

# Cufflinks

## Transcript assembly, differential expression, and differential regulation for RNA-Seq



**Please Note** If you have questions about how to use Cufflinks or would like more information about the software, please email [tophat.cufflinks@gmail.com](mailto:tophat.cufflinks@gmail.com), though we ask you to have a look at the [paper](#) and the [supplemental methods](#) first, as your question be answered there.

## Site Map

- [Home](#)
- [Getting started](#)
- [Manual](#)
- [How Cufflinks works](#)
- [FAQ](#)

## News and updates

New releases and related tools will be announced through the [mailing list](#)

## Getting Help

Questions about Cufflinks should be sent to [tophat.cufflinks@gmail.com](mailto:tophat.cufflinks@gmail.com). Please do not email technical questions to Cufflinks contributors directly.

## Releases

version 1.0.3

6/1/2011

[Source code](#)

[Linux x86\\_64 binary](#)

[Mac OS X x86\\_64 binary](#)

## Related Tools

- [TopHat](#): Alignment of short RNA-Seq reads
- [Bowtie](#): Ultrafast short read alignment

## Publications

- Trapnell C, Williams BA, Pertea G, Mortazavi AM, Kwan G, van Baren MJ, Salzberg SL, Wold B, Pachter L. [Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation](#) *Nature Biotechnology* doi:10.1038/nbt.1621
- Roberts A, Trapnell C, Donaghey J, Rinn JL, Pachter L. [Improving RNA-Seq expression estimates by correcting for fragment bias](#) *Genome Biology* doi:10.1186/gb-2011-12-3-r22

## Contributors

- [Cole Trapnell](#)
- [Adam Roberts](#)
- Geo Pertea
- Brian Williams

- [Ali Mortazavi](#)
- Gordon Kwan
- Jeltje van Baren
- [Steven Salzberg](#)
- [Barbara Wold](#)
- [Lior Pachter](#)

## Links

- [Berkeley LMCB](#)
- [UMD CBCB](#)
- [Wold Lab](#)

# Manual

- [Prerequisites](#)
- [Running Cufflinks](#)
  - [Input Files](#)
  - [Output Files](#)
    - [Transfrags in GTF](#)
    - [Transcript-level expression](#)
    - [Gene-level expression](#)
- [Running Cuffcompare](#)
  - [Input Files](#)
  - [Output Files](#)
    - [Transfrags for each reference transcript](#)
    - [Classification for transfrag](#)
    - [Tracking transfrags through multiple samples](#)
  - [Transfrag class codes](#)
- [Merging assemblies with cuffmerge](#)
  - [Input Files](#)
  - [Output Files](#)
- [Running Cuffdiff](#)
  - [Input Files](#)
  - [Output Files](#)
    - [FPKM tracking](#)
    - [Differential expression](#)
    - [Differential splicing](#)
    - [Differential coding sequence output](#)
- [FPKM Tracking Format](#)
- [Library Types](#)

## Prerequisites

Cufflinks runs on intel-based computers running Linux or Mac OS X and that have GCC 4.0 or greater installed. You can install pre-compiled binaries or build Cufflinks from the source code. If you wish to build Cufflinks yourself, you will need to install the [Boost C++ libraries](#). See [Installing Boost](#), on the getting started page. You will also need to build and install the [SAM tools](#), but you should take a look at the getting started page for detailed instructions, because the headers and `libbam` must be accessible to the Cufflinks build scripts.

## Running Cufflinks

Run `cufflinks` from the command line as follows:

```
Usage: cufflinks [options]* <aligned_reads.(sam/bam)>
```

The following is a detailed description of the options used to control Cufflinks:

**Arguments:**

`<aligned_reads.(sam/bam)>`

A file of RNA-Seq read alignments in the [SAM format](#). SAM is a standard short read alignment, that allows aligners to attach custom tags to individual alignments, and Cufflinks requires that the alignments you supply have some of these tags. Please see [Input formats](#) for more details.

**General Options:**

`-h/--help`

Prints the help message and exits

`-o/--output-dir <string>`

Sets the name of the directory in which Cufflinks will write all of its output. The default is `"/"`.

`-p/--num-threads <int>`

Use this many threads to align reads. The default is 1.

`-G/--GTF <reference_annotation.(gtf/gff)>`

Tells Cufflinks to use the supplied reference annotation ([a GFF file](#)) to estimate isoform expression. It will not assemble novel transcripts, and the program will ignore alignments not structurally compatible with any reference transcript.

`-g/--GTF-guide <reference_annotation.(gtf/gff)>`

Tells Cufflinks to use the supplied reference annotation ([GFF](#)) to guide [RABT assembly](#). Reference transcripts will be tiled with faux-reads to provide additional information in assembly. Output will include all reference transcripts as well as any novel genes and isoforms that are assembled.

`-M/--mask-file <mask.(gtf/gff)>`

Tells Cufflinks to ignore all reads that could have come from transcripts in this GTF file. We recommend including any annotated rRNA, mitochondrial transcripts other abundant transcripts you wish to ignore in your analysis in this file. Due to variable efficiency of mRNA enrichment methods and rRNA depletion kits, masking these transcripts often improves the overall robustness of transcript abundance estimates.

`-b/--frag-bias-correct <genome.fa>`

Providing Cufflinks with a multifasta file via this option instructs it to run our new bias detection and correction

`-u/--multi-read-correct`

algorithm which can significantly improve accuracy of transcript abundance estimates. See [How Cufflinks Works](#) for more details.

`--library-type`

Tells Cufflinks to do an initial estimation procedure to more accurately weight reads mapping to multiple locations in the genome. See [How Cufflinks Works](#) for more details.

