Briefings in Bioinformatics, 2022, **23(2)**, 1–30

https://doi.org/10.1093/bib/bbab563

Review

A simple guide to *de novo* transcriptome assembly and annotation

Venket Raghavan[†], Louis Kraft[†], Fantin Mesny[‡] and Linda Rigerte[‡]

Corresponding authors: Venket Raghavan, Quantitative and Computational Biology, Max Planck Institute for Biophysical Chemistry, 37077 Göttingen, Germany. E-mail: vraghav@mpibpc.mpg.de; Louis Kraft, Quantitative and Computational Biology, Max Planck Institute for Biophysical Chemistry, 37077 Göttingen, Germany. E-mail: louis.kraft@mpibpc.mpg.de

[†]These authors are joint first coauthors.

[‡]These authors are joint second coauthors.

Abstract

A transcriptome constructed from short-read RNA sequencing (RNA-seq) is an easily attainable proxy catalog of protein-coding genes when genome assembly is unnecessary, expensive or difficult. In the absence of a sequenced genome to guide the reconstruction process, the transcriptome must be assembled *de novo* using only the information available in the RNA-seq reads. Subsequently, the sequences must be annotated in order to identify sequence-intrinsic and evolutionary features in them (for example, protein-coding regions). Although straightforward at first glance, *de novo* transcriptome assembly and annotation can quickly prove to be challenging undertakings. In addition to familiarizing themselves with the conceptual and technical intricacies of the tasks at hand and the numerous pre- and post-processing steps involved, those interested must also grapple with an overwhelmingly large choice of tools. The lack of standardized workflows, fast pace of development of new tools and techniques and paucity of authoritative literature have served to exacerbate the difficulty of the task even further. Here, we present a comprehensive overview of *de novo* transcriptome assembly and annotation. We discuss the procedures involved, including pre- and post-processing steps, and present a compendium of corresponding tools.

Keywords: de novo, transcriptome, assembly, annotation, tools, RNA-seq

Introduction

Ribonucleic acids (RNAs) are an important class of biomolecules in cells and organisms. They represent the output of the genome being transcribed or expressed—the *transcriptome*. Numerous types of RNA exist, with each playing an important role in gene expression, and ultimately, in linking the genotype to the phenotype [1]. Ribosomal RNAs (rRNAs) and transfer RNAs (tRNAs), for instance, constitute the translation machinery that synthesizes proteins. The latter along with non-coding RNA (ncRNA) species also exert regulatory control over important biological processes [2, 3] including gene expression itself [4]. Messenger RNAs (mRNAs) constitute an important class of RNA. The sequences of mRNAs encode information that is used by the ribosomal machinery to synthesize proteins (translation). Hitherto non-mRNA species have been considered ‘non-coding’, assuming that they cannot be translated. However, this has been challenged by recent evidence indicating that regulatory long non-coding RNAs (lncRNAs) can in fact code for short peptides [5], underscoring the need for improving our understanding of these important molecules.

With the advent of affordable next-generation sequencing (NGS) platforms [6], high-throughput profiling of RNA using sequencing (RNA-seq) [7, 8] has become the preferred method of interrogating transcriptomes [7, 9]. RNA-seq can be used for a variety of purposes [7, 10]. The most popular use cases are establishing a catalog of an organism’s genes and proteins (transcriptome functional annotation) and studying changes in gene expression (differential expression analysis). ‘RNA-seq’ commonly refers to the so-called ‘bulk’ RNA-seq approach wherein material from a population of cells are pooled together for sequencing (e.g. all cells in a protozoan organism) as opposed to the increasingly popular single-cell RNA-seq (scRNA-seq) approach [11] wherein RNAs are isolated individually from single cells. We focus on the bulk RNA-seq approach in this paper.

Most RNA-seq studies today rely on short-read sequencing [7, 12, 13]. Here, the RNA molecules are isolated and enriched (usually for mRNA [7]), and reverse transcribed into complementary DNA (cDNA). The cDNA sequences are fragmented, randomly primed and amplified using PCR to yield an RNA-seq cDNA library which is then processed by the sequencing instrument

The authors are comprised of students at various stages of academia with an interest in various aspects of applied bioinformatics.
Fantin Mesny is a PhD candidate studying plant–fungal interactions.
Louis Kraft is a masters student studying *de novo* transcriptome assembly.
Venket Raghavan and **Linda Rigerte** are scientific research assistants working on RNA-seq and plant–fungal interactions, respectively.

Received: September 9, 2021. **Revised:** December 3, 2021. **Accepted:** December 9, 2021

[12, 14]. The sequencing output is in the form of millions of 'short' reads, which are sequences over an alphabet denoting a series of nucleotides (e.g. `GATTACA`). Such short-read sequences may be anywhere between 50 and 250 bp (base pairs) long; the library used for sequencing is often 'sized' (i.e. filtered) to retain only fragments of a certain length (e.g. 350 bp). The short reads must then be assembled into the sequences they originated from. This is the computationally challenging task of `transcriptome assembly` [15].

The sequences can be assembled either reference-guided or *de novo* [15]. The reference-guided approach requires the genome of the organism or a closely related species as an input. The reads can then be mapped to this 'reference' genome to determine which genes the reads originated from, and subsequently reconstruct the corresponding transcripts [15]. ***De novo* transcriptome assembly**, in contrast, is 'reference-free'. The process is *de novo* (Latin for 'from the beginning') as there is no external information available to guide the reconstruction process. It must be accomplished using the information contained in the reads alone. This approach is useful when a genome is unavailable, or when a reference-guided assembly is undesirable. For instance, in opposition to a *de novo* assembler successfully producing a transcript, a reference-guided approach might not be able to reconstruct it correctly if it were to correspond to a region on the reference containing sequencing or assembly gaps [15, 16]. *De novo* assembly is discussed in detail in Section *De novo* transcriptome assembly. However, *de novo* assembled sequences are uninformative on their own. They must be assigned human-readable identifiers and have their functional and evolutionary properties characterized in order to have their biological relevance elucidated. This is the process of **transcriptome annotation**. As the objective of the procedure is to elucidate the functions of the sequences, it is also often referred to as 'functional' annotation.

Short-read RNA-seq is affordable, easily accessible and has low error rates. And importantly, it has a large community of established practitioners, literature, tools and other resources. As a result, the popularity of the approach continues to proliferate across the biological sciences. It has become especially popular for studying non-model organisms (for example, in the ecological sciences [17]), as a *de novo* transcriptome is an acceptable substitute for an absent genome. For example, it has been used to study zooplankton [18], bats [19], fruits [20] and pathogens [21]. There is now also considerable interest in 'in-housing' the *in silico* assembly and annotation workflows as the required computational resources have become easily accessible [22, 23]. It is now possible to sequence, assemble *de novo* and annotate a transcriptome within the confines of one's own laboratory. However, the path to an annotated, *de novo* assembled transcriptome can be challenging. Those interested must not only acquaint themselves with the procedures involved, but also select the right set of tools for this purpose. These issues are non-trivial, and can become overwhelming. RNA-seq literature reveals many variations on the same theme, with a variety of tools and combinations of processing steps having been used. Furthermore, RNA-seq is a computationally intensive task. Becoming acquainted with the computational resources necessary can also be a hurdle.

Here, we present a step-by-step overview of the *de novo* transcriptome assembly and annotation workflow (Figure 1). In brief, the RNA-seq data must first be quality controlled (Figure 1 panel (A), Section 'Pre-assembly quality control and filtering'). For instance, this can include excluding reads originating from rRNAs, and removing adapter sequences. Subsequently, the data can be assembled *de novo* to obtain the transcriptome, whereafter they must be quality controlled once again in order to produce a final assembly free of assembly artifacts (Figure 1 panel (B), Sections '*De novo* transcriptome assembly', 'Post-assembly quality control', 'Alignment and abundance estimation' and 'Assembly thinning and redundancy reduction'). Read alignment and transcript abundance estimation (Figure 1 panel (C), Section 'Alignment and abundance estimation') are performed both as quality control measures, and to estimate gene/transcript expression levels for differential expression analysis (Figure 1 panel (D), Section 'Differential expression analysis'). If the RNA-seq data are suspected to contain non-mRNA species, RNA classification can be carried out to classify and filter the data (Figure 1 panel (E), Section 'RNA classification'). Protein sequences are useful in many contexts (including annotation), and therefore, the transcriptomic sequences can be translated into their amino acid counterparts (Figure 1 panel (E), Section 'Sequence translation'). Finally, the nucleotide (and/or translated protein) sequences can be annotated to assign human-readable identifiers to them, and elucidate their biological roles (Figure 1 panel (F), Section 'Transcriptome functional annotation').

In the subsequent sections, alongside a brief conceptual introduction of each procedure, we present a compendium of the relevant state-of-the-art-tools. As transcriptome annotation is not well-addressed in literature, we have discussed this procedure in detail. Transcriptome annotation involves a myriad of processes which we present and discuss as independent, compartmentalized steps. We also discuss a number of transcriptome annotation pipelines that automate the entire procedure (Section 'Transcriptome annotation suites'). The need may arise to compare multiple transcriptomes, for instance to infer conserved orthologs [24]. We have discussed comparison of transcriptomes and relevant tools in the Section 'Comparing transcriptome assemblies'. *De novo* assembly and annotation workflows continue to grow in complexity, both in terms of the number of tools used and samples processed. Therefore, automated workflows are needed to make the procedures tractable, scalable and reproducible. To this end, we have devoted an entire section to the important topic of bioinformatic workflow managers which can be used to construct and orchestrate such workflows (Section 'Workflow

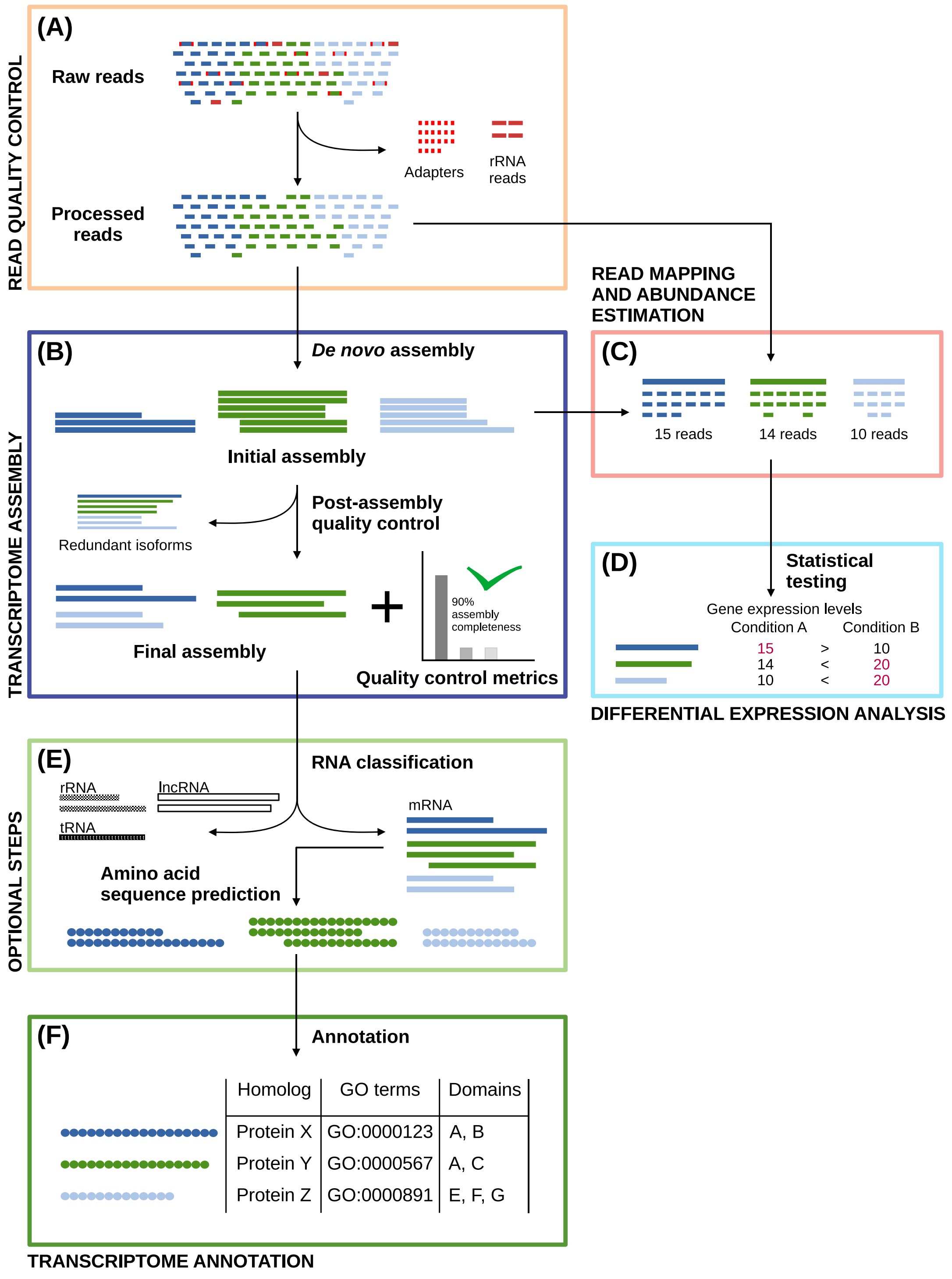

Figure 1. Assembly and annotation workflow. **(A)** Quality control of the raw reads by filtering for erroneous reads and sequencing artifacts. **(B)** Sequence assembly including clustering into groups of isoforms and removing redundant sequences (isoforms are transcript variants arising from alternative splicing). **(C)** Mapping the raw reads to the assembled sequences for either quality control of the assembly or for differential expression analysis. **(D)** Applying statistical tests for identification of changes in expression levels. **(E)** Classifying sequences by RNA species and translating into protein sequences before annotation. **(F)** Annotating sequences on the basis of sequence similarity, identifying sequence features (such as functional domains) and annotating Gene Ontology terms.

managers'). For the interested newcomer to the field, we briefly summarize some of the computational prerequisites to be aware of in Section 'Computational and programmatic considerations'. Finally, it can potentially be unclear as to what one should annotate in a *de novo* transcriptome, and where these annotations can be published. We address these issues in the final section of this document (Section 'What to annotate and where to publish'). As we name and discuss well over 100 different tools in this paper, we have also supplied a spreadsheet summarizing these as a supplement (Table S2). Our hope is that this publication can serve as a primer to the topic,

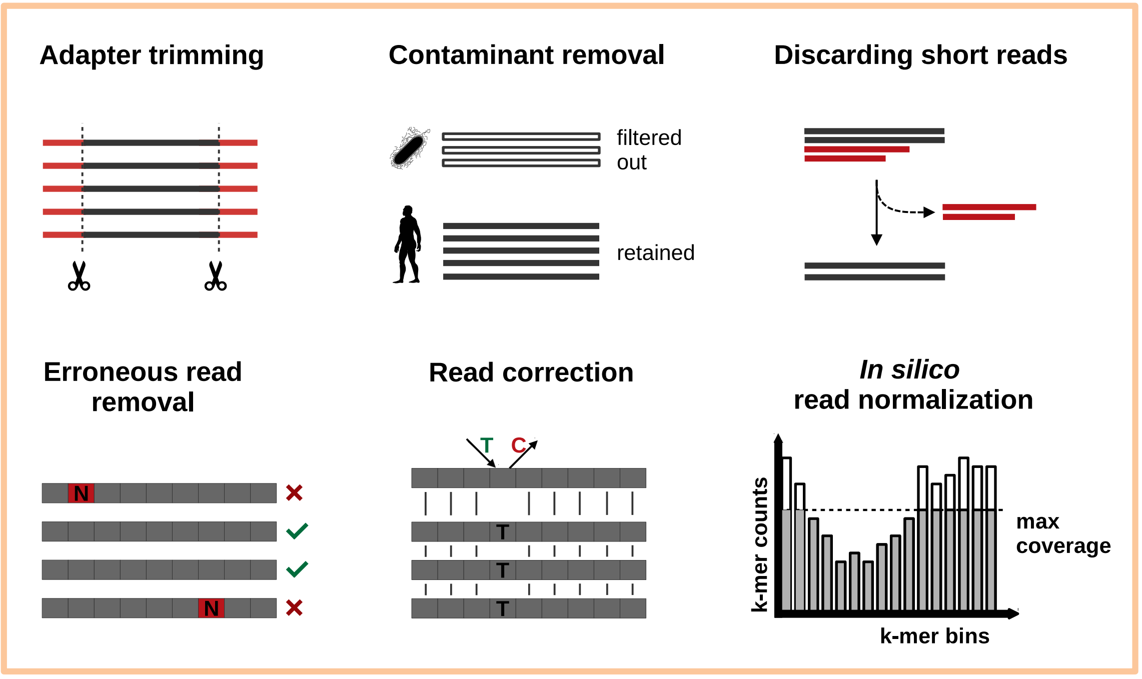

Figure 2. Short-read quality control and data cleansing involve procedures such as adapter trimming, removing short reads and erroneous reads containing N-bases, read correction by comparison to other reads, and excluding reads originating from contaminant sources (e.g. pathogens in a host species). *In silico* read normalization can be a useful pre-processing step for very large data sets (>200M reads) where it can significantly improve assembler performance by selectively reducing the reads in a manner such that the transcriptomic complexity of the original data set is retained.

and as a 'directory' of procedures, tools and literature that users can consult and use in pursuit of the perfect *de novo* assembled transcriptome.

Pre-assembly quality control and filtering

The reads generated by the sequencer constitute the data underpinning the assembly. While modern sequencers have low error rates, the data they produce are not error-free [25]. Properties of the reads including their abundance, read length, stranded-ness, paired-ness, overall GC content, `k-mer` composition and embedded errors directly affect the quality of the assembly, and by extension all subsequent procedures [26]. Therefore, the first step in *de novo* transcriptome assembly involves quality controlling the raw read data (Figure 2 highlights some such procedures). Quality control here implies both inspection of the data, and subsequent correction or filtering if considered necessary.

The short-read sequence inspection tool `FastQC` can be deployed as the first step of the pre-assembly quality control process. The tool provides a summarized overview of read quality metrics such as per-base PHRED quality scores, average incidence of 'N' (i.e. undefined) bases, GC content, read length distributions, identities of overrepresented sequences and presence of adapter sequences. A brief perusal of the report should indicate the measures that need to be taken. For instance, adapter sequences present in the reads may have to be removed, and the reads may perhaps have to be screened for contamination from non-target species. A recent alternative to `FastQC` is `Falco` [27], which can perform many of the same functions as `FastQC`. If multiple read data sets are being handled together, the bioinformatics report aggregator `MultiQC` [28] can be used to simultaneously inspect reports from not only `FastQC` but also numerous other tools (see https://multiqc.info/#supported-tools).

Subsequently, several measures can be applied to either correct or exclude aberrant reads. The first such procedure that can be applied is `k-mer` based read error correction using the tool `Rcorrector` [29]. This can be used to fix random errors generated during sequencing. However, such errors can be indistinguishable from single nucleotide polymorphisms (SNPs), and can lead to sequence variants being lost from the assembly.

If quality control metrics indicate the presence of adapter sequences in the data, these should be removed prior to assembly. Although adapter removal may have been performed by the sequencing facility, it is a good practice to scan for and eliminate residual adapters all the same. If only adapter trimming is desired, the dedicated trimming software `cutadapt` [30] is a good option as it is capable of error-tolerant adapter detection. The tool `TrimGalore` is a wrapper built around `cutadapt` and `FastQC` featuring some added functionality such as length-based sequence filtering which can be useful for discarding extremely short reads resulting from adapter removal. Finally, `BBDuk` from the `BBTools` [31] suite can also be used for the purpose of adapter removal. All three tools accept user-defined adapter sequences. `BBDuk` includes a set of common adapters and contaminants such as vectors. Therefore, explicit user input is not required in most cases.

