Throughput and robustness of bioinformatics pipelines for genome-scale data analysis

Paweł Sztromwasser

Dissertation for the degree of Philosophiae Doctor (PhD)

Department of Informatics
University of Bergen

2014

Scientific environment

The work presented in this thesis was funded by a PhD grant from the University of Bergen, and carried out at the *Department of Informatics* (II) of the *University of Bergen*, and at the *Computational Biology Unit* (CBU) of the *Uni Computing* department of *Uni Research AS*. I worked in a close collaboration with the eSysbio project managed through *Uni Computing*, and funded by the *Norwegian Research Council*. I was also affiliated with the *Molecular and Computational Biology Research School* at the *University of Bergen*. My work was supervised by Dr. Kjell Petersen employed at CBU, who was assisted by two co-supervisors: Dr. Pål Puntervoll at CBU, and professor Inge Jonassen at II and CBU.

My affiliation with the eSysbio project is important for the scope of contributions presented in this thesis. In the eSysbio project, we explored the possibility of using standard Web service technologies for building a virtual workspace for collaborative systems biology research. One of the open questions was applicability of the Web service technology to high-throughput data analysis. This thesis attempts to answer this question, and also promote the potential of Web services for building bioinformatics infrastructure.

Acknowledgements

Although the cover page of this thesis bears only my name, there is a number of people that deserve a share of the credit. Here is a non-exhaustive list of those that supported me with their knowledge, advice, and heart during the last five years.

First, and foremost I would like to thank my supervisor, Kjell Petersen, for the excellent work he has done mentoring me in the course of the studies. I appreciate the independence I was provided with, the systematic and dedicated supervision, and the trust in the final success. Kjell, you have an enormous capacity to manage multiple projects, and despite a frequently busy schedule, you were always able to find time to discuss with me. Even when I stormed into your office and flushed a torrent of unstructured thoughts, you were always able to make sense of it. And most remarkably, you were able to restore my hopes and trigger new ideas for attacking a problem, every time I nearly gave up on solving it.

I owe my deepest gratitude to Inge Jonassen, both for the active co-supervision of my work, and for being an excellent leader for the CBU, making it such a great workplace. Inge, your overview of the diverse areas in bioinformatics and detailed knowledge about them has impressed me from the beginning. Above all, thank you for replying to my email "a while" back. If it were not for this email, I might have never had the opportunity to come to the beautiful Bergen and start this adventure. A great deal of credit goes to my co-supervisor Pål Puntervoll, who has often surprised me with outside-the-box thinking, and the ability to take a step back and reconsider the choices made. Pål, thank you for introducing me to the world of Web services, and for showing that a concise and informative abstract can be made more concise and more informative at the same time.

I would like to thank all the present and former colleagues at CBU for creating such a great working environment, and for the after-work retreats in the fjords and mountains. My colleague and office-mate, Sattanathan, for the productive collaboration and many interesting discussions ranging from service computing to house renovation. The entire eSysbio team consisting of Anne-Kristin, Armin, Håkon, Francisco, Inge, Kidane, Kjell, Matuš, Michi, Prabu, Pål, Sattanathan, and Siv, for the spirit and mutual help in the struggle with the Web service technology. I am deeply grateful to professor Ingvar Eidhammer, and my colleagues Sandhya and Matuš, who all reviewed this thesis on a very short notice, and provided highly valuable input. Also, many thanks to the members of the Norwegian Microarray/Genomics Consortium in Bergen, where I spent a quarter of my work-time during the studies. I hope that I was of more help than trouble:)

Many of the projects carried out as part of this thesis required heavy computing. I am grateful for the access to the high-performance computing resources, and the support I received from the engineers at Parallab, NOTUR, and CSC in Helsinki. Saerda, you can breathe easy, I am not planning to use *fimm* in a while.

Huge thanks to my fantastic wife Aneta, who patiently supported my postponing of the thesis hand-in, taking care of our little daughters, and providing me with valuable time to eventually finish this work. NG, if you are reading this, I can confidently say that I am done now. Thank you for joining me in the Bergen-adventure, and bringing to the world the two lovely angels, Julia and Jagoda, whose company was a sought-after getaway after the long workdays these last months.

Heartfelt thanks to: my parents, who supported my choice of bioinformatics from the very beginning, carefully monitoring the progress of my studies, and ensuring that I was not excessively delayed with the thesis; to my grandfather for being a role-model of a researcher; and to my entire family for the support, and frequent visits to rainy Bergen to bring the feeling of being home.

Also, many thanks to my friend Rafał for sending the ’are we there yet?’ cards every few days during the last month of the thesis writing, and to all other people who have supported me in the course of the PhD studies and in the writing process.

Summary

The post-genomic era has been heavily influenced by the rapid development of high-throughput molecular-screening technologies, which has enabled genome-wide analysis approaches on an unprecedented scale. The constantly decreasing cost of producing experimental data resulted in a data deluge, which has led to technical challenges in distributed bioinformatics infrastructure and computational biology methods. At the same time, the advances in deep-sequencing allowed intensified interrogation of human genomes, leading to prominent discoveries linking our genetic makeup with numerous medical conditions. The fast and cost-effective sequencing technology is expected to soon become instrumental in personalized medicine. The transition of the methodology related to genome sequencing and high-throughput data analysis from the research domain to a clinical service is challenging in many aspects. One of them is providing medical personnel with accessible, robust, and accurate methods for analysis of sequencing data.

The computational protocols used for analysis of the sequencing data are complex, parameterized, and in continuous development, making results of data analysis sensitive to factors such as the software used and the parameter values selected. However, the influence of parameters on results of computational pipelines has not been systematically studied. To fill this gap, we investigated the robustness of a genetic variant discovery pipeline against changes of its parameter settings. Using two sensitivity screening methods, we evaluated parameter influence on the identified genetic variants, and found that the parameters have irregular effects and are inter-dependent. Only a fraction of parameters were identified to have considerable impact on the results, suggesting that screening parameter sensitivity can lead to simpler pipeline configuration. Our results showed, that although a simple metric can be used to examine parameter influence, more informative results are obtained using a criterion related to the accuracy of pipeline results. Using the results of sensitivity screening, we have shown that the influential pipeline parameters can be adjusted to effectively increase the accuracy of variant discovery. Such information is invaluable for researchers tuning pipeline parameters, and can guide the search for optimal settings for computational pipelines in a clinical setting. Contrasting the two applied screening methods, we learned more about specific requirements of robustness analysis of computational methods, and were able to suggest a more tailored strategy for parameter screening. Our contributions demonstrate the importance and the benefits of systematic robustness analysis of bioinformatics pipelines, and indicate that more efforts are needed to advance research in this area.

Web services are commonly used to provide interoperable, programmatic access to bioinformatics resources, and consequently, they are natural building blocks of bioinformatics analysis workflows. However, in the light of the data deluge, their usability for data-intensive applications has been questioned. We investigated applicability of standard Web services to high-throughput pipelines, and showed how throughput and performance of such pipelines can be improved. By developing two complementary approaches, that take advantage of established and proven optimization mechanisms, we were able to enhance Web service communication in a non-intrusive manner. The first strategy increases throughput of Web service interfaces by a stream-like invocation pattern. This additionally allows for data-pipelining between consecutive steps of a workflow. The second approach facilitated peer-to-peer data transfer between Web services to increase the capacity of the workflow engine. We evaluated the impact of the enhancements on genome-scale pipelines, and showed that high-throughput data analysis using standard Web service pipelines is possible, when the technology is used sensibly. However, considering the contemporary data volumes and their expected growth, methods capable of handling even larger data should be sought.

Systematic analysis of pipeline robustness requires intensive computations, which are particularly demanding for high-throughput pipelines. Providing more efficient methods of pipeline execution is fundamental for enabling such examinations on a large-scale. Furthermore, the standardized interfaces of Web services facilitate automated executions, and are perfectly suited for coordinating large computational experiments. I speculate that, provided wide adoption of Web service technology in bioinformatics pipelines, large-scale quality control studies, such as robustness analysis, could be automated and performed routinely on newly published computational methods. This work contributes to realizing such a conception, providing technical basis for building the necessary infrastructure and suggesting methodology for robustness analysis.

List of publications

Paper I
: Paweł Sztromwasser, Pål Puntervoll, and Kjell Petersen. *Data partitioning enables the use of standard SOAP Web Services in genome-scale workflows*. Journal of Integrative Bioinformatics, 8(2), 2011.

Paper II
: Sattanathan Subramanian, Paweł Sztromwasser, Pål Puntervoll, and Kjell Petersen. *Direct data transfer between SOAP web services in Orchestration*. In the International Conference on Information Integration and Web-based Applications & Services (iiWAS). ACM, 2012.

Paper III
: Sattanathan Subramanian, Paweł Sztromwasser, Pål Puntervoll, and Kjell Petersen. *Pipelined Data-flow Delegated Orchestration for Data-Intensive eScience Workflows*. International Journal of Web Information Systems, 9(3), 2013.

Paper IV
: Paweł Sztromwasser, Kjell Petersen, and Inge Jonassen. *Sensitivity screening reveals influential parameters of a variant calling pipeline*. Manuscript in preparation.

Related publications not included in the scientific results

Sattanathan Subramanian, Pål Puntervoll, and Paweł Sztromwasser. *Optimizing the Data-Traffic of Centrally Coordinated Scientific Workflow Systems*. In the International Conference on Web Services (ICWS). IEEE, 2010.

Håkon Sagehaug, Prabakar Venkataraman, Armin Topfer, Kidane Tekle, Matuš Kalaš, Paweł Sztromwasser, Anne-Kristin Stavrum, Michael Dondrup, Sattanathan Subramanian, Francisco Roque, Siv Midtun Hollup, Inge Jonassen, Kjell Petersen, and Pål Puntervoll. *eSysbio: an adaptable workbench for collaborative life science research*. Manuscript in preparation.

Abbreviations

API	Application Programming Interface
BPEL	Business Process Execution Language
DNA	Deoxyribonucleic Acid
EXI	Efficient XML Interchange
GWAS	Genome-Wide Association Study
HTTP	Hypertext Transfer Protocol
MIME	Multipurpose Internet Mail Extensions
PCR	Polymerase Chain Reaction
REST	Representational State Transfer
SNP	Single Nucleotide Polymorphism
SNV	Short Nucleotide Variant
SOAP	Simple Object Access Protocol
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WSDL	Web Service Description Language
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations

Contents

Part I

Synthesis

Chapter 1

Background

1.1 Bioinformatics analysis pipelines

Bioinformaticians develop computational methods that extract biologically relevant information from experimental data. These methods are frequently complex processes composed of elementary tasks that transform the input data step-by-step into meaningful knowledge. The chain of tasks is referred to as a pipeline, or a workflow.

Similar to an algorithm represented in a pseudo-code, bioinformatics pipelines have abstract representations. Most commonly, workflows are depicted as directed graphs with nodes representing interleaving data and data-processing steps, and edges outlining the sequence of steps (Figure 1.1). The source nodes represent pipeline inputs, including input data and parameters, whereas the sink nodes mark pipeline output. In concrete pipeline implementations, the data-processing nodes are bioinformatics programs or databases, and the workflow itself is a program coordinating the sequence of execution and the flow of the data between the components.

Workflows are often used to systematically automate computational protocols [1], and as a generic method of resource integration, they are prevalent in many areas of scientific computing [2]. In this thesis, I focus particularly on bioinformatics pipelines that are used for analysis of high-throughput data. To delineate the challenges involved in the development of such pipelines, I review their most important characteristics.

1.1.1 Characteristics

Geographical distribution and heterogeneity of resources

The basic components of bioinformatics pipelines are biological databases and bioinformatics programs. Both, the databases and the programs, are developed by researchers and shared with the public domain freely available. The Nucleic Acid Research journal keeps a record of these resources, and publishes updated collections annually. Most recently, over 1500 distinct data repositories [6] and nearly 1500 unique tools [7] were listed. This wealth of resources constitutes a diverse, complex and distributed [1] infrastructure for life science research. The Internet makes it easy to publish

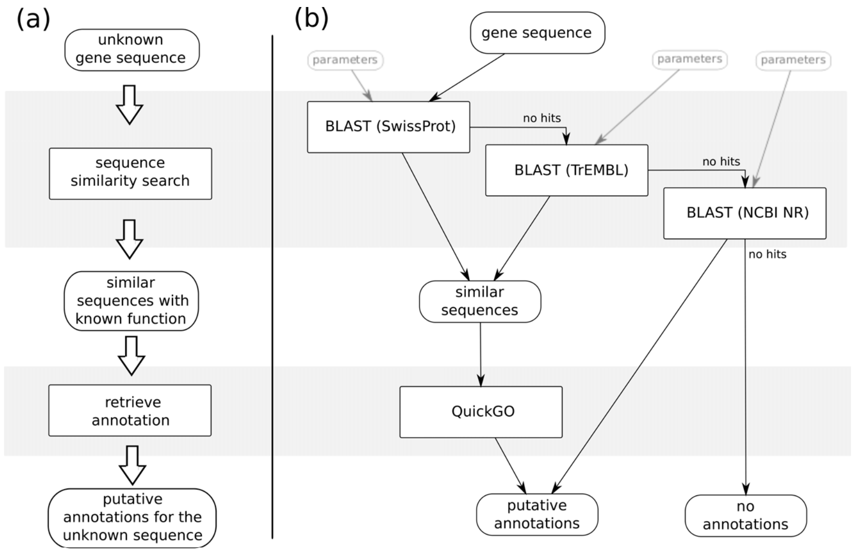

Figure 1.1: The figure shows an example of a bioinformatics pipeline - a simplified approach to predicting functions of uncharacterized gene sequences. The depicted workflow exploits the fact that all organisms and their genes are related through the process of evolution. This implies that if a newly sequenced gene has a high degree of sequence similarity to a gene of known function from a different species, it is likely to have the same or a similar function. The figure **(a)** depicts an abstract representation of the annotation pipeline, with the data in oval boxes, and processing steps in square boxes. From the top: the input sequence of an unknown gene (pipeline input) is first associated with a similar gene from another specie using sequence similarity search; next, functional annotations of the similar sequence (intermediate data) are retrieved from a database; in the end, the retrieved annotations (pipeline output) can be used to assign putative functions to the unknown gene. In figure **(b)** the data processing nodes show concrete tools used in the pipeline; the input, output, and intermediate data are presented in oval boxes. Tool parameters are also considered important input, but to distinguish them from primary inputs, the parameters are shown in a lighter color. The sequence similarity search is performed using BLAST [3], querying three distinct databases in the order of decreasing quality of annotation (and increasing size), i.e. UniProt/SwissProt, UniProt/TrEMBL, and NCBI NR. Each of the three database queries can be customized (using parameters exposed by the BLAST tool) to yield best possible results in identifying evolutionary related sequences. When a similar sequence is found, its functional (Gene Ontology [4]) annotations are retrieved using the QuickGO service [5]. The pipeline output includes both, putative annotations and sequences that failed to match a known gene in any of the databases, and thus could not be annotated.

new resources, both for the central data providers in the field (European Bioinformatics Institute (EBI) in UK, National Center for Biotechnology Information (NCBI) in USA, and DNA Data Bank of Japan (DDBJ)), and for smaller institutes and research groups around the world that make a reputation as service providers [1]. Collectively, these resources host vast amounts of biological information, and although partly overlapping [1], they represent a great value for the research community. However, the degree of fragmentation and geographical dispersion is challenging with respect to integrating the data resources [8], which also holds true for their use as building blocks in bioinformatics pipelines.

	Standalone application	Library	Web application	Web service
Installation and administration	user	user	provider	provider
Access constraints	operating system, dependencies	programming language	Internet access	Internet access
Programmatic access	yes / no	yes	no	yes

Table 1.1: Comparison of four classes of resource interfaces. In standalone applications, programmatic interface is available depending on type of the user interface, i.e. command-line applications can be invoked programmatically, in contrast to applications with graphical user interface.

The distribution of efforts in building the infrastructure for research in life sciences, in combination with autonomy of individual research groups [1], has resulted in a very diverse spectrum of resources. The heterogeneity, although enriching the field, creates a major challenge for resource interoperability and data integration [9, 1]. The diversity among resources is only partly caused by semantically diverse data-types present in bioinformatics. It is also (and mostly) due to syntactically different representations of the data, software design choices (platforms, programming languages), and technical skills of the implementers [1]. Lack of canonical data formats for biological information has led to proliferation of miscellaneous representations, modifications, and extensions of existing formats. For instance, multiple sequence alignments can be found in over 20 different representations according to EDAM[1], an ontology of bioinformatics data and operations [10]. Abundance of distinct representations of semantically identical data makes implementation of conceptually simple pipelines difficult, due to the necessary format conversions [11, 12]. Such interoperability problems are greatly alleviated by standardization [1]. Thus, several initiatives have promoted common data exchange formats, both for everyday data-types such as sequences, alignments, and annotations [11], and for more specialized types like systems-biology models [13] or protein structures [14].

