Bits of DNA

Reviews and commentary on computational biology by Lior Pachter

Near-optimal RNA-Seq quantification with kallisto

May 10, 2015 in papers, q-bio.QM, RNA-Seq | Tags: bootstrap, de Bruijn graph, kallisto, pseudoalignment, RNA-Seq, Sailfish, target de Bruin graph

Today I posted the preprint N. Bray, H. Pimentel, P. Melsted and L. Pachter, Near-optimal RNA-Seq quantification with kallisto to the arXiv. It describes the RNA-Seq quantification program **kallisto**. [Update April 5, 2016: a revised version of the preprint has been published: Nicolas L. Bray, Harold Pimentel, Páll Melsted and Lior Pachter, Near-optimal probabilistic RNA-Seq quantification, Nature Biotechnology (2016), doi:10.1038/nbt.3519 published online April 4, 2016.]

The project began in August 2013 when I wrote my second blog post, about another arXiv preprint describing a program for RNA-Seq quantification called Sailfish (now a published paper). At the time, a few students and postdocs in my group read the paper and then discussed it in our weekly journal club. It advocated a philosophy of "lightweight algorithms, which make frugal use of data, respect constant factors and effectively use concurrent hardware by working with small units of data where possible". Indeed, two themes emerged in the journal club discussion:

1. Sailfish was much faster than other methods by virtue of being *simpler*.

2. The simplicity was to replace *approximate* alignment of *reads* with *exact* alignment of *k-mers*. When reads are shredded into their constituent *k*-mer "mini-reads", the difficult read -> reference alignment problem in the presence of errors becomes an exact matching problem efficiently solvable with a hash table.

We felt that the shredding of reads must lead to reduced accuracy, and we quickly checked and found that to be the case. In fact, in our simulations, we saw that Sailfish significantly underperformed methods such as RSEM. However the fact that simpler was ***so*** much faster led us to wonder whether the prevailing wisdom of seeking to improve RNA-Seq analysis by looking at increasingly complex models was ill-founded. Perhaps simpler could be not only fast, but also accurate, or at least close enough to best-in-class for practical purposes.

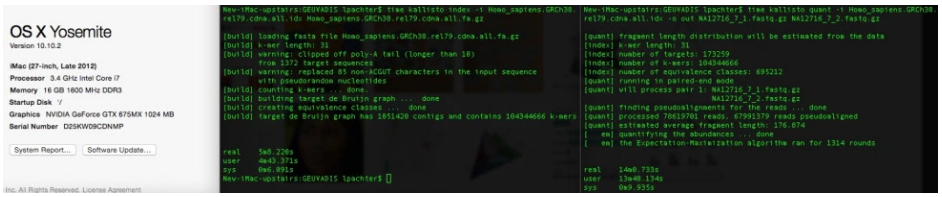
After thinking about the problem carefully, my (now former) student Nicolas Bray realized that the key is to abandon the idea that alignments are necessary for RNA-Seq quantification. Even Sailfish makes use of alignments (of *k*-mers rather than reads, but alignments nonetheless). In fact, thinking about all the tools available, Nick realized that every RNA-Seq analysis program was being developed in the context of a "pipeline" of first aligning reads or parts of them to a reference genome or transcriptome. Nick had the insight to ask: what can be gained if we let go of that paradigm?

By April 2014 we had formalized the notion of "pseudoalignment" and Nick had written, in Python, a prototype of a pseudoaligner. He called the program kallisto. The basic idea was to determine, for each read, not *where* in each transcript it aligns, but rather which transcripts it is *compatible* with. That is asking for a lot less, and as it turns out, pseudoalignment can be *much* faster than alignment. At the same time, the information in

pseudoalignments is enough to quantify abundances using a simple model for RNA-Seq, a point made in the isoEM paper, and an idea that Sailfish made use of as well.

Just how fast is pseudoalignment? In January of this year Páll Melsted from the University of Iceland came to visit my group for a semester sabbatical. Páll had experience in exactly the kinds of computer science we needed to optimize kallisto; he has written about efficient *k*-mer counting using the bloom filter and de Bruijn graph construction. He translated the Python kallisto to C++, incorporating numerous clever optimizations and a few new ideas along the way. His work was done in collaboration with my student Harold Pimentel, Nick (now a postdoc with Jacob Corn and Jennifer Doudna at the Innovative Genomics Initiative) and myself.

