Bioinformatics Advance Access published June 6, 2016

Bioinformatics, 2016, 1–8
doi: 10.1093/bioinformatics/btw302
Advance Access Publication Date: 13 May 2016
Original Paper

Genome analysis

SoFIA: a data integration framework for annotating high-throughput datasets

Liam Harold Childs[1,*], Soulafa Mamlouk[2], Jörgen Brandt[1], Christine Sers[2] and Ulf Leser[1]

[1]Wissenmanagement in der Bioinformatik, Humboldt-Universität zu Berlin, Berlin, Germany and [2]DKTK Deutsches Konsortium Für Translationale Krebsforschung, Partner site Charite Berlin, Berlin, Germany

*To whom correspondence should be addressed.

Associate Editor: Alfonso Valencia

Received on October 29, 2015; revised on April 13, 2016; accepted on May 6, 2016

Abstract

Motivation: Integrating heterogeneous datasets from several sources is a common bioinformatics task that often requires implementing a complex workflow intermixing database access, data filtering, format conversions, identifier mapping, among further diverse operations. Data integration is especially important when annotating next generation sequencing data, where a multitude of diverse tools and heterogeneous databases can be used to provide a large variety of annotation for genomic locations, such a single nucleotide variants or genes. Each tool and data source is potentially useful for a given project and often more than one are used in parallel for the same purpose. However, software that always produces all available data is difficult to maintain and quickly leads to an excess of data, creating an information overload rather than the desired goal-oriented and integrated result.
Results: We present SoFIA, a framework for workflow-driven data integration with a focus on genomic annotation. SoFIA conceptualizes workflow templates as comprehensive workflows that cover as many data integration operations as possible in a given domain. However, these templates are not intended to be executed as a whole; instead, when given an integration task consisting of a set of input data and a set of desired output data, SoFIA derives a minimal workflow that completes the task. These workflows are typically fast and create exactly the information a user wants without requiring them to do any implementation work. Using a comprehensive genome annotation template, we highlight the flexibility, extensibility and power of the framework using real-life case studies.
Availability and Implementation: https://github.com/childsish/sofia/releases/latest under the GNU General Public License
Contact: liam.childs@hu-berlin.de
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Biological entity annotation is the process of retrieving the broader biological context around a biological entity from available data sources. It is a crucial step in essentially all current biological analyses based on high-throughput experiments (Ng *et al.*, 2010; Wang *et al.*, 2010). Only by considering the biological context within which an experiment took place can the results be properly interpreted and a pertinent conclusion reached. As increasingly more knowledge about biological systems is discovered and stored in structured databases, the effort required to integrate all relevant information in projects producing comprehensive experimental datasets becomes more and more involved. Providing investigators all available information can lead to information overload, as such 'complete' annotation sets contain much more data than is relevant to the investigated hypothesis. There is a lack of fast and intuitive methods that (i) allow researchers to specify exactly the type of information that they think is best suited to their investigations and (ii) produce this information quickly and fully automatically.

Data integration for the life sciences is by no means a new topic (Blankenberg *et al.*, 2010; Cingolani *et al.*, 2012; Huang *et al.*, 2009a,b; McLaren *et al.*, 2010, McCarthy et al., 2014; Wang *et al.*, 2010)

Current approaches can be broadly categorized into three classes. So-called data warehouses are relational databases that integrate a selected set of data into a common schema (Kasprzyk, 2011; Trissl *et al.*, 2005). Accessing the data in a data warehouse requires either the ability to program complex queries (usually in SQL), or the usage of specific point-and-click user interfaces encapsulating such queries. The latter solution is the only option when programming expertise is lacking, but is inflexible and involves costly interface development. A second class of data integration systems are based on linked open data and Semantic Web standards (Livingston *et al.*, 2015; Machado *et al.*, 2013). These offer more flexibility in terms of data modelling, but require efforts comparable to data warehousing for building semantically integrated datasets (Bechhofer *et al.*, 2013). Both approaches perform data integration prior to any concrete analysis, which implies that they usually try to be as comprehensive as possible to cover unforeseen applications. Creating or updating this large integrated dataset is highly complex and time consuming, increasing the danger of using outdated data (Jörg and Dessloch, 2008; Lee *et al.*, 2006).

More recently, a third class of systems has emerged that are based on flexible integration workflows (Giardine *et al.*, 2005; McLaren *et al.*, 2010; Ríos *et al.*, 2009; Wolstencroft *et al.*, 2013). In these approaches, data integration is performed by starting a pipeline of steps that are defined in advance by a workflow developer. Results of these workflows are typically directly consumed by the user or by other tools and not meant to be materialized in a persistent, maintained manner. Accordingly, every analysis uses the most recent data available. To be fast, these workflows are specialized; a drawback when no available workflow exactly meets the user's requirements. Either a new workflow has to be developed, or multiple workflows with potentially overlapping subtasks have to be executed, yielding inflexibility and unnecessary computation. What is lacking is a data integration method that, based on a formalized understanding of an application domain, is able to automatically determine the minimal complete sequence of steps required to fulfil a given user request starting from a given set of input data.