Heterogeneity of resource interfaces

The heterogeneity of the resources is also manifested on the level of user interfaces, which in case of bioinformatics software also include programmatic interfaces (APIs). Four classes of interfaces can be distinguished for bioinformatics resources (Table 1.1): standalone applications, Web applications, programming libraries, and Web services. Standalone applications are locally installable programs providing interactive user-interface, either via a command-line (e.g. BWA [15], BLAST [3]), or a graphical user interface (e.g. JExpress [16], JalView [17], Cytoscape [18]). The command-line applications are suitable for embedding in scripts, in particular if they consume and produce data in canonical formats (e.g. VCF [19] or BAM [20]), while applications with graphical user interface usually do not offer programmatic execution. Locally installed standalone applications require an operating system supported by the program,

[1]`http://edamontology.org`

sufficient hardware resources, installation of necessary dependencies, and performing periodic updates. Also downloadable databases have similar requirements. In contrast, Web applications (also referred to as Web servers) which are also used to provide interactive access to biological databases (e.g. Uniprot [21], ELM [22], OMIM [23]) and computational tools (e.g. NCBI BLAST [24], WEBnm@ [25]), are maintenance free from the user perspective. They combine great user experience tailored to the offered functionality with effortless availability (no installation nor maintenance is needed). Since Web servers are intended for interactive, manual use, and do not provide programmatic access *per se*, their usability in automated pipelines is limited. In the past this lack of proper programmatic access has led to a wide-spread 'web-scraping' practice [9], i.e. an automatic parsing of Web content formatted and intended for visual display to humans. Despite emergence of alternative methods, this practice continues [26].

Resources designed to be programmatically accessed are provided as libraries (e.g. Bioconductor [27], Biopython [28], Bioperl [29]), or Web services[2] [30]. The libraries are installed locally, and can only be used from the programming language they are written in. In contrast, Web services are platform- and language-independent and offer remote, automatic access using standard Internet protocols such as the HTTP. A Web service "is a software system designed to support interoperable machine-to-machine interaction over a network" [3], or simply a programmatic interface [30, 31]. Today, many institutions offer tools [32] and access to data [33] using Web service interfaces, and the technology has become a *de facto* standard for data integration in the bioinformatics community [26, 34]. It should be noted that prior to Web services, the standard for interoperable communication between machines was CORBA[4], but in contrast to Web services [31], it has not become widely used [1].

Data size

Many bioinformatics analysis pipelines are unarguably low-throughput. However, since the wide-spread adoption of microarrays, and later deep-sequencing technologies, genome-wide analyzes have become a routine. The fields of genomics, transcriptomics, and proteomics are flourishing, and high-throughput instruments generate enormous amounts of data, challenging existing software and hardware infrastructures [35]. DNA sequencing capacity doubles every 9 months [35] and has outpaced the growth in computer power [36] approximated by Moore's law (Figure 1.2). As the trend continues, it is not data production, but data management and computational data analysis that becomes the technical bottleneck of research in life sciences [36].

Analysis of high-throughput data requires efficient computational methods, that aid extraction of relevant information from the deluge of data. Due to the complexity of the assessed problems and the increasing data sizes, many of the programs and pipelines require long-running calculations and powerful computer resources. Software developers focus strongly on increasing performance of their programs, and runtime is among the most frequently benchmarked qualities of bioinformatics tools and algorithms (e.g.

[2]`http://www.w3.org/TR/ws-arch`
[3]`http://www.w3.org/TR/ws-arch` (cited Nov 2013)
[4]`http://www.omg.org/spec/CORBA/Current`

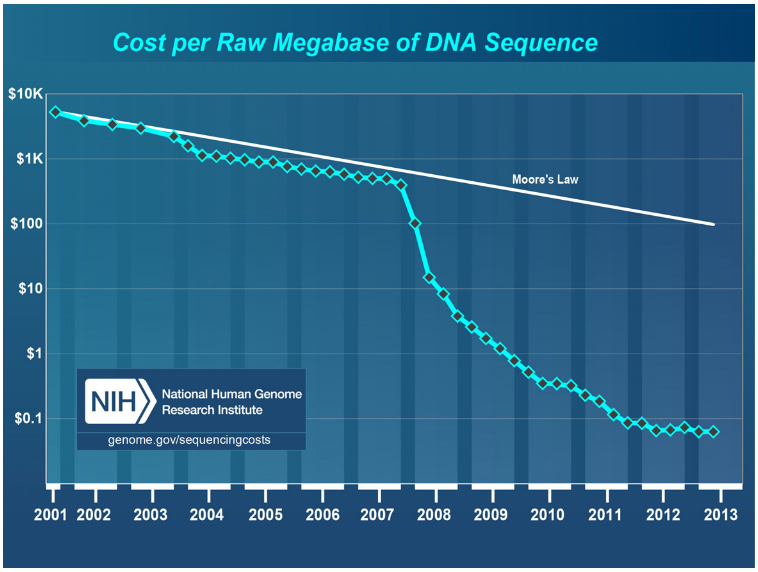

Figure 1.2: The decreasing cost of DNA sequencing in comparison with the Moore's law. Graphics taken with permission from `http://www.genome.gov/SequencingCosts` [37].

[38],[39]). In consequence, many programs support parallel computing, often taking advantage of a key property of biological data - cardinality. Typically, sizable datasets consist of large numbers of small or medium-size entities, that represent individual molecules or measurements. If these entities are independent, the dataset can easily be split for concurrent processing to effectively reduce the time necessary for the analysis [40]. Parallel programs need to be executed on infrastructure that supports concurrent processing on a large scale, for instance high-performance computing centers, computational grids, or clouds. For particularly critical applications, like for instance short read mapping to a reference sequence, additional efforts are made to boost the performance further, by for instance custom-made hardware [41, 42]. A recently proposed concept of *compressive algorithms* [36] addresses the challenges of searching and storing large genomic datasets, i.e. taking advantage of great redundancy of information in re-sequenced human genomes, the data can be compressed in a manner allowing efficient similarity searches using customized versions of BLAST and BLAT algorithms. Still, the expected data growth puts forward strong requirements for software systems that support high-throughput data analysis in life sciences.

1.1.2 Methods of pipeline composition

The previous section described bioinformatics workflows as automated data analysis processes that integrate distributed and heterogeneous resources. High-throughput pipelines can be further characterized as computationally-expensive, long-running, and data-intensive. The explorative and collaborative nature of research raises additional requirements regarding pipeline management systems. The workflows need to be rapid in prototyping and flexible in adapting to continuously evolving analysis practices [12, 43]. Sharing workflows between researchers should be easy to avoid duplication of efforts and simplify collaboration. The researchers should have the possibility to document the execution of a pipeline, together with exact versions of resources used, parameter values, and provenance of the data, to ensure reproducibility of the experiments [44, 45]. In practice, these pose challenges with respect to the workflow technology, as it needs to flexibly orchestrate diverse distributed resources in an efficient manner.

Scripting

Scripts are the most basic method of automating data analysis in bioinformatics. They are short programs written *ad-hoc*; "quick-and-dirty" [46] one-offs that allow to quickly get an answer to a question [9, 27]. All scripting and programming languages can be used for this purpose, but shell scripts, Perl, Python, and R are probably most commonly used. Shell scripts are very useful in automating execution of command-line programs, file operations, and they provide access to rich text-processing support in the Unix shell. Scripting languages, like Perl and Python, provide in addition more advanced programming constructs, and comprehensive libraries for diverse purposes (e.g. database access, Web service invocation). More importantly, toolkits of bioinformatics methods, such as Bioperl [29] and Biopython [28], are available for these languages. These toolkits help to parse files, convert between common data formats, and access external data resources, effectively limiting the duplication of efforts among bioinformaticians [9] The statistical programming language R [47] displays a similar strength, providing arguably the most comprehensive data analysis library for biological data (i.e. Bioconductor [27]), in addition to a rich set of statistical and plotting functionalities.

Data analysis pipelines developed directly in a programming language can easily access both local and remote components, and provide greater control over implementation details, which can help, for instance, in optimizing performance. Developing one-off scripts is quick and it facilitates rapid prototyping, but the resulting code is hardly reusable [27, 48, 9]. These frequently undocumented scripts [9] are not readily accessible for fellow programmers, hamper transparency and reproducibility, and cannot easily be used to document data provenance. They can also be inaccessible for researchers unfamiliar with programming [49, 50].

Workbenches

Streamlining data analysis with scripting languages requires programming skills, which are not very common among life science researchers [50, 49]. For them, applications called workbenches offer a possibility to perform multi-step data analysis using a predefined set of tools and resources, without the need for coding. The workbenches can also provide visualization utilities, data storage and management, transparent provenance tracking; altogether creating a powerful integrated environment for data analysis. Examples of such applications are Galaxy [50–52], Chipster [53], and GenePattern [49].

Galaxy is focused on making genomic data analysis accessible and reproducible. The workbench provides a virtual workspace where users can store the data, analyze it with a set of genome analysis tools, and save the history of a step-by-step analysis as a workflow. The workflows can be conveniently shared, both for reuse by other researchers, and for documenting analysis history. Galaxy has a Web-based client application where users interact with the tools and data, and a back-end server where the computations are executed. While a public instance of the Galaxy server is freely available[5], researchers who need better performance, or custom tools and resources can download Galaxy and deploy it locally. Chipster is a platform for high-throughput data analysis; it offers a spectrum of computational and visualization methods, predefined workflows and the possibility of creating user-defined pipelines based on execution history. Chipster workflows can be saved as files and shared with other users of the system. The system has a client-server architecture consisting of a graphical desktop application, and one or more computational servers. GenePattern was initially used for gene expression analysis [49], but currently its scope covers also proteomics, flow-cytometry, and SNP analysis. The workbench supports connecting individual analysis steps into pipelines, which can subsequently be added to the tool inventory or exported for sharing with other users of the system. Users interact with the system using a Web-based application, while the computations and analysis is performed on a server, similarly to Galaxy. An attractive feature of GenePattern is the programmatic interface that allows using the analysis modules in an automated fashion outside of the application.

Workbenches are convenient and accessible environments for data analysis in bioinformatics. They focus primarily on interactive analysis, but each one of them supports automation of repetitive analysis processes in a form of pipelines. The default selection of tools and resources in these workbenches corresponds to the research area they are focused on, but all the workbenches have a possibility of extending the default toolbox.

Workflow management systems

Another type of applications that can facilitate automation of data analysis without requiring programming skills are (scientific) workflow management systems. In a workflow management system, workflows are defined, managed, and executed on computing resources [2]. Instead of aggregating tools and resources to perform data analysis centrally (like workbenches), workflow systems provide means of coordinating distributed analysis components. Workflow management systems are popular in grid environments (e.g. Kepler [55], Pegasus [56], and Triana [57]), but are also used in bioinformatics where the dominant workflow engine is Taverna [58].

The distributed analysis components invoked during workflow execution are frequently services of various sorts, e.g. Web services, Grid services, cloud services [58]. Combining distributed (and heterogeneous) services is a complex procedure [8], and the workflow systems assist the user by providing graphical environments for workflow design (Figure 1.3) [57, 55, 58]. This theoretically makes it possible to develop composite pipelines with no programming skills. In practice however, considering that the bioinformatics Web services alone are not a uniform group of software systems [30, 8], it is challenging to ensure interoperability between them. In Taverna, the heterogeneous data representations and technical differences are taken care of using shims [58, 54, 8] -

[5]`http://usegalaxy.org`

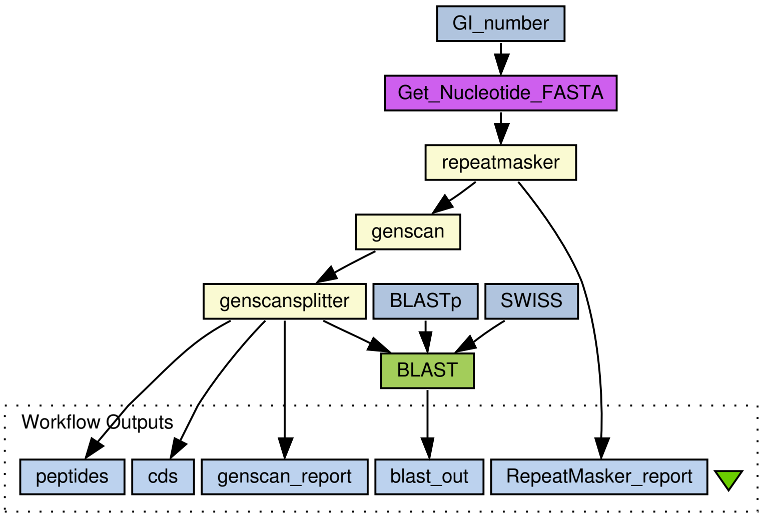

Figure 1.3: A genome annotation pipeline implemented as a Taverna workflow [54].
©Duncan Hull, Hannah Tipney / myExperiment / CC BY-ND 3.0

small programs that align the input and the output of connected services [12]. Another challenge are services with poorly documented interfaces, which can require manual investigation of service capabilities, including formats of the input and output data [54]. Also stability of the tools in a distributed inventory can be a concern [54, 1], in contrast to locally maintained resources.

The numerous scientific workflow management systems lack a common standard format for describing workflows [2]. Proliferation of peculiar formats [59] hinders reuse of workflow designers [2], and prevents reuse of pipelines between the environments. In the IT industry a *de facto* standard [60] for defining workflows is the Business Process Execution Language (BPEL)[6] [61] specified by the standardization organization OASIS[7]. Several engines that are able to execute BPEL workflows have been developed, with the most prominent open-source representative being Apache ODE[8], jBPM[9], and OW2 Orchestra[10]. Although BPEL is a standardized format capable of expressing scientific workflows [60], it has nearly no recognition in the bioinformatics community. The reasons for it could include: exclusive support for the standard Web services [62], making it difficult to build pipelines using local computations and other types of services; limited selection of open-source tool support [62], besides workflow execution [60] and design [59]; verbosity of the language [62, 59]; and performance problems related to processing large data using Web services [63, 64].

[6]http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html
[7]http://www.oasis-open.org
[8]http://ode.apache.org
[9]http://www.jboss.org/jbpm
[10]http://orchestra.ow2.org

eSysbio

The eSysbio project[11] explored the possibility of building a Web-based bioinformatics workbench using standard Web service technologies. We have built a proof-of-concept system that combines rich data management functionality, allowing to store, visualize, and share data, with a distributed analysis toolbox consisting of standard Web services, R scripts, and BPEL workflows (the manuscript describing the eSysbio system is in Appendices, in Part III). Similar to Taverna, a user of eSysbio can extend his or her toolbox by providing a URL to a service interface, taking advantage of the wealth of utilities gathered, for instance in BioCatalogue [65]. The modules of the system are standard Web services themselves, allowing programmatic access to the systems from external applications, as well as the default Web-based user-interface. As a member of the project, the author studied performance of high-throughput pipelines composed of standard Web services.

1.1.3 High-throughput pipelines of Web services

Data analysis using distributed and heterogeneous bioinformatics resources requires a technology that allows interoperable communication over the Internet. The interoperability of the programmatic interfaces implemented using Web services, empowered by the ”alphabet soup of standards” that accompanies the technology [9], and ability to communicate over the network, matches perfectly the needs of distributed and diverse bioinformatics resources. Thus, the technology of Web services was recognized early as a promising platform for integration in bioinformatics [9]. Since the landmark commentary by Lincoln Stein [9] who envisioned ”bioinformatics nation” united by Web services exchanging data in common formats, the technology has spread in the field in many forms [30, 31, 8, 66, 67, 34]. Today, many groups offer their tools and access to data using Web service interfaces, and the technology is considered a *de facto* standard for data integration in the community [26, 8, 34]. Initiatives like EMBRACE[12] [30], BioHackathon[13][34], and recently ELIXIR[14] actively helped in promoting use of Web services by contributing guidelines, training, tools, and service registries. Previously, the EMBRACE registry of Web services [68], and at present the BioCatalogue [65] list over 2300[15] bioinformatics services deployed around the world. The importance of Web services was also marked by the Nucleic Acid Research journal, which since year 2010 has been including a separate Web service category in its annual Web Server issue [69].