It is often insufficient to perform only adapter removal. For instance, sequencing data often include reads containing ambiguous base calls (identified via the character

N in the sequence). Retaining such sequences only serves to confound the assembly and downstream analyses, as the exact nucleotide at that position in the read cannot be ascertained. Similarly, the data may also be filtered to retain only those reads (or portions thereof) containing bases with a certain minimum quality (Q) score. These quality scores [32] encode the probability of that particular base-call being wrong; for instance, a base with a Q value of 30 has a 0.001% chance of being erroneous. Reads carrying some maximum number of low-quality base calls can either be discarded entirely, or trimmed if the bases occur on the flanks. Likewise, it may be beneficial to discard reads that are extremely short (e.g. ∼30 nt). Although these steps can be performed by user-written scripts, it is more efficient to carry them out using purpose-built tools. One such all-in-one tool for NGS read quality control is `fastp` [33]. It can perform a wide variety of read quality control procedures including (but not limited to) automated adapter detection and removal, `N`-containing read removal, low-quality base filtering, overlap-based read correction (with paired-end reads), paired-end read merging and poly-X read trimming. An equivalent alternative is the tool `Trimmomatic` [34] which shares many of its features.

Once basic cleaning has been performed, the data can be assessed for the presence of contaminants. These are typically reads that do not originate from the organism and/or RNA species of interest. Contaminants can be broadly classified into two categories: foreign sequences and cognate contaminants. As the name suggests, foreign contaminants are reads belonging to off-target species (for instance, reads originating from an endosymbiont bacterium in an eukaryote organism of interest). Foreign contaminants can be detected—and optionally removed—using a short-read taxonomic classifier. `kraken2` [35] is a fast short-read taxonomic classifier intended for metagenomic analysis. In the RNA-seq context, it can be used to classify and remove all reads not originating from the taxon of interest. For instance, with a eukaryotic read dataset, `kraken2` could be used to exclude reads classified as bacterial, archaeal, fungal or from plants. `kraken2` offers ready-made reference sequence databases for classification; these can be found at https://benlangmead.github.io/aws-indexes/k2. An alternative to `kraken2` is `Centrifuge` [36] which can perform the same classifications, but with a smaller memory footprint. `FastQ Screen` is a screen-only alternative that can detect—but not remove—contaminants based on a user-supplied database.

In contrast, cognate contaminants are reads originating from off-target RNA species. For instance, although most RNA-seq methods select for mRNA sequences, it is still possible for off-target species to get represented in the data set in sizable quantities. This is especially true for rRNA sequences[37–39]. Reads originating from rRNAs are best detected and removed using `SortMeRNA` [40]. This tool was originally designed to filter out rRNA reads from metatranscriptomic data, but it can also be used with RNA-seq data. The tool maps inputs against custom rRNA databases (derived from `Rfam` [41] and `SILVA` [42]) to classify them as rRNA or non-rRNA reads. This is useful for enriching the data for reads from coding sequences prior to assembly. Other cognate contaminants such as long non-coding RNAs (lncRNAs) are best detected and dealt with post-assembly. This has been discussed further in the Section 'RNA classification'.

Modern RNA-seq studies now routinely sequence hundreds of millions of reads with the objective of reconstructing all expressed transcripts to full length to construct so-called 'reference' transcriptomes. Although this enhances sensitivity for recovery of lowly expressed transcripts [43, 44], it also has the side effect of producing a large number of reads for transcripts that are already well represented with significantly fewer total reads. Such an overabundance of reads (for well-represented transcripts) can quickly lead to unacceptable assembler performance and very long runtimes. This typically appears to occur at read depths exceeding 200 million reads [45]. In such situations, performing *in silico* normalization on the reads prior to assembly can significantly alleviate the aforementioned performance issues. Here, reads are quantified on the basis of their `k-mer` abundances, and are either retained or rejected based on user-defined thresholds [45]. The outcome is a strong reduction of the read volume in such a manner that full length reconstruction of a large majority of the transcript cohort can be achieved despite fewer reads being input to the assembler [45, 46]. Some tools that can perform *in silico* read normalization include `khmer` (using the `diginorm` algorithm) [47], `Bignorm` [48], `NeatFreq` [49] and `ORNA` [50]. The `Trinity` [46] assembler also offers in-built *in silico* normalization [45, 46].

Links:

`BBTools` - https://sourceforge.net/projects/bbmap/, https://jgi.doe.gov/data-and-tools/bbtools/

`Bignorm` - https://git.informatik.uni-kiel.de/axw/Bignorm

`Centrifuge` - https://github.com/DaehwanKimLab/centrifuge

`cutadapt` - https://github.com/marcelm/cutadapt

`Falco` - https://github.com/smithlabcode/falco

`fastp` - https://github.com/OpenGene/fastp

`FastQC` - https://www.bioinformatics.babraham.ac.uk/projects/fastqc/

`FastQ Screen` - https://www.bioinformatics.babraham.ac.uk/projects/fastq_screen/

`khmer` - https://github.com/dib-lab/khmer

`Kraken2` - https://github.com/DerrickWood/kraken2

`MultiQC` - https://multiqc.info

`NeatFreq` - https://github.com/bioh4x/NeatFreq

`ORNA` - https://github.com/SchulzLab/ORNA

`rCorrector` - https://github.com/mourisl/Rcorrector

`SortMeRNA` - https://github.com/biocore/sortmerna

`TrimGalore` - https://github.com/FelixKrueger/TrimGalore, https://www.bioinformatics.babraham.ac.uk/projects/trim_galore/

`Trimmomatic` - https://github.com/usadellab/Trimmomatic

De novo transcriptome assembly

RNA-seq reads contain a mixture of fragments corresponding to different parts of different transcripts. Transcriptional noise [51, 52], sequencing artifacts [53] and transcript isoforms originating from alternative splicing [54, 55] are also represented in these data. The objective of assembly is to accurately disambiguate the origin of the reads and reconstruct an accurate representation of the parent sequences. This is typically achieved by examining overlaps between reads (or subsequences thereof) in order to concatenate them into longer contiguous sequences (`contigs`) [15, 56].

Assembly algorithms are mostly based on using `k-mers` as assembly units instead of whole reads. A *k-mer* is a sub-string of length `k` derived from a particular read [46]. The first step in the assembly process is to construct a dictionary of all possible `k-mers` (for a given `k`) and the reads these `k-mers` originate from. Most modern assemblers are graph-based in that they represent the *k-mers* as nodes in a so-called De Bruijn graph (Figure 3). Subsequently a contig is a path through the graph, where each distinct `k-mer` represents a vertex in the graph. Edges are formed between two `k-mer` vertices if they have an overlap of exactly k-1 nucleotides. In this way paths through the graph correspond to possible sequences the `k-mers` originated from (Figure 3). Paths are extended until no further overlap-based extensions are possible [46]. Then each possible path through the graph is traversed and recovered as a separate contig corresponding to a single transcript.

De novo assembly does not produce a single, large De Bruijn graph like in a genome assembly but instead many disconnected subgraphs that, when disentangled, correspond to groups of related sequences (transcript isoforms or very closely related paralogs). Subsequently, post-processing steps may be implemented to filter and group contigs to yield a representative set of the assembled sequences. De Bruijn graph-based assembly is sensitive to the choice of `k-mer` length as it dictates the set of contigs assembled by controlling the complexity of the graphs. Generally speaking, shorter `k-mer` lengths imply a higher chance of error-free `k-1` overlap between any two `k-mers`. As such, shorter `k-mer` lengths contribute to the recovery of lowly expressed transcripts, but also lead to a larger number of false positive (incorrect/non-existent) contigs being assembled by connecting `k-mers` from unrelated reads [56, 57]. On the other hand, choosing a longer `k-mer` length would reduce the total number of contigs assembled, but also suppress the recovery of lowly expressed transcripts as fewer reads would be able to satisfy the `k-1` overlap requirement in an error-free manner. Therefore, the choice of the `k-mer` length defines a major trade-off in the assembly process [56]. Assembly tools usually supply a default value (or range thereof) for *k* which can be modified by the user.

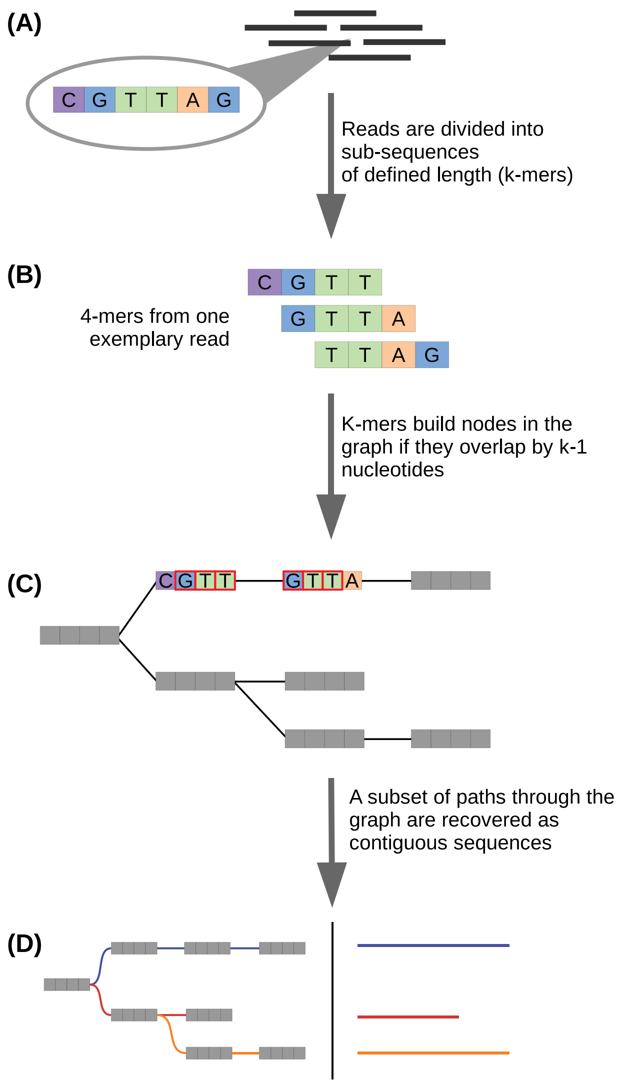
(A)
C G T T A G
Reads are divided into sub-sequences of defined length (k-mers)
(B)
4-mers from one exemplary read
C G T T
G T T A
T T A G
K-mers build nodes in the graph if they overlap by k-1 nucleotides
(C)
C G T T
G T T A
A subset of paths through the graph are recovered as contiguous sequences
(D)

Figure 3. A typical graph-based approach to *de novo* transcriptome assembly. The basic idea is to establish a catalog of sub-strings from the RNA-seq reads, and compose these into a graph (or set of graphs) wherein the sub-strings are connected if an overlap between them exists. This establishes paths through the graph(s) which correspond to the transcripts the reads (potentially) originated from. **(A)** Short nucleotide reads 50–250 nt in length are the inputs for the assembly process. If paired-end reads are supplied, the respective mates are merged into a single contiguous read prior to assembly. Highlighted here is a 6 nt portion of a single read (`CGTTAG`). **(B)** For each read, all possible sub-sequences of length `k` are generated (`k-mers`). The `4-mers` (`k = 4`) originating from the 6 nt nucleotide fragment from the previous step are indicated here as examples. **(C)** Subsequently, each `k-mer` becomes a node (also called vertex) in the graph, and an edge is established between any two nodes that share a `k-1` nucleotide overlap with each other. Edges are established between any two nodes that satisfy this overlap requirement. As a simplified example, an edge connecting the first and second `4-mers` from the previous step is highlighted here as existing as a part of a De Brujin graph. **(D)** Finally, different paths through the graph(s) are traversed and recovered as independent sequences. Not all paths through the graph are recovered; the subset of paths that represent valid transcripts is determined algorithmically.

A large number of tools are available for *de novo* assembly, and choosing one is a critical step in the workflow.

The most prominent De Bruijn graph-based assembler is `Trinity` [45, 46]. Since its release in 2011, the corresponding publication [46] has been cited over 10 000 times. It is robust and easy to use with an extensive set of associated tools, and a large user community. Most importantly, its general performance is consistently high and on par with other novel assemblers [56, 58], making it a trustworthy choice for assembly. A salient feature of `Trinity` is that it identifies sets of contigs that may be biologically related to one another (e.g. splice variants [59, 60]), and designates these as gene isoforms. This feature is especially useful for differential gene expression analysis with *de novo* assembled data, where it is common practice to aggregate the expression of related transcript isoforms into that of a representative 'gene', as this is considered to be robust [61, 62]. There are numerous other equally capable *de novo* assemblers [58]. A non-exhaustive list includes `SOAPdenovo-Trans` [63], `Oases` [64], `Trans-ABySS` [65], `IDBA-Tran` [66], `inGAP-CDG` [66], `RNA-Bloom` [67] and `rnaSPAdes` [56]. All of these tools except for `SOAPdenovo-Trans` apply a multiple `k-mer` strategy, aiming to make use of the advantages of small and large k-mer lengths to maximize transcript recovery. `RNA-Bloom` is actually specialized toward assembling single-cell RNA-seq but can also assemble bulk RNA-seq. In terms of performance and assembly output, `rnaSPAdes` and `Trans-ABySS` are similar to `Trinity` [58].

Splicing graph assemblers are a variant of De Bruijn graph assemblers. In this approach, each vertex corresponds to an exon, while the edges represent splice junctions [68, 69]. The paths through the graphs correspond to transcript isoforms. The graphs generated are less entangled in comparison to a traditional De Bruijn graph [70]. Representatives assemblers from this category include `Bridger` [71], `BinPacker` [57], `TransLig` [72], `DTA-SiST` [68] and `IsoTree` [70].

Choosing an assembler can be a difficult task. *Holzer et al.* [58] recently concluded in a broad evaluation of common transcriptome assemblers—using a variety of data sets from different species—that assembler performance is very dependent on the data supplied to it. As a consequence, they were unable to declare an unanimous 'best' assembler. We concur, and recommend comparing at least two different assemblers and multiple k-mer lengths.

Links:

`BinPacker` - https://github.com/macmanes-lab/BINPACKER

`Bridger` - https://github.com/fmaguire/Bridger_Assembler

`inGAP-CDG` - https://sourceforge.net/projects/ingap-cdg/

`DTA-SiST` - https://github.com/jzbio/DTA-SiST

`IDBA-tran` - https://github.com/loneknightpy/idba

`IsoTree` - https://github.com/david-cortes/isotree

`Oases` - https://github.com/dzerbino/oases

`RNA-Bloom` - https://github.com/bcgsc/RNA-Bloom

`rnaSPAdes` - https://github.com/ablab/spades

`SOAPdenovo-Trans` - https://github.com/aquaskyline/SOAPdenovo-Trans

`Trans-ABySS` - https://github.com/bcgsc/transabyss

`TransLig` - https://sourceforge.net/projects/transcriptomeassembly/

`Trinity` - https://github.com/trinityrnaseq/trinityrnaseq

Post-assembly quality control

Due to the noisy nature of RNA-seq data, *de novo* assemblies can contain intronic sequences and other 'transcriptional' byproducts. Further, the assembly process itself is not error-free [61]. For instance, unrelated but highly similar transcripts may be incorrectly fused together into a single contig during the assembly process (i.e. a chimera [73]). Or the choice of `k-mer` length might have been inappropriate, leading to a highly fragmented assembly wherein multiple contigs together would yield a longer, complete sequence (that might have been otherwise assembled with a different choice of `k-mer` length). Therefore, assessing the quality of a *de novo* transcriptome assembly is a crucial step before annotation and other downstream procedures. A low-quality assembly can lead to erroneous interpretations in a variety of scenarios including gene identification and differential expression analysis.

The quality of an assembly can be assessed from several perspectives. First is sequence length and fragmentation. An assembly with many short contigs can be considered fragmented. It is possible that this is the result of improper assembly or poor sequencing. Tools such as `SeqKit` [74] can be used to calculate sequence length statistics (such as the `N50` value) that are helpful in this regard. Second is read support—the fraction of all reads that map back to the assembly. A good quality assembly will have made use of most of the reads that went into it. Further, the proportion of reads that map to multiple sequences would be low (but this cannot be guaranteed, as a gene may genuinely have many transcript isoforms). All of these metrics can be checked easily by aligning the reads against the assembled sequences. Read support and alignment estimation tools are discussed in Section 'Alignment and abundance estimation'. The `GitHub Wiki` of the `Trinity` *de novo* assembler https://github.com/trinityrnaseq/trinityrnaseq/wiki lists several other methods to assess the quality of an assembly including interrogating the strand-specificity of the assembly in case of prior strand-specific sequencing, and calculating the `ExN50` statistic [58, 75].

At this juncture, we would like to take a moment to caution readers with regards to the application of the `N50` statistic to transcriptome assemblies. The `N50` is a simple metric which describes the sequence length at which half the nucleotides in the genome assembly are in sequences equal in or longer than this length [76]. The goal would then be to maximize the `N50` value as this would indicate complete assembly of all genomic elements (e.g. multiple chromosomes). This is inappropriate for transcriptome assemblies as the objective is recovery of many (relatively) short full-length sequences, and not the construction of a few very long contigs. Given the presence of transcript isoforms, short contigs resultant from transcripts with low coverage, and overly long contigs resultant from overzealous assembly of multiple isoforms, the N50 statistic can become heavily skewed, thereby presenting a biased overview of the assembly. The ExN50 metric is a modification to the traditional N50 making it suitable for assessing transcriptome assemblies. Here, the N50 value is calculated only for the top X% of the cumulative expression levels. The length reported as corresponding to ExN50 is a 'gene' length obtained as the expression-weighted sum of the corresponding isoform lengths. This approach neatly sidesteps the issues posed by the plurality of short, lowly expressed transcripts and long isoforms from highly expressed transcripts, as these are now gathered into representative genes. This metric is currently only implemented for the Trinity assembler.

An alternative approach to checking the quality of the assembly is to assess its composition. A good quality assembly would ideally have recovered a large fraction of the transcriptome that had been sequenced. The most popular method in this regard is to test the assembly for the presence of orthologs to certain genes that are universal, persistently expressed and occur almost exclusively as single copies in the genome. If a transcriptome has been properly sequenced and assembled, orthologs to a large majority of these should be found. This analysis can be performed using the tool BUSCO (Benchmarking Universal Single-Copy Orthologs) [77]. The tool maintains curated sets of universal single-copy genes from OrthoDB [78]. The 'completeness' of the assembly is assessed by how many of the universal genes have matches in the input data and whether these matches are duplicated, fragmented or full length. As a general rule a good quality assembly should have fairly high BUSCO completeness scores: > 80% BUSCO genes should have matches in the transcriptome, and very few matches should be missing or fragmented. If an assembly has a high proportion of missing and fragmented BUSCO genes, this is indicative of poor quality. In general, *de novo* transcriptome assemblies will have many duplicate matches to the BUSCO sequences. This is caused by the presence of closely related transcripts that represent splicing isoforms, and thus is not necessarily indicative of unwanted redundancy in the assembly. The issue of redundancy is discussed in detail in Section 'Assembly thinning and redundancy reduction'. As the tool was originally designed for genomic assemblies, BUSCO does not account for this phenomenon. An alternative to BUSCO is the domain-based quality assessment tool DOGMA [79]. In this case, instead of scoring on the basis of conserved genes, completeness is instead assessed on the basis of conserved protein domains.

In a similar vein, the assembly quality can also be checked on the basis of the provenance of the assembled sequences. Ideally, for a given organism, a vast majority of the assembled transcriptome sequences should map to its own sequences in an external database (e.g. from genomic sequencing; or those from closely related species). Concomitantly, sequences that do not map in this manner (or map to off-target organisms) can be considered contaminants and filtered out, yielding an improved assembly. But this may potentially discard novel, un-annotated sequences, so it must be done with caution. There are several popular sequence search/alignment tools and sequence databases that can be used for checking the provenance of the assembled sequences. These are discussed in Section 'Identity assignment via homology transfer'. If the sequencing reads have been processed prior to assembly (as discussed in Section Pre-assembly quality control and filtering), this quality control may not be as useful.

