SEDA: a Desktop Tool Suite for FASTA Files Processing

Hugo López-Fernández, Pedro Duque, Noé Vázquez, Florentino Fdez-Riverola, Miguel Reboiro-Jato, Cristina P. Vieira, and Jorge Vieira

Abstract—SEDA (*SEquence DAtaset builder*) is a multiplatform desktop application for the manipulation of FASTA files containing DNA or protein sequences. The convenient graphical user interface gives access to a collection of simple (filtering, sorting, or file reformatting, among others) and advanced (BLAST searching, protein domain annotation, gene annotation, and sequence alignment) utilities not present in similar applications, which eases the work of life science researchers working with DNA and/or protein sequences, especially those who have no programming skills. This paper presents general guidelines on how to build efficient data handling protocols using SEDA, as well as practical examples on how to prepare high-quality datasets for single gene phylogenetic studies, the characterization of protein families, or phylogenomic studies. The user-friendliness of SEDA also relies on two important features: (*i*) the availability of easy-to-install distributable versions and installers of SEDA, including a Docker image for Linux, and (*ii*) the facility with which users can manage large datasets. SEDA is open-source, with GNU General Public License v3.0 license, and publicly available at GitHub (https://github.com/sing-group/seda). SEDA installers and documentation are available at https://www.sing-group.org/seda/.

Index Terms— Sequence analysis, FASTA, Gene annotation

1 Introduction

FASTA is one of the most widely used formats to store DNA and protein sequences. Many public databases use it to provide a large number of datasets, including genomes and their annotations. In order to extract the sets of sequences of interest, operations such as similarity search (BLAST), filtering (e.g. retrieving sequences having specific amino acid motifs or domains), quality control (e.g. removing sequences showing ambiguity codes or that are likely missannotated), merging, or sorting, among others, often need to be performed. Header editing is also an essential step since sequences downloaded from public databases, such as the NCBI assembly database [1], show a header structure that is from what is needed for many analyses. For instance, when showing a phylogenetic tree, the user often wants to show only the accession number of the sequences used, and specific taxonomic information (species and family from which the sequences were obtained, for instance). Therefore, the user may need to delete, replace, or change the order of the information present in the sequence headers as well as add new information, such as species name, which is missing when downloading data from the NCBI assembly database. Despite the apparent simplicity of such manipulations, it is infeasible to carry them out manually in large sets of files, and they require the use of dedicated software tools.

Although there are many command-line interface (CLI) tools for performing the most common manipulations of FASTA files (e.g. format conversion, searching, filtering, splitting, or shuffling, among others) such as SeqKit [2], seqmagick [3], or seqtk [4], many researchers in life sciences do not feel comfortable using them, preferring instead user-friendly graphical-user interfaces (GUI). Indeed, a GUI makes computer operations more intuitive, easier to learn and use, and does not require prior knowledge of any programming language or basic scripting abilities in command-line. It is thus not surprising that software applications that started as CLI applications sometimes evolve into GUI applications (for instance

- *Hugo López Fernández is with ESEI: Escuela Superior de Ingeniería Informática, University of Vigo, Edificio Politécnico, Campus Universitario As Lagoas s/n 32004, Ourense, Spain; CINBIO - Centro de Investigaciones Biomédicas, University of Vigo, Campus Universitario Lagoas-Marcosende, 36310, Vigo, Spain; SING Research Group, Galicia Sur Health Research Institute (IIS Galicia Sur). SERGAS-UVIGO. E-mail: hlfernandez@uvigo.es.*
- *Pedro Duque is with Instituto de Investigação e Inovação em Saúde (I3S), Universidade do Porto, Rua Alfredo Allen, 208, 4200-135 Porto, Portugal; Instituto de Biologia Molecular e Celular (IBMC), Rua Alfredo Allen, 208, 4200-135 Porto, Portugal. E-mail: pedro.duque@i3s.up.pt.*
- *Noé Vázquez is with ESEI: Escuela Superior de Ingeniería Informática, University of Vigo, Edificio Politécnico, Campus Universitario As Lagoas s/n 32004, Ourense, Spain. E-mail: nvazquezg@gmail.com.*
- *Florentino Fdez-Riverola is with ESEI: Escuela Superior de Ingeniería Informática, University of Vigo, Edificio Politécnico, Campus Universitario As Lagoas s/n 32004, Ourense, Spain; CINBIO - Centro de Investigaciones Biomédicas, University of Vigo, Campus Universitario Lagoas-Marcosende, 36310, Vigo, Spain; SING Research Group, Galicia Sur Health Research Institute (IIS Galicia Sur). SERGAS-UVIGO. E-mail: riverola@uvigo.es.*
- *Miguel Reboiro-Jato is with ESEI: Escuela Superior de Ingeniería Informática, University of Vigo, Edificio Politécnico, Campus Universitario As Lagoas s/n 32004, Ourense, Spain; CINBIO - Centro de Investigaciones Biomédicas, University of Vigo, Campus Universitario Lagoas-Marcosende, 36310, Vigo, Spain; SING Research Group, Galicia Sur Health Research Institute (IIS Galicia Sur). SERGAS-UVIGO. E-mail: mrjato@uvigo.es*
- *Cristina P. Vieira is with Instituto de Investigação e Inovação em Saúde (I3S), Universidade do Porto, Rua Alfredo Allen, 208, 4200-135 Porto, Portugal; Instituto de Biologia Molecular e Celular (IBMC), Rua Alfredo Allen, 208, 4200-135 Porto, Portugal. E-mail: cgvieira@ibmc.up.pt.*
- *Jorge Vieira is with Instituto de Investigação e Inovação em Saúde (I3S), Universidade do Porto, Rua Alfredo Allen, 208, 4200-135 Porto, Portugal; Instituto de Biologia Molecular e Celular (IBMC), Rua Alfredo Allen, 208, 4200-135 Porto, Portugal. E-mail: jbvieira@ibmc.up.pt.*

EMBOSS [5] vs. JEMBOSS [6]), nor that the number of GUIs developed in the last years for life sciences has seen significant growth. An interesting example was that of FaBox [7], an online application for processing FASTA sequences. Although it provides a set of useful operations, it is unsuitable for processing large datasets, which may be several gigabytes in size. As a reference, the size of the compressed file containing all animal coding sequences (CDS) present in the RefSeq database is 7.6 GB. If researchers need to use genome assemblies rather than CDS, they must deal with a much larger file. Indeed, the size of the file containing all animal genome assemblies present in the RefSeq database in compressed format is 213.8 GB (July 2020). It is within this scenario that we have developed SEDA, a desktop application with a convenient GUI for manipulating FASTA files in an efficient way.

The main goal of SEDA is to empower life-scientists (ranging from undergraduate students to experienced researchers) with no training or experience in the use of CLI-based applications and/or no programming skills to process and analyze big datasets of DNA or protein sequences in a batch and automated mode, thus fostering the use of all available data rather than a small subset that can be handled by hand. In addition, SEDA's operations go beyond common manipulations (e.g. sequence filtering, sorting, or file reformatting) by offering operations that implement gene annotation pipelines based on the utilization of external software (such as Splign/Compart [8], [9] or Augustus [10]) and operations to analyze sequences using well-known web services (such as NCBI BLAST [11], UniProt BLAST [12], [13], and PfamScan [14]). Easy-to-install distributable versions and installers of SEDA are provided to facilitate its adoption by the aforementioned researchers, avoiding the need to install the external dependencies required by some operations thanks to the use of Docker images. In addition, a Docker image for Linux systems for SEDA is also available at the Bioinformatics Docker Images Project web site (https://pegi3s.github.io/dockerfiles/). In this case, Linux users need only install Docker to have access to the last version of SEDA.

Two examples of data preparation protocols based on the use of SEDA have already been published [20], [21], illustrating the versatility of SEDA and its suitability for users without advanced skills. The different SEDA operations, shown in detail in the *Results and Discussion* section, including the gene annotation operations that are a unique feature of SEDA, allow users to address a variety of scientific issues using all available genomic data.

