UiT
THE ARCTIC
UNIVERSITY
OF NORWAY

Faculty of Science and Technology
Department of Computer Science

Toward Reproducible Analysis and Exploration of High-Throughput Biological Datasets

—

Bjørn Fjukstad

A dissertation for the degree of Philosophiae Doctor – 2018

“Ta aldri problemene på forskudd, for da får du dem to ganger, men ta gjerne seieren på forskudd, for hvis ikke er det altfor sjelden du får oppleve den.”

–Ivar Tollefsen

Abstract

There is a rapid growth in the number of available biological datasets due to the advent of high-throughput data collection instruments combined with cheap compute infrastructure. Modern instruments enable the analysis of biological data at different levels, from small DNA sequences through larger cell structures, and up to the function of entire organs. These new datasets have brought the need to develop new software tools and packages to enable novel insights into the underlying biological mechanisms in the development and progression of diseases such as cancer.

The heterogeneity of biological datasets require researchers to tailor the exploration and analyses with a range of different tools and systems. However, despite the need for their integration, few of them provide standard interfaces for analyses implemented using different programming languages and frameworks. In addition, because of the many tools, different input parameters, and references to databases, it is necessary to record these correctly. The lack of such details complicates reproducing the original results and the reuse of the analyses on new datasets. This increases the analysis time and leaves unrealized potential for scientific insights.

This dissertation argues that we can develop unified systems for reproducible exploration and analysis of high-throughput biological datasets. We propose an approach, Small Modular Entities (SMEs), for developing data analysis pipelines and data exploration applications in cancer research. We realize SMEs using software container technologies together with well-defined interfaces, configuration, and orchestration. It simplifies developing such applications, and provides detailed information needed to reproduce the analyses.

Through this approach we have developed different applications for analyzing high-throughput DNA sequencing datasets, and for exploring gene expression data integrated with questionnaires, registry, and online databases. The evaluation shows how we effectively capture provenance in analysis pipelines and data exploration applications. Our approach simplifies the sharing of methods, data, tools, and applications, all fundamental to enable reproducible science.

Acknowledgements

First I would like to thank my advisor, Professor Lars Ailo Bongo for his relentless support and encouragement during my time as a PhD student. He has indeed shown me what tough love is, and I am grateful for that.

I would like to thank my co-advisors Professor Eiliv Lund and Associate Professor Karina Standahl Olsen for their wonderful ideas and warm welcome into a research field that was not my own.

I would like to extend my gratitude to Professor Michael Hallett and Vanessa Dumeaux for their hospitality when I visited their lab in Montreal in 2016. I do not think this thesis would have been as interesting without the projects I was fortunate enough to be a part of. Thank you!

I would like to thank my long time office wife Einar, Morten, Nina, and the BDPS lab at UiT.

Thank you to past or current students at UiT: Jan-Ove, Vegard, Helge, Magnus, Erlend, Kristian, Martin, Amund, Michael, and many more. You have all contributed to nine wonderful years at the University!

I would like to thank my colleagues at the Department of Computer Science, especially the technical staff, led by Maria Wulff Hauglann.

Thank you to everyone in the NOWAC research group, you have all been wonderful to collaborate with!

Thank you to the PhD students at Nordlandssykehuset in Bodø who have been my closest colleagues during the final push of my PhD.

I would like to thank my mom and dad, and my younger brother for their ever-present support.

Finally, Ane for her continuous love and support, and her *endurance* through all of my big or small projects.

Contents

List of Figures

List of Tables

List of Abbreviations

API Application Programming Interface

CLI Command-line Interface

CRAN Comprehensive R Archive Network

CSV comma-separated values

CWL Common Workflow Language

DAG directed acyclic graph

DNA Deoxyribonucleic acid

GATK Genome Analysis Toolkit

GB Gigabyte

GPU graphical processing unit

GUI Graphical User Interface

HPC high-performance computing

HTS High-throughput Sequencing

IDE integrated development environment

JSON JavaScript Object Notation

KEGG Kyoto Encyclopedia of Genes and Genomes

KGML KEGG Markup Language

MIxT Matched Interactions Across Tissues

NGS Next-generation Sequencing

NOWAC Norwegian Women and Cancer

PFS Pachyderm File System

PPS Pachyderm Processing System

REST Representational state transfer

RNA Ribonucleic acid

SCM source code management

SME Small Modular Entity

SNP Single Nucleotide Polymorphism

SR Systemic Response

VM Virtual Machine

WES whole-exome sequencing

WGCNA Weighted Gene Co-expression Network Analysis

WGS whole-genome sequencing

XML Extensible Markup Language

YAML YAML Ain't Markup Language

1

Introduction

There is a rapid growth in the number of available biological datasets due to the decreasing costs of data collection. This brings opportunities for gaining novel insights into the underlying biological mechanisms in the development and progression of diseases such as cancer, possibly leading to the development of new diagnostic tests or drugs for treatment. The wide range of different biological datasets has led to the development of hundreds of software packages and systems to explore and analyze these datasets. However, there are few systems that are designed with the full analysis process in mind, from raw data into interpretable and reproducible results. While existing systems are used to provide novel insights in diseases, there is little emphasis on reporting and sharing detailed information about the analyses. This leads to unnecessary difficulties when reusing known methods, and reproducing the analyses, which in turn leads to a longer analysis process and therefore unrealized potential for scientific insights. For clinicians, inaccurate results from improperly developed analyses can lead to negative consequences for patient care.[1]

We have identified four main challenges for application developers to undertake when building systems for analyzing and exploring biological datasets in research and the clinic. These challenges are common for large datasets such as high-throughput sequencing data that require long-running, deep analysis pipelines, as well as smaller datasets, such as microarray data, that require complex, but short-running analysis pipelines. The first challenge is managing datasets and analysis code for use by data exploration applications and data analysis pipelines. This includes storing all information that is necessary to

a data analyst when he or she is interpreting the data, as well as any analysis code that can be used to analyze the data. The second challenge is to develop data exploration applications that provide sufficient information to fully document every step that went into the analyses up to an end result. This includes reporting input parameters, tool versions, database versions, and dataset versions. The third challenge is developing applications that require the integration of disparate systems. These are often developed using different programming languages and provide different functionality, e.g., the combination of a web-based visualization with a graphical processing unit (GPU) accelerated statistical method, or the integration of a remote biological database. The final challenge is to develop applications and systems so that they can be easily shared and reused across research institutions.

As a result, there is a wealth of specialized approaches and systems to manage and analyze modern biological data. Systems such as Galaxy[2] provide simple Graphical User Interfaces (GUIs) for setting up and running analysis pipelines. However, it is difficult to install and maintain, and less flexible for explorative analyses where it is necessary to try out new tools and different tool configurations.[3] With R and its popular package repository Bioconductor[4], researchers can select from a wide range of packages to tailor their analyses. These provide specialized analysis environments, but makes it necessary for the analyst to manually record information about data, tools, and tool versions. Systems such as Pachyderm[5] or the Common Workflow Language (CWL)[6] and its different implementations, can help users with standardizing the description and sharing of analysis pipelines. However, many of these require complex compute infrastructure and are too cumbersome to set up for institutions without dedicated technical staff. Shiny[7] and OpenCPU[8] provide frameworks for application developers to build systems to interactively explore results from statistical analyses. These are useful for building exploration applications that integrate with statistical analyses. With the addition of new datasets and methods every year, it seems that analysis of biological data requires a wide array of different tools and systems.

This dissertation argues that, instead, we can facilitate the development of reproducible data analysis and exploration systems for high-throughput biological data, through the integration of disparate systems and data sources. In particular, we show how software container technologies together with well-defined interfaces, configurations, and orchestration provide the necessary foundation for these systems. This allows for easy development and sharing of specialized analysis systems.

The resulting approach, which we have called Small Modular Entities (SMEs), argues that applications for analyzing and exploring biological datasets should be modeled as a composition of individual systems and tools. We believe that the

Unix philosophy to "*Do one thing and do it well*"[9] appropriately summarizes many existing tools in bioinformatics, and we should aim to build applications as compositions of these tools. Our SME approach resembles the traditional Unix-like pipelines, in combination with the service-oriented architecture[10] or the microservice architectural style now popularized by web-scale distributed systems.[11]

The approach has several key advantages when implementing systems to analyze and explore biological data:

- It enables and simplifies the development of applications that integrate disparate tools.
- It enables reproducible research by packaging applications and tools within containerized environments.
- With well-defined interfaces it is a simple task to add new components to a system, or modify existing ones.
- Through software container technology it becomes a simple task to deploy and scale up such applications.

In collaboration with researchers in systems epidemiology and precision medicine we developed a set of applications and systems necessary to organize, analyze, and interpret their datasets. From these systems we extraplolated a set of general design principles to form a unified approach. We evaluate this approach through these systems using real datasets to show its viability.

From a longer-term perspective we discuss the general patterns for implementing reproducible data analysis systems for use in biomedical research. As more datasets are produced every year, research will depend on the simplicity of the systems for analyzing these, and that they provide the necessary functionality to reproduce and share the analysis pipelines.

Thesis statement: A unified development model based on software container infrastructure can efficiently provide reproducible and easy to use environments to develop applications for exploring and analyzing biological datasets.

1.1 Problems with Data Analysis and Exploration in Bioinformatics

High-throughput technologies for cheaper and faster data generation, as well as simpler access to the datasets have revolutionized biology.[12, 13] While these datasets can reveal the genetic basis of disease in patients, they require the collaborative efforts of experts from different fields to design and perform the analyses, and to to interpret the results.[14] Since the inferences are only as good as the information they are based on, researchers have to constantly ensure the quality of the underlying data and analyses.[15]

Today shell scripts are often used for building analysis pipelines in bioinformatics. This comes from the familiarity of the shell environment and the Command-line Interface (CLI) of the different tools. However, there is a move towards using more sophisticated approaches for analyzing biological datasets using workflow and pipeline mangers such as Snakemake[16], and the different implementations of the CWL[6] such as Galaxy[2] and Toil[17]. These simplify setting up and executing the analysis pipeline. However, these tools still have their limitations, such as maintenance and tool updates. Other programming environments and scripting languages such as Python or R both provide a wide variety of software packages to read and process biological datasets. Especially the package repository Bioconductor[4] provides a long list of well-maintained software packages. Both these languages require the researchers to set up their own analyses, but can be tailored to fit their data precisely. For visually exploring biological data there are a range of tools, such as Cytoscape[18] and Circos[19], that support importing an already-analyzed dataset to visualize and browse the data. One problem with these are that they are decoupled from the analysis, making it difficult to retrace the underlying analyses.

Although there are efforts to develop tools to help researchers explore and analyze biological datasets, they current tools have several drawbacks:

1. **Standardization:** Because of the specialized nature of each data analysis tool, a full workflow for exploring or analyze biological data will have to combine multiple tools. The tools provide different interfaces and processing data often require data wrangling between the tools.

2. **Decoupling:** Data exploration tools are often decoupled from the statistical analyses. This often makes it a difficult to document and retrace the analyses through the full workflow.

3. **Complexity:** Analyses that start as a simple script quickly become more complex to maintain and develop as developers add new functionality

to the analyses.

4. **Reusability:** Data exploration tools are often developed as a single specialized application, making it difficult to reuse parts of the application for other analyses or datasets. This leads to duplicate development effort and abandoned projects.

5. **Reproducibility:** While there are tools for analyzing most data types today, these require the analyst to manually record versions, input parameters, and reference databases. This makes analysis results difficult to reproduce because of the large number of variables that may impact the results.

Because of these drawbacks, a approach for unifying reproducible data analysis and exploration systems would reduce the time-to-interpretation of biological datasets significantly.

1.2 Small Modular Entities (SMEs)

In collaboration with researchers in systems epidemiology and biology we have developed an approach for designing applications for three specific use cases. The first is to manage and standardize the analysis of datasets from a large population-based cohort, NOWAC.[20]. The second is to enable interactive exploration of these datasets. The final use case is to develop pipelines for analyzing sequencing datasets for use in a precision medicine setting. Although these use cases require widely different systems with different requirements, the applications share common design patterns. Figure 1.1 shows the applications we have developed and the underlying systems.

We discuss how the approach is suitable for different use cases before highlighting why it is suitable for all of them. Figure 1.2 shows the three different use cases and one such SME. We can use it in data exploration applications, analysis pipelines, and for building data management systems.

1.2.1 Data Management and Analysis

Modern epidemiological studies integrate traditional questionnaire data with information from public registries and biological datasets. These often span multiple biological levels, i.e., different data types and collection sites. While traditional survey based datasets require few specialized analysis tools because of the relatively simple nature of the data, biological datasets require specialized

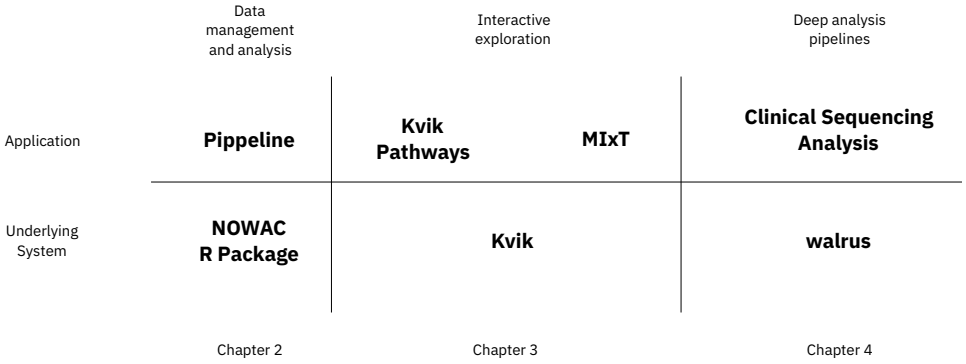


Figure 1.1: The applications and their underlying systems discussed in this thesis.

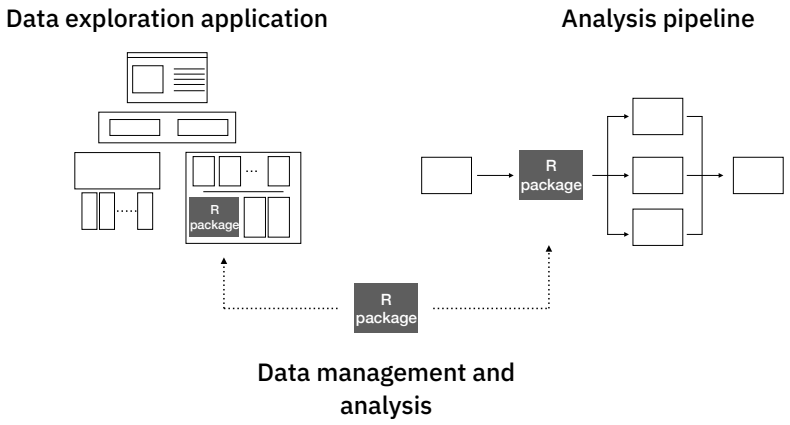


Figure 1.2: An illustration of how we envision the SME approach in data management systems, data exploration applications and analysis pipelines. In this example we reuse an R package for all use cases.

tools for reading, analyzing, and interpreting the data. Package repositories such as Bioconductor[4] provide a wealth of packages for analyzing these datasets. These packages typically provide analysis tools, example data, and comprehensive documentation. While the analysis code can be shared within projects, the datasets are often stored in in-house databases or shared file systems with specialized permissions. Together the packages and datasets form building blocks that researchers can develop their analyses on top of. They can compose their analyses using packages that fit their specific needs. The analysis code in the NOWAC study may constitute such a building block. Therefore, we combined the datasets from the NOWAC cohort with documentation, analysis scripts, and integration with registry datasets, into a single package. This approach simplifies the researcher's first steps in the analysis of the different data in our study. On top of the NOWAC package we then implemented a user-friendly preprocessing pipelining tool named Pippeline.

Inspired by the ecosystem of packages in the R programming language we implemented our approach as the NOWAC R package. Users simply install the package and get access to documentation, datasets, and utility functions for analyzing datasets related to their area of research. We use version control for both code and the data, making it possible to track changes over time as the research study evolves. Pippeline is a web-based interface for running the standardized preprocessing steps before analyzing gene expression datasets in the NOWAC cohort.

1.2.2 Interactive Data Exploration Applications

The final results from an analysis pipeline require researchers to investigate and evaluate the final output. In addition, it may be useful to explore the analysis parameters and re-run parts of the analyses. As with analysis pipelines, there are complete exploration tools as well as software libraries to develop custom applications for exploration of analysis results. The tools often require users to import already analyzed datasets but provide interactive visualizations and point-and-click interfaces to explore the data. Users with programming knowledge can use the wealth of software packages for visualization within languages such as R or Python. Frameworks such as BioJS[21] now provide developers with tools to develop web applications for exploring biological datasets. It is apparent that these types of systems also consist of multiple smaller components that together can be orchestrated into a single application. These applications typically include of three major parts: (i) data visualization; (ii) integration with statistical analyses and datasets; and (iii) integration with online databases. While each of these are specialized for each type of data exploration application, they share components that can be reused across different types of applications.

To facilitate the integration with statistical analyses and datasets, we wrote an interface to the R programming language, that would allow us to interface with the wealth of existing software packages, e.g., the NOWAC package, for biological data analyses from a point-and-click application. New data exploration applications could access analyses directly through this interface, removing the previous decoupling between the two. We followed the same approach to integrate with online databases. We could standardize the interface from the applications to the different databases, and implement an application on top of these.

We implemented all components as a part of *Kvik*, a collection of packages to develop new data exploration applications.[22] Kvik allows applications written in any modern programming language to interface with the wealth of bioinformatics packages in the R programming language, as well as information available through online databases. To provide reproducible execution environments we packaged these interfaces into software containers that can be easily deployed and shared. We have used Kvik to develop the MIXT system[23] for exploring and comparing transcriptional profiles from blood and tumor samples in breast cancer patients, in addition to applications for exploring biological pathways[22].

1.2.3 Deep Analysis Pipelines

Analysis of high-throughput sequencing datasets requires deep analysis pipelines with many steps that transform raw data into interpretable results.[24] There are many tools available that perform the different processing steps, written in a wide range of programming languages. The tools and their dependencies, can be difficult to install, and they require users to correctly manage a range of input parameters that affects the output results. With software container technology it is a simple task for developers to share container images with analysis tools pre-installed. Then, by designing a text-based specification for the analyses, we can orchestrate the execution of an entire analysis pipeline and record the flow of data through the pipeline. As with the previous use case, we develop an analysis pipeline by composing smaller entities, or tools, into a complete pipeline.

We implemented the approach in `walrus`, a tool that lets users create and run analysis pipelines. In addition, it tracks full provenance of the input, intermediate, and output data, as well as tool parameters. With `walrus` we have successfully built analysis pipelines to detect somatic mutations in breast cancer patients, as well as an Ribonucleic acid (RNA)-seq pipeline for comparison with gene expression datasets. `walrus` has also been successfully used to analyze DNA methylation and microRNA datasets.