Although the World Wide Web Consortium (W3C)[16] defines Web services as a concrete standard[17], in bioinformatics the term *Web services* covers a heterogeneous group of programmatic interfaces accessed over the Web [30]. Among these, two main types

[11]`http://esysbio.org`
[12]`http://www.embracegrid.info`
[13]`http://www.biohackathon.org`
[14]`http://www.elixir-europe.org`
[15]accessed Nov 2013
[16]`http://www.w3c.org`
[17]`http://www.w3.org/TR/ws-gloss`

can be distinguished: REST and SOAP-based [30]. Both types of services are commonly used to implement programmatic interfaces to bioinformatics resources, for instance the DAS services [66] belong to the REST category, while BioMoby services [67] are SOAP-based. Depending on the requirements of the resource, one type can be more suitable than the other [70], and ”both technologies can be used to achieve useful results” [30]. The SOAP-based Web services are carefully specified by a set of W3C standards, including SOAP[18] messaging protocol, machine-readable interface description format WSDL[19], XML[20] for data representation, and XML Schema[21] for defining the data formats. These specifications are further constrained by the guidelines of the OASIS Web Service Interoperability (WS-I)[22] profile [30] that provides a ’gold standard’ which allows the service providers to improve the quality of their Web service interfaces, and increase their interoperability [30]. Supporting the claim that ”standardization can largely alleviate interoperability issues” [1], we have concentrated our efforts on the SOAP-based Web services. Consequently, from this point onwards the term Web services will denote SOAP Web services compliant with the WS-I profile.

The critics of Web services argue that the technology is getting heavier than it needs to be to support flexible integration [1], and that it does not meet the requirements of large-data processing. Using a Web service interface, it is not straight-forward to provide long-running computations due to the limited time a HTTP connection can be open [34] (this applies also to the REST services). The issue is often circumvented using so-called *status-polling* [30, 34], which allows the service invocations to be short-lived: in response to sending input data, a job (session) identifier is returned; the identifier can be used to repetitively check the status of the started job, while waiting for its completion; and in the end, it is used to retrieve the result of the computation.

More challenging for high-throughput applications is the poor performance of Web services in handling large data [63, 64, 31, 34]. Although the management of sizable data is generally problematic in distributed computing, the XML used for Web service communication aggravates the issue on two levels. First, information encoded in XML is inflated making a SOAP message 4 to 10 times larger in size than a machine representation of the data it contains [63]. Second, conversion from in-memory to textual representation of the data (serialization) and *vice-versa* (de-serialization) are both resource-demanding processes [63, 64, 71], that limit the amount of data that can be handled by existing Web service libraries [34]. Several standardized mechanisms are able to increase efficiency of data handling in Web services [64], including Message Transfer Optimization Mechanism (MTOM)[23] which allows to send bulky data as an attachment to a SOAP message, and thus avoid the expensive serialization and de-serialization [34]. Still, it is challenging to apply the regular Web service paradigm directly in data-intensive applications, and enabling high-throughput pipelines of Web services requires increasing performance and throughput of Web service communication.

[18]`http://www.w3.org/TR/soap`
[19]`http://www.w3.org/TR/wsdl`
[20]`http://www.w3.org/TR/xml11`
[21]`http://www.w3.org/TR/xmlschema-0`
[22]`http://www.oasis-ws-i.org`
[23]`http://www.w3.org/TR/soap12-mtom`

Improving performance and throughput of Web service pipelines

Various efforts address efficiency in handling voluminous data on the pipeline level, and many of them question the applicability of the typically centralized workflow coordination (orchestration) to data-intensive workflows. (Note the difference between centralized computations in bioinformatics workbenches, and centralized coordination of decentralized services). In orchestration, a central workflow engine controls the invocation order of distributed component services, and acts as a broker for all data exchange. Centralized coordination makes it easier, for instance, to execute the workflow, monitor progress, and handle failures. However, in high-throughput pipelines, the central data broker can become a communication bottleneck constraining scalability [72] and performance [73]. In contrast, in decentralized coordination (choreography), the services are more autonomous and communicate directly with each other. Choreography is a more natural model for the data-driven scientific analysis pipelines, but due to the decentralized control, it is difficult to implement in practice [74, 72].

To combine the advantages of orchestration (i.e. central control) and choreography (i.e. peer-to-peer data transfer), a hybrid model that separates the control-flow from the data-flow was proposed [72]. The *centralized control-flow decentralized data-flow* model was shown to have lower aggregated cost than the purely centralized approach [72], and it was used in several solutions that aimed at improving performance of data-intensive workflows. Liu *et al.* [75] developed an infrastructure and protocol supporting the *centralized control-flow decentralized data-flow* approach to building composite services. Barker *et al.* [73] and Binder *et al.* [76] suggested that data can be exchanged between intermediate components (proxies) positioned between the workflow engine and the service. Similarly, in Data-Grey-Box Web services [77], the data-part of the message was separated from the part containing functional parameters, and exchanged separately using more efficient means. Zhang *et al.* [78] proposed and implemented result forwarding to the next service in the pipeline, using Web Service Resource Framework (WSRF)[24] resources. Also a pragmatic approach of sending a link to the data (a reference, or a handle) instead of the actual data [55, 34], can be considered as a variant of this model, since the data is retrieved using a separate transport channel.

Parallelization on different levels of workflow execution can also increase pipeline throughput. The ability to concurrently execute independent branches of a workflow graph is referred to as workflow-parallelism, and is implemented in many workflow engines [79]. Data-parallelism, so simultaneous processing of independent data elements with minimal performance loss, can be implemented on the service [79] and the workflow engine level, e.g. intra-processor parallelism in Taverna [80]. The third type of parallelism, service-parallelism [79], allows for overlapping computation and communication during pipeline execution, and has been implemented for instance in Kepler [81], Taverna [80], and grid services [82]. In workflows where data processing is more expensive than communication, service-parallelism (also called pipelining) can nearly eliminate the influence of communication on the workflow runtime. Additionally, pipelining allows for production of the first batches of final results much faster, which can be helpful in for instance debugging workflows.

All the above strategies address efficiency of data-handling in pipelines and facilitate

[24]`https://www.oasis-open.org/standards#wsrfv1.2` (accessed Nov 2013)

increase of throughput. However, none of them is focused on standard Web services exchanging data in XML. Section 3.1 presents two approaches that aim at optimizing data-handling in pipelines composed of such Web services.

1.2 Genetic variants

The building recipe for any living organism is encoded in a chain of DNA. The sequence of nucleotides that make up DNA is unique to every person, and it is called a genome. The differences between genomes of people, genetic variants, underlie the biological diversity among us. On average 1% of the 3 billion nucleotides that make up a human genome are variants differing from the human reference genome [83]. The genetic variants range from single nucleotide substitutions (Figure 1.4), short insertions and deletions of DNA fragments (indels), to larger rearrangements. Single nucleotide substitutions that occur frequently in a population are referred to as single nucleotide polymorphisms (SNPs), whereas the term short nucleotide variants (SNVs) also encompasses indels.

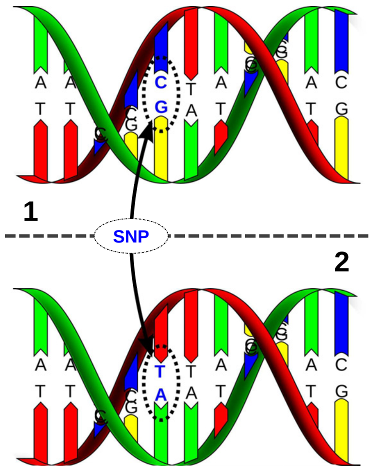

Figure 1.4: Two double-stranded DNA molecules differing in one nucleotide-pair display a single nucleotide substitution (SNP in this case). The DNA strands are composed of four types of nucleotides: adenine (A), cytosine (C), guanine (G), and thymine (T).
©David Hall / Wikimedia Commons / GFDL / CC-BY-SA-3.0 / CC-BY-2.5

Genetic variants have been found to be the causes of many rare genetic disorders [84]. Also common diseases, such as type-2 diabetes, are linked to hundreds of genetic variants through genome-wide association studies (GWAS) [85, 86]. Somatic (not inherited) mutations found in patients diagnosed with cancer point to genes that can be involved in tumor progression [87–89]. The advent of deep-sequencing technologies, allowing for rapid and cost-effective interrogation of genomes in the search for genetic variation, can thus be seen as a milestone in biology and biomedicine [90]. The sequencing technology is an excellent tool for finding genetic causes of inherited diseases [91], classifying cancer patients for the most promising drug-treatment [92, 93], and molecular diagnostics of diseases with ambiguous symptoms [94]. The advances in sequencing technology has made it accessible enough to consider transferring the technology to the clinic [95, 92, 93]. However, the bioinformatics analysis of deep-sequencing data is challenging [44], and currently ”the $1000 genome sequence comes with $20000 analysis price tag” [96].

1.2.1 Bioinformatics pipeline for genetic variant discovery

Deep-sequencing data used for genetic variant discovery consists of millions of short sequences called reads. Depending on the platform used for sequencing, the reads have different properties (pairing, length, error-rate) [90, 97], and they can be more, or less suitable for genome re-sequencing and variant calling. The reads are produced by a sequencer from a sample of amplified genetic material, which depending on a particular application, can either represent an entire genome (whole genome sequencing), or regions of interest (targeted sequencing). A commonly targeted part of the genome are the protein-coding regions (exome), as they contains the best-understood functional elements [98]. This approach is referred to as exome-sequencing, and relies on a laboratory procedure that is able to selectively capture the targeted part of the genome.

Assuming an appropriate sequencing platform is chosen, a generic data analysis pipeline for discovering small nucleotide variants (genotyping) using deep exome-sequencing is depicted in Figure 1.5. A detailed explanation of the steps involved is provided together with the figure. Briefly, the analysis starts with aligning reads to a reference genome, followed by several steps where the alignment is refined, to eventually be used for calling variants. Depending on the aim of the study, the resulting variants can be the final result (e.g. in association or population studies), or input to a rigorous manual analysis by a geneticist (e.g. search for the causal mutation). In the latter case, the causative variants identified need to be confirmed using lower-throughput laboratory methods, such as Sanger sequencing or PCR [95, 92].

Accurate determination of genetic variants is challenging [105], and the complex computational methods employed at every step in the pipeline are unfamiliar to many biomedical researchers [44]. The analysis programs process considerably large data, are computationally intensive, and frequently use parallel processing [100, 15, 101]. All this makes it difficult to offer routine analysis of deep-sequencing data without a powerful computing infrastructure, and assistance from bioinformaticians. Sustaining the growing application of deep-sequencing in biomedicine will require accurate [92], robust [92, 95], and more accessible bioinformatics tools [95, 92, 44], transparency of the computational details [44], and reproducibility of the results [44]. Together with concerns regarding data privacy [35] these are also the main bioinformatical challenges that need to be addressed before deep-sequencing data can be used on a large scale in health care.

As part of this thesis, robustness of a state-of-the-art genetic variant calling pipeline was studied. Section 3.2 summarizes results of the study, and their implications for the accuracy of the variant calling. For reference, definitions of the terms used in that section, are presented below.

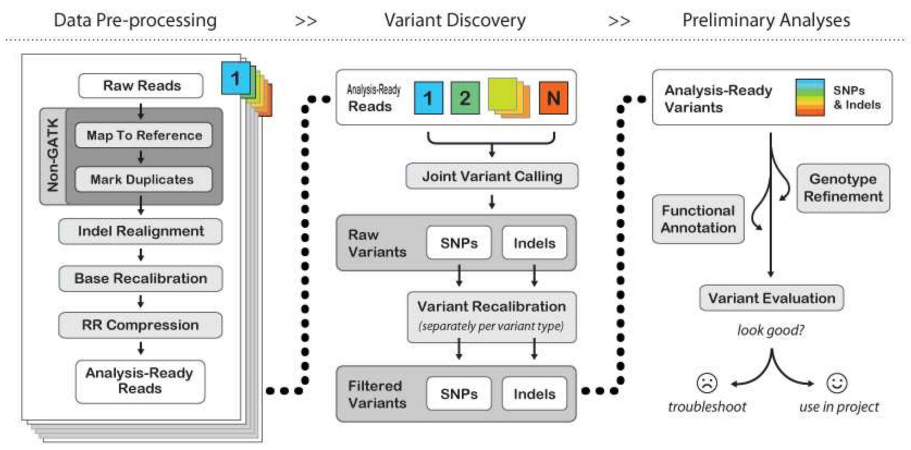

Figure 1.5: The Best Practices workflow [99] for variant calling with the GATK [100]. The processing starts with the alignment (mapping) of reads to a reference genome. This step results in an alignment file, documenting the most likely original position in the genome for each sequenced read. Due to mismatches between reads and the reference sequence, which can stem from sequencing errors or biological differences, this is a challenging step and often a read cannot be mapped unambiguously to a single position [15]. The alignment step has a critical effect on quality of variant calling [97], and must be done carefully. Several programs that implement distinct mapping algorithms [39] are available [15, 101–104], and offer variable performance and accuracy [38]. After the alignment is generated, it can be refined by for instance: removing duplicate reads being artifacts of the DNA amplification (utilities in toolkits SAMTools [20] or Picard[25]can be used for this); correcting erroneously aligned bases around indels (using GATK [105]); and recalibrating of base quality scores (GATK [105]). Next, the alignment is used to call variants, i.e. find mismatches between the analyzed sample and the reference sequence. This is done using variant calling software (e.g. GATK UnifiedGenotyper [105], SAMtools mpilup [20], SOAPsnp [106]) using either a single sample, or multiple samples (alignments) together to improve sensitivity of the process [105]. The resulting raw variant calls can contain considerable portion of calls that are artifacts of sequencing errors or data processing [105]. Hence, in order to increase specificity, calls with quality scores indicating a false-discovery can be filtered out based on either raw quality values provided by the genotyper, or better, recalibrated variant quality scores [105]. Also publicly available SNP datasets, like HapMap [107] or dbSNP [108], can serve as reference information in the filtering. Finally, the refined set of variants is ready for evaluation by a geneticist in the light of other functional information linked to the genome sequence.
(source: Broad Institute, `http://www.broadinstitute.org/gatk`)

1.2.2 Metrics of accuracy - recall and precision

The accuracy of a method describes the ability of the method to provide correct results. Accurate variant calling requires both the capacity to find the locations where the analyzed genome differs from the reference (real biological variants), and the ability to discard sequencing and misalignment errors. The correctly called genetic variants are *true positives*. A variant call in a location where the analyzed genome is equal to the reference is a type I error, and the call is a *false positive*. Missing a real genetic variant

[25]`http://picard.sourceforge.net`

is a type II error, and the call is a *false negative*.

To formally define these terms, let L be the set of all locations in the considered fragment of a genome, $V \subseteq L$ be the set of locations of real genetic variants, and $C \subseteq L$ be the set of locations identified by a variant calling method. Perfect variant calling would yield $C = V$. If TP is the set of *true positives*, FP the set of *false positives*, and FN the set of *false negatives*, then:

$$\begin{aligned} TP_{C,V} &= \{c : c \in C \land c \in V\} \\ FP_{C,V} &= \{c : c \in C \land c \notin V\} \\ FN_{C,V} &= \{c : c \notin C \land c \in V\} \end{aligned}$$

Recall rate of the calls in C, also known as sensitivity (of the variant calling), describes the ability to detect the biological variants (V), and is defined as

$$Recall(C,V) = \frac{|TP_{C,V}|}{|TP_{C,V}| + |FN_{C,V}|} = \frac{|TP_{C,V}|}{|V|}$$

Precision of the calls in C (positive predictive value), describes the capacity to avoid erroneous calls, and is defined as:

$$Precision(C,V) = \frac{|TP_{C,V}|}{|TP_{C,V}| + |FP_{C,V}|} = \frac{|TP_{C,V}|}{|C|}$$

Another frequently used accuracy metric (more relevant for binary classifiers) is specificity. Specificity describes the ability to identify negative results, so in case of variant calling, the locations with no real variants and no variant calls (*true negatives*, TN):

$$TN_{C,V} = \{c : c \notin C \land c \notin V\}$$

$$Specificity(C,V) = \frac{|TN_{C,V}|}{|TN_{C,V}| + |FP_{C,V}|}$$

In this thesis (Section 3.2), we used recall and precision to describe accuracy of variant calling.