TABLE 1
EXTERNAL DEPENDENCIES REQUIRED TO RUN SOME SEDA OPERATIONS

Tool	Version	Linux	Windows	MacOS
BLAST [15]	2.6.0	Yes	Yes	Yes
Clustal Omega [16]	1.2.4	Yes	Yes	Yes
bedtools [17]	2.25.0	Yes	No	Yes (MacPorts, Homebrew)
EMBOSS [5]	6.6.0	Yes	No	Yes (Native, Homebrew)
Splign/ Compart [8], [9]	N/A	Yes	No	No
ProSplign/ ProCompart [18]	N/A	Yes	No	No
SAPP [19]	12/09/19	Yes	No	No

2 IMPLEMENTATION

SEDA is implemented in Java 8 using the GC4S library for GUI development [22]. The project has a plugin-based architecture and comprises a core module, which provides the SEDA main operations and data structures (classes for representing and managing sequences and files), and several modules that provide additional operations. This way, as explained in the developer's section of the SEDA manual (https://www.sing-group.org/seda/manual/developers.html), SEDA can be easily extended by adding new plugins that provide operations that are automatically integrated by the main application.

2.1 Third party dependencies and Docker

Some operations in SEDA require the use of third-party software such as BLAST, EMBOSS, or bedtools. Table 1

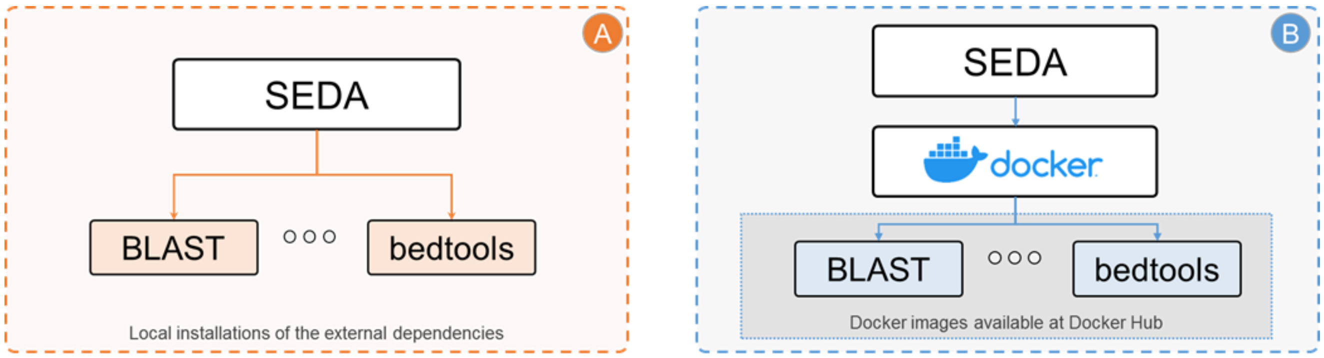

A
SEDA
BLAST
bedtools
Local installations of the external dependencies
B
SEDA
docker
BLAST
bedtools
Docker images available at Docker Hub

Fig. 1. SEDA third-party dependencies execution scenarios. (A) Operations requiring third-party dependencies (e.g. BLAST, EMBOSS) allow users to provide the paths where they are available. (B) SEDA only requires Docker to run the third-party dependencies via existing Docker images belonging to the SEDA project.

shows the complete list of third party dependencies used in SEDA, including the versions used in development as well as their availability in the main operating systems. The Windows, Linux and Mac OS distributions of SEDA allow users to provide the paths of each of these dependencies in case they want to use the operations that require them (Fig. 1A). However, as shown in Table 1, it is important to note that some of these dependencies are only available for Linux systems (e.g. Splign/Compart, ProSplign/ProCompart) and/or can be hard to install. To overcome this issue and to facilitate the use of SEDA, we have created one Docker image for each of these dependencies and equipped SEDA with the possibility of running the third-party software via these Docker images (Fig. 1B) instead of using the local binaries. This way, when running operations that rely on third-party applications, the only dependency of SEDA is Docker, which can be easily installed in the three main operating systems. This possibility enhances the user-friendliness philosophy of SEDA, knocking down the most common barriers that hamper users with no advanced informatics skills from using this kind of software, namely, the installation and configuration of third party dependencies. It is important to note that these external dependencies are only required by the gene annotation pipelines, the BLAST-based operations and the Clustal Omega alignment; most of the SEDA operations can be used without them.

2.2 Handling big files

By default, SEDA loads the sequences into RAM memory to process them, thus ensuring fast processing when there is enough RAM memory to handle them. To deal with big FASTA files or medium-sized files in computers with low RAM memory, SEDA also includes an 'in-disk processing mode' that processes sequences without loading them into memory first, thus saving RAM memory at the expense of running time. In this processing mode, SEDA creates an in-memory FASTA index with the location of the sequences in their corresponding files, which is then used to access them quickly only when they are requested. Therefore, sequence contents are only loaded into memory when they are needed.

2.3 Extending SEDA

Because of its modular and plugin-based architecture, developers can easily add their own operations to the SEDA GUI. In SEDA, each operation is an implementation of the *SedaGuiPlugin* interface, which requires classes implementing operations to provide a name for the operation, a name for the group of the operation, a GUI component to show in the SEDA's GUI and allow users to configure the operation (shown in Fig. 2B), and an implementation of the *TransformationProvider* interface. The processing of sequences in SEDA is built around the idea of transformations. A transformation receives a sequence, a group of sequences (i.e. a FASTA file) or a dataset (i.e. a set of FASTA files) and returns a transformed item of the same type (i.e. a sequence if it receives a sequence as input). *TransformationProvider* implementations must provide a dataset transformation that is responsible for processing all the selected files in SEDA. Developers can combine multiple sequence, file or dataset-level transformations to build the final dataset transformation which constitutes the operation.

Developers must also provide a *SedaPluginFactory* implementation that lists their operations. At runtime, SEDA will find out all the available implementations to add their operations to the operation selection panel. A more detailed guide for developers is provided in the SEDA manual (https://www.sing-group.org/seda/manual/developers.html), including a quick-start section that guides users through the creation of a module containing a simple operation for trimming sequences.

3 Results and Discussion

SEDA is a versatile toolbox that includes several operations whose application depends on the specific purposes of analysis. This means that there is not just one way to use SEDA and researchers must determine which operations to apply and in which order, as the order of the operations can affect both the outcomes and the performance. This section presents general guidelines for the creation of efficient data handling protocols, as well as a brief description of SEDA's operations.

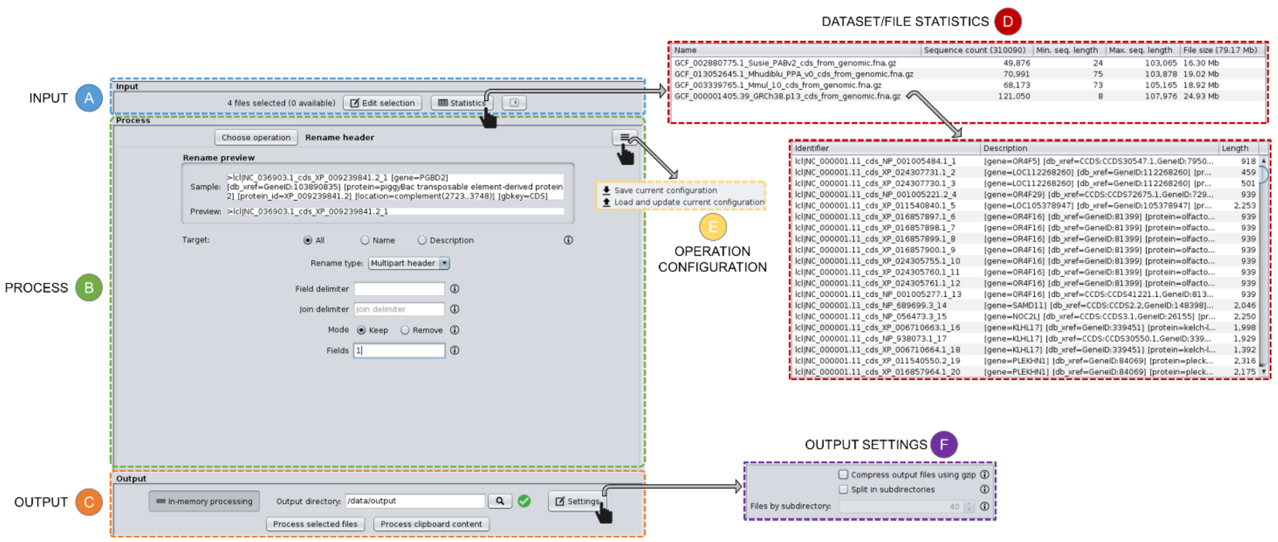
3.1 Using the SEDA GUI

The SEDA GUI was carefully designed to allow users to get used to the application quickly and to be as user-friendly as possible. As Fig. 2 shows, this GUI has three main areas (*Input*, *Process* and *Output)*, arranged in the logical order in which they are frequently used.