1.2.4 Similarity

The above approaches for building data analysis and exploration applications share the same design principles. In all areas we decompose the system, into small modular entities, and package these into software containers which are then orchestrated together. These containers are configured and communicate using open protocols that make it possible to interface with them using any programming language. We track the configuration of the containers and their orchestration using software versioning systems, and provide the necessary information to set up the system and reproduce their results. We believe that the SME approach is applicable to every step in the long process from raw data collection to interpretable results, and that it makes this process more transparent.

1.3 Applications Developed with SMEs

In this section we outline the different systems we have built using SMEs. We detail how we implemented SME in the NOWAC package, `walrus`, and Kvik, and show applications that use these.

1.3.1 Data Management and Analysis

To standardize the preprocessing of biological datasets in the NOWAC study. With the NOWAC package we could implement a preprocessing pipeline on top of it that used its datasets and utility functions to generate analysis-ready datasets for the researchers. This preprocessing pipeline called Pippeline was developed as a web application which allows the data managers in our study to generate datasets for researchers. The pipeline performs all necessary steps before researchers can perform their specialized analyses.

1.3.2 Interactive Data Exploration Applications

The first interactive data exploration application that we built was Kvik Pathways. It allows users to explore gene expression data from the NOWAC cohort in the context of interactive pathway maps.[22] It is a web application that integrates with the R programming language to provide an interface to the statistical analyses. We used Kvik Pathways to repeat the analyses in a previous published project that compared gene expression in blood from healthy women with high and low plasma ratios of essential fatty acids.[25]

From the first application it became apparent that we could reuse parts of the application in the implementation of later systems. In particular, the interface to run analyses as well as the integration with the online databases could be implemented as services, packaged into containers, and reused in the next application that we developed. Both of these were designed and implemented in Kvik, which could then be used and shared later.

The second application that we built was the MIXT web application. A system to explore and compare transcriptional profiles from blood and tumor samples in breast cancer patients. The application is built to simplify the exploration of results from the Matched Interactions Across Tissues (MIxT) study. Its goal was to identify genes and pathways in the primary breast tumor that are tightly linked to genes and pathways in the patient blood cells.[26] The web application interfaces with the methods implemented as an R package and integrates the results together with information from biological databases through a simple user interface.

A third application that we developed was a simple re-deployment of the MIXT web application with a new dataset. In this application that we simply replaced the R package with a new package that interfaced with different data. All the other components are reused. It demonstrates the flexibility of the approach.

1.3.3 Deep Analysis Pipelines

The first system that we built on top of `walrus` was a pipeline to analyze a patient's primary tumor and adjacent normal tissue, including subsequent metastatic lesions.[27] We packaged the necessary tools for the analyses into software containers and wrote a pipeline description with all the necessary data processing steps. Some steps required us to develop specialized scripts to generate customized plots, but these were also wrapped in a container. From the analyses we discovered, among other findings, inherited germline mutations that are recognized to be among the top 50 mutations associated with an increased risk of familial breast cancer. These were then shared with the treating oncologists to aid the treatment plan.

The second analysis pipeline we implemented was to enable comparison of a RNA-seq dataset to microarray gene expression values collected from the same samples. The pipeline preprocesses the RNA dataset for all samples, and generates transcript quantifications. Like the first pipeline, we used existing tools together with specialized analysis scripts packaged into a container to ensure that we could reproduce the execution environments.

Combined these systems and applications demonstrate how small modular entities are useful for both batch processing of datasets and interactive applications.

1.4 Summary of Results

We show the viability of our approach through real-world applications in systems epidemiology and precision medicine. Through our `nowac` package and Pippeline, we demonstrate its usefulness for enabling reproducible analyses of biological datasets in a complex epidemiological study. We demonstrate its usefulness for building interactive data exploration application, implemented in *Kvik*. We show the applicability of small modular entities in deep analysis pipelines, as implemented in `walrus`.

We have used `walrus` to analyze a whole-exome dataset to from a sample in the McGill Genome Quebec [MGGQ] dataset (GSE58644)[28] to discover Single Nucleotide Polymorphisms (SNPs), genomic variants and somatic mutations. Using `walrus` to analyze a dataset added 10% to the runtime and doubled the space requirements, but reduced days of compute time down to seconds when restoring a previous pipeline configuration.

We have used the packages in Kvik to develop a web application, MIxT blood-tumor, for exploring and comparing transcriptional profiles from blood and tumor samples in breast cancer patients. In addition, we have used it to build an application to explore gene expression data in the context of biological pathways. We show that developing an application using a microservice approach allows us to reduce database query times down to 90%, and that we can provide an interface to statistical analyses that is up to 10 times as fast as alternative approaches.

Together the results show that our approach, small modular entities, can be used to enable reproducible data analysis and exploration of high-throughput biological datasets while still providing the required performance.

1.5 List of papers

This section contains the list of papers along with short descriptions and my contributions to each paper.

Paper 1

Title	Kvik: three-tier data exploration tools for flexible analysis of genomic data in epidemiological studies
Authors	**Bjørn Fjukstad**, Karina Standahl Olsen, Mie Jareid, Eiliv Lund, and Lars Ailo Bongo
Description	The initial description of Kvik, and how we used it to implement Kvik Pathways, a web application for browsing biologicap pathway maps integrated with gene expression data from the NOWAC cohort.
Contribution	I designed, implemented, and deployed Kvik and Kvik Pathways. Evaluated the system and wrote the manuscript.
Publication date	15 March 2015
Publication venue	F1000
Citation	[22] B. Fjukstad, K. S. Olsen, M. Jareid, E. Lund, and L. A. Bongo, "Kvik: three-tier data exploration tools for flexible analysis of genomic data in epidemiological studies," *F1000Research*, vol. 4, 2015

Paper 2

Title	Building Applications For Interactive Data Exploration In Systems Biology.
Authors	**Bjørn Fjukstad**, Vanessa Dumeaux, Karina Standahl Olsen, Michael Hallett, Eiliv Lund, and Lars Ailo Bongo.
Description	Describes how we further developed the ideas from Paper 1 into an approach that we used to build the MIXT web application.
Contribution	I designed, implemented, and deployed Kvik and the MIXT web application. Evaluated the system and wrote the manuscript.
Publication date	20 August 2017.
Publication venue	The 8th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics (ACM BCB) August 20–23, 2017.
Citation	[23] B. Fjukstad, V. Dumeaux, K. S. Olsen, E. Lund, M. Hallett, and L. A. Bongo, "Building applications for interactive data exploration in systems biology," in *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*. ACM, 2017, pp. 556–561

Paper 3

Title	Interactions Between the Tumor and the Blood Systemic Response of Breast Cancer Patients
Authors	Vanessa Dumeaux, **Bjørn Fjukstad**, Hans E Fjosne, Jan-Ole Frantzen, Marit Muri Holmen, Enno Rodegerdts, Ellen Schlichting, Anne-Lise Børresen-Dale, Lars Ailo Bongo, Eiliv Lund, Michael Hallett.
Description	Describes the MIXT system which enables identification of genes and pathways in the primary tumor that are tightly linked to genes and pathways in the patient Systemic Response (SR).
Contribution	I designed, implemented, and deployed the MIXT web application. Contributed to the writing of the manuscript.
Publication date	28 September 2017.
Publication venue	PLoS Computational Biology
Citation	[26] V. Dumeaux, B. Fjukstad, H. E. Fjosne, J.-O. Frantzen, M. M. Holmen, E. Rodegerdts, E. Schlichting, A.-L. Børresen-Dale, L. A. Bongo, E. Lund *et al.*, "Interactions between the tumor and the blood systemic response of breast cancer patients," *PLoS Computational Biology*, vol. 13, no. 9, p. e1005680, 2017

Paper 4

Title	A Review of Scalable Bioinformatics Pipelines
Authors	**Bjørn Fjukstad**, Lars Ailo Bongo.
Description	This review survey several scalable bioinformatics pipelines and compare their design and their use of underlying frameworks and infrastructures.
Contribution	I performed the literature review and wrote the manuscript.
Publication date	23 October 2017
Publication venue	Data Science and Engineering
Citation	[29] B. Fjukstad and L. A. Bongo, "A review of scalable bioinformatics pipelines," *Data Science and Engineering*, vol. 2, no. 3, pp. 245–251, 2017

Paper 5

Title	nsroot: Minimalist Process Isolation Tool Implemented With Linux Namespaces.
Authors	Inge Alexander Raknes, **Bjørn Fjukstad**, Lars Ailo Bongo.
Description	Describes a tool for process isolation built using Linux namespaces.
Contribution	I contributed to the writing of the manuscript, specifically to the literature review and related works.
Publication date	26 November 2017
Publication venue	Norsk Informatikkonferanse 2017.
Citation	[30] I. A. Raknes, B. Fjukstad, and L. Bongo, "nsroot: Minimalist process isolation tool implemented with linux namespaces," *Norsk Informatikkonferanse*, 2017

Paper 6

Title	Reproducible Data Analysis Pipelines for Precision Medicine
Authors	**Bjørn Fjukstad**, Vanessa Dumeaux, Michael Hallett, Lars Ailo Bongo
Description	This paper outlines how we used the SMEs approach to build `walrus`.
Contribution	I designed, implemented, and performed the evaluation of `walrus`. I also wrote the manuscript.
Publication	To appear in the proceedings of the 2019 27th Euromicro International Conference On Parallel, Distributed and Network-based Processing (PDP).
Citation	[27] B. Fjukstad, V. Dumeaux, M. Hallett, and L. A. Bongo, "Reproducible data analysis pipelines for precision medicine," To appear in the proceedings of 2019 27th Euromicro International Conference On Parallel, Distributed and Network-based Processing (PDP). IEEE, 2019

In addition to the above papers I have also contributed to the following papers during the project:

- Y. Kiselev, S. Andersen, C. Johannessen, B. Fjukstad, K. S. Olsen, H. Stenvold, S. Al-Saad, T. Donnem, E. Richardsen, R. M. Bremnes *et al.*, "Tran-

scription factor pax6 as a novel prognostic factor and putative tumour suppressor in non-small cell lung cancer," *Scientific reports*, vol. 8, no. 1, p. 5059, 2018

- B. Fjukstad, N. Angelvik, M. W. Hauglann, J. S. Knutsen, M. Grønnesby, H. Gunhildrud, and L. A. Bongo, "Low-cost programmable air quality sensor kits in science education," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. ACM, 2018, pp. 227–232

These are not included in the thesis but they demonstrate other usage examples of our approach.

1.6 Dissertation Plan

This thesis is organized as follows. Chapter 2 describes the characteristics of state-of-the-art biological datasets in systems epidemiology and how we have developed an approach to analyze these. In Chapter 3 we describe how we used the same ideas and model to develop applications for interactively exploring results from statistical analyses. Chapter 4 explores how we can develop analysis pipelines for high-throughput sequencing datasets in precision medicine. It describes in detail how we use a container centric development model to build a tool, walrus, to develop and execute these pipelines. Finally, Chapter 5 concludes the work and discusses future directions.

2

Modern Biological Data Management and Analysis

From the discovery of the DNA structure by Watson and Crick in 1953[33] to the sequencing of the human genome in 2001,[34, 35] and the massively parallel sequencing platforms in the later years[36], the scientific advances have been tremendous. Today, single week-long sequencing runs can produce as much data as did entire genome centers just years ago.[12] These technologies allow researchers to produce data faster, cheaper and more efficiently, now making it possible to sequence the entire genome from a patient in less than a day. In addition to faster data generation, the new datasets are also of higher quality.

Ensuring reproducibility through sharing of analysis code and datasets is necessary to advance science.[37] From the many obstacles to replicate results from the most influential papers in cancer research[38], it is apparent that it is important to thoroughly document the entire workflow from data collection to interpretable results. This requires implementing best practices for data storage and processing. Such best practices are also necessary for large and complex research studies where data collection, analysis, and interpretation may span decades, and therefore be done in several iterations.

Ensuring reproducible science is a necessity for individual researchers, within research groups, between research groups, and to the greater society. It is not

just about simplifying the replication of results, but is also related to advancing science from known results and methods. Within science, it is problematic for individual research and research groups to waste time and effort to re-apply previous results to new datasets because of poorly documented studies and results. Outside of science, it is problematic to *trust* science when studies are difficult or impossible to replicate or reproduce.

In this chapter we describe our efforts to establish an approach for reproducible analysis of biological data in a complex epidemiological study. We first give a short introduction to high-throughput datasets, before describing the needs of the researchers in the NOWAC study. While we have used the NOWAC study as a motivating example, we believe that these needs are found in other complex research studies. We describe the previous practice for data management and analysis, and propose a new approach to achieve reproducible analyses. Continuing, we show that our approach to manage research data and code can be used to develop a standardized data analysis pipeline. Further we provide best practices for data analysis and management.

2.1 High-Throughput Datasets for Research and Clinical Use

High-throughput technologies that are now widely used to study complex diseases such as cancer. DNA sequencing is the process of determining the order of nucleotides within a strand of DNA. High-throughput Sequencing (HTS), or Next-generation Sequencing (NGS), is a term used to describe newer technology that enables massively-parallel sequencing of DNA. HTS instruments sequence millions of short base pairs, and we assemble these in the data analysis process. Typical sequencing datasets are in the size of hundreds of Gigabytes (GBs) per sample.

While HTS can study the sequence of bases, microarrays have been used to study the transcriptome, or the genes actively expressed. While the genome is mostly fixed for an organism, the transcriptome is continuously changing. These instruments report the expression levels of many target genes, and by profiling these we can study which genes are active in the biological sample. Microarray datasets are in the size of megabytes per sample.

Another technique to study the transcriptome is to use RNA-seq technology based on HTS. RNA-seq instruments also read millions of short base pairs in parallel, and can be used in gene expression analysis. Because of its higher quality output, RNA-seq is the successor to microarray technology. These datasets

are also in the size of hundreds of GBs.

Precision medicine uses patient-specific molecular information to diagnose and categorize disease to tailor treatment to improve health outcome.[39] Important research goal in precision medicine are to learn about the variability of the molecular characteristics of individual tumors, their relationship to outcome, and to improve diagnosis and therapy.[40] International cancer institutions are therefore offering dedicated personalized medicine programs, but while the data collection and analysis technology is emerging, there are still unsolved problems to enable reproducible analyses in clinical settings. For cancer, HTS is the main technology to facilitate personalized diagnosis and treatment, since it enables collecting high quality genomic data from patients at a low cost.

2.2 Norwegian Women and Cancer (NOWAC)

In this thesis we have used data from the NOWAC study extensively. The NOWAC study is a prospective population-based cohort that tracks 34% (170.000) of all Norwegian women born between 1943–57.[20] The data collection started in NOWAC in 1991 with surveys to cover, among others, the use of oral contraceptives and hormonal replacement therapy, reproductive history, smoking, physical activity, breast cancer, and breast cancer in the family. The datasets are also integrated with data from The Norwegian Cancer Registry, and The Cause of Death Registry in Statistics Norway. In addition to the questionnaire data, the study includes blood samples from 50.000 women, as well as more than 300 biopsies. From the biological samples the first gene expression dataset was generated in 2009, and the study now also features miRNA, methylation, metabolomics, and RNA-seq datasets.

The data in the NOWAC cohort allows for a number of different study designs. While it is a prospective cohort study, we can also draw a case-control study from the cohort, or a cross-section study from the cohort. From the NOWAC cohort there has been published a number of research papers that investigate the questionnaire data together with the gene expression datasets.[25, 41] We have also used the gene expression datasets to explore gene expression signals in blood and interactions between the tumor and the blood systemic response of breast cancer patients.[42, 26]. Some analyses have resulted in patents[43] and commercialization efforts. While many interesting patterns and results have been studied, there are still many unexplored areas in the available datasets.

In the NOWAC study we are a traditional group of researchers, PhD and Post-Doc students, and administrative and technical staff. Researchers have backgrounds

from statistics, medicine, or epidemiology, and now also computer science. The administrative and technical staff is responsible for managing the data, both data collection and data delivery to researchers.

2.2.1 Data Management and Analysis

Surveys are the traditional data collection method in epidemiology. But today, questionnaire responses are increasingly integrated with molecular data. However, surveys are still important for designing a study that can answer particular research questions. In this section we describe how such integrated data analysis was done in NOWAC before we developed our approach. We believe many studies have, or are still, analyzing epidemiological data using a similar practice.

In the NOWAC study we have stored the raw survey and registry data in an in-house database backed up to an independent storage node. Previously, researchers had to apply to get data exported from the database by an engineer. This was typically done through SAS scripts that did some preprocessing, e.g. selecting applicable variables or samples, before the data was sent to researchers as SAS data files. The downstream analysis was typically done in SAS. Researchers used e-mail to communicate and send data analysis scripts, so there was not a central hub with all the scripts and data.

In addition to the questionnaire data, the NOWAC study also integrates with registries which are updated regularly. The datasets from the different registries are typically delivered as comma-separated values (CSV) files to our scientific staff, which are then processed into a standardized format. Since the NOWAC study is a prospective cohort, a percentage of the women are expected to get a cancer and move from the list of controls into the list of cases.

In the NOWAC study we have processed our biological samples outside our research institution. The received raw datasets were then stored on a local server and made available to researchers on demand. Because of the complexity of the biological datasets, many of these require extensive pre-processing before they are analysis-ready.

2.3 Enabling Reproducible Research

To enable reproducible research in the NOWAC study we have developed a system for managing and documenting the available datasets, a standardized data preprocessing and preparation system, and a set of best practices for data

analysis and management. We designed our management and analysis system as a SME that we could later use in the Pippeline system. To determine the demands of the users, we collaboratively identified issues with our previous practice and a set of requirements for a system to solve these issues.

We first identified issues with our previous practice:

1. It was difficult to keep track of the available datasets, and to determine how these had been processed. We had no standard data storage platform or structure, and there were limited reports for exported datasets used in different research projects.

2. There was no standard approach to preprocess and initiate data analysis. This was because the different datasets were analyzed by different researchers, and there was little practice for sharing reusable code between projects.

3. It became difficult to reproduce the results reported in our published research manuscripts. This was because the lack of standardized preprocessing, sharing of analysis tools, and full documentation of the analysis process.

To solve these issues and enable reproducible research in the NOWAC study, we had to develop a system for managing the data, code, and our proposed best practices for analyzing the data. We started with identifying a set of requirements for a system to manage and document the different datasets:

- It should provide users with a single interface to access the datasets, their respective documentation, and utility functions to access and analyze the data.

- It should provide version history for the data and analysis code.

- The system should provide reproducible data analysis reports[1] for any dataset that has been modified in any way.

- It should be portable and reusable by other systems or applications.