See [Library Types](#)

### Advanced Abundance Estimation Options:

`-m/--frag-len-mean <int>`

This is the expected (mean) fragment length. The default is 200bp.

**Note: Cufflinks now learns the fragment length mean for each SAM file, so using this option is no longer recommended with paired-end reads.**

`-s/--frag-len-std-dev <int>`

The standard deviation for the distribution on fragment lengths. The default is 80bp.

**Note: Cufflinks now learns the fragment length standard deviation for each SAM file, so using this option is no longer recommended with paired-end reads.**

`-N/--upper-quartile-norm`

With this option, Cufflinks normalizes by the upper quartile of the number of fragments mapping to individual loci instead of the total number of sequenced fragments. This can improve robustness of differential expression calls for less abundant genes and transcripts.

`--total-hits-norm`

With this option, Cufflinks counts all fragments, including those not compatible with any reference transcript, towards the number of mapped hits used in the FPKM denominator. This option can be combined with `-N/--upper-quartile-norm`. It is active by default.

`--compatible-hits-norm`

With this option, Cufflinks counts only those fragments compatible with some reference transcript towards the number of mapped hits used in the FPKM denominator. This option can be combined with `-N/--upper-`

`quartile-norm`. It is inactive by default, and can only be used in combination with `--GTF`. Use with either RABT or ab initio assembly is not supported

`--num-importance-samples <int>`

Sets the number of importance samples generated for each locus during abundance estimation. Default: 1000

`--max-mle-iterations <int>`

Sets the number of iterations allowed during maximum likelihood estimation of abundances. Default: 5000

### Advanced Assembly Options:

`-L/--label`

Cufflinks will report transfrags in GTF format, with a prefix given by this option. The default prefix is "CUFF".

`-F/--min-isoform-fraction <0.0-1.0>`

After calculating isoform abundance for a gene, Cufflinks filters out transcripts that it believes are very low abundance, because isoforms expressed at extremely low levels often cannot reliably be assembled, and may even be artifacts of incompletely spliced precursors of processed transcripts. This parameter is also used to filter out introns that have far fewer spliced alignments supporting them. The default is 0.1, or 10% of the most abundant isoform (the major isoform) of the gene.

`-j/--pre-mrna-fraction <0.0-1.0>`

Some RNA-Seq protocols produce a significant amount of reads that originate from incompletely spliced transcripts, and these reads can confound the assembly of fully spliced mRNAs. Cufflinks uses this parameter to filter out alignments that lie within the intronic intervals implied by the spliced alignments. The minimum depth of coverage in the intronic region covered by the alignment is divided by the number of spliced reads, and if the result is lower than this parameter value, the intronic alignments are ignored. The default is 15%.

`-I/--max-intron-length <int>`

The maximum intron length. Cufflinks will not report transcripts with introns longer than this, and will ignore SAM alignments with REF\_SKIP CIGAR operations longer than

`-a/--junc-alpha <0.0-1.0>`

this. The default is 300,000.

The alpha value for the binomial test used during false positive spliced alignment filtration. Default: 0.001

`-A/--small-anchor-fraction <0.0-1.0>`

Spliced reads with less than this percent of their length on each side of the junction are considered suspicious and are candidates for filtering prior to assembly. Default: 0.09.

`--min-frags-per-transfrag <int>`

Assembled transfrags supported by fewer than this many aligned RNA-Seq fragments are not reported. Default: 10.

`--overhang-tolerance <int>`

The number of bp allowed to enter the intron of a transcript when determining if a read or another transcript is mappable to/compatible with it. The default is 8 bp based on the default bowtie/TopHat parameters.

`--max-bundle-length <int>`

Maximum genomic length allowed for a given bundle. The default is 3,500,000 bp.

`--min-intron-length <int>`

Minimum intron size allowed in genome. The default is 50 bp.

`--trim-3-avgcov-thresh <int>`

Minimum average coverage required to attempt 3' trimming. The default is 10.

`--trim-3-dropoff-frac <int>`

The fraction of average coverage below which to trim the 3' end of an assembled transcript. The default is 0.1.

### Advanced Reference Annotation Based Transcript (RABT) Assembly Options:

*These options have an affect only when used in conjunction with `-g/--GTF-guide`.*

`--3-overhang-tolerance <int>`

The number of bp allowed to overhang the 3' end of a reference transcript when determining if an assembled transcript should be merged with it (ie, the assembled transcript is not novel). The default is 600 bp.

`--intron-overhang-tolerance <int>`

The number of bp allowed to enter the intron of a reference transcript when determining if an assembled transcript should be merged with it (ie, the assembled transcript is not novel). The default is 50 bp.

`--no-faux-reads`

This option disables tiling of the reference transcripts with faux reads. Use this if you only want to use sequencing reads in assembly but do not want to output assembled transcripts that lay within reference transcripts. All reference transcripts in the input annotation will also be included in the output.

#### Advanced Program Behavior Options:

`-v/--verbose`

Print lots of status updates and other diagnostic information.

`-q/--quiet`

Suppress messages other than serious warnings and errors.

`--no-update-check`

Turns off the automatic routine that contacts the Cufflinks server to check for a more recent version.

## Cufflinks Input

Cufflinks takes a text file of SAM alignments, or a binary SAM (BAM) file as input. For more details on the SAM format, see the [specification](#). The RNA-Seq read mapper [TopHat](#) produces output in this format, and is recommended for use with Cufflinks. However Cufflinks will accept SAM alignments generated by any read mapper. Here's an example of an alignment Cufflinks will accept:

```
s6.25mer.txt-913508      16      chr1 4482736 255 14M431N11M * 0 0 \
CAAGATGCTAGGCAAGTCTTGAAG IIIIIIIIIIIIIIIIIIIIIII NM:i:0 XS:A:-
```

Note the use of the custom tag `XS`. This attribute, which must have a value of "+" or "-", indicates which strand the RNA that produced this read came from. While this tag can be applied to any alignment, including unspliced ones, it **must** be present for all spliced alignment records (those with a 'N' operation in the CIGAR string).