The *Input* area (Fig. 2A) allows loading and selecting the input FASTA files to be processed, which can be compressed using gzip or not. The dialog to load and select files includes some useful functionalities, such as a recursive search of files in directories or the possibility to save and load a list of files. In addition, the *Statistics* button displays a table with basic information about each selected FASTA file (Fig. 2D): file name, sequence count, minimum/maximum sequence length, and file size. For each file, an additional table displaying the name, identifier and length of each sequence can be also opened (Fig. 2D).

The *Process* area (Fig. 2B) permits selecting and configuring the operation that should be applied to the selected FASTA files. Supplementary Table S1 lists all the operations of the current SEDA version (v1.2.0). In this area, the menu shown in Fig. 2E allows the user to save the state of the current configuration operation into a text file that can be loaded later to reapply the same parameter settings.

Finally, in the *Output* area (Fig. 2C) users can choose the output directory, configure some output settings (Fig. 2F), and execute the selected operation. In this area users can also choose whether they want to use an 'in-disk' or an 'in-memory' processing profile (as explained previously in subsection 2.2). If some of the input FASTA files are located in the current output directory, the application displays a warning icon and shows a confirmation dialog before processing them to avoid undesired data overwrit-



4 files selected (0 available)

Rename preview

Sample: >lcl|NC_036903.1_cds_XP_009239841.2_1 [gene=PGBD2] [db_xref=GeneID:103890835] [protein=piggyBac transposable element-derived protein 2] [protein_id=XP_009239841.2] [location=complement(2723..3748)] [gbkey=CDS]

Preview: >lcl|NC_036903.1_cds_XP_009239841.2_1

Target: [✓ All] Name Description

Rename type: Multipart header

Field delimiter

Join delimiter: Join delimiter

Mode [✓ Keep] Remove

Fields 1

Output directory: /data/output

Compress output files using gzip

Split in subdirectories

Files by subdirectory: 40

Name	Sequence count (310090)	Min. seq. length	Max. seq. length	File size (79.17 Mb)
GCF_002880775.1_Susie_PABv2_cds_from_genomic.fna.gz	49,876	24	103,065	16.30 Mb
GCF_013052645.1_Mhudiblu_PPA_v0_cds_from_genomic.fna.gz	70,991	75	103,878	19.02 Mb
GCF_003339765.1_Mmul_10_cds_from_genomic.fna.gz	68,173	73	105,165	18.92 Mb
GCF_000001405.39_GRCh38.p13_cds_from_genomic.fna.gz	121,050	8	107,976	24.93 Mb

Identifier	Description	Length
lcl|NC_000001.11_cds_NP_001005484.1_1	[gene=OR4F5] [db_xref=CCDS:CCDS30547.1,GeneID:7950...	918
lcl|NC_000001.11_cds_XP_024307731.1_2	[gene=LOC112268260] [db_xref=GeneID:112268260] [pr...	459
lcl|NC_000001.11_cds_XP_024307730.1_3	[gene=LOC112268260] [db_xref=GeneID:112268260] [pr...	501
lcl|NC_000001.11_cds_NP_001005221.2_4	[gene=OR4F29] [db_xref=CCDS:CCDS72675.1,GeneID:729...	939
lcl|NC_000001.11_cds_XP_011540840.1_5	[gene=LOC105378947] [db_xref=GeneID:105378947] [pr...	2,253
lcl|NC_000001.11_cds_XP_016857897.1_6	[gene=OR4F16] [db_xref=GeneID:81399] [protein=olfacto...	939
lcl|NC_000001.11_cds_XP_016857898.1_7	[gene=OR4F16] [db_xref=GeneID:81399] [protein=olfacto...	939
lcl|NC_000001.11_cds_XP_016857899.1_8	[gene=OR4F16] [db_xref=GeneID:81399] [protein=olfacto...	939
lcl|NC_000001.11_cds_XP_016857900.1_9	[gene=OR4F16] [db_xref=GeneID:81399] [protein=olfacto...	939
lcl|NC_000001.11_cds_XP_024305755.1_10	[gene=OR4F16] [db_xref=GeneID:81399] [protein=olfacto...	939
lcl|NC_000001.11_cds_XP_024305760.1_11	[gene=OR4F16] [db_xref=GeneID:81399] [protein=olfacto...	939
lcl|NC_000001.11_cds_XP_024305761.1_12	[gene=OR4F16] [db_xref=GeneID:81399] [protein=olfacto...	939
lcl|NC_000001.11_cds_NP_001005277.1_13	[gene=OR4F16] [db_xref=CCDS:CCDS41221.1,GeneID:813...	939
lcl|NC_000001.11_cds_NP_689699.3_14	[gene=SAMD11] [db_xref=CCDS:CCDS2.2,GeneID:148398]...	2,046
lcl|NC_000001.11_cds_NP_056473.3_15	[gene=NOC2L] [db_xref=CCDS:CCDS3.1,GeneID:26155] [pr...	2,250
lcl|NC_000001.11_cds_XP_006710663.1_16	[gene=KLHL17] [db_xref=GeneID:339451] [protein=kelch-l...	1,998
lcl|NC_000001.11_cds_NP_938073.1_17	[gene=KLHL17] [db_xref=CCDS:CCDS30550.1,GeneID:339...	1,929
lcl|NC_000001.11_cds_XP_006710664.1_18	[gene=KLHL17] [db_xref=GeneID:339451] [protein=kelch-l...	1,392
lcl|NC_000001.11_cds_XP_011540550.2_19	[gene=PLEKHN1] [db_xref=GeneID:84069] [protein=pleck...	2,316
lcl|NC_000001.11_cds_XP_016857964.1_20	[gene=PLEKHN1] [db_xref=GeneID:84069] [protein=pleck...	2,175

Fig. 2. Details of the Graphical User Interface (GUI) of SEDA. Parts A (Input), B (Process) and C (Output) correspond to the main areas. A typical user interaction with SEDA will consist of: (*i*) loading and selecting the input FASTA files in the Input area; (*ii*) selecting and configuring the desired operation in the Process area; and (*iii*) selecting an output directory and running the operation. Additionally, users can explore the selected files (part D), can save/load the operation configuration (part E), or configure the output settings (part F).

ing. The *Process selected files* button applies the configured operation to all the selected files, generating the results in the output directory. Alternatively, the *Process clipboard content* can be used to apply it to the clipboard contents, provided that they are sequences in FASTA format.

3.2 Guidelines for designing SEDA-based pipelines

The SEDA GUI is appealing to researchers without a background in computing. Nevertheless, because of its ease of manipulation and modular structure, it is also attractive to researchers who spend a reasonable amount of time launching single line commands in the console or even writing small scripts, every time they want to acquire sequence data for a particular project. Although SEDA can be used to perform a single operation on a single or large number of files, it can be used for a much advanced and customised chaining of operations. By being able to combine the different operations in any order, the user can easily implement a protocol for data handling, keeping in mind the planned final use of the FASTA file(s). In this case, the output of one operation is used as the input of the next one until the desired format is achieved. Creating a folder with the name of the operation to be executed inside the folder where the input FASTA files are becomes a convenient way of processing the data when a large number of operations are to be performed.

Supplementary Figures S1-S2 show general guidelines on how to easily build a highly curated dataset for a single or a small number of proteins/CDS using SEDA. The suggested order of the operations likely covers most data formatting needs, and shows the original motivation for developing the operations that are currently available in SEDA. Nevertheless, there are likely alternative ways of building equally efficient protocols using SEDA. When developing a protocol to retrieve sequences for only one or a few genes, the first operation should dramatically reduce the number of sequences to be processed in the next step. This is why, when using proteins/CDS, the first step of the protocol is usually a BLAST search, a pattern filtering step or a base presence filtering step (Supplementary Figure S1). While BLAST operations can take a while (for instance, the local BLAST operation using all RefSeq animal CDS as the database takes about 20 minutes), the subsequent operations usually take less than 1 second.

The great advantage of remotely performing BLAST searches on NCBI or UniProt using SEDA, is that BLAST searches can be faster and there is no need to download a large number of large FASTA files to the computer where SEDA is running. It should be pointed out, however, that NCBI will move searches to a slower queue, or, in extreme cases, block the requests, if BLAST searches are performed too often. Our experience shows that the time it takes to perform the same BLAST search at NCBI at different moments in time can be very variable. The other disadvantage is that the user is restricted to the seven databases (https://ncbi.github.io/blast-cloud/blastdb/available-blastdbs.html) that are made available by NCBI. These include some popular databases such as the “Nucleotide collection”, the “Non-redundant” protein collection or the “Non-redundant UniProtKB/SwissProt sequences” database, but there is not, for instance, a RefSeq CDS database. Moreover, the popular “Nucleotide collection” is a mix of data from several sources, meaning that the resulting output will contain non-equivalent types of data (for instance, the output of the BLAST search can show mRNA, CDS, and whole chromosomes sequences), meaning that an extra filtering step (which can be performed using SEDA; Supplementary Figure S1) is usually needed to obtain a homogeneous data set.