To satisfy the above requirements we developed the `nowac` R package, a software package in the R programming language that provides access to all data, documentation, and utility functions. Since it is a requirement that

1. Such as an R Markdown file which, when executed, generates the output data and optional documentation including plots, tables etc.

it should be reusable we could then implement a data preparation system, Pippeline, ontop of this R package. We identified a set of requirements for this data preprocessing and preparation system as well:

- The data preprocessing and preparation system should provide users with an interactive point-and-click interface to generate anlaysis-ready datasets from the NOWAC study.
- It should use the `nowac` R package to retrieve datasets.
- It should provide users with a list of possible options for filtering, normalization, and other options required to preprocess a microarray dataset.
- It should genererate a reproducible report along with any exported dataset.

Finally, we developed a set of best practices for data analysis in our study. In the rest of the section we detail how we built the `nowac` package, the Pippeline, and the best practices for data analysis.

2.3.1 The `nowac` Package

The `nowac` R package is our solution for storing, documenting, and providing analyis functions to process the datasets in the NOWAC study. We use `git` to version control the analysis code and datasets, and store the repository on a self-hosted git server. We bundle together all datasets in the `nowac` package. This includes both questionnaire, registry, and gene expression datasets. Because none of these are particularly large (no single dataset being more than tens of GBs) we are able to distribute them with our R package. Some datasets require pre-processing steps such as outlier removal before the analysts can explore the datasets. For these datasets we store the *raw* datasets, processed data, and the analysis-ready clean datasets. We store the raw datasets in their original format, while clean and processed datasets are stored as R data files to simplify importing them in R. In addition to the datasets themselves we store the R code we used to generate the datasets. For clarity, we decorate the scripts with specially formatted comments that can be used with knitr[44] to generate reproducible data analysis reports. These highlight the transformation of the data from raw to clean, with information such as removed samples or data normalization methods.

We have documented every dataset in R package. The documentation includes information such as data collection date, instrument types, the persons involved with data collection and analysis, pre-processing methods etc. When users

install the `nowac` package the documentation is used to generate interactive help pages which they can browse in R, either through a command line or through an integrated development environment (IDE) such as RStudio. We can also export this documentation to a range of different formats, and researchers can also view them in the R interface. Figure 2.1 shows the user interface of RStudio where the user has opened the documentation page for one of the gene expression dataset.

In the NOWAC package we also provide utility functions to get started with the analysis of our datasets. Because of the specialized nature of the different research project the NOWAC package only contains helper functions to start analyzing NOWAC data, e.g. retrieving questionnaire data.

We use a single repository for the R package, but have opted to use `git submodules` for datasets in the R package. This allows us to separate the access to the datasets, and the documentation and analysis code. Everyone with access to the repository can view the documentation and analysis code, but only scientific staff have access to the data. There are however drawbacks to creating one large repository for both data and code. Since git stores every version of a file, these types of repositories may become large if the datasets are changing a lot over time, and are stored in binary formats, e.g. gene expression datasets. We have explored different techniques to minimize our repository and have opted to store all datasets as git submodules[45]. Submodules allow us to keep the main repository size down while still versioning the data. There are extensions to git for versioning large datasets. git-raw[46], git-annex[47] git-lfs[48] all provide extensions that essentially replace large files in a git repository with pointers or other metadata, and store the actual files in an external storage server. Since our datasets are relatively small and static, we did not opt for any of these. Future versions may investigating these extensions, but the key point is to version all datasets using a familiar tool, namely git.

2.4 Standardized Data Analysis

Analyzing the biological data in the NOWAC study consists of four major parts as show on Figure 2.2. First, as explained above the raw datasets are added to the `nowac` R package and documented thoroughly by a data manager. Second, we manually examine the biological datasets to detect outliers. We add information about outliers to the `nowac` R package along with reports that describe why an observation is marked as an outlier. Third, the data manager generates an analysis-ready dataset for a research project using the interactive Pippeline tool. This dataset is preprocessed, and integrated with questionnaire and registry datasets. Fourth, researchers analyze the dataset with their tools of choice, but

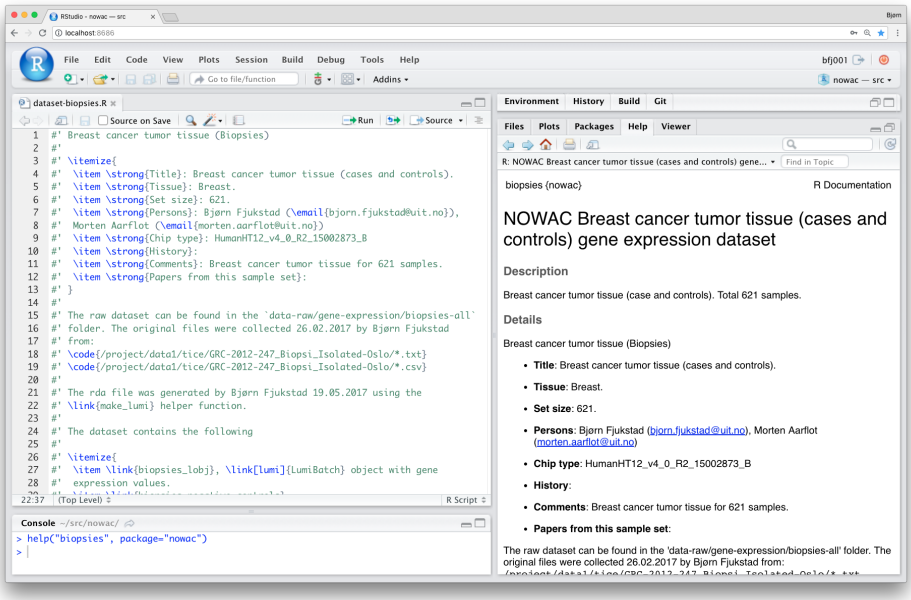


Figure 2.1: A screenshot of the user interface of R Studio viewing the documentation help page for the "Biopsies" dataset in the NOWAC study. The right-hand panel shows the documentation generated by the code in the top left panel. The bottom left panel shows the R command that brought up the help page.

following our best practices for data analysis.

2.4.1 Pippeline

We have developed our preprocessing pipeline for gene expression data as a point-and-click web application called Pippeline. The web application is standalone and does not require the users to use any command-line tools or have any programming knowledge. Pippeline generates an analysis-ready dataset by integrating biological datasets together with questionnaire and registry data, all found in our `nowac` package. It uses pre-discovered outliers to exclude samples, and presents the user with a list of possible processing options. It exports the analysis-ready R data files together with a reproducible data analysis report, an R script, that describes all processing steps. Figure 2.3 shows the filtering step in Pippeline where users define at what level they wish to exclude gene expression probes in the dataset.

The web application is implemented in R using the `Shiny` framework. It uses the `nowac` R package to retrieve all datasets.

2.5 Best Practices

From our experiences we have developed a set of best practices for data analysis. These apply both to researchers, and developers and the technical staff managing the data in a research study:

Document every step in the analysis. Analysis of modern datasets is a complex exercise with the possibility of introducing an error in every step. Analysts often use different tools and systems that require a particular set of input parameters to produce results. Thoroughly document every step from raw data to the final tables that go into a manuscript.

In the NOWAC study we write help pages and reports for all datasets, and the optional pre-processing steps.

Generate reports and papers using code. With tools such as R Markdown[49] and kntir there are few reasons for decoupling analysis code with the presentation of the results through reports or scientific papers. Doing so ensures the correctness reported results from the analyses, and greatly simplifies reproducing the results in a scientific paper.

In the NOWAC study we produce reports from R code. These include pre-

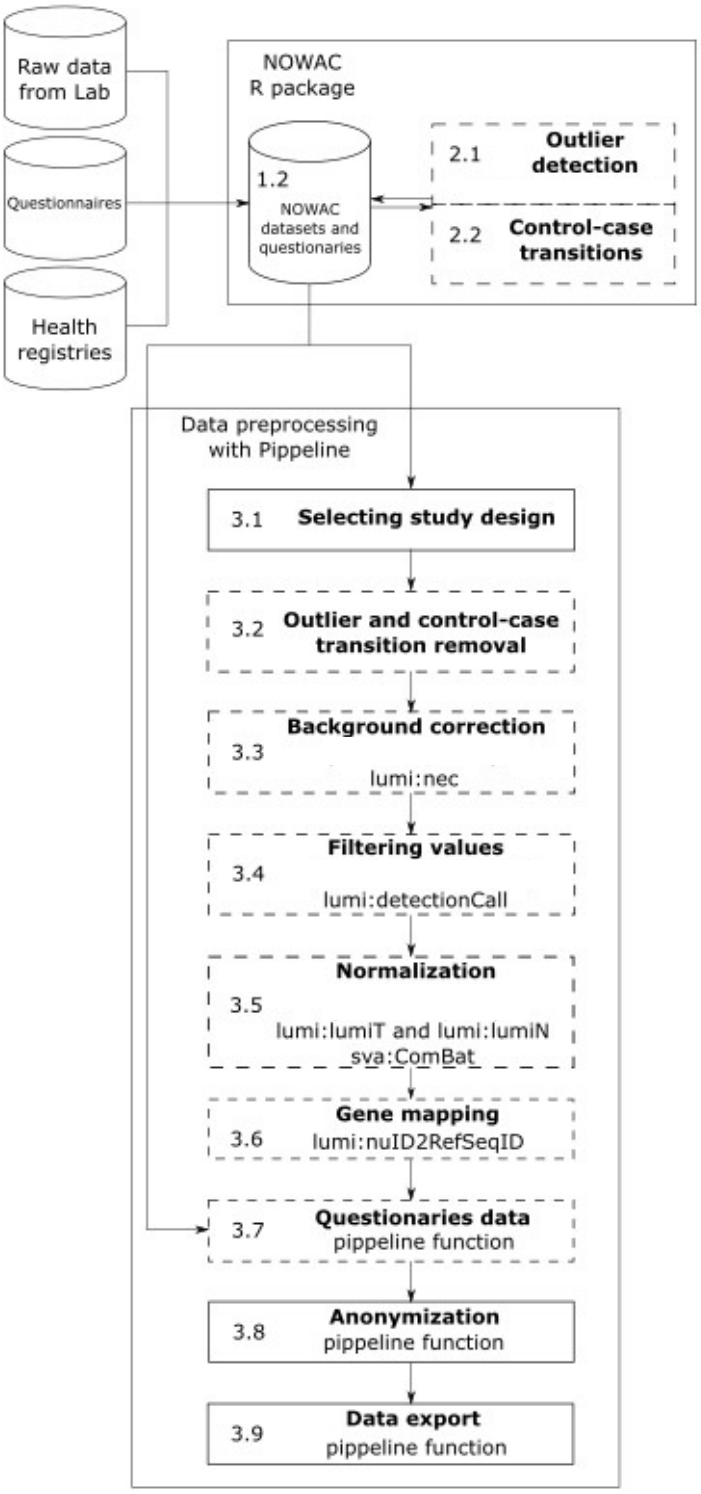


Figure 2.2: The standardized data processing pipeline for gene expression data analysis in the NOWAC study. Steps with a dashed line are optional, while steps marked with a solid line are mandatory.

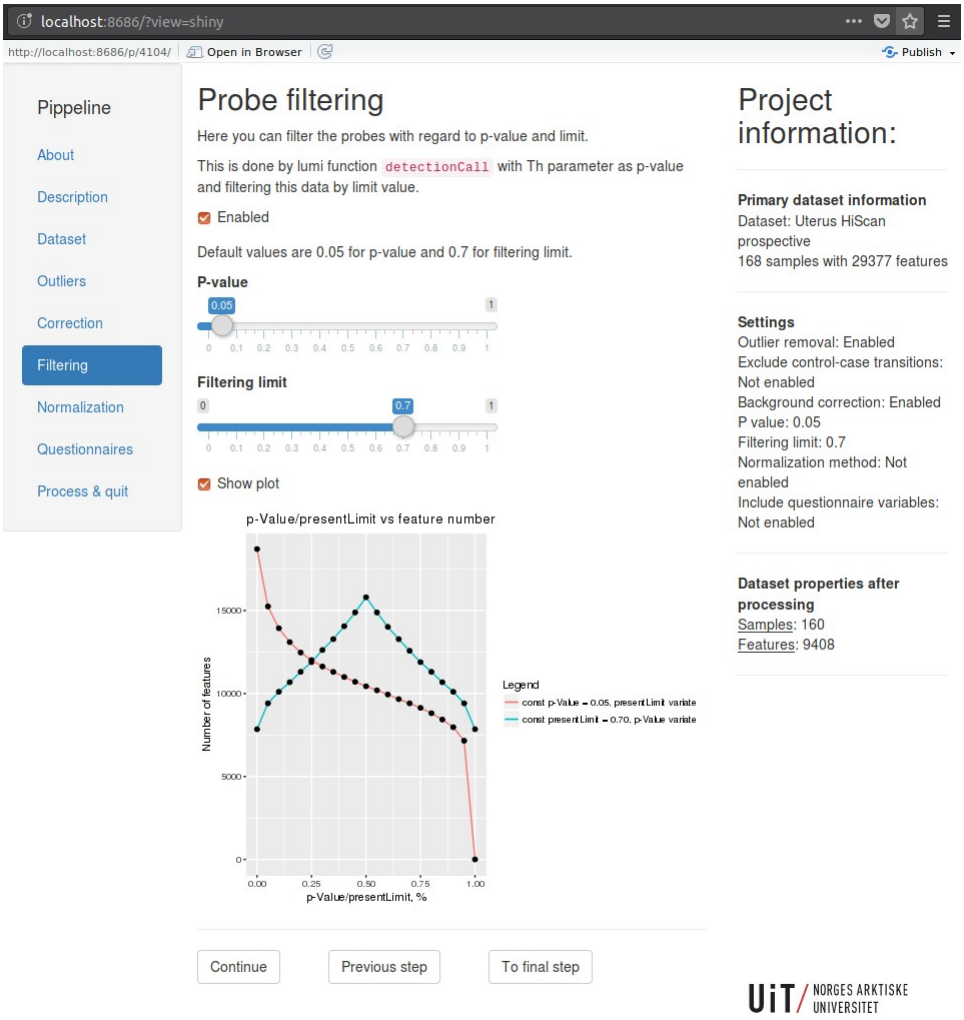


Figure 2.3: A screenshot of the web-interface of Pippeline. In the screenshot users can define at what level they want to filter out probes in the gene expression dataset. Users can define that the output dataset will only include gene expression probes that are present in a percent of the observation.

processing and data delivery of datasets to researchers. One example of a report is the analyses done in [31] where we documented the association between PAX6 gene expression and PAX6 target genes. Through a simple R script we could share the results and underlying analysses.

Version control everything. Both code and data changes over the course of a research project. Version control everything to make it possible to retrace changes and the person responsible for them. It is often necessary to roll back to previous versions or a dataset or analysis code, or to identify the researches that worked on specific analyses.

In the NOWAC study we encourage the use of git to version control both source code and data.

Collaborate and share code through source code management (SCM) systems. Traditional communication through e-mail makes it difficult to keep track of existing analyses and their design choices both for existing project members and new researchers. With SCM hosting systems such as Github developing analysis code becomes more transparent to other collaborators, and encourages collaboration. It also simplifies the process or archiving development decisions such as choosing a normalization method.

In the NOWAC study we collaborate on data analysis through a self-hosted Gitlab[50] installation. We also open-source code on Github.

2.6 Discussion

In this chapter we have proposed an approach to enable reproducible analyses in a complex epidemiological study. While we applied our approach to a specific epidemiological research study, we believe that it is generalizable to other biomedical analyses and even other scientific disciplines.

Making science reproducible is of growing interest. In this chapter we outlined the main best practices from our experiences in systems epidemiology research, and believe that these are generalizable to other fields as well. The best practices we arrived at follow the lines of other have described before us,[51] and we believe that these are necessary for both our research group, but also to the scientific community, to follow.

Bundling and sharing the analysis code together with the datasets behind a research paper is not a new idea. Sharing these collections, or compendia, of data, text, and code have been described more than a decade ago.[52] There

are many examples of studies that put in significant efforts to develop tools in R for transparent data science, to produce better science in less time.[53, 54, 55] In common is the explicit documentation of the final results using reproducible data analysis reports, and functions from shared R packages to generate these. They also structure the datasets and document these in a standardized manner to simplify the analysis. Many of these packages are open-sourced for everyone to use, and we hope to do the same.

While the majority of the researchers in NOWAC have previously used the closed-source and heavily licensed SAS or STATA for their analyses of the questionnaire data, all researchers working on molecular data are using R. We developed an R package for researchers in our study to simplify their analyses on both questionnaire and molecular datasets. With the R package researchers could investigate the available datasets and analyze them in the same environment. The great strength of R comes from its many up-to-date and actively maintained packages for analyzing, plotting, and interpreting data. Bioconductor[4] and the Comprehensive R Archive Network (CRAN)[56] provide online hosing for many packages, and users can mix and match these packages to fit their need. In addition, R is open-source and free to use on a wide range of operating systems and environments. Providing a single software package shortens the time to analysis, and improves researcher productivity since they do not have to start from scratch when analyzing a new dataset. In addition, it standardizes the analyses and makes the data analysis process more transparent. We believe that our solution can be applied to other datasets and projects within different scientific disciplines, enabling more researchers to take advantage of the many collected, but not yet analysis-ready datasets.

While taking advantage of powerful computational tools is beneficial, they often require trained users. A potential drawback of using an R package version controlled in `git` to manage, document, and analyze research datasets is the prerequisite programming skills for researchers. This may be an obstacle for many researchers, but once they master the skills needed to analyze their data programmatically, not just through a point-and-click interface, we believe that it provides deeper knowledge into the analyses. While programming skills may be absent in the training of many researchers, we believe that it is just a matter of time before programming skills are common in the scientific community.

There are many approaches to store and analyze biological datasets. One major drawback with the implementation of our approach in the `nowac` R package is its size. While microarray datasets are relatively small compared to sequencing data, when these datasets grow in number the total size grows as well. This will impact the compile time for the R package, and also its size when it is shared with other researchers. Others have also reported that package size is an issue, but are also investigating alternatives.[55] With larger datasets we

might experiment with extensions to git, e.g. `git-lfs`, as we have done in Chapter 4.

Since we developed the Pippeline to preprocess our gene expression datasets, it has been expanded to work with RNA-seq, Methylation and microRNA datasets as well. By using the Pippeline with new datasets researchers now have access to the full preprocessing history behind each dataset available in the research study.

As mentioned, we believe that our approach is applicable data management and anlysis in other research groups as well. Other research groups can follow the steps as described in this chapter to organize datasets and code in a software package, e.g. an R package, and share this both within and outside the research group. Sharing the analysis software through websites such as Github will help other researchers apply the techniques on their own datasets. While we aim to make all our code, documentation, and datasets public, we are unfortunately not there yet. We are working on a public version of the `nowac` R package and the Pippeline, but we must guarantee that the respective repositories do not contain any sensitive information from the datasets. This is ongoing work, but an important step toward making the research more transparent.