The SAM file supplied to Cufflinks **must** be sorted by reference position. If you aligned your reads with TopHat, your alignments will be properly sorted already. If you used another tool, you may want to make sure they are properly sorted as follows:

```
sort -k 3,3 -k 4,4n hits.sam > hits.sam.sorted
```

**NOTE:** Cufflinks currently only supports SAM alignments with the CIGAR match ('M') and reference skip ('N') operations. Support for the other operations, such as insertions, deletions, and clipping, will be added in the future.

## Cufflinks Output

Cufflinks produces three output files:

### 1) transcripts.gtf

This [GTF](#) file contains Cufflinks' assembled isoforms. The first 7 columns are standard GTF, and the last column contains attributes, some of which are also standardized ("gene\_id", and "transcript\_id"). There one GTF record per row, and each record represents either a transcript or an exon within a transcript. The columns are defined as follows:

Column number	Column name	Example	Description
1	seqname	chrX	Chromosome or contig name
2	source	Cufflinks	The name of the program that generated this file (always 'Cufflinks')
3	feature	exon	The type of record (always either "transcript" or "exon").
4	start	77696957	The leftmost coordinate of this record (where 1 is the leftmost possible coordinate)
5	end	77712009	The rightmost coordinate of this record, inclusive.
6	score	77712009	The most abundant isoform for each gene is assigned a score of 1000. Minor isoforms are scored by the ratio (minor FPKM/major FPKM)
7	strand	+	Cufflinks' guess for which strand the isoform came from. Always one of "+", "-", "."
7	frame	.	Cufflinks does not predict where the start and stop codons (if any) are located within each transcript, so this field is not used.
8	attributes	...	See below.

Each GTF record is decorated with the following attributes:

Attribute	Example	Description
gene_id	CUFF.1	Cufflinks gene id
transcript_id	CUFF.1.1	Cufflinks transcript id
FPKM	101.267	Isoform-level relative abundance in <b>F</b> ragments <b>P</b> er <b>K</b> ilobase of exon model per <b>M</b> illion mapped fragments
frac	0.7647	Reserved. Please ignore, as this attribute may be deprecated in the future
conf_lo	0.07	Lower bound of the 95% confidence interval of the abundance of this isoform, as a fraction of the isoform abundance. That is, lower bound = FPKM * (1.0 - conf_lo)
conf_hi	0.1102	Upper bound of the 95% confidence interval of the abundance of this isoform, as a fraction of the isoform abundance. That is, upper bound = FPKM * (1.0 + conf_lo)
cov	100.765	Estimate for the absolute depth of read coverage across the whole transcript
full_read_support	yes	When RABT assembly is used, this attribute reports whether or not all introns and internal exons were fully covered by reads from the data.

## 2) isoforms.fpkm\_tracking

This file contains the estimated isoform-level expression values in the generic [FPKM Tracking Format](#). Note, however that as there is only one sample, the "q" format is not used.



### 3) genes.fpkm\_tracking

This file contains the estimated gene-level expression values in the generic [FPKM Tracking Format](#). Note, however that as there is only one sample, the "q" format is not used.

## Running Cuffcompare

Cufflinks includes a program that you can use to help analyze the transfrags you assemble. The program `cuffcompare` helps you:

- Compare your assembled transcripts to a reference annotation
- Track Cufflinks transcripts across multiple experiments (e.g. across a time course)

From the command line, run `cuffcompare` as follows:

```
cuffcompare [options]* <cuff1.gtf> [cuff2.gtf] ... [cuffN.gtf]
```

## Cuffcompare Input

Cuffcompare takes Cufflinks' GTF output as input, and optionally can take a "reference" annotation (such as from [Ensembl](#))

### Arguments:

`<cuff1.gtf>`      A GTF file produced by cufflinks.

### Options:

`-h`                Prints the help message and exits

`-o <outprefix>`   All output files created by Cuffcompare will have this prefix (e.g. `<outprefix>.loci`, `<outprefix>.tracking`, etc.). If this option is not provided the default output prefix being used is: "cuffcmp"

`-r`                An optional "reference" annotation [GFF file](#). Each sample is matched against this file, and sample isoforms are tagged as overlapping, matching, or novel where appropriate. See the [refmap](#) and [tmap](#) output file descriptions below.

`-R`                If `-r` was specified, this option causes `cuffcompare` to ignore reference transcripts that are not overlapped by any transcript in one of `cuff1.gtf`, ..., `cuffN.gtf`. Useful for ignoring annotated transcripts that are not present in your RNA-Seq samples and thus adjusting the "sensitivity" calculation in the accuracy report written in the `<outprefix>` file

`-s <seq_dir>`     Causes `cuffcompare` to look into for fasta files with the underlying genomic sequences (one file per contig) against which your reads were aligned for some optional classification functions. For example, Cufflinks transcripts consisting mostly of lower-case bases are classified as repeats. Note that `<seq_dir>` **must** contain one fasta file per reference chromosome, and each file must be named after the chromosome, and have a `.fa` or `.fasta` extension.

`-C`                Enables the "contained" transcripts to be also written in the `<outprefix>.combined.gtf` file, with the attribute "contained\_in" showing the first container transfrag found. By default, without this option, `cuffcompare` does not write in that file isoforms that were found to be fully contained/covered (with the same compatible intron structure) by other transfrags in the same locus.