The user should also be aware that when using both the NCBI BLAST and the UniProt BLAST operations, the further processing of the files can be more complex. For instance, if a local BLAST search is performed using

SEDA and the data downloaded from public databases, the BLAST results can be obtained for each species in a separate file. Therefore, in order to use SEDA's Remove isoforms operation (Supplementary Figure S2), the user needs to specify a single parameter (Reference size). When using the NCBI BLAST operation the user gets the result of the BLAST in a single file, and thus the "Header matcher configuration" option of the Remove isoforms operation must be used in order to split the input sequences into groups that will be processed separately. This means using regular expressions, but not all users may be familiar with such operations. Moreover, the NCBI Rename operation, which allows adding taxonomy information to the sequence headers, cannot be used, since it requires the presence of GCA/GCF codes in the FASTA file names. Such codes are present in the FASTA files when downloading data from GenBank and RefSeq databases. The amount of time needed to download CDS data, although significant, is not a major hurdle. For instance, when using a high-speed internet connection (1 Gb/s download speed so that the limitation is likely on the server side), the download from RefSeq of the 7.6 GB compressed file containing all animal CDS takes about 12 minutes. When downloading from GenBank it takes about 15 minutes to download the 4.5 GB zipped file containing all animal CDS. The downloaded datasets are static, which is advantageous if analyses need to be repeated or refined, but this can also be a disadvantage because gene annotations may change after the data have been downloaded.

The download of non-annotated genomes, for the purpose of annotating the genes of interest in genomes that are still non-annotated typically takes between 2 seconds and 2.5 minutes (depending on the animal genome size) per genome. Although significant, for most purposes, it is unlikely that there would be tens or hundreds of non-annotated genomes of interest.

Generally speaking, there are two operations for performing gene annotations depending on whether the gene to be annotated is a mono or multi exonic gene. In the former case, the getorf (EMBOSS) operation can be used to obtain all open reading frames longer than the value specified, followed by a BLAST search (Supplementary Figure S1). In the latter case, since the implemented gene annotation operations can take a while to run, it is advisable to first perform a local BLAST search using the genome data in order to retrieve the hit regions as well as the neighbouring regions only. The retrieved sequences are then re-assembled using SEDA's grow sequences operation. This way, Augustus, ProSplign/ProCompart or Splign/Compart (the three other gene annotation operations available in SEDA) can be used to annotate the genes present in these much smaller genome regions known to contain the gene of interest only.

The same basic protocol may be used in a large number of situations. For instance, when performing phylogenetic studies using CDS, no matter the gene(s) of interest, the basic SEDA protocol described by López-Fernández et al. [20] (an illustrated step-by-step protocol is presented in Suplementary Figure S3) is usually a good option since, when performing such studies, researchers often want to display information regarding the species name, family name (if a wide range of species is to be used), and a unique identifier to facilitate the readability of the phylogram/cladogram that will be obtained using an additional phylogenetics software package(s). Therefore, the headers of the FASTA file given as input to the selected software application(s) must show only that information. One critical step in this protocol is the use of the Remove isoforms operation since only one sequence per gene is used in phylogenetic studies, and in many genomes several CDS isoforms are annotated (Supplementary Figure S2). The protocol described in López-Fernández et al. [20] can also be slightly adapted to include for instance a filtering by domain step, or to include an alignment step to check for the presence of sequences that are clearly miss-annotated since they generate non-multiple of three alignments.

Using SEDA, the user can also retrieve all members of a given protein family, such as mucins, and look for the associated protein domains in different species. In Supplementary Figure S4, an illustrated step-by-step protocol shows how simple it is to perform this type of operation using SEDA.

SEDA's BLAST: two-way ortholog identification operation can be used to automate data acquisition for multiple genes and infer the phylogenetic relationships of the species studied using, for instance, a multi gene concatenated sequence alignment (Supplementary Figure S5). Most phylogenetic approaches ignore gapped sequence alignment regions. Therefore, it may be useful to remove gene datasets that show fewer sequences than the number of species analyzed, which can be done using SEDA's Filtering operation. The sequence alignment of the large number of files and the concatenation of the aligned sequences can be done using SEDA's Clustal Omega Alignment and Concatenate Sequences operations, respectively. SEDA's BLAST: two-way ortholog identification operation can also be used to prepare FASTA files for polymorphism studies or studies aimed at identifying positively selected amino acid sites, for instance. Supplementary Figure S6 shows an illustrated step-by-step protocol for a typical phylogenomics study.

These are just a few examples of the utility of SEDA for data handling. The 'Protocols' section of the SEDA manual (https://www.sing-group.org/seda/manual/protocols.html) includes step-by-step execution guides of the three use cases presented here. It should be noted that although SEDA covers most data handling needs, it was developed so that new operations can be added to easily solve presently unforeseen needs. By maximizing SEDA's flexibility there is no need to waste time in every individual lab to solve the same data handling needs. The next subsections briefly discuss SEDA's operations.

3.3 Alignment-related operations

This group includes five operations: *Clustal Omega alignment*, *Concatenate sequences*, *Consensus sequence, Trim alignment*, and *Undo alignment*.

The *Clustal Omega alignment* operation permits using Clustal Omega to align the input FASTA files, allowing users to provide additional command-line parameters as well.

The *Concatenate sequences* operation allows all the selected input FASTA files to be merged into a single output FASTA by concatenating equivalent sequences. The equivalence of the sequences is determined based on their headers, using two criteria: (*i*) *'Sequence name'*, which means that the sequences are concatenated if they have the same sequence names (identifiers), or (*ii*) *'Regular expression'*, which means sequences are concatenated by matching headers using a regular expression.

The *Consensus sequence* operation permits the creation of a consensus sequence from a set of aligned sequences. The consensus sequence is constructed by looking at each position for the nucleotide (DNA) or amino acid (protein) that is present at a frequency higher than that specified by the user.

The *Trim alignment* operation allows the removal of alignment gap stretches in the beginning and end of each alignment.

Finally, the *Undo alignment* operation simply allows the user to undo a sequence alignment by removing gaps from sequences.

3.4 BLAST

This group includes four operations: *BLAST*, *BLAST: two-way ortholog identification*, *NCBI BLAST*, and *UniProt BLAST*.

The first operation, simply named *BLAST* in SEDA, allows users to run BLAST queries using all the input FASTA files in SEDA as database. In this regard, SEDA allows two possible modes: querying against all the selected FASTA files at the same time (i.e. a single database formed by all files) or querying against each FASTA file separately. Regarding the query, there are also two possibilities: using the sequences in one of the selected FASTA as queries or using the sequences in an external FASTA file as queries. When performing this operation, one BLAST query is executed for each sequence in the FASTA file. This operation allows choosing the BLAST type (blastn, blastp, tblastn, etc.), parameters like E-value or the maximum number of target sequences to retrieve and to provide additional command-line parameters.

The *BLAST: two-way ortholog identification* operation allows finding the orthologous sequences in a set of FASTA files. This operation implements the method known as Reciprocal Best Hits (RBH), which assumes two genes (each in a different genome) to be orthologs if they can find each other as the best hit in the other genome [23]. Like the *BLAST* operation, this operation allows choosing the BLAST type (blastn, blastp, tblastn, etc.) and E-value, and to provide additional command-line parameters.

Finally, the *NCBI BLAST* and the *UniProt BLAST* operations allows BLAST queries to be run through the NCBI and UniProt web servers respectively. The *NCBI BLAST* operation uses the *Common URL API* (https://ncbi.github.io/blast-cloud/dev/api.html) to submit the BLAST queries through the NCBI BLAST web server [11]. The *UniProt BLAST* operation uses the UniProtJAPI [13] to submit the BLAST queries. To meet the NCBI and UniProt/EMBL-EBI usage guidelines and to avoid problems, these two operations limit users to query one sequence at a time.

3.5 Filtering

This group includes five operations for filtering FASTA files and reducing the number of sequences in them: *Base presence filtering*, *Filtering*, *Pattern filtering*, *Remove isoforms*, and *Remove redundant sequences*.