SoFIA, Software for Flexible Integration of Annotation, aims at filling this gap. Using SoFIA, workflow designers specify comprehensive workflow templates covering as much of a given application domain as possible, much like defining the process used to populate a data warehouse. However, these templates are not intended to be executed in their entirety. Instead, they should be understood as a formalized knowledge base of processes transforming various types of input data into various types of annotations using background knowledge. When given a set of input data (e.g. results from wet lab experimentation) and a desired output (e.g. gene names related to the experimental data in a specific way), SoFIA uses the template to infer a minimal set of actions necessary to produce the output from the inputs. In case there is a unique way for doing so, the process runs fully automatically. We see that the SoFIA approach has the following important advantages:

a. It is fast, as only the necessary steps are executed;
b. It supports reproducibility, as users can choose which data versions will be integrated;
c. From a user's perspective, it is very simple to use, as only inputs and outputs have to be specified;
d. It relieves from the need to build and update a large, comprehensive database for unforeseen applications;
e. It clearly separates data provisioning from data transformation/annotation steps, which yields a clear workflow design.

We developed SoFIA specifically for performing multiple types of analysis in next generation sequencing (NGS), such as variant annotation, RNA-seq analysis, ChIP-seq analysis, or epigenetic analysis. Common aspects of these experiments are that they (i) map experimental read-outs to parts of a genomic sequence, (ii) require specific information on these genome parts to produce biologically meaningful results and (iii) multiple tools and databases exist for each step in such a pipeline. The design of SoFIA directly targets these requirements. In this paper, we present our framework for data integration, describe a template that is readily provided with the system, and demonstrate the flexibility and power of SoFIA by case studies in variant annotation for a breast cancer dataset, comparison of the impact of using different gene models, and an analysis of translation efficiencies using *E.coli* protein expression data.

2 Methods

2.1 Data model

SoFIA builds on a semi-structured entity model, as is common in many biological databases (Lacroix *et al.*, 2003). The most fundamental element is the entity type; a general term that can refer to biological concepts like genes, sequences or pathways, or even to single values like expression measurements or variant frequencies. Entity types may be nested using both a 'has-a' hierarchy that allows automatic entity extraction (e.g. a *variant* 'has a' *alternate allele*, and the presence of a *variant* implies the presence of an *alternate allele*), and an 'is-a' hierarchy that allows more specialized entities to be used in place of their more general predecessors (e.g. a *variant* 'is a' *genomic position*, and can be treated as a *genomic position*). An example can be seen in Figure 1 (Note that our model is very close to the popular JSON format.). Any concrete entity or value is an instance of exactly one entity type, which determines its schema.

To keep the framework simple, we do not enforce any consistency constraints; for instance, if a given gene (specified by an identifier) appears twice in different contexts, we do not ensure that these two instances refer to the same, single object or that the attribute values associated to the two instances are identical. Instead, we allow special attributes (specified as key-value pairs) to be assigned to entities for tracking meta-information on their origin and the way they were created. This allows workflow resolution to 'react' to specific attributes and adjust the resolved workflow accordingly (example in Section 2.4 below).

Entities may be derived from external sources, such as a database, or are an intermediate product of an integration workflow. Thus, in principle, entities could be accessed remotely, for instance through a web service. Our implementation, SoFIA, currently assumes all sources are locally available as downloaded database files. This requires a certain amount of storage space, but improves access speeds and control over versioning. We keep these files up-to-date with CRON jobs.

2.2 Data integration processes

We model all types of information as entity types, including numeric attributes such as start or stop positions of a genomic interval. SoFIA interprets entity integration as the process of finding

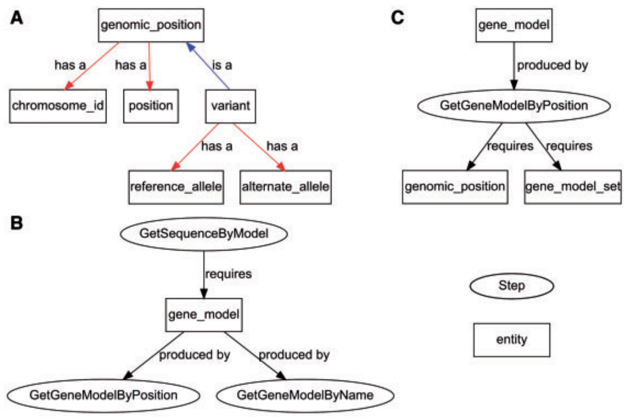

Fig. 1. Data model. SoFIA uses a bipartite graph to model the process of data integration where the two partitions are used to represent the entities (rectangles) and steps (ovals). (**A**) The relationships among entities is modelled using 'is-a' and 'has-a' relationships permitting a significant degree of expression with few concise terms. These relationships specify the minimum fields needed for a workflow to process an entity. Thus, even if a *variant* can be described by more data, only the *chromosome id*, *position*, *reference allele* and *alternate allele* will be used in the example definition. (**B**) Entities are both produced and consumed by steps, referred to as the preceding and succeeding steps. Only one preceding step needs to be executed for an entity to be considered available, upon which it is available to all succeeding steps. (**C**) Steps both consume and produce entities. All preceding entities must be available for a step to be executed, upon which all succeeding entities are considered available

