

Counting reads in features with `htseq-count`

Given a file with aligned sequencing reads and a list of genomic features, a common task is to count how many reads map to each feature.

A feature is here an interval (i.e., a range of positions) on a chromosome or a union of such intervals.

In the case of RNA-Seq, the features are typically genes, where each gene is considered here as the union of all its exons. One may also consider each exon as a feature, e.g., in order to check for alternative splicing. For comparative ChIP-Seq, the features might be binding region from a pre-determined list.

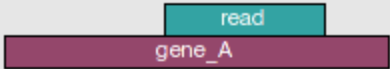
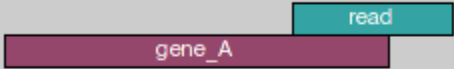


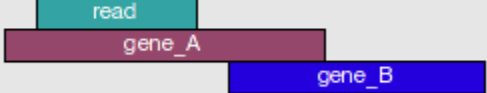
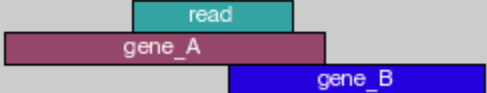
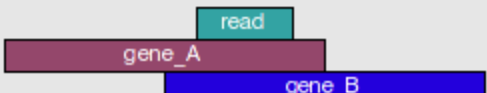
Special care must be taken to decide how to deal with reads that overlap more than one feature. The `htseq-count` script allows to choose between three modes. Of course, if none of these fits your needs, you can write your own script with HTSeq. See the chapter [A tour through HTSeq](#) for a step-by-step guide on how to do so. See also the FAQ at the end, if the following explanation seems to technical.

The three overlap resolution modes of `htseq-count` work as follows. For each position i in the read, a set $S(i)$ is defined as the set of all features overlapping position i . Then, consider the set S , which is (with i running through all position within the read or a read pair)

- the union of all the sets $S(i)$ for mode `union`. This mode is recommended for most use cases.
- the intersection of all the sets $S(i)$ for mode `intersection-strict`.
- the intersection of all non-empty sets $S(i)$ for mode `intersection-nonempty`.

If S contains precisely one feature, the read (or read pair) is counted for this feature. If it contains more than one feature, the read (or read pair) is counted as `ambiguous` (and not counted for any features), and if S is empty, the read (or read pair) is counted as `no_feature`.

The following figure illustrates the effect of these three modes:

	union	intersection_strict	intersection_nonempty
	gene_A	gene_A	gene_A
	gene_A	no_feature	gene_A
	gene_A	no_feature	gene_A
	gene_A	gene_A	gene_A
	gene_A	gene_A	gene_A
	ambiguous	gene_A	gene_A
	ambiguous	ambiguous	ambiguous

Usage

After you have installed HTSeq (see [Prerequisites and installation](#)), you can run `htseq-count` from the command line:

```
htseq-count [options] <alignment_file> <gff_file>
```

If the file `htseq-qa` is not in your path, you can, alternatively, call the script with

```
python -m HTSeq.scripts.count [options] <alignment_file> <gff_file>
```

The `<alignment_file>` contains the aligned reads in the SAM format. (Note that the [SAMtools](#) contain Perl scripts to convert most alignment formats to SAM.) Make sure to use a splicing-aware aligner such as TopHat. HTSeq-count makes full use of the information in the CIGAR field.

To read from standard input, use `-` as `<alignment_file>`.

If you have paired-end data, pay attention to the `-r` option described below.

The `<gff_file>` contains the features in the [GFF format](#).

The script outputs a table with counts for each feature, followed by the special counters, which count reads that were not counted for any feature for various reasons. The names of the special counters all start with a double **underscore**, to facilitate filtering. (Note: The double underscore was absent up to version 0.5.4). The special counters are:

- `__no_feature`: reads (or read pairs) which could not be assigned to any feature (set `S` as described above was empty).
- `__ambiguous`: reads (or read pairs) which could have been assigned to more than one feature and hence were not counted for any of these (set `S` had more than one element).
- `__too_low_aQual`: reads (or read pairs) which were skipped due to the `-a` option, see below
- `__not_aligned`: reads (or read pairs) in the SAM file without alignment
- `__alignment_not_unique`: reads (or read pairs) with more than one reported alignment. These reads are recognized from the `NH` optional SAM field tag. (If the aligner does not set this field, multiply aligned reads will be counted multiple times, unless they getv filtered out by due to the `-a` option.)