The *Base presence filtering* operation allows filtering sequences based on the percentages of their nucleotides or amino acids. Sequences with units whose percentage of presence is outside the specified thresholds are removed from the output FASTA file.

The *Filtering* operation allows sequences and/or entire FASTA files to be filtered based on the following set of criteria, which are applied in this same order when one or more are selected:

- *Valid starting codons*: filters sequences and keeps only those starting with specified codons.
- *Remove sequences with a non-multiple of three size*: filters sequences and keeps only those having a length that is a multiple of 3.
- *Remove sequences with in-frame stop codons*: filters sequences and keeps only those without in-frame stop codons.
- *Minimum/maximum sequence length*: filters sequences and keeps only those with the specified minimum/maximum sequence length. A value of 0 indicates that no minimum/maximum sequence length is required.
- *Header count filtering (sequences)*: if the header count filtering option is selected at the sequences level, then it filters sequences and keeps (or removes) only those meeting the specified criteria regarding header counts. For instance, this option can be used to keep (or remove) only those sequences whose sequence identifier appears a certain number of times.
- *Minimum/maximum number of sequences*: filters FASTA files and keeps only those with the specified minimum/maximum number of sequences.
- *Header count filtering (files):* if the header count filtering option is selected at the files level, then it filters FASTA files and keeps (or removes) only those files where all sequences meet the specified criteria regarding header counts.
- *Remove by size difference*: filters sequences and keeps only those with the specified difference when compared to a reference sequence. The reference sequence can be one from the input FASTA file being processed (specified by its index) or one provided in an external file.

The *Pattern filtering* operation allows sequences to be filtered based on text patterns or regular expressions. The pattern filtering can be applied to the sequence headers or the sequence contents. In the latter case, nucleotide sequences can be translated into amino acid sequences before applying the filtering. If so, note that the corresponding output FASTA file will contain the original input sequences, not the translated ones.

The *Remove isoforms* operation detects isoforms and allows only one to be kept in each input FASTA file by applying the following algorithm:

1) Start with the first sequence (FS) and compare it against the remaining ones.
2) Each pair of sequences (FS vs. Second Sequence, SS) is considered to be isoforms if they share a word of a given minimum length (*Minimum word length* parameter).
3) If they are isoforms, the SS is marked as isoform of the FS, so that SS will not be taken for further comparisons.
4) Repeat steps 1 to 3 for the remaining sequences.
5) Finally, for each group of isoforms, the *isoform selection criteria* configured is applied to select which isoform should go to the output FASTA file.

These criteria include a *reference size*, which means that the isoform with the length closest to this reference size will be selected. In case of having two isoforms that are at the same distance, the *tie break mode* parameter allows specifying which one should be selected. It can be: (*i*) *shortest*, which means that the sequence with fewer units (i.e. nucleotides or amino acids) will be retained, or (*ii*) *longest*, which means that the sequence with more units will be retained.

This process is applied to all sequences in each input FASTA file. Nevertheless, it is possible to split them in groups to be processed separately, something useful for those situations where sequences from two or more species are mixed in the same file, since this operation should be applied to each species separately.

Finally, the *Remove redundant sequences* operation allows the removal of sequences with exactly the same sequence nucleotides or amino acids. This operation also has a *'Remove also subsequences'* option, which tells SEDA to also remove sequences contained within longer sequences.

3.6 Gene Annotation

This group includes three operations implementing gene annotation pipelines based on Augustus [10] as implemented in SAPP [19], Splign/Compart [8], [9], and ProSplign/ProCompart [18], and one operation for getting open reading frames (using the *getorf* program from EMBOSS [5]).

The *Augustus (SAPP)* operation makes it possible to annotate a eukaryotic genome or sequence of interest by predicting genes using Augustus [10], [19]. Specifically, this operation applies the following steps to each input FASTA file:

1) Converts the input FASTA file to HDT using the *fasta2hdt* command from the SAPP Conversion module, setting the organism and genetic code specified by the user as parameters.
2) Runs Augustus on the converted file using the *genecaller* module, producing a GFF3 file with the annotations.
3) Filters the GFF3 annotations file in order to keep only those corresponding to the CDS regions of the annotated genes.
4) Extracts these regions from the FASTA file to produce the output FASTA file with the annotations using bedtools.

The *Splign/Compart Pipeline* operation permits the annotation of exons or genes using the protocol described by the NCBI [8], [9]. Note that this operation requires a reference nucleotide CDS to be available from a closely related species. In this operation, each input FASTA file is used as a target and users must provide an external CDS file in the operation configuration. For each pair of target FASTA and external CDS, this operation applies the following steps:

1) Create a bidirectional genome for the target FASTA (i.e. a FASTA file containing both the original and the reversed sequences).
2) Create BLAST databases for both the bidirectional genome and the external CDS.
3) Run the *mklds* option of Splign (*splign --mklds*) on the working directory to create an LDS index that Splign will use to access the FASTA sequences.
4) Run Compart to produce the preliminary cDNA-to-genomic alignments (i.e. the compartments).
5) Run the *ldsdir* option of Splign (*splign --ldsdir*) to obtain the annotations using the obtained compartments as input.
6) Convert the *ldsdir* output annotations into a BED file.
7) Extract the regions in the BED file from bidirectional genome FASTA file to produce the output FASTA file with the annotations using bedtools.
8) If the concatenate exons option is selected, the adjacent exons are concatenated in the output FASTA file. Using this option, if an annotation is obtained for every exon of a given gene then the resulting sequence will be the complete CDS.

The *ProSplign/ProCompart Pipeline* operation makes it possible to obtain CDS annotations using protein reference sequences following the protocol described by the NCBI [18]. This pipeline can be seen as an alternative to the Splign/Compart pipeline. This operation uses protein reference sequences rather than reference nucleotide CDS. Since protein sequences change at a slower pace than nucleotide sequences, in principle, the reference and target sequences can be more distantly related than when using the Splign/Compart pipeline, but it is difficult to quantify how distantly related they can be. Each input FASTA file is used as nucleotide subject file and an external file with the protein query must be provided by the users in the operation configuration. For each pair of nucleotide subject file and protein query file, this operation applies the following steps:

1) Create a BLAST database for the nucleotide subject file.
2) Run a *tblastn* using the protein query file against the subject database, sorting the output by subject and query.
3) Run ProCompart to find the approximate loca-

tions of the protein instances on the nucleic acid sequences.

4) Run ProSplign using the compartment file generated in the previous step and the two FASTA files as input to generate an alignment for each compartment.
5) Extract the sequences from the alignments file in order to create the output FASTA file with the annotations.

Finally, the *getorf (EMBOSS)* operation permits finding and extracting open reading frames (ORFs) using the *getorf* program from the EMBOSS suite [5]. This operation is able to annotate genes that are both in the forward and reverse strand.

3.7 Protein annotation

This group includes one operation named *PfamScan* to search and annotate sequences against the Pfam-A HMM library using the EMBL-EBI web service (https://www.ebi.ac.uk/Tools/pfa/pfamscan/) [14]. This operation submits each input sequence to the PfamScan web service and obtains a set of Pfam-A HMM annotations. Each sequence header is then modified to contain the original sequence identifier along with a summary of the PfamScan annotations. To meet the EMBL-EBI usage guidelines and to avoid problems, this operation runs PfamScan queries in batches of 30 sequences. In addition, the amount of time SEDA waits between batches is equal to the time required to analyze the first batch (this delay can be controlled using the *'Batch delay factor'* parameter).

3.8 Reformatting

This group includes the following six operations to change the format of the FASTA files in different ways: *Disambiguate sequence names*, *NCBI rename*, *Reallocate reference sequences*, *Reformat file*, *Rename header*, and *Sort*.

The *Disambiguate sequence names* operation allows duplicated sequence names (identifiers) to be disambiguated. This operation can work in two modes: (*i*) *Rename*, to add a numeric prefix to disambiguate duplicate names, or (*ii*) *Remove*, to remove sequences with duplicate identifiers and keep the first occurrence.

The *Reallocate reference sequences* operation uses a text pattern to find one or more sequences (i.e. the reference sequences) and reallocates them at the beginning of the file. For instance, this operation can be used to place a sequence of interest at the beginning of a FASTA file and then use it as the reference sequence in the *'Remove by size difference'* option of the *Filtering* operation.