`-v`                Cuffcompare is a little more verbose about what it's doing, printing messages to stderr, and it will also show warning messages about any inconsistencies or potential issues found while reading the given GFF file(s).

## Cuffcompare Output

Cuffcompare produces the following output files:

### 1) <outprefix>.stats

Cuffcompare reports various statistics related to the "accuracy" of the transcripts in each sample when compared to the reference annotation data. The typical gene finding measures of "sensitivity" and "specificity" (as defined in Burset, M., Guigó, R. : **Evaluation of gene structure prediction programs** (1996) *Genomics*, 34 (3), pp. 353-367. [doi: 10.1006/geno.1996.0298](https://doi.org/10.1006/geno.1996.0298)) are calculated at various levels (nucleotide, exon, intron, transcript, gene) for each input file and reported in this file. The **Sn** and **Sp** columns show specificity and sensitivity values at each level, while the *fSn* and *fSp* columns are "fuzzy" variants of these same accuracy calculations, allowing for a very small variation in exon boundaries to still be counted as a "match". (If the -o option was not given the default prefix "cuffcmp" is used and these stats will be printed into a file named *cuffcmp.stats* in the current directory)

### 2) <outprefix>.combined.gtf

Cuffcompare reports a GTF file containing the "union" of all transfrags in each sample. If a transfrag is present in both samples, it is thus reported once in the combined gtf.

### 3) <outprefix>.tracking

This file matches transcripts up between samples. Each row contains a transcript structure that is present in one or more input GTF files. Because the transcripts will generally have different IDs (unless you assembled your RNA-Seq reads against a reference transcriptome), *cuffcompare* examines the structure of each the transcripts, matching transcripts that agree on the coordinates and order of all of their introns, as well as strand. Matching transcripts are allowed to differ on the length of the first and last exons, since these lengths will naturally vary from sample to sample due to the random nature of sequencing.

If you ran *cuffcompare* with the -r option, the first and second columns contain the closest matching reference transcript to the one described by each row.

Here's an example of a line from the tracking file:

```
TCONS_00000045 XLOC_000023 Tcea|uc007afj.1      j      \
      q1:exp.115|exp.115.0|100|3.061355|0.350242|0.350207 \
      q2:60hr.292|60hr.292.0|100|4.094084|0.000000|0.000000
```

In this example, a transcript present in the two input files, called *exp.115.0* in the first and *60hr.292.0* in the second, doesn't match any reference transcript exactly, but shares exons with *uc007afj.1*, an isoform of the gene *Tcea*, as indicated by the [class code](#) *j*. The first three columns are as follows:

Column number	Column name	Example	Description
1	Cufflinks transfrag id	TCONS_00000045	A unique internal id for the transfrag
2	Cufflinks locus id	XLOC_000023	A unique internal id for the locus
3	Reference gene id	Tcea	The gene_name attribute of the reference GTF record for this transcript, or '-' if no reference transcript overlaps this Cufflinks transcript

4	Reference transcript id	uc007afj.1	The transcript_id attribute of the reference GTF record for this transcript, or '-' if no reference transcript overlaps this Cufflinks transcript
5	Class code	c	The type of match between the Cufflinks transcripts in column 6 and the reference transcript. See <a href="#">class codes</a>

Each of the columns after the fifth have the following format:

qJ:<gene\_id>|<transcript\_id>|<FMI>|<FPKM>|<conf\_lo>|<conf\_hi>|<cov>|<len>

A transcript need not be present in all samples to be reported in the tracking file. A sample not containing a transcript will have a "-" in its entry in the row for that transcript.

(The following output files are created for each of the <cuff\_in> file given, in the same directories where the <cuff\_in> files reside)

#### 4) <cuff\_in>.refmap

This tab delimited file lists, for each reference transcript, which cufflinks transcripts either fully or partially match it. There is one row per reference transcript, and the columns are as follows:

Column number	Column name	Example	Description
1	Reference gene name	Myog	The gene_name attribute of the reference GTF record for this transcript, if present. Otherwise gene_id is used.
2	Reference transcript id	uc007cr1.1	The transcript_id attribute of the reference GTF record for this transcript
3	Class code	c	The type of match between the Cufflinks transcripts in column 4 and the reference transcript. One of either 'c' for partial match, or '=' for full match.
4	Cufflinks matches	CUFF.23567.0,CUFF.24689.0	A comma separated list of Cufflinks transcript ids matching the reference transcript

#### 5) <cuff\_in>.tmap

This tab delimited file lists the most closely matching reference transcript for each Cufflinks transcript. There is one row per Cufflinks transcript, and the columns are as follows:

Column number	Column name	Example	Description
1	Reference gene name	Myog	The gene_name attribute of the reference GTF record for this transcript, if present. Otherwise gene_id is used.
2	Reference transcript id	uc007cr1.1	The transcript_id attribute of the reference GTF record for this transcript
3	Class code	c	The type of relationship between the Cufflinks transcripts in column 4 and the reference transcript (as described in the <b>Class Codes</b> section below)
4	Cufflinks gene id	CUFF.23567	The Cufflinks internal gene id

5	Cufflinks transcript id	CUFF.23567.0	The Cufflinks internal transcript id
6	Fraction of major isoform (FMI)	100	The expression of this transcript expressed as a fraction of the major isoform for the gene. Ranges from 1 to 100.
7	FPKM	1.4567	The expression of this transcript expressed in FPKM
8	FPKM_conf_lo	0.7778	The lower limit of the 95% FPKM confidence interval
9	FPKM_conf_hi	1.9776	The upper limit of the 95% FPKM confidence interval
10	Coverage	3.2687	The estimated average depth of read coverage across the transcript.
11	Length	1426	The length of the transcript
12	Major isoform ID	CUFF.23567.0	The Cufflinks ID of the gene's major isoform

### Class Codes

If you ran `cuffcompare` with the `-r` option, tracking rows will contain the following values. If you did not use `-r`, the rows will all contain "-" in their class code column.

