

GTF to BED Conversion - Comprehensive Guide

Overview

GTF (Gene Transfer Format) and BED (Browser Extensible Data) are both formats for representing genomic features, but they serve different purposes and have different structures. Many bioinformatics tools, particularly quality control tools like RSeQC, require BED format rather than GTF.

Applicable to: RNA-seq QC, ChIP-seq analysis, ATAC-seq, any workflow using genomic annotations

Format Comparison

GTF Format

Structure: Multiple lines per gene

```
chr1  ENSEMBL  exon  1000  1500  .  +  .  gene_id "GENE1"; transcript_id "TRANS1";
chr1  ENSEMBL  exon  2000  2800  .  +  .  gene_id "GENE1"; transcript_id "TRANS1";
chr1  ENSEMBL  exon  4400  5000  .  +  .  gene_id "GENE1"; transcript_id "TRANS1";
chr1  ENSEMBL  CDS   1200  1500  .  +  0  gene_id "GENE1"; transcript_id "TRANS1";
chr1  ENSEMBL  CDS   2000  2800  .  +  2  gene_id "GENE1"; transcript_id "TRANS1";
```

Characteristics:

- 9 columns (tab-separated)
 - One line per feature (exon, CDS, start_codon, etc.)
 - Rich metadata in attributes column
 - Verbose but informative
-

BED12 Format

Structure: One line per transcript

```
chr1  1000  5000  TRANS1  1000  +  1200  2800  0  3  500,800,600  0,1000,3400
```

Characteristics:

- 12 columns (tab-separated)
 - Single line represents entire transcript
 - Compact representation
 - Exons encoded as "blocks"
-

BED12 Column Specification

Columns 1-3: Genomic Position

Column	Name	Description	Example
1	chrom	Chromosome name	chr1
2	chromStart	Start position (0-based)	1000
3	chromEnd	End position (1-based, exclusive)	5000

Coordinate system:

- **0-based, half-open** [start, end)
 - chromStart: inclusive
 - chromEnd: exclusive
 - Example: [1000, 5000) means bases 1000-4999
-

Columns 4-6: Basic Attributes

Column	Name	Description	Example
4	name	Feature name	ENST000000001
5	score	Display score (0-1000)	1000
6	strand	+ or -	+

Score field:

- Often set to 1000 or 0
 - Used for visualization intensity
 - Not critical for most analyses
-

Columns 7-9: CDS Information

Column	Name	Description	Example
7	thickStart	CDS start position	1200
8	thickEnd	CDS end position	2800
9	itemRgb	RGB color	0

CDS boundaries:

- thickStart: Where coding sequence begins
- thickEnd: Where coding sequence ends

- Region before thickStart: 5' UTR
- Region after thickEnd: 3' UTR
- itemRgb: Usually 0 (not used)

Columns 10-12: Exon Structure (Blocks)

Column	Name	Description	Example
10	blockCount	Number of exons	3
11	blockSizes	Comma-separated exon lengths	500,800,600
12	blockStarts	Comma-separated relative starts	0,1000,3400

Block encoding:

Transcript from 1000-5000 with 3 exons:

- Exon 1: 1000-1500 (length 500, start at position 0 relative to chromStart)
- Exon 2: 2000-2800 (length 800, start at position 1000 relative to chromStart)
- Exon 3: 4400-5000 (length 600, start at position 3400 relative to chromStart)

Encoded as:

```
blockCount: 3
blockSizes: 500,800,600
blockStarts: 0,1000,3400
```

Calculating absolute exon positions:

```
Exon_absolute_start = chromStart + blockStart
Exon_absolute_end = chromStart + blockStart + blockSize
```

Example for Exon 2:

Start: 1000 + 1000 = 2000

End: 1000 + 1000 + 800 = 2800

Conversion Process

Two-Step Conversion

GTF cannot be directly converted to BED12. The standard approach uses an intermediate format:

GTF → genePred → BED12

Why two steps?:

- No single tool converts GTF \rightarrow BED12 directly
 - genePred is UCSC's internal gene model format
 - genePred \rightarrow BED12 conversion is well-defined
 - Maintains exon/intron structure correctly
-

Step 1: GTF \rightarrow genePred

Tool: gtfToGenePred (UCSC tools)

gtfToGenePred input.gtf output.genePred

Command breakdown:

```
gtfToGenePred \           # UCSC conversion tool
  input.gtf \             # Input GTF file
  output.genePred         # Output genePred file
```

What happens:

1. Reads GTF line by line
2. Groups features by transcript_id
3. Combines exons into single transcript entry
4. Outputs tabular genePred format

genePred format (10+ columns):

```
TRANS1  chr1  +  1000  5000  1200  2800  3  1000,2000,4400  1500,2800,5000
```

Columns: name, chrom, strand, txStart, txEnd, cdsStart, cdsEnd, exonCount, exonStarts, exonEnds

Step 2: genePred \rightarrow BED12

Tool: genePredToBed (UCSC tools)

genePredToBed input.genePred output.bed

Command breakdown:

```
genePredToBed \           # UCSC conversion tool
  input.genePred \        # Input genePred file
  output.bed              # Output BED12 file
```

What happens:

1. Reads genePred entries
2. Converts coordinates to BED12 format
3. Calculates block counts, sizes, and starts
4. Outputs BED12 format

Complete Conversion Example

```
# Input: annotation.gtf
# Output: annotation.bed

# Step 1: GTF to genePred
gtfToGenePred annotation.gtf temp.genePred

# Step 2: genePred to BED12
genePredToBed temp.genePred annotation.bed

# Clean up intermediate file
rm temp.genePred
```

What Gets Preserved vs Lost

Preserved in BED12

Genomic coordinates:

- Chromosome
- Start and end positions
- Strand

Exon structure:

- Number of exons
- Exon boundaries
- Exon order

CDS boundaries:

- Coding sequence start
- Coding sequence end
- UTR regions (implicit)

Feature names:

- Transcript IDs
 - Gene names (depending on input)
-

Lost in BED12

Detailed metadata:

- gene_biotype (protein_coding, lncRNA, etc.)

- gene_source (ENSEMBL, HAVANA, etc.)
- transcript_biotype
- Additional attributes

Feature types:

- No distinction between exon types
- No separate CDS, start_codon entries
- All merged into blocks

Multiple transcripts per gene:

- Each transcript is separate line
- Gene-level grouping not preserved

Impact: Sufficient for coordinate-based analysis, insufficient for detailed annotation queries

Use Cases

When BED12 is Required

1. RSeQC Tools:

- read_distribution.py
- geneBody_coverage.py
- junction_annotation.py
- inner_distance.py

2. Genome Browsers:

- UCSC Genome Browser
- IGV (supports both, BED faster)

3. Bedtools Operations:

- intersectBed
- coverageBed
- closestBed

4. Custom Scripts:

- Faster parsing (one line per transcript)
 - Simpler coordinate lookups
-

When GTF is Preferred

1. Gene-centric Analysis:

- Need gene_biotype filtering

- Multi-transcript awareness
- Detailed feature types

2. Quantification Tools:

- featureCounts (uses GTF)
- HTSeq (uses GTF/GFF)
- RSEM (uses GTF)

3. Annotation Queries:

- Need full metadata
- Gene classification
- Transcript variant analysis

Validation

Checking Conversion Success

Basic checks:

```
# 1. File exists and is not empty
ls -lh annotation.bed

# 2. Count transcripts
wc -l annotation.bed

# 3. Check format (should have 12 columns)
head -n 1 annotation.bed | awk '{print NF}'

# 4. View first few entries
head -n 5 annotation.bed

# 5. Check for errors
grep -c "^chr" annotation.bed # Count lines starting with chr
```

Common Issues

Issue 1: Chromosome Name Mismatch Problem: GTF has "1" but BED should have "chr1" (or vice versa)