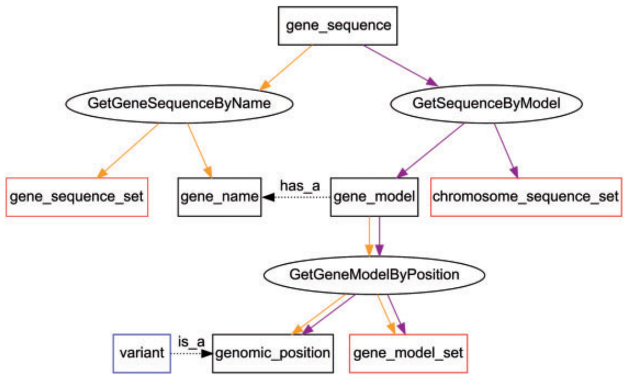

Fig. 2. Example template. This is example template can be used to obtain a gene sequence from a variant. Two possible solutions are possible, shown as solid and dashed paths, depending on the entities provided. A *gene sequence set*, *chromosome sequence set* or *gene model set* can be provided by the user. The target entity is the *variant*. The solid solution is only possible if a *gene sequence set* is provided and the dashed solution is only possible if both a *gene model set* and a *chromosome sequence set* is provided. Any other combination will lead to ambiguous or incomplete solutions. Ambiguous solutions can be resolved by specifying which entities the gene sequence must be derived from; either a gene sequence set or either of the gene model set or a chromosome sequence set

associations with other entities, possibly through a chain of intermediates. Such chains are defined on entity types, i.e. every step in such a chain associates one set of entities with another. For example, if the pathway affected by a mutation is sought, one can model the chain of associations as (i) a step associating the genomic position of mutations to one or more genes and (ii) a step associating genes to one or more pathways.

Such processes can be naturally modelled as directed acyclic bipartite graphs, where the nodes of one partition are the entity types (entity nodes) and the nodes of the other partition are the steps associating entity types with each other (step nodes). Step nodes are connected to the entity nodes they produce and the entity nodes they consume; in turn, entity nodes are connected to the step nodes that can produce them and those that can consume them (Fig. 2). Furthermore, an entity node may be a source (the input data) or a sink (a result not further worked upon). Within this general model, SoFIA enforces the following constraints defining a clear and simple execution semantic: (i) Step nodes have at least one incoming edge from entity nodes, its predecessors. For a step node to be executed, all predecessors must be available. (ii) Step nodes have one or more outgoing edges to entity nodes, its successors. If a step node is executed all successors are available upon completion. (iii) All entity nodes except source nodes have at least one incoming edge from step nodes. Unlike step nodes, the execution of only one predecessor is necessary to ensure the availability of an entity; if multiple preceding step nodes exist, the entity node is called ambiguous and any of these predecessors may be executed to produce the entity type. Otherwise it is called unique. (iv) All entity nodes except sink nodes have one or more outgoing edges to step nodes.

Attributes are optional and can be defined on any entity including the input and output entities of intermediate steps. This can have the effect of restricting connections between steps or to allow the workflow resolution algorithm to react to properties resulting from executing particular steps. For example, some variant callers may only be compatible with certain read aligners, or a workflow may need to include extra steps if a particular attribute is not defined.

Every such bipartite graph, together with a model of the entity types it contains, is called a workflow template. An example workflow is shown in Figure 2.

2.3 Workflow resolution

Templates are intended to be comprehensive for a given application domain meaning that they should be designed by an expert with the goal to fulfil as many user needs as possible. Having a large template carries no disadvantages as it is not executed as a whole; instead, it is used as a blueprint to derive a concrete workflow execution plan for a given user-specified task. This process is called workflow resolution. Resolving a workflow in SoFIA is simple: a user defines an annotation task by indicating which entity types are available (referred to as the *provided entities* and requesting the types of entities they wish to derive (the *requested entities*).

With this information, SoFIA infers a minimal series of steps that produces all requested entities from the provided entities. To this end, the template is traversed by attempting to find a resolution for each requested entity in turn. Entity nodes are considered resolved if *any* of the preceding step nodes can be resolved, possibly leading to multiple solutions if multiple step nodes can be resolved. Step nodes are considered resolved if *all* of the preceding entity nodes can be resolved and all combinations of the solutions from the preceding entity nodes are evaluated. The potentially very large number of possible solutions is limited by applying a filtering heuristic at each node that permits only the solutions that require the least number of extra provided entities. The solutions for each requested entity are merged and the smallest workflow is used as the resolved workflow (RW); a minimal workflow computing all request output data. If there are multiple, equally small RWs with equally few extra entities, the user is prompted for additional information.

Using the template in Figure 2 as an example, consider finding the *gene sequence* for the gene affected by a *variant*, a common, multi-step annotation task. There are two possibilities starting after a *gene model* is found using the *variant's* position. (i) Assemble the *gene sequence* using the exon information from the *gene model* and a *chromosome sequence set*. (ii) Get the *gene sequence* from a *gene sequence set*, using the *gene model's gene name*. Assuming the user provides a set of *variants* for annotation and a *gene model set*, and specifies the *gene sequence* as the requested entity, a resolution algorithm will first attempt to resolve the *gene sequence* entity node. This will trigger a recursive descent through the graph until all nodes are visited. The traversal will fork at the *gene sequence* entity node to explore the two steps that can produce it. In the left branch, the traversal will find that it requires a *gene sequence set*. If provided by the user, along with a *gene model* set this will yield a valid solution. Otherwise the absence is noted and traversal of this branch is aborted. A similar scenario occurs in the right branch yielding a valid solution if the user provided a *chromosome sequence set*.

2.4 Attribute resolution

During workflow resolution, the resolution algorithm implemented in SoFIA can react to entity attributes and adjust the RW accordingly. This is used to trigger implicit computational steps. One example of such steps is the ability to perform automatic identifier conversion working as follows: During resolution, the attributes of the incoming entities are compared. If the entities have matching attributes but different attribute values, a conversion step can be inserted that uses a lookup table to convert one of the values to match the other. We currently produce such tables using BioMart (Kasprzyk, 2011) and CRONOS (Waegele *et al.*, 2009). Similarly, attributes can be used to define which provided entities a requested entity must be derived from.