Priority	Code	Description
1	=	Complete match of intron chain
2	c	Contained
3	j	Potentially novel isoform (fragment): at least one splice junction is shared with a reference transcript
4	e	Single exon transfrag overlapping a reference exon and at least 10 bp of a reference intron, indicating a possible pre-mRNA fragment.
5	i	A transfrag falling entirely within a reference intron
6	o	Generic exonic overlap with a reference transcript
7	p	Possible polymerase run-on fragment (within 2Kbases of a reference transcript)
8	r	Repeat. Currently determined by looking at the soft-masked reference sequence and applied to transcripts where at least 50% of the bases are lower case
9	u	Unknown, intergenic transcript
10	x	Exonic overlap with reference on the opposite strand
11	s	An intron of the transfrag overlaps a reference intron on the opposite strand (likely due to read mapping errors)
12	.	(.tracking file only, indicates multiple classifications)

## Merging assemblies with cuffmerge

Cufflinks includes a script called `cuffmerge` that you can use to merge together several Cufflinks assemblies. It handles also handles running `Cuffcompare` for you, and automatically filters a number of transfrags that are probably artifacts. If you have a

reference GTF file available, you can provide it to the script in order to gracefully merge novel isoforms and known isoforms and maximize overall assembly quality. The main purpose of this script is to make it easier to make an assembly GTF file suitable for use with Cuffdiff. From the command line, run `cuffmerge` as follows:

```
cuffmerge [options]* <assembly_GTF_list.txt>
```

## cuffmerge Input

`cuffmerge` takes several assembly GTF files from Cufflinks' as input. Input GTF files must be specified in a "manifest" file listing full paths to the files.

### Arguments:

```
<assembly_list.txt>
```

Text file "manifest" with a list (one per line) of GTF files that you'd like to merge together into a single GTF file.

### Options:

```
-h/--help
```

Prints the help message and exits

```
-o <outprefix>
```

Write the summary stats into the text output file `<outprefix>` (instead of stdout)

```
-g/--ref-gtf
```

An optional "reference" annotation GTF. The input assemblies are merged together with the reference GTF and included in the final output.

```
-p/--num-threads <int>
```

Use this many threads to align reads. The default is 1.

```
-s/--ref-sequence <seq_dir>/<seq_fasta>
```

This argument should point to the genomic DNA sequences for the reference. If a directory, it should contain one fasta file per contig. If a multifasta file, all contigs should be present. The merge script will pass this option to `cuffcompare`, which will use the sequences to assist in classifying transfrags and excluding artifacts (e.g. repeats). For example, Cufflinks transcripts consisting mostly of lower-case bases are classified as repeats. Note that `<seq_dir>` **must** contain one fasta file per reference chromosome, and each file must be named after the chromosome, and have a `.fa` or `.fasta` extension.

## cuffmerge Output

`cuffmerge` produces a GTF file that contains an assembly that merges together the input assemblies.

```
<outprefix>/merged.gtf
```

## Running Cuffdiff

Cufflinks includes a program, "Cuffdiff", that you can use to find significant changes in transcript expression, splicing, and promoter use. From the command line, run `cuffdiff` as follows:

```
cuffdiff [options]* <transcripts.gtf> <sample1_replicate1.sam[,...,sample1_replicateM]>
<sample2_replicate1.sam[,...,sample2_replicateM.sam]>...
```

```
[sampleN.sam_replicate1.sam[,...,sample2_replicateM.sam]]
```

## Cuffdiff Input

Cuffdiff takes a GTF2/GFF3 file of transcripts as input, along with two or more SAM files containing the fragment alignments for two or more samples. It produces a number of output files that contain test results for changes in expression at the level of transcripts, primary transcripts, and genes. It also tracks changes in the relative abundance of transcripts sharing a common transcription start site, and in the relative abundances of the primary transcripts of each gene. Tracking the former allows one to see changes in splicing, and the latter lets one see changes in relative promoter use within a gene. If you have more than one **replicate** for a sample, supply the SAM files for the sample as a single **comma-separated** list. It is not necessary to have the same number of replicates for each sample. Cuffdiff requires that transcripts in the input GTF be annotated with certain attributes in order to look for changes in primary transcript expression, splicing, coding output, and promoter use. These attributes are:

Attribute	Description
tss_id	The ID of this transcript's inferred start site. Determines which primary transcript this processed transcript is believed to come from. Cuffcompare appends this attribute to every transcript reported in the <i>.combined.gtf</i> file.
p_id	The ID of the coding sequence this transcript contains. This attribute is attached by Cuffcompare to the <i>.combined.gtf</i> records only when it is run with a reference annotation that include CDS records. Further, differential CDS analysis is <b>only</b> performed when all isoforms of a gene have p_id attributes, because neither Cufflinks nor Cuffcompare attempt to assign an open reading frame to transcripts.

**Note:** If an arbitrary GTF/GFF3 file is used as input (instead of the *.combined.gtf* file produced by Cuffcompare), these attributes will not be present, but Cuffcompare can still be used to obtain these attributes with a command like this:

```
cuffcompare -s /path/to/genome_seqs.fa -r annotation.gtf annotation.gtf
```

The resulting *cuffcmp.combined.gtf* file created by this command will have the tss\_id and p\_id attributes added to each record and this file can be used as input for cuffdiff.

