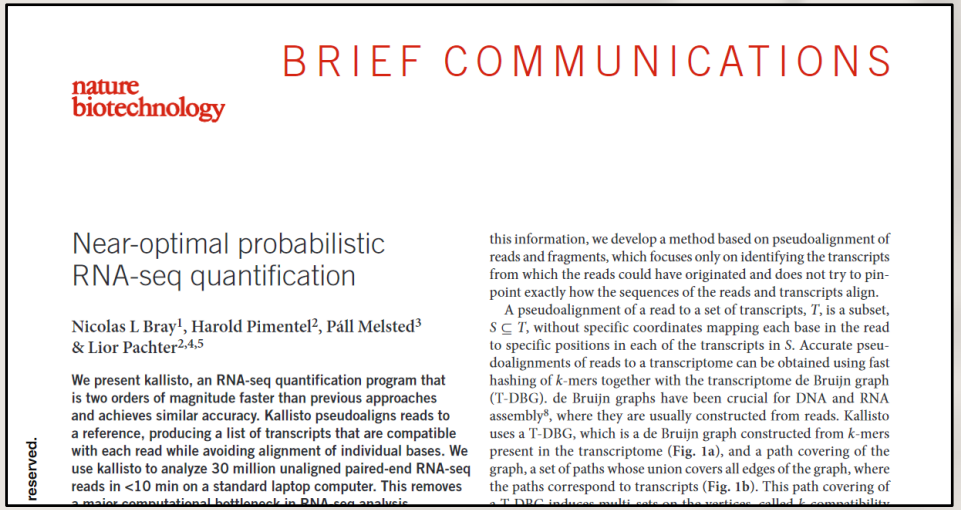
Kallisto

"Near-optimal **probabilistic** RNA-seq quantification"

Webpage: https://pachterlab.github.io/kallisto/

What is kallisto?



BRIEF COMMUNICATIONS

nature biotechnology

Near-optimal probabilistic RNA-seq quantification

Nicolas L Bray[1], Harold Pimentel[2], Páll Melsted[3] & Lior Pachter[2,4,5]

We present kallisto, an RNA-seq quantification program that is two orders of magnitude faster than previous approaches and achieves similar accuracy. Kallisto pseudoaligns reads to a reference, producing a list of transcripts that are compatible with each read while avoiding alignment of individual bases. We use kallisto to analyze 30 million unaligned paired-end RNA-seq reads in <10 min on a standard laptop computer. This removes a major computational bottleneck in RNA-seq analysis.

this information, we develop a method based on pseudoalignment of reads and fragments, which focuses only on identifying the transcripts from which the reads could have originated and does not try to pinpoint exactly how the sequences of the reads and transcripts align.

A pseudoalignment of a read to a set of transcripts, T, is a subset, $S \subseteq T$, without specific coordinates mapping each base in the read to specific positions in each of the transcripts in S. Accurate pseudoalignments of reads to a transcriptome can be obtained using fast hashing of k-mers together with the transcriptome de Bruijn graph (T-DBG). de Bruijn graphs have been crucial for DNA and RNA assembly[8], where they are usually constructed from reads. Kallisto uses a T-DBG, which is a de Bruijn graph constructed from k-mers present in the transcriptome (Fig. 1a), and a path covering of the graph, a set of paths whose union covers all edges of the graph, where the paths correspond to transcripts (Fig. 1b). This path covering of

- Program for **<u>quantifying abundances of transcripts</u>**
  - target sequences using high-throughput sequencing reads
  - Bulk/Single cell RNA-seq data
- Based on ***pseudoalignment*** (alignment-free)
- *„we develop a method based on pseudoalignment of reads and fragments, which focuses only on identifying the transcripts from which the reads could have originated and does not try to pinpoint exactly how the sequences of the reads and transcripts align.“*

What is kallisto?