Detection:

```
# Check GTF chromosomes
cut -f1 annotation.gtf | sort -u | head

# Check BED chromosomes
```

```
cut -f1 annotation.bed | sort -u | head

# Check BAM chromosomes (for comparison)
samtools view -H sample.bam | grep "^@SQ" | cut -f2
```

Solution:

```
# Add "chr" prefix if needed
sed 's/^/chr/' annotation.bed > annotation.chr.bed

# Remove "chr" prefix if needed
sed 's/^chr//' annotation.bed > annotation.nochr.bed
```

Issue 2: Invalid Coordinates Problem: chromEnd < chromStart or negative positions

Detection:

```
# Find invalid entries (end < start)
awk '$3 < $2' annotation.bed

# Find negative coordinates
awk '$2 < 0 || $3 < 0' annotation.bed
```

Solution:

- Fix source GTF
- Filter out problematic entries
- Check annotation source quality

Issue 3: Malformed Blocks Problem: blockCount doesn't match number of sizes/starts

Detection:

```
# Count commas in blockSizes and blockStarts
awk -F'\t' '{
    sizes = gsub(/,/ , ",", $11);
    starts = gsub(/,/ , ",", $12);
    if (sizes != starts) print $0
}' annotation.bed
```

Solution:

- Re-run conversion
- Check GTF quality
- Ensure complete exon entries in GTF

Issue 4: Undefined Strand Problem: Strand is "." instead of "+" or "-"

Detection:

```
# Find entries with undefined strand
awk '$6 == "." annotation.bed | wc -l
```

Solution:

- Fix source GTF (strand must be defined)
 - Or filter out these entries
 - Some tools may accept "." but most don't
-

Housekeeping Gene Subset

Purpose

For some QC analyses (gene body coverage), analyzing all genes is slow. A subset of "housekeeping" genes provides faster results with similar accuracy.

Creating Subset

Strategy: Select longest transcripts

```
# Add length column, sort by length, take top N
awk 'BEGIN {OFS="\t"} {print $0, $3-$2}' full.bed | \
  sort -k13,13rn | \
  head -n 5000 | \
  cut -f1-12 > housekeeping.bed
```

Command breakdown:

```
awk 'BEGIN {OFS="\t"} {print $0, $3-$2}' full.bed
# Calculate transcript length ($3-$2)
# Append as 13th column
# OFS="\t" ensures tab separation

sort -k13,13rn
# Sort by column 13 (length)
# -r: reverse (descending)
# -n: numeric sort

head -n 5000
# Take top 5000 longest
```

```
cut -f1-12
# Remove length column (back to BED12)
```

Why 5000?:

- Sufficient for robust statistics
- ~10-20x faster than full annotation
- Represents diverse gene expression levels
- Longer genes = better coverage sampling

Alternative strategies:

- Select by gene_biotype (if preserved in name field)
 - Random sampling
 - Specific gene lists (actual housekeeping genes)
-

Tool Requirements

UCSC Tools Installation

Method 1: Conda (Recommended):

```
conda install -c bioconda ucsc-gtftogenepred ucsc-genepredtobed
```

Method 2: Direct download:

```
# Download binaries
wget http://hgdownload.soe.ucsc.edu/admin/exe/linux.x86_64/gtfToGenePred
wget http://hgdownload.soe.ucsc.edu/admin/exe/linux.x86_64/genePredToBed

# Make executable
chmod +x gtfToGenePred genePredToBed

# Move to PATH
mv gtfToGenePred genePredToBed /usr/local/bin/
```

Verification:

```
gtfToGenePred 2>&1 | head -n 1
genePredToBed 2>&1 | head -n 1
```

Alternative Conversion Methods

Method 1: UCSC Table Browser (Online)

Pros:

- No installation needed
- User-friendly interface

- Handles complex annotations

Cons:

- Requires internet
- Manual process
- Not reproducible
- Limited to UCSC assemblies

Steps:

1. Go to <https://genome.ucsc.edu/cgi-bin/hgTables>
2. Select organism and assembly
3. Select group: "Genes and Gene Predictions"
4. Select table: Your annotation
5. Output format: BED
6. Click "get output"
7. Select "12 columns"