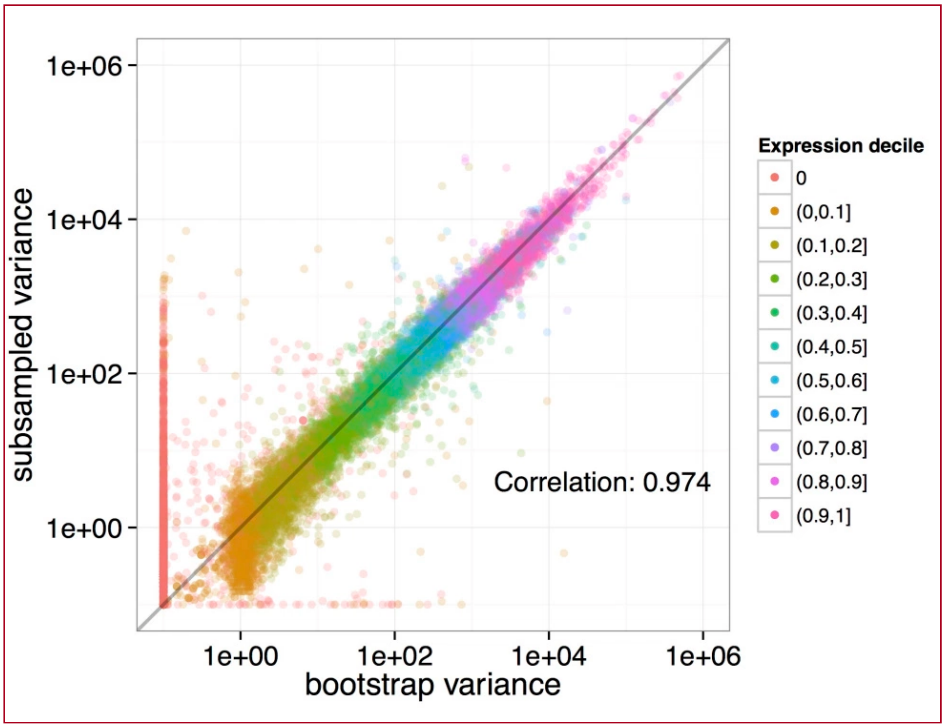
The screenshot below shows kallisto being used on my 2012 iMac desktop to build an index of the human transcriptome (5 min 8 sec), and then quantify 78.6 million GEUVADIS human RNA-Seq reads (14 min). When we first saw these results we thought they were simply too good to be true. Let me repeat: **The quantification of 78.6 million reads takes 14 minutes on a standard desktop using a single CPU core.** In some tests we've gotten even faster running times, up to 15 million reads quantified per minute.



The results in our paper indicate that kallisto is not just fast, but also very accurate. This is not surprising: underlying RNA-Seq analysis are the alignments, and although kallisto is pseudoaligning instead, it is almost always only the compatibility information that is used in actual applications. As we show in our paper, from the point of view of compatibility, the pseudoalignments and alignments are almost the same.

Although accuracy is a primary concern with analysis, we realized in the course of working on kallisto that speed is also paramount, and not just as a matter of convenience. The speed of kallisto has three major implications:

1. It allows for efficient bootstrapping. All that is required for the bootstrap are reruns of the EM algorithm, and those are particularly fast within kallisto. The result is that we can accurately estimate the uncertainty in abundance estimates. One of my favorite figures from our paper, made by Harold, is this one:



It is based on an analysis of 40 samples of 30 million reads subsampled from 275 million rat RNA-Seq reads. Each dot corresponds to a transcript and is colored by its abundance. The *x*-axis shows the variance estimated from kallisto bootstraps on a single subsample while the *y*-axis shows the variance computed from the different subsamples of the data. We see that the bootstrap recapitulates the empirical variance. This result is non-trivial: the standard dogma, that the technical variance in RNA-Seq is "Poisson" (i.e. proportional to the mean) is false, as shown in Supplementary Figure 3 of our paper (the correlation becomes 0.64). Thus, the bootstrap will be invaluable when incorporated in downstream application and we are already working on some ideas.

2. It is not just the kallisto quantification that is fast; the index building, and even compilation of the program are also easy and quick. The implication for biologists is that RNA-Seq analysis now becomes interactive. Instead of "freezing" an analysis that might take weeks or even months, data can be explored dynamically, e.g. easily quantified against different transcriptomes, or re-quantified as transcriptomes are updated. The ability to analyze data locally instead of requiring cloud computation means that analysis is portable, and also easily secure.

3. We have found the fast turnaround of analysis helpful in improving the program itself. With kallisto we can quickly check the effect of changes in the algorithms. This allows for much faster debugging of problems, and also better optimization. It also allows us to release improvements knowing that users will be able to test them without resorting to a major computation that might take months. For this reason we're not afraid to say that some improvements to kallisto will be coming soon.

As someone who has worked on RNA-Seq since the time of 32bp reads, I have to say that kallisto has personally been extremely liberating. It offers freedom from the bioinformatics core facility, freedom from the cloud, freedom from the multi-core server, and in my case freedom from my graduate students– for the first time in years I'm analyzing tons of data on my own; because of the simplicity and speed I find I have the time for it. Enjoy!

SHARE THIS:

Email Print More

Like

2 bloggers like this.