Important: The default for strandedness is `yes`. If your RNA-Seq data has not been made with a strand-specific protocol, this causes half of the reads to be lost. Hence, make sure to set the option `--stranded=no` unless you have strand-specific data!

Options

-f <format>, **--format**=<format>

Format of the input data. Possible values are `sam` (for text SAM files) and `bam` (for binary BAM files). Default is `sam`.

-r <order>, **--order**=<order>

For paired-end data, the alignment have to be sorted either by read name or by alignment position. If your data is not sorted, use the `samtools sort` function of `samtools` to sort it. Use this option, with `name` or `pos` for <order> to indicate how the input data has been sorted. The default is `name`.

If `name` is indicated, `htseq-count` expects all the alignments for the reads of a given read pair to appear in adjacent records in the input data. For `pos`, this is not expected; rather, read alignments whose mate alignment have not yet been seen are kept in a buffer in memory until the mate is found. While, strictly speaking, the latter will also work with unsorted data, sorting ensures that most alignment mates appear close to each other in the data and hence the buffer is much less likely to overflow.

-s <yes/no/reverse>, **--stranded**=<yes/no/reverse>

whether the data is from a strand-specific assay (default: `yes`)

For `stranded=no`, a read is considered overlapping with a feature regardless of whether it is mapped to the same or the opposite strand as the feature. For `stranded=yes` and single-end reads, the read has to be mapped to the same strand as the feature. For paired-end reads, the first read has to be on the same strand and the second read on the opposite strand. For `stranded=reverse`, these rules are reversed.

-a <minqual>, **--a**=<minqual>

skip all reads with alignment quality lower than the given minimum value (default: 10 — Note: the default used to be 0 until version 0.5.4.)

-t <feature type>, **--type**=<feature type>

feature type (3rd column in GFF file) to be used, all features of other type are ignored (default, suitable for RNA-Seq analysis using an [Ensembl GTF](#) file: `exon`)

-i <id attribute>, **--idattr**=<id attribute>

GFF attribute to be used as feature ID. Several GFF lines with the same feature ID will be considered as parts of the same feature. The feature ID is used to identity the counts in the output table. The default, suitable for RNA-Seq analysis using an Ensembl GTF file, is `gene_id`.

-m <mode>, **--mode**=<mode>

Mode to handle reads overlapping more than one feature. Possible values for <mode> are union, intersection-strict and intersection-nonempty (default: union)

-o <samout>, **--samout**=<samout>

write out all SAM alignment records into an output SAM file called <samout>, annotating each line with its assignment to a feature or a special counter (as an optional field with tag 'XF')

-q, **--quiet**

suppress progress report and warnings

-h, **--help**

Show a usage summary and exit

Frequently asked questions

My shell reports “command not found” when I try to run “htseq-count”. How can I launch the script?

The file “htseq-count” has to be in the system’s [search path](#). By default, Python places it in its script directory, which you have to add to your search path. A maybe easier alternative is to write `python -m HTSeq.scripts.count` instead of `htseq-count`, followed by the options and arguments, which will launch the htseq-count script as well.

Why are multi-mapping reads and reads overlapping multiple features discarded rather than counted for each feature?

The primary intended use case for `htseq-count` is *differential* expression analysis, where one compares the expression of the same gene across samples and not the expression of different genes within a sample. Now, consider two genes, which share a stretch of common sequence such that for a read mapping to this stretch, the aligner cannot decide which of the two genes the read originated from and hence reports a multiple alignment. If we discard all such reads, we undercount the total output of the genes, but the *ratio* of expression strength (the “fold change”) between samples or experimental condition will still be correct, because we discard the same fraction of reads in all samples. On the other hand, if we counted these reads for both genes, a subsequent differential-expression analysis might find false positives: Even if only one of the gene changes increases its expression in reaction to treatment, the additional read caused by this would be counted for both genes, giving the wrong appearance that both genes reacted to the treatment.