### Arguments:

<transcripts.(gtf/gff)>	A transcript annotation file produced by cufflinks, cuffcompare, or other source.
<sample1.sam>	A SAM file of aligned RNA-Seq reads. If more than two are provided, Cuffdiff tests for differential expression and regulation between all pairs of samples.

### Options:

-h/--help	Prints the help message and exits
-o/--output-dir <string>	Sets the name of the directory in which Cuffdiff will write all of its output. The default is ".".
-L/--labels <label1,label2,...,labelN>	Specify a label for each sample, which will be included in various output files produced by Cuffdiff.
-p/--num-threads <int>	Use this many threads to align reads. The default is 1.
-T/--time-series	Instructs Cuffdiff to analyze the provided samples as a time series, rather than testing for differences between all pairs of samples. Samples should be provided in increasing time order at the command line (e.g first time point SAM, second timepoint SAM, etc.)

<code>-N/--upper-quartile-norm</code>	With this option, Cufflinks normalizes by the upper quartile of the number of fragments mapping to individual loci instead of the total number of sequenced fragments. This can improve robustness of differential expression calls for less abundant genes and transcripts.
<code>--total-hits-norm</code>	With this option, Cufflinks counts all fragments, including those not compatible with any reference transcript, towards the number of mapped fragments used in the FPKM denominator. This option can be combined with <code>-N/--upper-quartile-norm</code> . It is inactive by default.
<code>--compatible-hits-norm</code>	With this option, Cufflinks counts only those fragments compatible with some reference transcript towards the number of mapped fragments used in the FPKM denominator. This option can be combined with <code>-N/--upper-quartile-norm</code> . Using this mode is generally recommended in Cuffdiff to reduce certain types of bias caused by differential amounts of ribosomal reads which can create the impression of falsely differentially expressed genes. It is active by default.
<code>-b/--frag-bias-correct &lt;genome.fa&gt;</code>	Providing Cufflinks with the multifasta file your reads were mapped to via this option instructs it to run our bias detection and correction algorithm which can significantly improve accuracy of transcript abundance estimates. See <a href="#">How Cufflinks Works</a> for more details.
<code>-u/--multi-read-correct</code>	Tells Cufflinks to do an initial estimation procedure to more accurately weight reads mapping to multiple locations in the genome. See <a href="#">How Cufflinks Works</a> for more details.
<code>-c/--min-alignment-count &lt;int&gt;</code>	The minimum number of alignments in a locus for needed to conduct significance testing on changes in that locus observed between samples. If no testing is performed, changes in the locus are deemed not significant, and the locus' observed changes don't contribute to correction for multiple testing. The default is 10 fragment alignments.
<code>-M/--mask-file &lt;mask.(gtf/gff)&gt;</code>	Tells Cuffdiff to ignore all reads that could have come from transcripts in this GTF file. We recommend including any annotated rRNA, mitochondrial transcripts other abundant transcripts you wish to ignore in your analysis in this file. Due to variable efficiency of mRNA enrichment methods and rRNA depletion kits, masking these transcripts often improves the overall robustness of transcript abundance estimates.
<code>--FDR &lt;float&gt;</code>	The allowed false discovery rate. The default is 0.05.

#### Advanced Options:

<code>--library-type</code>	See <a href="#">Library Types</a>
<code>-m/--frag-len-mean &lt;int&gt;</code>	This is the expected (mean) fragment length. The default is 200bp. <b>Note: Cufflinks now learns the fragment length mean for each SAM file, so using this option is no longer recommended with paired-end reads.</b>
<code>-s/--frag-len-std-dev &lt;int&gt;</code>	The standard deviation for the distribution on fragment lengths. The default is 80bp. <b>Note: Cufflinks now learns the fragment length standard deviation for each SAM file, so using this option is no longer recommended with paired-end reads.</b>

<code>--num-importance-samples &lt;int&gt;</code>	Sets the number of importance samples generated for each locus during abundance estimation. Default: 1000
<code>--max-mle-iterations &lt;int&gt;</code>	Sets the number of iterations allowed during maximum likelihood estimation of abundances. Default: 5000
<code>-v/--verbose</code>	Print lots of status updates and other diagnostic information.
<code>-q/--quiet</code>	Suppress messages other than serious warnings and errors.
<code>--no-update-check</code>	Turns off the automatic routine that contacts the Cufflinks server to check for a more recent version.
<code>--poisson-dispersion</code>	Use the Poisson fragment dispersion model instead of learning one in each condition.
<code>--emit-count-tables</code>	Cuffdiff will output a file for each condition (called <code>&lt;sample&gt;_counts.txt</code> ) containing the fragment counts, fragment count variances, and fitted variance model.
<code>-F/--min-isoform-fraction &lt;0.0-1.0&gt;</code>	Cuffdiff will round down to zero the abundance of alternative isoforms quantified at below the specified fraction of the major isoforms. This is done after MLE estimation but before MAP estimation to improve robustness of confidence interval generation and differential expression analysis. The default is 1e-5, and we recommend you not alter this parameter.