Method 2: AGAT (Another GTF/GFF Analysis Toolkit)

Installation

```
conda install -c bioconda agat
```

Conversion

```
agat_convert_sp_gxf2bed.pl -gff annotation.gtf -o annotation.bed
```

Pros:

- Single command
- Handles GFF3 and GTF
- More format validation

Cons:

- Slower than UCSC tools
 - Additional dependency
-

Method 3: Custom Script

Python example:

Simplified - see full implementation in AGAT or similar tools

```
from collections import defaultdict
```

```
def gtf_to_bed12(gtf_file, bed_file):
    transcripts = defaultdict(list)
```

```

# Parse GTF
with open(gtf_file) as f:
    for line in f:
        if line.startswith('#'):
            continue
        fields = line.strip().split('\t')
        if fields[2] == 'exon':
            # Extract exon info
            # Group by transcript_id
            # Build transcript structure
            pass

# Write BED12
with open(bed_file, 'w') as f:
    for tid, exons in transcripts.items():
        # Calculate blocks
        # Format BED12 line
        # Write output
        pass

```

Caution: Implementing correctly is complex. Use established tools.

Performance Considerations

Conversion Speed

Genome	Transcripts	Time	Memory
Human	~240,000	10-30s	<2GB
Mouse	~150,000	5-15s	<2GB
Drosophila	~35,000	1-5s	<1GB
Yeast	~6,000	<1s	<500MB

Factors:

- Annotation complexity
 - Disk I/O speed
 - Number of transcripts
-

File Sizes

Genome	GTF Size	BED12 Size	Compression
Human	1.5GB	400MB	3.75x smaller
Mouse	800MB	200MB	4x smaller
Drosophila	150MB	40MB	3.75x smaller

Why BED is smaller:

- One line per transcript vs many
- Less metadata
- More compact encoding

Best Practices

For Reproducibility

Document versions:

```
# Save conversion command and versions
echo "gtfToGenePred version: $(gtfToGenePred 2>&1 | head -n1)" > conversion.log
echo "Source GTF: annotation.gtf" >> conversion.log
echo "Date: $(date)" >> conversion.log
```

Validate output:

```
# Check line count
wc -l annotation.bed

# Verify format
head annotation.bed

# Check for errors
grep -v "^chr" annotation.bed | head
```

Keep intermediate files (for debugging):

```
# Don't delete genePred immediately
gtfToGenePred annotation.gtf annotation.genePred
genePredToBed annotation.genePred annotation.bed
# Keep annotation.genePred for troubleshooting
```

For Performance

Create housekeeping subset for QC:

```
# Faster gene body coverage
head -n 5000 annotation.bed > housekeeping.bed
```

Index for random access (bedtools):

```
sort -k1,1 -k2,2n annotation.bed > annotation.sorted.bed
bgzip annotation.sorted.bed
tabix -p bed annotation.sorted.bed.gz
```

Troubleshooting

Problem: Empty Output File

Possible causes:

1. Invalid GTF format
2. No transcript_id in GTF
3. Incomplete exon entries

Diagnosis:

```
# Check GTF format
head annotation.gtf

# Look for transcript_id
grep "transcript_id" annotation.gtf | head

# Count exon entries
grep "exon" annotation.gtf | wc -l
```

Problem: Conversion Errors

Common error messages:

Error: Expecting 9 columns

Solution: GTF is malformed, check tab separation

Warning: no exons for transcript X

Solution: GTF incomplete, missing exon entries

Error: chromosome names must match

Solution: Inconsistent chromosome naming in GTF

Related Documentation

- RSeQC Tools: docs/rseqc.md
- GTF Format Specification: <https://mblab.wustl.edu/GTF22.html>

- **BED Format Specification:** <https://genome.ucsc.edu/FAQ/FAQformat.html#format1>
 - **UCSC Tools:** <http://hgdownload.soe.ucsc.edu/admin/exe/>
-

Document Version: 2.0

Last Updated: January 2026

Applicable to: Any workflow requiring BED format annotations