2.5 Workflow execution

SoFIA executes a RW starting from the source nodes. First, step nodes depending only on source nodes are executed producing new entity sets thus enabling new step nodes to be executed and so on. This process continues until all desired output nodes have been computed.

2.6 Implementation and extensibility

The framework was implemented as a Python program. The program is divided into three main parts, template definition, workflow resolution and workflow execution. Templates are defined by writing Python classes that represent steps in a workflow. In each class, the incoming and outgoing entities are declared and the calculations needed to perform the step are defined via calls to external tools or coded within the step itself. Steps are connected to each other by connecting outgoing entities to incoming entities where the entity types match. To extend an existing template, a new step is written with incoming and outgoing entity types compatible to the existing template.

When implementing SoFIA, we followed a 'one input – one output' strategy to aid interpretability, enable data-dependent parallelization and reduce memory footprint. Execution of SoFIA requires the user to designate one source as the 'target'. The requested entities computed are then output in a tabular format with one column per requested entity and one row per entity in the target source. Optionally, users can specify their own output format enabling the production of *fasta* files or *vcf* files. Given this rule, if the user designates a *vcf* file as the target, then one line of output is produced for each line in the target containing a variant in a direct one-to-one relationship leaving little-to-no room for confusion.

To give users greater control over versioning and improve access speeds all data used by SoFIA must be local either as files database downloads or provided by the user (experimental results and parameters). Further documentation on using and extending the framework is available in the supplementary and on the project home page: https://github.com/childsish/sofia.

3 Results

We developed SoFIA, a framework that models biological entity annotation as an integration workflow that associates provided input data from high-throughput experiments to requested annotations with little effort on the part of the user. SoFIA is freely available and comes with a comprehensive template for annotating genomic data which we demonstrate here with three real-life use cases: annotating sequence variants, comparing gene model consequences and calculating sequence features that purportedly estimate translational efficiency.

3.1 Genomics workflow template

We created a comprehensive workflow template that covers many aspects of genomics data integration including (i) interval manipulation and querying, (ii) sequence extraction, transcription and translation, (iii) sequence feature calculation and (iv) sequence variant manipulation and categorization (Supplementary Material 1). The template makes use of tabix (Li, 2011) from htslib (Li *et al.*, 2009) and a custom library to index source files dramatically improving annotation speed. Basic caching has also been implemented for sources, also improving speed when several consecutive queries return the same result.

3.2 Sequence variant annotation

An area where entity annotation is of utmost importance is large-scale genomic sequencing, which is typically performed to derive genomic variations whose interpretation critically depends on proper annotations (Cingolani *et al.*, 2012; McLaren *et al.*, 2010; Wang *et al.*, 2010). Our genomics template follows the guidelines set out by the Human Variation Nomenclature Society for variant descriptions (Den Dunnen and Antonarakis, 2000) wherever possible and applies the sequence ontology to describe variant effects (Eilbeck *et al.*, 2005).

We showcase the capability and flexibility of our framework by annotating breast cancer (BRCA) variants for three theoretical labs with different requirements. One lab is interested finding which variants could be natural variants from the 1000 genomes project (1000 Genomes Project Consortium *et al.*, 2015) and dbSNP (Sherry *et al.*, 2001), another is interested in finding which variants exist in cancer variant databases such as COSMIC (Forbes *et al.*, 2015) and TCGA (Stratton *et al.*, 2009) and the final lab is interested in finding which drugs from DrugBank (Law *et al.*, 2014) or Genomics of Drug Sensitivity in Cancer (Yang *et al.*, 2013) could be used to target affected genes. Downloading and converting BRCA into variant call format resulted in 43 278 unique sequence variant positions (Supplementary Data 3). SoFIA was provided the BRCA dataset, Gencode gene models, the hg19 reference genome, PROVEAN estimates of variant impact (Choi *et al.*, 2012) and KEGG pathways (Ogata *et al.*, 1999), and was requested to produce gene names, amino acid variants, variant types, gene annotations, annotated pathways, homopolymer regions, overlaps with genomic regions of interest and variant impact on protein function for all three theoretical labs. In addition, the extra entities and resources required for

Table 1. Run-times for each SoFIA request. SoFIA demonstrates rapid flexibility when handling different requests

Annotation purpose	Extra data sources	Workflow resolution time	Workflow execution time	Output size
Natural variants	1000 genomes dbSNP	2.1 s	14 min 5 s	3.4 Mb
Cancer mutants	COSMIC TCGA	3.1 s	9 min 56 s	3.9 Mb
Drug response	DrugBank GDSC	2.3 s	9 min 28 s	3.5 Mb

Three requests for different types of data were tested and the workflow resolution and executions times were measured. In each encounter with a new data and request, SoFIA can quickly resolve a suitable workflow within a matter of seconds and annotate the entire dataset in under 15 min. In contrast, the creation of a new workflow or adaptation of an existing workflow by hand is likely to take significantly longer.

each labs individual needs were provided individually for each lab's request. Each run took approximately 15 min on a 2.6 GHz Pentium i5 CPU, without parallelization. The command line used to run SoFIA and a summary of all annotations produced by the resolved workflows are provided in Supplementary Material 2 and Supplementary Data 4–6, respectively.

This example shows that SoFIA exhibits remarkably quick adaptability to new data and requests. In each case, a sensible workflow is resolved in a matter of seconds saving the time required to create a new workflow or adapt an existing one by hand. In each case several non-trivial steps are automatically resolved including matching variants from multiple variant file formats (.vcf and .maf), interval queries and several identifier conversion steps, Overall, SoFIA greatly simplifies many otherwise tedious tasks (Table 1).