Several integrative tools have been developed over the years with an eye on assessing the quality of *de novo* transcriptome assemblies. These tools generally expand upon the basic read mapping metrics mentioned above and calculate additional statistics. They may also offer the option to run BUSCO and other tools internally, compare two or more versions of an assembly and compare the assembled sequences against a genome or a database of known sequences (as an example, see metrics indicated in thiswebsite). The most popular tool in this regard is TransRate [80] which incorporates many of the metrics mentioned above. The tool also checks for the presence of chimeric sequences, whose removal generally improves transcriptome assemblies [56, 81]. DETONATE [82] and rnaQUAST [83] are tools developed in the same vein as TransRate, but only rnaQUAST is still being developed (as of this publication). TransRate and DETONATE, however, appear to continue to be in use judging from recent citations (for instance, see supplementary materials from *Ceschin et al.* [84]). A recent development is the Bellerophon pipeline [85], which offers a comprehensive quality assessment and filtration tool that integrates several tools including TransRate, the clustering suite CD-HIT [86] and BUSCO. In addition to assessing quality, the tool also automatically applies measures (such as filtering out very lowly expressed transcripts) to improve the quality of the assembly. The only inputs required are the assembly and the reads.

Quality controlling a *de novo* assembly can require multiple rounds of assembly (for instance to test different k-mer lengths), which can quickly become a tedious undertaking. There are several tools that encapsulate pre-processing, assembly, quality control measures and even annotation together (often using bioinformatic workflow managers; see Section 'Workflow managers') to enable turnkey production of high-quality transcriptomes. Some of the popular tools in this regard include DRAP [87], EvidentialGene and the multi-assembler approach-based pipelines Oyster River Protocol [88], TransPi [89] and Pincho [90].

Links:

Bellerophon Pipeline - https://github.com/JesseKerkvliet/Bellerophon

BUSCO - https://busco.ezlab.org/

DETONATE - https://github.com/deweylab/detonate

DOGMA - https://domainworld-services.uni-muenster.de/dogma/ (web server), https://ebbgit.uni-muenster.de/domainWorld/DOGMA (source code)

DRAP - http://www.sigenae.org/drap/

EvidentialGene - http://arthropods.eugenes.org/EvidentialGene/

The Oyster River Protocol - https://oyster-river-protocol.readthedocs.io/en/latest/index.html

Pincho - https://github.com/RandyOrtiz/Pincho

rnaQUAST - https://github.com/ablab/rnaquast

TransRate - https://github.com/blahah/transrate and http://hibberdlab.com/transrate/

SeqKit - https://github.com/shenwei356/seqkit

TransPi - https://github.com/palmuc/TransPi

Trinity Wiki - https://github.com/trinityrnaseq/trinityrnaseq/wiki

Alignment and abundance estimation

Read alignment and transcript abundance estimation are typically used for differential expression analysis in the broader context of RNA-seq. Read mapping is a pre-requisite for abundance estimation [91]. However, alignment metrics can also be used to quality control the assembly. A ‘good quality’ *de novo* assembled transcriptome would have a large majority of the reads mapping/aligning to the assembly, i.e. most reads will have had been used in its construction. This is assessed as the ‘read support’ for the assembly. As a thumb rule, a good assembly would have $>$ 80% read support, and would have a low proportion of un-mapped reads. Reads can also map to more than one contig (multi-mapping reads). This can occur, for instance, when the assembly contains transcript isoforms that share exons. (Multi-mapping reads are discussed also in Section ‘Assembly thinning and redundancy reduction’.)

Abundance estimation, as the name implies, refers to the process of inferring the expression level of the transcripts in the assembly. Abundances are *estimated* because it is impossible to disambiguate the source of multi-mapping reads, and the true expression levels of transcripts are usually unknown. As such, most techniques typically produce maximum likelihood values for transcript abundances. These values which include read support (on a per-transcript basis) and a normalized expression metric such as transcript per million (TPM) [91]. These values are crucial for differential expression analysis (see Section ‘Differential expression analysis’), but can also be used for assembly quality control purposes. For instance, transcripts of questionable biological significance typically have low expression levels, and can be filtered out from the assembly based on their TPM metrics. TPM calculations can be easily performed using a dedicated tool such as TPMCalculator [92].

Read alignment and abundance estimation can usually be done together. There are two main approaches to the combined procedure. The first is a two-step process where the reads are first aligned to the assembled contigs using a general purpose aligner such as Bowtie2 [93] or STAR [94]. The output is typically a BAM file which lists the sequences and the reads aligned to them (*Li et al.* [95] and http://www.htslib.org). This is then fed to a tool such as RSEM [96] (RNA-seq by Expectation-Maximization) to obtain abundance estimates. The alternative is a single-step approach known as pseudoalignment. Read alignment is computationally expensive as every nucleotide from the reads and assembled contigs must be compared. Pseudoalignment eschews this in favor of establishing the association between reads and contigs on the basis of k-mer similarities between them. There are two popular pseudoalignment tools, namely Kallisto [97] and Salmon [98]. Both tools are based on very similar approaches.

We would recommend using one of the pseudoalignment tools as opposed to the alignment-estimation workflow due to their speed [99], comparably high accuracy [100–102] and ease of use. Additionally, these tools can also run under an alignment-based mode if necessary, making them the versatile choice.

Links:

Bowtie2 - https://github.com/BenLangmead/bowtie2

Kallisto - https://github.com/pachterlab/kallisto

RSEM - https://github.com/deweylab/RSEM

Salmon - https://github.com/COMBINE-lab/salmon

STAR - https://github.com/alexdobin/STAR

TPMCalculator - https://github.com/ncbi/TPMCalculator

Assembly thinning and redundancy reduction

De novo transcriptome assemblers typically produce many more sequences than would be expected based on number genes in the genome. For example, *Bryant et al.* [75] report having assembled over 1.5 million sequences for a transcriptome of the axolotl (*Ambystoma mexicanum*). The genome, in comparison, has ca. 23 000 genes [103]. The discrepancy between the number of genes and the number of transcripts assembled *de novo* boils down to the perception that transcription is a noisy, pervasive process. For instance, *Dunham et al.* [104] state that over 80% of the *Homo sapiens* genome gets transcribed even though less than 3% [105] of the transcribed products code for proteins. As such, the *de novo* assembled contigs include transcriptional artifacts, pre-mRNA and ncRNA in addition to the protein-coding transcripts [61]. Another source of extra sequences is alternative splicing [59, 60, 106] which manifests as transcript isoforms. It may not always be necessary to retain all such sequences. Assembly thinning can therefore be an important step toward obtaining a sequence set of a manageable size.

A straightforward approach to thinning is to manually select contigs that can be considered representative with respect to the entire assembly. This is infeasible unless the relationships between the assembled contigs is known *a priori*. Luckily, most popular assemblers classify the transcripts into groups of isoforms automatically. A representative isoform can be chosen in several different ways: the isoform with the highest read support, the longest isoform, or the isoform that produces the longest translated amino acid sequence, or even the isoform whose coding sequence (CDS) has the highest read support. All of these approaches may be equally effective, and are likely to be data set-dependent. It is recommended to choose a method based on the BUSCO scores and other quality metrics.

Should the gene-isoform relationship be unavailable, a simple approach to thinning would be to exclude transcripts that can be considered as being lowly expressed on the basis of abundance metrics such as TPM. These metrics can be calculated easily using one of the tools mentioned in the Section 'Alignment and abundance estimation'. Thereafter, contigs with read support below some threshold (e.g. TPM < 1.00) could be discarded from the assembly.

A more rigorous approach for assembly thinning is to use a clustering tool. This is especially useful in cases where the assembled contigs do not have the gene–isoform relationship disambiguated or the assembly is genuinely redundant (i.e. many contigs with nearly identical sequence have been assembled). The clustering tools CD-HIT [86, 107] and MMSeqs2 [108–111] use a combination of sequence identity and sequence coverage thresholds to group sequences together into clusters and extract representative sequences. The representatives are typically either the longest sequence in each cluster or the sequence with the most commonality with the cluster members. There are also several tools that have been developed specifically with *de novo* transcriptome assemblies in mind. Many of these tools work on the premise that shared read support—i.e. high proportions of multi-mapping reads within a set of reads corresponding to a set of transcripts—can be used to cluster the sequences together. Tools in this category include Corset [62], Grouper [112] and Compacta [113].

An interesting approach to assembly thinning is presented by the SuperTranscripts [114] tool. Instead of choosing a representative isoform for each gene cluster, the tool simply stitches all unique exons from the isoforms into a single, linear sequence. This 'transcript-hybrid' does not necessarily exist in a real biological context, but can nevertheless be useful. Super transcripts have great potential not only for analysis, e.g. for studying differential transcript usage, but also for assembly thinning without any sequence information loss. Assembly thinning is not the main objective but rather a side-effect.

It is important to note that assembly thinning should be performed only if absolutely necessary. The provenance of *de novo* assembled contigs are unknown, and they all therefore can carry significant biological information. Assembly thinning is an inherently heuristic task. It is entirely possible, for instance, to tune the parameters such that closely related paralogs get clustered together. In such an event, sequences that should be represented in the assembly will be lost. Further, in the case of isoforms, it is often impossible to identify a single best isoform [45]. For instance, the longest isoform is not necessarily the most expressed (and vice versa). It is not even necessary that the longest or the most expressed isoform is the one that is actually representative of the gene and the concomitant protein. The longest isoform may be the result of the assembler erroneously overextending the biologically relevant contig, or the result of an intron being retained in the transcript. Subsequently, the corresponding protein may not be the longest protein in the cohort, or may even be absent as a result of the corresponding ORF being aberrant. As such, extreme caution must be exercised when performing assembly thinning and redundancy reduction, as irreverent thinning can result in the loss of otherwise informative sequences from downstream analyses.

Links:

CD-HIT - http://weizhongli-lab.org/cd-hit/
Corset - https://github.com/Oshlack/Corset
Compacta - https://github.com/bioCompU/Compacta
Grouper - https://github.com/COMBINE-lab/grouper
MMseqs2 - https://github.com/soedinglab/MMseqs2
SuperTranscripts - https://github.com/Oshlack/Lace

Differential expression analysis

Assessing changes in gene expression in response to changes in physiological or environmental conditions is one of the main objectives of the RNA-seq approach. In the simplest case, this is achieved by capturing the RNA from independent samples (in replicate) exposed to experimental and control conditions. Thereafter, the sequenced reads can be mapped to the organism's genes to assess how differently the genes are expressed under the experimental circumstances as opposed to the control scenario. This is known as **differential expression (DE) analysis** [115]. Through such comparisons of expression, it is possible to obtain an understanding of the activity of genes under various circumstances.

In order to perform a DE analysis, a collection of gene sequences from the organism is required. The genes themselves can be used if an annotated genome is available. If no genome is available, a *de novo* assembled transcriptome can be used, with the transcripts acting as proxies for the genes. In a *de novo* transcriptome assembly for a DE analysis, the reads from all conditions and all replicates are pooled together for assembly: this produces a single, common 'reference' transcriptome against which the reads can then be mapped and quantified.

Subsequently, the data can be analyzed for indications of differential expression. The analytical procedure is the same irrespective of whether a genome or a transcriptome was used as the reference. In both cases, the result is a table wherein each row represents a unique sequence, and each column represents a unique sample and replicate. Each cell in this table indicates the number of reads assigned to that particular sequence in that particular sample-replicate. A statistical approach is adopted wherein the mean value of the read counts for each sequence over the sample replicates is compared between the conditions of interest.

There are a number of packages in various programming languages that are capable of performing DE analysis. Although the methods they implement differ [91], they all perform the following tasks: (1) normalizing the read counts to account for differences in sequencing depths between the samples [116], (2) noise reduction [117] (optional), (3) fitting a read counts distribution to the data, and using it to test differential expression of each gene between the conditions of interest and (4) correcting the produced *P*-values for multiple testing.

Whether or not a gene or transcript has been differentially expressed is indicated through a set of numerical values, of which two are of particular importance in the context of biological interpretation. The aforementioned corrected *P*-values indicate whether the difference in expression of a gene/transcript between two conditions is *statistically significant*. A small *P*-value indicates that the probability of the read counts being different between the two conditions purely due to chance is very low: i.e. it is highly probable that the source of the difference is a biological phenomenon. The `log2FoldChange` value describes the *magnitude* of the difference in expression: one of the two conditions is taken as the baseline and the change in expression in the other is calculated relative to this. As the name suggests, this is the $\log_2$ value of the ratio of the mean counts of the two conditions being compared. A positive `log2FoldChange` (`lfc`) value indicates upregulation, and a negative value indicates downregulation with respect to the condition being adopted as the basis for comparison. Lowly expressed genes/transcripts tend to have higher variability in their support, leading to the `lfc` being overestimated for these. Therefore, an important—but often overlooked—step is to correct the `lfc` estimates with a shrinkage algorithm (such as `apeglm` [118] or `ashr` [119]) before using them for biological interpretation. It is conventional to consider only those genes/transcripts that have a certain level of statistical significance and magnitude of difference in expression (e.g. *P*-value <= 0.05 and `log2FoldChange` $\notin \{-1, 1\}$) as being differentially expressed.

The most popular packages for DE analysis today have all been developed for use with the `R` [120] statistical programming language. The three popular packages in this regard are `DESeq2` [121], `edgeR` [122] and `limma` [123, 124]. Among the three packages, `DESeq2` appears to be the most conservative, detecting fewer differentially expressed genes in general in comparison to `edgeR` and `limma` [125]. A number of tools have also been developed to facilitate import/export of the requisite data into the `R` environment, and pre-process them for DE analysis. In this regard, we recommend the use of `tximport` [126] which is capable of preparing data from commonly used abundance estimators such as `RSEM`, `Kallisto` and `Salmon` for analysis with all three aforementioned DE packages. Other packages to facilitate DE analyses exist. For example, `RUVSeq` [127] can be used to correct for batch effects in the data, `SARTools` [128] can be used to obtain standardized DE analysis templates, `MetaCycle` [129] can be used to perform time-series RNA-seq analysis [130] and `consensusDE` [131] can be used to perform DE analysis employing a multi-algorithmic approach.

Finally, we like to point out that DE analysis has been covered in much detail elsewhere (e.g. *Conesa et al.* [91], *Van den Berge et al.* [132], *Schurch et al.* [133], *Finotello and Di Camillo*[134], *Li and Li*[135], *McDermaid et al.* [124]), and we defer to those publications for an in-depth discussion of the topic. In specific, *McDermaid et al.*[124] offer an excellent overview of DE analysis packages, and *Conesa et al.* [91] offer a comprehensive review plus recommendations for RNA-seq experiments with a focus on DE applications.

Links:

`apeglm` - https://bioconductor.org/packages/release/bioc/html/apeglm.html

`ashr` - https://github.com/stephens999/ashr, https://cran.r-project.org/web/packages/ashr/index.html

`consensusDE` - https://bioconductor.org/packages/release/bioc/html/consensusDE.html

`DESeq2` - https://bioconductor.org/packages/release/bioc/html/DESeq2.html

`edgeR` - https://bioconductor.org/packages/release/bioc/html/edgeR.html

`limma` - https://kasperdanielhansen.github.io/genbioconductor/html/limma.html https://bioconductor.org/packages/release/bioc/html/limma.html

`MetaCycle` - https://cran.r-project.org/web/packages/MetaCycle/index.html, https://github.com/gangwug/MetaCycle

`RUVSeq` - https://bioconductor.org/packages/release/bioc/html/RUVSeq.html

`SARTools` - https://github.com/PF2-pasteur-fr/SARTools

`tximport` - https://github.com/mikelove/tximport

RNA classification

The method used to isolate, enrich and sequence a sample will affect the composition of the sequencing data in terms of the types of RNA species represented and their relative abundances [12, 14, 39, 136]. Most RNA-seq studies are interested in protein-coding transcripts, and appropriately use Poly-A capture—or rRNA depletion if focusing on prokaryotes—to enrich for mRNA molecules [14]. Such enrichment is especially necessary to diminish the abundance of rRNAs, which would otherwise represent a majority of the sequenced molecules [12, 39]. However, 'contaminant' RNA species can still make their way into the assembled data, despite applying pre-assembly filtering measures to exclude such species (see section 2). *In silico* RNA sequence classification can therefore be used to enrich the data post-assembly for the RNA of interest.

In silico classification is mostly performed ad hoc. If the purpose of classification is simply to sieve out mRNAs from the rest, this can be easily achieved by assessing the coding potentials of the assembled contigs using tools like `CPC2` [137] or `CPAT` [138], and retaining only those contigs that score above some satisfactory coding potential threshold. An alternative path to the same end result would be to retain only those contigs that produce a peptide sequence when passed through a translation tool (see Section 'Sequence translation' for an overview of these tools).

More granular classification can be obtained by using the tool `Infernal` [139]. `Infernal` (INFERence of RNA ALignment) is capable of classifying input sequences into rRNAs, tRNAs and lncRNAs on the basis of sequence comparison against a reference database. `Infernal` uses co-variance models and the `Rfam` [41] database to classify the input sequences. By process of elimination (i.e. whatever is not an rRNA/tRNA/lncRNA), `Infernal` can indirectly help identify mRNAs from the assembly. The `Infernal-Rfam` workflow's output can be difficult to parse for the purpose of RNA classification, and we therefore recommend reading this portion (https://docs.rfam.org/en/latest/faq.html#rfam-and-infernal) of the `Rfam` documentation.

If only rRNA classification is needed, `barrnap` is a fast and lightweight option. It uses the `Rfam`, `SILVA` [42] and `NCBI RefSeq mitochondrial` [140] databases to identify and annotate rRNAs. If `SortMeRNA` or other an equivalent classification tool has been deployed before assembly, then the reads will have to be assembled separately prior to annotation with `barrnap`. An older tool, `RNAmmer` [141], continues to be available for use as in stand-alone and web server formats for the purpose of rRNA identification also.
lncRNAs are RNA molecules longer than 200 nucleotides with low coding potential [142, 143]. These molecules typically play regulatory roles in the cell, either directly as RNA entities, or via the short 'micropeptides' that result from their translation [5, 143]. Classification/identification of lncRNAs is typically achieved by elimination; that is, all sequences that are of sufficient length and have not been classified as some other RNA species (e.g. mRNA or rRNA) and have a low coding potential must be lncRNAs. We direct the interested reader to consult Motheramgari et al. [144] and Kashyap et al. [145] for demonstrations of elimination techniques for classifying lcnRNAs.

Finally, RNA classification can also be achieved via sequence searches against appropriate databases (e.g. `NCBI RefSeq RNA`). We discuss sequence searches in Section 'Identity assignment via homology transfer'.

Links:
`barrnap` - https://github.com/tseemann/barrnap
`CPAT` - https://github.com/liguowang/cpat, http://lilab.research.bcm.edu/ (web server)
`CPC2` - https://github.com/gao-lab/CPC2_standalone, http://cpc2.gao-lab.org/ (web server)
`Infernal` - http://eddylab.org/infernal/, https://github.com/EddyRivasLab/infernal
`NCBI RefSeq` - https://www.ncbi.nlm.nih.gov/refseq/
`Rfam` - http://rfam.xfam.org/, https://docs.rfam.org/en/latest/index.html
`SILVA` - https://www.arb-silva.de/
`RNAmmer` - http://www.cbs.dtu.dk/services/RNAmmer/ (web server, standalone download link)

Sequence translation

A core element in the downstream analysis for RNA-seq data involves the translation of assembled sequences into their corresponding amino acid sequences, and on the nucleotide level into the protein coding sequences (CDS) not containing any untranslated regions (UTRs). A correct characterization of CDS is not only important for profiling the protein-coding fraction of a transcriptome, but also for an accurate classification of UTRs and non-coding sequences/regions which may be of interest in the context of gene regulation [146].