1.3 Reproducibility and robustness of pipelines

The reproducibility of analysis results is a recognized concern in genomics research [109, 44], and evidently it is even more critical in a clinical setting. An unambiguous and detailed description of the data processing pipeline including input data, as well as access to exact versions of software and parameter settings used for the analysis is required to reproduce a computational analysis [44]. The peer-review process helps improving the situation by making such documentation compulsory for publication, but the extent of the information provided varies greatly, making it frequently impossible to reproduce the computational analysis [44]. The analysis environments that transparently document the history of the data analysis, and allow for publishing it together with the results [58, 50, 49], are promising steps towards increasing reproducibility.

The attention given to the details of the computational analysis in order to reproduce results suggests that results of computational pipelines can be fragile. Indeed, it has been shown that the software programs used in the analysis and their parameterization have considerable impact on the results [110, 111]. Individual tools are rigorously evaluated in benchmarks (e.g. [38, 112, 90, 113]), and increasingly more often studied in the context of pipelines [114, 110]. But the influence of distinct program settings is not examined, since the benchmarks use default or ”near-default” parameter values, e.g. [38, 114, 110]. The parameters of individual tools are usually documented, but the quality of the documentation varies between different software [95], making informed selection of settings difficult. In essence, the extent of parameter influence on the results of bioinformatics pipelines remains largely unknown, and needs to be explored to meet the demand for robust analysis-methods in the clinical setting [95, 92].

1.3.1 Evaluating robustness with sensitivity analysis methods

The robustness of a method is ”a measure of its capacity to remain unaffected by small, but deliberate variations in method parameters” [115]. It provides an indication of method reliability during normal usage [115], but it is not directly related to the quality of the findings. A robust pipeline produces stable and predictable results irrespective of small perturbations in the input (both input data and parameters). In contrast, results of a sensitive pipeline can heavily depend on the parameter values used, and small variations in the input data. The connection between robustness and sensitivity allows us to use sensitivity analysis methods to evaluate robustness of the pipeline and its sensitivity to varying parameter values.

Sensitivity analysis tries to explain ”how uncertainty in the output of a model can be apportioned to different sources of uncertainty in the model input” [116]. If we consider a pipeline as a computational model, that for given inputs (including parameters) produces an output, the sensitivity analysis can help in explaining the relations between varying parameter values and the result changes. Depending on characteristics of the model (pipeline), its inputs and outputs, different sensitivity analysis methods can be used. In computational pipelines the relationship between inputs and outputs of a pipeline are collectively described in the algorithms of the component programs, and little can be assumed about them (e.g. linearity). A standard strategy is to view the model as a black box [116] which converts a set of inputs into a set of outputs. In such an approach, the relationships between inputs and outputs are identified and quantified using repetitive executions of the model (pipeline) with varying inputs, and observing the effects on the output.

Focusing on the pipeline settings, the number of possible parameter value combinations grows exponentially with the number of parameters, ruling out exhaustive analysis even for insignificant number of parameters. Instead, Monte Carlo or quasi-Monte Carlo methods are commonly used to randomly sample a reasonable number of input settings for evaluation. The latter provides a more even distribution of pseudo-random points in the sampled multidimensional space, by using for instance a low-discrepancy Sobol’ sequence [117] or the Latin Hypercube design [118]. Based on pipeline evaluations for the sampled settings, the sensitivity can be quantified using local methods based on partial derivatives, or global methods that rely on variance decomposition (e.g. FAST [119, 120], Sobol' indices [121]). Local analysis methods, although cost-efficient, are only applicable to models where linear relationships between inputs and outputs can be assumed [122]. In contrast, global methods for sensitivity analysis can quantify sensitivity of non-linear models, but are more complex and require hundreds or thousands of model evaluations per parameter [122]. Using such methods for long-running bioinformatics pipelines, that may have tens of parameters, is impracticable. Approximating approaches rely on meta-modeling methods that can emulate execution of expensive models [116], High-Dimensional Model Representations [123], or group sampling [116]. A two step approach has also been proposed [122, 124], where first, the problem dimensionality is reduced by screening for influential parameters [125, 126], then subsequently only these are examined in a quantitative analysis.

Although the analogy between computational models and scientific workflows is rather clear, we were not able to find published examples of sensitivity analysis methods being applied to computational pipelines. The promising potential of these methods needs to be explored, and if necessary, the methods should be adapted to specific needs of testing robustness of computational pipelines. As part of this thesis (Section 3.2), a sensitivity screening method was applied to a bioinformatics pipeline, to study influence of pipeline parameters on the results. The details of the methods used are described below for reference.

1.3.2 Elementary effects method

The elementary effects method (also called the Morris method) [125] is a sensitivity screening technique designed to classify parameters of computational models into three categories: non-influential, having linear influence, and influential with non-linear or dependent effect. The method describes a strategy for global sampling of multidimensional parameter space, that allows to efficiently compute effects of individual parameters (elementary effects). In addition, Morris proposed to characterize the parameters of a model using simple statistics over the computed elementary effects.

In this section, first, the elementary effect is defined. Next, two strategies of sampling the multidimensional parameter space are presented: a local nominal-range sampling, and an efficient global strategy of the Morris methods [125]. The description of the global sampling strategy is followed by a brief summary of the improvements suggested by Campolongo *et al.* [126]. The approach to modification of parameter values suggested by Morris is presented next. In the end, the measures of parameter influence based on elementary effects are defined. As here the methods were used to screen for influential parameters in a computational pipeline, the terminology is chosen accordingly.

Elementary Effect

Let P be a pipeline that processes input in and produces output out. P is controlled by k parameters $p_1, p_2, ..., p_k$, that together with input have influence on the result, and thus $out = P(in, p_1, p_2, ..., p_k)$. In analysis of parameter influence, all the input except parameter values is kept constant, so in can be omitted from the list of arguments. An elementary effect (ee_i) is a change in the result introduced by a modification (Δ) of a single parameter value (p_i), relative to the modification:

$$ee_i = \frac{P_{in}(p_1, p_2, ..., p_i + \Delta, ..., p_k) - P_{in}(p_1, p_2, ..., p_i, ..., p_k)}{\Delta} \tag{1.1}$$

An inherent requirement for computing an elementary effect is the possibility of quantifying the difference between the results of two pipeline runs. Although pipeline results rarely are scalar values that can be directly subtracted, a function capturing essential characteristic of the results can frequently be defined to return a scalar. For a variant calling pipeline it could be the number of variants called, or accuracy of the result (such as recall or precision rates defined in Section 1.2.2). And if a single result cannot be represented as a number, the difference between two pipeline result can possibly be quantified and inserted into the equation, e.g. cardinality of the symmetric difference between two sets of variant calls.

Sampling the parameter space

According to Equation 1.1 the calculation of an elementary effect requires that the two contrasted pipeline runs differ in exactly one parameter value. Then, the elementary effect describes change caused by modification of exactly one parameter, which facilitates straight-forward interpretation. Different strategies exist for sampling parameter values, such that only one parameter value is different for a pair of settings. In a nominal range analysis which uses a local method for sampling parameter values, the parameter values are changed individually, always starting from the nominal parameter values [122] (in pipelines, the nominal values are the default parameter settings). In such an analysis, the search space is very limited and concentrates around the default (nominal) parameter values. Settings where two parameter values differ from the default are not evaluated, and thus the sensitivity screening provides a limited picture of parameter influences [122].

In contrast, a global sampling method probes settings from entire multidimensional parameter space. In the Morris method [125], the sampling is done in the following way. For a set of k parameters ($p_1, ..., p_k$) to be analyzed, let X be a matrix of $k+1$ rows of length k. First row of X is filled with a vector of random parameter values $X_1 = x_1, x_2, ..., x_k$, where x_i ($i \in \{1, 2, ..., k\}$) is a valid value of parameter p_i. In k steps, k subsequent rows of X are filled by copying the preceding row and modifying exactly one random parameter value, e.g.

$$X_2 = x_1, x_2, ..., x_i + \Delta_i, ..., x_j, ..., x_{k-1}, x_k$$
$$X_3 = x_1, x_2, ..., x_i + \Delta_i, ..., x_j + \Delta_j, ..., x_{k-1}, x_k$$

where Δ_i is a predefined modifier for parameter p_i, and $x_i + \Delta_i$ is a valid value of parameter p_i, for any $i \in \{1, 2, ..., k\}$. The order of changed parameter values is determined by a random permutation $\sigma \cdot (1, .., k)$, ensuring value of each parameter is changed exactly once. After the k steps, the matrix X contains $k+1$ rows of valid pipeline settings (parameter values for all the parameters) with the following property:

$$X_{i+1} - X_i = 0, 0, ..., \Delta_{\sigma(i)}, ..., 0, 0,$$
$$\text{for } i \in \{1, 2, ..., k-1\}$$

This property guarantees that exactly one parameter value is modified between any pair of adjacent rows in X. Hence, any such pair satisfies the requirement for calculating an elementary effect (Equation 1.1), and from $k+1$ pipeline settings in X, exactly k elementary effects can be computed. In the space of possible pipeline settings, X represents a trajectory starting in the random point X_1, and traversing the space by jumping k-times a $\Delta_{\sigma(i)}$ distance in the $\sigma(i)$-th dimension. Each of the k jumps produces an elementary effect for a different parameter, and a trajectory can generate one elementary effect for each pipeline parameter requiring $k+1$ pipeline executions.

Computing multiple trajectories allows to sample the parameter space better, and produce several elementary effect per parameter. In consequence, the estimates of parameter effects are more accurate. The random starting point in each trajectory (i.e. X_1) provides incidental spread of the trajectories in the multidimensional parameter space, but cannot guarantee even coverage of all its areas. An improvement aiming at more exhaustive scan of the parameter space, without increasing computational cost was proposed by Campolongo *et al.* [126]. The enhancement depends on generating a considerable number of trajectories to provide a background for choice of a smaller subset of trajectories with maximal dispersion. The distance between two trajectories was defined as a sum of geometric (e.g. Euclidean) distances between every pair of points of the two trajectories. Computing the distances between every pair of trajectories in the background set, allows to find a subset with highest sum of distances, providing maximal dispersion of points in the parameter space. In the sensitivity screening presented in this thesis (Section 3.2), we used the improved sampling strategy.

Parameter value modifications

The parameter values in X must be valid parameter values, before and after modification with Δ. Assuming that parameter p_i takes values from range $[min_i, max_i]$, we define a set P_i such that:

$$P_i = \{\frac{j \cdot (max_i - min_i)}{q_i - 1} + min_i, for j \in \{0, 1, ..., q_i - 1\}\}$$

P_i is a set of q_i points uniformly distributed within the range of possible p_i values. Random values of parameter p_i are sampled from P_i. To ensure that the modified parameter values $(x_i + \Delta_i)$ also belong to P_i, and to facilitate uniform sampling of the values in P_i, the following two constrains are proposed:

$$q_i \mod 2 = 0 \tag{1.2}$$

$$\Delta_i = \pm\frac{(max_i - min_i)}{2} \tag{1.3}$$

Satisfying constraint in Equation 1.2 ensures that P_i can be split into equally-sized halves, i.e. $x_i < min_i + \Delta_i$ and $x_i > max_i - \Delta_i$. A randomly sampled x_i originates with equal probability in any of the halves. Constraint 1.3 guarantees that after modification with an appropriate sign, $x_i + \Delta_i$ is in the other half. Together, 1.2 and 1.3 assure that each pair of corresponding p_i values (i.e. x_i and $x_i + \Delta_i$), and thus each value in P_i has equal probability of being sampled.

Classification of parameters

In [125] the influence of the parameters was classified using two statistics: mean of effects (μ) and standard deviation of the effects (σ). For a parameter p_i analyzed in r trajectories, these statistics are defined as:

$$\mu = \sum_{i=1}^{r} ee_i \tag{1.4}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{r}(\mu - ee_i)^2}{r}} \tag{1.5}$$

Highly influential parameters are characterized by high μ, while high σ indicates non-linearity of parameter effect, or dependence on values of other parameters. However, as observed in [126], non-monotonous parameter effects can nullify each other, resulting in insignificant values of μ. To eliminate such possibility, the authors suggested using mean of absolute effect values (μ^*), as a more representative measure:

$$\mu^* = \sum_{i=1}^{r} |ee_i| \tag{1.6}$$

In the sensitivity screening carried out as part of this thesis (Section 3.2), we used σ and μ^* to describe influence of the parameters.

Chapter 2

Aims of the thesis

1. Investigate the feasibility of using standard Web services in high-throughput pipelines.
2. Improve performance and throughput of pipelines composed of Web services.
3. Explore methods for testing robustness of high-throughput bioinformatics pipelines.
4. Analyze the influence of pipeline parameters on the results of a genetic variant discovery pipeline.

The feasibility of using standard Web services for high-throughput pipelines (Aim 1) is addressed in **Paper I**. Approaches that contribute to improved performance and throughput of Web service pipelines (Aim 2) are presented in **Paper I**, **Paper II**, and **Paper III**. In **Paper IV**, Aims 3 and 4 are addressed by using two sensitivity screening methods to study robustness of a variant discovery pipeline, and the influence of its parameters on the resulting variant calls.

Chapter 3

Contributions

The results of this thesis are described in detail in four articles. Three of the articles address the technical challenges of using Web services in high-throughput bioinformatics pipelines. The fourth demonstrates suitability of sensitivity screening methods to identify influential parameters in a pipeline for genetic variant discovery. Below, the motivation and the results of the individual articles are summarized.

3.1 Performance and throughput of data-intensive workflows composed of Web services

Efficient use of the distributed bioinformatics programs and databases requires a technological platform that will enable seamless integration of heterogeneous resources. The standard Web service technology was identified as such a platform, and proposed to unify the bioinformatics resources by increasing their interoperability. As described in Section 1.1.3, the Web services fit perfectly for this purpose in many aspects, but they lack support for efficient communication of large data encoded in XML. Hence, one of the concerns of the bioinformatics community with respect to Web service technology is related to the possibility of using Web services in high-throughput data processing pipelines. We set out to investigate the feasibility of executing genome-scale pipelines composed of standard Web-services, and explore methods that allow increased performance and throughput of such pipelines, while retaining the highest level of interoperability.

3.1.1 Partitioning and pipelining of the data

Genome-wide data often consists of thousands of elements (e.g. biological molecules, measurements), that although individually small, when aggregated over a genome, become considerable in size. In some analysis workflows, such as sequence annotation or read alignment, these data elements are completely independent. In many others, although the individual data elements are coupled, they can be split sensibly without hindering analysis (e.g. alignment in variant calling). This feature of biological data is frequently used to accelerate time-consuming computations by parallel processing. In

Paper I and **Paper III** we used the data-partitioning strategy to reduce size of individual input portions, making them easier to transfer and handle by Web services.

Paper I presented the pattern of communication for standard Web services that enables partitioning of input and output data. The new pattern of communication was evaluated on a genome-scale bioinformatics workflow for annotation of DNA sequences. The partitioned data case had lower memory requirements, which improved performance of the data-handling by the Web services, allowed for the reduction in workflow run-time, and thus increased throughput. Most importantly, we found that partitioning facilitates processing of unlimited size of data, that is otherwise constrained by the available memory of the Web services. Thus, we could conclude that when used sensibly the standard Web services can be used for genome-scale data analysis.

We discovered that a common way of iterating over single data elements (i.e. partitioning into single units), although scaling well and also eliminating the data-size constraint, was highly inefficient. Too frequent communication resulted in considerable message processing overheads, reduced throughput, and when unrestrained, could break the service. Corresponding observations were made in **Paper III**, where we found that balancing the partition size has a considerable influence on workflow runtime. Also, the experiments in **Paper III**, which were designed to measure the benefits of data partitioning solely in Web service communication, confirmed the findings of **Paper I**.

The partitioned communication presented in **Paper I** was enabled by an extension of the status-polling mechanism for service interaction (Section 1.1.3). The extended interface transformed the service-job into a data-processing stream, allowing to append new data partitions to the already started computation, and at the same time retrieve completed output parts, while the remaining computations were continuing. This way, a Web service could be seen as a processor on a stream of data, enabling overlap of data-processing and transfer (pipelining or service parallelism). To facilitate straightforward development, **Paper I** was accompanied with a software framework that supports implementation of data-partitioning Web services.

3.1.2 Direct data-transfer between Web services

Centralized coordination is the typical model for workflow execution, where a central workflow engine invokes distributed component services, and is an intermediary in the data-traffic between them. In data-intensive workflows, brokering the voluminous data traffic is resource demanding, and limits performance and scalability of the workflow system. To remove this bottle-neck and reduce the cost of workflow execution, the responsibility for the data transfer can be distributed between component services. In [127] and **Paper II**, we proposed how this can be implemented in standard SOAP Web services.