The *Reformat file* operation allows changes to be made to the format of the FASTA file, including:

- *Fragment length*: the fragment length or number of columns in which sequences are split into multiple lines. This option includes a *'Remove line breaks'* checkbox that specifies that sequences should not be fragmented into different lines.
- *Line breaks*: the type of line breaks of the file, which can be *'Windows'* or *'Unix'*.
- *Case*: the case of the sequences, which can be *'Original'*, meaning that the original case in input sequences is kept, or *'Lower case'* / *'Upper case'* to convert sequences to lower or upper case bases respectively.

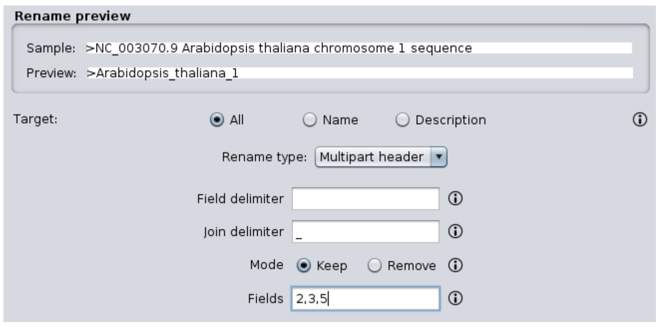
Despite the simplicity of the *Reformat file* operation, it has proved to be extremely useful since some software require the FASTA files to have a specific format (e.g. not supporting fragmented lines and requiring Windows line breaks).

Sequence headers are known to cause issues when processing FASTA files with different software tools. Selection of sequences with long sequence names is a problem for DnaSP [24] and in MEGA [25], although it is not a problem that impedes using the software, it leads to a highly compressed view of the sequence alignment and of the phylogeny. Moreover, when running local BLAST, the inclusion of special characters in sequence headers can lead to a variety of problems. To address these and other issues, the *Rename header* operation allows modifying the sequence headers using four modes:

- *Multipart header*: this mode allows splitting the sequence header into fields by specifying a *field delimiter* character. Then, it is possible to choose the fields to keep or remove in the modified sequence headers. When keeping fields, it is also possible to rearrange the order of the fields.
- *Replace word*: this mode allows replacing one or more words (*targets*) with a *replacement* word. In addition, regular expressions can be used as *targets*.
- *Replace interval*: this mode allows replacing a text interval (defined by two text strings *from* and *to*) with a *replacement* word.
- *Add prefix/suffix*: this mode allows adding a word to the header as prefix or suffix separated by a *delimiter*, or completely replacing it (*'Override'*). In addition, this mode also allows adding a correlative index to each renamed sequence.

Since this operation requires a precise configuration that depends on the aspect of the headers of the input FASTA files, SEDA includes a preview showing the effect of applying the current renaming configuration to the first header of one randomly selected file (Fig. 3).

The *NCBI rename* operation makes it possible to add/replace NCBI accession identifiers with the associated organism name and additional information from the



Rename preview

Sample: >NC_003070.9 Arabidopsis thaliana chromosome 1 sequence
Preview: >Arabidopsis_thaliana_1

Target: [✓ All] Name Description

Rename type: Multipart header

Field delimiter:

Join delimiter: _

Mode: [✓ Keep] Remove

Fields: 2,3,5

Fig. 3. Screenshot of the *Rename header* operation showing the *Multipart header* mode. The field *Sample* shows the current look of a header, while the *Preview* field shows how it will look after the operation. In this case, the header is split into fields by using a white space as delimiter. Renamed headers are formed by joining the second, third and fifth fields separated by the '_' character.

NCBI Taxonomy Browser. This addition/replacement can be done for both FASTA file names and the sequence headers of the sequences contained in them. This operation requires the presence of the GCA/GCF codes in the FASTA file names. This operation has many configuration parameters that are covered in the documentation in detail. Fig. 4 shows a simple case where the three input FASTA files correspond to three genome assemblies downloaded from NCBI. First, the *Rename header* operation is used to keep only the protein accession numbers (highlighted in purple) in the sequence headers. Note that the intermediate file names produced by this first operation are not modified and the GCA/GCF codes are still there. Secondly, the *NCBI Rename* operation is applied using such intermediate files. First, the *NCBI Rename* operation extracts the NCBI accession identifiers from the file names (highlighted in red) and makes a query to NCBI to retrieve the associated organism names. Then, using the *'Override'* mode, the name of each output FASTA file is replaced with the selected taxonomy information (family and order). Finally, the operation modifies the sequence headers by prefixing the sequence headers with the corresponding requested taxonomy information. This operation is important in the context of evolutionary questions that can be answered using phylogenetic inferences, since, for instance, the species name is not included when downloading CDSs from the NCBI RefSeq database [20], [21]. Moreover, when using data from hundreds of species, it is useful to add to sequence headers information on the species family or class to check if sequences are easily grouped into the phylogenetic analyses according to known phylogenetic relationships.

Finally, the *Sort* operation allows sequences to be sorted based either on sequence headers or on the content of the sequences. The order can be determined by length or alphabetical order.

3.9 General

This group includes eight operations whose functions do not fit in the previous groups, namely: *Compare*, *Grow sequences*, *Merge*, *Split*, *Regular expression split*, *Remove stop codons*, *Reverse Complement*, and *Translate*.

The *Compare* operation allows all the possible pairwise comparisons on the input FASTA files to be made. For each pair of FASTA files under comparison (e.g. *Input1* and *Input2*) three output files are produced: (*i*) *Input1_vs_Input2_both.fasta*, containing sequences present in both files, (*ii*) *Input1_vs_Input2_only_Input1.fasta*, containing those sequences that only appear in *Input1*, and (*iii*) *Input1_vs_Input2_only_Input2.fasta*, containing those sequences that only appear in *Input2*. Sequences can be compared by their headers or contents.

The *Grow sequences* operation allows sequences to grow or expand by merging those with a specified minimum of overlapping bases. This operation applies the following algorithm:

1) Take the first sequence as the reference sequence.
2) Compare the reference sequence with the rest of sequences. For each pair of sequences, check if there is an overlapping of bases of at least the minimum size specified. This overlapping is searched for at the beginning of the reference sequence and at the ending of the sequence being compared.
    a. If an overlap is found, merge the two sequences. The merged sequences are removed from the set of sequences and the new one is added. Return to step 1.
    b. If an overlap is not found between the first reference sequence and the rest of sequences, then step 2 is repeated for the rest of sequences repeatedly.
3) The process stops when all sequences have been compared without merging any of them.

For instance, this operation can be useful to obtain full size transcripts starting from the results of different transcriptome projects.

The *Merge* operation simply allows all the input FASTA files to be merged into a single output FASTA file.

The *Split* operation allows each input FASTA file to be split into several files in different ways. The way of splitting them is controlled by the *Split mode* parameter, which can be:

- *Fixed number of sequences per file*: each input FASTA is split into several files containing the defined *number of sequences* in each one.

- *Fixed number of files*: each input FASTA is split into the specified *number of files*, each one containing the same number of sequences.
- *Fixed number of sequences per defined number of files*: each input FASTA is split into the defined *number of files*, each one containing the *number of sequences*. It is important to note that the result of multiplying *number of files* by *number of sequences* should be less than or equal to the number of sequences contained in the input FASTA file being processed, otherwise there are not enough sequences to do the split. Nevertheless, for occasions when it may be necessary to do that, this operation includes an *independent extractions* parameter that allows sequences to be taken several times when creating the splits.

This operation includes a *randomize* parameter, which allows sequences in the input FASTA to be randomly ordered before generating the splits. To ensure reproducibility, the random seed can be specified before shuffling the sequences.

The *Regular expression split* is a powerful operation that allows each input FASTA file to be split based on regular expression patterns applied to the sequence headers. This can be useful when, for instance, sequences from two or more species are mixed in the same FASTA file and one FASTA file per species is wanted for subsequent analysis. By defining the appropriate regular expression to extract the species names in the sequence headers, this operation can perform the desired separation.

The *Remove stop codons* operation allows the sequences in each input FASTA file to be modified by removing the stop codons (i.e. TGA, TAG, and TAA) placed at the end of them.

The *Reverse Complement* operation allows nucleic sequences to be converted into their reverse, complement, or reverse complement counterparts.

Finally, the *Translate* operation allows nucleic acid sequences to be translated into their corresponding amino acid sequences. It can translate to the three forward and three reverse frames, and output multiple frame translations at once. Users can choose between twenty-five genetic codes (obtained from NCBI) or provide a file with a custom one.