There are a number of tools that can predict coding regions, and subsequently translate them into amino acid sequences. These tools are based on probabilistic models that take nucleotide composition as well as length of open reading frames (ORFs) into account for their predictions [45]. Tools like `TransDecoder` [45], `Prodigal` [147], `GeneMarkS-T` [148] and `CodAn` [146] are so-called *ab initio* predictors, meaning that the prediction model is based on self-training methods. This includes identifying a certain number of long ORFs from within the assembly, which serve as test set for predicting CDS from the remaining contigs afterwards [146, 148]. A recently released tool named `BOrf` [149] focuses on ORF prediction for strand-specific RNA-seq, but also performs acceptably with non-specific data.

Alternatively, translations can also be obtained by simply scanning the inputs for ORFs in all six reading frames, and reporting all translations. There are many tools that can perform this including the web-based `NCBI ORFfinder` and `EBI EMBOSS-Sixpack` [150], as well as `esl-translate` from the `HMMER` suite [151] and `extractorfs` from `MMseqs2` [108–111]. Finally, a novel approach to recovering a protein data set from RNA-seq data is presented in the tool `PLASS` [152] which directly scans short reads for ORFs and extends these into amino acid contigs by examining overlaps between the translations.

CDS prediction and sequence translation is not always performed, but it is recommended as sequence comparisons (necessary for annotation, see Section 'Transcriptome functional annotation') are more sensitive with protein sequences rather than with the corresponding nucleotide counterparts. We direct the interested reader to refer to Section 4.2, Chapter 4 of Koonin and Galperin [153] and Pearson [154] for explanations.

Links:

`BOrf` - https://github.com/betsig/borf

`CodAn` - https://github.com/pedronachtigall/CodAn

`EMBOSS-Sixpack` - https://www.ebi.ac.uk/Tools/st/emboss_sixpack/

`esl-translate` - http://hmmer.org/, https://github.com/EddyRivasLab/easel

`GeneMarkS-T` - http://exon.gatech.edu/GeneMark/license_download.cgi

`ORFfinder` - https://www.ncbi.nlm.nih.gov/orffinder/ (web server)

`PLASS` - https://github.com/soedinglab/plass

`Prodigal` - https://github.com/hyattpd/Prodigal

`TransDecoder` - https://github.com/TransDecoder/TransDecoder

Transcriptome functional annotation

Once a transcriptome has been assembled and quality controlled, its sequences can be studied to elucidate the functionality they individually and collectively represent in the circumstances under which the data were obtained. For instance, an assembled transcript that is overrepresented in the assembled transcriptome may code for a structural protein, indicating that the cell was in a state of enhanced structural modification activity at the time of sampling.

Functional annotation is the process of inferring and assigning information concerning the biological functionality of the sequence using *in silico* methods. Functional annotation is usually understood to refer to the annotation of mRNAs, as it is the proteins, which these sequences are translated into, that carry out the various activities within the cell (and hence contribute to the functioning of the cell). As such it can be argued that the process of functional annotation begins with RNA classification and amino acid sequence prediction (Sections 'RNA classification' and 'Sequence translation'). However, as these steps do not yield information regarding the exact functionality of the transcripts, we do not include them under the aegis of functional annotation.

There appears to be no given definition for what constitutes a standard approach to transcriptome functional annotation. A survey of relevant literature reveals that a variety of methods have been adopted in the past. For instance *Chabikwa et al.* [20] only used homology transfer (see Section 'Identity assignment via homology transfer'), while *Sayadi et al.* [155] used a combination of different approaches to annotate their transcriptome. Based on a review of 18 papers describing annotations of *de novo* assembled transcriptomes (Table S1), we describe the transcriptome functional annotation procedure as comprising of the following steps (see also Figure 4):

- Homology transfer and identity assignment via sequence search.
- Sequence feature annotation.
- Gene ontology (GO) and biochemical pathway annotation.

We caution that these aforementioned steps are not necessarily independent nor strictly compartmentalized. For instance, sequence features *can* be annotated based on homology transfer, and need not always be performed as an independent step. Nor is it the case that all three indicated steps are mandatory to establish an annotated transcriptome. Instead, the objective is to delineate the myriad of aspects involved in transcriptome annotation—and introduce the associated tools and resources—in a succinct and concise manner.

Identity assignment via homology transfer

Homology transfer can be considered the most basic form of transcriptome annotation. Here, a descriptive identity (e.g. 'Protein kinase') and functional properties are assigned to a hitherto undecorated sequence on the basis of a sequence search. In this method the assembled sequences are supplied to sequence search tools as `queries`. A database of well-annotated reference sequences are provided as the `targets`. The tools then use heuristic methods [156] to find `matches` between these inputs. Typically, each `query` has more than one `matched target`. The best `match` is usually chosen based on the significance of the so-called `e-value` (more on this below). This leaves each `query` with a single `match`, whose identity and annotation are assigned to the query. It is appropriate to transfer annotations in this manner because the `e-value` is an indicator of the likelihood of the observed sequence similarity arising purely by chance [154]. In other words, a sufficiently low `e-value` (e.g. `0.00000001`) is indicative of homology (shared evolutionary ancestry) which subsequently implies conserved function [154, 157]. Hence the name 'homology transfer' [154].

Homology transfer can be performed both with nucleotide sequences as well as (translated) protein sequences from transcriptomes. Proteins are more conserved than their corresponding mRNA sequences (see Chapter 4 of *Koonin and Galperin* [153]). Protein sequence searches are also more sensitive and faster, due to the expanded alphabet of 21 amino acids and shorter sequence length in comparison to their nucleotide counterparts. Further most functional properties (e.g. enzymatic domains) are only really meaningful in the context of a protein sequence. Because of these reasons, it is customary to either use translated searches, or pre-translated sequence sets (see Section 'Sequence translation'), for functional annotation.

In addition to identifying homologs to the sequence, sequence features such as domains can also be transferred if the sequences are similar enough (if, for instance, they have the same length). However, such

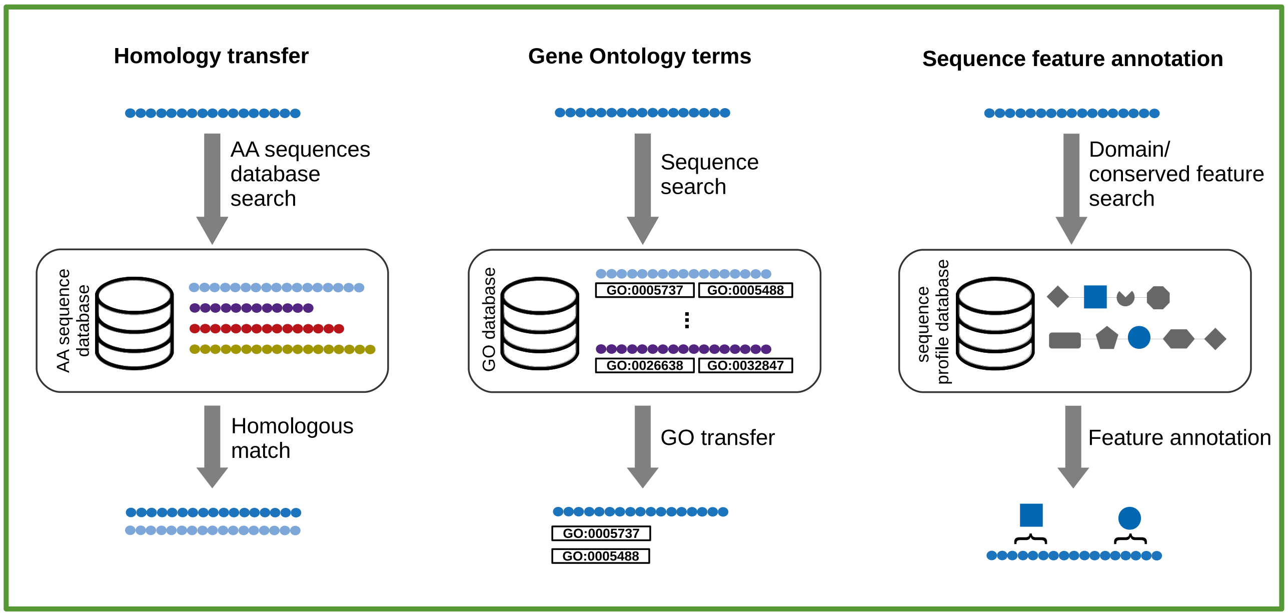

Figure 4. Transcriptome functional annotation comprises of techniques to assign human-comprehensible identifiers and functional characteristics to the transcripts. It includes searching for homologs based on sequence similarities and identifying assembled sequences (homology transfer), domain and other sequence feature identification (sequence feature annotation) and assigning standardized descriptors for the sequences' biological properties (Gene Ontology terms).

annotations would be insufficiently resolved as they would have been transferred only on the basis of sequence similarity. For instance, although two sequences are highly similar, they might not necessarily share all domains, and annotating one of them with the domains of the other on the basis of similarity alone could yield erroneous domain attributions.

TOOLS: The most commonly used sequence search tool for this purpose is the famous `BLAST` [158] suite (more precisely `BLAST+` [159]). `BLAST` comprises of several sub-tools specialized for different types of search strategies. `blastn` can be used to perform searches with nucleotide sequence `queries` versus nucleotide sequence `targets`. `blastp` is its counterpart for amino acid `queries` and `targets`. The suite can also perform translated searches with `blastx`. Here, `query` nucleotide sequences are translated and searched against an amino acid sequence `targets` database. The inverse search operation (amino acid `queries` versus nucleotide `targets`) can be performed with `tblastn`. The suite offers `rpsblast` and `rpsblastn` to facilitate identification of conserved domains in amino acid and nucleotide `queries`, respectively. Finally, two options are offered by `BLAST` for high sensitivity searches. `deltablast` can perform highly sensitive searches with amino acid `queries` against `amino acid` databases. `psiblast` can be to identify protein homologs for amino acid `queries` against a database of amino acid `targets` using sequence-profile searches.

Although `BLAST` is the mainstay of sequence search tools, it is very slow, and does not scale well in terms of speed with growing input size. For instance, *Buchfink et al.* [160] indicate `blastp` running on ca. 21 000 CPUs would take around 2 months to scan 280 million `queries` against 40 million `targets`. Given that *de novo* transcriptomes may contain upwards of 100 000 transcripts to annotate, `BLAST` becomes an infeasible option—especially as a part of larger workflows. Luckily, alternatives to `BLAST` exist that are just as sensitive but magnitudes faster. Two such tools are discussed in the next few paragraphs below.

`Diamond` [160] is a special-purpose tool that is exclusively geared toward searching against protein databases. As of version `v2.0.9.147`, `Diamond` is as sensitive as `blastp` while being 80× faster. The tool is an almost drop-in replacement for `blastp`, both due to its speed, and due to the fact that it mimics the `BLAST` command line function calls and output formats. The main drawback of the tool is that it can only operate with amino acid sequences as `targets`. However, it does accept both nucleotide and protein `queries`. Therefore, it is a great choice for performing protein versus protein (or translated nucleotide versus protein) searches while annotating *de novo* assembled transcriptomes.

The other main alternative to `BLAST` is `MMseqs2` [108–111] (Many-against-Many sequence searching). In some senses, it is the more equivalent alternative to `BLAST` as it is also a modular software suite in its own right with extensive capabilities. `MMseqs2` supports nucleotide and amino acid sequences as both `queries` and `targets`, and supports translated searches via a bespoke `search` module. Although not nearly as fast as `Diamond` at equal levels of sensitivity, `MMseqs2` is still 8–10× faster than `BLAST` at comparable levels of sensitivity. One drawback of `MMseqs2` is that it uses its own database format which is incompatible with the `BLAST` database format.

However, it can present outputs in the default BLAST format. But on the other hand MMseqs2 offers sequence–sequence search, sequence–profile search, sequence clustering and taxonomy assignment, making it a one-stop solution transcriptome annotation workflows. For instance, it can be used to replace CD-HIT for clustering and BLAST for sequence search.

DATABASES: The quality of annotation via homology transfer depends upon the quality of the reference databases used. It is advisable to use multiple databases encompassing different standards of curation and taxonomic scope. While a single database of references from closely related species will potentially result in fewer false annotations, a database that is taxonomy-agnostic will be invaluable in annotating novel sequences that might have otherwise been missed.

There are several general-purpose sequence databases which can be used in their entirety as reference databases, or as sources for a manually curated reference sequence set. NCBI's [161] NR (protein) and NT (nucleotide) are non-curated, and are the largest sequence databases available today. For a well-curated set, the non-redundant NCBI RefSeq database might be preferable. The UniProt [162] consortium's Swiss-Prot database contains the highest quality, manually curated protein sequence set available anywhere. It can be considered the gold standard annotation source. The UniProt/TrEMBL database is the uncurated counterpart with a larger number of sequences. If all UniProt sequences are desired, the UniRef [163, 164] series of databases may be of interest, which represent subsets obtained by clustering at various levels of sequence identity. There are also taxon-specific databases maintained by various consortia. Some examples include FlyBase [165] (*Drosophila*), WormBase [166] (nematodes) and PLAZA [167, 168] (plants). Such sequence repositories are best found by reviewing relevant literature.

Links:

BLAST - https://blast.ncbi.nlm.nih.gov/Blast.cgi (web server), https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/ (standalone tool download page)

Diamond - https://github.com/bbuchfink/diamond

FlyBase - https://flybase.org/

MMseqs2 - https://github.com/soedinglab/MMseqs2, https://search.mmseqs.com/search (web server)

NCBI RefSeq - https://www.ncbi.nlm.nih.gov/refseq/, https://ftp.ncbi.nlm.nih.gov/refseq/release/ (FTP)

NCBI NR and NCBI NT - https://ftp.ncbi.nlm.nih.gov/blast/db/FASTA/ (FTP)

PLAZA - https://bioinformatics.psb.ugent.be/plaza/

UniProt - https://www.uniprot.org/

WormBase - https://wormbase.org/

Sequence feature annotation

A very important aspect of annotation is the precise identification of functional sequence features such as protein domains, disordered regions, motifs, transmembrane helices and so forth. While these can be annotated via homology transfer, the process can be error prone and have poor resolution. For instance, an assembled partial sequence may be identified as being homologous to a protein containing a bZIP domain, without explicitly aligning to the sub-sequence corresponding to that domain. Annotating the sequence with a bZIP domain would be erroneous in this case. Therefore, approaches that explicitly detect the presence of such features is preferable for the purposes of such annotations.

Sequence features such as domains are typically annotated by comparing the query sequence against databases of Hidden Markov Model (HMM) [169] representations of sequence profiles [170, 171]. Sequence profiles are compact representations of multiple sequence alignments (MSAs) [172] of protein families wherein the aligned residues correspond to domains or other conserved features. Domains on the query sequence(s) can be detected by performing a sequence-profile alignment against the HMMs using a tool such as HMMER3 [151]. But not all sequence features are predicted this way. For instance, signal peptides are predicted by the tool SignalP using a deep learning method [173], the tool fLPS [174] uses a statistical approach called probability minimization to predicted biased regions in amino acid sequences and protein motifs [175] can be predicted using simple pattern matching techniques. As such a large variety of tools and databases exist to facilitate annotation of various sequence features.

InterProScan [176] is a metatool that integrates a number of feature prediction methods, databases and analyses into a single user-friendly interface. It negates the need for having to install and maintain a variety of databases and tooling manually. In addition to annotating protein functional and structural domains, it can also be used to classify sequences (e.g. into protein families on the basis of gene ontology), and detect transmembrane and disordered regions. InterProScan can be run on both nucleotide and amino acid sequneces. A web server version of the tool is also available. The interested reader can refer to https://interproscan-docs.readthedocs.io/en/latest/HowToRun.html#included-analyses for a complete list of analyses included in the tool.

A large number of resources are available for annotating a myriad variety of sequence features. It is advisable to scan recent literature for relevant tools for niche use-cases. In addition to InterProScan, we would like to highlight two tool repositories that should be of interest. The software portal at DTU Health Tech (https://services.healthtech.dtu.dk/software.php) hosts a number of useful annotation tools including predictors for post-translational modifications. The European Bioinformatics Institute (EMBL-EBI) provides a wide variety of tools and data resources at https://www.ebi.ac.uk/services that may also be of interest in the context of sequence annotation.

The huge variety of annotatable sequence features can be overwhelming to choose from. It is advisable to only annotate those features that will be of interest for downstream applications. For a standard transcriptome annotation workflow, it should suffice to annotate protein functional domains (e.g. against PFam [177]) and structural domains (e.g. against CATH-Gene3D [178, 179]) using a tool such as InterProScan.

Links:

CATH-Gene3D - https://www.cathdb.info/

fLPS - https://biology.mcgill.ca/faculty/harrison/flps.html, https://github.com/pmharrison/flps

HMMER3 - http://hmmer.org/, https://www.ebi.ac.uk/Tools/hmmer/ (web server)

InterProScan - https://github.com/ebi-pf-team/interproscan, https://www.ebi.ac.uk/interpro/ (web server)

Pfam - http://pfam.xfam.org/

Tools at DTU Health Tech - https://services.healthtech.dtu.dk/software.php

Tools at EMBL-EBI - https://www.ebi.ac.uk/services

Gene ontology and pathway annotation

It is useful to assign descriptors from a controlled vocabulary (ontology) that associates the sequences with specific biological phenomena in a consistent manner. For this purpose the assembled sequences can be annotated with Gene Ontology (GO) terms [180, 181] (see *Dessimoz and Škunca* [182] for details on GO terms and their usage). There are several possibilities for annotating sequences with GO terms.

Firstly, GO terms can be transferred from homologous sequences via sequence search (Section 'Identity assignment via homology transfer'). If InterProScan [176] (Section 'Sequence feature annotation') is being used, it can be asked to annotate GO terms using the --goterms commandline switch. If a standalone GO annotation tool is required, the functional annotation tool eggNOG-mapper [183, 184] is one of the only open source, free-to-use options (see Section 'Transcriptome annotation suites'). Alternatively, the Orthologous Matrix Browser (OMA Browser) [185] offers some web-based options for GO annotation. For those willing to pay a licensing fee (or use a free version with limited capabilities), the BLAST2GO [186] functional annotation suite is available as an alternative.

Pathway annotation refers to assigning the sequences to one or more biochemical pathways. There are two popular pathway annotation databases: the Kyoto Encyclopedia of Genes and Genomes (KEGG) [187–189] and reactome [190]. The latter is more human-centric, making KEGG the more widely used database, especially for non-model organisms. InterProScan, eggNOG-mapper, and BLAST2GO all transfer pathway annotations alongside GO annotations, so no additional tooling is usually necessary. The annotation suite Trinotate mentioned in Section 'Transcriptome annotation suites' also provides GO annotations by transitive assignment from Swiss-Prot to BLAST hits. If only a homology transfer (Section 'Identity assignment via homology transfer') is being performed, pathway annotations can be transferred akin to GO annotations. Pathway assignments can also be obtained independently by annotating the transcriptome via the KEGG Automatic Annotation Server (KAAS) or reactome web servers, respectively. For KEGG annotations, the GhostKOALA [191], BlastKOALA [191] and KofamKOALA provide additional functional annotation options.

Links:

BLAST2GO - https://www.blast2go.com/

eggNOG-mapper - https://github.com/eggnogdb/eggnog-mapper, http://eggnog-mapper.embl.de/ (web server), http://eggnog5.embl.de/#/app/home (eggNOG database)

InterProScan - https://github.com/ebi-pf-team/interproscan, https://www.ebi.ac.uk/interpro/ (web server)

KAAS - https://www.genome.jp/kegg/kaas/

KEGG - https://www.genome.jp/kegg/

BlastKOALA - https://www.kegg.jp/blastkoala/

GhostKOALA - https://www.kegg.jp/ghostkoala/

KofamKOALA - https://www.genome.jp/tools/kofamkoala/

OMA Browser - https://omabrowser.org/oma/home/

reactome - https://reactome.org/ (including analysis web server)

Transcriptome annotation suites

Transcriptome annotation involves a number of different tools and databases, dealing with which can quickly become a cumbersome task in of itself. In recent years, a number of annotation *suites* have been developed with the objective of making this an easier process. In most cases, these wrap existing tools into a single easy-to-use interface while adding features useful for transcriptome annotation (e.g. expression-based filtering). In all cases, some combination of sequence identity assignment, sequence feature detection and gene ontology/-pathway assignment are performed as a part of the annotation procedure (as described in Section 'Transcriptome functional annotation').