2.7 Conclusion

In summary, we believe that there are four general rules toward reproducible analyses. We believe that they apply to both our research study and other similar epidemiological studies:

- Document and version control datasets and analysis code within the study.
- Share datasets and analysis code through statistical software packages.
- Share and report findings through reproducible data analysis reports.
- Standardize and document common data preprocessing and wrangling steps.

In this chapter we have demonstrated one approach for reproducible management and analysis of biological data. The needs of the users that we describe in this chapter helped form the work in the next two chapters.

3

Interactive Data Exploration Applications

The main goal of a data exploration application in bioinformatics is to help users discover interesting patterns in a biological dataset. Because of the complexity of biological data and analyses, we need specialized software to help find these patterns. Explorative analysis is essential for understanding biological functions in high-throughput biological datasets. Applications that provide interactive interfaces and visualizations can help researchers study the datasets to discover emerging patterns.

Analysing high-throughput biological datasets require specialized analysis software. Such software is usually written in statistical programming languages, e.g., Python or R, which provide a wealth of statistical packages and frameworks. However, these specialized programming environments often do not provide interactive interfaces for researchers that want to explore the results from the analyses without using a programmatic interface. Frameworks such as Shiny[7] and OpenCPU[8] allow application developers to build systems to interactively explore results from statistical analyses in R. These systems can then provide understandable graphical user interfaces on top of complex statistical software that require programming skills to navigate. To interpret data, experts regularly exploit prior knowledge via database queries and the primary scientific literature. There are a wealth of online databases, some of which provide open Application Programming Interfaces (APIs) in addition to

web user interfaces that application developers can make use of. For visually exploring biological data there are a range of tools, such as Cytoscape[18] and Circos[19], that support importing an already-analyzed dataset to visualize and browse the data. One problem with these are that they are decoupled from the analysis, making it difficult to retrace the data processing prior to the end results.

The main issue for developing these types of data exploration applications is that they require the integration of disparate systems and tools. The datasets require specialized analysis software, often with large computational resources, and the end users require simple point-and-click interface available on their device. In addition it is crucial for reproducibility to keep track of the data processing steps that were used to generate end visualizations.

We have developed two data exploration applications, Kvik Pathways[22] and MIxT[23, 26] for exploring transcriptional profiles in the NOWAC study through interactive visualizations integrated with biological databases. We first developed Kvik Pathways to explore transcriptional profiles in the context of biological pathway maps. It is a three-tiered web application consisting of three central components, that we later refactored into three separate microservices for use in other applications. These three microservices make up the SMEs in our approach for building data exploration applications. With these microservices we implemented the MIxT web application, and generalized our efforts into general design principles for data exploration applications. While our applications provide specialized user interfaces, we show how the design patterns and ideas can be used in a wide range of use cases. We also provide an evaluation that shows that our approach is suitable for this type of interactive applications.

This chapter is based on Papers 1 and 2, as well as the general descriptions of the MIxT system in Paper 3. The rest of the chapter is organized as follows: First we present the two motivating use cases for our applications. We then detail the requirements for these types of interactive applications. Following the requirements we detail the Kvik Pathways application, including its architecture and implementation. We then show how we use this first application to generalize its design principle and show we can use them to build applications that follow the SME approach. Following is a description of the implementation of the SMEs approach in the microservices in Kvik. We present how we used these to develop the MIxT web application. Finally we discuss our approach in context of related work, and provide a conclusion.

3.1 Motivating Examples

The need for interactive applications has come from two different previous projects in the NOWAC study. Both of these rely on advanced statistical analyses and produce comprehensive results that are interpreted by researchers in the context of related information from online biological databases. The end results from the statistical analyses are typically large tables that require manual inspection and linking with known biology. Below we describe the two applications before we detail the requirements, design and implementation of the applications.

3.1.1 High and Low Plasma Ratios of Essential Fatty Acids

The aim of the first application was a to explore the results from a previous published project that compared gene expression in blood from healthy women with high and low plasma ratios of essential fatty acids.[25] Gene expression differences where assessed and determined that there were 184 differentially expressed genes. When exploring this list of 184 genes, functional information was retrieved from GeneCards and other repositories, and the list was analyzed for overlap with known pathways using MSigDB (available online at broadinstitute.org/gsea/msigdb). The researchers had to manually maintain overview of single genes, gene networks or pathways, and gather functional information gene by gene while assessing differences in gene expression levels. With this approach, researchers were limited by their own capacity to retrieve information manually from databases and keep it up to date. An application could automate the retrieval and ensure that the data is correct and up to date.

3.1.2 Tumor-Blood Interactions in Breast Cancer Patients

The aim of the Matched Interactions Across Tissues (MIxT) study was to identify genes and pathways in the primary breast tumor that are tightly linked to genes and pathways in the patient blood cells.[26] We generated and analyzed expression profiles from blood and matched tumor cells in 173 breast cancer patients included in the NOWAC study. The MIxT analysis starts by identifying sets of genes tightly co-expressed across all patients in each tissue. Each group of genes or modules were annotated based on a priori biological knowledge about gene functionality. Then the analyses investigate the relationships between tissues by asking if specific biologies in one tissue are linked with (possibly distinct) biologies in the second tissue, and this within different subgroup of patients (i.e. subtypes of breast cancer).

3.2 Requirements

From these two studies we identified a set of requirements that the data exploration applications should satisfy:

Interactive The applications should provide interactive exploration of datasets through visualizations and integration with relevant information.

Familiar The applications should use familiar visual representations to present information to researchers. For more efficient data exploration it is effective to use representations that researchers are familiar with both from the literature and from other applications.

Simple to use Researchers should not need to install software to explore their data through the applications. The applications should protect the researcher from the burden of installing and keeping an application up to date.

Lightweight Data presentation and computation should be separated to make it possible for researchers to explore data without having to have the computational power to run the analyses. With the growing rate data is produced at, we cannot expect that researchers have the resources to store and analyze data on their own computers.

With these requirements in mind we set out to develop two applications for interactively explore the results from the studies along with information from online databases.

3.3 Kvik Pathways

The first application we developed was Kvik Pathways. Kvik Pathways allows users to interactively explore a molecular dataset, such as gene expression, through a web application. It provides pathway visualizations and detailed information about genes and pathways from the KEGG database. Figure 3.1 shows a screenshot of the user interface of Kvik Pathways. Through pathway visualizations and integration with the KEGG databases, users can perform targeted exploration of pathways and genes to get an overview of the biological functions that are involved with gene expression from the underlying dataset. Kvik Pathways gathers information about related pathways and retrieves relevant information about genes, making it unnecessary for researchers to spend valuable time looking up this information manually. Previously researchers had to manually retrieve information from KEGG while browsing pathway maps,

interrupting the visual analysis process. Kvik Pathways retrieves information about genes without the researcher having to leave the pathway visualization to retrieve relevant information.

3.3.1 Analysis Tasks

To efficiently develop the application we designed 3 analysis tasks that the application supports.

A1: Explore gene expression in the context of KEGG pathway maps. It provides users with a list of pathway maps to choose from, and the application will generate an interactive visualization including gene expression values.

A2: Investigate and retrieve relevant biological information. It provides users with direct links to online databases with up to date information.

A3: Explore relationships between pathway maps. When users select a gene from a pathway map they get a list of other pathway maps that this gene is found in, in addition to their similarity. This allows users to investigate the biological processes the genes are a part of.

3.3.2 Architecture

Kvik Pathways has a three-tiered architecture of independent layers (Figure 3.2). The browser layer consists of the web application for exploring gene expression data and biological pathways. A front-end layer provides static content such as HTML pages and stylesheets, as well as an interface to the data sources with dynamic content such as gene expression data or pathway maps to the web application. The backend layer contains information about pathways and genes, as well as computational and storage resources to process genomic data such as the NOWAC data repository. We have used the packages in Kvik to develop the backend layer. These are discissed in detail in Section 3.5.

The Data Engine in the backend layer provides an interface to the NOWAC data repository stored on a secure server on our local supercomputer. In Kvik Pathways all gene expression data is stored on the computer that runs the Data Engine. The Data Engine runs an R session accessible over remote procedure calls (RPCs) from the front-end layer using RPy2[57] to interface with R. To access data and run analyses the Data Interface exposes a HTTP API to the browser layer (Table 3.1 provides the interfaces).

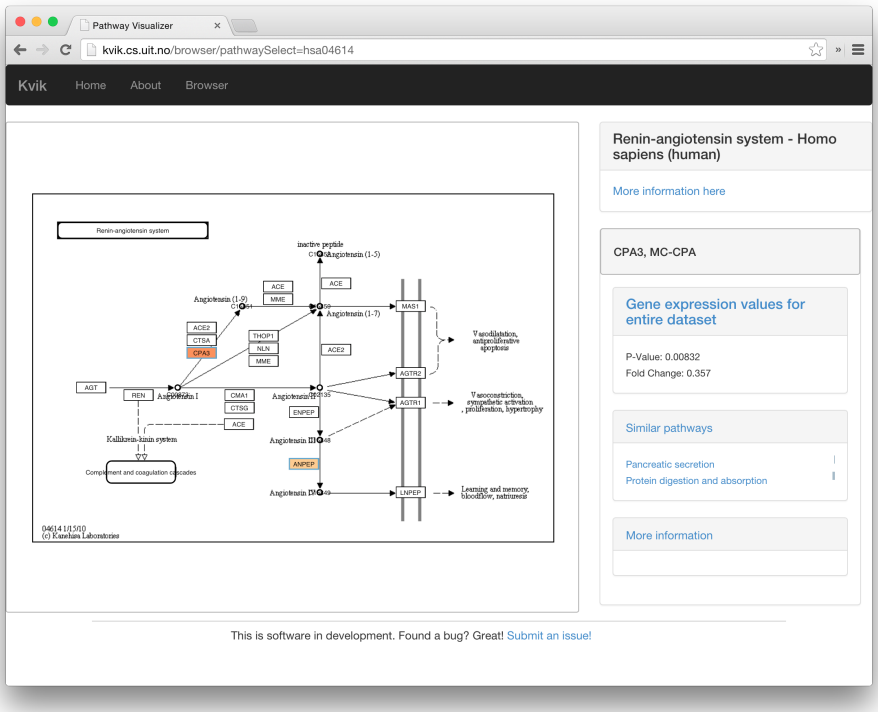


Figure 3.1: Screenshot of the renin-angiotensin pathway (KEGG pathway id hsa04614) in Kvik Pathways. Researchers can visually explore the pathways and read relevant information about genes in the right-hand panel.

Table 3.1: The REST interface to the Data Engine. For example, use `/genes/` to retrieve all available genes in our dataset.

URL	Description
/fc/[genes...]	Calculate and retrieve fold-change for the specified genes
/pvalues/[genes...]	Calculate and retrieve *p*-values for the specified genes
/exprs/[genes...]	Get the raw gene expression values from the dataset
/genes	Get a list of all genes in the dataset

3.3.3 Implementation

To create pathway visualizations the Kvik backend retrieves and parses the KEGG Markup Language (KGML) representation and pathway image from KEGG databases through its REST API.[58] This KGML representation of a pathway is an XML file that contains a list of nodes (genes, proteins or compounds) and edges (reactions or relations). Kvik parses this file and generates a JSON representation that Kvik Pathway uses to create pathway visualiza-

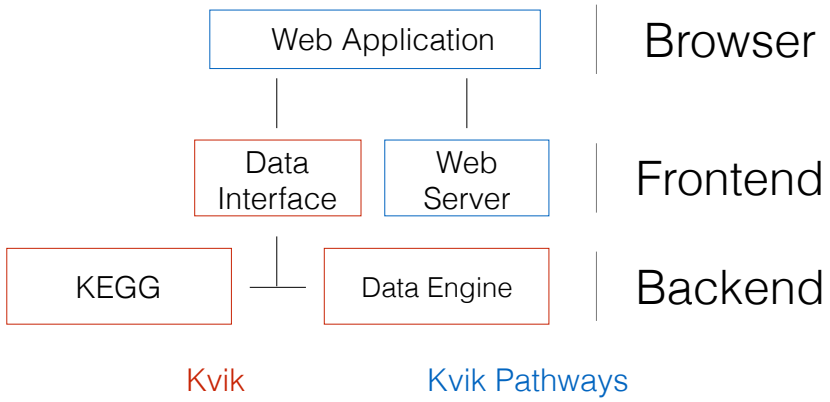
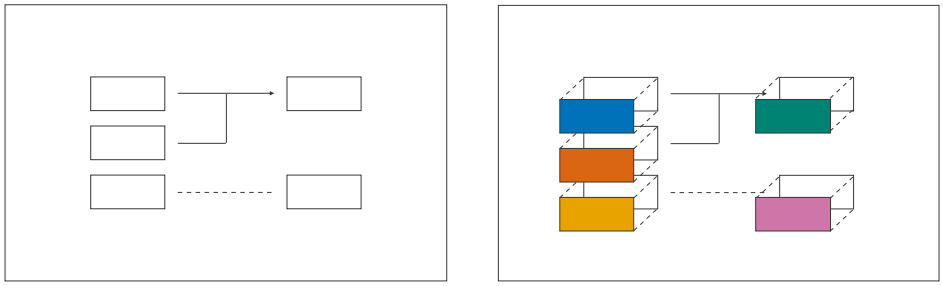


Figure 3.2: The three-tiered architecture of Kvik Pathways.

tions. Kvik Pathways uses Cytoscape.js[59] to create a pathway visualization from the list of nodes and edges and overlay the nodes on the pathway image. See Figure 3.3 for a graphical illustration of the process. To reduce latency when using the KEGG Representational state transfer (REST) API, we cache every response on our servers. We use the average fold change between the groups (women with high or low plasma ratios of essential fatty acids) in the dataset to color the genes within the pathway maps. To highlight p-values, the pathway visualization shows an additional colored frame around genes. We visualize fold change values for individual samples as a bar chart in a side panel. This bar chart gives researchers a global view of the fold change in the entire dataset.

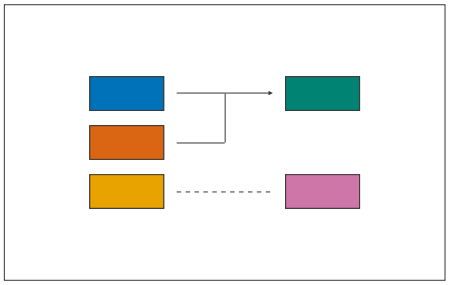
Kvik provides a flexible statistics backend where researchers can specify the analyses they want to run to generate data for later visualization. For example, in Kvik Pathways we retrieve fold change for single genes every time a pathway is viewed in the application. These analyses are run ad hoc on the backend servers and generates output that is displayed in the pathways in the client's web browser. The data analyses are implemented in an R script and can make use of all available libraries in R, such as Bioconductor.

Researchers modify this R script to, for example, select a normalization method, or to tune the false discovery rate (FDR) used to adjust the p-values that Kvik Pathways uses to highlight significantly differentially expressed genes. Since Kvik Pathways is implemented as a web application and the analyses are run ad hoc, when the analyses change, researchers get an updated application by simply refreshing the Kvik Pathways webpage.



(a) Original static pathway image from KEGG, placed as a background node

(b) Overlaying graph nodes from the KGML representation of the pathway



(c) Final visualization

Figure 3.3: Visualizing gene expression data on KEGG pathway maps.

3.3.4 Use Case: Analysis of Renin-Antiotensin Pathway

As an example of practical use of Kvik Pathways, we chose one of the significant pathways from the overlap analysis, the renin-angiotensin pathway (Supplementary table S5 in [25]). The pathway contains 17 genes, and in the pathway map we could instantly identify the two genes that drive this result. The color of the gene nodes in the pathway map indicates the fold change, and the statistical significance level is indicated by the color of the node's frame. We use this image of a biological process to see how these two genes (and their expression levels) are related to other genes in that pathway, giving a biologically more meaningful context as compared to merely seeing the two genes on a list.

3.4 Building Data Exploration Applications

Through the experiences developing the Kvik Pathways we identified a set of components and features that are central to building data exploration applications:

1. A low-latency language-independent approach for integrating, or embedding, statistical software, such as R, directly in a data exploration application.

2. A low-latency language-independent interface to online reference databases in biology that users can query to explore results in context of results in context of known biology.

3. A simple method for deploying and sharing the components of an application between projects.

In the following sections we describe how we designed and implemented these in Kvik, and how they later formed the bases of the SME approach that the MIXT web application builds upon.

3.5 Kvik

Kvik is a collection of software packages in the Go programming language. It is designed for developers that want to develop interactive data exploration applications. It is the foundation in our two data exploration applications, and has been iteratively developed through the last years. We have previously refered to Kvik as Kvik Framework[22], but we now refer to it only as Kvik. Kvik provides an interface to the R statistical programming language, both as a stand-alone service, a client library, and through an OpenCPU server. It provides an R-based pipelining tool that allows useres to specify and run statistical analysis pipelines in R. Kvik also contains a Javascript package for visualizing KEGG pathways using d3.[60] In addition it provides an interface with online databases such as MsigDB[61] and KEGG[62].

We used the experience building Kvik Pathways to completely re-design and re-implement the R interface in Kvik. From having an R server that can run a set of functions from an R script, it now has a clean interface to call any function from any R package, not just retrieving data as a text string but in a wide range of formats. We also re-built the database interface, which is now a separate service. This makes it possible to leverage its caching capabilities to improve latency. This transformed the application from being a single monolithic application into a system that consists of a web application for visualizing biological pathways, a database service to retrieve pathway images and other metadata, and a compute service for interfacing with the gene expression data in the NOWAC cohort. We could then re-use the database and the compute service in the MIxT application.

We have used these packages to develop the SME approach through services that provide open interfaces to the R programming language and the online databases. We outline these services in 3.5.1. In short the interfaces are accessible through an HTTP interface and can be used from any platform.

3.5.1 Microservices

We generalized our efforts from Kvik Pathways into the following design principles for building applications in bioinformatics:

Principle 1: Build applications as collections of language-agnostic microservices. This enables re-use of components and does not enforce any specific programming language on the user interfaces or the underlying components of the application.

Principle 2: Use software containers to package each service. This has a number of benefits: it simplifies deployment, ensures that dependencies and libraries are installed, and simplifies sharing of services between developers.

Compute Service

We have built a compute service that provides an open interface directly to the R programming language, thus providing access to a wealth of algorithm and statistical analysis packages that exists within the R ecosystem. Application developers can use the compute service to execute specialized analyses and retrieve results either as plain text or binary data such as plots. By interfacing directly with R, developers can modify input parameters to statistical methods directly from the user-facing application.