- **High-speed** (30 mil. human reads in less than 3 minutes on Mac desktop / index ca. 10 min.)
- Pseudoalignment of reads preserves key information needed for quantification and kallisto is therefore not only fast, but also as **accurate** as existing quantification tools
- Pseudoalignment procedure is robust to **errors** in the reads – in many benchmarks kallisto significantly outperforms existing tools

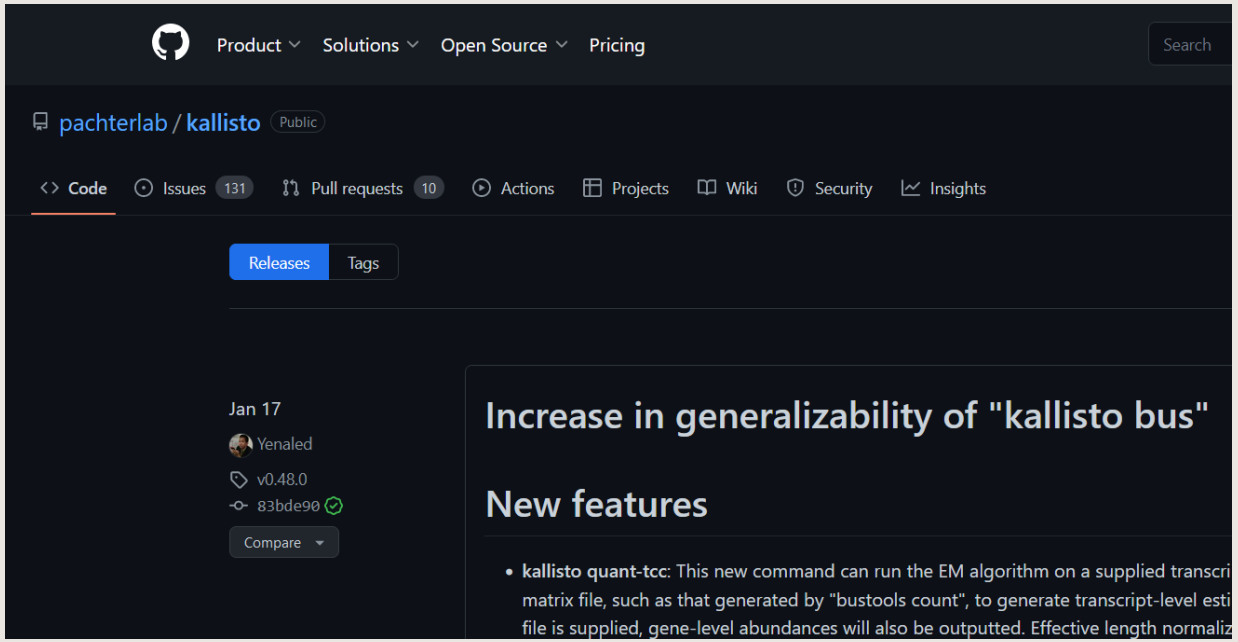
What is kallisto?

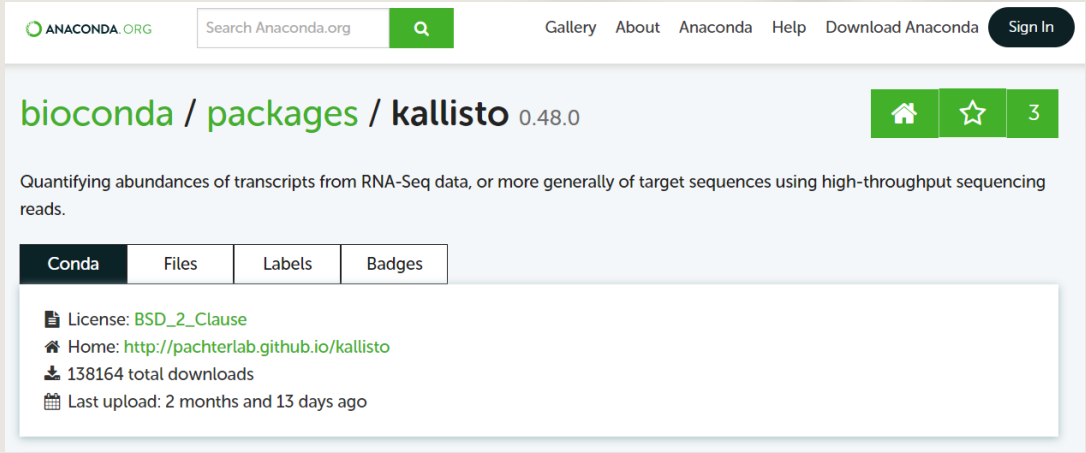
- **Released:** 2015/2016
- **Latest release:** Jan 17 2022
- **Distribution:** Windows, Mac/Linux, Rock64

Releases

The **kallisto** GitHub repository is here.