Trinotate [192] is arguably the most well-known open source, free-to-use annotation suite. It accepts both nucleotide and amino acid sequences as inputs. The former are translated using TransDecoder. It uses BLAST+ for homology search, and HMMER3 (against Pfam) for sequence feature annotation. Optionally, it can run rnammer for RNA classification, Signalp for signal peptide identification and tmhmm [193] for predicting transmembrane domains. User-supplied databases are also accepted in addition to the default UniProt/Swiss-Prot database for the homology search step. GO annotations are provided via transitive assignments from top homology search hits. Trinotate uses a SQLite to collate and summarize the results. A graphical user interface (GUI)–TrinotateWeb–is available for visualizing and navigating the results. The tool's main advantage is its tight integration with the Trinity assembler.

Dammit is a popular alternative to Trinotate. The tool is written in python, and is available via the conda package management system. This ensures ease of installation (all dependencies come pre-packaged), upgrade and use. The tool offers a default set of reference databases (including NCBI NR) but also accepts user-supplied ones. Rather than relying on homologs for annotation, Dammit searches with a specialized reciprocal best hit method for orthologs (using LAST), while accounting for issues caused by the presence of transcript isoforms in the assembly. Annotating via orthology is superior as these are genes related by speciation that have the same function as opposed to generic homologs which may be paralogs where function need not be conserved (see *Altenhoff et al.* [194–196]). Sequence feature annotation is performed using HMMER3 against the Pfam database, and RNA classification using Infernal against Rfam. BUSCO scores are included in the annotation report as the tool is a part of the pipeline as well.

EnTAP [197] is a recently released annotation suite with a focus on *de novo* assemblies of non-model eukaryotes. It uses the fast Diamond aligner internally for identity assignment via homology, and uses eggNOG-mapper for gene ontology annotation. Sequence features can be annotated via homology transfer or via an optional InterProScan run. EnTAP accepts both nucleotide and amino acid sequences. With the former it is possible to filter on the basis of expression level using an FPKM threshold before translation with TransDecoder or GeneMarkS-T. No nucleotide-level annotations (e.g. RNA classification) are possible as of the current release (v0.10.8). EnTAP offers several unique features helpful for annotations of non-model organisms. For instance, if NCBI or UniProt are used, annotations can be enriched and/or filtered out on the basis of taxonomic scope with regards to the species being annotated. e.g. metazoan matches in the sequence search can be prioritized while bacterial sequences can be indicated as contaminants when annotating an arthropod. The tool also presents the annotation results in multiple formats, and incorporates useful statistics and figures.

Annocript [198] is an annotation suite built around BLAST+. Annotations via homology transfer are based on either user-defined reference sets or a default UniProt database. Sequence features are annotated using rps-blast and NCBI's Conserved Domain Database (CDD) [199]. Annotations are stored and summarized via a MySQL database. Annotations include GO terms and pathways.

Sma3s (Sequence massive annotator using 3 modules) [200] is a general purpose annotation suite that can also be used with transcriptomes. Written in perl, its only dependency is the BLAST+ suite. Annotations are made against the UniProt/UniRef90 database using homology transfer, and include sequence identifiers, GO terms, as well as pathways. The tool can also be used with custom reference databases.

TOA (Taxonomy-oriented Annotation) [201] and TRAPID 2.0 [202, 203] are transcriptome annotation platforms with a focus on plant species. The former is a platform-agnostic, offline tool while the latter is a web server that requires registration. Both tools use methods similar to the more mainstream annotation suites, but restrict the reference databases to select plant-related ones.

The Transcriptome Computational Workbench (TCW) [204] is an interesting annotation tool written in Java that can not only annotate multiple transcriptomes but can also perform comparisons between them. In addition, the tool has built-in functionality to carry out differential expression analysis. TCW is arguably one of the oldest transcriptome annotation tool in existence, having undergone continuous development since 2013 [205].

Given the increasing complexity of RNA-seq experiments and concerns regarding reproducibility, the use of bioinformatics workflow managers (see Section 'Workflow managers') to orchestrate reproducible and extensible workflows has become a popular approach. The domain of *de novo* transcriptome assembly and annotation has not been exempted from this revolution. FA-nf [206] and transXpress are two such annotation platform. The former is functional annotation suite billed as being specialized for non-model organisms, while the latter is a complete assembly and annotation pipeline that can be operated almost turnkey.

The functional annotator eggNOG-mapper deserves a honorable mention here, since it provides a full set of relevant annotations including orthologs, domains (from Pfam), GO terms and pathways despite not being billed as a transcriptome annotation tool. Likewise, the Orthologous Matrix (OMA) Browser mentioned in Section 'Transcriptome annotation suites' offers a stand-alone option (OMA StandAlone [207]). This tool can perform orthology predictions and GO annotations, but does not provide domain annotations. Other general purpose functional annotation tools such as the WebMGA [208] web server and PANNZER2 [209] can also be used to annotate transcriptomes via their translated sequence sets.

Finally, BLAST2GO is perhaps the most popular transcriptome annotation tool. It is not open-source and requires a paid subscription for full functionality. The tool is built around annotation transfer based on BLAST+ homology searches, coupled with a user-friendly GUI. A 'basic' version with limited capabilities is available for free use. The BLAST2GO tool is a part of the larger OmicsBox (https://www.biobam.com/) bioinformatics platform which offers a wide variety of bioinformatics-related tools and analysis (including *de novo* transcriptome assembly).

Links:

Annocript - https://github.com/frankMusacchia/Annocript

Dammit - https://github.com/dib-lab/dammit, http://dib-lab.github.io/dammit eggnog-mapper - https://github.com/eggnogdb/eggnog-mapper, http://eggnog-mapper.embl.de/ (web server)

EnTAP - https://github.com/harta55/EnTAP

FA-nf - https://github.com/guigolab/FA-nf/tree/0.3.1

OMA StandAlone - https://omabrowser.org/standalone/

PANNZER2 - http://ekhidna2.biocenter.helsinki.fi/sanspanz/

Sma3s - https://github.com/UPOBioinfo/sma3s, http://www.bioinfocabd.upo.es/web_bioinfo/sma3s

TCW - http://www.agcol.arizona.edu/software/tcw/, https://github.com/csoderlund/TCW

TOA - https://github.com/GGFHF/TOA

TRAPID 2.0 - http://bioinformatics.psb.ugent.be/trapid_02/

transXpress - https://github.com/transXpress/transXpress-nextflow (**Nextflow** version), https://github.com/transXpress/transXpress-snakemake (**Snakeake** version)

Trinotate - https://github.com/Trinotate

WebMGA - http://weizhong-lab.ucsd.edu/webMGA/server/

Comparing transcriptome assemblies

A set of assemblies can be used in a comparative transcriptomics approach, for instance, to identify conserved genes or specific gene expression patterns associated with different organisms of interest. If more than two organisms are studied, a first step in such analysis consists in constructing a phylogenetic tree describing the evolutionary relationship between the representative transcriptomes. To do so, a suitable approach taking advantage of the previously identified BUSCO genes (during post-assembly quality control, see Section Post-assembly quality control) can be used [77]. BUSCO genes have been curated to represent a conserved set of slowly evolving housekeeping genes which can be used for phylogenetic analysis. A common approach consists of retrieving the translated transcript sequences associated with each BUSCO gene in the different transcriptomes. Then, MSAs are performed with tools like MAFFT [210] or FAMSA [211], for each housekeeping gene with a single copy in every transcriptome of interest. These MSAs can then be used to construct phylogenetic trees. There are many tools that can be used to build such trees, e.g. RAxML [212]. A representative consensus species tree reflecting the phylogeny of the total set of single copy BUSCO gene trees can then be reconstructed, using dedicated methods like ASTRAL-III [213]. To analyze the presence or absence of genes across multiple transcriptomes, and be able to compare the expression of the conserved ones, it is essential to identify orthologs and paralogs within the studied data set [194]. While numerous orthology prediction methods have been developed over the last two decades, OrthoFinder [214] has become widely adopted and quasi-standardized, due to its speed and ease of use. The tool uses a combination of sequence clustering and tree building methods to group sequences (from all input samples) into orthogroups. Orthogroups basically represent collections of sequences that are related at *their* root node by speciation [194]. From these data, OrthoFinder is able to estimate gene copy numbers, orthogroup trees, a consensus species tree and other useful evolutionary data (e.g. sets of single-copy orthologs, pairwise orthologs, etc.). A recent alternative to OrthoFinder is the very fast JustOrthologs method [215]. As it performs comparisons between pairs of organisms, it is especially adapted to the study of pairs of transcriptomes, but its use can be extended to the comparison of numerous ones using the associated CombineOrthoGroups script, which combines pairs of orthologs into orthogroups. Likewise, the OMA StandAlone [207] function annotation tool can also perform comparisons between the input assemblies.

While BUSCO-derived phylogenies and orthlogy prediction have been commonly adopted in the last decade for comparing assembled transcriptomes, a recent study addressed the biases and limits of such approach [216]. By comparing low- and high-quality transcriptome assemblies (scored with TransRate [80], see Section 'Post-assembly quality control'), it highlighted that some important skews in phylogenetic and orthology prediction data can come from using low-quality assemblies. Not to draw any wrong biological interpretation from comparative transcriptomics, it is therefore important to consider assembly quality at every point in such an analysis.

Links:

BUSCO - https://busco.ezlab.org/

FAMSA - http://sun.aei.polsl.pl/REFRESH/famsa

JustOrthologs - https://github.com/ridgelab/JustOrthologs

MAFFT - https://mafft.cbrc.jp/alignment/server

OMA StandAlone - https://omabrowser.org/standalone/

OrthoFinder - https://github.com/davidemms/OrthoFinder

RAxML - https://raxml-ng.vital-it.ch

Workflow managers

Modern biological science is high-throughput and highly data-driven. Investigations often deploy composite computational analyses using multiple tools to process the data. A collection of such tools/programs organized in a specific manner to produce results from which biological inferences can be drawn is known as a workflow or pipeline [217]. Similar to how a 'wet-lab' protocol represents the set of steps required to transform a 'raw' sample into comprehensible output (e.g. sequencing an RNA molecule), a bioinformatics workflow/pipeline represents an equivalent collection of steps to do the same with digital data [218] (e.g. identifying an RNA sequence as an mRNA). A workflow consisting of a small number of tools and/or a small amount of data can be handled by the investigator(s) by executing each step/tool manually. However, for projects dealing with large volumes of data and/or a complex, interconnected collection of tools, automatization of the workflow becomes unavoidable [219].

It is in such cases that workflow managers/workflow management systems (WfMS) become useful. A WfMS is a specially designed programmatic framework that can be used to automate a pipeline consisting of numerous steps that must be manually executed [217]. It allows the user to define the computational pipeline as graph wherein each node represents a particular processing step. The edges connecting the nodes are directed and represent data flowing from one node after being processed by it (its output) to another node as its input. Thus, the graph also describes the order in which the components of the pipeline will be executed. The user must define the individual steps of the workflow in terms of the inputs, expected outputs and the tool(s) required to generate them. Typically, it is required that the user specifies the exact command to run the tool using placeholder values to define the inputs and outputs (for example `mytool inputfile outputfile`). The workflow manager then handles the execution of the pipeline. This includes allocating resources (processing threads, memory, etc.) and deducing the order in which the individual commands have to be executed. The advantage of using a workflow manager is that analyses become optimized, especially when dealing with large volumes of data and metadata as the execution details are abstracted away from the user [217]. Further, as the user only needs to define the workflow but not the specifics of execution, the same pipeline can be executed on a local server, cluster or in the cloud, making pipelines scalable and easy to prototype [220]. Finally, using a workflow manager also makes analyses reproducible, shareable and easy to run as workflows can be run anywhere, and can often also install the correct versions of the tools by themselves [221]. This also makes bioinformatics accessible—as non-experts can avail themselves of pre-existing workflows for their own research [222].

Workflow managers can be sorted into two groups—command-line interface-based (CLI) and GUI-based. The two groups primarily differ in how the workflow manager itself is presented to the user. A CLI-based WfMS is a command-line program that executes a text document (script) describing the analytical workflow. Most WfMS have a particular programming language they can recognize, and the script must be written in this language. In comparison, a GUI-based manager exposes the same equipment to the user via a point-and-click environment. Users are able to construct workflows by dragging and dropping and interconnecting icons representing tools and data. Experienced users will save time by working with CLI managers, since writing a command for a particular process is faster than manually navigating the interface panels of a GUI program. On the other hand, GUI WfMS are much more user-friendly and do not demand knowledge of programming.

Recent publications [217, 222, 223] indicate that the most popular WfMS today include `Nextflow` [224], `Snakmake` [221] and `Common Workflow Language` (`CWL`) [225]. All three are CLI-based, open-source and free-to-use, but have their differences.

`Snakemake` is based on `Python` which is among the most popular programming languages [226]. As a result of `Python`'s user-friendly syntax, workflows written in `Snakemake` are not only very readable but also approachable for beginners. In addition to facilitating custom workflows, users can also import external pipelines, and merge and edit them depending on their needs [221]. In this regard, some so-called wrapper scripts are offered through the `Snakemake Wrapper Repository` (https://snakemake-wrappers.readthedocs.io/en/stable/) that are templated for common bioinformatics tasks. `Snakemake` pipelines are portable and scalable. They can be executed on a wide range of environments starting from single-core workstations to HPC clusters [227]. The only requirements are `Python` and `Snakemake` itself.

`Nextflow` is a powerful WfMS based on the `Groovy` programming language. The central idea is that most bioinformatics tools are `Unix`-based, and data are passed between the tools (and processed additionally) using custom scripts often written in different languages (e.g. `Python` and `R`). Consequently, `Nextflow` permits chaining together scripts (and tools) written in different languages as long as they can be executed on a `Unix`-like operating system [228]. Like `Snakemake` `Nextflow` is also scalable and platform-agnostic with regards to execution capabilities. A salient feature of `Nextflow` is `nf-core` (https://nf-co.re/) [229]. This is a curated repository of bioinformatics pipelines written in `Nextflow` that cover a range of use cases including RNA-seq, sequence assembly, phylogenetics and sequence annotation.

`Common Workflow Language` (`CWL`) is another CLI-based WfMS. However, while the previous two are focused on pipeline development, `CWL` also represents a set of standards defining what a workflow language should look like and contain. This is because the advent of ‘big data’ in biology has led to the introduction many WfMS implementations (not discussed here) all of which use different approaches for describing their pipelines [230]. Adherence to `CWL` standards would allow pipelines to be shared, easing the process of testing and comparing new methods acquired from other researchers, despite having been implemented in different WfMS [218]. `CWL` itself represents a set of standards, and cannot be used to draft a workflow. A set of `CWL`-compliant WfMS implementations—e.g. `CWL-Airflow` (https://github.com/Barski-lab/cwl-airflow) [230]—can be found on its website (https://www.commonwl.org/#Implementations); a reference implementation (`cwltool`) developed by the `CWL` team is also available.

`Workflow Description Language` [231] (`WDL`) is a WfMS with straightforward syntax. Analogous to `CWL`, it also represents a language definition and is not executable in of itself: a WDL-compliant execution engine is required to execute workflows. The two main execution engines are Cromwell and miniwdl. WDL is under active development, supports multiple programming languages and has a growing ecosystem of tools and pre-defined workflows. Finally, as suggested above, tooling to design and execute workflows (bioinformatics or otherwise) exists elsewhere—often as language-specific implementations. For instance, the Targets [232] package enables this in the R programming language popular among biologists and bioinformaticians. Such implementations permit users to design and execute workflows using a language familiar to them.

Galaxy [23] is arguably the most used web-based data analysis platform for biology [233]. It is intended to serve researchers from a broad variety of backgrounds looking to investigate large quantities of data with complex tools, even those with limited programming experience [234]. It is an open-source WfMS with a GUI that allows for work to be carried out entirely in a web browser. Galaxy is analysis-agnostic: although originally written for genomic analyses in mind, it has since been used for a vast variety of research (e.g. biophysics [235]). A large, open repository of tools contributed by the user community is available through the Galaxy ToolShed [236]; installation is easy as Galaxy automatically activates the required dependencies also. A large collection of prescripted workflows for a variety of common analytical tasks are also available, reducing the need for recreating boilerplate routines. The Galaxy approach also ensure easy documentation of workflows as workflow components can be directly annotated through the GUI. Needless to say, the platform ensure easy reproducibility of workflows. In addition, the platform not only takes care of resource allocation for workflow execution, but also provides the resources themselves in the event that the user is operating on the free public server (https://usegalaxy.org/). Alternatively, the platform itself is available as an open-source tool that can be downloaded, installed and configured for local use (e.g. on a personal computer or an HPC environment). A plethora of customizations to make Galaxy even more user-friendly (e.g. Galaksio [237]) are available and continue to be developed.

Although Galaxy dominates the GUI-based WfMS space, there are a few other alternatives worth mentioning. The Unipro UGENE [238] bioinformatics suite offers an integrated WfMS for constructing workflows with in-built tools. Although the suite is open source and cross-platform, it cannot be used on HPC environments. GenePattern [239] is a more equivalent competitor to Galaxy offering many of the same features including a public server and a version for stand-alone installation.

Links:

Cromwell - https://github.com/broadinstitute/cromwell
CWL - https://www.commonwl.org/
Galaxy - https://galaxyproject.org/ (homepage of the project), https://usegalaxy.org/ (free to use public server)
GenePattern - https://www.genepattern.org/#, https://genepattern.org/ (public server), https://github.com/genepattern (GitHub repository)
miniwdl - https://github.com/chanzuckerberg/miniwdl
Nextflow - https://www.nextflow.io/
Snakemake - https://snakemake.github.io/
Unipro UGENE - https://ugene.net/
WDL - https://github.com/openwdl/wdl

Computational and programmatic considerations

All the necessary tools must be acquired and installed before embarking on the task of assembling and annotating a transcriptomic data set. There is a large variety of tools, all with varying levels of availability and support. It is often the case that the tool is available only on a specific operating system (OS) or requires specialized domain knowledge for installation. Executing tools can be all the more challenging than acquiring them, especially when multiple tools need to be used in concert on extremely large data sets. Assessing the computational resources for deploying these tools can also be very difficult. All these aspects invoke additional considerations that the researcher must take into account before and during the analysis. Addressing all these topics in a thorough manner is a non-trivial endeavor. However, in the interest of signposting useful resources that could be consulted, we address these in an introductory manner below.

Operating systems, programming languages and computational resources

Most tools and software for bioinformatics and analysis in biology have been written for Unix-like operating systems (https://en.wikipedia.org/wiki/Unix-like), and are often designed to be run from within a command line shell [240]. Thus, it is preferable to have access to a computer or computing environment equipped with such an OS.

The most popular Unix-like OSes in use today are Apple's closed-source macOS(https://www.apple.com/macos) and the various 'flavors' of the open-source and free-to-use Linux [241] family. Users of Microsoft Windows (https://www.microsoft.com/en-us/windows/) can install the Windows Subsystem for Linux application to avail themselves of a Unix-like environment on this operating system. macOS users have access to an in-built command line shell.

As a general recommendation, we suggest using the Linux-based Ubuntu operating system and the included GNU Bash shell. This combination is well documented due to a large install base, and is open source, free to use and continually maintained by the developers and community. Many tools of interest are also readily available for this platform via Ubuntu's package manager (https://ubuntu.com/server/docs/package-management), as pre-compiled binaries/executables from the developers, or as source code that can be compiled easily. External package management systems (refer Section 'Tool management') are also easily available for this platform.