The compute service offers three main operations to interface with R: i) to call a function with one or more input parameters from an R package, ii) to get the results from a previous function call, and iii) a catch-all term that both calls a function and returns the results. We use the same terminology as OpenCPU[8] and have named the three operations Call, Get, and RPC respectively. These three operations provide the necessary interface for applications to include statistical analyses in the applications.

The compute service is implemented as an HTTP server that communicates with a pre-set number of R processes to execute statistical analyses. At initiation of the compute service, a user-defined number of R worker sessions are launched for executing analyses (default is 5). The compute service uses a round-robin scheduling scheme to distribute incoming requests to the workers. We provide

a simple FIFO queue for queuing of requests. The compute service also provides the opportunity for applications to cache analysis results to speed up subsequent calls.

Database Service

We have built a database service to interface with online biological databases. The service provides a low latency interface, it minimizes the number of queries to remote databases, and stores additional metadata to capture query parameters and database information. The database service provides an open HTTP interface to biology databases for retrieving meta-data on genes and processes. We currently have packages for interfacing with E-utilities[63], MSigDB, HGNC[64], and KEGG.

Both the compute and the databases service in Kvik build on the standard *net/http* package in the Go programming language.[1] The database service use the *gocache*[2] package to cache any query to an online database. In addition we deploy each service as Docker containers.[3]

3.6 MIXT

The MIXT system is an online web application for exploring and comparing transcriptional profiles from blood and tumor samples. It provides users with an interface to explore high-throughput gene expression profiles of breast cancer tumor data with matched profiles from the patients blood. We have used the microservices in Kvik to interface with statistical analyses and information from online biology databases.

3.6.1 Analysis Tasks

To efficiently develop the application we defined six analysis tasks (A1-A6) that the application supports:

A1: Explore co-expression gene sets in tumor and blood tissue. Users can explore gene expression patterns together with clinicopathological variables

1. `golang.org`
2. `github.com/fjukstad/gocache`.
3. Available at `hub.docker.com/r/fjukstad/kvik-r` and `hub.docker.com/r/fjukstad/db`.

(e.g. patient or tumor grade, stage, age) for each module. In addition we enable users to study the underlying biological functions of each module by including gene set analyses between the module genes and known gene sets.

A2: Explore co-expression relationships between genes. Users can explore the co-expression relationship as a graph visualization. Here genes are represented in the network with nodes and edges represent statistically significant correlation in expression between the two end-points.

A3: Explore relationships between modules from each tissue. We provide two different metrics to compare modules, and the web application enables users to interactively browse these relationships. In addition to providing visualizations the compare modules from each tissue, users can explore the relationships, but for different breast cancer patient groups.

A4: Explore relationships between clinical variables and modules. In addition to comparing the association between modules from both tissues, users also have the possibility to explore the association with a module and a specific clinical variable. It is also possible to explore the associations after first stratifying the tumors by breast cancer subtype (an operation that is common in cancer related studies to deal with molecular heterogeneity).

A5: Explore association between user-submitted gene lists and computed modules. We want to enable users to explore their own gene lists to explore them in context of the co-expression gene sets. The web application must handle uploads of gene lists and compute association between the gene list and the MIxT modules on demand.

A6: Search for genes or gene lists of interest. To facilitate faster lookup of genes and biological processes, the web application provides a search functionality that lets users locate genes or gene lists and show association to the co-expression gene sets.

3.6.2 Architecture

We structured the MIxT application with a separate view for each analysis task. To explore the co-expression gene sets (**A1**), we built a view that combines both static visualizations from R together with interactive tables for gene overlap analyses. Figure 3.4 shows the web page presented to users when they access the co-expression gene set 'darkturquoise' from blood. To explore the co-expression relationship between genes (**A2**) we use an interactive graph visualization build with Sigma.[65] We have built visualization for both tissues, with graph sizes of 2705 nodes and 90 348 edges for the blood network, and

2066 nodes and 50 563 edges for the biopsy network. To visualize relationships between modules from different tissues (**A3**), or their relationship to clinical variables (**A4**) we built a heatmap visualization. We built a simple upload page where users can specify their gene sets (**A5**). The file is uploaded to the web application which redirects it to a backend service that runs the analyses. Similarly we can take user input to search for genes and processes (**A6**).

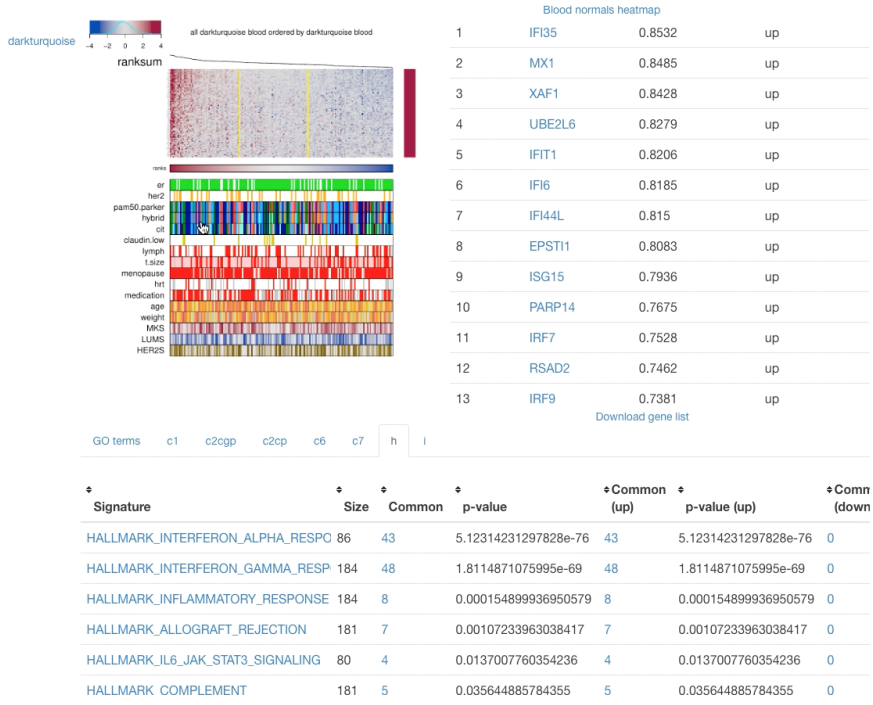


Figure 3.4: MIxT module overview page. The top left panel contains the gene expression heatmap for the module genes. The top right panel contains a table of the genes found in the module. The bottom panel contains the results of gene overlap analyses from the module genes and known gene sets from MSigDB.

For the original analyses we built an R package, mixtR,[4] with the statistical methods and static visualizations for identifying associations between modules across tissues. The mixtR package is based on the Weighted Gene Co-expression Network Analysis (WGCNA) R package to compute the correlation networks[66]. To make the results more easily accessible we built a web application that interfaces with the R package, but also online databases to retrieve relevant metadata. To make it possible to easily update or re-implement parts of the system without effecting the entire application, and we developed it using a microservice architecture. The software containers allowed the application to be deployed on a wide range of hardware, from local installations to

4. Available online at `github.com/vdumeaux/mixtR`.

cloud systems.

3.6.3 Implementation

From the six analysis tasks we designed and implemented MIxT as a web application that integrates statistical analyses and information from biological databases together with interactive visualizations. Figure 3.5 shows the system architecture of MIxT which consists of three parts i) the web application itself containing the user-interface and visualizations; ii) the compute service performing the MIxT analyses developed in an R package, delivering data to the web application; and iii) the database service providing up-to-date information from biological databases. Each of these components run in Docker containers making the process of deploying the application simple.

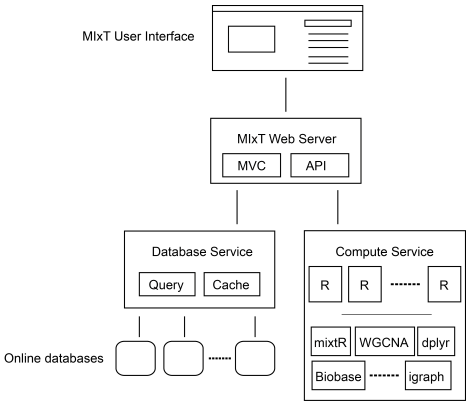


Figure 3.5: The architecture of the MIxT system. It consists of a web application, the hosting web server, a database service for retrieving metadata and a compute service for performing statistical analysis. Note that only the web application and the R package are specific to MIxT, the rest of the components can be reused in other applications.

The web application is hosted by a custom web server. This web server is responsible for dynamically generating the different views based on data from the statistical analyses and biological databases, and serve these to users. It also serves the different JavaScript visualization libraries and style sheets.

3.6.4 Evaluation

We evaluate the MIxT application by investigating response times for a set of queries to each of its two supporting services.

To evaluate the database service we measure the query time for retrieving

information about a specific gene with and without caching.[5] This illustrates how we can improve performance in an application by using a database service rather than accessing the database directly. We use a AWS EC2 *t2.micro*[6] instance to host and evaluate the database service. The results in Table 3.2 confirm a significant improvement in response time when the database service caches the results from the database lookups. In addition by serving the results out of cache we reduce the number of queries to the online database down to one.

Table 3.2: Time to retrieve a gene summary for a single gene, comparing different number of concurrent requests.

	1	2	5	10	15
No cache	956ms	1123ms	1499ms	2147ms	2958ms
Cache	64ms	64ms	130ms	137ms	154ms

We evaluate the compute service by running a benchmark consisting of two operations: first generate a set of 100 random numbers, then plot them and return the resulting visualization.[7] We use two *c4.large* instances on AWS EC2 running the Kvik compute service and OpenCPU base docker containers. The servers have caching disabled. Table 3.3 shows the time to complete the benchmark for different number of concurrent connections. We see that the compute service in Kvik performs better than the OpenCPU[8] alternative. We believe that speedup is because we keep a pool of R processes that handle requests. In OpenCPU a new R process is forked upon every request that results in any computation executed in R. Other requests such as retrieving previous results do not fork new R processes.

In summary our results show that the interface to the R programming language provides faster latencies, and that implementing a service for database lookups have clear benefits with regards to latency.

Table 3.3: Time to complete the benchmark with different number of concurrent connections.

	1	2	5	10	15
Kvik	274ms	278ms	352ms	374ms	390ms
OpenCPU	500ms	635ms	984ms	1876ms	2700ms

5. More details online at `github.com/fjukstad/kvik`.
6. See `aws.amazon.com/ec2/instance-types` for more information about AWS EC2 instance types.
7. More details at `github.com/fjukstad/kvik`.
8. Built using the *opencpu-server* Docker image.

3.6.5 Tumor Epithelium-Stroma Interactions in Breast Cancer

In addition to the MIxT web application for exploring the link between breast tumor and primary blood, we have also deployed a web application that investigates the link in another dataset[67] We have deployed the application online at `mixt-tumor-stroma.bci.mcgill.ca`. The web application is identical, but the underlying dataset is different.

3.6.6 air:bit

We have also used the microservice architecture in an application where users can upload and explore air pollution data from Northern Norway.[32] In the project, air:bit, students from upper secondary schools in Norway collect air quality data from sensor kits that they have built and programmed. The web application lets the students upload data from their kits, and provides a graphical interface for them to explore data from their own, and other participating schools. The system consists of a web server frontend that retrieves air pollution data from a backend storage system to build interactive visualizations. It also integrates the data with other sources such as the Norwegian Institute for Air Research and the The Norwegian Meteorological Institute.

3.7 Related Work

There are different technologies for developing different data exploration applications. We have surveyed comparable applications for exploring similar datasets to the ones we describe in this chapter, and underlying technology for developing these applications.

3.7.1 Applications

There are a wealth of resources for exploring biological pathway maps. KEGG[62] provides a large collection of static pathway maps that users can navigate through and download. They provide both static images of the pathways, as well as a textual representation of the pathway in the KEGG Markup Language (KGML). KEGG provides a REST API that developers can use to integrate both pathway maps and other information in their application. In KEGG Pathways we heavily rely on the data from KEGG. Reactome is an open-source peer-reviewed online knowledgebase of biomolecular pathways.[68] Users can download the entire graph database or explore it in their pathway visualization

tool. They have not yet made an API open for developers, but are planning to do so. Libraries such as KEGGViewer[69] allow developers to integrate pathway visualization maps in web applications, but these are generated using the KGML representations, that do not include additional visual cues found in the static KEGG pathway maps. enRoute[70] is a desktop application for exploring pathway maps from KEGG that combines the static pathway maps from KEGG in an interactive application. Pathview is both an R package and an online web application for exploring pathway maps.[71] The online web application is built on top of the R package and provides the same functionality, but through a GUI. Pathway generates static pathway visualizations based on pathway maps from KEGG.

There are few related systems that provide visualizations of WGCNA results. The R package from the original paper provides a wide range of different utility functions for visualization, but it is only accessible within the R environment. The WGCNA Shiny app[9] is an interactive application for performing, and exploring results from, WGCNA. The online version allows users to explore two demo datasets, and it is possible to download the application and change out the datasets locally. In short it is a web implementation of the WGCNA R package that allows users without any R experience perform WGCNA. It is developed and maintained by the eTRIKS platform.[72]

3.7.2 Technology

OpenCPU is a system for embedded scientific computing and reproducible research.[8] Similar to the compute service in Kvik, it offers an HTTP API to the R programming language to provide an interface with statistical methods. It allows users to make function calls to any R package and retrieve the results in a wide variety of formats such as JSON or PDF. OpenCPU provides a JavaScript library for interfacing with R, as well as Docker containers for easy installation, and has been used to build multiple applications.[10]. The compute service in Kvik follows many of the design patterns in OpenCPU. Both systems interface with R packages using a hybrid state pattern over HTTP. Both systems provide the same interface to execute analyses and retrieve results. Because of the similarities in the interface to R in Kvik we provide packages for interfacing with our own R server or OpenCPU R servers.

Shiny is a web application framework for R[11] It allows developers to build web applications in R without having to have any knowledge about HTML, CSS, or

9. Online a shiny.etriks.org/wgcna
10. `opencpu.org/apps.html`.
11. `shiny.rstudio.com`.

Javascript. While it provides an easy alternative to build web applications on top of R, it cannot be used as a service in an application that implements the user-interface outside of R.

Renjin is a JVM-based interpreter for the R programming language.[73] It allows developers to write applications in Java that interact directly with R code. This makes it possible to use Renjin to build a service for running statistical analyses on top of R. One serious drawback is that existing R packages must be re-built specifically for use in Renjin.

Cytoscape is an open source software platform for visualizing complex networks and integrating these with any type of attribute data.[74] Through a Cytoscape App, cyREST, it allows external network creation and analysis through a REST API[75], making it possible to use Cytoscape as a service. To bring the visualization and analysis capabilities to the web applications the creators of Cytoscape have developed Cytoscape.js[12], a JavaScript library to create interactive graph visualizations. Another alternative for biological data visualization in the web browser is BioJS It provides a community-driven online repository with a wide range components for visualizing biological data contributed by the bioinformatics community.[21] BioJS builds on node.js[13] providing both server-side and client-side libraries. In MIxT we have opted to build the visualizations from scratch using sigma.js and d3 to have full control over the appearance and functionality of the visualizations.

3.8 Discussion

In this chapter we have given a description of how we successfully built two data exploration applications for high-throughput biological datasets. We have iteratively developed these and ended up with an approach that allows us to develop applications from disparate systems.

As we have seen in 3.7 there are many applications that provide visualization tools to view and browse pathway maps, most of which use KEGG as its main data source. The applications then either reuse the pathway maps, and augment them with gene expression data, or use the underlying KGML description and generate their own graphical representation with gene expression data. Using the first method will provide the additional visual ques found in the static pathway images, but the visualizations are less flexible with regards to node and edge placement. Using the second method provides more flexible graphs

12. `js.cytoscapejs.org`.
13. `nodejs.org`.

with regards to layout, but this could make the visualizations less familiar to the users interpreting them. We believe that using the techniques that provides the most familiar representation of the pathways are easiest to interpret for the users.

With both of these techniques the underlying gene expression datasets are loaded using different techniques. Most systems allow users to specify gene expression values in some table format and render the values in top of the pathway map. These values are typically the end result of a long analysis process which users have to track manually. By integrating the visualization with the analysis software, typically R, it is possible to access data from anywhere in the analysis process, and also provide detailed information to the user regarding the underlying data analysis process. What separates our approach in Kvik Pathways to the other related systems, is this integration between the end visualization and the gene expression datasets. By using Kvik it is possible to develop applications that automatically lets users access the underlying data analysis, and thereby connecting the interpretable end results with the analyses.

The WGCNA Shiny app provides similar visualizations as our MIxT web application, but the application is limited to that of a web application. Shiny lets its users develop applications written purely in R, including the backend server and the user interfaces. In MIxT we developed an R package with a set of resources, or endpoints, for application developers to access through a Kvik R service. This allows application developers to develop the user-facing logic using any type of technology or framework. The resources are available through the HTTP API in Kvik making it possible for anyone to develop an application on top of the dataset and analyses. We acknowledge the strength of R for data analysis, but not for developing complex user-facing applications.

There are different arguments for reusing and sharing microservices over libraries in bioinformatics applications, that would justify the cost of hosting an maintaining a set of distributed microservices. We argue that applications that require large computational or storage resources can benefit from the microservices approach because the applications can share the underlying compute infrastructure between multiple applications and users. This makes it possible to deploy an application on a lightweight system that uses a common service for computation and storage. In addition, benefits such as using different programming languages for a single application, and packaging a microservice as a software container, help to outweigh the operational burden related to using microservices to build applications.

Of the related systems, OpenCPU provides the most similar interface to analyze datasets as the R interface in Kvik. While we started to explore OpenCPU for use

in our applications, we found through our benchmarking that it did not provide satisfactory performance for our applications. It does however provide a richer set of functionality, such as exporting data in many more formats and running user-submitted scripts. We did not find it necessary for these additions and implemented our own R interface that could provide the necessary interface for us to implement data exploration applications.

We have reused the microservices for running statistical analyses and fetch biological metadata, and share these between applications. This makes it possible for multiple applications to use one or more powerful servers for hosting the services. In the case of statistical analyses we simply install the necessary R packages for each application on the compute service and run it as we would for one single application.

3.9 Future Work

We hope to continue development on applications for interactively exploring biological datasets. Through our approach, and especially the interface to R, we are now able to develop applications that can use any function or retrieve datasets from any R package. This includes the `nowac` package in Chapter 2. We believe that there is a large potential in the available datasets, and that researchers would benefit from being able to interactively explore these.

