.
9. **itemRgb** - An RGB value of the form R,G,B (e.g. 255,0,0). Not used in gappedPeak type, set to 0.
10. **blockCount** - The number of blocks (exons) in the BED line.
11. **blockSizes** - A comma-separated list of the block sizes. The number of items in this list should correspond to *blockCount*.
12. **blockStarts** - A comma-separated list of block starts. The first value must be 0 and all of the *blockStart* positions should be calculated relative to *chromStart*. The number of items in this list should correspond to *blockCount*.

13. **signalValue** - Measurement of overall (usually, average) enrichment for the region.
14. **pValue** - Measurement of statistical significance (-log10). Use -1 if no pValue is assigned.
15. **qValue** - Measurement of statistical significance using false discovery rate (-log10). Use -1 if no qValue is assigned.

Here is an example of gappedPeak format:

```
track name=gappedPeakExample type=gappedPeak
chr1 171000 171600 Anon_peak_1 55 . 0 0 0 2 400,100 0,500 4.04761 7.53255 5.52807
```

ENCODE tagAlign: BED3+3 format (historical)

Index ▷

tagAlign was used with hg18, but is no longer in use with hg19. Tag Alignment provided genomic mapping of short sequence tags. It is a BED3+3 format.

1. **chrom** - Name of the chromosome.
2. **chromStart** - The starting position of the feature in the chromosome. The first base in a chromosome is numbered 0.
3. **chromEnd** - The ending position of the feature in the chromosome or scaffold. The chromEnd base is not included in the display of the feature. For example, the first 100 bases of a chromosome are defined as chromStart=0, chromEnd=100, and span the bases numbered 0-99.
4. **sequence** - Sequence of this read.
5. **score** - Indicates uniqueness or quality (preferably 1000/alignmentCount).
6. **strand** - Orientation of this read (+ or -).

Here is an example of tagAlign format:

```
chrX 8823384 8823409 AGAAGGAAAATGATGTGAAGACATA 1000 +
chrX 8823387 8823412 TCTTATGTCTTCACATCATTTTCCT 500  -
```

ENCODE pairedTagAlign: BED6+2 format (historical)

Index ▷

pairedTagAlign was used with hg18, but is no longer in use with hg19. Tag Alignment Format for Paired Reads was used to provide genomic mapping of paired-read short sequence tags. It is a BED6+2 format.

1. **chrom** - Name of the chromosome.
2. **chromStart** - The starting position of the feature in the chromosome. The first base in a chromosome is numbered 0.
3. **chromEnd** - The ending position of the feature in the chromosome or scaffold. The chromEnd base is not included in the display of the feature. For example, the first 100 bases of a chromosome are defined as chromStart=0, chromEnd=100, and span the bases numbered 0-99.
4. **name** - Identifier of paired-read.
5. **score** - Indicates uniqueness or quality (preferably 1000/alignment-count).
6. **strand** - Orientation of this read (+ or -).
7. **seq1** - Sequence of first read.
8. **seq2** - Sequence of second read.

ENCODE peptideMapping: BED6+4 format

Index ▷

PeptideMapping. The peptide mapping format was used to provide genomic mapping of proteogenomic mappings of peptides to the genome, with information that is appropriate for assessing the confidence of the mapping.

1. **chrom** - Name of the chromosome.
2. **chromStart** - The starting position of the feature in the chromosome. The first base in a chromosome is numbered 0.
3. **chromEnd** - The ending position of the feature in the chromosome or scaffold. The chromEnd base is not included in the display of the feature. For example, the first 100 bases of a chromosome are defined as chromStart=0, chromEnd=100, and span the bases numbered 0-99.
4. **name** - The peptide sequence.
5. **score** - Indicates uniqueness or quality (preferably 1000/alignment-count).
6. **strand** - Orientation of this read (+ or -).
7. **rawScore** - Raw score for this hit, as estimated through HMM analysis.
8. **spectrumId** - Non-unique identifier for the spectrum file
9. **peptideRank** - Rank of this hit, for peptides with multiple genomic hits>
10. **peptideRepeatCount** - Indicates how many times this same hit was observed

Fasta Format

Index ▷

http://genetics.bwh.harvard.edu/pph/FASTA.html

FastQ Format

Index ▷

http://maq.sourceforge.net/fastq.shtml