3.3 Gene model comparison

The flexible nature of the framework makes it easy to compare similar information from multiple sources. Choosing the most suitable gene models is a very real and concrete problem many bioinformaticians face when analysing NGS datasets (McCarthy *et al.*, 2014). We used our genomics template to compare the impact of gene models from different sources on variant annotation. SoFIA was provided RefSeq gene models from NCBI (Pruitt *et al.*, 2014) and Gencode gene models (Harrow *et al.*, 2006) from the Encode project along with the BRCA dataset and the hg19 human genome sequence from the previous example, and was requested to output all gene identifiers as RefSeq identifiers to facilitate a direct comparison (Supplementary Material 2). The resolved workflow associated BRCA variants with gene models from RefSeq and Gencode, respectively, found amino acid variants for both sources as well as the variant effect. SoFIA seamlessly integrated the two different file formats provided by RefSeq and Gencode to provide the requested annotations (Supplementary Data 7).

A comparison of the two resulting annotation sets (on the same input) shows that there exist quite a few genes that show largely different annotation depending on which gene model was used, although the majority of variants are annotated in the same way by both models. Generally, variants were more often labelled missense variants using RefSeq than with Gencode. In contrast, Gencode had a much higher proportion of intron variants (Fig. 3A). We took a closer look at the genes with greater than 10 variant annotation differences (Fig. 3B). Notably, RefSeq and Gencode sometimes use very different gene names for the same genes (e.g. DSSP versus RP11-742B18, MYH8 versus RP11-799N11, PKD1L1 versus HUS1 and MYH1 versus RP11-799N11). Occasionally, variants belonging to one gene in Gencode are found in two genes in RefSeq (FLG and HRNR versus FLG-AS1, MYH8 and MYH1 versus RP11-799N11). Furthermore, many variants annotated as missense and synonymous by RefSeq are annotated as intronic by Gencode hinting at a higher number of exons in the RefSeq models. Only one gene shows the opposite. In two cases a gene in one dataset is completely missing in

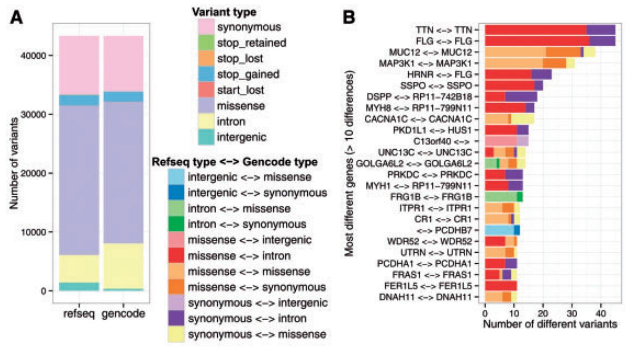

Fig. 3. Comparison of gene model providers. By and large the majority of variants get identical annotation. (**A**) However, Gencode shows a greater number of intron variants and a smaller number of missense variants when compared to RefSeq. The proportions of other types of variant remain roughly the same. (**B**) The most affected genes (>10 differences) show a similar story. Eleven genes are fully annotated in RefSeq with missense and synonymous variants but only appear as intron variants in Gencode. Only one gene shows the reverse case. Ten genes show a disagreement between missense variants and each provider misses one annotated gene each. By and large, the providers use similar names for the genes but in some cases the names used are distinctly different

the other (C13orf40 in RefSeq and PCDHB7 in Gencode). Furthermore, many missense variants in both datasets do not show the same amino acid change.

3.4 Sequence feature annotation

Another task for which SoFIA is well suited is the generation of sequence features derived from mRNA or protein sequences, such as nucleotide or amino acid frequencies. Sequence features are commonly computed to study the influence of intrinsic sequence properties on transcriptional and translational efficiency and various features have been found to have significant correlations (Navon and Pilpel, 2011; Sharp and Li, 1987).

Using SoFIA, we correlated previously published protein expression data (Lu *et al.*, 2007) with various sequence features that have been proposed to have correlation with protein expression in different systems including (i) the codon adaptation index (CAI) (Navon and Pilpel, 2011; Sharp and Li, 1987), a measure of how closely a given gene's codon usage matches that of known highly expressed genes, (ii) the effective number of codons (Sun *et al.*, 2013), an alternative to the CAI that measures the diversity of codons being used in a gene, (iii) the number of PEST sequences, sequences rich in the amino acids proline, glutamic acid, serine and threonine, which are associated with a short intracellular half-life, (iv) the number of upstream ORFs, of which a high number is expected to interfere with translation initiation and (v) the minimum free energy of the start codon RNA secondary structure, a measure of how much RNA structure interferes with translation initiation (Scharff *et al.*, 2011;

Tuller *et al.*, 2010) (command line in Supplementary Material 2, output in Supplementary Data 8).

As expected, the results (Fig. 4) show significant correlations. Specifically, mRNA expression correlates better than any of the sequence features, which correlate in order from best to worst starting with the effective number of codons followed by the codon adaptation index, the number of PEST sequences, the number of upstream ORFs and ending with the minimum free energy of the translation start site. Note that the resolved SoFIA workflow altogether combines two data sources with sixteen computational steps, producing twelve intermediate entity types.