3.9.1 MIxT

We intend to address few points in future work, both in the MIxT web application as well as the supporting microservices. The first issue is to improve the user experience in the MIxT web application. Since it is executing many of the analyses on demand, the user interface may seem unresponsive. We are working on mechanisms that gives the user feedback when the computations are taking a long time, but also reducing analysis time by improving the performance the underlying R package. The database service provides a sufficient interface for the MIxT web application. While we have developed the software packages for interfacing with more databases, these haven't been included in the database service yet. In future versions we aim to make the database service an interface for all our applications. We also aim to improve how we capture data provenance. We aim to provide database versions and meta-data about when a specific item was retrieved from the database.

One large concern that we haven't addressed in this chapter is security. In particular one security concern that we aim to address in Kvik is the restrictions

on the execution of code in the compute service. We aim to address this in the next version of the compute service, using methods such as AppArmor[76] that can restrict a program's resource access. In addition to code security we will address data access, specifically put constraints on who can access data from the compute service. We also aim to explore different alternatives for scaling up the compute service. Since we already interface with R we can use the Sparklyr[77] or SparkR[78] packages to run analyses on top of Spark.[79] Using Spark as an execution engine for data analyses will enable applications to explore even larger datasets.

3.10 Conclusion

We have designed an approach for building data exploration applications in cancer research. We first implemented Kvik Pathways, a web application for exploring a gene expression dataset in the context of pathway maps. We used our experiences to generalize our efforts into a set of central components that these types of applications require. Further we realized these in our SME approach implemented as a set of microservices. Using these services we have built a web application, MIxT, that integrates statistical analyses, interactive visualizations, and data from biological databases. While we have used our approach to build an application in cancer research, we believe that the microservice architecture can be used to build data exploration systems in other disciplines as well.

In summary, our primary lesson learned from the experiences with develop our two applications, is to compose and develop a data exploration system from independent parts. We chose to implement our systems using three separate services. A compute service to provide statistical analyses, a database service to provide access to biological databases, and the user interface. This makes it possible to quickly re-implement parts of the system, but also allow others to interface with its underlying components, not just the user interface.

4

Deep Analysis Pipelines

In this chapter we discuss our approach to analyzing high-throughput genomic datasets through deep analysis pipelines, and its implementation in walrus.[27] We also evaluate the performance of walrus and show its usefulness in a precision medicine setting. While walrus was developed in this context we also show its usefulness in other areas, specifically for RNA-seq analyses.

4.1 Use Case and Motivation

Precision medicine uses patient-specific molecular information to diagnose and categorize disease to tailor treatment to improve health outcome.[39] Important goals in precision medicine are to learn about the variability of the molecular characteristics of individual tumors, their relationship to outcome, and to improve diagnosis and therapy.[40] Cancer institutions are therefore now offering dedicated personalized medicine programs.

For cancer, high throughput sequencing is an emerging technology to facilitate personalized diagnosis and treatment since it enables collecting high quality genomic data from patients at a low cost. Data collection is becoming cheaper, but the downstream computational analysis is still time-consuming and thereby a costly part of the experiment. This is because of the manual efforts to set up, analyze, and maintain the analysis pipelines. These pipelines consist of many steps that transform raw data into interpretable results.[24] These

pipelines often consists of in-house or third party tools and scripts that each transform input files and produce some output. Although different tools exist, it is necessary to carefully explore different tools and parameters to choose the most efficient to apply for a dedicated question.[80] The complexity of the tools vary from toolkits such as the Genome Analysis Toolkit (GATK) to small custom `bash` or `R` scripts. In addition, some tools interface with databases whose versions and content will impact the overall result.[81]

Improperly developed analysis pipelines for precision medicine may generate inaccurate results, which may have negative consequences for patient care.[1] Users and clinicians therefore need systems that can track pipeline tool versions, their input parameters, and data. Both to thoroughly document what produced the final clinical reports, and to iteratively improve the quality of the pipeline during development. Because of the iterative process of developing the analysis pipeline, it is necessary to use analysis tools that facilitate modifying pipeline steps and adding new ones with little developer effort.

Developing a system that enables researchers to write and share reproducible analysis pipelines will enable the scientific community to analyze high-throughput genomic datasets faster and more unified. By combining versioning of datasets and pipeline configurations, a pipeline management system will provide interpretable and reproducible results long after the initial data analysis will have completed. These features will together promote reproducible science and improve the overall quality of the analyses.

4.1.1 Initial Data Analysis Pipeline

We have analyzed DNA sequence data from a breast cancer patient's primary tumor and adjacent normal cells to identify the molecular signature of the patient's tumor and germline. When the patient later relapsed we analyzed sequence data from the patient's metastasis to provide an extensive comparison against the primary and to identify the molecular drivers of the patient's tumor.

We used whole-genome sequencing (WGS) to sequence the primary tumor and adjacent normal cells at an average depth of 20, and whole-exome sequencing (WES) at an average depth of 300. The biological samples were sequenced at the Genome Quebec Innovation Centre, and we stored the raw datasets on our in-house server. From the analysis pipelines we generated reports with end results, such as detected somatic mutations, that was distributed to both the patient and the treating oncologists. These could be used to guide diagnosis and treatment, and give more detailed insight into both the primary and metastasis. When the patient relapsed we analyzed WES data using our own pipeline

manager, `walrus`, to investigate the metastasis and compare it to the primary tumor.

For the initial WGS analysis we developed a pipeline to investigate somatic and germline mutations based on Broad Institute's best practices. We developed the analysis pipeline on our in-house compute server using a *bash* script under version control with *git* to track changes as we developed the analysis pipeline. The pipeline consisted of tools including picard[82], fastqc[83], trimmomatic[84], and the GATK.[85] While the analysis tools themselves provide the necessary functionality to give insights in the disease, ensuring that the analyses could be fully reproduced later left areas in need of improvement.

We chose a command-line script over more complex pipelining tools or workbenches such as Galaxy[86] because of its fast setup time on our available compute infrastructure, and familiar interface. More complex systems could be beneficial in larger research groups with more resources to compute infrastructure maintenance, whereas command-line scripting languages require little infrastructure maintenance over normal use. In addition, while there are offsite solutions for executing scientific workflows, analyzing sensitive data often put hard restrictions on where the data can be stored and analyzed.

After we completed the first round of analyses we summarized our efforts and noted features that pipeline management systems should satisfy:

- Datasets and databases should be under version control and stored along with the pipeline description. In the analysis script we referenced to datasets and databases by their physical location on a storage system, but these were later moved without updating the pipeline description causing extra work. A solution would be to add the data to the same version control repository hosting the pipeline description.

- The specific pipeline tools should also be kept available for later use. Often in bioinformatics, just installing a tool is a time-consuming process because of their many dependencies.

- It should be easy to add new tools to an existing pipeline and execution environment. This includes installing the specific tool and adding to an existing pipeline. Bundling tools within software containers, such as Docker, and hosting them on an online registry simplifies the tool installation process since the only requirement is the container runtime.

- While bash scripts have their limitations, using a well-known format that closely resembles the normal command-line use clearly have its advantages. It is easy to understand what tools were used, their input

parameters, and the data flow. However, from our experience when these analysis scripts grow too large they become too complex to modify and maintain.

- While there are new and promising state-of-the art pipeline managers, many of these also require state-of-the-art computing infrastructure to run. This may not be the case at cancer research and clinical institutions.

The above problem areas are not just applicable to our research group, but common to other research and precision medicine projects as well. Especially when hospitals and research groups aim to apply personalized medicine efforts to guide therapeutic strategies and diagnosis, the analyses will have to be able to be easily reproducible later. We used the lessons learned to design and implement `walrus`, a command line tool for developing and running data analysis pipelines. It automatically orchestrates the execution of different tools, and tracks tool versions and parameters, as well as datasets through the analysis pipeline. It provides users a simple interface to inspect differences in pipeline runs, and retrieve previous analysis results and configurations. In the remainder of the chapter we describe the design and implementation of `walrus`, its clinical use, its performance, and how it relates to other pipeline managers.

4.2 `walrus`

`walrus` is a tool for developing and executing data analysis pipelines. It stores information about tool versions, tool parameters, input data, intermediate data, output data, as well as execution environments to simplify the process of reproducing data analyses. Users write descriptions of their analysis pipelines using a familiar syntax and `walrus` uses this description to orchestrate the execution of the pipeline. In `walrus` we package all tools in software containers to capture the details of the different execution environments. While we have used `walrus` to analyze high-throughput datasets in precision medicine, it is a general tool that can analyze any type of data, e.g. image datasets for machine learning. It has few dependencies and runs on any platform that supports Docker containers. While other popular pipeline managers require the use of cluster computers or cloud environment, we focus on single compute node systems often found in smaller clinical research environments.

`walrus` is implemented as a command-line tool in the Go programming language. We use the popular software container implementation Docker[87] to provide reproducible execution environments, and interface with git together with `git-lfs`[48] to version control datasets and pipeline descriptions. By

choosing Docker and git we have built a tool that easily integrates with current bioinformatic tools and workflows. It runs both natively or within its own Docker container to simplify its installation process.

With `walrus` we target pipeline developers that use command-line tools and scripting languages to build and run analysis pipelines. Users can use existing Docker containers from sources such as BioContainers[88] or build containers with their own tools. We integrate with the current workflow using git to version control analysis scripts, and use `git-lfs` for versioning of datasets as well. We have designed the pipeline description format resembles the command line syntax as much as possible. This is one of the major strengths of walrus. It uses a familiar syntax and format, and does not require the users to explicitly declare which files in the pipeline to version control.

4.2.1 Pipeline Configuration

Users configure analysis pipelines by writing pipeline description files in a human readable format such as JavaScript Object Notation (JSON) or YAML Ain't Markup Language (YAML). A pipeline description contains a list of stages, each with inputs and outputs, along with optional information such as comments or configuration parameters such as caching rules for intermediate results. Listing 4.1 shows an example pipeline stage that uses MuTect[89] to detect somatic point mutations. Users can also specify the tool versions by selecting a specific Docker image, for example using MuTect version 1.1.7 as in Listing 4.1, line 3.

Users specify the flow of data in the pipeline within the pipeline description, as well as the dependencies between the steps. Since pipeline configurations can become complex, users can view their pipelines using an interactive web-based tool, or export their pipeline as a DOT file for visualization in tools such as Graphviz.[90]

Listing 4.1: Example pipeline stage for a tool that detects somatic point mutations. It reads a reference sequence file together with both tumor and normal sequences, and produces an output file with the detected mutations.

```
{
  "Name": "mutect",
  "Image": "fjukstad/mutect:1.1.7",
  "Cmd": [
    "--analysis_type","MuTect",
    "--reference_sequence","/walrus/input/reference.fasta",
    "--input_file:normal","/walrus/input/normal.bam",
    "--input_file:tumor","/walrus/input/tumor.bam",
    "-L","/walrus/input/targets.bed",
    "--out","/walrus/mutect/mutect-stats-txt",
    "--vcf","/walrus/mutect/mutect.vcf"
  ],
```

```
    "Inputs":[
       "input"
    ]
}
```

Users add data to an analysis pipeline by specifying the location of the input data in the pipeline description, and `walrus` automatically mounts it to the container running the analysis. The location of the input files can either be local or remote locations such as an FTP server. When the pipeline is completed, `walrus` will store all the input, intermediate and output data to a user-specified location which is under version control.

4.2.2 Pipeline Execution

When users have written a pipeline description for their analyses, they can use the command-line interface of `walrus` to run the analysis pipeline. `walrus` builds an execution plan from the pipeline description and runs it for the user. It uses the input and output fields of each pipeline stage to construct a directed acyclic graph (DAG) where each node is a pipeline stage and the links are input/output data to the stages. From this graph `walrus` can determine parallelizable stages and coordinate the execution of the pipeline.

In `walrus`, each pipeline stage is run in a separate container, and users can specify container versions in the pipeline description to specify the correct version of a tool. We treat a container as a single executable and users specify tool input arguments in the pipeline description file using standard command line syntax. `walrus` will automatically build or download the container images with the analysis tools, and start these with the user-defined input parameters and mount the appropriate input datasets. While the pipeline is running, `walrus` monitors running stages and schedules the execution of subsequent pipeline stages when their respective input data become available. We have designed `walrus` to execute an analysis pipeline on a single large server, but since the tools are run within containers, these can easily be orchestrated across a range of servers in future versions.

Users can select from containers pre-installed with bioinformatics tools, or build their own using a standard Dockerfile. Through software containers `walrus` can provide a reproducible execution environment for the pipeline, and containers provide simple execution on a wide range of software and hardware platforms. With initiatives such as BioContainers, researchers can make use of already existing containers without having to re-write their own. Data in each pipeline step is automatically mounted and made available within each Docker container. By simply relying on Docker `walrus` requires little

software setup to run different bioinformatics tools.

While `walrus` executes a single pipeline on one physical server, it supports both data and tool parallelism, as well as any parallelization strategies within each tool, e.g. multi-threading. To enable data and tool parallelism, e.g. run the same analyses to analyse a set of samples, users list the samples in the pipeline description and `walrus` will automatically run each sample through the pipeline in parallel. While we can parallelize the independent pipeline steps, the performance of an analysis pipeline relies on each of the independent tools and available compute power. Techniques such as multithreading can improve the performance of a tool, and `walrus` users can make use of these techniques if their are available through the command line interfaces of the tools.

Upon successful completion of a pipeline run, `walrus` will write a verbose pipeline description file to the output directory. This file contains information on the runtime of each step, which steps were parallelized, and provenance related information to the output data from each step. Users can investigate this file to get a more detailed look on the completed pipeline. In addition to this output file `walrus` will return a unique version ID for the pipeline run, which later can be used to investigate a previous pipeline run.

4.2.3 Data Management

In `walrus` we provide an interface for users to track their analysis data through a version control system. This allows users to inspect data from previous pipeline runs without having to recompute all the data. `walrus` stores all intermediate and output data in an output directory specified by the user, which is under version control automatically by `walrus` when new data is produced by the pipeline. We track changes at file granularity.

In `walrus` we interface with `git` to track any output file from the analysis pipeline. When users execute a pipeline, `walrus` will automatically add and commit output data to a git repository using `git-lfs`. Users typically use a single repository per pipeline, but can share the same repository to version multiple pipelines as well. With `git-lfs`, instead of writing large blobs to a repository it writes small pointer files that contains the hash of the original file, the size of the file, and the version of`git-lfs` used. The files themselves are stored separately which makes the size of the repository small and manageable with git. Once `walrus` has started to track output datasets, users can use regular git commands to inspect its version history. The main reason why we chose git and `git-lfs` for version control is that git is the de facto standard for versioning source code, and we want to include versioning of datasets without altering the typical development workflow.

Since we are working with potentially sensitive datasets `walrus` is targeted at users that use a local compute and storage servers. It is up to users to configure a remote tracker for their repositories, but we provide command-line functionality in `walrus` to run a `git-lfs` server that can store users' contents. They can use their default remotes, such as Github, for hosting source code, but they must themselves provide the remote server to host their data.

4.2.4 Pipeline Reconfiguration and Re-execution

Reconfiguring a pipeline is common practice in precision medicine, e.g. to ensure that genomic variants are called with a desired sensitivity and specificity. To reconfigure an existing pipeline users make the applicable changes to the pipeline description and re-run it with `walrus`. `walrus` will then recompute the necessary steps and return a version ID for the newly run pipeline. This ID can be used to compare pipeline runs, the changes made, and optionally restore the data and configuration from a previous run. Reconfiguring the pipeline to use updated tools or reference genomes will alter the pipeline configuration and force `walrus` to recompute the applicable pipeline stages.

The command-line interface of `walrus` provides functionality to restore results from a previous run, as well as printing information about a completed pipeline. To restore a previous pipeline run, users use the `restore` command line flag in `walrus` together with the version ID of the respective pipeline run. `walrus` will interface with git to restore the files to their state at the necessary point in time.

4.3 Results

To evaluate the usefulness of `walrus` we demonstrate its use in a clinical research setting, and the low computational time and storage overhead to support reproducible analyses.

4.3.1 Clinical Application

We have used `walrus` to analyze a whole-exome data from a sample in the McGill Genome Quebec [MGGQ] dataset (GSE58644)[28] to discover SNPs, genomic variants and somatic mutations. We interactively developed a pipeline description that follows the best-practices of The Broad Institute[1] and generated

1. Online at `software.broadinstitute.org/gatk/best-practices`.

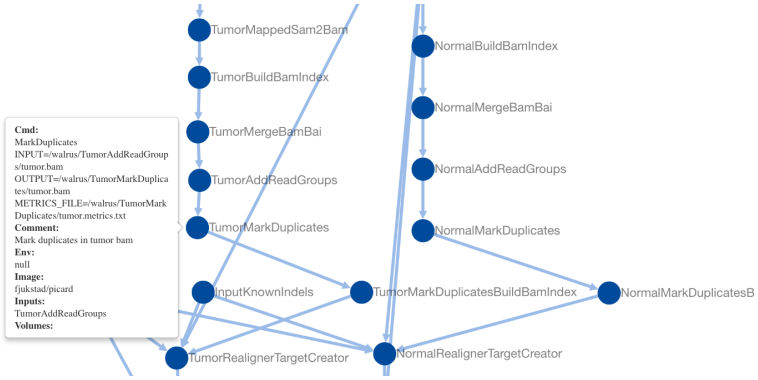


Figure 4.1: Screenshot of the web-based visualization in `walrus`. The user has zoomed in to inspect the pipeline step which marks duplicate reads in the tumor sequence data.

reports that summarized the findings to share the results. Figure 4.1 shows a screenshot from the web-based visualization in `walrus` of the pipeline.

From the analyses we discovered inherited germline mutations that are recognized to be among the top 50 mutations associated with an increased risk of familial breast cancer. We also discovered a germline deletion which has been associated with an increased risk of breast cancer. We also discovered mutations in a specific gene that might explain why specific drug had not been effective in the treatment of the primary tumor. From the profile of the primary tumor we discovered many somatic events (around 30 000) across the whole genome with about 1000 in coding regions, and 500 of these were coding for non-synonymous mutations. We did not see amplification or constituent activation of growth factors like HER2, EGFR or other players in breast cancer. Because of the germline mutation, early recurrence, and lack of DNA events, we suspect that the patient's primary tumor was highly immunogenic. We have also identified several mutations and copy number changes in key driver genes. This includes a mutation in a gene that creates a premature stop codon, truncating one copy of the gene.

While we cannot share the results in details or the sensitive dataset, we have made the pipeline description available at `github.com/uit-bdps/walrus` along with other example pipelines.

4.3.2 Example Dataset

To demonstrate the performance of `walrus` and the ability to track and detect changes in an analysis pipeline, we have implemented one of the variant calling

pipelines from [91] using tools from picard and the GATK. We show the storage and computational overhead of our approach, and the benefit of capturing the pipeline specification using a pipeline manager. The pipeline description and code is available along with `walrus` at `github.com/uit-bdps/walrus`. Figure 4.2 shows a simple graphical representation of the pipeline.