The *Data-Flow Delegation* was theoretically defined and evaluated in [127], and it allows the workflow engine to dynamically instruct a service to forward its output data to another service. **Paper II** contributed a non-intrusive implementation of the *Data-Flow Delegation* model for Web services, and experimentally examined performance improvement on a genetic sequence annotation pipeline. The results showed that by delegating responsibility for data-communication to the services, resource consumption of the workflow engine is nearly eliminated. At the same time, we did not observe any significant decrease in performance of the component services, despite the added functionality for data-forwarding. Workflows that use the *Data-Flow Delegation* have lower aggregated cost of execution, and shorter run-time. We found that these performance gains are proportional to the data-size.

The dynamic invocation of services for the purpose of forwarding the data was based on automatic parsing of the standard Web service interface description (WSDL), and could easily be abstracted into a generic library. The support for *Data-Flow Delegation* on the service level requires overloading the service operation with an optional element, allowing it to be invoked with and without the *Data-Flow Delegation* functionality. In **Paper III**, we have shown that with a minor extension to the overloading element, Web services could support data partitioning to allow scalability with respect to data-size, and to further improve workflow run-time and throughput.

3.2 Robustness to parameter changes in bioinformatics pipelines

The results of high-throughput bioinformatics analysis pipelines are increasingly more often seen as relevant in a clinical context, and thus ensuring accuracy and robustness of the computational methods is critical. Pipeline parameters are a known source of variability in the results, but the extent of their influence and robustness against parameter changes have not been explored systematically. A method that can evaluate pipeline robustness and identify the most influential parameters, could help assessing pipeline quality and guide informed selection of pipeline settings.

In **Paper IV**, we used a method for global sensitivity screening to assess robustness, and identify influential parameters in a pipeline for genetic variant discovery. We discovered that effects of the parameters were highly inter-dependent or non-linear, and that only a few of the 24 tested parameters had a substantial effect on the results. Four out of five of the most influential parameters tuned the variant calling step of the pipeline. To compare the effects of parameter changes on the resulting variant calls, we developed a set of metrics to quantify the difference between the calls. Irrespective of the metric used, the same parameters were indicated as being highly influential.

By using a reference set of variants in one of the metrics, we were able to relate the parameter effects to variant calling accuracy. This metric also enabled quantification of precision and recall rates for all of the tested combinations of parameter values. We found that modifying parameter values can considerably change the accuracy of variant calling, but the pipeline was robust to modest changes in the default settings. Although the default pipeline settings offered very good compromise between accuracy and recall, we identified combinations of parameter values that yielded higher precision and higher recall rate, in comparison with the default settings. Using these results, we showed and proved on a different sample, that the influential parameters effectively controlled the balance between recall and precision in variant calls, and that they are promising targets for parameter settings optimization.

Finally, to evaluate the global sensitivity screening method, we contrasted it with a cost-efficient local-screening approach. The local method identified nearly the same parameters, but it was not able to quantify the predictability and monotonicity of parameter effects, nor provide the comprehensive results that laid ground for settings optimization. On the other hand, the results of the local screening showed robustness of the pipeline for 'near-default' settings. Considering the advantages of the local and the global screening methods, we outlined an efficient sampling strategy dedicated to evaluating sensitivity of parameters, and robustness of computational pipelines.

Chapter 4

Discussion

The accessibility of deep-sequencing and the prospects it offers for personalized medicine, pushes the technology into the clinic [93]. Consequently, one of the main challenges in modern biomedicine is to bridge the gap between ease of data generation, and the complexity of processing and analyzing it [96], for the purpose of explaining genetic causes of diverse medical conditions [92, 94]. Addressing this challenge requires providing complex bioinformatics methods in an accessible manner [92, 95, 44]. The standardized Web service technology promoted in **Papers I-III** facilitate building a modularized infrastructure, that uses interoperable components; allows for flexible scaling; and can hide complexity of computations, integration, and distribution of resources. Provided easy-to-use integration frameworks, even complex bioinformatics analysis (such as the genetic variant discovery pipeline studied in **Paper IV**) could be executed by medical personnel from hospital premises. The contributions from **Papers I-III** help to increase the throughput of pipelines composed of standard Web services, allow to produce computational results faster and cheaper, and contribute to bridging the cost-gap between data generation and its analysis [96]. The increased throughput, together with standardized interfaces that ease automation, are fundamental for enabling *in silico* experiments that previously were unfeasible or down-prioritized due to the costs involved, such as the study of parameter sensitivity presented in **Paper IV**.

4.1 High-throughput bioinformatics pipelines

High-throughput data analysis using distributed bioinformatics resources requires a technology that would allow interoperable communication over the Internet, and at the same time provide a mechanism for efficient data transport. The technology of Web services meets the requirements for building the interoperable infrastructure, but the usability of Web services for high-throughput bioinformatics pipelines is limited due to inefficient data-handling. To investigate the applicability of the technology to data-intensive pipelines, we proposed and evaluated two approaches that improve throughput of pipelines of Web services. In **Paper I**, we showed that throughput of a Web service, and consequently a pipeline, can be largely improved by partitioning the data and stream-like invocation pattern. **Paper II** showed that the capacity of a workflow engine can be increased by relieving it from the data-transport mediation, in consequence allowing orchestration of larger number of workflows. And finally, in **Paper III** we demonstrated that the two approaches are complementary, and applying them together results in greatest performance gain.

4.1.1 Improving throughput of pipelines composed of Web-services

Data partitioning

The approach presented in **Paper I** was based on the observation that high-throughput datasets are frequently composed of thousands of independent elements of the same type (e.g. genes, proteins). Processing of such data can be easily parallelized, and hence, data-parallelism [79] is a commonly practiced technique for optimizing pipelines [40]. Web services processing large data, although having access to powerful high-performance computing resources, can be constrained by the throughput of their Web service interfaces. To aid this, we proposed a stream-like interface and a communication pattern that allow highly-scalable Web service messaging using partitioned data. We showed that communication with partitioned data has considerably lower resource demands with respect to XML serialization and de-serialization, and allows for transferring arbitrarily large data in appropriately-sized partitions. Lowered resource demands, increase responsiveness and throughput of the Web service communication, and consequently, decrease workflow runtime. In addition, the independent data partitions enable pipelining of the data between consecutive services in a workflow (service-parallelism [79]), which further contribute to reduction in workflow runtime by overlapping computation and data-transport. Overall, **Paper I** demonstrated that using data partitioning, execution of genome-size pipelines composed of standard Web services is feasible.

A workflow engine supporting both data- and service-parallelism (e.g. Taverna [58]), could provide similar benefits as data-partitioning, when invoking batch-processing services. However, the limitation on the number of concurrently open service sessions (and threads controlling them) [80] largely constrains scalability and throughput of such a pipeline. The stream-like Web service interface proposed in **Paper I** is free of this constraint and allows higher throughput. Implementing a workflow that uses the stream-like interfaces is equally easy as other standard Web services, but handling the data-streams can be rather complex, requiring two threads for every service. A pipeline composed of several data-partitioning services would require that the workflow engine handles multiple threads, each sending and receiving many messages with data partitions. Hence, although the approach scales in terms of data size, the number of partitioning services that can be efficiently coordinated by one workflow engine is limited. Using the Data-Flow Delegation method proposed in **Paper II** can alleviate this limit.

Data-Flow Delegation

For classical service orchestration, it has been shown that a direct data-traffic between services, instead of via the workflow engine, can improve aggregated performance of a workflow system [72]. The model of *distributed data-flow* was applied to many different types of services [75, 77, 78], including proxies to standard Web services [73].

In **Paper II**, we proposed how this model can be implemented in standard SOAP Web services, using Data-Flow Delegation. The evaluation of the implemented solution was carried out on a gene annotation pipeline, and it supported the findings of the previous studies that assessed the *distributed data-flow* model using other types of services [72, 78]. We observed that Data-Flow Delegation nearly eliminated the load on the workflow engine, without making a noticeable difference to the services. The minimal resources required to coordinate one pipeline allow the workflow engine to coordinate multiple pipelines in parallel, considerably increasing its throughput.

The approach of Data-Flow Delegation is neutral to existing clients, since it is based on overloading operation signatures in the service interface (WSDL). This allows the workflow system to use both regular (centralized) and the distributed data-flow model, depending on the requirements; makes it easy to mix and match in one pipeline Web services supporting Data-Flow Delegation and regular Web services; and allows to route the intermediate data via the coordinator for data-provenance purposes. It can be argued that peer-to-peer communication between services increases coupling between workflow components. However, the connection is established dynamically based on the instruction obtained from the workflow engine, and services have no prior knowledge about their interaction partners. The support for on-the-fly data conversion offered by the workflow engines that mediate the data-flow (e.g. ”shims” in Taverna [54], or XSLT in BPEL engines), was not included in the implementation proposed in **Paper II**. However, this could easily be supported: simple data conversions described using standard methods for XML manipulation (e.g. XQuery[1], XSLT[2]) could be sent as part of the data-delegation instruction from the engine, and be executed by the service.

While experimenting with data-partitioning (**Paper I**) we found that the granularity of Web service communication is an important factor in the overall performance of a workflow system. Consequently, the software framework we provided allowed for custom sizes of data partitions, helping to balance the frequency of messaging and partition sizes. In **Paper III**, we combined data-partitioning and Data-Flow Delegation, and showed that the messaging granularity has an optimum. This optimum can be found by probing different message (data partition) sizes, and ideally, in a workflow system that supports data-partitioning, the granularity and the frequency of communication should dynamically adapt to the varying load in the system.

Complementarity of the two approaches, data-partitioning and Data-Flow Delegation, was demonstrated in **Paper III**. Data-partitioning enables processing of unlimited data sizes, data-, and service-parallelism, which are not possible with Data-Flow Delegation alone. In turn, the Data-Flow Delegation helps in concurrent handling of multiple data-partition streams in a workflow engine, by reducing the resources required for workflow coordination. Both enhancements increase performance of Web service pipelines, enabling execution of genome-scale workflows, as shown in **Paper I**. Although semantically similar to other approaches, they are the only ones implemented using standard Web services. Hence, the optimized services can be used in all systems which support the execution of standard Web services, such as Taverna, Galaxy [50], and BPEL [61] engines.

[1]http://www.w3.org/TR/xquery/
[2]http://www.w3.org/TR/xslt

Maintaining full interoperability between the optimized Web services and other components of the Web service environment was our main design principle. The proposed enhancements do not interfere with the Web service communication stack, are designed on the application level, on top of ordinary Web service libraries, and use exclusively standard technology assets. The generality of the approaches is also demonstrated in the fact that the experiments were carried out using implementations in different programming languages and using different Web service libraries. Ideally, both data-partitioning and data-flow delegation should be supported on the level of Web service definition and communication, including interface description (WSDL), messaging (SOAP), and code generation by Web service libraries. Such support would provide more transparency to the programmer, reduce software engineering effort needed to develop and invoke optimized Web services, and possibly further improve the performance.

4.1.2 Other mechanisms of optimizing data transport in standard Web services

Exchanging data among Web services using textual XML is associated with large performance loss, but it has numerous advantages: complete typing of Web service interface, message validation, and fully transparent transition from the textual to in-memory representation of the data (data-binding). Thanks to data-binding, the programmer ”can abstract from the textual XML appearance of the data exchanged by SOAP messages, and regard it as a medium for directly transferring data objects” [11]: the data is ’parsed-on-arrival’ and accessible as objects in the code. From the interoperability perspective, these are undoubtedly important features, and they motivated our focus on Web services exchanging all data in XML. However, considering the rapid growth in data-production due to the high-throughput technologies, arguing for representing all data in textual XML is unreasonable, especially if dedicated canonical file formats exist, for instance SAM and BAM [20] for deep-sequencing data.

The enhancements presented in **Papers I-III** offer the greatest performance gain for services that exchange all data in XML, but they can also improve performance when the bulk of the data is encoded in a more efficient format. As pointed out by van Engelen [64], the Web service technology does not necessitate encoding all the data in XML. Web services provide several standard optimization mechanisms that improve their performance in handling data, e.g. message compression, binary attachments (MTOM[3]), and HTTP chunking. For instance, a service being simply the Web interface to a program that takes a BAM file as input, could get the input file sent as an attachment to bypass the costly XML encoding/decoding [34], whereas additional data pertaining to the request (e.g. parameters), could be encoded directly in the SOAP message. The approach of isolating the bulky input from the rest of the SOAP message has practically no drawbacks, if the data is represented in a canonical format. Otherwise, it is difficult to document the expected format of the attachment, as the Web service interface description (WSDL) does not allow detailed format definitions for attached files (beyond MIME types). Providing support for this could lead to increased interoperability

[3]http://www.w3.org/TR/soap12-mtom

of services that exchange data in files instead of XML documents.

A recent development which can considerably advance efficiency of XML processing, is the new W3C standard for binary encoding of XML - Efficient XML Interchange (EXI)[4] [128]. EXI is a representation of XML, which can be compressed using detailed information about the structure of an XML document (its XML Schema) to significantly reduce data volume, and at the same time improve performance of encoding/decoding [129]. The format may enable use of XML for types of data that previously could not be represented in XML for practical reasons, allowing them to take advantage of the XML technology. Providing universal support for EXI in many Web service libraries will be critical for ensuring interoperability between distinct programming languages, and wide-adoption of the standard.

The standardized optimization mechanisms improve performance of Web services considerably [64, 129], and are able to mitigate problems related to data processing in many pipelines that otherwise could not be implemented using Web services. However, for multi-step pipelines that process deep-sequencing data, repetitive transmission of large files over the network, while possible, is not an optimal solution. Approaches that facilitate co-location of pipeline components, and centralized computations on the data, can offer much better performance.

4.1.3 Distributed versus centralized data analysis

We chose a distributed model of data analysis, where the pipeline components are geographically distributed and accessed remotely using Web service interfaces. Such a model allows for the loose-coupling of the tools and data-sources in composite pipelines, which are executed by a workflow engine, e.g. Taverna or BPEL servers [130]. The distributed model of execution nicely mirrors the distributed nature of bioinformatics resources, but carries the aforementioned challenge of transmitting large data between them. Therefore, majority of the bioinformatics workbenches rely mainly on centralized data analysis, e.g. Galaxy, Chipster [53], and GenePattern [49]. Tools in these workbenches are installed on the same system as the workbench, and the large public datasets being part of the analysis are pulled into the system prior to the analysis. Consequently, all the elements of an analysis pipeline are in proximity and the large data does not need to be transferred over the network between consecutive pipeline steps.

However, deploying resources locally on an institutional server is not always possible [8], for instance due to platform incompatibility, computational demands [131], or sheer size [35]. From the perspective of a workbench user, the locally-installed toolbox may seem inflexible and hinder rapid workflow prototyping, since adding and configuring tools requires assistance of the system administrator [50]. And last, but not least, the maintenance of a system with a diverse portfolio of applications is a tedious task, requiring dedicated personnel and resources. In contrast, the workflow engines that coordinate distributed resources do not need a powerful server, since the computations are performed remotely; they require no maintenance of the tools since responsibility

[4]http://www.w3.org/TR/exi/

for this is distributed among tool providers; and the toolbox can dynamically be extended by simply providing a link to the interface of a new tool. On the other hand, the workflow engine has no control over the external pipeline components, so the heterogeneity and quirks of the individual resources [9, 34, 30] cannot be easily hidden from the workflow creator. This is a challenge for the accessibility of such systems, as composition of pipelines from Web services frequently requires programming skills (e.g. ”shims” in Taverna [54, 58]).

Geographical data distribution in bioinformatics is a fact, and both, the centralized and the distributed analysis systems are dependent on accessing external resources. The distributed systems however make more extensive use of the remote resources, which in case of high-throughput data can noticeably extend the analysis time. Consequently, in many cases the centralized approach to high-throughput data analysis is admittedly more practical, e.g. the variant calling pipeline in **Paper IV** was implemented as a shell script invoking locally installed programs. In practice, the two opposite models of pipeline execution are intermixed; for instance Taverna provides support for computations using local scripts [58], and Galaxy allows Web service invocation [132]. Also other analysis environments [27] and workflow systems [55, 57] support both models. We believe that when practiced sensibly, taking advantage of efficient methods for data transport (e.g. discussed in Sections 4.1.1 and 4.1.2), the distributed approach to execution of high-throughput data analysis pipelines can be an interesting alternative to the centralized model.