4 Discussion and conclusion

SoFIA is a framework for entity annotation building on a comprehensive domain-specific data transformation and annotation accession model encoded in the form of a workflow template. Concrete integration pipelines that annotate a given input with the requested information are automatically inferred from this model. Conceptually, our approach can be seen as a hybrid between data warehouses, from which it inherits its ability to encode complex integration logic, and federated databases, from which it takes its on-demand execution strategy. This novel architecture brings several advantages. First, it allows for the separation of concerns where analysis specialists can focus on building specialized tools, which domain specialists assemble into workflow templates, which in turn are used by end users who do not need to worry about how their required information can be produced. Second, it leads to fast annotation as only the minimally necessary integration and transformation steps are executed. This speed allows for up-to-date results, as it costs little to rerun a pipeline after updating data sources. Third, workflows are naturally modular, which increases maintainability and extensibility of templates. Fourth, SoFIA generates complete provenance traces for any annotation it produces, an important cornerstone to achieve reproducibility and transparency of scientific studies and a topic of growing concern in the community (Nekrutenko and Taylor, 2012). Fifth, it enables researchers to quickly generate their own personalized view into their data by focussing specifically on the details that interest them. Sixth, the approach is rather robust, as, for a concrete integration goal, it is not necessary that every step of the template is executable. If a part breaks that is used only occasionally, for instance due to format changes in a data sources, most usages of the embracing template will not be affected.

The flexibility that this approach provides should also prove useful when new experimental techniques and new biological entities emerge from technological advances in research. At some point in the past, the concept of a *variant* was also new and once the technology to call them in vast amounts was developed, several new tools were developed to annotate them. Due to the extensibility of workflow templates, it would take little effort to incorporate any new biological entity.

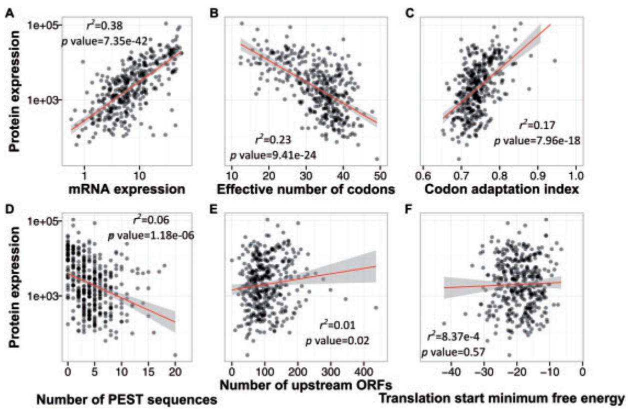

Fig. 4. Sequence feature correlation with translation efficiency. Sequence features for all *E.coli* genes were calculated using SoFIA and correlated with previously published protein expression. The results show significant correlation for four features (effective number of codons, codon adaptation index, number of PEST sequences and number of upstream ORFs) after multiple testing correction using the Benjamini–Hochberg method

4.1 Comparison to stand-alone tools

Stand-alone programs such as Variant Effect Predictor (McLaren *et al.*, 2010), Annovar (Wang *et al.*, 2010), snpEff (Cingolani *et al.*, 2012) or DAVID (Huang *et al.*, 2009a,b) often provide convenient graphical user interfaces, but can be inflexible as they are restricted to a predefined tool set and produce a fixed output (McCarthy *et al.*, 2014). This places two major restrictions on their use when compared to SoFIA: First, it is difficult for a researcher to create tailored annotation for their data resulting in the inclusion of extraneous, and the exclusion of pertinent, information. In contrast, SoFIA is purpose built for such an on-demand generation of annotations. Second, these tools are typically developed as monolithic applications and changing or exchanging parts of them that become outdated by new technologies and data can only be accomplished by the original developers. This is very different for the modular approach, as implemented in workflow systems like SoFIA, where every data source or tool can be replaced with comparable ease (note that format adaptations sometimes become necessary). These issues emphasize the advantages of an open and modular framework that places control in the hands of a community, a role we hope SoFIA can fulfil. Here, we want to re-emphasize that SoFIA is not restricted to annotating variants, but can link any entities defined within the same template a task that is currently outside the scope of existing stand-alone tools.

4.2 Comparison to scientific workflow systems

SoFIA essentially consists of three parts: A workflow execution system, a workflow specification language and templates for deriving concrete workflows. The first two components are at the core of any scientific workflow execution system. For instance, Taverna workflows are specified using SCUFL (Oinn *et al.*, 2004) and executed on a custom execution engine. Galaxy workflows are graphically designed and encoded in a custom XML format (Giardine *et al.*, 2005); their execution can be performed locally or with compute clusters using CloudMan. SAASFEE (Bux *et al.*, 2015) accepts workflows specified in Cuneiform, a functional dataflow language, and executes them on Apache Hadoop. In principle, any of these systems could also be used as basis for SoFIA, which would require the compilation of resolved workflows in scripts of the respective language or format. Actually, we believe this is an interesting venue for future research; in our vision, SoFIA would focus on template provisioning and resolution, while other frameworks are used for robust and fast workflow execution.

From the user perspective, the addition of a template to workflow systems introduces the ability to produce workflows on demand. Previously, obtaining a workflow for a given problem required either the creation of an entirely new workflow, the adaptation of an old one, or the download of a workflow from an open workflow repository (Goderis *et al.*, 2008; Starlinger *et al.*, 2016). While the former two approaches require considerable investment of time, the latter can be faster if an appropriate workflow is present. With a template, a workflow system is able to quickly adapt to a multitude of different requests as demonstrated in the first example. This situation arises particularly often during biological annotation and we specifically sought to support this through the development of SoFIA. However, the SoFIA approach to workflow resolution also has a number of disadvantages when compared classical workflow systems that directly execute a given specification: (i) SoFIA does not optimally support large parameter sweeps or interactive configuration through parameterization. Defining parameters for specific steps using SoFIA is possible through the use of entity attributes, however, this approach may prove unwieldy. (ii) Before executing a workflow, SoFIA must resolve it, and the generated workflow will not be optimized for performance. Thus, when high throughput is required or a workflow is run hundreds of times on different data, a system which allows for specialized performance optimization is in order.