4.3.3 Performance and Resource Usage

We first run the variant calling pipeline without any additional provenance tracking or storing of output or intermediate datasets. This is to get a baseline performance measurement for how long we expect the pipeline to run. We then run a second experiment to measure the overhead of versioning output and intermediate data. Then we introduce a parameter change in one of the pipeline steps which results in new intermediate and output datasets. Specifically we change the `–maxReadsForRealignment` parameter in the indel realigner step back to its default (See the online pipeline description for more details). This forces `walrus` to recompute the indel realigner step and any subsequent steps. To illustrate how `walrus` can restore old pipeline configurations and results, we restore the pipeline to the initial configuration and results. We show the computational overhead and storage usage of restoring a previous pipeline configuration.

Reproducing results from a scientific publication can be a difficult task. For example, because the rendering of the online version of the pipeline in [91] converts two consecutive hypens (-) into single em dashes (—), the pipeline will not run using the specified input parameters. However, PDF versions of the paper lists the parameters correctly. In addition, the input filenames in the variant calling step do not correspond to any output files in previous steps, but because of their similarity to previous output files we assume that this is just a typo. These issues in addition to missing commands for e.g. the filtering step highlights the clear benefit of writing and reporting the analysis pipeline using a tool such as `walrus`.

Table 4.1 shows the runtime and storage use of the different experiments. In the second experiment we can see the added overhead of adding version control to the dataset. In total, an hour is added to the runtime and the data size is doubled. The doubling comes from git-lfs hard copying the data into a subdirectory of the `.git` folder in the repository. With git-lfs users can move all datasets to a remote server reducing the local storage requirements. In the third experiment we can see that only the downstream analyses from configuring the indel realignment parameter is executed. It generates 30GB of additional data, but the execution time is limited to the applicable stages. Restoring the pipeline to a previous configuration is almost instantaneous since the data is

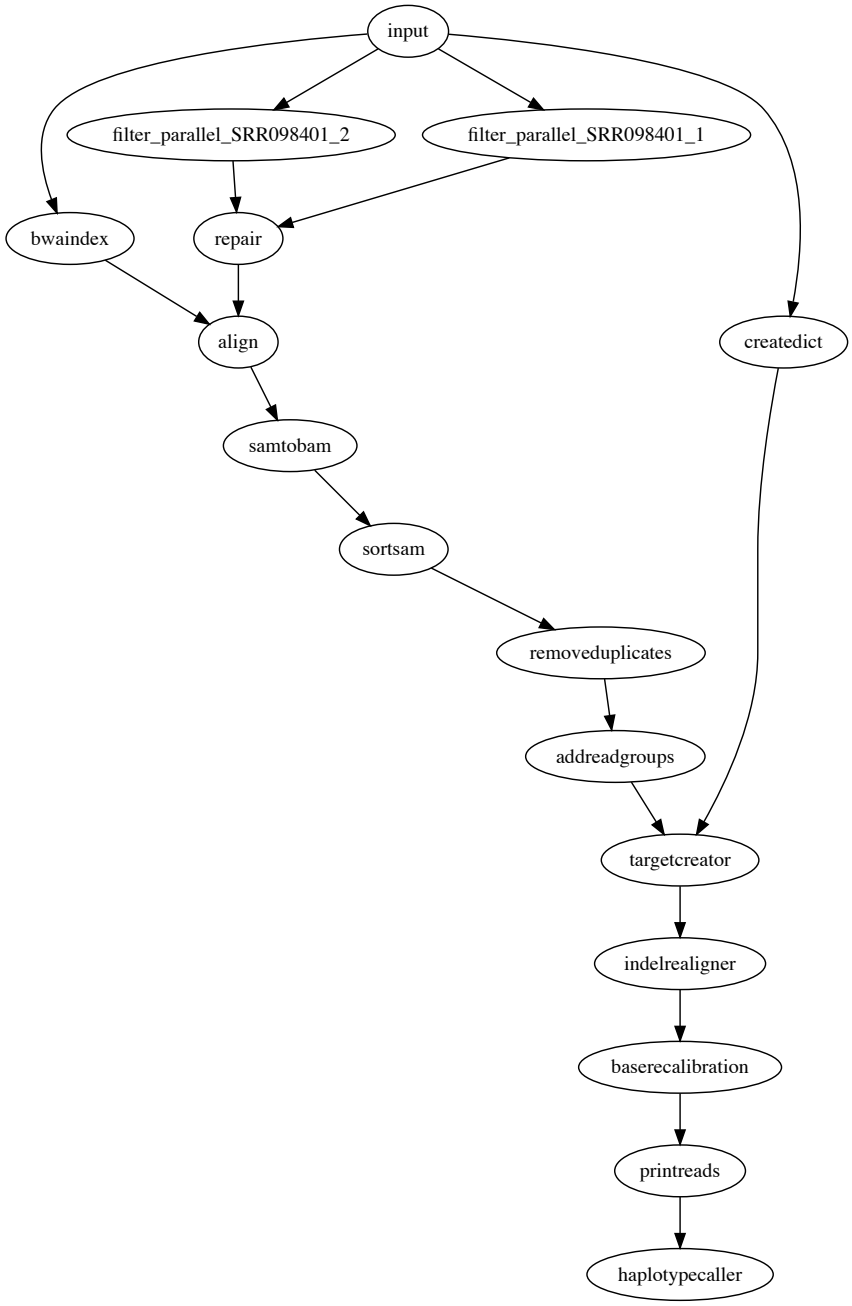


Figure 4.2: In addition to the web-based inteactive pipeline visualization, `walrus` can also generate DOT representations of pipelines. The figure shows the example variant calling pipeline we used in the performance evaluation.

already available locally and git only has to modify the pointers to the correct files in the `.git` subdirectory.

Table 4.1: Runtime and storage use of the example variant-calling pipeline developed with `walrus`.

Experiment	Task	Runtime	Storage Use
1	Run pipeline with default configuration	21 hours 50 minutes	235 GB
2	Run the default pipeline with version control of data	23 hours 9 minutes	470 GB
3	Re-run the pipeline with modified indel realignment parameter	13 hours	500 GB
4	Restoring pipeline back to the default configuration	< 1 second	500GB

4.4 Related Work

There are a wealth of pipeline specification formats and workflow managers available. Some are targeted at users with programming experience while others provide simple GUIs.

We have previously conducted a survey of different specialized bioinformatics pipelines.[29] The pipelines were selected to show how analysis pipelines for different applications use different technologies for configuring, executing and storing intermediate and output data. In the review, we targeted specialized analysis pipelines that support scaling out the pipelines to run on high-performance computing (HPC) or cloud computing platforms.

Here we describe general systems for developing data analysis pipelines, not just specialized bioinformatics pipelines. While most provide viable options for genomic analyses, we have found many of these pipeline systems require complex compute infrastructure beyond the smaller clinical research institutions. We discuss tools that use the common CWL pipeline specification and systems that provide versioning of data.

CWL is a specification for describing analysis workflows and tools.[6] A pipeline is written as a JSON or YAML file, or a mix of the two, and describes each step in detail, e.g. what tool to run, its input parameters, input data and output data.

The pipeline descriptions are text files that can be under version control and shared between projects. There are multiple implementations of CWL workflow platforms, e.g. the reference implementation cwl_runner[6], Arvados[92], Rabix[93], Toil[17], Galaxy[86], and AWE.[94] It is no requirement to run tools within containers, but implementations can support it. There are few of these tools that support versioning of the data. Galaxy is an open web-based platform for reproducible analysis of large high-throughput datasets.[86] It is possible to run Galaxy on local compute clusters, in the cloud, or using the online Galaxy site.[2] In Galaxy users set up an analysis pipeline using a web-based graphical interface, and it is also possible to export or import an existing workflow to an Extensible Markup Language (XML) file.[3] We chose not to use Galaxy because of missing command-line and scripting support, along with little support for running workflows with different configurations.[3] Rabix provides checksums of output data to verify it against the actual output from the pipeline. This is similar to the checksums found in the git-lfs pointer files, but they do not store the original files for later. An interesting project that uses CWL in production is The Cancer Genomics Cloud[95]. They currently support CWL version 1.0 and are planning on integrating Rabix as its CWL executor. Arvados stores the data in a distributed storage system, Keep, that provides both storage and versioning of data. We chose not to use CWL and its implementations because of its relaxed restrictions on having to use containers, its verbose pipeline descriptions, and the complex compute architecture required for some implementations. We are however experimenting with an extension to `walrus` that translates pipeline descriptions written in `walrus` to CWL pipeline descriptions.

Pachyderm is a system for running big data analysis pipelines. It provides complete version control for data and leverages the container ecosystem to provide reproducible data processing.[5] Pachyderm consists of a file system (Pachyderm File System (PFS)) and a processing system (Pachyderm Processing System (PPS)). PFS is a file system with git-like semantics for storing data used in data analysis pipelines. Pachyderm ensures complete analysis reproducibility by providing version control for datasets in addition to the containerized execution environments. Both PFS and PPS is implemented on top of Kubernetes.[96] There are now recent efforts to develop bioinformatics workflows with Pachyderm that show great promise. In [97], the authors show the potential performance improvements of single workflow steps, not the full pipeline, when executing a pipeline in Pachyderm. They unfortunately do not show the time to import data into PFS, run the full pipeline, and optionally investigate different versions of the intermediate, or output datasets.

2. Available at `usegalaxy.org`.
3. An alpha version of Galaxy with CWL support is available at `github.com/common-workflow-language/galaxy`.

We believe that the approach in Pachyderm with version controlling datasets and containerizing each pipeline step is, along with walrus, the correct approach to truly reproducible data analysis pipelines. The reason we did not use Kubernetes and Pachyderm was because our compute infrastructure did not support it. In addition, we did not want to use a separate tool, PFS, for data versioning, we wanted to integrate it with our current practice of using git for versioning.

Snakemake is a long-running project for analyzing bioinformatic datasets.[16] It uses a Python-based language to describe pipelines, similar to the familiar Makefile syntax, and can execute these pipelines on local machines, compute clusters or in the cloud. To ensure reproducible workflows, Snakemake integrates with Bioconda to provide the correct versions of the different tools used in the workflows. It integrates with Docker and Singularity containers[98] to provide isolated execution, and in later versions Snakemake allows pipeline execution on a Kubernetes cluster. Because Snakemake did not provide necessary integration with software containers at the time we developing our analysis pipeline, we did not find it to be a viable alternative. For example, support for pipelines consisting of Docker containers pre-installed with bioinformatics tools came a year later than walrus.

Another alternative to develop analysis pipelines is Nextflow.[99] Nextflow uses its own language to describe analysis pipelines and supports execution within Docker and Singularity containers. Nextflow uses a dataflow programming model that streams data through a pipeline as apposed to fist constructing a DAG and executing it.

While the previous related systems all package each tool into a single container, Bio-Docklet and elasticHPC are systems that bundle entire pipelines into single Docker containers. Bio-Docklets are standardized workflows contained in a single Docker image, and have been used used to build NGS analysis pipelines.[100] elasticHPC is an initiative to make it easier to deploy containerized analysis pipeline on private or commercial cloud solutions such as Amazon.[101]

As discussed in [30, 29], recent projects propose to use containers for life science research. The BioContainers and Bioboxes[102] projects address the challenge of installing bioinformatics data analysis tools by maintaining a repository of Docker containers for commonly used data analysis tools. Docker containers are shown to have better than, or equal performance as Virtual Machines (VMs), and introduce negligible overhead opposed to executing on bare metal.[103] While Docker containers require a bootstrapping phase before executing any code, this phase is negligible in the compute-intensive precision medicine pipelines that run for several hours. Containers have also been proposed as

a solution to improve experiment reproducibility, by ensuring that the data analysis tools are installed with the same responsibilities.[104]

4.5 Discussion

`walrus` is a general tool for analyzing any type of dataset from different scientific disciplines, not just genomic datasets in bioinformatics. Users specify a workflow using either a YAML or JSON format, and each step in the workflow is run within a Docker container. `walrus` tracks input, intermediate, and output datasets with git to ensure transparency and reproducibility of the analyses. Through these features `walrus` helps to ensure repeatability of the computation analyses of a research project.

Precision medicine requires flexible analysis pipelines that allow researchers to explore different tools and parameters to analyze their data. While there are best practices to develop analysis pipelines for genomic datasets, e.g. to discover genomic variants, there is still no de-facto standard for sharing the detailed descriptions to simplify re-using and reproducing existing work. With `walrus` we provide one alternative to develop and share pipeline descriptions.

Pipelines typically need to be tailored to fit each project and patient, and different patients will typically elicit different molecular patterns that require individual investigation. In our WES analysis pipeline we followed the best practices, and explored different combinations of tools and parameters before we arrived at the final analysis pipeline. For example, we ran several rounds of preprocessing (trimming reads and quality control) before we were sure that the data was ready for analysis. `walrus` allowed us to keep track of different intermediate datasets, along with the pipeline specification, simplifies the task of comparing the results from pipeline tools and input parameters.

`walrus` is a very simple tool to set up and start using. Since we only target users with single large compute nodes, `walrus` can run within a Docker container making Docker its only dependency. Systems such as Nextflow, Galaxy or Pachyderm all require users to set up and manage complex compute infrastructures. As previously mentioned, since we leverage existing Docker images without any modification in `walrus`, users can reuse existing container images from BioContainers or Bioboxes in their workflows. The simplicity of `walrus` enables repeatable computational analyses without any of these obstacles, and is one of the strengths of our tool.

Unlike other proposed solutions for executing data analysis pipelines, `walrus` is the only system we have discovered that explicitly uses git, and git-lfs, to store

output datasets. Other systems either use a specialized storage system, or ignore data versioning at all. We believe that using a system that bioinformaticians already use for source control management is the simplest way to allow users version their data along-side their analysis code. The alternative of using a new data storage platform that provides data versioning requires extra time and effort for researchers both to learn and integrate in their current workflow.

We have seen that there are other systems to develop, share, and run analysis pipelines in both bioinformatics and other disciplines. Like `walrus`, many of these use textual representations in JSON or other languages to describe the analysis pipeline, and Docker to provide reproducible and isolated execution environments. In `warlus` we provide pipeline descriptions that allows users to reuse the familiar command-line syntax. The only new additional information they have to add is the dependencies between tasks. Systems such as CWL requires that users also describe the input and output data verbosely. We believe that the tool, `walrus`, can detect these, and will handle this for the user. This will in turn make the pipeline descriptions of `walrus` shorter in terms of lines of code.

While systems such as Galaxy provide a graphical user interface, `walrus` requires that its users know how to navigate the commandline and have experience with systems such as git and Docker, to analyze a dataset. Using a commandline interface to run analysis pipelines has the potential of speeding up the analysis process, since its users do not have to click through a user interface before running a pipeline.

We have tried to minimize the number of available commands in `walrus`, and compared to other tools it shows its benefit when comparing a pipeline run to previous results. E.g. in Pachyderm users have to explicitly import data into the system using a set of commands. `walrus` does not require explicit import of data, and allows users to investigate, or roll back, data to a previous run in a single command.

While we provide one approach to version control datasets, there are still some drawbacks. `git-lfs` supports large files, but in our results it added 5% in runtime. This makes the entire analysis pipeline slower, but we argue that having the files under version control outweigh the runtime. In addition, there are only a few public `gif-lfs` hosting platforms for datasets larger than a few gigabytes, making it necessary to host these in-house. In-house hosting may also be a requirement at different medical institutions.

We aim to investigate the performance of running analysis pipelines with `walrus`, and the potential benefit of its built-in data parallelism. While our

WES analysis pipeline successfully run steps in parallel for the tumor and adjacent normal tissue, we have not demonstrated the benefit of doing so. This includes benchmarking and analyzing the system requirements for doing precision medicine analyses. We are also planning on exploring parallelism strategies where we can split an input dataset into chromosomes and run some steps in parallel for each chromosome, before merging the data again.

We believe that future data analysis systems for precision medicine will follow the lines of our proposed approach. Software container solutions provide valuable information in the reporting of the analyses, and they impose little performance overhead. Further, the development of container orchestration systems such as Kubernetes is getting wide adoption nowadays, especially in web-scale internet companies. However, the adoption of such systems in a clinical setting depend on support from more tools, and also the addition of new compute infrastructure.

4.6 Conclusions

We have designed and implemented `walrus`, a tool for developing reproducible data analysis pipelines for use in precision medicine. Precision medicine requires that analyses are run on hospital compute infrastructures and results are fully reproducible. By packaging analysis tools in software containers, and tracking both intermediate and output data, `walrus` provides the foundation for reproducible data analyses in the clinical setting. We have used `walrus` to analyze a patient's metastatic lesions and adjacent normal tissue to provide insights and recommendations for cancer treatment.

5

Conclusion

How should we design systems for analyzing and exploring the high-throughput datasets that facilitate sharing, reuse, and reproducibility? This dissertation shows that in many cases the solution is to decompose the applications into small entities that communicate using open protocols. This enables the development of unified systems for reproducible exploration and analysis.

While high-throughput datasets and computing systems will undoubtedly evolve, we believe that the SME approach proposed here can offer a new perspective on developing applications for exploring and analyzing biological data. We hope that our approach can steer the development of bioinformatics applications away from large monolithic applications to applications composed of diverse systems. We believe that this approach can help the community develop new systems to faster meet the needs of the upcoming biological dataset analyses.

In Chapter 2 we show an approach to store the microarray data and analysis code from a complex epidemiological study in a shareable software package. We show how we explicitly track versions of code and data, and how we can generate reproducible data analysis reports for the processed datasets. We believe that future studies can benefit from applying our approach, and that future advances in cancer research is dependent on sharing of both datasets and analysis code. In chapter 3 we show how we can build interactive data exploration applications that interface with these software packages through a microservice architecture. We have implemented this approach through the

microservices in *Kvik*. We show that this architecture style is suitable for building such applications, and have used it to develop the *Kvik Pathways* and *MIxT* web applications. These have been successfully used to explore transcriptional profiles in the NOWAC study, especially to investigate the interactions between genes and pathways in the patient tumor and blood cells. We believe that the cancer research community in general will benefit greatly if more projects start to develop their applications using our approach. It simplifies sharing of computational resources, and we believe that the future of cancer research will depend on collaborative efforts. In chapter 4 use the same approach, to compose systems of disparate tools, for developing biological data analysis pipelines, implemented in `walrus`. To ensure reproducible results, we supplement the processing with data versioning to track provenance of the data through the pipeline and across pipeline versions. We have used `walrus` in the clinical setting to develop a WES pipeline for discovering SNPs, genomic variants, and somatic mutations, in a breast cancer patient's metastatic lesion.

Combined, these systems demonstrate the applicability of our approach across a range of different use cases.

In the rest of this chapter we summarize end-to-end lessons learned during this work. We then discuss the work in the context of cancer research and in the clinical setting, before we propose areas for future work.