3.10 Reproducibility

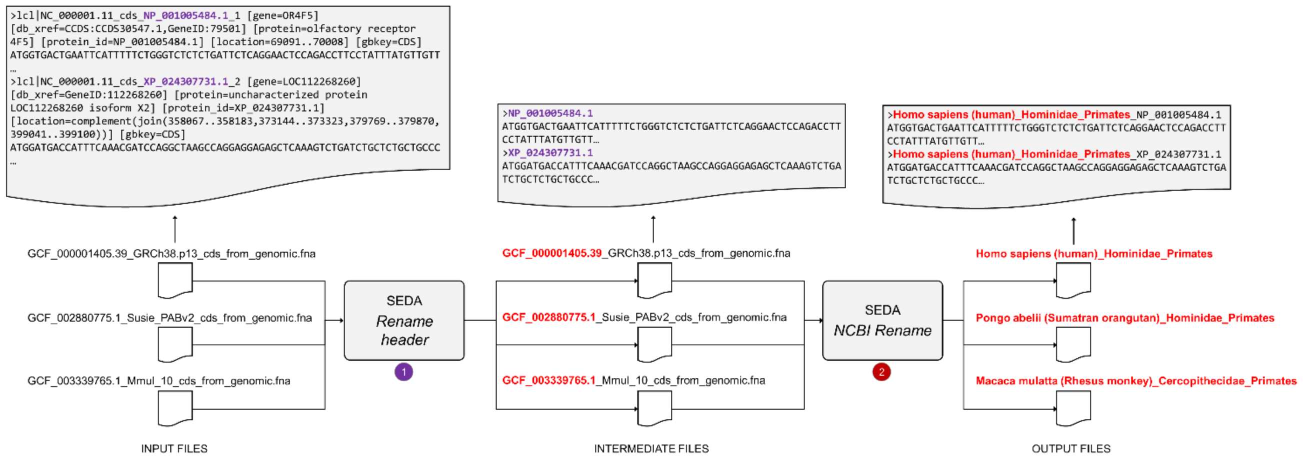
SEDA makes it possible to save the configuration of each operation into a text file that can be loaded later to reapply the same parameter settings. However, SEDA does not handle workflows involving multiple operations and file sets. We are evaluating the possibility of introducing this feature into SEDA as it was not originally conceived to operate in such automated way. For instance, some SEDA workflows can include multiple data paths that are merged at some point [20] or require processing steps (e.g. manual verifications) that impede a complete automatization.

4 Conclusion

We have presented SEDA, a multiplatform software application for the manipulation of FASTA files, specifically designed to allow users less proficient in bioinformatics to process big sets of FASTA files easily, thus improving their research capabilities. The SEDA repertoire of operations covers common manipulations (e.g. sort, reformat, translation, etc.) to more advanced functionalities such as gene annotation workflows or NCBI renaming.

SEDA can be easily installed in Windows, Mac OS, and Linux, requiring minimal configuration effort. The majority of SEDA operations do not require any dependencies, and those requiring external dependencies to run (e.g. BLAST, EMBOSS, etc.) can use Docker. This way, the only dependency of SEDA would be Docker, which is available for the main operating systems.

SEDA is open to further extension, in particular to new operations or improvements of the existing ones. SEDA (https://www.sing-group.org/seda) is freely distributed



```
>lcl|NC_000001.11_cds_NP_001005484.1_1 [gene=OR4F5]
[db_xref=CCDS:CCDS30547.1,GeneID:79501] [protein=olfactory receptor
4F5] [protein_id=NP_001005484.1] [location=69091..70008] [gbkey=CDS]
ATGGTGACTGAATTCATTTTTCTGGGTCTCTCTGATTCTCAGGAACTCCAGACCTTCCTATTTATGTTGTT
…
>lcl|NC_000001.11_cds_XP_024307731.1_2 [gene=LOC112268260]
[db_xref=GeneID:112268260] [protein=uncharacterized protein
LOC112268260 isoform X2] [protein_id=XP_024307731.1]
[location=complement(join(358067..358183,373144..373323,379769..379870,
399041..399100))] [gbkey=CDS]
ATGGATGACCATTTCAAACGATCCAGGCTAAGCCAGGAGGAGAGCTCAAAGTCTGATCTGCTCTGCTGCCC
…
```

```
>NP_001005484.1
ATGGTGACTGAATTCATTTTTCTGGGTCTCTCTGATTCTCAGGAACTCCAGACCTT
CCTATTTATGTTGTT…
>XP_024307731.1
ATGGATGACCATTTCAAACGATCCAGGCTAAGCCAGGAGGAGAGCTCAAAGTCTGA
TCTGCTCTGCTGCCC…
```

```
>Homo sapiens (human)_Hominidae_Primates_NP_001005484.1
ATGGTGACTGAATTCATTTTTCTGGGTCTCTCTGATTCTCAGGAACTCCAGACCTT
CCTATTTATGTTGTT…
>Homo sapiens (human)_Hominidae_Primates_XP_024307731.1
ATGGATGACCATTTCAAACGATCCAGGCTAAGCCAGGAGGAGAGCTCAAAGTCTGA
TCTGCTCTGCTGCCC…
```

GCF_000001405.39_GRCh38.p13_cds_from_genomic.fna
GCF_002880775.1_Susie_PABv2_cds_from_genomic.fna
GCF_003339765.1_Mmul_10_cds_from_genomic.fna
SEDA Rename header
1
GCF_000001405.39_GRCh38.p13_cds_from_genomic.fna
GCF_002880775.1_Susie_PABv2_cds_from_genomic.fna
GCF_003339765.1_Mmul_10_cds_from_genomic.fna
SEDA NCBI Rename
2
Homo sapiens (human)_Hominidae_Primates
Pongo abelii (Sumatran orangutan)_Hominidae_Primates
Macaca mulatta (Rhesus monkey)_Cercopithecidae_Primates
INPUT FILES
INTERMEDIATE FILES
OUTPUT FILES

Fig. 4. Example of application of the *NCBI Rename* operation, preceeded with the *Rename header* operation. (1) The three input FASTA files on the left are processed with the *Rename header* operation to generate three intermediate FASTA files where the sequence headers only retain the protein accession numbers (highlighted in purple in the upper text boxes). (2) These files are then processed with the *NCBI Rename* operation to produce three output files, with the new names being the family and the order corresponding to their NCBI accession identifiers (highlighted in red). The upper text boxes show the content of these files to illustrate how sequence headers are also modified to append the taxonomy information as prefix.

under the GPLv3 license. A comprehensive user manual is available at https://www.sing-group.org/seda/manual/index.html.

Acknowledgment

This work was partially supported by Consellería de Educación, Universidades e Formación Profesional (Xunta de Galicia) under the scope of the strategic funding ED431C2018/55-GRC Competitive Reference Group and through FCT—Fundação para a Ciência e a Tecnologia, I.P., under the project UIDB/04293/2020. SING group thanks the CITI (Centro de Investigación, Transferencia e Innovación) from the University of Vigo for hosting its IT infrastructure. The authors are also grateful to BioData.pt, funded by project 22231/01/SAICT/2016.

References

[1] P. A. Kitts *et al.*, "Assembly: a resource for assembled genomes at NCBI," *Nucleic Acids Res*, vol. 44, no. Database issue, pp. D73–D80, Jan. 2016, doi: 10.1093/nar/gkv1226.

[2] W. Shen, S. Le, Y. Li, and F. Hu, "SeqKit: A Cross-Platform and Ultrafast Toolkit for FASTA/Q File Manipulation," *PLoS ONE*, vol. 11, no. 10, p. e0163962, Oct. 2016, doi: 10.1371/journal.pone.0163962.

[3] "seqmagick." [Online]. Available: https://fhcrc.github.io/seqmagick/. [Accessed: 20-Mar-2020]

[4] H. Li, *lh3/seqtk*. 2020 [Online]. Available: https://github.com/lh3/seqtk. [Accessed: 20-Mar-2020]

[5] P. Rice, I. Longden, and A. Bleasby, "EMBOSS: the European Molecular Biology Open Software Suite," *Trends Genet.*, vol. 16, no. 6, pp. 276–277, Jun. 2000.

[6] T. Carver and A. Bleasby, "The design of Jemboss: a graphical user interface to EMBOSS," *Bioinformatics*, vol. 19, no. 14, pp. 1837–1843, Sep. 2003, doi: 10.1093/bioinformatics/btg251.

[7] P. Villesen, "FaBox: an online toolbox for fasta sequences," *Mol Ecol Notes*, vol. 7, no. 6, pp. 965–968, Nov. 2007, doi: 10.1111/j.1471-8286.2007.01821.x.