4.3 SoFIA design choices

When developing SoFIA we chose to follow the aforementioned ‘one input – one output’ strategy that brings multiple advantages and disadvantages. One advantage is that the strategy aids interpretability. If provided a set of variants for instance, for each variant a single line of output is produced. Additional or fewer lines may result in a perceived discrepancy between the number of input and output variants. In addition, the strategy enables data-dependent parallelization meaning each entity within the target source can be processed in parallel, improving compute time and reducing the memory footprint. Among the disadvantages is limited multiplicity capability. Although a step can output multiple entities, the following step must implement its own way to handle this. Where this applies, this typically results in a comma separated field within the tab-delimited output. Furthermore, it becomes difficult to implement whole set operations such as calculating the rank of an entity from some ordering of its set.

The heuristic used by SoFIA when resolving workflows provides multiple benefits. Most importantly, extraneous information that was not requested by the user will not be included. For example, if the user provides a variant set and requests the variant quality, it would make little sense to first find a matching variant from another set and provide the quality of that variant. Thus, the heuristic prevents the incorporation of irrelevant information that would give incorrect results. The information provided to a user is dependent on the formulation of the problem (the requested and proved entities). For a properly formulated problem, there is only one solution. If improperly formulated, there are either no solutions or multiple solutions, and SoFIA will inform the user which case their problem falls under and give suggestions about how the problem may be better formulated. The heuristic employed may miss the most optimal workflow satisfying the problem instead. An additional benefit of the heuristic is faster workflow resolution and smaller workflows.

The genomics workflow was implemented to rely completely on local sources, introducing a number of benefits and drawbacks. We made this restriction to give the user complete control over data versioning and to reduce concerns about data privacy. However, this choice introduces the need for local space and manual updating. The exact amount of space required depends heavily on the sources that the user wants to provide to SoFIA. However, the minimum required space for basic annotation (gene name, amino acid variant and variant type) is roughly 3 GB. As data continues to grow, local storage may become impractical and SoFIA is fully capable of performing operations remotely via the Python urllib library, potentially mitigating this outcome. Regardless, we envisage the main users of SoFIA to be embedded bioinformaticians in small to medium sized research groups where access to core facilities and highly parallel computing may be limited, but data demands are also low.

4.4 Conclusion

We strongly believe that supporting individualized data annotation is an important yet often neglected matter. Even when using the same type of data, two different groups may have different annotation needs depending on their overall research goal. For instance, while some groups may seek information from drug response databases such as DrugBank (Law *et al.*, 2014) or context specific genetic dependencies such as Achilles (Cowley *et al.*, 2014), others may be more interested in the population genetics which requires comparing their variant set to those from projects like the 1000 Genomes Project (1000 Genomes Project Consortium *et al.*, 2015) or dbSNP (Sherry *et al.*, 2001). A third group might study the mechanistic effects of variants for particular types of diseases, which requires them to integrate data from diseases-specific sequencing panels like The Cancer Genome Atlas (Stratton *et al.*, 2009) or the International Cancer Genome Consortium (Husdon, 2011). Supporting all such needs with a single, up-to-date data warehouse requires complex schemas, costly maintenance operations, and continuous systems re-engineering to properly integrate novel datasets and types into the schema. The same extensions in SoFIA would require only programming a series of data transformations, which are later only executed on-demand.

Funding

This work has been supported by the German Ministry for Research and Education, collaborative research grant ONCOPATH.

Conflict of Interest: none declared.

References

1000 Genomes Project Consortium. *et al.* (2015) A global reference for human genetic variation. *Nature*, **526**, 68–74.

Bechhofer,S. *et al.* (2013) Why linked data is not enough for scientists. *Future Generat. Comput. Syst.*, 599–611.

Blankenberg,D. *et al.* (2010) Galaxy: a web-based genome analysis tool for experimentalists. *Curr. Protoc. Mol. Biol*, **89**, 1–21.

Bux,M. *et al.* (2015) SAASFEE: scalable scientific workflow execution engine. *Very Large Data Bases*, **8**, 1892–1895.

Choi,Y. *et al.* (2012) Predicting the functional effect of amino acid substitutions and indels. *PLoS One*, **7**, e46688.

Cingolani,P. *et al.* (2012) A program for annotating and predicting the effects of single nucleotide polymorphisms, SnpEff: SNPs in the genome of *Drosophila melanogaster* strain w 1118; iso-2; iso-3. *Fly (Austin)*, **6**, 80–92.

Cowley,G.S. *et al.* (2014) Parallel genome-scale loss of function screens in 216 cancer cell lines for the identification of context-specific genetic dependencies. *Sci. Data*, **1**, 140035.

Den Dunnen,J.T. and Antonarakis,S.E. (2000) Mutation nomenclature extensions and suggestions to describe complex mutations: a discussion. *Hum. Mutat.*, **15**, 7–12.

Eilbeck,K. *et al.* (2005) The sequence ontology: a tool for the unification of genome annotations. *Genome Biol.*, **6**, R44.