RELATED

Straining metagenomics
In "papers"

A sleuth for RNA-Seq
In "papers"

Sailfish
In "*Seq"

61 comments

Comments feed for this article

May 11, 2015 at 2:25 am
Jami M

Hi!

This looks like really good work!

Could this idea be used to generate very fast “(raw) gene counts” (such that instead of of using transcriptome one would use gene-ome [just sequences of genes]; of course this would ignore the reads mapping on splice sites but this might be ok for DEG analysis) which could be used directly for DEG analsys with edgeR/DESeq/etc.?

Jami

Reply

May 11, 2015 at 10:13 am
Lior Pachter

kallisto transcript counts can be summed to obtain gene counts. We will be providing a convenient infrastructure for doing that with the program sleuth that is forthcoming. I don’t recommend working with “raw gene counts” for any purpose, including differential analysis, for reasons explained here: http://www.nature.com/nbt/journal/v31/n1/full/nbt.2450.html

Reply

July 7, 2015 at 4:32 am
Amit

Hi Lior
Presumably if you simply sum the transcript counts, you will be double-counting reads that hit shared exons between transcripts, and therefore overestimating the gene counts? If so I guess something like ‘summarizeoverlaps’ is the

answer, no…
Amit

July 7, 2015 at 4:39 am
Lior Pachter

One should not sum transcript ***counts*** but rather transcript ***abundances***. In the case of the former, you are correct that the arithmetic goes awry in estimating gene counts. The latter approach allows one to obtain consistent estimates of gene abundances.

June 3, 2016 at 8:07 am
Brian Haas

Can you please elaborate on how the arithmetic goes awry when determining counts of reads per gene based on the counts of reads per transcript? I was under the impression that gene expression levels can be computed by simply summing up the TPM values for all isoforms of that gene. Similarly, if one wanted counts of reads per gene, since the reads that multi-map (multiple equivalence classes in kallisto parlance) are fractionally assigned to isoforms based on the EM, one could simply sum up the isoform read counts to obtain gene-level read counts without a worry of multi-counting reads. Where is my logic going awry? Thanks in advance!

June 4, 2016 at 2:43 pm
Lior Pachter

You are right that gene *counts* can be correctly estimated by summing up the estimated counts associated to the constitutive isoforms. Similarly, gene *abundances* (say as measured in TPM units) can be estimated by summing up the abundances of the constitutive isoforms. The issue is that while isoform counts can serve as proxies for isoform abundances across samples (after normalizing for sequencing depth) gene counts cannot.

Say there is an isoform with counts x1 in sample 1 and x2 in sample 2 (assume depth of sequencing is the same). Then the abundances of the isoform should be computed as x1/L and x2/L (where L is its length). It follows that the relative abundance is therefore (x1/L) / (x2/L) = x1/x2. The length L canceled out!

However at the gene level, if the counts are y1 and y2 in the two samples, it is *not* true that the relative abundance of the gene is y1/y2. That is because the constitutive isoforms have different lengths so the ratio of the counts does not simplify to the ratio of *abundance* like it does in the isoform case. In fact, the ratio of counts does not even transform to ratio of abundance in a nice way: it can be higher, or lower… and extremely poor as a proxy depending on the (length) structure and abundances of the constituent isoforms. I tried to explain this in my talk on stories from the supplement. Hope that helps.

May 11, 2015 at 3:07 am
Paul Korir

Interesting idea. I find the title confusing though. What is optimality in reference to? Accuracy? Speed?

Reply

May 11, 2015 at 10:11 am
Lior Pachter

Both. In terms of speed, we find that in many tests kallisto takes less than 5x the amount of time it takes to read the number of characters in the read file (using wc). Considering that kallisto is doing pseudoalignment and quantification, and character count is a lower bound, it appears to be near-optimal. In terms of accuracy, we are comparable to (and in many cases exceed) the performance of state-of-the-art methods.

Reply

May 11, 2015 at 2:29 pm
Anon

Out of curiosity, what makes you think that no further gains in accuracy can be made? You're argument about optimality for speed is compelling, just curious to hear your thoughts on why you think the same is true for accuracy. Thanks!

May 11, 2015 at 2:54 pm
Lior Pachter

Our simulation experiments with long reads and no error give us a pretty good upper bound for performance. Again, we're close.

May 11, 2015 at 11:45 am
Robert Flight

With regards to the error not being "poisson", or proportional to the mean. I don't think your figure above or figure S3 actually show that at all. If you have to use a log-scale to make it linear, then there is a proportional variance of some kind.

To truly investigate the error structure, I would look at mean counts (RSEM, x axis) plotted against the standard deviation of the counts. I would expect that as the mean count value increases, the variance of the standard deviation will widen as well, indicating that the variance is proportional to the mean.

Reply

May 11, 2015 at 12:46 pm
pmelsted

The rainbow plot and S3 are displayed as a log-scale, but the correlation is based on raw values. The reason it is displayed on a log-scale is the large dynamic range of expression; in a plot of the raw values all the low expression transcripts are squished in the lower left corner.