## Cuffdiff Output

### 1) FPKM tracking files

Cuffdiff calculates the FPKM of each transcript, primary transcript, and gene in each sample. Primary transcript and gene FPKMs are computed by summing the FPKMs of transcripts in each primary transcript group or gene group. The results are output in FPKM tracking files in the format described [here](#). There are **four** FPKM tracking files:

<code>isoforms.fpk_tracking</code>	Transcript FPKMs
<code>genes.fpk_tracking</code>	Gene FPKMs. Tracks the summed FPKM of transcripts sharing each <code>gene_id</code>
<code>cds.fpk_tracking</code>	Coding sequence FPKMs. Tracks the summed FPKM of transcripts sharing each <code>p_id</code> , independent of <code>tss_id</code>
<code>tss_groups.fpk_tracking</code>	Primary transcript FPKMs. Tracks the summed FPKM of transcripts sharing each <code>tss_id</code>

### 2) Differential expression tests

This tab delimited file lists the results of differential expression testing between samples for spliced transcripts, primary transcripts, genes, and coding sequences. For each pair of samples *x* and *y*, four files are created

<code>isoform_exp.diff</code>	Transcript differential FPKM.
<code>gene_exp.diff</code>	Gene differential FPKM. Tests difference in the summed FPKM of transcripts sharing each <code>gene_id</code>



`tss_group_exp.diff` Primary transcript differential FPKM. Tests differences in the summed FPKM of transcripts sharing each `tss_id`

`cds_exp.diff` Coding sequence differential FPKM. Tests differences in the summed FPKM of transcripts sharing each `p_id` independent of `tss_id`

Each of the above files has the following format:

Column number	Column name	Example	Description
1	Tested id	XLOC_000001	A unique identifier describing the transcript, gene, primary transcript, or CDS being tested
2	gene	Lyplal	The <code>gene_name(s)</code> or <code>gene_id(s)</code> being tested
3	locus	chr1:4797771-4835363	Genomic coordinates for easy browsing to the genes or transcripts being tested.
4	sample 1	Liver	Label (or number if no labels provided) of the first sample being tested
5	sample 2	Brain	Label (or number if no labels provided) of the second sample being tested
6	Test status	NOTEST	Can be one of OK (test successful), NOTEST (not enough alignments for testing), LOWDATA (too complex or shallowly sequenced), or FAIL, when an ill-conditioned covariance matrix or other numerical exception prevents testing.
7	FPKM <sub>x</sub>	8.01089	FPKM of the gene in sample <i>x</i>
8	FPKM <sub>y</sub>	8.551545	FPKM of the gene in sample <i>y</i>
9	$\ln(\text{FPKM}_y/\text{FPKM}_x)$	0.06531	The natural log of the fold change <i>y/x</i>
10	test stat	0.860902	The value of the test statistic used to compute significance of the observed change in FPKM
11	p value	0.389292	The <b>uncorrected</b> <i>p</i> -value of the test statistic
12	q value	0.985216	The <b>FDR-adjusted</b> <i>p</i> -value of the test statistic
13	significant	no	Can be either "yes" or "no", depending on whether <i>p</i> is greater than the FDR <b>after</b> Benjamini-Hochberg correction for multiple-testing

### 3) Differential splicing tests - `splicing.diff`

This tab delimited file lists, for each primary transcript, the amount of overloading detected among its isoforms, i.e. how much differential splicing exists between isoforms processed from a single primary transcript. Only primary transcripts from which two or more isoforms are spliced are listed in this file.

Column number	Column name	Example	Description
---------------	-------------	---------	-------------

1	Tested id	TSS10015	A unique identifier describing the primary transcript being tested.
2	gene name	Rtkn	The <code>gene_name</code> or <code>gene_id</code> that the primary transcript being tested belongs to
3	locus	chr6:83087311-83102572	Genomic coordinates for easy browsing to the genes or transcripts being tested.
4	sample 1	Liver	Label (or number if no labels provided) of the first sample being tested
5	sample 2	Brain	Label (or number if no labels provided) of the second sample being tested
6	Test status	OK	Can be one of OK (test successful), NOTEST (not enough alignments for testing), LOWDATA (too complex or shallowly sequenced), or FAIL, when an ill-conditioned covariance matrix or other numerical exception prevents testing.
7	Reserved	0	
8	Reserved	0	
9	$\sqrt{JS(x,y)}$	0.22115	The splice overloading of the primary transcript, as measured by the square root of the Jensen-Shannon divergence computed on the relative abundances of the splice variants
10	test stat	0.22115	The value of the test statistic used to compute significance of the observed overloading, equal to $\sqrt{JS(x,y)}$
11	p value	0.000174982	The <b>uncorrected</b> <i>p</i> -value of the test statistic.
12	significant	yes	Can be either "yes" or "no", depending on whether <i>p</i> is greater then the FDR <b>after</b> Benjamini-Hochberg correction for multiple-testing

#### 4) Differential coding output - cds.diff

This tab delimited file lists, for each gene, the amount of overloading detected among its coding sequences, i.e. how much differential CDS output exists between samples. Only genes producing two or more distinct CDS (i.e. multi-protein genes) are listed here.

Column number	Column name	Example	Description
1	Tested id	XLOC_000002-[chr1:5073200-5152501]	A unique identifier describing the gene being tested.
2	gene name	Atp6v1h	The <code>gene_name</code> or <code>gene_id</code>
3	locus	chr1:5073200-5152501	Genomic coordinates for easy browsing to the genes or transcripts being tested.

4	sample 1	Liver	Label (or number if no labels provided) of the first sample being tested
5	sample 2	Brain	Label (or number if no labels provided) of the second sample being tested
6	Test status	OK	Can be one of OK (test successful), NOTEST (not enough alignments for testing), LOWDATA (too complex or shallowly sequenced), or FAIL, when an ill-conditioned covariance matrix or other numerical exception prevents testing.
7	Reserved	0	
8	Reserved	0	
9	$\sqrt{JS(x,y)}$	0.0686517	The CDS overloading of the gene, as measured by the square root of the Jensen-Shannon divergence computed on the relative abundances of the coding sequences
10	test stat	0.0686517	The value of the test statistic used to compute significance of the observed overloading, equal to $\sqrt{JS(x,y)}$
11	p value	0.00546783	The <b>uncorrected</b> $p$ -value of the test statistic
12	significant	yes	Can be either "yes" or "no", depending on whether $p$ is greater than the FDR <b>after</b> Benjamini-Hochberg correction for multiple-testing

## 5) Differential promoter use - promoters.diff

This tab delimited file lists, for each gene, the amount of overloading detected among its primary transcripts, i.e. how much differential promoter use exists between samples. Only genes producing two or more distinct primary transcripts (i.e. multi-promoter genes) are listed here.