Version	Date					
Release notes: v0.46.1	October 04, 2019	Mac	Linux	Windows	Rock64	Source
Release notes: v0.46.0	June 12, 2019	Mac	Linux	Windows	Rock64	Source
Release notes: v0.45.0	November 17, 2018	Mac	Linux	Windows	Rock64	Source

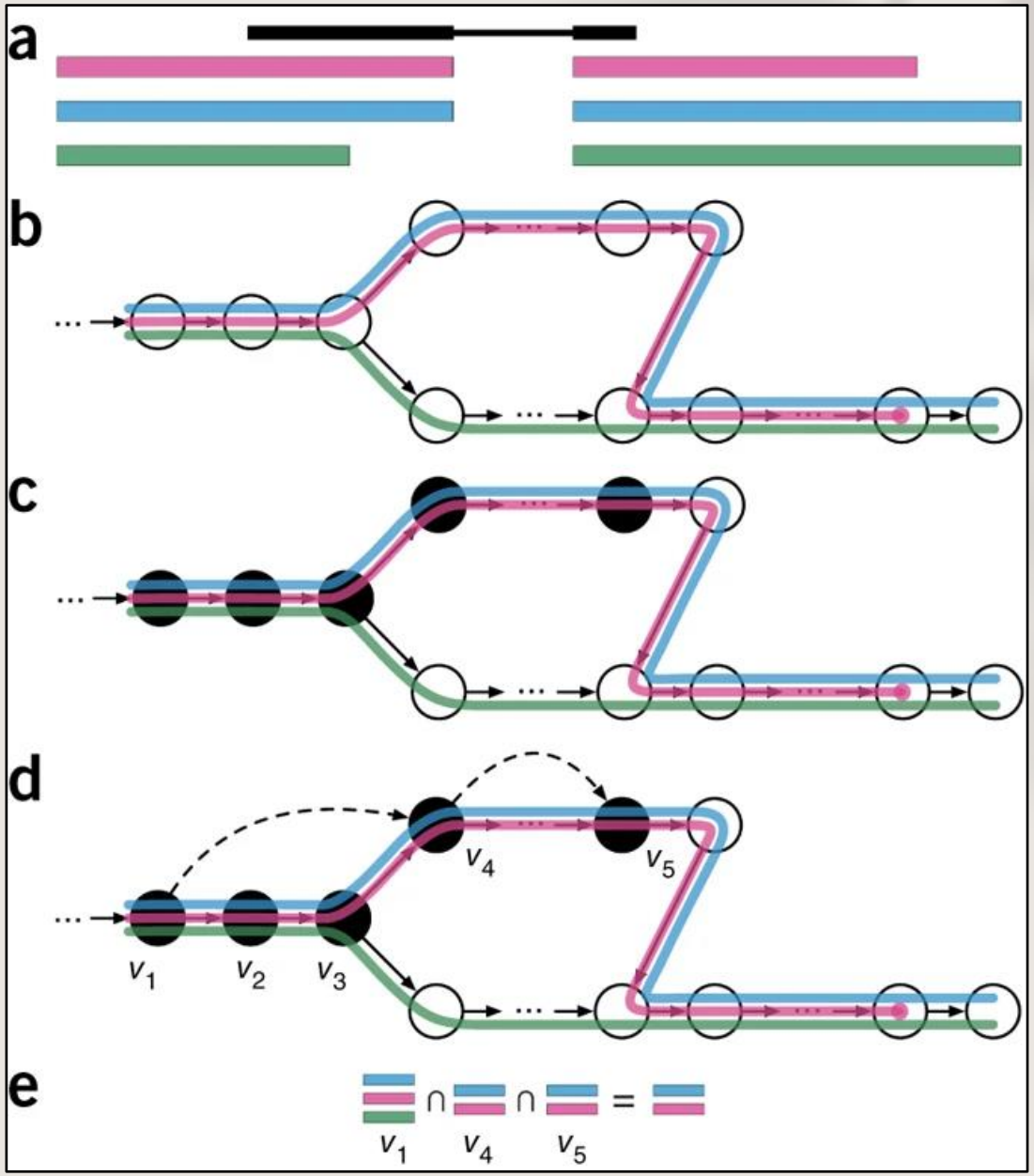




Webpage: https://pachterlab.github.io/kallisto/
GitHub: https://github.com/pachterlab/kallisto/
Bioconda: https://anaconda.org/bioconda/kallisto/

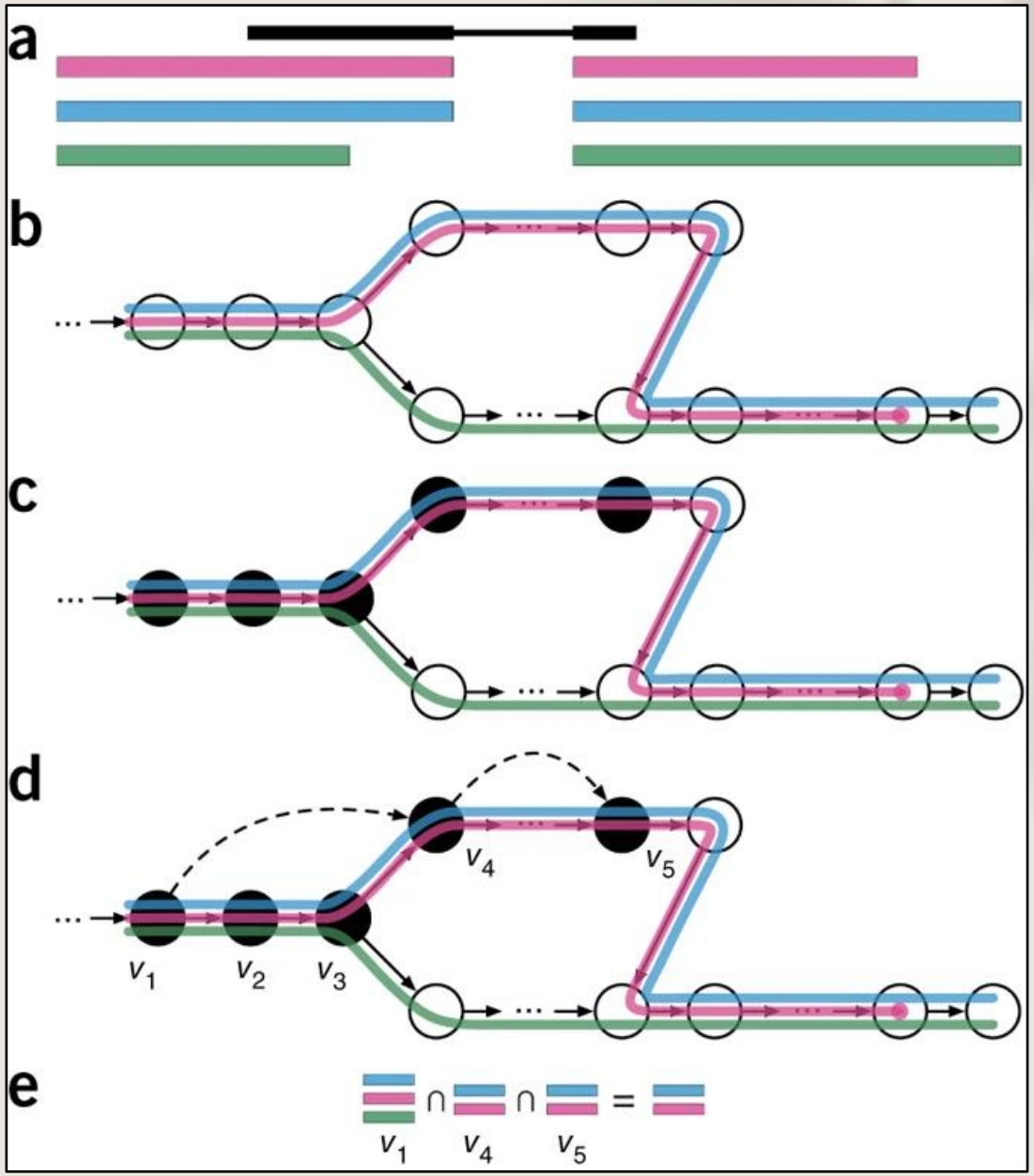
How does it work?