We will be releasing a makefile for all the analysis for the paper that will allow others, such as yourself, to explore the data if you wish.

Reply

May 11, 2015 at 3:09 pm
Mike Lin

Are real genetic polymorphisms treated equivalently to sequencing errors in pseudoalignment? (Does the preprint use "error" to refer to both?)
Could one then increase pseudoalignment sensitivity, perhaps by epsilon, by representing common variants in the T-DBG?

Reply

May 13, 2015 at 12:05 am
Marnie Dunsmore

Lior, this seems like ground breaking work. Key to the pseudoalignment process your team has developed seems to be this statement from the paper "However with very high probability, an error in a k-mer will result in it not appearing in the transcriptome and such k-mers are simply ignored. The issue of errors is also ameliorated by a technique that we implemented to improve the efficiency of kallisto's pseudoalignment process by removing redundant k-mers from the computation based on information contained in the T-DBG."

I will have to read the Supplement that describes how redundant k-mers are removed.

I'm interested in why an error in a k-mer simply results, with high probability, in not appearing in the transcriptome.

This seems to be a key reason was to why your pseudoalignment algorithm works.

Very interesting. Can you talk more about that?

Reply

May 13, 2015 at 7:14 am
Jim Auer

Excellent question. I was wondering the very same thing.

Reply

May 13, 2015 at 7:24 am
Lior Pachter

Thanks for your question. The number of k-mers in the human transcriptome is on the order of 80 million, or O(10^7). On the other hand, the number of k-mers for k=21 is O(10^13) and for k=13 O(10^18). Therefore a random k-mer is highly unlikely to appear in the transcriptome. Granted, changing a transcriptome k-mer by one base does not result in one that behaves like a random k-mer, because genome duplications have results in sequences that are highly similar/repetitive (this of course

means that many k-mers match to multiple places in addition to their true location of origin, but that is not a problem in pseudoalignment due to the intersection). As far as errors go, in our experiments we have still found that random changes rarely result in k-mers that exist elsewhere in the transcriptome. We can add results of sensitivity and specificity of pseudoalignment that make the case to our arXiv preprint shortly.

Reply

May 16, 2015 at 1:36 am
Jami

You wrote "The number of k-mers in the human transcriptome is on the order of 80 million, or O(10^7)." Is this for k=31?

May 16, 2015 at 9:00 am
Lior Pachter

The exact number can be easily counted with kallisto. For example, using the ENSEMBL release 79 human transcriptome we've made available at http://bio.math.berkeley.edu/kallisto/transcriptomes/ with k=31 the number of k-mers is 104344666 (~100 million). You can reproduce this as follows:

Air13:kallisto_tests lpachter$ kallisto index -k 31 -i Homo_sapiens.GRCh38.rel79.cdna.all.idx Homo_sapiens.GRCh38.rel79.cdna.all.fa.gz

[build] loading fasta file Homo_sapiens.GRCh38.rel79.cdna.all.fa.gz
[build] k-mer length: 31
[build] warning: clipped off poly-A tail (longer than 10)
from 1372 target sequences
[build] warning: replaced 85 non-ACGUT characters in the input sequence with pseudorandom nucleotides
[build] counting k-mers … done.
[build] building target de Bruijn graph … done
[build] creating equivalence classes … done
[build] target de Bruijn graph has 1051420 contigs and contains 104344666 k-mers

I should have specified that the number of ~80 million I gave was based on a reduced human transcriptome of polyadenylated transcripts that we have been using in-house.

If you're curious the number of 21-mers for the rel.79 transcriptome is just slightly lower than for k=31 at 101034694.

May 13, 2015 at 4:34 am
Roye Rozov

With the timings your report here, you guys should suggest to NCBI to put your program on their servers. People could then

just run it & download the quantification results instead of the reads in many cases 🙂

Reply

May 13, 2015 at 7:01 am
Lior Pachter

I agree 🙂

For now, Andrew McKenzie has written up a nice post describing how to quantify samples in the SRA easily on your desktop:

How to run kallisto on NCBI SRA RNA-Seq data for differential expression using the mac terminal

One advantage of being able to do it locally is that quantifications can be easily (re)computed with respect to custom annotations.

Reply

May 13, 2015 at 7:24 am
Roye Rozov

Thanks, and I see your point. But it seems like moving the data is now the real bottleneck- annotations are tiny in comparison. Your development meshes nicely with this one: http://biorxiv.org/content/early/2015/03/26/017087

Conceivably, you can now find the 'right' data set with the above, and analyse it on the spot quickly.

May 13, 2015 at 6:10 am
Jason Merkin

Hi Lior,

I thought it would be better explained with > 140 characters. I asked on twitter about reference bias due to common or private variants. I thought the method uses only perfect read matches, so then reads would have poorer (or non-existent) mapping in the vicinity of variants, leading to lower read density for individuals containing these variants.