I have used a GTF file generated by the Table Browser function of the UCSC Genome Browser, and most reads are counted as ambiguous. Why?

In these files, the `gene_id` attribute incorrectly contains the same value as the `transcript_id` attribute and hence a different value for each transcript of the same gene. Hence, if a read maps to an exon shared by several transcripts of the same gene, this will appear to `htseq-count` as and overlap with several genes. Therefore, these GTF files cannot be used as is. Either correct the incorrect `gene_id` attributes with a suitable script, or use a GTF file from a different source.

Can I use htseq-count to count reads mapping to transcripts rather than genes?

In principle, you could instruct htseq-count to count for each of a gene’s transcript individually, by specifying `--idattr transcript_id`. However, all reads mapping to exons shared by several transcripts will then be considered ambiguous. (See second question.) Counting them for each transcript that contains the exons would be possible but makes little sense for typical use cases. (See first question.) If you want to perform differential expression analysis on the level of individual transcripts, maybe have a look at [our paper on DEXSeq](#) for a discussion on why we prefer performing such analyses on the level of exons instead.

For paired-end data, does htseq-count count reads or read pairs?

Read pairs. The script is designed to count “units of evidence” for gene expression. If both mates map to the same gene, this still only shows that one cDNA fragment originated from that gene. Hence, it should be counted only once.

What happens if the two reads in a pair overlap two different features?

The same as if one read overlaps two features: The read or read pair is counted as ambiguous.

What happens if the mate of an aligned read is not aligned?

For the default mode “union”, only the aligned read determines how the read pair is counted. For the other modes, see their description.

Most of my RNA-Seq reads are counted as “__no_feature”. What could have gone wrong?

Common causes include: - The `--stranded` option was set wrongly. Use a genome browser (e.g., IGV) to check. - The GTF file uses coordinates from another reference assembly as the SAM file. - The chromosome names differ between GTF and SAM file (e.g., `chr1` in one file and `jsut 1` in the other).

Which overlap mode should I use?

When I wrote `htseq-count`, I was not sure which option is best and included three possibilities. Now, several years later, I have seen very few cases where the default `union` would not be appropriate and hence tend to recommend to just stick to `union`.

I have a GTF file? How do I convert it to GFF?

No need to do that, because GTF is a tightening of the GFF format. Hence, all GTF files are GFF files, too. By default, `htseq-count` expects a GTF file.

I have a GFF file, not a GTF file. How can I use it to count RNA-Seq reads?

The GTF format specifies, inter alia, that exons are marked by the word `exon` in the third column and that the gene ID is given in an attribute named `gene_id`, and `htseq-count` expects these words to be used by default. If your GFF file uses a word other than `exon` in its third column to mark lines describing exons, notify `htseq-count` using the `--type` option. If the name of the attribute containing the gene ID for exon lines is not `gene_id`, use the `--idattr`. Often, it is, for example, `Parent`, `GeneID` or `ID`. Make sure it is the gene ID and not the exon ID.

How can I count overlaps with features other than genes/exons?

If you have a GFF file listing your features, use it together with the `--type` and `--idattr` options. If your feature intervals need to be computed, you are probably better off writing your own counting script (provided you have some knowledge of Python). Follow the tutorial in the other pages of this documentation to see how to use HTSeq for this.

Why is the sum of all counts different from the number of reads in my FASTQ file?

A read with more than one reported alignment appears only once in the FASTQ file, but several times in the SAM file (once for each alignment), and each time htseq-count encounters one, it increased the `__alignment_not_unique` counter by one. Therefore, multiply aligned reads are counted multiple times. Other possible reasons: Non-aligned reads may be present in the FASTQ but missing in the SAM file. And in case of paired-end data, many aligners report pairs of which only one mate has been aligned in an incorrect manner by omitting the record for the unaligned mate, which can cause inconsistencies.

How should I cite htseq-count in a publication?

Please cite HTSeq as follows: “S Anders, T P Pyl, W Huber: *HTSeq — A Python framework to work with high-throughput sequencing data*. Bioinformatics, 2014, in print; online at [doi:10.1093/bioinformatics/btu638](https://doi.org/10.1093/bioinformatics/btu638)”.