- a) construction of **de Bruijn graph** from k-mers present in the transcriptome (T-DBG)
- b) path covering corresponding to transcripts = **compatibility classes**; nodes = k-mers
- c) association of compatibility classes to an error-free read = representing as a **path in the graph**, based on the similarity of k-mers



How does it work?

- d) Removing **redundant** k-mers for the pseudoalignment = *speed increase*
- e) An **equivalence** class for a read is a multi-set of transcripts associated with the read
- ideally it represents the transcript a read could have originated from
- equivalence classes are quantified via use of Expectation Maximization (EM) algorithm to determine maximum likelihood



How do you use it?

- **1. Indexing**

```
kallisto index -i transcripts.idx transcripts.fasta.gz
```

- **2. Quantification**

```
kallisto quant -i index -o output pairA_1.fastq pairA_2.fastq pairB_1.fastq pairB_2.fastq
```

```
kallisto quant -i index -o output --single -l 200 -s 20 file1.fastq.gz file2.fastq.gz file3.fastq.gz
```

How do you use it?

- **Outputs:**
  - table in *.h5 / *.tsv
  - run information (*.json)

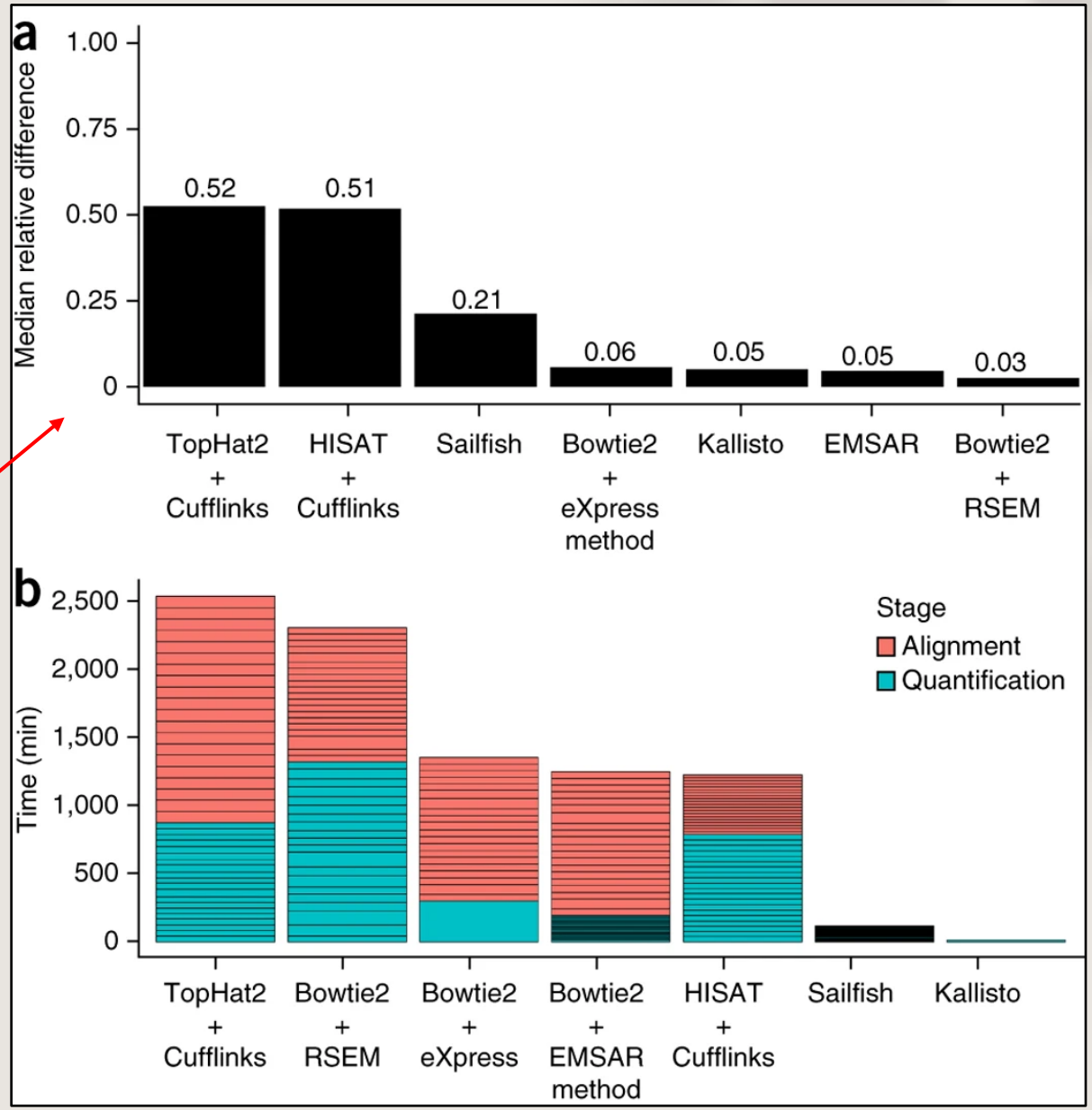
```
{
        "n_targets": 14,
        "n_bootstraps": 30,
        "n_processed": 10000,
        "n_pseudoaligned": 9413,
        "n_unique": 7174,
        "p_pseudoaligned": 94.1,
        "p_unique": 71.7,
        "kallisto_version": "0.44.0",
        "index_version": 10,
        "start_time": "Tue Jan 30 09:34:31 2018",
        "call": "kallisto quant -i transcripts.kidx -
ads_2.fastq.gz"
}
```

```
total 568
-rw-r--r--  1 username  staff  282480 May  3 10:10 abundance.h5
-rw-r--r--  1 username  staff     589 May  3 10:10 abundance.tsv
-rw-r--r--  1 username  staff     227 May  3 10:10 run_info.json
```