4.1.4 Outlook

In application areas like deep-sequencing data analysis, data exchange using binary XML or dedicated canonical binary file-formats seems the only viable solution for distributed analysis. However, there is a tipping point when sending the sizable data over the network becomes more resource demanding than transferring the program with all its dependencies, to facilitate local computation where the data resides. Analysis of such sizable datasets can be facilitated by a dynamic infrastructure where the programs have the ability to relocate closer to the data [35]. A promising technology for realizing such a model is hardware virtualization, which facilitates dynamic instantiation of virtual machines (servers). These virtual machines can be pre-configured computational environments (e.g. CloudBioLinux [131]) ready to be sent for instantaneous deployment to the cloud where the large data is located. Provided Web service interfaces to the tools installed on the virtual machine (like in the JABAWS toolkit [133]), both the management of the cloud infrastructure[5] and the analysis could be controlled programmatically, using a homogeneous technology. Consequently, the transfer of the analysis to the data could happen transparently to the user, and be decided by the workflow engine considering feasibility and the cost.

The cloud infrastructure enables dynamic relocation of the computation to the data, but if the results of the analysis are not much smaller than the input dataset, the approach is not advantageous. Then, the mobile analysis environment needs to provide tools for

[5]Amazon, at present the largest provider of cloud computing resources worldwide, offers Web service interfaces for managing cloud infrastructure

interactive manipulation on the large result (e.g. using Galaxy such as in CloudMan [134]). Also, the size of the experimental input dataset can sometimes be negligible in comparison with the reference data used, for instance the 1000 Genomes [135] dataset. The EBI, which hosts major public data repositories (including reference data), offers a private cloud solution for its collaborating partners[6] who need high-throughput access to one of the resources. It is clear that such a model cannot scale for all the researchers accessing these data collections.

Sustainable infrastructure for high-throughput data analysis in bioinformatics will require replication of the major public resources across several compute centers. Access must be offered in multiple interoperable ways including, among others, canonical cloud infrastructure and Web services. In Europe, efforts towards this goal are coordinated by the collaborative ELIXIR initiative [136]. The responsibility of the tool providers is to enable transparent access to their software, and furnish both, the life science researchers and bioinformaticians, with convenient interfaces that hide the technical complexity of distributed data analysis, and heterogeneity of the data. It is evident that the Web service technology will play a role in building this infrastructure.

4.2 Robustness of bioinformatics pipelines

Providing easy-to-use infrastructure that facilitates efficient analysis of high-throughput data is vital to enable the use of deep-sequencing in routine clinical applications [92, 95, 44]. In addition to accessibility, another challenge is ensuring robustness of these methods [110, 95, 44]. In **Paper IV** we investigated the possibility of using sensitivity analysis methods to evaluate robustness of a genetic variant calling pipeline. The examined pipeline was a state-of-the-art variant calling method employed in many medical genetic studies [137–139]. It has a potential for becoming a tool supporting medical decisions [92, 94, 140, 141] and as such, warrants critical evaluation of accuracy, robustness, and reproducibility [110, 95, 44]. Complementary to the characterization of influence of the sequencing technologies [142, 98, 143, 110], and influence of pipeline components [114, 110] on the variant calling accuracy, **Paper IV** provided insights into the robustness of the computational pipeline, and the impact of varying pipeline parameter values on the result.

4.2.1 Influential parameters

The screening of parameter sensitivity carried out in **Paper IV** revealed that it is difficult to reliably predict impact of a single parameter change. Parameter effects in the studied variant calling pipeline were highly inter-dependent and/or non-linear. Due to the extent of the comprehensive screening, we could describe these effects more precisely using 50 single effects per parameter. This way it was possible to identify and characterize a group of parameters that displayed the largest impact across several effect

[6]personal communication

measures. Surprisingly, four out of five of these parameters tuned behavior of the variant calling step, and only one parameter belonged to the aligner. Counter-intuitively, this does not stand in contrast to a general belief that an accurate read alignment step is crucial in the pipeline [97]. Rather it indicates that the parameters of the variant caller are more sensitive to changes than those of the aligner, and that they should be tuned with care. Such information is valuable to researchers adjusting pipeline settings and tool providers that can mark sensitive parameters for the benefit of software users. Correspondingly, the least-influential parameters could be hidden from the users to decrease interface obscurity. Together with documented sensitive parameters, this can contribute to improved accessibility of the bioinformatics methods [95]. Also tool developers would benefit from the feedback aiming at increasing robustness of their software.

Similarly to a benchmarking study, the results of parameter sensitivity analysis can depend on the input dataset. Obtaining truly robust results in such an experiment requires analyzing several datasets to compensate for both, technological and biological variability. In case of deep exome-sequencing, the panel of samples would need to encompass biological replicates, different exome-capture techniques, and distinct sequencing platforms (similarly as in [110]). In **Paper IV**, a selection of findings was validated on one biological sample, which limits the generality of conclusions that can be drawn from the quantitative results. In contrast, the qualitative results pertaining to the utility of such an analysis to computational pipelines, are more universal. Thus, **Paper IV** should be viewed as a pilot experiment surveying plausible benefits of applying sensitivity analysis to bioinformatics pipelines, rather than a detailed characterization of parameters of a particular pipeline.

4.2.2 Optimization prospects

The variant calling pipeline analyzed in **Paper IV** was robust to modifications of single parameter values in the default settings, and showed good accuracy (both recall and precision) for the ’near-default’ settings. However, more divergent settings could produce results with considerably diverse accuracy. Obtaining accurate variant calls from deep-sequencing data is challenging [105], and in **Paper IV** we have shown that the influential parameters identified in the sensitivity screening can play a key role in customizing variant calling accuracy. Hence, sensitivity analysis can contribute to the search for settings that yield the desired accuracy of the results.

Naturally, the combinatorial complexity of finding the optimal parameter values can be reduced by disregarding the non-influential parameters of the pipeline (as suggested for expensive sensitivity analysis methods [122, 124]). Also, the parameters that are characterized with monotonic influence by the sensitivity analysis can be directly used to optimize the result. Among the parameter values sampled and evaluated for the purpose of sensitivity analysis, we found combinations of parameter values that yielded very high precision, and very high recall rate. These settings can serve as seeds for local optimization towards any of the two measures, depending on the need: ”Mendelian disease projects can select settings that provide a more inclusive set of calls with a higher error rate to avoid missing that single, high-impact variant, whereas community resource projects like the 1000 Genomes Project [135] can prioritize precision" [105]. Reverting all the non-influential parameters in the high-scoring settings back to their default values, resulted in nearly no change in accuracy. This is a notable discovery indicating that the values of the few influential parameters can effectively control recall and precision of genetic variant calling, and that these parameters are the most interesting candidates in the search of optimal settings.

A critical component of sensitivity evaluation is comparison of pipeline outputs. In **Paper IV** we have shown that even elementary measures (like set cardinality and overlap) can capture parameter influence and facilitate robustness assessment. However, in order to maximize utility of the findings, the parameter influence should be represented in terms pertaining to the quality of the pipeline, for instance result accuracy. We have proposed an accuracy metric for variant calls, but it is slightly biased towards the results obtained with the default settings, and in practice only applicable to samples sequenced to a very high coverage. A similar measure of accuracy was used in [144], whereas Heinrich et al. [145] proposed a sample- and platform-independent metric based on genotype frequencies observed in the 1000 Genomes Project [135]. Emergence of such metrics enables the comparison of results across heterogeneous datasets and is an important step towards standardized quality control of genetic variant discovery.

Considering the likely medical future of variant calling, standardization of the methodology must take place at some point. This will require, for instance, decisions that stabilize the set of tools used and their parameter values. The presented sensitivity analysis, combined with parameter optimization, can prove useful in the making of such decisions.

4.2.3 Sensitivity screening

In addition to the global screening that provided the main results, **Paper IV** also presented results obtained using a local (nominal range) sensitivity analysis method, which can be considered an intuitive approach to searching for good parameter values. Despite a great difference in the computational cost, both methods showed a surprising correspondence in the rankings of influential parameters. However, the global screening method delivered much more comprehensive insights: it identified predictability of parameter changes, parameter inter-dependency, and monotonicity of the effects; it provided the "context" in which the local sensitivity analysis could be recognized as an effective method; and allowed establishing the relationship between the influential parameters and accuracy optimization. Although these are relevant merits with practical utility for pipeline users, the cost of performing a global screening on a high-throughput pipeline can be daunting. Hence, it is important to identify cost-effective methods that can comprehensively characterize parameter effects. In **Paper IV** we suggested that a screening strategy that concentrates intensive screening on the area close to the default settings can have desirable properties for the analysis of computational pipelines. Other relevant sensitivity screening methods should also be identified and evaluated, in order to broaden the selection of techniques supporting robustness analysis of pipelines.

The screening methods provide mainly qualitative results for ranking parameters according to importance. To precisely gauge parameter sensitivity of a model, the most influential parameters identified in screening are often studied further using quantitative methods [122]. Since **Paper IV** settled on the screening alone, the practical value of performing complete sensitivity analysis of a computational pipeline remains unknown. Determining it should enable a cost-benefit comparison with screening approaches, that can suggest if the extra cost involved in quantitative sensitivity analysis can be translated into a boost in the quality of a pipeline or pipeline results.

Robustness of bioinformatics tools and pipelines have not been extensively studied, and many details of this practice remain to be established. **Paper IV** shows utility of sensitivity analysis methods for this purpose, and can hopefully trigger more efforts towards assessing robustness of bioinformatics pipelines, especially in applications where consistency of pipeline results is critical.

Chapter 5

Conclusion

Web services are popular for providing access to data and computations in bioinformatics. We examined their utility for building high-throughput pipelines, and conclude that using conventional Web service implementations is highly inefficient, and can even be unfeasible. By waiving some of the benefits of using the XML technology, which is at the heart of the problem, standard optimization mechanisms provided with Web services are able to mitigate the inefficiency of processing large data. Our contributions complement these mechanisms by providing two approaches to Web service communication that can facilitate execution of high-throughput pipelines without losing the benefits of representing data in XML. Having the choice, developers of bioinformatics resource can decide to represent the data using canonical file- or XML-based formats, depending on what is more suitable, as opposed to what is feasible, in high-throughput pipelines.

Although sensible use of the Web service technology allows for execution of data-intensive pipelines, the distributed analysis of large data is not the most efficient approach. The overhead of frequent data transfers between remote sites is a substantial disadvantage when compared to centralized data processing, and continues to be such even when weighted with the convenience of distributed responsibility for maintenance and development of the resources. Processing increasingly larger datasets demands centralized computing infrastructure equipped with a broad and extensible analysis toolbox, and convenient remote access methods, both programmatic and interactive. Due to its flexibility and scalability, the cloud computing paradigm will be popular for building such infrastructures, and the programmatic coordination of the cloud-based resources will continue to be realized using Web services.

The cloud can provide the necessary transparency of the complex computational infrastructure needed for bioinformatics analysis, and ease development of accessible and powerful tools for biomedical researchers and clinicians. Increased throughput and automated interoperable access to resources can lay a solid foundation for large-scale meta-experiments on high-throughput pipelines, such as the analysis of the impact of parameter settings on pipeline results. In that study, we explored the usability of two sensitivity screening methods to assess the robustness of a variant discovery pipeline. We showed that even using relatively low-cost approaches, robustness of a high-throughput pipeline can be evaluated, but with a more expensive global screening method, a comprehensive characterization of parameter influence is possible. Although the number of biological replicates used in the experiment did not permit drawing general conclusions about the parameters of the analyzed pipeline, our findings clearly indicate possible application areas for the results of a sensitivity screening in a computational pipeline. The information about the most influential parameters can be used to simplify parameterization of complex computational methods, and to document the effect of prominent parameters in terms of, for instance, accuracy. The discovered link between the influential parameters and guided accuracy optimization (towards sensitive or precise calls), can be used to tune pipeline settings according to the needs of a particular study. And finally, the results of the sensitivity screening can aid choice of components in computational pipelines, and help to fixate pipeline settings to ensure the most reliable results in a clinical setting.

To our knowledge, the comprehensive and systematic survey of pipeline parameter effects was a pioneering effort. Consequently, many methodological aspects of robustness analysis of computational protocols remain to be evaluated and established. We hope that the outlined prospects of studying parameter sensitivity in computational pipelines can spark further research in this area, in particular when considering the future of deep-sequencing analysis pipelines in routine clinical services. As such experiments on high-throughput pipelines are computationally demanding, systematic evaluations of computational methods will require side-by-side developments in both the methodology for pipeline evaluation, and the computing infrastructure. Taking the full advantage of the potential of the Web service technology, I expect that such examinations will be largely automated in the future.

Bibliography

[1] Goble, C & Stevens, R. (2008) State of the nation in data integration for bioinformatics. *Journal of Biomedical Informatics* 41, 687–93. 1.1, 1.1.1, 1.1.1, 1.1.2, 1.1.3

[2] Yu, J & Buyya, R. (2005) A taxonomy of workflow management systems for grid computing. *Journal of Grid Computing* 3, 171–200. 1.1, 1.1.2

[3] Altschul, S. F, Gish, W, Miller, W, Myers, E. W, & Lipman, D. J. (1990) Basic local alignment search tool. *Journal of Molecular Biology* 215, 403–410. 1.1, 1.1.1

[4] Ashburner, M, Ball, C. A, Blake, J. A, Botstein, D, Butler, H, Cherry, J. M, Davis, A. P, Dolinski, K, Dwight, S. S, Eppig, J. T, Harris, M. A, Hill, D. P, Issel-Tarver, L, Kasarskis, A, Lewis, S, Matese, J. C, Richardson, J. E, Ringwald, M, Rubin, G. M, & Sherlock, G. (2000) Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature Genetics* 25, 25–29. 1.1

[5] Binns, D, Dimmer, E, Huntley, R, Barrell, D, O'Donovan, C, & Apweiler, R. (2009) QuickGO: a web-based tool for Gene Ontology searching. *Bioinformatics* 25, 3045–3046. 1.1

[6] Fernández-Suárez, X. M & Galperin, M. Y. (2013) The 2013 Nucleic Acids Research Database Issue and the online Molecular Biology Database Collection. *Nucleic Acids Research* 41, D1–D7. 1.1.1

[7] Brazas, M. D, Yamada, J. T, & Ouellette, B. F. F. (2010) Providing web servers and training in bioinformatics: 2010 update on the Bioinformatics Links Directory. *Nucleic Acids Research* 38, W3–W6. 1.1.1

[8] Smedley, D, Swertz, M. a, Wolstencroft, K, Proctor, G, Zouberakis, M, Bard, J, Hancock, J. M, & Schofield, P. (2008) Solutions for data integration in functional genomics: a critical assessment and case study. *Briefings in Bioinformatics* 9, 532–44. 1.1.1, 1.1.2, 1.1.3, 4.1.3

[9] Stein, L. (2002) Creating a bioinformatics nation. *Nature* 417, 119–120. 1.1.1, 1.1.1, 1.1.2, 1.1.3, 4.1.3

[10] Ison, J, Kalaš, M, Jonassen, I, Bolser, D, Uludag, M, McWilliam, H, Malone, J, Lopez, R, Pettifer, S, & Rice, P. (2013) EDAM: an ontology of bioinformatics operations, types of data and identifiers, topics and formats. *Bioinformatics* 29, 1325–1332. 1.1.1

[11] Kalaš, M, Puntervoll, P, Joseph, A, Bartaševičiūtė, E, Töpfer, A, Venkataraman, P, Pettifer, S, Bryne, J. C, Ison, J, Blanchet, C, Rapacki, K, & Jonassen, I. (2010) BioXSD: the common data-exchange format for everyday bioinformatics web services. *Bioinformatics* 26, i540–i546. 1.1.1, 4.1.2

[12] Oinn, T, Greenwood, M, Addis, M, Alpdemir, M. N, Ferris, J, Glover, K, Goble, C, Goderis, A, Hull, D, Marvin, D, & et al. (2006) Taverna: lessons in creating a workflow environment for the life sciences. *Concurrency and Computation: Practice and Experience* 18, 1067–1100. 1.1.1, 1.1.2, 1.1.2

[13] Hucka, M, Finney, A, Sauro, H. M, Bolouri, H, Doyle, J. C, Kitano, H, & et al. (2003) The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* 19, 524–531. 1.1.1

[14] Westbrook, J, Ito, N, Nakamura, H, Henrick, K, & Berman, H. M. (2005) PDBML: the representation of archival macromolecular structure data in XML. *Bioinformatics* 21, 988–992. 1.1.1