Column number	Column name	Example	Description
1	Tested id	XLOC_000019	A unique identifier describing the gene being tested.
2	gene name	Tmem70	The <code>gene_name</code> or <code>gene_id</code>
3	locus	chr1:16651657-16668357	Genomic coordinates for easy browsing to the genes or transcripts being tested.

4	sample 1	Liver	Label (or number if no labels provided) of the first sample being tested
5	sample 2	Brain	Label (or number if no labels provided) of the second sample being tested
6	Test status	OK	Can be one of OK (test successful), NOTEST (not enough alignments for testing), LOWDATA (too complex or shallowly sequenced), or FAIL, when an ill-conditioned covariance matrix or other numerical exception prevents testing.
7	Reserved	0	
8	Reserved	0	
9	$\sqrt{JS(x,y)}$	0.0124768	The promoter overloading of the gene, as measured by the square root of the Jensen-Shannon divergence computed on the relative abundances of the primary transcripts
10	test stat	0.0124768	The value of the test statistic used to compute significance of the observed overloading, equal to $\sqrt{JS(x,y)}$
11	p value	0.394327	The <b>uncorrected</b> <i>p</i> -value of the test statistic
12	significant	no	Can be either "yes" or "no", depending on whether <i>p</i> is greater than the FDR <b>after</b> Benjamini-Hochberg correction for multiple-testing

## FPKM Tracking Files

FPKM tracking files use a generic format to output estimated expression values. Each FPKM tracking file has the following format:

Column number	Column name	Example	Description
1	tracking_id	TCONS_00000001	A unique identifier describing the object (gene, transcript, CDS, primary transcript)
2	class_code	=	The <code>class_code</code> attribute for the object, or "-" if not a transcript, or if <code>class_code</code> isn't present
3	nearest_ref_id	NM_008866.1	The reference transcript to which the class code refers, if any
4	gene_id	NM_008866	The <code>gene_id</code> (s) associated with the object
5	gene_short_name	Lypla1	The <code>gene_short_name</code> (s) associated with the object
6	tss_id	TSS1	The <code>tss_id</code> associated with the object, or "-" if not a transcript/primary transcript, or if <code>tss_id</code> isn't present
7	locus	chr1:4797771-4835363	Genomic coordinates for easy browsing to the object
8	length	2447	The number of base pairs in the transcript, or '-' if not a transcript/primary transcript

9	coverage	43.4279	Estimate for the absolute depth of read coverage across the object
10	status	OK	Can be one of OK (deconvolution successful), LOWDATA (too complex or shallowly sequenced), or FAIL, when an ill-conditioned covariance matrix or other numerical exception prevents deconvolution.
11	q0_FPKM	8.01089	FPKM of the object in sample 0
12	q0_FPKM_lo	7.03583	the lower bound of the 95% confidence interval on the FPKM of the object in sample 0
13	q0_FPKM_hi	8.98595	the upper bound of the 95% confidence interval on the FPKM of the object in sample 0
14	q1_FPKM	8.55155	FPKM of the object in sample 1
15	q1_FPKM_lo	7.77692	the lower bound of the 95% confidence interval on the FPKM of the object in sample 1
16	q1_FPKM_hi	9.32617	the upper bound of the 95% confidence interval on the FPKM of the object in sample 1
$3N + 11$	q $N$ _FPKM	7.34115	FPKM of the object in sample $N$
$3N + 12$	q $N$ _FPKM_lo	6.33394	the lower bound of the 95% confidence interval on the FPKM of the object in sample $N$
$3N + 13$	q $N$ _FPKM_hi	8.34836	the upper bound of the 95% confidence interval on the FPKM of the object in sample $N$

## Library Types

In cases where Cufflinks cannot determine the platform and protocol used to generate input reads, you can supply this information manually, which will allow Cufflinks to infer source strand information with certain protocols. The available options are listed below. For paired-end data, we currently only support protocols where reads are point towards each other.

Library Type	Examples	Description
fr-unstranded (default)	Standard Illumina	Reads from the left-most end of the fragment (in transcript coordinates) map to the transcript strand, and the right-most end maps to the opposite strand.
fr-firststrand	dUTP, NSR, NNSR	Same as above except we enforce the rule that the right-most end of the fragment (in transcript coordinates) is the first sequenced (or only sequenced for single-end reads). Equivalently, it is assumed that only the strand generated during first strand synthesis is sequenced.

fr-secondstrand	Directional Illumina (Ligation), Standard SOLiD	Same as above except we enforce the rule that the left-most end of the fragment (in transcript coordinates) is the first sequenced (or only sequenced for single-end reads). Equivalently, it is assumed that only the strand generated during second strand synthesis is sequenced.
-----------------	---	--

Please contact [Adam Roberts](#) to request support for a new protocol.

This research was supported in part by NIH grants R01-LM06845 and R01-GM083873, NSF grant CCF-0347992 and the Miller Institute for Basic Research in Science at UC Berkeley.

Administrator: [Cole Trapnell](#).  
Design by [David Herreman](#)