target_id	length	eff_length	est_counts	tpm
ENST00000513300.5	1924	1746.98	102.328	11129.2
ENST00000282507.7	2355	2177.98	1592.02	138884
ENST00000504685.5	1476	1298.98	68.6528	10041.8
ENST00000243108.4	1733	1555.98	343.499	41944.9
ENST00000303450.4	1516	1338.98	664	94221.8
ENST00000243082.4	2039	1861.98	55	5612.36
ENST00000303406.4	1524	1346.98	304.189	42908.2
ENST00000303460.4	1936	1758.98	47	5076.85
ENST00000243056.4	2423	2245.98	42	3553.05
ENST00000312492.2	1805	1627.98	228	26609.9
ENST00000040584.5	1889	1711.98	4295	476675
ENST00000430889.2	1666	1488.98	623.628	79578.2
ENST00000394331.3	2943	2765.98	85.6842	5885.85
ENST00000243103.3	3335	3157.98	962	57879.3

Why should you use it?

- Test simulation
  - 20 RNA-seq simulations/experiments
  - Curated reference sample
  - 75 bp paired-end RNA-seq reads
  - 30 mil. reads
  - qPCR control for transcript abundance
  - efficiency testing



Why you should (or should not?) use it?

- **Accuraccy**
  - Uses T-DBG graph – deals with multimapping reads via path covering (compatibility / equivalent classes) and maximum likelihood algorithm (also for overlaps)
  - Relies on high-quality transcriptome for indexing
  - Does not discard reads with low mapping rates – if there is not a better match, these reads are pseudoaligned due to ML algorithm even though there is only a single k-mer match
- **Speed**
  - Removes k-mers where sequencing errors are observed (can't be found in the index)
  - Removes redundant k-mers from computation
- **Resources**
  - Multithreading (all datasets in parallel)
  - Relatively low RAM and CPU usage (small laptop test runtime: 10 minutes)

Thank you for your attention!