Interfacing with one or more programming languages is an aspect potential users of RNA-seq tools will have to consider. Users will often encounter situations where the output from one tool must be fed to another tool as its input, but the output and input formats are incompatible (e.g. a table with four columns is required as an input, but it exists as a table with five columns). In such cases, the user will have to intervene and transform/manipulate the data in order to pass it on through subsequent steps of the analysis. There may also be situations where some portion of the analysis *must* be done in a programming language; for example, almost all popular DE analysis tools (see Section 'Differential expression analysis') are packages that must be accessed through a programming language. In any case, in the interest of reproducibility, efficiency and making problems tractable, it is advisable to become familiar with one or more programming languages.

There are a number of such languages that are popular in bioinformatics (and in biology in general). This includes the eponymous scripting language of the GNU bash shell itself, Python [242] and R [120]. Each of these have their own strengths and weaknesses. Bash is ubiquitous and powerful but has a cumbersome syntax and is only really convenient for short programs. Python is a general purpose language with a very friendly syntax, and is nearly as ubiquitous as Bash. However, its ecosystem for bioinformatics analyses is relatively limited. R is not as prevalent as the other two but is excellent for manipulating and analyzing large amounts of data. Furthermore, it is the language of choice for bioinformatics analysis due to the large number of packages and tools it supports in this regard—especially for '-omics' analyses through the Bioconductor [243] ecosystem.

It is very common to see bioinformatics workflows interspersed with scripts written by the researcher. In larger analytical workflows, e.g. hundreds of samples, 20–30 different tools, bioinformatics workflow managers come into play to ensure that the procedures can be orchestrated automatically in a fully reproducible manner (see Section 'Workflow managers' for a brief-but-thorough introduction to this topic).

The question of computational resources is another issue that researchers must tackle in order to be effective in their analyses. Computational resources is a catch-all phrase, and has multiple aspects to it, importantly, the number of central processing units (CPUs) and their clock speeds, the amount of random-access memory (RAM) available per CPU and storage type and capacity (hard disk drives/HDDs and/or solid state disks/SSDs). A personal computer (e.g. laptop) with a dual core CPU, 8GB (Gigabytes) of RAM and a 250GB SSD is sufficient for executing a small analysis in R or python. However, it will be grossly insufficient for running a *de novo* transcriptome assembler; for instance, the Trinity assembler (see Section *De novo* transcriptome assembly) can consume upwards of 27GB of RAM during its execution [58]. The runtime—the duration it takes for the tool to finish executing—would also be excessively long with constrained resources; case in point, even with 48 CPU cores and 512GB of RAM, Trinity can take about 6–7 h (or even days) to run [58]. Likewise, storage capacity on the order of at least 1–2TB would be required. Our personal experience indicates tools such as Trinity can routinely consume several 100GBs of both disk space and RAM during execution, and produce output directories that are themselves at least 10–20GB in size. Therefore, researchers must factor in having to acquire computational resources on this order of magnitude for workflows incorporating *de novo* assemblies. Broadly speaking, there are two ways in which such resources can be requisitioned. Depending on the usage frequency, a workstation/server with the necessary capacity may be rented or purchased outright for institutional/departmental use [244]. Very often, research and educational institutions will have their own centralized computational infrastructure (e.g. high performance compute clusters) from which such resources can be requested [244]. Computational resources may also be acquired from national-scale compute infrastructure projects [245, 246], non-profit foundations that offer bioinformatic-as-a-service (e.g. Galaxy [23]; see Section 'Workflow managers') or private cloud compute providers (e.g. Google Cloud Life Sciences, Amazon Web Services and Microsoft Azure).

Links:

Amazon Web Services - https://aws.amazon.com/health/

Bash - https://www.gnu.org/software/bash/

Bioconductor - http://bioconductor.org/

Google Cloud Life Sciences - https://cloud.google.com/life-sciences

Microsoft Azure - https://azure.microsoft.com/en-us/solutions/high-performance-computing/health-and-life-sciences/

Linux - https://www.linux.org/

Python - https://www.python.org/

R - https://www.r-project.org/

Ubuntu - https://ubuntu.com/

Windows Subsystem for Linux - https://docs.microsoft.com/en-us/windows/wsl/about

Tool management

Almost all tools indicated in this publication are available online for download and installation. In a vast majority of the cases, the tools are available via a GitHub or GitLab repository. In some instances, tools may either be found on the author's (e.g. a particular research group) website or on other code repositories such as `SourceForge`. Most tools are accompanied by a descriptive academic publication that also normally indicates where the tool can be found: e.g. the publication for the `rnaSPAdes` assembler [56] cites https://github.com/ablab/spades. The repositories of most tools are also usually easily found via appropriate search engine queries.

Tool installation may be as simple as de-compressing and extracting from an archive (e.g. a `.tar.gz` file), or can be a complicated procedure that requires compilation (ref. https://www.linuxjournal.com/article/216 for an introduction to compilation). Typically both the source code for compilation as well as pre-compiled binaries targeting a few chosen platforms are made available for download by the tool developers.

However, the best method for installing tools today would be via the open-source package manager `Conda`. Almost all major standalone bioinformatics tools are available via the `Bioconda` [243] channel, and installation in most cases is as simple as creating a new `conda` environment and issuing the command `conda install -c bioconda exampletoolname`. Most dependencies (i.e. other tools/software required for operation) are also available via `conda` and should be installed automatically alongside. The `conda` package manager also permits easy updating of installed tools and packages. This is in sharp contrast to a compiled installation where an update would typically require compiling the newly downloaded source code again and also ensuring that all dependencies are also updated without compromising the functionality of the OS.

Some tools are also available as `Docker` and/or `Singularity` containers. A container is basically the software and everything that is needed to run it enclosed into a single unit (see https://www.docker.com/resources/what-container for an explanation). Bioinformatics tools made available as containers are typically those that are either too big to be shipped stand-alone, are too complicated to be installed directly by the user, or a combination of both. These are mostly tools that have a multitude of dependencies (i.e. other software) required to run and/or come with large amounts of bundled data. `Docker` containers require `root privileges` (https://www.ssh.com/academy/iam/user/root) to run while their `Singularity` counterparts normally do not. Root access is typically a no-go in high performance computing (HPC) environments [247], and therefore `Singularity` containers are more popular in that particular context. We direct readers to documentation from `Docker` and `Singularity` for instructions on how to execute containerized software.

Executing a command line tool requires an understanding of the inputs, options and outputs as related to the tool. The first point of contact for help information/documentation is typically the tool itself. Issuing the command `toolname -h`, `toolname -help` or `toolname --help` should print the in-built help page. Documentation can also be found in the included `README` files and often in the 'wiki' sections of the tool repositories. If tools have associated publications, these are also a good source of information and documentation. For more advanced support it may be necessary to either contact the developers/maintainers directly via e-mail or by opening an `issue` on the tool repository's issue tracker. It is inevitable that the researcher will encounter non-specific (but nevertheless important) questions/issues over the course of a bioinformatics analysis. Luckily, there are several popular online communities where such topics could be raised (e.g. `Biostars`).

More general questions can also be addressed to members of the bioinformatics community at large via online forums like Biostars, BioinformaticsStackExchange, BiologyStackExchange and StackOverflow among others.

Links:

`Bioconda` - https://bioconda.github.io/
`Biostars` - https://www.biostars.org/
`Conda` - https://docs.conda.io/en/latest/
`Docker` - https://www.docker.com/
`GitHub` - https://github.com/
`GitLab` - https://gitlab.com/
`Singularity` - https://sylabs.io/singularity/
`SourceForge` - https://sourceforge.net/

What to annotate and where to publish

What to annotate: Sequence annotations should ultimately serve the purposes of the study. Identity assignment via homology could be considered the bare minimum, as it allows the assembled sequences to be tied to human-comprehensible identifiers. GO terms are normally annotated because these can be aggregated to reveal the distribution of the transcriptomic output over various biological aspects (e.g. what percent of the transcriptome is involved in a biological process, etc.). They are also useful for differential expression studies wherein the GO terms of differentially expressed transcripts can be aggregated to obtain an overview of which biological phenomena are being influenced (GO enrichment analysis). It is also useful to annotate functional domains against a standard database such as `Pfam`.

Other annotations can be performed as the need arises. For instance, the objective of the study may be to profile simple sequence repeats in the mRNA alongside establishing a *de novo* transcriptome. In this case, the assembled sequences may be passed through an appropriate tool (e.g. the `MISA` web server [248]) to obtain the necessary annotations in addition to the aforementioned 'standard' annotations. In general, once the assembled sequences have been translated, a relatively broad variety of tools become available, opening up additional avenues for sequence annotation that can be pursued as necessary. The necessary tools are best found by consulting the literature. Continuing with the example above, `MISA` can be found cited in a relevant study such as Pinosio et al. [249].

Where to publish: Typically, an assembly and annotation workflow would result in at least one FASTA file containing the assembled sequences, and at least one tabular file (e.g. a TSV file) containing one row per sequence with individual columns representing the various annotations. Almost all studies submit their raw sequencing data (i.e. the FASTQ files) and the assembly to NCBI's Sequence Read Archive (SRA)[250], and Transcriptome Shotgun Assembly Sequence Database (TSA), respectively. In contrast annotation files do not seem to have a standard destination. Some studies prefer to upload data to research data dissemination portals such as figshare and Zenodo [251] that can generate stable Digital Object Identifiers (DOIs) [252] to the data themselves. Annotations can also be submitted to the TSA (see https://www.ncbi.nlm.nih.gov/genbank/tsaguide/), but this is allegedly a cumbersome and tedious process. In some instances, annotation files have been provided alongside the publication as a supplementary file (e.g. *Thunders, Cavanagh and Li* [253]). Depending on the volume of data, creative solutions such as hosting the annotations on the cloud may also be valid solutions.

Links:

Digital Object Identifiers - https://www.doi.org/

figshare - https://figshare.com/

NCBI Sequence Read Archive - https://www.ncbi.nlm.nih.gov/sra

NCBI Transcriptome Shotgun Assembly Sequence Database - https://www.ncbi.nlm.nih.gov/genbank/tsa/

Zenodo - https://zenodo.org/

Conclusions

As *Stark, Grzelak and Hadfield* [7] highlight in their review 'RNA sequencing: the teenage years', RNA-seq has become a ubiquitous tool in biology, and has steadily proliferated into allied fields of research such as ecology [17]. The advent of long-read RNA-seq [254–257] has proffered exciting prospects such as direct sequencing of RNA molecules sans cDNA synthesis [258] and sequencing RNA from single cells [259]. Despite these challenges, bulk RNA-seq via short-read sequencing remains a prominent method. One reason is the sustained (and growing) popularity of *de novo* transcriptome assembly and annotation for the purposes of studying non-model organisms. Here, well-annotated *de novo* assembled transcriptomes represent an inexpensive route for thoroughly cataloging transcripts, and identifying interesting gene products.

This enduring and widespread interest has ensured an unabated deluge of ever-improving tools, databases and workflows to facilitate assembly, annotation and associated analyses. While these utilities have greatly eased the effort of scientific discovery, the staggering variety of resources available has nevertheless made the task of choosing a suitable approach for a specific research question a complex and confusing exercise. This is perhaps especially true for non-expert practitioners who now have the means to perform RNA-seq experiments entirely in-house. For these individuals, *de novo* transcriptomics holds great promise as they can now study nearly any organism(s) of their choosing.

To this end, we presented a comprehensive and beginner-friendly overview of the major processes and tools involved in *de novo* transcriptome assembly and annotation of short-read bulk RNA-seq data. We hope that this material will aid both incoming and established researchers alike in their quest to obtain high-quality transcriptomes.

Key Points

- *De novo* transcriptome assembly and annotation ideal for studying non-model organisms and establishing gene catalogs thereof.
- In-housing marred by overabundance of tools, paucity of authoritative literature and non-standardized workflows.
- We present a comprehensive-but-beginner-friendly step-by-step review featuring accessible conceptual explanations and an overview of popular tools.

Acknowledgments

We would like to thank Dr. Brian J. Haas (The Broad Institute, USA) and Dr. Johannes Söding (Max Planck Institute for Biophysical Chemistry, Germany) for valuable discussions, and for providing critical feedback on the manuscript. We also gratefully acknowledge helpful feedback from Abrar Aljahani (Max Planck Institute for Biophysical Chemistry, Germany), Magdalena Karpinska (Max Planck Institute for Biophysical Chemistry, Germany), Ruoshi Zhang (Max Planck Institute for Biophysical Chemistry, Germany), and Yajie Zhu (Universitäts Medizin Göttingen, Germany). F.M. thanks the SPP DECRyPT 2125 funding program for covering his salary. We also gratefully acknowledge Matt Crook, whose bacterium pictogram (http://phylopic.org/name/4fc5abf4-3c1a-4edd-bec4-58bf6382ad00) was used in Figure 2: Contaminant removal (Creative Common license https://creativecommons.org/licenses/by-sa/3.0/).

Author contributions statement

V.R. and L.K. conceived the idea for the manuscript. L.R. contributed the section on workflow managers and to the section on Computational and programmatic considerations, and F.M. contributed the sections on differential expression analysis and comparing transcriptome assemblies. V.R. and L.K. contributed all other sections. All authors contributed to proofreading and correcting the manuscript.

References

1. Buccitelli C, Selbach M. mRNAs, proteins and the emerging principles of gene expression control. *Nat Rev Genet* October 2020;**21**(10):630–44.
2. Schimmel P. The emerging complexity of the tRNA world: mammalian tRNAs beyond protein synthesis. *Nat Rev Mol Cell Biol* January 2018;**19**(1):45–58.
3. Statello L, Guo C-J, Chen L-L, *et al.* Gene regulation by long non-coding RNAs and its biological functions. *Nat Rev Mol Cell Biol* February 2021;**22**(2):96–118.
4. Holoch D, Moazed D. RNA-mediated epigenetic regulation of gene expression. *Nat Rev Genet* February 2015;**16**(2):71–84.
5. Li J, Liu C. Coding or noncoding, the converging concepts of RNAs. *Front Genet* May 2019;**10**:496.
6. Slatko BE, Gardner AF, Ausubel FM. Overview of next-generation sequencing technologies. *Curr Protoc Mol Biol* April 2018;**122**(1):e59.
7. Stark R, Grzelak M, Hadfield J. RNA sequencing: the teenage years. *Nat Rev Genet* November 2019;**20**(11):631–56.
8. Wang Z, Gerstein M, Snyder M. RNA-Seq: a revolutionary tool for transcriptomics. *Nat Rev Genet* January 2009;**10**(1):57–63.
9. Mantione KJ, Kream RM, Kuzelova H, *et al.* Comparing bioinformatic gene expression profiling methods: microarray and RNA-Seq. *Med Sci Monit Basic Res* August 2014;**20**:138–42.
10. Han Y, Gao S, Muegge K, *et al.* Advanced applications of RNA sequencing and challenges. *Bioinform Biol Insights* November 2015;**9**(Suppl 1):29–46.
11. Chen G, Ning B, Shi T. Single-cell RNA-seq technologies and related computational data analysis. *Front Genet* April 2019;**10**:317.
12. Kukurba KR, Montgomery SB. RNA sequencing and analysis. *Cold Spring Harb Protoc* April 2015;**2015**(11):951–69.
13. Salzberg SL. Next-generation genome annotation: we still struggle to get it right. *Genome Biol* May 2019;**20**(1):92.
14. Hrdlickova R, Toloue M, Tian B. RNA-Seq methods for transcriptome analysis. *Wiley Interdiscip Rev RNA* January 2017;**8**(1).
15. Martin JA, Wang Z. Next-generation transcriptome assembly. *Nat Rev Genet* September 2011;**12**(10):671–82.
16. Peona V, Weissensteiner MH, Suh A. How complete are "complete" genome assemblies?-an avian perspective. *Mol Ecol Resour* November 2018;**18**(6):1188–95.
17. Todd EV, Black MA, Gemmell NJ. The power and promise of RNA-seq in ecology and evolution. *Mol Ecol* March 2016;**25**(6):1224–41.
18. Sneha Asai, Remo Sanges, Chiara Lauritano, Penelope K Lindeque, Francesco Esposito, Adrianna Ianora, and Ylenia Carotenuto. DE novo transcriptome assembly and gene expression profiling of the copepod calanus helgolandicus feeding on the PUA-producing diatom skeletonema marinoi. *Mar Drugs*, **18**(8):392, July 2020.
19. Moreno-Santillán DD, Machain-Williams C, Hernández-Montes G, *et al.* De novo transcriptome assembly and functional annotation in five species of bats. *Sci Rep* April 2019;**9**(1): 6222.
20. Chabikwa TG, Barbier FF, Tanurdzic M, *et al.* De novo transcriptome assembly and annotation for gene discovery in avocado, macadamia and mango. *Sci Data* January 2020; **7**(1):9.
21. Rosen R, Lebedev G, Kontsedalov S, *et al.* A de novo transcriptomics approach reveals genes involved in thrips tabaci resistance to spinosad. *Insects* January 2021;**12**(1):67.
22. Alvarez RV, Mariño-Ramírez L, Landsman D. Transcriptome annotation in the cloud: complexity, best practices, and cost. *Gigascience* January 2021;**10**(2).
23. Afgan E, Baker D, Batut B, *et al.* The galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. *Nucleic Acids Res* July 2018;**46**(W1): W537–44.
24. Carruthers M, Yurchenko AA, Augley JJ, *et al.* De novo transcriptome assembly, annotation and comparison of four ecological and evolutionary model salmonid fish species. *BMC Genomics* January 2018;**19**(1):32.
25. Stoler N, Nekrutenko A. Sequencing error profiles of illumina sequencing instruments. *NAR Genom Bioinform* March 2021;**3**(1):lqab019.
26. Garcia TI, Shen Y, Catchen J, *et al.* Effects of short read quality and quantity on a de novo vertebrate transcriptome assembly. *Comp Biochem Physiol C Toxicol Pharmacol* January 2012;**155**(1):95–101.
27. de Sena Brandine G, Smith AD. Falco: high-speed FastQC emulation for quality control of sequencing data. *F1000Res* November 2019;**8**:1874.
28. Ewels P, Magnusson M, Lundin S, *et al.* MultiQC: summarize analysis results for multiple tools and samples in a single report. *Bioinformatics* October 2016;**32**(19):3047–8.
29. Song L, Florea L. Rcorrector: efficient and accurate error correction for illumina RNA-seq reads. *Gigascience* October 2015;**4**(1):48.
30. Martin M. Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet J* May 2011;**17**(1):10.
31. Bushnell B, Rood J, Singer E. BBMerge – accurate paired shotgun read merging via overlap. *PLoS One* October 2017;**12**(10):e0185056.
32. Ewing B, Green P. Base-calling of automated sequencer traces using phred. II. Error probabilities. *Genome Res* March 1998;**8**(3):186–94.
33. Chen S, Zhou Y, Chen Y, *et al.* fastp: an ultra-fast all-in-one FASTQ preprocessor. *Bioinformatics* September 2018;**34**(17):i884–90.
34. Bolger AM, Lohse M, Usadel B. Trimmomatic: a flexible trimmer for illumina sequence data. *Bioinformatics* August 2014;**30**(15):2114–20.
35. Wood DE, Lu J, Langmead B. Improved metagenomic analysis with kraken 2. *Genome Biol* November 2019;**20**(1):257.
36. Kim D, Song L, Breitwieser FP, *et al.* Centrifuge: rapid and sensitive classification of metagenomic sequences. *Genome Res* December 2016;**26**(12):1721–9.
37. Zhao S, Zhang Y, Gamini R, *et al.* Evaluation of two main RNA-seq approaches for gene quantification in clinical RNA sequencing: polya+ selection versus rRNA depletion. *Sci Rep* December 2018;**8**(1).
38. Li X, Nair A, Wang S, *et al.* Quality control of RNA-seq experiments. In: *RNA Bioinformatics*, Vol. **1269**. New York: Springer, 2015, 137–46.
39. Morlan JD, Qu K, Sinicropi DV. Selective depletion of rRNA enables whole transcriptome profiling of archival fixed tissue. *PLoS One* August 2012;**7**(8):e42882.
40. Kopylova E, Noé L, Touzet H. SortMeRNA: fast and accurate filtering of ribosomal RNAs in metatranscriptomic data. *Bioinformatics* December 2012;**28**(24):3211–7.
41. Kalvari I, Nawrocki EP, Ontiveros-Palacios N, *et al.* Rfam 14: expanded coverage of metagenomic, viral and microRNA families. *Nucleic Acids Res* January 2021;**49**(D1):D192–200.