Can you elaborate a little more on this? If this is addressed and I missed it or I misunderstood something, I'll go back and read more carefully.

Reply

May 13, 2015 at 6:57 am
Lior Pachter

Hi Jason,

Thanks for your question. For a read to pseudoalign to a transcript it is true that at least *one* of its kmers must match exactly to the transcript. But unlike Bowtie/2 or most other alignment tools, this k-mer does not have to be in a specific location in the read (i.e. a seed). Since k is now small relative to read length, it is extremely unlikely that a read will not map due to a variant in the transcriptome (or for that

matter an error in the read). The extent of "extremely" depends of course on the number of differences as measured across the read, but single variants leave a lot of real estate for a k-mer to match. Having said that, of course its true that there will be some reference bias, because a variant in the transcriptome affects probability of matching ever so slightly. Therefore, it is ideal to pseudoalign to the haplotypes of an individual (with the added benefit of allele-specific expression calls to boot). We've actually benchmarked kallisto's accuracy for ASE alongside other programs and it performs very well, in a manner similar to the results for standard reference quantification. Since, as you highlight, this is an important point, we'll add it to the preprint shortly.

Reply

May 13, 2015 at 7:18 am
Jason Merkin

Thanks, Lior. I was conflating error-free read match with error-free kmer match.

May 13, 2015 at 8:49 am
Chris Klijn

Hi Lior,

This looks like a very big leap in RNA-Seq processing, thanks for the work. I gather that overall accuracy is good, but did you observe any gene-specific deviations? For example, is there a reduction of accuracy for genes with many expressed pseudogenes, or closely related gene families (Keratins)?

Reply

May 13, 2015 at 8:58 am
Lior Pachter

Thanks Chris! We were asked the question about gene families by a number of people so we have examined our benchmarks with a focus on such genes, and the short answer is that the results look very good and although, as expected, there is a reduction of accuracy for all programs, kallisto performs well. We'll put them up in a revision to the arXiv preprint shortly.

Reply

May 13, 2015 at 9:28 am
Chris Klijn

Thanks for the quick reply, looking forward to the analysis!

May 15, 2015 at 11:43 am
Shawn Driscoll

1 minute 28 seconds to quantify 34 million single-end 50bp reads. I don't know how well the expression estimation works with short single-end reads but that's just silly fast.

Reply

May 15, 2015 at 12:21 pm
Shawn Driscoll

I should add that it was against 94,000 mouse transcripts. Not bad!

Reply

May 18, 2015 at 8:42 am
Jason Merkin

Can you elaborate on the commercial license? Is that to prevent a company from incorporating it into a product they sell, or so that someone like me (working for pharma) can't use it without purchasing a license?

Reply

May 18, 2015 at 9:00 am
Lior Pachter

Hi Jason,

The goal of the license is not to prevent use but to follow the guidelines of UC Berkeley on software disclosure. We certainly hope that companies will both incorporate it into products they sell, and that also individuals like yourself will make use of it in pharma and other industries. For pricing and terms please contact the office of technology licensing at the address in the license.

Thanks,
Lior

Reply

May 19, 2015 at 8:55 am
Eyal Peer

Hi Lior.
This seems like groundbreaking and very exciting work. I've already downloaded and installed kallisto on our server and I have to admit it is silly fast.
I have just one question: Why do you write in your FAQ that kallisto can't be used for finding differentialy expressed genes?
Thanks,
Eyal

Reply

May 19, 2015 at 9:18 am
Lior Pachter

Hi Eyal,

Sorry for the confusion. The program kallisto is itself not a tool for differential analysis of experiments, but that is not to say the quantifications are not useful as a starting point for analysis. In fact, we believe the bootstraps that kallisto provides are essential for differential expression analysis (of both genes and isoforms). The thing is we are developing a companion tool for that purpose, called sleuth, that will work together with kallisto. So it will be kallisto-sleuth that finds differentially expressed genes, and not kallisto alone.

People have asked us when sleuth will be available, and the answer is “soon”. We’ve got a lot of functionality implemented but want to make sure that it is released in the same polished/finalized form as kallisto. Because there are many modes of differential analysis, and multiple issues to consider, we are trying to be both comprehensive and careful. The feedback we are receiving from initial users of kallisto is also very helpful in the development of sleuth.

Thanks again,
Lior

Reply

May 19, 2015 at 10:01 am
Eyal Peer

Thanks for the quick reply.
Looking forward to using kallisto-sleuth soon.

Good luck!