[15] Li, H & Durbin, R. (2009) Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics* 25, 1754–1760. 1.1.1, 1.2.1, 1.5

[16] Dysvik, B & Jonassen, I. (2001) J-Express: exploring gene expression data using Java. *Bioinformatics* 17, 369–370. 1.1.1

[17] Waterhouse, A. M, Procter, J. B, Martin, D. M. A, Clamp, M, & Barton, G. J. (2009) Jalview version 2—a multiple sequence alignment editor and analysis workbench. *Bioinformatics* 25, 1189–1191. 1.1.1

[18] Cline, M. S, Smoot, M, Cerami, E, Kuchinsky, A, Landys, N, Workman, C, Christmas, R, Avila-Campilo, I, Creech, M, Gross, B, et al. (2007) Integration of biological networks and gene expression data using Cytoscape. *Nature protocols* 2, 2366–2382. 1.1.1

[19] Danecek, P, Auton, A, Abecasis, G, Albers, C. a, Banks, E, DePristo, M. a, Handsaker, R. E, Lunter, G, Marth, G. T, Sherry, S. T, & et al. (2011) The variant call format and VCFtools. *Bioinformatics* 27, 2156–8. 1.1.1

[20] Li, H, Handsaker, B, Wysoker, A, Fennell, T, Ruan, J, Homer, N, Marth, G, Abecasis, G, Durbin, R, et al. (2009) The sequence alignment/map format and SAMtools. *Bioinformatics* 25, 2078–2079. 1.1.1, 1.5, 4.1.2

[21] Magrane, M & Consortium, U. (2011) UniProt Knowledgebase: a hub of integrated protein data. *Database* 2011, 1–13. 1.1.1

[22] Dinkel, H, Michael, S, Weatheritt, R. J, Davey, N. E, Van Roey, K, Altenberg, B, Toedt, G, Uyar, B, Seiler, M, Budd, A, et al. (2012) ELM–the database of eukaryotic linear motifs. *Nucleic Acids Research* 40, D242–D251. 1.1.1

[23] Hamosh, A, Scott, A. F, Amberger, J. S, Bocchini, C. a, & McKusick, V. a. (2005) Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders. *Nucleic Acids Research* 33, D514–D517. 1.1.1

[24] Johnson, M, Zaretskaya, I, Raytselis, Y, Merezhuk, Y, McGinnis, S, & Madden, T. L. (2008) NCBI BLAST: a better web interface. *Nucleic Acids Research* 36, W5–W9. 1.1.1

[25] Hollup, S. M, Salensminde, G, & Reuter, N. (2005) WEBnm@: a web application for normal mode analyses of proteins. *BMC Bioinformatics* 6, 52. 1.1.1

[26] Glez-Peña, D, Lourenço, A, López-Fernández, H, Reboiro-Jato, M, & Fdez-Riverola, F. (2013) Web scraping technologies in an API world. *Briefings in Bioinformatics*. 1.1.1, 1.1.3

[27] Gentleman, R. C, Carey, V. J, Bates, D. M, Bolstad, B, Dettling, M, Dudoit, S, Ellis, B, Gautier, L, Ge, Y, Gentry, J, & et al. (2004) Bioconductor: open software development for computational biology and bioinformatics. *Genome Biology* 5, R80. 1.1.1, 1.1.2, 4.1.3

[28] Cock, P. J. a, Antao, T, Chang, J. T, Chapman, B. a, Cox, C. J, Dalke, A, Friedberg, I, Hamelryck, T, Kauff, F, Wilczynski, B, & et al. (2009) Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics* 25, 1422–1423. 1.1.1, 1.1.2

[29] Stajich, J. E, Block, D, Boulez, K, Brenner, S. E, Chervitz, S. a, Dagdigian, C, Fuellen, G, Gilbert, J. G. R, Korf, I, Lapp, H, & et al. (2002) The Bioperl toolkit: Perl modules for the life sciences. *Genome Research* 12, 1611–1618. 1.1.1, 1.1.2

[30] Stockinger, H, Attwood, T, Chohan, S, Cote, R, Cudre-Mauroux, P, Falquet, L, Fernandes, P, Finn, R, Hupponen, T, Korpelainen, E, & et al. (2008) Experience using web services for biological sequence analysis. *Briefings in Bioinformatics* 9, 493–505. 1.1.1, 1.1.2, 1.1.3, 4.1.3

[31] Neerincx, P. B. T & Leunissen, J. a. M. (2005) Evolution of web services in bioinformatics. *Briefings in Bioinformatics* 6, 178–188. 1.1.1, 1.1.3

[32] McWilliam, H, Li, W, Uludag, M, Squizzato, S, Park, Y. M, Buso, N, Cowley, A. P, & Lopez, R. (2013) Analysis tool web services from the EMBL-EBI. *Nucleic Acids Research* 41, W597–W600. 1.1.1

[33] Flicek, P, Ahmed, I, Amode, M. R, Barrell, D, Beal, K, Brent, S, Carvalho-Silva, D, Clapham, P, Coates, G, Fairley, S, & et al. (2013) Ensembl 2013. *Nucleic Acids Research* 41, D48–55. 1.1.1

[34] Katayama, T, Arakawa, K, Nakao, M, Ono, K, Aoki-Kinoshita, K. F, Yamamoto, Y, Yamaguchi, A, Kawashima, S, Chun, H.-W, Aerts, J, & et al. (2010) The DBCLS BioHackathon: standardization and interoperability for bioinformatics web services and workflows. *Journal of Biomedical Semantics* 1, 1–19. 1.1.1, 1.1.3, 1.1.3, 4.1.2, 4.1.3

[35] Kahn, S. D. (2011) On the future of genomic data. *Science* 331, 728–729. 1.1.1, 1.2.1, 4.1.3, 4.1.4

[36] Loh, P.-R, Baym, M, & Berger, B. (2012) Compressive genomics. *Nature Biotechnology* 30, 627–630. 1.1.1, 1.1.1

[37] Wetterstrand, K. (2013) DNA sequencing costs: Data from the NHGRI Genome Sequencing Program (GSP). 1.2

[38] Ruffalo, M, Laframboise, T, & Koyutürk, M. (2011) Comparative analysis of algorithms for next-generation sequencing read alignment. *Bioinformatics* 27, 2790–2796. 1.1.1, 1.5, 1.3

[39] Li, H & Homer, N. (2010) A survey of sequence alignment algorithms for next-generation sequencing. *Briefings in Bioinformatics* 11, 473–483. 1.1.1, 1.5

[40] Mohammed, Y, Shahand, S, Korkhov, V, Luyf, A. C, Schaik, B. D. v, Caan, M. W, Kampen, A. H. v, Palmblad, M, & Olabarriaga, S. D. (2011) Data decomposition in biomedical e-science applications. *2011 IEEE Seventh International Conference on e-Science Workshops* 1, 158–165. 1.1.1, 4.1.1

[41] Olson, C. B, Kim, M, Clauson, C, Kogon, B, Ebeling, C, Hauck, S, & Ruzzo, W. L. (2012) Hardware acceleration of short read mapping. *2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines* p. 161–168. 1.1.1

[42] Chen, Y, Schmidt, B, & Maskell, D. (2013) A hybrid short read mapping accelerator. *BMC Bioinformatics* 14, 67. 1.1.1

[43] Gil, Y, Deelman, E, Ellisman, M, Fahringer, T, Fox, G, Gannon, D, Goble, C, Livny, M, Moreau, L, & Myers, J. (2007) Examining the challenges of scientific workflows. *Computer* 40, 24–32. 1.1.2

[44] Nekrutenko, A & Taylor, J. (2012) Next-generation sequencing data interpretation: enhancing reproducibility and accessibility. *Nature Reviews. Genetics* 13, 667–672. 1.1.2, 1.2, 1.2.1, 1.3, 4, 4.2

[45] Simmhan, Y. L, Plale, B, & Gannon, D. (2005) A survey of data provenance in e-science. *SIGMOD Rec.* 34, 31–36. 1.1.2

[46] Altunay, M, Colonnese, D, & Warade, C. (2005) High throughput web services for life sciences. *Health Care* p. 1–6. 1.1.2

[47] R Core Team. (2012) *R: A Language and Environment for Statistical Computing* (R Foundation for Statistical Computing, Vienna, Austria). 1.1.2

[48] Oinn, T, Addis, M, Ferris, J, Marvin, D, Carver, T, Pocock, M. R, & Wipat, A. (2004) Taverna: A tool for the composition and enactment of bioinformatics workflows. *Bioinformatics* 20, 3045–3054. 1.1.2

[49] Reich, M, Liefeld, T, Gould, J, Lerner, J, Tamayo, P, & Mesirov, J. P. (2006) GenePattern 2.0. *Nature Genetics* 38, 500–501. 1.1.2, 1.1.2, 1.3, 4.1.3

[50] Goecks, J, Nekrutenko, A, Taylor, J, & Team, T. G. (2010) Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology* 11, R86. 1.1.2, 1.1.2, 1.3, 4.1.1, 4.1.3

[51] Blankenberg, D, Kuster, G. V, Coraor, N, Ananda, G, Lazarus, R, Mangan, M,

Nekrutenko, A, & Taylor, J. (2010) Galaxy: A web-based genome analysis tool for experimentalists. *Current Protocols in Molecular Biology* pp. 19–10. 1.1.2

[52] Giardine, B, Riemer, C, Hardison, R, Burnharns, R, Elnitski, L, Shah, P, Zhang, Y, Blankerberg, D, Albert, I, Miller, W, Kent, J, & Nekrutenko, A. (2005) Galaxy: A platform for interactive large-scale genome analysis. *Genome Research* 15, 1451–1455. 1.1.2

[53] Kallio, M. A, Tuimala, J, Hupponen, T, Klemela, P, Gentile, M, Scheinin, I, Koski, M, Kaki, J, & Korpelainen, E. (2011) Chipster: user-friendly analysis software for microarray and other high-throughput data. *BMC Genomics* 12, 507. 1.1.2, 4.1.3

[54] Hull, D, Wolstencroft, K, Stevens, R, Goble, C, Pocock, M. R, Li, P, & Oinn, T. (2006) Taverna: a tool for building and running workflows of services. *Nucleic Acids Research* 34, W729–W732. 1.1.2, 1.3, 4.1.1, 4.1.3

[55] Ludäscher, B, Altintas, I, Berkley, C, Higgins, D, Jaeger, E, Jones, M, Lee, E, Tao, J, & Zhao, Y. (2006) Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice and Experience* 18, 1039–1065. 1.1.2, 1.1.3, 4.1.3

[56] Deelman, E, Blythe, J, Gil, Y, Kesselman, C, Mehta, G, Patil, S, Su, M.-H, Vahi, K, & Livny, M. (2004) *Pegasus: Mapping scientific workflows onto the grid*, Grid Computing. (Springer), pp. 11–20. 1.1.2

[57] Taylor, I, Shields, M, Wang, I, & Harrison, A. (2007) The Triana workflow environment: Architecture and applications. *Workflows for e-Science* p. 320–339. 1.1.2, 4.1.3

[58] Wolstencroft, K, Haines, R, Fellows, D, Williams, A, Withers, D, Owen, S, Soiland-Reyes, S, Dunlop, I, Nenadic, A, Fisher, P, & et al. (2013) The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. *Nucleic Acids Research* 41, W557–W561. 1.1.2, 1.3, 4.1.1, 4.1.3

[59] Wassermann, B, Emmerich, W, Butchart, B, Cameron, N, Chen, L, & Patel, J. (2007) Sedna: A BPEL-based environment for visual scientific workflow modeling. *Workflows for e-Science* p. 428–449. 1.1.2

[60] Emmerich, W, Butchart, B, Chen, L, Wassermann, B, & Price, S. (2005) Grid service orchestration using the business process execution language (bpel). *Journal of Grid Computing* 3, 283–304. 1.1.2

[61] Jordan, D, Evdemon, J, Alves, A, Arkin, A, Askary, S, Barreto, C, Bloch, B, Curbera, F, Ford, M, Goland, Y, et al. (2007) Web services business process execution language version 2.0. *OASIS Standard* 11. 1.1.2, 4.1.1

[62] Tan, W, Missier, P, Foster, I, Madduri, R, Roure, D. D, & Goble, C. (2010) A comparison of using taverna and bpel in building scientific workflows: the case of cagrid. *Concurrency and Compututation : Practice and Experience* 22, 1098–1117. 1.1.2

[63] Chiu, K, Govindaraju, M, & Bramley, R. (2002) Investigating the limits of soap performance for scientific computing. *Proceedings 11th IEEE International Symposium on High Performance Distributed Computing* pp. 246–254. 1.1.2, 1.1.3

[64] Engelen, R. A. V. (2003) *Pushing the SOAP Envelope with Web Services for Scientific Computing*, In the Proceedings of the International Conference on Web Services (ICWS). (IEEE, Las Vegas, NV, USA), pp. 346–352. 1.1.2, 1.1.3, 4.1.2

[65] Bhagat, J, Tanoh, F, Nzuobontane, E, Laurent, T, Orlowski, J, Roos, M, Wolstencroft, K, Aleksejevs, S, Stevens, R, Pettifer, S, Lopez, R, & Goble, C. (2010) BioCatalogue: a universal catalogue of web services for the life sciences. *Nucleic Acids Research* 38, W689–W694. 1.1.2, 1.1.3

[66] Prlić, A, Down, T. a, Kulesha, E, Finn, R. D, Kähäri, A, & Hubbard, T. J. P. (2007) Integrating sequence and structural biology with DAS. *BMC bioinformatics* 8, 333. 1.1.3

[67] Wilkinson, M. D & Links, M. (2002) BioMOBY: An open source biological web services proposal. *Briefings in Bioinformatics* 3, 331–341. 1.1.3

[68] Pettifer, S, Ison, J, Kalaš, M, Thorne, D, McDermott, P, Jonassen, I, Liaquat, A, Fernández, J. M, Rodriguez, J. M, Partners, I, & et al. (2010) The EMBRACE web service collection. *Nucleic Acids Research* 38, 683–688. 1.1.3

[69] Benson, G. (2010) Editorial. *Nucleic Acids Research* 38, W1–W2. 1.1.3

[70] Pautasso, C, Zimmermann, O, & Leymann, F. (2008) *Restful web services vs. big'web services: making the right architectural decision*, In the Proceeding of the 17th International Conference on World Wide Web. (ACM), p. 805–814. 1.1.3

[71] Davis, D & Parashar, M. (2002) *Latency Performance of SOAP Implementations*, 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid. pp. 407–407. 1.1.3

[72] Liu, D, Law, K. H, & Wiederhold, G. (2002) *Analysis of integration models for service composition*, Proceedings of the 3rd International Workshop on Software and Performance (WOSP). (ACM, New York, NY, USA), pp. 158–165. 1.1.3, 4.1.1

[73] Barker, A, Weissman, J. B, & Hemert, J. v. (2008) *Orchestrating Data-Centric Workflows*, In the Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CCGRID). (Lyon, France). 1.1.3, 4.1.1

[74] Pedraza, G & Estublier, J. (2009) in *Trustworthy Software Development Processes*, Lecture Notes in Computer Science, eds. Wang, Q, Garousi, V, Madachy, R, & Pfahl, D. (Springer Berlin Heidelberg) Vol. 5543, pp. 75–86. 1.1.3

[75] Liu, D, Law, K. H, & Wiederhold, G. (2003) *Data-flow Distribution in FICAS Service Composition Infrastructure*, Proceedings of the 15th International Conference on Parallel and Distributed Computing Systems. 1.1.3, 4.1.1

[76] Binder, W, Constantinescu, I, & Faltings, B. (2006) *Decentralized Orchestration of Composite Web Services*, In the Proceedings of the International Conference on Web Services (ICWS). (IEEE, Los Alamitos, CA, USA). 1.1.3

[77] Habich, D, Richly, S, & Grasselt, M. (2007) *Data-grey-box web services in data-centric environments*, In the Proceedings of the International Conference on Web Services (ICWS). (IEEE, Utah, USA), pp. 976–983. 1.1.3, 4.1.1

[78] Zhang, D, Coddington, P, & Wendelborn, A. (2011) Web services workflow with result data forwarding as resources. *Future Generation Computing System* 27. 1.1.3, 4.1.1

[79] Glatard, T, Montagnat, J, & Pennec, X. (2006) Efficient services composition for grid-enabled data-intensive applications. *International Symposium on High-Performance Distributed Computing* 0, 333–334. 1.1.3, 4.1.1