42. Quast C, Pruesse E, Yilmaz P, *et al.* The SILVA ribosomal RNA gene database project: improved data processing and web-based tools. *Nucleic Acids Res* January 2013;**41**(Database issue):D590–6.
43. Wang Y, Ghaffari N, Johnson CD, *et al.* Evaluation of the coverage and depth of transcriptome by RNA-Seq in chickens. *BMC Bioinformatics* October 2011;**12**Suppl 10(S10):S5.
44. Tarazona S, García-Alcalde F, Dopazo J, *et al.* Differential expression in RNA-seq: a matter of depth. *Genome Res* December 2011;**21**(12):2213–23.
45. Haas BJ, Papanicolaou A, Yassour M, *et al.* De novo transcript sequence reconstruction from RNA-seq using the trinity platform for reference generation and analysis. *Nat Protoc* August 2013;**8**(8):1494–512.
46. Grabherr MG, Haas BJ, Yassour M, *et al.* Full-length transcriptome assembly from RNA-Seq data without a reference genome. *Nat Biotechnol* May 2011;**29**(7):644–52.
47. Crusoe MR, Alameldin HF, Awad S, *et al.* The khmer software package: enabling efficient nucleotide sequence analysis. *F1000Res* September 2015;**4**:900.
48. Wedemeyer A, Kliemann L, Srivastav A, *et al.* An improved filtering algorithm for big read datasets and its application to single-cell assembly. *BMC Bioinformatics* July 2017;**18**(1):324.
49. McCorrison JM, Venepally P, Singh I, *et al.* NeatFreq: reference-free data reduction and coverage normalization for de novo sequence assembly. *BMC Bioinformatics* November 2014;**15**(1):357.
50. Durai DA, Schulz MH. Improving in-silico normalization using read weights. *Sci Rep* March 2019;**9**(1):5133.
51. Cavallaro M, Walsh MD, Jones M, *et al.* 3 '-5 ' crosstalk contributes to transcriptional bursting. *Genome Biol* February 2021;**22**(1):56.
52. Struhl K. Transcriptional noise and the fidelity of initiation by RNA polymerase II. *Nat Struct Mol Biol* February 2007;**14**(2):103–5.
53. Hansen KD, Brenner SE, Dudoit S. Biases in illumina transcriptome sequencing caused by random hexamer priming. *Nucleic Acids Res* July 2010;**38**(12):e131.
54. Ozsolak F, Milos PM. RNA sequencing: advances, challenges and opportunities. *Nat Rev Genet* February 2011;**12**(2):87–98.
55. Canzar S, Andreotti S, Weese D, *et al.* CIDANE: comprehensive isoform discovery and abundance estimation. *Genome Biol* January 2016;**17**(1):16.
56. Bushmanova E, Antipov D, Lapidus A, *et al.* rnaSPAdes: a de novo transcriptome assembler and its application to RNA-Seq data. *Gigascience* September 2019;**8**(9).
57. Liu J, Li G, Zheng C, *et al.* BinPacker: packing-based DE novo transcriptome assembly from RNA-seq data. *PLoS Comput Biol* February 2016;**12**(2):e1004772.
58. Hölzer M, Marz M. De novo transcriptome assembly: a comprehensive cross-species comparison of short-read RNA-Seq assemblers. *Gigascience* May 2019;**8**(5).
59. Zhang Y, Qian J, Chunyan G, *et al.* Alternative splicing and cancer: a systematic review. *Signal Transduct Target Ther* February 2021;**6**(1):78.
60. McManus CJ, Graveley BR. RNA structure and the mechanisms of alternative splicing. *Curr Opin Genet Dev* August 2011;**21**(4):373–9.
61. Freedman AH, Clamp M, Sackton TB. Error, noise and bias in de novo transcriptome assemblies. *Mol Ecol Resour* January 2021;**21**(1):18–29.
62. Davidson NM, Oshlack A. Corset: enabling differential gene expression analysis for de novo assembled transcriptomes. *Genome Biol* July 2014;**15**(7):410.
63. Xie Y, Wu G, Tang J, *et al.* SOAPdenovo-trans: de novo transcriptome assembly with short RNA-Seq reads. *Bioinformatics* June 2014;**30**(12):1660–6.
64. Schulz MH, Zerbino DR, Vingron M, *et al.* Oases: robust de novo RNA-seq assembly across the dynamic range of expression levels. *Bioinformatics* April 2012;**28**(8):1086–92.
65. Robertson G, Schein J, Chiu R, *et al.* De novo assembly and analysis of RNA-seq data. *Nat Methods* November 2010;**7**(11):909–12.
66. Yu P, Leung HCM, Yiu S-M, *et al.* IDBA-Tran: a more robust de novo de bruijn graph assembler for transcriptomes with uneven expression levels. *Bioinformatics* July 2013;**29**(13):i326–34.
67. Nip KM, Chiu R, Yang C, *et al.* RNA-bloom enables reference-free and reference-guided sequence assembly for single-cell transcriptomes. *Genome Res* August 2020;**30**(8):1191–200.
68. Zhao J, Feng H, Zhu D, *et al.* DTA-SiST: de novo transcriptome assembly by using simplified suffix trees. *BMC Bioinformatics* December 2019;**20**(Suppl 25):698.
69. Heber S, Alekseyev M, Sze S-H, *et al.* Splicing graphs and EST assembly problem. *Bioinformatics* 2002;**18**(Suppl 1):S181–8.
70. Zhao J, Feng H, Zhu D, *et al.* IsoTree: a new framework for de novo transcriptome assembly from RNA-seq reads. *IEEE/ACM Trans Comput Biol Bioinform* May 2020;**17**(3):938–48.
71. Chang Z, Li G, Liu J, *et al.* Bridger: a new framework for de novo transcriptome assembly using RNA-seq data. *Genome Biol* February 2015;**16**(1):30.
72. Liu J, Yu T, Zengchao M, *et al.* TransLiG: a de novo transcriptome assembler that uses line graph iteration. *Genome Biol* April 2019;**20**(1):81.
73. Mühr LSA, Lagheden C, Hassan SS, *et al.* De novo sequence assembly requires bioinformatic checking of chimeric sequences. *PLoS One* August 2020;**15**(8):e0237455.
74. Shen W, Le S, Li Y, *et al.* SeqKit: a cross-platform and ultrafast toolkit for FASTA/Q file manipulation. *PLoS One* October 2016;**11**(10):e0163962.
75. Bryant DM, Johnson K, DiTommaso T, *et al.* A tissue-mapped axolotl DE novo transcriptome enables identification of limb regeneration factors. *Cell Rep* January 2017;**18**(3):762–76.
76. International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature* February 2001;**409**(6822):860–921.
77. Seppey M, Manni M, Zdobnov EM. BUSCO: Assessing genome assembly and annotation completeness. In: *Gene Prediction*, Vol. **1962**. New York: Springer, 2019, 227–45.
78. Zdobnov EM, Kuznetsov D, Tegenfeldt F, *et al.* OrthoDB in 2020: evolutionary and functional annotations of orthologs. *Nucleic Acids Res* January 2021;**49**(D1):D389–93.
79. Dohmen E, Kremer LPM, Bornberg-Bauer E, *et al.* DOGMA: domain-based transcriptome and proteome quality assessment. *Bioinformatics* September 2016;**32**(17):2577–81.
80. Smith-Unna R, Boursnell C, Patro R, *et al.* TransRate: reference-free quality assessment of de novo transcriptome assemblies. *Genome Res* August 2016;**26**(8):1134–44.
81. Smith-Unna R, Boursnell C, Patro R, *et al.* TransRate: reference-free quality assessment of de novo transcriptome assemblies. *Genome Res* August 2016;**26**(8):1134–44.
82. Li B, Fillmore N, Bai Y, *et al.* Evaluation of de novo transcriptome assemblies from RNA-Seq data. *Genome Biol* December 2014;**15**(12):553.

83. Bushmanova E, Antipov D, Lapidus A, *et al.* rnaQUAST: a quality assessment tool forde novotranscriptome assemblies: table 1. *Bioinformatics* July 2016;**32**(14):2210–2.
84. Ceschin DG, Pires NS, Mardirosian MN, *et al.* The rhinella arenarum transcriptome: de novo assembly, annotation and gene prediction. *Sci Rep* January 2020;**10**(1):1053.
85. Kerkvliet J, de Fouchier A, van Wijk M, *et al.* The bellerophon pipeline, improving de novo transcriptomes and removing chimeras. *Ecol Evol* September 2019;**9**(18):10513–21.
86. Limin F, Niu B, Zhu Z, *et al.* CD-HIT: accelerated for clustering the next-generation sequencing data. *Bioinformatics* December 2012;**28**(23):3150–2.
87. Cabau C, Escudié F, Djari A, *et al.* Compacting and correcting trinity and oases RNA-Seq de novo assemblies. *PeerJ* February 2017;**5**(e2988):e2988.
88. MacManes MD. The oyster river protocol: a multi-assembler and kmer approach for de novo transcriptome assembly. *PeerJ* August 2018;**6**:e5428.
89. Rivera-Vicéns RE, Garcia-Escudero CA, Conci N, *et al.* TransPi – a comprehensive TRanscriptome ANalysiS PIpeline for de novo transcriptome assemblybioRxiv. February 2021.
90. Ortiz R, Gera P, Rivera C, *et al.* Pincho: a modular approach to high quality DE novo transcriptomics. *Genes (Basel)* June 2021;**12**(7):953.
91. Conesa A, Madrigal P, Tarazona S, *et al.* A survey of best practices for RNA-seq data analysis. *Genome Biol* January 2016;**17**:13.
92. Alvarez RV, Pongor LS, Mariño-Ramírez L, *et al.* TPMCalculator: one-step software to quantify mRNA abundance of genomic features. *Bioinformatics* June 2019;**35**(11):1960–2.
93. Langmead B, Salzberg SL. Fast gapped-read alignment with bowtie 2. *Nat Methods* March 2012;**9**(4):357–9.
94. Dobin A, Davis CA, Schlesinger F, *et al.* STAR: ultrafast universal RNA-seq aligner. *Bioinformatics* January 2013;**29**(1):15–21.
95. Li H, Handsaker B, Wysoker A, *et al.* The sequence alignment/map format and SAMtools. *Bioinformatics* August 2009;**25**(16):2078–9.
96. Li B, Dewey CN. RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. *BMC Bioinformatics* August 2011;**12**(1):323.
97. Bray NL, Pimentel H, Melsted P, *et al.* Near-optimal probabilistic RNA-seq quantification. *Nat Biotechnol* May 2016;**34**(5):525–7.
98. Patro R, Duggal G, Love MI, *et al.* Salmon provides fast and bias-aware quantification of transcript expression. *Nat Methods* April 2017;**14**(4):417–9.
99. Zhang C, Zhang B, Lin L-L, *et al.* Evaluation and comparison of computational tools for RNA-seq isoform quantification. *BMC Genomics* December 2017;**18**(1).
100. Everaert C, Luypaert M, Maag JLV, *et al.* Benchmarking of RNA-sequencing analysis workflows using whole-transcriptome RT-qPCR expression data. *Sci Rep* May 2017;**7**(1):1559.
101. Schaarschmidt S, Fischer A, Zuther E, *et al.* Evaluation of seven different RNA-Seq alignment tools based on experimental data from the model plant arabidopsis thaliana. *Int J Mol Sci* March 2020;**21**(5):1720.
102. Wu DC, Yao J, Ho KS, *et al.* Limitations of alignment-free tools in total RNA-seq quantification. *BMC Genomics* December 2018;**19**(1).
103. Nowoshilow S, Schloissnig S, Fei J-F, *et al.* The axolotl genome and the evolution of key tissue formation regulators. *Nature* February 2018;**554**(7690):50–5.
104. The ENCODE Project Consortium. An integrated encyclopedia of DNA elements in the human genome. *Nature* September 2012;**489**(7414):57–74.
105. Hangauer MJ, Vaughn IW, McManus MT. Pervasive transcription of the human genome produces thousands of previously unidentified long intergenic noncoding RNAs. *PLoS Genet* June 2013;**9**(6):e1003569.
106. Zhao S. Alternative splicing, RNA-seq and drug discovery. *Drug Discov Today* June 2019;**24**(6):1258–67.
107. Li W, Godzik A. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics* July 2006;**22**(13):1658–9.
108. Steinegger M, Söding J. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat Biotechnol* November 2017;**35**(11):1026–8.
109. Steinegger M, Söding J. Clustering huge protein sequence sets in linear time. *Nat Commun* December 2018;**9**(1).
110. Mirdita M, Steinegger M, Söding J. MMseqs2 desktop and local web server app for fast, interactive sequence searches. *Bioinformatics* August 2019;**35**(16):2856–8.
111. Mirdita M, Steinegger M, Breitwieser F, *et al.* Fast and sensitive taxonomic assignment to metagenomic contigs. *Bioinformatics* March 2021;**37**(18):3029–31.
112. Malik L, Almodaresi F, Patro R. Grouper: graph-based clustering and annotation for improved de novo transcriptome analysis. *Bioinformatics* October 2018;**34**(19):3265–72.
113. Razo-Mendivil FG, Martínez O, Hayano-Kanashiro C. Compacta: a fast contig clustering tool for de novo assembled transcriptomes. *BMC Genomics* February 2020;**21**(1): 148.
114. Davidson NM, Hawkins ADK, Oshlack A. SuperTranscripts: a data driven reference for analysis and visualisation of transcriptomes. *Genome Biol* December 2017;**18**(1).
115. Oshlack A, Robinson MD, Young MD. From RNA-seq reads to differential expression results. *Genome Biol* December 2010;**11**(12):220.
116. Zyprych-Walczak J, Szabelska A, Handschuh L, *et al.* The impact of normalization methods on RNA-seq data analysis. *Biomed Res Int* 2015, June 2015;621690.
117. Wilfinger WW, Miller R, Eghbalnia HR, *et al.* Strategies for detecting and identifying biological signals amidst the variation commonly found in RNA sequencing data. *BMC Genomics* May 2021;**22**(1):322.
118. Zhu A, Ibrahim JG, Love MI. Heavy-tailed prior distributions for sequence count data: removing the noise and preserving large differences. *Bioinformatics* 2019;**35**.
119. Stephens M. False discovery rates: a new deal. *Biostatistics* 2017;**18**.
120. R Core Team. R: a language and environment for statistical computing. 2021.
121. Love MI, Huber W, Anders S. Moderated estimation of fold change and dispersion for rna-seq data with deseq2. *Genome Biol* 2014;**15**.
122. Robinson MD, McCarthy DJ, Smyth GK. Edger: a bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics* 2010;**26**.
123. Ritchie ME, Phipson B, Wu DI, *et al.* Limma powers differential expression analyses for rna-sequencing and microarray studies. *Nucleic Acids Res* 2015;**43**.
124. McDermaid A, Monier B, Zhao J, *et al.* Interpretation of differential gene expression results of RNA-seq data: review and integration. *Brief Bioinform* November 2019;**20**(6): 2044–54.
125. Shahjaman M, Akter H, Rashid MM, *et al.* Robust and efficient identification of biomarkers from rna-seq data using median control chart. *F1000Research* 2019;**8**.

126. Love MI, Soneson C, Robinson MD. Importing transcript abundance datasets with tximport. *Dim Txi Inf Rep Sample1* 2017;**1**.
127. Risso D, Ngai J, Speed TP, *et al.* Normalization of RNA-seq data using factor analysis of control genes or samples. *Nat Biotechnol* September 2014;**32**(9):896–902.
128. Varet H, Brillet-Guéguen L, Coppée J-Y, *et al.* SARTools: a DESeq2- and EdgeR-based R pipeline for comprehensive differential analysis of RNA-Seq data. *PLoS One* June 2016;**11**(6):e0157022.
129. Wu G, Anafi RC, Hughes ME, *et al.* MetaCycle: an integrated R package to evaluate periodicity in large scale data. *Bioinformatics* November 2016;**32**(21):3351–3.
130. Vera-Khlara SO, Li RW. Temporal dynamic methods for bulk RNA-Seq time series data. *Genes (Basel)* February 2021;**12**(3):352.
131. Waardenberg AJ, Field MA. consensusDE: an R package for assessing consensus of multiple RNA-seq algorithms with RUV correction. *PeerJ* December 2019;**7**:e8206.
132. Van den Berge K, Hembach KM, Soneson C, *et al.* RNA sequencing data: Hitchhiker's guide to expression analysis. *Annu Rev Biomed Data Sci* July 2019;**2**(1):139–73.
133. Schurch NJ, Schofield P, Gierliński M, *et al.* How many biological replicates are needed in an RNA-seq experiment and which differential expression tool should you use? *RNA* June 2016;**22**(6):839–51.
134. Finotello F, Di Camillo B. Measuring differential gene expression with RNA-seq: challenges and strategies for data analysis. *Brief Funct Genomics* March 2015;**14**(2):130–42.
135. Li WV, Li JJ. Modeling and analysis of RNA-seq data: a review from a statistical perspective. *Quant Biol* September 2018;**6**(3):195–209.
136. Guo Y, Zhao S, Sheng Q, *et al.* RNAseq by total RNA library identifies additional RNAs compared to poly(a) RNA library. *Biomed Res Int* 2015, October 2015;862130.
137. Kang Y-J, Yang D-C, Kong L, *et al.* CPC2: a fast and accurate coding potential calculator based on sequence intrinsic features. *Nucleic Acids Res* July 2017;**45**(W1):W12–6.
138. Wang L, Park HJ, Dasari S, *et al.* CPAT: coding-potential assessment tool using an alignment-free logistic regression model. *Nucleic Acids Res* April 2013;**41**(6):e74.
139. Nawrocki EP, Eddy SR. Infernal 1.1: 100-fold faster RNA homology searches. *Bioinformatics* November 2013;**29**(22):2933–5.
140. O'Leary NA, Wright MW, Brister JR, *et al.* Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation. *Nucleic Acids Res* January 2016;**44**(D1):D733–45.
141. Lagesen K, Hallin P, Rødland EA, *et al.* RNAmmer: consistent and rapid annotation of ribosomal RNA genes. *Nucleic Acids Res* April 2007;**35**(9):3100–8.
142. Kapranov P, Cheng J, Dike S, *et al.* RNA maps reveal new RNA classes and a possible function for pervasive transcription. *Science* June 2007;**316**(5830):1484–8.
143. Amaral PP, Dinger ME, Mattick JS. Non-coding RNAs in homeostasis, disease and stress responses: an evolutionary perspective. *Brief Funct Genomics* May 2013;**12**(3):254–78.
144. Motheramgari K, Curell RV-B, Tzani I, *et al.* Expanding the chinese hamster ovary cell long noncoding RNA transcriptome using RNASeq. *Biotechnol Bioeng* July 2020;**117**(10):3224–31.
145. Kashyap A, Rhodes A, Kronmiller B, *et al.* Pan-tissue transcriptome analysis of long noncoding RNAs in the american beaver castor canadensis. *BMC Genomics* February 2020;**21**(1):153.
146. Nachtigall PG, Kashiwabara AY, Durham AM. CodAn: predictive models for precise identification of coding regions in eukaryotic transcripts. *Brief Bioinform* May 2021;**22**(3).
147. Hyatt D, Chen G-L, Locascio PF, *et al.* Prodigal: prokaryotic gene recognition and translation initiation site identification. *BMC Bioinformatics* March 2010;**11**(1):119.
148. Tang S, Lomsadze A, Borodovsky M. Identification of protein coding regions in RNA transcripts. *Nucleic Acids Res* July 2015;**43**(12):e78.
149. Signal B, Kahlke T. Borf: improved ORF prediction in de-novo assembled transcriptome annotationbioRxiv. April 2021.
150. Madeira F, Park YM, Lee J, *et al.* The EMBL-EBI search and sequence analysis tools APIs in 2019. *Nucleic Acids Res* July 2019;**47**(W1):W636–41.
151. Eddy SR. Accelerated profile HMM searches. *PLoS Comput Biol* October 2011;**7**(10):e1002195.
152. Steinegger M, Mirdita M, Söding J. Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold. *Nat Methods* July 2019;**16**(7):603–6.
153. Koonin EV, Galperin MY. *Sequence - Evolution - Function: Computational Approaches in Comparative Genomics*. Kluwer Academic, 2003.
154. Pearson WR. An introduction to sequence similarity ("homology") searching. *Curr Protoc Bioinformatics* June 2013;**Chapter 3**(1):Unit3.1.
155. Sayadi A, Immonen E, Bayram H, *et al.* The de novo transcriptome and its functional annotation in the seed beetle callosobruchus maculatus. *PLoS One* July 2016;**11**(7):e0158565.
156. Pearson WR. BLAST and FASTA similarity searching for multiple sequence alignment. In: *Multiple Sequence Alignment Methods*, Vol. **1079**. Humana Press, 2014, 75–101.
157. Punta M, Ofran Y. The rough guide to in silico function prediction, or how to use sequence and structure information to predict protein function. *PLoS Comput Biol* October 2008;**4**(10):e1000160.
158. Altschul SF, Gish W, Miller W, *et al.* Basic local alignment search tool. *J Mol Biol* October 1990;**215**(3):403–10.
159. Camacho C, Coulouris G, Avagyan V, *et al.* BLAST+: architecture and applications. *BMC Bioinformatics* December 2009;**10**(1):421.
160. Buchfink B, Reuter K, Drost H-G. Sensitive protein alignments at tree-of-life scale using DIAMOND. *Nat Methods* April 2021;**18**(4):366–8.
161. NCBI Resource Coordinators. Database resources of the national center for biotechnology information. *Nucleic Acids Res* January 2018;**46**(D1):D8–13.
162. UniProt Consortium. UniProt: the universal protein knowledgebase in 2021. *Nucleic Acids Res* January 2021;**49**(D1):D480–9.
163. Suzek BE, Wang Y, Huang H, *et al.* UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics* March 2015;**31**(6):926–32.
164. Suzek BE, Huang H, McGarvey P, *et al.* UniRef: comprehensive and non-redundant UniProt reference clusters. *Bioinformatics* May 2007;**23**(10):1282–8.
165. Larkin A, Marygold SJ, Antonazzo G, *et al.* FlyBase: updates to the drosophila melanogaster knowledge base. *Nucleic Acids Res* January 2021;**49**(D1):D899–907.
166. Harris TW, Arnaboldi V, Cain S, *et al.* WormBase: a modern model organism information resource. *Nucleic Acids Res* January 2020;**48**(D1):D762–7.
167. Bel, Diels T, Vancaester E, *et al.* PLAZA 4.0: an integrative resource for functional, evolutionary and comparative plant genomics. *Nucleic Acids Res* January 2018;**46**(D1):D1190–6.
168. Vandepoele K, Van Bel M, Richard G, *et al.* Pico-PLAZA, a genome database of microbial photosynthetic eukaryotes. *Environ Microbiol* August 2013;**15**(8):2147–53.