[8] Y. Kapustin, A. Souvorov, T. Tatusova, and D. Lipman, "Splign: algorithms for computing spliced alignments with identification of paralogs," *Biology Direct*, vol. 3, no. 1, p. 20, May 2008, doi: 10.1186/1745-6150-3-20.

[9] National Center for Biotechnology Information (NCBI), "Splign." [Online]. Available: https://www.ncbi.nlm.nih.gov/sutils/splign/splign.cgi. [Accessed: 20-Mar-2020]

[10] K. J. Hoff and M. Stanke, "Predicting Genes in Single Genomes with AUGUSTUS," *Current Protocols in Bioinformatics*, p. e57, Nov. 2018, doi: 10.1002/cpbi.57.

[11] M. Johnson, I. Zaretskaya, Y. Raytselis, Y. Merezhuk, S. McGinnis, and T. L. Madden, "NCBI BLAST: a better web interface," *Nucleic Acids Research*, vol. 36, no. Web Server, pp. W5–W9, May 2008, doi: 10.1093/nar/gkn201.

[12] E. Boutet, D. Lieberherr, M. Tognolli, M. Schneider, and A. Bairoch, "UniProtKB/Swiss-Prot," in *Plant Bioinformatics*, D. Edwards, Ed. Totowa, NJ: Humana Press, 2007, pp. 89–112 [Online]. Available: http://link.springer.com/10.1007/978-1-59745-535-0_4. [Accessed: 03-Aug-2020]

[13] S. Patient, D. Wieser, M. Kleen, E. Kretschmann, M. Jesus Martin, and R. Apweiler, "UniProtJAPI: a remote API for accessing UniProt data," *Bioinformatics*, vol. 24, no. 10, pp. 1321–1322, May 2008, doi: 10.1093/bioinformatics/btn122.

[14] F. Madeira *et al.*, "The EMBL-EBI search and sequence analysis tools APIs in 2019," *Nucleic Acids Research*, vol. 47, no. W1, pp. W636–W641, Jul. 2019, doi: 10.1093/nar/gkz268.

[15] C. Camacho *et al.*, "BLAST+: architecture and applications," *BMC Bioinformatics*, vol. 10, no. 1, p. 421, 2009, doi: 10.1186/1471-2105-10-421.

[16] F. Sievers and D. G. Higgins, "Clustal Omega for making accurate alignments of many protein sequences: Clustal Omega for Many Protein Sequences," *Protein Science*, vol. 27, no. 1, pp. 135–145, Jan. 2018, doi: 10.1002/pro.3290.

[17] A. R. Quinlan and I. M. Hall, "BEDTools: a flexible suite of utilities for comparing genomic features," *Bioinformatics*, vol. 26, no. 6, pp. 841–842, Mar. 2010, doi: 10.1093/bioinformatics/btq033.

[18] National Center for Biotechnology Informatio (NCBI), "ProSplign." [Online]. Available: https://www.ncbi.nlm.nih.gov/sutils/static/prosplign/prosplign.html. [Accessed: 20-Mar-2020]

[19] J. J. Koehorst, J. C. J. van Dam, E. Saccenti, V. A. P. Martins dos Santos, M. Suarez-Diez, and P. J. Schaap, "SAPP: functional genome annotation and analysis through a semantic framework using FAIR principles," *Bioinformatics*, vol. 34, no. 8, pp. 1401–1403, Apr. 2018, doi: 10.1093/bioinformatics/btx767.

[20] H. López-Fernández *et al.*, "Bioinformatics Protocols for Quickly Obtaining Large-Scale Data Sets for Phylogenetic Inferences," *Interdiscip Sci Comput Life Sci*, vol. 11, no. 1, pp. 1–9, Mar. 2019, doi: 10.1007/s12539-018-0312-5.

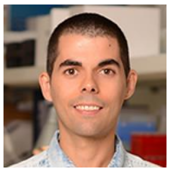
[21] H. López-Fernández *et al.*, "A Bioinformatics Protocol for Quickly Creating Large-Scale Phylogenetic Trees," in *Practical Applications of Computational Biology and Bioinformatics, 12th International Conference*, vol. 803, F. Fdez-Riverola, M. S. Mohamad, M. Rocha, J. F. De Paz, and P. González, Eds. Cham: Springer International Publishing, 2019, pp. 88–96 [Online]. Available: http://link.springer.com/10.1007/978-3-319-98702-6_11. [Accessed: 22-Aug-2018]

[22] H. López-Fernández, M. Reboiro-Jato, D. Glez-Peña, R. Laza, R. Pavón, and F. Fdez-Riverola, "GC4S: A bioinformatics-oriented Java software library of reusable graphical user interface components," *PLoS ONE*, vol. 13, no. 9, p. e0204474, Sep. 2018, doi: 10.1371/journal.pone.0204474.

[23] G. Moreno-Hagelsieb and K. Latimer, "Choosing BLAST options for better detection of orthologs as reciprocal best hits," *Bioinformatics*, vol. 24, no. 3, pp. 319–324, Feb. 2008, doi: 10.1093/bioinformatics/btm585.

[24] J. Rozas *et al.*, "DnaSP 6: DNA Sequence Polymorphism Analysis of Large Data Sets," *Molecular Biology and Evolution*, vol. 34, no. 12, pp. 3299–3302, Dec. 2017, doi: 10.1093/molbev/msx248.

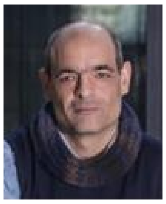
[25] S. Kumar, G. Stecher, and K. Tamura, "MEGA7: Molecular Evolutionary Genetics Analysis Version 7.0 for Bigger Datasets," *Molecular Biology and Evolution*, vol. 33, no. 7, pp. 1870–1874, Jul. 2016, doi: 10.1093/molbev/msw054.



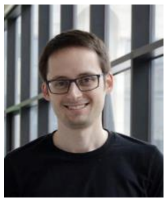
Hugo López Fernández received the BS degree in computer science from University of Vigo. He received the MS and PhD degrees from University of Vigo, where he is Postodctoral Researcher in the Next Generation Computer Systems group (http://sing-group.org/). As of April 2020, his 41 scientific contributions indexed in Scopus hold 209 citations from 180 different documents (h-index = 7).

Pedro Duque received the BS degree in Biology and the MS in Molecular and Cell Biology from University of Porto. He is a PhD student in the Phenotypic Evolution group at i3S (University of Porto; https://www.i3s.up.pt/research-group?x=44). As of April 2020, his 4 scientific contributions indexed in Scopus hold 7 citations from 7 different documents (h-index = 2).

Noé Vázquez received the BS degree in computer science from University of Oviedo. He received the MS and PhD degrees from University of Vigo. He is currently lead developer at INTECCA - UNED and tComet. As of April 2020, his 12 scientific contributions indexed in Scopus hold 15 citations from 11 different documents (h-index = 3).

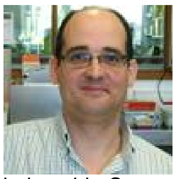


Florentino Fdez-Riverola received the BS degree in computer science from University of Oviedo. He received the MS and PhD degrees from University of Vigo, where he is Professor in the Department of Computer Science and Coordinator of the Next Generation Computer System group (http://sing-group.org/). He is author of 212 scientific contributions, of which more than one hundred were published in journals indexed in the SCI. His publications hold 1659 citations from 1300 different documents.



Miguel Reboiro-Jato received the BS degree in computer science from University of Vigo. He received the MS and PhD degrees from University of Vigo, where he is Associate Professor in the Department of Computer Science and Header Researcher at the Next Generation Computer System group (http://sing-group.org/). As of April 2020, his 66 scientific contributions indexed in Scopus hold 632 citations from 525 different documents (h-index = 12).

Cristina P Vieira received the BS degree in Biology from University of Porto, and the PhD in Plant Population Genetics from Edinburgh University. She is a senior researcher of the Phenotypic Evolution group at i3S (University of Porto; https://www.i3s.up.pt/research-group?x=44). As of April 2020, her 71 scientific contributions indexed in Scopus hold 937 citations from 645 different documents (h-index = 19).



Jorge Vieira received the BS degree in Biology, the MS in Apllied Ecology and the PhD in Biomedical Sciences from University of Porto. He is the coordinator of the Phenotypic Evolution group at i3S (University of Porto; https://www.i3s.up.pt/research-group?x=44). As of April 2020, his 85 scientific contributions indexed in Scopus hold 1193 citations from 812 different documents (h-index = 20).