May 19, 2015 at 11:31 am
Shawn Driscoll

Hi Lior,

Do you feel we are at a point now (finally) with kallisto and maybe RSEM or eXpress that we can make pretty reliable inference of differential splicing (via isoform abundances) and, if so, do you or your labbies have any preference in differential testing approaches for evaluating changes in a gene locus? I believe cuffdiff attempted to do something like this by running an analysis to test if the distribution of expression to isoforms in a loucs was the same between conditions but I honestly never had any decent looking results from that pipeline. Then there is EBSeq which claims to incorporate some additional variance model for the dependent nature of isoform expressions within a gene locus but EBSeq seems to like to produce a lot of false positive calls so I never use it and besides that I’m partially at odds with the idea you can call a single isoform differentially expressed without considering the entire locus. Currently when we need to infer differential splicing I have had the most success in finding alternative events via the metrics described here (http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3546801/) followed by my own analysis which is based loosly on how DESeq and the like work. When the question is about isoforms, however, and all you have are a bunch of alternative donors and acceptors it can be a very messy pile of numbers and p-values to deal with (especially for the molecular biologists I work with) though maybe not impossible to string together to imply some isoform biases. I’d love to be able to confidently hand off to them a list of significantly different locus-level isoform usage results but up to now I haven’t seen much confidence from people in the isoform level abundace calls (i.e. people seem to typically use these programs and then sum the isoform estimates to gene level estimates). Thanks for your time.

By the way I LOVE that kalliso actually exports all of the bootstrap iteration abundance estimate results and not just summary statistics. That really gets me pumped up.

Shawn

Reply

May 19, 2015 at 7:37 pm
Ariel Schwartz (@ArielSSchwartz)

Great work. Very useful. I wish you could do away with the commercial license restriction though, since it could severely limit the adoption of kallisto, especially as a BaseSpace app where it could be very powerful.

I have a few questions/comments about the manuscript.

It looks like you are using bowtie2 in the default global alignment mode since RSEM only supports that mode. On the other hand, since kallisto ignores k-mers with errors it is essentially "soft-clipping" the reads. The reads from the GEUVADIS sample have somewhat lower quality bases at the beginning and end of the reads, so a better comparison of (pseudo)alignment rate will be after quality-trimming the raw reads. A similar issue arises with reads that run into the poly-A tails (and UTRs if they are not part of the reference).

Also, based on kallisto's FAQ page you are ignoring any strand-specific information, which means that anti-sense transcripts will be aligned and counted towards expression level of protein-coding genes. If I correctly understand kallisto's algorithm, by ignoring incompatible k-mers immature RNA transcripts and non-coding RNA that overlap with coding transcripts will also count towards expression level of coding genes. What is the minimal number of matching k-mers required for a read to be pseudo-aligned? If a read has only a single k-mer from the hash table will it still be pseudo-aligned to the k-compatibility class? I would add an evaluation on a real dataset from a total RNA library (with ribosomal RNA depletion).

Thanks,

Ariel

Reply

May 20, 2015 at 9:52 am
pmelsted

Hi Ariel.

The alignment parameters for bowtie2 used for RSEM can be seen here https://github.com/pachterlab/kallisto_paper_analysis/blob/master/simulations

For the bowtie and kallisto pseudoalignment comparison kallisto required a perfect 31-mer to get any sort of match. bowtie2 would have found this match with the seed strategy (default length of 22, extracting a seed every 10bp) so

the fact that bowtie loses some reads has more to do with the local alignment, which takes errors into account. I think the main difference between the two is that bowtie2 was run with –no-mixed, so both pairs have to map, whereas kallisto requires only one of the ends to map.

The minimal number of k-mers needed to match is 1, however if any two that match are discordant, i.e. their equivalence classes are disjoint, then we don't pseudoalign the read.

Thanks,
Páll

Reply

May 20, 2015 at 10:42 am
Ariel Schwartz (@ArielSSchwartz)

Hi Páll. Bowtie2 is using global alignment (–end-to-end is the default mode) which is very different from the local pseudo-alignments of kallisto. A 75bp read with only 31bp matching the reference will not be aligned by Bowtie2 in global alignment mode. For the purpose of assessing alignment rates you should use bowtie2 in –local mode.

The fact that a single k-mer is sufficient even if the rest of the read does not match the reference could cause a problem in the presence of immature RNA, anti-sense RNA, overlapping non-coding RNA, etc.

For example, if you take the three transcripts in Figure 1 of the paper. An unspliced read that covers the whole region including the intron will be incorrectly assigned to the blue reference transcript, if the k-mers in the intron do not appear anywhere else in the reference.

May 21, 2015 at 9:25 am
Golsheed

Thanks for the great work, extremely useful! I have two questions if you don't mind:
(1) What's the best way to estimate transcript proportions (relative abundance) from Kallisto outputs? can I use the following TPM_t/sum_i(TPM_i)?
(2) Can we use the estimated counts from Kallisto for differential analysis, say, with DESeq2?

Thanks,
Golsheed

Reply

May 25, 2015 at 3:09 pm
devil

On reading the paper, my assessment was that the main high level point was that a simple model for describing an RNA-

seq experiment suffices to do inference fairly accurately (and much more quickly than more elaborateStreaming fragment assignment for real-time analysis of sequencing experiments models used in eXpress). Is this a fair assessment?