[80] Missier, P, Soiland-Reyes, S, Owen, S, Tan, W, Nenadic, A, Dunlop, I, Williams, A, Oinn, T, & Goble, C. (2010) *Taverna, reloaded*, Scientific and Statistical Database Management. (Springer), p. 471–481. 1.1.3, 4.1.1

[81] Abramson, D, Kommineni, J, & Altintas, I. (2005) *Flexible IO Services in the Kepler Grid Workflow System*, In the International Conference on e-Science and Grid Computing (e-Science). (Melbourne, Australia). 1.1.3

[82] Blower, J, Haines, K, & Llewellin, E. (2005) *Data streaming, workflow and firewall-friendly Grid Services with Styx.*, Proceedings of the UK e-Science All Hands Meeting. (IEEE), pp. 19–22. 1.1.3

[83] Baker, M. (2012) Functional genomics: the changes that count. *Nature* p. 4–8. 1.2

[84] Botstein, D & Risch, N. (2003) Discovering genotypes underlying human phenotypes: past successes for mendelian disease, future approaches for complex disease. *Nature genetics* 33, 228–237. 1.2

[85] Manolio, T. a, Collins, F. S, Cox, N. J, Goldstein, D. B, Hindorff, L. a, Hunter, D. J, McCarthy, M. I, Ramos, E. M, Cardon, L. R, Chakravarti, A, & et al. (2009) Finding the missing heritability of complex diseases. *Nature* 461, 747–753. 1.2

[86] Hindorff, L. a, Sethupathy, P, Junkins, H. a, Ramos, E. M, Mehta, J. P, Collins, F. S, & Manolio, T. a. (2009) Potential etiologic and functional implications of genome-wide association loci for human diseases and traits. *Proceedings of the National Academy of Sciences* 106, 9362–9367. 1.2

[87] Puente, X. S, Pinyol, M, Quesada, V, Conde, L, Ordóñez, G. R, Villamor, N, Escaramis, G, Jares, P, Beà, S, González-Díaz, M, et al. (2011) Whole-genome sequencing identifies recurrent mutations in chronic lymphocytic leukaemia. *Nature* 475, 101–105. 1.2

[88] Agrawal, N, Frederick, M, & Pickering, C. (2011) Exome sequencing of head and neck squamous cell carcinoma reveals inactivating mutations in NOTCH1. *Science* 333, 1154–1157. 1.2

[89] Yan, X.-J, Xu, J, Gu, Z.-H, Pan, C.-M, Lu, G, Shen, Y, Shi, J.-Y, Zhu, Y.-M, Tang, L, Zhang, X.-W, & et al. (2011) Exome sequencing identifies somatic mutations of dna methyltransferase gene DNMT3A in acute monocytic leukemia. *Nature Genetics* 43, 309–315. 1.2

[90] Horner, D. S, Pavesi, G, Castrignanò, T, Meo, P. D. D, Liuni, S, Sammeth, M, Picardi, E, & Pesole, G. (2010) Bioinformatics approaches for genomics and post genomics applications of next-generation sequencing. *Briefings in Bioinformatics* 11, 181–197. 1.2, 1.2.1, 1.3

[91] Ng, S & Nickerson, D. (2010) Massively parallel sequencing and rare disease. *Human Molecular Genetics* 19, 119–124. 1.2

[92] Valencia, A & Hidalgo, M. (2012) Getting personalized cancer genome analysis into the clinic: the challenges in bioinformatics. *Genome Medicine* 4, 61. 1.2, 1.2.1, 1.3, 4, 4.2

[93] Hayden, E. C. (2012) Sequencing set to alter clinical landscape. *Nature* 482, 288. 1.2, 4

[94] Ku, C.-S, Cooper, D. N, Polychronakos, C, Naidoo, N, Wu, M, & Soong, R. (2012) Exome sequencing: dual role as a discovery and diagnostic tool. *Annals of Neurology* 71, 5–14. 1.2, 4, 4.2

[95] Lyon, G. J & Wang, K. (2012) Identifying disease mutations in genomic medicine settings: current challenges and how to accelerate progress. *Genome Medicine* 4, 58. 1.2, 1.2.1, 1.3, 4, 4.2, 4.2.1

[96] McPherson, J. (2009) Next-generation gap. *Nature Methods* 6, S2–S5. 1.2, 4

[97] Nielsen, R, Paul, J. S, Albrechtsen, A, & Song, Y. S. (2011) Genotype and SNP calling from next-generation sequencing data. *Nature reviews. Genetics* 12, 443–451. 1.2.1, 1.5, 4.2.1

[98] Asan, Xu, Y, Jiang, H, Tyler-Smith, C, Xue, Y, Jiang, T, Wang, J, Wu, M, Liu, X, Tian, G, Wang, J, Wang, J, Yang, H, & Zhang, X. (2011) Comprehensive comparison of three commercial human whole-exome capture platforms. *Genome Biology* 12, R95. 1.2.1, 4.2

[99] Auwera, G. A, Carneiro, M. O, Hartl, C, Poplin, R, del Angel, G, Levy-Moonshine, A, Jordan, T, Shakir, K, Roazen, D, Thibault, J, et al. (2013) From fastq data to high-confidence variant calls: The Genome Analysis Toolkit best practices pipeline. *Current Protocols in Bioinformatics* pp. 11–10. 1.5

[100] McKenna, A, Hanna, M, Banks, E, Sivachenko, A, Cibulskis, K, Kernytsky, A, Garimella, K, Altshuler, D, Gabriel, S, Daly, M, & et al. (2010) The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Research* 20, 1297–1303. 1.2.1, 1.5

[101] Homer, N, Merriman, B, & Nelson, S. F. (2009) BFAST: an alignment tool for large scale genome resequencing. *PloS One* 4, e7767. 1.2.1, 1.5

[102] Langmead, B, Trapnell, C, Pop, M, & Salzberg, S. L. (2009) Ultrafast and

memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology* 10, R25. 1.5

[103] Li, R, Yu, C, Li, Y, Lam, T.-W, Yiu, S.-M, Kristiansen, K, & Wang, J. (2009) SOAP2: an improved ultrafast tool for short read alignment. *Bioinformatics* 25, 1966–7. 1.5

[104] Rumble, S. M, Lacroute, P, Dalca, A. V, Fiume, M, Sidow, A, & Brudno, M. (2009) SHRiMP: accurate mapping of short color-space reads. *PLoS Computational Biology* 5, e1000386. 1.5

[105] DePristo, M. a, Banks, E, Poplin, R, Garimella, K. V, Maguire, J. R, Hartl, C, Philippakis, A. a, Angel, G. d, Rivas, M. a, Hanna, M, & et al. (2011) A framework for variation discovery and genotyping using next-generation DNA sequencing data. *Nature Genetics* 43, 491–8. 1.2.1, 1.5, 4.2.2

[106] Li, R, Li, Y, Fang, X, Yang, H, Wang, J, Kristiansen, K, & Wang, J. (2009) SNP detection for massively parallel whole-genome resequencing. *Genome Research* 19, 1124–32. 1.5

[107] The International Hapmap Consortium. (2005) A haplotype map of the human genome. *Nature* 437, 1299–320. 1.5

[108] Sherry, S. T, Ward, M. H, Kholodov, M, Baker, J, Phan, L, Smigielski, E. M, & Sirotkin, K. (2001) dbSNP: the NCBI database of genetic variation. *Nucleic Acids Research* 29, 308–311. 1.5

[109] Sugden, L. a, Tackett, M. R, Savva, Y. a, Thompson, W. a, & Lawrence, C. E. (2013) Assessing the validity and reproducibility of genome scale predictions. *Bioinformatics* 29, 2844–2851. 1.3

[110] O'Rawe, J, Jiang, T, Sun, G, Wu, Y, Wang, W, Hu, J, Bodily, P, Tian, L, Hakonarson, H, Johnson, W. E, & et al. (2013) Low concordance of multiple variant-calling pipelines: practical implications for exome and genome sequencing. *Genome medicine* 5, 28. 1.3, 4.2, 4.2.1

[111] Frith, M, Hamada, M, & Horton, P. (2010) Parameters for accurate genome alignment. *BMC Bioinformatics* 11, 80. 1.3

[112] Rapaport, F, Khanin, R, Liang, Y, Pirun, M, Krek, A, Zumbo, P, Mason, C. E, Socci, N. D, & Betel, D. (2013) Comprehensive evaluation of differential gene expression analysis methods for RNA-seq data. *Genome Biology* 14, R95. 1.3

[113] Lunter, G & Goodson, M. (2011) Stampy: a statistical algorithm for sensitive and fast mapping of illumina sequence reads. *Genome Research* 21, 936–939. 1.3

[114] Farrer, R. a, Henk, D. a, MacLean, D, Studholme, D. J, & Fisher, M. C. (2013) Using false discovery rates to benchmark SNP-callers in next-generation sequencing projects. *Scientific Reports* 3, 1512. 1.3, 4.2

[115] Heyden, Y. V, Nijhuis, a, Smeyers-Verbeke, J, Vandeginste, B. G, & Massart,

D. L. (2001) Guidance for robustness/ruggedness tests in method validation. *Journal of Pharmaceutical and Biomedical Analysis* 24, 723–753. 1.3.1

[116] Saltelli, A, Ratto, M, Andres, T, Campolongo, F, Cariboni, J, Gatelli, D, Saisana, M, & Tarantola, S. (2008) *Global sensitivity analysis: the primer.* (Wiley. com). 1.3.1

[117] Sobol', I. (1976) Uniformly distributed sequences with an additional uniform property. *USSR Computational Mathematics and Mathematical Physics* p. 1332–1337. 1.3.1

[118] McKay, M, Beckman, R, & Conover, W. (1979) Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* 21, 239–245. 1.3.1

[119] Cukier, R. I, Fortuin, C. M, Shuler, K. E, Petschek, A. G, & Schaibly, J. H. (1973) Study of the sensitivity of coupled reaction systems to uncertainties in rate coefficients. i theory. *The Journal of Chemical Physics* 59, 3873. 1.3.1

[120] Cukier, R, Levine, H, & Shuler, K. (1978) Nonlinear sensitivity analysis of multiparameter model systems. *Journal of Computational Physics* 42, 1–42. 1.3.1

[121] Sobol', I. (2001) Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Mathematics and Computers in Simulation* 55, 271–280. 1.3.1

[122] Campolongo, F, Tarantola, S, & Saltelli, a. (1999) Tackling quantitatively large dimensionality problems. *Computer Physics Communications* 117, 75–85. 1.3.1, 1.3.2, 4.2.2, 4.2.3

[123] Rabitz, H, Aliş, O, Shorter, J, & Shim, K. (1999) Efficient input—output model representations. *Computer Physics Communications* 117, 11–20. 1.3.1

[124] Campolongo, F & Saltelli, A. (1997) Sensitivity analysis of an environmental model: an application of different analysis methods. *Reliability Engineering & System Safety* 57, 49–69. 1.3.1, 4.2.2

[125] Morris, M. D. (1991) Factorial sampling plans for preliminary computational experiments. *Technometrics* 33, 161–174. 1.3.1, 1.3.2, 1.3.2, 1.3.2

[126] Campolongo, F, Cariboni, J, & Saltelli, A. (2007) An effective screening design for sensitivity analysis of large models. *Environmental Modelling and Software* 22, 1509–1518. 1.3.1, 1.3.2, 1.3.2, 1.3.2

[127] Subramanian, S, Puntervoll, P, & Sztromwasser, P. (2010) *Optimizing the Data-traffic of Centrally Coordinated Scientific Workflow Systems*, In the Proceedings of the International Conference on Web Services (ICWS). (Miami, FL, USA). 3.1.2

[128] Kamiya, T & Schneider, J. (2011) Efficient xml interchange (exi) format 1.0. *World Wide Web Consortium Recommendation REC-exi-20110310.* 4.1.2

[129] Peintner, D, Kosch, H, & Heuer, J. (2009) *Efficient XML Interchange for rich internet applications*, IEEE International Conference on Multimedia and Expo (ICME). (IEEE), pp. 149–152. 4.1.2

[130] Milanovic, N & Malek, M. (2004) Current solutions for web service composition. *Internet Computing* 8, 51–59. 4.1.3

[131] Krampis, K, Booth, T, Chapman, B, Tiwari, B, Bicak, M, Field, D, & Nelson, K. E. (2012) Cloud BioLinux: pre-configured and on-demand bioinformatics computing for the genomics community. *BMC Bioinformatics* 13, 42. 4.1.3, 4.1.4

[132] Wang, R, Brewer, D, Shastri, S, Swayampakula, S, Miller, J, Kraemer, E, & Kissinger, J. (2009) *Adapting the Galaxy Bioinformatics Tool to Support Semantic Web Service Composition*, 2009 World Conference on Services-I. (IEEE), pp. 283–290. 4.1.3

[133] Troshin, P. V, Procter, J. B, & Barton, G. J. (2011) Java bioinformatics analysis web services for multiple sequence alignment–JABAWS:MSA. *Bioinformatics* 27, 2001–2. 4.1.4

[134] Afgan, E, Chapman, B, Jadan, M, Franke, V, & Taylor, J. (2012) Using cloud computing infrastructure with CloudBioLinux, CloudMan, and Galaxy. *Current Protocols in Bioinformatics* pp. 11–19. 4.1.4

[135] The 1000 Genomes Consortium. (2010) A map of human genome variation from population-scale sequencing. *Nature* 467, 1061–1073. 4.1.4, 4.2.2

[136] Crosswell, L. C & Thornton, J. M. (2012) ELIXIR: a distributed infrastructure for European biological data. *Trends in Biotechnology* 30, 241 – 242. 4.1.4

[137] Gupta, I. R, Baldwin, C, Auguste, D, Ha, K. C. H, Andalousi, J. E, Fahiminiya, S, Bitzan, M, Bernard, C, Akbari, M. R, Narod, S. a, & et al. (2013) ARHGDIA: a novel gene implicated in nephrotic syndrome. *Journal of Medical Genetics* 50, 330–338. 4.2

[138] Samuels, M. E, Majewski, J, Alirezaie, N, Fernandez, I, Casals, F, Patey, N, Decaluwe, H, Gosselin, I, Haddad, E, Hodgkinson, A, & et al. (2013) Exome sequencing identifies mutations in the gene TTC7A in French-Canadian cases with hereditary multiple intestinal atresia. *Journal of Medical Genetics* 50, 324–329. 4.2

[139] Bilguvar, K, Tyagi, N. K, Ozkara, C, Tuysuz, B, Bakircioglu, M, Choi, M, Delil, S, Caglayan, A. O, Baranoski, J. F, Erturk, O, & et al. (2013) Recessive loss of function of the neuronal ubiquitin hydrolase UCHL1 leads to early-onset progressive neurodegeneration. *Proceedings of the National Academy of Sciences* 110, 3489–3494. 4.2

[140] Lemke, J. R, Riesch, E, Scheurenbrand, T, Schubach, M, Wilhelm, C, Steiner, I, Hansen, J, Courage, C, Gallati, S, Bürki, S, & et al. (2012) Targeted next generation sequencing as a diagnostic tool in epileptic disorders. *Epilepsia* 53, 1387–1398. 4.2

[141] Choi, M, Scholl, U. I, Ji, W, Liu, T, Tikhonova, I. R, Zumbo, P, Nayir, A, Bakkaloğlu, A, Özen, S, Sanjad, S, & et al. (2009) Genetic diagnosis by whole exome capture and massively parallel DNA sequencing. *Proceedings of the National Academy of Sciences* 106, 19096–19101. 4.2

[142] Melum, E, May, S, Schilhabel, M. B, Thomsen, I, Karlsen, T. H, Rosenstiel, P, Schreiber, S, & Franke, A. (2010) SNP discovery performance of two second-generation sequencing platforms in the NOD2 gene region. *Human Mutation* 31, 875–885. 4.2

[143] Hedges, D, Guettouche, T, Yang, S, & Bademci, G. (2011) Comparison of three targeted enrichment strategies on the SOLiD sequencing platform. *PLoS One* 6, e18595. 4.2

[144] Meynert, A. M, Bicknell, L. S, Hurles, M. E, Jackson, A. P, & Taylor, M. S. (2013) Quantifying single nucleotide variant detection sensitivity in exome sequencing. *BMC Bioinformatics* 14, 195. 4.2.2

[145] Heinrich, V, Kamphans, T, Stange, J, Parkhomchuk, D, Hecht, J, Dickhaus, T, Robinson, P. N, & Krawitz, P. M. (2013) Estimating exome genotyping accuracy by comparing to data from large scale sequencing projects. *Genome Medicine* 5, 69. 4.2.2