169. Martin Gollery. *Handbook of Hidden Markov Models in Bioinformatics*. Chapman & Hall/CRC Mathematical and Computational Biology Series. CRC Press, 2008.
170. Gribskov M, McLachlan AD, Eisenberg D. Profile analysis: detection of distantly related proteins. *Proc Natl Acad Sci U S A* July 1987;**84**(13):4355–8.
171. Eddy SR. Profile hidden markov models. *Bioinformatics* 1998;**14**(9):755–63.
172. Chatzou M, Magis C, Chang J-M, *et al.* Multiple sequence alignment modeling: methods and applications. *Brief Bioinform* November 2016;**17**(6):1009–23.
173. Armenteros JJA, Tsirigos KD, Sønderby CK, *et al.* SignalP 5.0 improves signal peptide predictions using deep neural networks. *Nat Biotechnol* April 2019;**37**(4):420–3.
174. Harrison PM. fLPS: fast discovery of compositional biases for the protein universe. *BMC Bioinformatics* November 2017;**18**(1):476.
175. Van Roey K, Uyar B, Weatheritt RJ, *et al.* Short linear motifs: ubiquitous and functionally diverse protein interaction modules directing cell regulation. *Chem Rev* July 2014;**114**(13):6733–78.
176. Jones P, Binns D, Chang H-Y, *et al.* InterProScan 5: genome-scale protein function classification. *Bioinformatics* May 2014;**30**(9):1236–40.
177. Mistry J, Chuguransky S, Williams L, *et al.* Pfam: the protein families database in 2021. *Nucleic Acids Res* January 2021;**49**(D1):D412–9.
178. Sillitoe I, Bordin N, Dawson N, *et al.* CATH: increased structural coverage of functional space. *Nucleic Acids Res* January 2021;**49**(D1):D266–73.
179. Lewis TE, Sillitoe I, Dawson N, *et al.* Gene3D: extensive prediction of globular domains in proteins. *Nucleic Acids Res* January 2018;**46**(D1):D435–9.
180. Gene Ontology Consortium. The gene ontology resource: enriching a GOld mine. *Nucleic Acids Res* January 2021;**49**(D1):D325–34.
181. Ashburner M, Ball CA, Blake JA, *et al.* Gene ontology: tool for the unification of biology. The gene ontology consortium. *Nat Genet* May 2000;**25**(1):25–9.
182. Christophe Dessimoz and Nives Skunca, editors. *The gene ontology handbook*. Methods in molecular biology (Clifton, N.J.). Humana Press, New York, NY, 1 edition, November 2016.
183. Huerta-Cepas J, Forslund K, Coelho LP, *et al.* Fast genome-wide functional annotation through orthology assignment by eggNOG-mapper. *Mol Biol Evol* August 2017;**34**(8): 2115–22.
184. Huerta-Cepas J, Szklarczyk D, Heller D, *et al.* eggNOG 5.0: a hierarchical, functionally and phylogenetically annotated orthology resource based on 5090 organisms and 2502 viruses. *Nucleic Acids Res* January 2019;**47**(D1):D309–14.
185. Altenhoff AM, Train C-M, Gilbert KJ, *et al.* OMA orthology in 2021: website overhaul, conserved isoforms, ancestral gene order and more. *Nucleic Acids Res* D1, January 2021;**49**:D373–9.
186. Götz S, García-Gómez JM, Terol J, *et al.* High-throughput functional annotation and data mining with the Blast2GO suite. *Nucleic Acids Res* June 2008;**36**(10):3420–35.
187. Kanehisa M, Furumichi M, Sato Y, *et al.* KEGG: integrating viruses and cellular organisms. *Nucleic Acids Res* January 2021;**49**(D1):D545–51.
188. Kanehisa M. Toward understanding the origin and evolution of cellular organisms. *Protein Sci* November 2019;**28**(11):1947–51.
189. Kanehisa M. KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res* 2000;**28**.
190. Jassal B, Matthews L, Viteri G, *et al.* The reactome pathway knowledgebase. *Nucleic Acids Res* January 2020;**48**(D1):D498–503.
191. Kanehisa M, Sato Y, Morishima K. BlastKOALA and GhostKOALA: KEGG tools for functional characterization of genome and metagenome sequences. *J Mol Biol* February 2016;**428**(4):726–31.
192. Bryant DM, Johnson K, DiTommaso T, *et al.* A tissue-mapped axolotl DE novo transcriptome enables identification of limb regeneration factors. *Cell Rep* January 2017;**18**(3):762–76.
193. Krogh A, Larsson B, von Heijne G, *et al.* Predicting transmembrane protein topology with a hidden markov model: application to complete genomes. *J Mol Biol* January 2001;**305**(3):567–80.
194. Altenhoff AM, Glover NM, Dessimoz C. Inferring orthology and paralogy. In: *Evolutionary Genomics*, Vol. **1910**. New York: Springer, 2019, 149–75.
195. Altenhoff AM, Studer RA, Robinson-Rechavi M, *et al.* Resolving the ortholog conjecture: orthologs tend to be weakly, but significantly, more similar in function than paralogs. *PLoS Comput Biol* May 2012;**8**(5):e1002514.
196. Cozzetto D, Jones DT. Computational methods for annotation transfers from sequence. In: *The Gene Ontology Handbook*, Vol. **1446**. New York: Springer, 2017, 55–67.
197. Hart AJ, Ginzburg S, Xu MS, *et al.* EnTAP: bringing faster and smarter functional annotation to non-model eukaryotic transcriptomes. *Mol Ecol Resour* March 2020;**20**(2):591–604.
198. Musacchia F, Basu S, Petrosino G, *et al.* Annocript: a flexible pipeline for the annotation of transcriptomes able to identify putative long noncoding RNAs. *Bioinformatics* July 2015;**31**(13):2199–201.
199. Lu S, Wang J, Chitsaz F, *et al.* CDD/SPARCLE: the conserved domain database in 2020. *Nucleic Acids Res* January 2020;**48**(D1):D265–8.
200. Casimiro-Soriguer CS, Muñoz-Mérida A, Pérez-Pulido AJ. Sma3s: a universal tool for easy functional annotation of proteomes and transcriptomes. *Proteomics* June 2017;**17**(12).
201. Mora-Márquez F, Chano V, Vázquez-Poletti JL, *et al.* TOA: a software package for automated functional annotation in non-model plant species. *Mol Ecol Resour* February 2021;**21**(2): 621–36.
202. Van Bel M, Proost S, Van Neste C, *et al.* TRAPID: an efficient online tool for the functional and comparative analysis of de novo RNA-Seq transcriptomes. *Genome Biol* December 2013;**14**(12):R134.
203. François Bucchini, Andrea Del Cortona, Łukasz Kreft, Alexander Botzki, Michiel Van Bel, and Klaas Vandepoele. TRAPID 2.0: a web application for taxonomic and functional analysis of de novo transcriptomes. *Nucleic Acids Res.*, **49**(17):e101, September 2021.
204. Soderlund CA. Transcriptome computational workbench (TCW): analysis of single and comparative transcriptomes. August 2019.
205. Soderlund C, Nelson W, Willer M, *et al.* TCW: transcriptome computational workbench. *PLoS One* July 2013;**8**(7):e69401.
206. Pulido TH, Vlasova A, Di Tommaso P. *guigolab/FA-nf: 0.3.1 release*, 2021.
207. Altenhoff AM, Levy J, Zarowiecki M, *et al.* OMA standalone: orthology inference among public and custom genomes and transcriptomes. *Genome Res* July 2019;**29**(7):1152–63.

208. Wu S, Zhu Z, Fu L, *et al.* WebMGA: a customizable web server for fast metagenomic sequence analysis. *BMC Genomics* September 2011;**12**(1):444.
209. Törönen P, Medlar A, Holm L. PANNZER2: a rapid functional annotation web server. *Nucleic Acids Res* July 2018;**46**(W1):W84–8.
210. Katoh K, Standley DM. Mafft multiple sequence alignment software version 7: improvements in performance and usability. *Mol Biol Evol* 2013;**30**.
211. Deorowicz S, Debudaj-Grabysz A, Gudyś A. Famsa: fast and accurate multiple sequence alignment of huge protein families. *Sci Rep* 2016;**6**.
212. Stamatakis A. Raxml version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics* 2014;**30**.
213. Zhang C, Sayyari E, Mirarab S. Astral-iii: increased scalability and impacts of contracting low support branches. In: *RECOMB international workshop on comparative genomics*. Springer, 2017.
214. Emms DM, Kelly S. Orthofinder: phylogenetic orthology inference for comparative genomics. *Genome Biol* 2019;**20**.
215. Miller JB, Pickett BD, Ridge PG. Justorthologs: a fast, accurate and user-friendly ortholog identification algorithm. *Bioinformatics* 2019;**35**.
216. Spillane JL, LaPolice TM, MacManes MD, *et al.* Signal, bias, and the role of transcriptome assembly quality in phylogenomic inference. *BMC ecology and evolution* 2021;**21**.
217. Leipzig J. A review of bioinformatic pipeline frameworks. *Brief Bioinform* March 2016;bbw020.
218. Perkel JM. Workflow systems turn raw data into scientific knowledge. *Nature* September 2019;**573**(7772):149–50.
219. Conery JS, Catchen JM, Lynch M. Rule-based workflow management for bioinformatics. *VLDB J* September 2005;**14**(3): 318–29.
220. Strozzi F, Janssen R, Wurmus R, *et al.* Scalable workflows and reproducible data analysis for genomics. In: *Evolutionary Genomics*, Vol. **1910**. Springer, New York, 2019, 723–45.
221. Mölder F, Jablonski KP, Letcher B, *et al.* Sustainable data analysis with snakemake. *F1000Res* January 2021;**10**:33.
222. Jackson M, Kavoussanakis K, Wallace EWJ. Using prototyping to choose a bioinformatics workflow management system. *PLoS Comput Biol* February 2021;**17**(2):e1008622.
223. Reiter T, Brooks PT, Irber L, *et al.* Streamlining data-intensive biology with workflow systems. *Gigascience* January 2021; **10**(1).
224. Di Tommaso P, Chatzou M, Floden EW, *et al.* Nextflow enables reproducible computational workflows. *Nat Biotechnol* April 2017;**35**(4):316–9.
225. Peter Amstutz, Michael R Crusoe, Nebojša Tijanić, Brad Chapman, John Chilton, Michael Heuer, Andrey Kartashov, Dan Leehr, Hervé Ménager, Maya Nedeljkovich, Matt Scales, Stian Soiland-Reyes, and Luka Stojanovic. *Common workflow language*, v1.0, July 2016.
226. stackoverflow. *Stack Overflow Developer Survey*, 2020.
227. Köster J, Rahmann S. Snakemake–a scalable bioinformatics workflow engine. *Bioinformatics* October 2012;**28**(19): 2520–2.
228. nextflow. Basic concepts - Nextflow 21.04.1 documentation.
229. Ewels PA, Peltzer A, Fillinger S, *et al.* The nf-core framework for community-curated bioinformatics pipelines. *Nat Biotechnol* March 2020;**38**(3):276–8.
230. Kotliar M, Kartashov AV, Barski A. CWL-airflow: a lightweight pipeline manager supporting common workflow language. *Gigascience* July 2019;**8**(7).
231. Voss K, Van Der Auwera G, Gentry J. Full-stack genomics pipelining with GATK4 + WDL + Cromwell. *ISCB Community Journal* 2017.
232. Landau W. The targets R package: a dynamic make-like function-oriented pipeline toolkit for reproducibility and high-performance computing. *J Open Source Softw* January 2021;**6**(57):2959.
233. Milicchio F, Rose R, Bian J, *et al.* Visual programming for next-generation sequencing data analytics. *BioData Min* April 2016;**9**(1):16.
234. Michael C. Schatz. *The missing graphical user interface for genomics Genome Biol* August 2010;**11**(8):128.
235. Walker MA, Madduri R, Rodriguez A, *et al.* Models and simulations as a service: exploring the use of galaxy for delivering computational models. *Biophys J* March 2016;**110**(5): 1038–43.
236. Blankenberg D, Von Kuster G, Bouvier E, *et al.* Dissemination of scientific software with galaxy ToolShed. *Genome Biol* February 2014;**15**(2):403.
237. Klingström T, Diego R H-D, Collard T, *et al.* Galaksio, a user friendly workflow-centric front end for galaxy. *EMBnet J* November 2017;**23**(0):897.
238. Okonechnikov K, Golosova O, Fursov M, *et al.* Unipro UGENE: a unified bioinformatics toolkit. *Bioinformatics* April 2012;**28**(8):1166–7.
239. Reich M, Liefeld T, Gould J, *et al.* GenePattern 2.0. *Nat Genet* May 2006;**38**(5):500–1.
240. William E. Shotts. *The Linux Command Line: A Complete Introduction*. No Starch Press, second edition edition, 2019.
241. McGrath M. *Linux in Easy Steps*. Easy Steps Limited, 2010.
242. Python Software Foundation. *Python: A dynamic, open source programming language*, 2021.
243. Grüning B, Dale R, Sjödin A, *et al.* Bioconda: sustainable and comprehensive software distribution for the life sciences. *Nat Methods* July 2018;**15**(7):475–6.
244. Courneya J-P, Mayo A. High-performance computing service for bioinformatics and data science. *J Med Libr Assoc* October 2018;**106**(4):494–5.
245. Castrignanò T, Gioiosa S, Flati T, *et al.* ELIXIR-IT HPC@CINECA: high performance computing resources for the bioinformatics community. *BMC Bioinformatics* August 2020;**21**(Suppl 10):352.
246. Lampa S, Dahlö M, Olason PI, *et al.* Lessons learned from implementing a national infrastructure in Sweden for storage and analysis of next-generation sequencing data. *Gigascience* June 2013;**2**(1):9.
247. Peréz-Sánchez H, Fassihi A, Cecilia JM, *et al.* Applications of high performance computing in bioinformatics, computational biology and computational chemistry. In: *Bioinformatics and Biomedical Engineering*, Lecture notes in computer science. Cham: Springer International Publishing, 2015, 527–41.
248. Beier S, Thiel T, Münch T, *et al.* MISA-web: a web server for microsatellite prediction. *Bioinformatics* August 2017;**33**(16):2583–5.
249. Pinosio S, Fratini S, Cannicci S, *et al.* De novo transcriptome assembly for pachygrapsus marmoratus, an intertidal brachyuran crab. *Mar Genomics* February 2021;**55**(100792):100792.
250. Leinonen R, Sugawara H, Shumway M, *et al.* The sequence read archive. *Nucleic Acids Res* January 2011;**39**(Database issue):D19–21.
251. European Organization for Nuclear Research and OpenAIRE. *Zenodo*, 2013.

252. DeRisi S, Kennison R, Twyman N. The what and whys of DOIs. *PLoS Biol* November 2003;**1**(2):E57.
253. Thunders M, Cavanagh J, Li Y. De novo transcriptome assembly, functional annotation and differential gene expression analysis of juvenile and adult e. fetida, a model oligochaete used in ecotoxicological studies. *Biol Res* February 2017; **50**(1):7.
254. Byrne A, Cole C, Volden R, *et al.* Realizing the potential of full-length transcriptome sequencing. *Philos Trans R Soc Lond B Biol Sci* November 2019;**374**(1786):20190097.
255. Amarasinghe SL, Su S, Dong X, *et al.* Opportunities and challenges in long-read sequencing data analysis. *Genome Biol* February 2020;**21**(1).
256. Mikheyev AS, Tin MMY. A first look at the oxford nanopore MinION sequencer. *Mol Ecol Resour* September 2014;**14**(6):1097–102.
257. Eid J, Fehr A, Gray J. Real-time DNA sequencing from single polymerase molecules. *Science* January 2009;**323**(5910):133–8.
258. Soneson C, Yao Y, Bratus-Neuenschwander A, *et al.* A comprehensive examination of nanopore native RNA sequencing for characterization of complex transcriptomes. *Nat Commun* December 2019;**10**(1).
259. Volden R, Palmer T, Byrne A, *et al.* Improving nanopore read accuracy with the R2C2 method enables the sequencing of highly multiplexed full-length single-cell cDNA. *Proc Natl Acad Sci U S A* September 2018;**115**(39):9726–31.