Forbes,S.A. *et al.* (2015) COSMIC: exploring the world’s knowledge of somatic mutations in human cancer. *Nucleic Acids Res.*, **43**, D805–D811.

Giardine,B. *et al.* (2005) Galaxy: a platform for interactive large-scale genome analysis. *Genome Res.*, **15**, 1451–1455.

Goderis,A. *et al*. (2008) Discovering Scientific Workflows: The myExperiment Benchmarks.

Harrow,J. *et al*. (2006) GENCODE: producing a reference annotation for ENCODE. *Genome Biol*., **7**, S4.1–S9.

Huang,D.W. *et al*. (2009a) Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists. *Nucleic Acids Res*., **37**, 1–13.

Huang,D.W. *et al*. (2009b) Systematic and integrative analysis of large gene lists using DAVID bioinformatics resources. *Nat. Protoc*., **4**, 44–57.

Husdon,T.J. (2011) International Cancer Genome Consortium. *Cancer*, **2011**, 1–20.

Jörg,T. and Dessloch,S. (2008) Towards generating ETL processes for incremental loading. In: *Proceedings of the 2008 international symposium on Database engineering & applications – IDEAS '08*, p. 101.

Kasprzyk,A. (2011) BioMart: Driving a paradigm change in biological data management. *Database*, **2011**, bar049.

Lacroix,Z. *et al*. (2003) Bioinformatics: Managing Scientific Data.

Law,V. *et al*. (2014) DrugBank 4.0: Shedding new light on drug metabolism. *Nucleic Acids Res*., **42**, D1091–D1097.

Lee,T.J. *et al*. (2006) BioWarehouse: a bioinformatics database warehouse toolkit. *BMC Bioinformatics*, **7**, 170.

Li,H. (2011) Tabix: fast retrieval of sequence features from generic TAB-delimited files. *Bioinformatics*, **27**, 718–719.

Li,H. *et al*. (2009) The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, **25**, 2078–2079.

Livingston,K.M. *et al*. (2015) KaBOB: ontology-based semantic integration of biomedical databases. *BMC Bioinformatics*, **16**, 126.

Lu,P. *et al*. (2007) Absolute protein expression profiling estimates the relative contributions of transcriptional and translational regulation. *Nat. Biotechnol*., **25**, 117–124.

Machado,C.M. *et al*. (2013) The semantic web in translational medicine: current applications and future directions. *Brief. Bioinform*., **16**, 89–103.

McCarthy,D.J. *et al*. (2014) Choice of transcripts and software has a large effect on variant annotation. *Genome Med*., **6**, 26.

McLaren,W. *et al*. (2010) Deriving the consequences of genomic variants with the Ensembl API and SNP Effect Predictor. *Bioinformatics*, **26**, 2069–2070.

Navon,S. and Pilpel,Y. (2011) The role of codon selection in regulation of translation efficiency deduced from synthetic libraries. *Genome Biol*., **12**, R12–R12.

Nekrutenko,A. and Taylor,J. (2012) Next-generation sequencing data interpretation: enhancing reproducibility and accessibility. *Nat. Rev. Genet*., **13**, 667–672.

Ng,S.B. *et al*. (2010) Exome sequencing identifies the cause of a mendelian disorder. *Nat. Genet*., **42**, 30–35.

Ogata,H. *et al*. (1999) KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res*., **27**, 29–34.

Oinn,T. *et al*. (2004) Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, **20**, 3045–3054.

Pruitt,K.D. *et al*. (2014) RefSeq: an update on mammalian reference sequences. *Nucleic Acids Res*., **42**, D756–D763.

Ríos,J. *et al*. (2009) Magallanes: a web services discovery and automatic workflow composition tool. *BMC Bioinformatics*, **10**, 334.

Scharff,L.B. *et al*. (2011) Local absence of secondary structure permits translation of mRNAs that lack ribosome-binding sites. *PLoS Genet*., **7**, e1002155.

Sharp,P.M. and Li,W.H. (1987) The codon adaptation index-a measure of directional synonymous codon usage bias, and its potential applications. *Nucleic Acids Res*., **15**, 1281–1295.

Sherry,S.T. *et al*. (2001) dbSNP: the NCBI database of genetic variation. *Nucleic Acids Res*., **29**, 308–311.

Starlinger,J. *et al*. (2016) Effective and efficient similarity search in scientific workflow repositories. *Future Gener. Comput. Syst*., **56**, 584–594.

Stratton,M.R. *et al*. (2009) The cancer genome atlas. *Nature*, **458**, 719–724.

Sun,X. *et al*. (2013) An improved implementation of effective number of codons (NC). *Mol. Biol. Evol*., **30**, 191–196.

Trissl,S. *et al*. (2005) Columba: an integrated database of proteins, structures, and annotations. *BMC Bioinformatics*, **6**, 81.

Tuller,T. *et al*. (2010) Translation efficiency is determined by both codon bias and folding energy. *Proc. Natl. Acad. Sci. USA*, **107**, 3645–3650.

Waegele,B. *et al*. (2009) CRONOS: The cross-reference navigation server. *Bioinformatics*, **25**, 141–143.

Wang,K. *et al*. (2010) ANNOVAR: functional annotation of genetic variants from high-throughput sequencing data. *Nucleic Acids Res*., **38**, e164.

Wolstencroft,K. *et al*. (2013) The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. *Nucleic Acids Res*., **41**, W557–W561.

Yang,W. *et al*. (2013) Genomics of Drug Sensitivity in Cancer (GDSC): a resource for therapeutic biomarker discovery in cancer cells. *Nucleic Acids Res*., **41**, D955–D961.