The other main idea is that of while hashing reads leads to a loss of information (for example for length 75 reads, with 1% error around 53% reads will have at least one error, which the hashing based approach would throw away), using T-DBG with taking an intersection of compatibility classes corresponding to each vertex, provides a low complexity method of basically hashing reads, without throwing out reads with a few errors.

Also, it is interesting that in around 60% (and quite surprising to me at least), it is mapped to a region of no ambiguity in the T-DBG.

Reply

May 25, 2015 at 4:45 pm
pmelsted

1. Yes, the key point is that the pseudoalignments are all that is needed to gather sufficient statistics for the simple likelihood function.

2. Yes, the errors in reads are devastating for 75bp reads, and even on perfect data treating both ends of a fragment independently is losing information. With the error using 31-mers (the default) makes you robust to errors. There are two ways that this scheme can lose information, a) an error means some k-mers map to a totally different region, we see this happening more with 15-mers then, e.g. 23-mers, b) the error k-mer doesn't appear in the transcriptome so we could possibly lose some specificity, but most of the time this doesn't happen and when it does the fragment still provides useful information.

Reply

May 25, 2015 at 7:10 pm
Lior Pachter

I would add that the speed allows for performing the bootstrap, which in my opinion will finally allow biologists to obtain meaningful isoform-level RNA-Seq results in a statistically rigorous way that takes into account the full extent of ambiguity introduced by multi-mapping reads.

Reply

May 28, 2015 at 12:19 pm
RvP

Hi,

Is the github of kallisto repository up? It seems to be declining requests to clone.

Thanks,
RvP.

Reply

May 28, 2015 at 1:00 pm
haroldpimentel

Hi RvP,

Which URL are you using to clone. I just realized we had the old URL in the 'INSTALL.md' file. You can clone using:

git clone https://github.com/pachterlab/kallisto.git

Please let us know if this works.

Thanks,

Harold

Reply

May 28, 2015 at 2:28 pm
RvP

Yes it works. Thanks Harold

May 31, 2015 at 5:17 pm
devil

I was wondering if the test data provided with the software is real data or just artificially constructed data? It looks like for each read one of the pairs there is has a burst of errors and the other is completely clean, which is very surprising!!

Reply

June 5, 2015 at 4:33 pm
Jamie

This seems to be set up quite well for ribosome profiling data. Has that been tested?

Reply

June 5, 2015 at 8:40 pm
Lior Pachter

Excellent point- and although we haven't tested the idea yet it is certainly in the plans.

Reply

June 18, 2015 at 2:57 am
Rune

Hi Lior,
I am going to quantify RNAseq reads, to confirm the expression of genes that has previously been indentified to contain mutations in a DNAseq study. As I understand it one of the strengths of kallisto is that it can use long but fewer k-mers while relying on a perfect match for individual k-mers. Will this be a

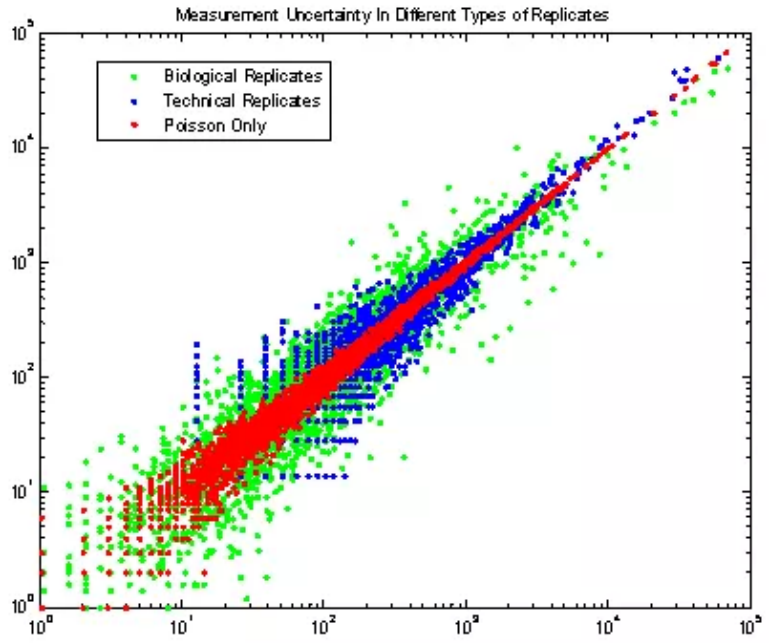
problem for me if my reference transcribtome does not contain the (previously identified) mutations?

Reply

August 18, 2015 at 1:31 pm
MB

This seems like nice work but I am not following what you are doing here in (1).

What are you defining as technical variance? I would define it at sampling noise from sampling the RNA fragments + measurement imprecision introduced by the sequencing process (PCR and sequencing junk). e.g.



which has to be overdispersed from Poisson because it's sampling noise plus a varying degree of slop.

The original Marioni et al dataset people reference had a tiny bit of slop because (I believe) they only did one library prep but different sequencing runs but even that it is still slightly over-dispersed from Poisson. Technical replicates from, e.g. low input libraries, will have a lot more slop.

Perhaps this is a naive question, but aren't you using a single library here? I don't understand how you can be estimating technical variance without using technical replicates (to quantify the slop).

Thank you!

Reply

August 18, 2015 at 3:28 pm
MB

I think it figured it out.

The random sampling of reads gives you counting noise. The boostrapping gives you an estimate of the computational

slop introduced by the mapping and stuff. The noise from the sampling is approximately poisson and mimics the sampling of the RNA in a real library.

If you did an ecoli library where everything maps uniquely and there is not too much informatics noise that would be about possion.

You end up quantifying the part of the technical variance that is informatics and sampling noise. There's extra variance from the library prep and sample handling, but it doesn't matter because you usually need biological reps anyway. Uncorrelated variances just add up and biological reps have all the slop.

Reply

August 19, 2015 at 4:18 am
Lior Pachter

Exactly.

April 18, 2016 at 11:43 am
alyamahmoud

Could this approach of 'not depending on exact alignment but rather which transcript the read originated from' be used for quantifying transcripts from closely related co-growing/co-cultured species ?

Reply

April 18, 2016 at 12:41 pm
Jason Merkin

They posted a subsequent preprint where they use it for strain-level metagenomics quantification, so I don't see why not.

Jason Merk

Reply

August 21, 2016 at 7:09 pm
shanrongzhao

I have questions on Supplementary Figure 5 (and 7 and 8). My real concern is bootstrap step — re-sampling with replacement. "With replacement" seems inappropriate, and that's why correlation in Figure S8 is worse than Figure S7. Thank you!

Reply

August 25, 2016 at 3:47 pm
Nicolas Bray

Hi Shanrong,

The use of sampling with replacement is standard when doing bootstrap analyses in general, but I'd be interested to hear your thoughts about any potential implications for this specific situation if you'd like to go into more detail.

Best,
Nick

Reply

October 19, 2016 at 12:22 pm
Krishna

Do the estimated read counts produced by Kallisto (and similar tools such as sailfish/salmon) work well with count-based DE tools tools such as DEseq2? With reference to Lior's talk at CSHL 2013 for instance, there are major problems using 'raw' counts from stuff like Htseq to do gene level DE analysis, in large part due to ambiguously mapped reads as well as various potential sources of bias which Cufflinks/RSEM/Kallisto can attempt to correct for when estimating abundances.

Do the estimated counts from Kallisto allow us to get around these issues? Presumably the summing up of transcript counts to gene level counts (which the DEseq authors recommend as part of a pipleline to use sailfish with DEseq2) would still be somewhat iffy for doing gene-level DE?

I am of course giving sleuth a try…

Thanks!

Reply

October 19, 2016 at 2:03 pm
Lior Pachter

> Do the estimated counts from Kallisto allow us to get around these issues?
Yes.

> Presumably the summing up of transcript counts to gene level counts (which the DEseq authors recommend as part of a pipleline to use sailfish with DEseq2) would still be somewhat iffy for doing gene-level DE?

The tximport program by Sonesson *et al.* does exactly that, and is much better as a method for obtaining gene counts than Htseq. However when it is used for DE with tools such as DESeq2 there is a new problem, namely that the uncertainty in the estimated counts is not being taken into account (this is not a problem with Htseq which involves counting reads directly without an intermediate inference step).

Reply

October 20, 2016 at 3:27 pm
Krishna

Thanks Lior.

Sleuth looks awesome so far.

In terms of reporting magnitudes of fold changes (Analogous to DEseq2 and Edger), am I correct in saying that we would report the 'b' (beta) values from

the Sleuth results tables, which indicate effect size?

October 21, 2016 at 6:40 am
haroldpimentel

Hi Krishna,

You are correct, ‘b’ indicates effect sizes on the log-transformed scale.

October 21, 2016 at 4:50 pm
Krishna

Thanks Harold.

What would you recommend as a sufficient level of bootstrapping with Kallisto to then run Sleuth?

I noticed you used n=100 for the sample data provided for testing out Sleuth and was wondering if you have a feel for what might be an ‘optimal’ level of bootstrapping, at which the DE tests you’ve run so far indicate some form of asymptote between the level of bootstrapping and improvements in statistical power of the analysis.

I suppose this would vary between experiments but wondered if there was some ballpark estimate for this behavior as increasing –bootstrap-samples does seem to slow Kallisto down. The EM step for each bootstrap took about 25-30s on my pc, so with >100 bootstraps, this really adds up.

Thanks again for your help.

Reply

LEAVE A REPLY

Enter your comment here...