5.1 Lessons Learned

Through the design of the SME approach for analyzing and exploring biological datasets, as well as the different implementations of the approach, we have solved challenges and we habe learned some key lessons.

There is no single solution programming language or system. In the field of bioinformatics there have been tremendous efforts to develop analysis tools for improving the analysis of new biological datasets. This has led to systems being written in a plethora of different languages, and deployed on top of different systems. This is the main motivation behind our SME approach together with software containers.

Take advantage of existing tools. The ability to develop applications for analyzing biological datasets comes from the availability of existing tools. By developing easy-to-use interfaces for the existing tools, it is possible to develop new applications without reimplementing key features.

Simplicity is key. When proposing a new approach for either managing

datasets, writing data exploration applications, or developing analysis pipelines, it is not possible to overstate the importance of the simplicity of the solution.

Researchers are not software engineers. When designing a new approach to store and analyze high-throughout biological datasets, it is important to remember that its users have limited software engineering backgrounds. Especially when the implementation is based on complex systems such as `git`, the learning curve for the system is steep and require training of its users. In our project we have organized workshops in both R and git to get the researchers in the NOWAC study comfortable with these systems to follow our best practices.

5.2 Future Work

As we have discussed in previous chapters, there are some limitations to our approach and its implementations. To summarize these, the main areas for improvement are:

- **Versioning of datasets:** `git` was not designed to version large binary files, such as biological datasets, and it does not provide the required performance or scalability to version the large biological data.

- **Additional evaluation:** while we have shown that the SME approach can be used to develop systems for managing research data, developing interactive applications and data analysis pipelines, we would like to better understand its performance and scalability.

- **Refactoring and test coverage:** while we provide fully implemented solutions for data storage, interactive applications, and data analysis pipelines, they all have areas of improvement with regards to performance, scalability, and robustness.

- **Distributed execution:** while `walrus` orchestrate execution of Docker containers, we do not support the execution of these on multiple compute nodes. Distributing the computation on multiple machines will reduce the execution time if we can share the data across the machines efficiently. We would also like to evaluate the possibility of using an existing container orchestration system, such as Kubernetes, to orchestrate the execution of an analysis pipeline. Many of these already provide functionality for distributed execution of software containers.

- **Wide adoption of a pipeline description format:** we are not the first to propose a new computing standard.[1] We found that the current standards were either too verbose, e.g., CWL, or did not enforce the use of software containers. This led us to our own description format, but we recognize the need for a single open standard, and hope to contribute to its development.

We aim to refine and continue development on our SMEs approach to address these challenges, and that we can inspire a more unified development community in bioinformatics. We believe that the future of cancer research relies on the successful integration of diverse data analysis and data management systems from different research institutions. This will definitely continue to be an interesting area of research.

1. xkcd.com/927

Bibliography

[1] S. Roy, C. Coldren, A. Karunamurthy, N. S. Kip, E. W. Klee, S. E. Lincoln, A. Leon, M. Pullambhatla, R. L. Temple-Smolkin, K. V. Voelkerding *et al.*, "Standards and guidelines for validating next-generation sequencing bioinformatics pipelines: A joint recommendation of the association for molecular pathology and the college of american pathologists," *The Journal of Molecular Diagnostics*, vol. 20, pp. 4–27, 2017.

[2] J. Goecks, A. Nekrutenko, and J. Taylor, "Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences," *Genome biology*, vol. 11, no. 8, p. R86, 2010.

[3] O. Spjuth, E. Bongcam-Rudloff, G. C. Hernández, L. Forer, M. Giovacchini, R. V. Guimera, A. Kallio, E. Korpelainen, M. M. Kańduła, M. Krachunov *et al.*, "Experiences with workflows for automating data-intensive bioinformatics," *Biology direct*, vol. 10, no. 1, p. 43, 2015.

[4] R. C. Gentleman, V. J. Carey, D. M. Bates, B. Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, J. Gentry *et al.*, "Bioconductor: open software development for computational biology and bioinformatics," *Genome biology*, vol. 5, no. 10, p. R80, 2004.

[5] Pachyderm, http://pachyderm.io.

[6] P. Amstutz, M. R. Crusoe, N. Tijanić, B. Chapman, J. Chilton, M. Heuer, A. Kartashov, D. Leehr, H. Ménager, M. Nedeljkovich, and et al., https://figshare.com/articles/Common_Workflow_Language_draft_3/3115156/2, Jul 2016.

[7] Shiny, http://shiny.rstudio.com.

[8] J. Ooms, "The opencpu system: Towards a universal interface for scientific computing through separation of concerns," *arXiv preprint arXiv:1406.4806*, 2014.

[9] E. S. Raymond, *The art of Unix programming*. Addison-Wesley Professional, 2003.

[10] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR, 2005.

[11] I. Nadareishvili, R. Mitra, M. McLarty, and M. Amundsen, *Microservice Architecture: Aligning Principles, Practices, and Culture*. " O'Reilly Media, Inc.", 2016.

[12] S. D. Kahn, "On the future of genomic data," *Science*, vol. 331, no. 6018, pp. 728–729, 2011.

[13] A. Alyass, M. Turcotte, and D. Meyre, "From big data analysis to personalized medicine for all: challenges and opportunities," *BMC medical genomics*, vol. 8, no. 1, p. 33, 2015.

[14] E. R. Mardis, "The $1,000 genome, the 100,000$ analysis?" *Genome medicine*, vol. 2, no. 11, p. 84, 2010.

[15] I. S. for Biocuration, "Biocuration: Distilling data into knowledge," *PLoS biology*, vol. 16, no. 4, p. e2002846, 2018.

[16] J. Köster and S. Rahmann, "Snakemake—a scalable bioinformatics workflow engine," *Bioinformatics*, vol. 28, no. 19, pp. 2520–2522, 2012.

[17] J. Vivian, A. A. Rao, F. A. Nothaft, C. Ketchum, J. Armstrong, A. Novak, J. Pfeil, J. Narkizian, A. D. Deran, A. Musselman-Brown *et al.*, "Toil enables reproducible, open source, big biomedical data analyses," *Nature Biotechnology*, vol. 35, no. 4, pp. 314–316, 2017.

[18] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker, "Cytoscape: a software environment for integrated models of biomolecular interaction networks," *Genome research*, vol. 13, no. 11, pp. 2498–2504, 2003.

[19] M. Krzywinski, J. Schein, I. Birol, J. Connors, R. Gascoyne, D. Horsman, S. J. Jones, and M. A. Marra, "Circos: an information aesthetic for comparative genomics," *Genome research*, vol. 19, no. 9, pp. 1639–1645, 2009.

[20] E. Lund, V. Dumeaux, T. Braaten, A. Hjartåker, D. Engeset, G. Skeie, and M. Kumle, "Cohort profile: the norwegian women and cancer study—nowac—kvinner og kreft," *International journal of epidemiology*,

vol. 37, no. 1, pp. 36–41, 2007.

[21] J. Gómez, L. J. García, G. A. Salazar, J. Villaveces, S. Gore, A. García, M. J. Martín, G. Launay, R. Alcántara, N. Del-Toro *et al.*, "Biojs: an open source javascript framework for biological data visualization," *Bioinformatics*, vol. 29, no. 8, pp. 1103–1104, 2013.

[22] B. Fjukstad, K. S. Olsen, M. Jareid, E. Lund, and L. A. Bongo, "Kvik: three-tier data exploration tools for flexible analysis of genomic data in epidemiological studies," *F1000Research*, vol. 4, 2015.

[23] B. Fjukstad, V. Dumeaux, K. S. Olsen, E. Lund, M. Hallett, and L. A. Bongo, "Building applications for interactive data exploration in systems biology," in *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*. ACM, 2017, pp. 556–561.

[24] Y. Diao, A. Roy, and T. Bloom, "Building highly-optimized, low-latency pipelines for genomic data analysis." in *Conference on Innovative Data Systems Research (CIDR)*, 2015.

[25] K. S. Olsen, C. Fenton, L. Frøyland, M. Waaseth, R. H. Paulssen, and E. Lund, "Plasma fatty acid ratios affect blood gene expression profiles-a cross-sectional study of the norwegian women and cancer post-genome cohort," *PLoS One*, vol. 8, no. 6, p. e67270, 2013.

[26] V. Dumeaux, B. Fjukstad, H. E. Fjosne, J.-O. Frantzen, M. M. Holmen, E. Rodegerdts, E. Schlichting, A.-L. Børresen-Dale, L. A. Bongo, E. Lund *et al.*, "Interactions between the tumor and the blood systemic response of breast cancer patients," *PLoS Computational Biology*, vol. 13, no. 9, p. e1005680, 2017.

[27] B. Fjukstad, V. Dumeaux, M. Hallett, and L. A. Bongo, "Reproducible data analysis pipelines for precision medicine," To appear in the proceedings of 2019 27th Euromicro International Conference On Parallel, Distributed and Network-based Processing (PDP). IEEE, 2019.

[28] A. Tofigh, M. Suderman, E. R. Paquet, J. Livingstone, N. Bertos, S. M. Saleh, H. Zhao, M. Souleimanova, S. Cory, R. Lesurf *et al.*, "The prognostic ease and difficulty of invasive breast carcinoma," *Cell reports*, vol. 9, no. 1, pp. 129–142, 2014.

[29] B. Fjukstad and L. A. Bongo, "A review of scalable bioinformatics pipelines," *Data Science and Engineering*, vol. 2, no. 3, pp. 245–251, 2017.

[30] I. A. Raknes, B. Fjukstad, and L. Bongo, "nsroot: Minimalist process isolation tool implemented with linux namespaces," *Norsk Informatikkonferanse*, 2017.

[31] Y. Kiselev, S. Andersen, C. Johannessen, B. Fjukstad, K. S. Olsen, H. Stenvold, S. Al-Saad, T. Donnem, E. Richardsen, R. M. Bremnes *et al.*, "Transcription factor pax6 as a novel prognostic factor and putative tumour suppressor in non-small cell lung cancer," *Scientific reports*, vol. 8, no. 1, p. 5059, 2018.

[32] B. Fjukstad, N. Angelvik, M. W. Hauglann, J. S. Knutsen, M. Grønnesby, H. Gunhildrud, and L. A. Bongo, "Low-cost programmable air quality sensor kits in science education," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. ACM, 2018, pp. 227–232.

[33] J. D. Watson, F. H. Crick *et al.*, "Molecular structure of nucleic acids," *Nature*, vol. 171, no. 4356, pp. 737–738, 1953.

[34] J. C. Venter, M. D. Adams, E. W. Myers, P. W. Li, R. J. Mural, G. G. Sutton, H. O. Smith, M. Yandell, C. A. Evans, R. A. Holt *et al.*, "The sequence of the human genome," *Science*, vol. 291, no. 5507, pp. 1304–1351, 2001.

[35] I. H. G. S. Consortium *et al.*, "Initial sequencing and analysis of the human genome," *Nature*, vol. 409, no. 6822, p. 860, 2001.

[36] M. L. Metzker, "Sequencing technologies—the next generation," *Nature reviews genetics*, vol. 11, no. 1, p. 31, 2010.

[37] M. Baker, "Why scientists must share their research code," *Nature News*, 2016.

[38] "Reproducibility in cancer biology: The challenges of replication," *eLife*, vol. 6, p. e23693, jan 2017.

[39] N. R. Council *et al.*, *Toward precision medicine: building a knowledge network for biomedical research and a new taxonomy of disease*. National Academies Press, 2011.

[40] I. F. Tannock and J. A. Hickman, "Limits to personalized cancer medicine," *New England Journal of Medicine*, vol. 375, no. 13, pp. 1289–1294, 2016.

[41] V. Dumeaux, K. S. Olsen, G. Nuel, R. H. Paulssen, A.-L. Børresen-Dale, and E. Lund, "Deciphering normal blood gene expression variation—the nowac postgenome study," *PLoS genetics*, vol. 6, no. 3, p. e1000873, 2010.

[42] M. Holden, L. Holden, K. Olsen, and E. Lund, "Local in time statistics for detecting weak gene expression signals in blood – illustrated for prediction of metastases in breast cancer in the nowac post-genome cohort," *Advances in Genomics and Genetics*, vol. 55, no. 2017:7, pp. 11–28, 2017.

[43] V. Dumeaux and E. Lund, "Gene expression profile in diagnostics," Oct. 22 2015, uS Patent App. 14/646,010.

[44] Y. Xie, *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, 2016.

[45] git-submodule, https://git-scm.com/docs/git-submodule.

[46] git-raw, https://github.com/atofigh/git-raw.

[47] git-annex, https://git-annex.branchable.com.

[48] Git LFS, https://git-lfs.github.com.

[49] R Markdown, http://rmarkdown.rstudio.com.

[50] Gitlab, https://gitlab.com/.

[51] G. K. Sandve, A. Nekrutenko, J. Taylor, and E. Hovig, "Ten simple rules for reproducible computational research," *PLoS computational biology*, vol. 9, no. 10, p. e1003285, 2013.

[52] R. Gentleman and D. Temple Lang, "Statistical analyses and reproducible research," *Journal of Computational and Graphical Statistics*, vol. 16, no. 1, pp. 1–23, 2007.

[53] J. S. S. Lowndes, B. D. Best, C. Scarborough, J. C. Afflerbach, M. R. Frazier, C. C. O'Hara, N. Jiang, and B. S. Halpern, "Our path to better science in less time using open data science tools," *Nature Ecology & Evolution*, vol. 1, no. 6, p. 0160, 2017.

[54] P. J. McMurdie and S. Holmes, "phyloseq: an r package for reproducible interactive analysis and graphics of microbiome census data," *PloS one*, vol. 8, no. 4, p. e61217, 2013.

[55] G. Finak, B. Mayer, W. Fulp, P. Obrecht, A. Sato, E. Chung, D. Holman, and R. Gottardo, "Datapackager: Reproducible data preprocessing, standardization and sharing using r/bioconductor for collaborative data

analysis," *Gates Open Research*, vol. 2, 2018.

[56] The Comprehensive R Archive Network (CRAN), https://cran.r-project.org.

[57] rpy2, https://rpy2.bitbucket.io.

[58] M. Tanabe and M. Kanehisa, "Using the KEGG database resource," *Current protocols in bioinformatics*, vol. 38, no. 1, pp. 1–12, 2012.

[59] M. Franz, C. T. Lopes, G. Huck, Y. Dong, O. Sumer, and G. D. Bader, "Cytoscape. js: a graph theory library for visualisation and analysis," *Bioinformatics*, vol. 32, no. 2, pp. 309–311, 2015.

[60] M. Bostock, V. Ogievetsky, and J. Heer, "D^3 data-driven documents," *IEEE transactions on visualization and computer graphics*, vol. 17, no. 12, pp. 2301–2309, 2011.

[61] A. Liberzon, A. Subramanian, R. Pinchback, H. Thorvaldsdóttir, P. Tamayo, and J. P. Mesirov, "Molecular signatures database (MSigDB) 3.0," *Bioinformatics*, vol. 27, no. 12, pp. 1739–1740, 2011.

[62] M. Kanehisa and S. Goto, "Kegg: kyoto encyclopedia of genes and genomes," *Nucleic acids research*, vol. 28, no. 1, pp. 27–30, 2000.

[63] E. Sayers, "Entrez programming utilities help," *http://www.ncbi.nlm.nih.gov/books/NBK25499*, 2009.

[64] K. A. Gray, B. Yates, R. L. Seal, M. W. Wright, and E. A. Bruford, "Genenames. org: the HGNC resources in 2015," *Nucleic acids research*, vol. 43, no. D1, pp. D1079–D1085, 2014.

[65] Sigma, http://sigmajs.org.

[66] P. Langfelder and S. Horvath, "Wgcna: an r package for weighted correlation network analysis," *BMC bioinformatics*, vol. 9, no. 1, p. 559, 2008.

[67] B. J. Boersma, M. Reimers, M. Yi, J. A. Ludwig, B. T. Luke, R. M. Stephens, H. G. Yfantis, D. H. Lee, J. N. Weinstein, and S. Ambs, "A stromal gene signature associated with inflammatory breast cancer," *International journal of cancer*, vol. 122, no. 6, pp. 1324–1332, 2008.

[68] A. Fabregat, F. Korninger, G. Viteri, K. Sidiropoulos, P. Marin-Garcia,

P. Ping, G. Wu, L. Stein, P. D'Eustachio, and H. Hermjakob, "Reactome graph database: Efficient access to complex pathway data," *PLoS computational biology*, vol. 14, no. 1, p. e1005968, 2018.

[69] J. M. Villaveces, R. C. Jimenez, and B. H. Habermann, "Keggviewer, a biojs component to visualize kegg pathways," *F1000Research*, vol. 3, 2014.

[70] C. Partl, A. Lex, M. Streit, D. Kalkofen, K. Kashofer, and D. Schmalstieg, "enroute: Dynamic path extraction from biological pathway maps for in-depth experimental data analysis," in *2012 IEEE Symposium on Biological Data Visualization (BioVis)*. IEEE, 2012, pp. 107–114.

[71] W. Luo, G. Pant, Y. K. Bhavnasi, S. G. Blanchard Jr, and C. Brouwer, "Pathview web: user friendly pathway visualization and data integration," *Nucleic acids research*, vol. 45, no. W1, pp. W501–W508, 2017.

[72] J. Bussery, L.-A. Denis, B. Guillon, P. Liu, G. Marchetti, and G. Rahal, "etriks platform: Conception and operation of a highly scalable cloud-based platform for translational research and applications development," *Computers in biology and medicine*, vol. 95, pp. 99–106, 2018.

[73] A. Bertram, "Renjin: The new R interpreter built on the JVM," in *The R User Conference, useR! 2013 July 10-12 2013 University of Castilla-La Mancha, Albacete, Spain*, vol. 10, no. 30, 2013, p. 105.

[74] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker, "Cytoscape: a software environment for integrated models of biomolecular interaction networks," *Genome research*, vol. 13, no. 11, pp. 2498–2504, 2003.

[75] K. Ono, T. Muetze, G. Kolishovski, P. Shannon, and B. Demchak, "Cyrest: Turbocharging Cytoscape access for external tools via a RESTful API," *F1000Research*, vol. 4, 2015.

[76] AppArmor, http://wiki.ubuntu.com/AppArmor.

[77] sparklyr: R interface for Apache Spark, http://spark.rstudio.com.

[78] SparkR, http://spark.apache.org/docs/latest/sparkr.html.

[79] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proceed-*

ings of the 9th USENIX conference on Networked Systems Design and Implementation. USENIX Association, 2012, pp. 2–2.

[80] N. Servant, J. Roméjon, P. Gestraud, P. La Rosa, G. Lucotte, S. Lair, V. Bernard, B. Zeitouni, F. Coffin, G. Jules-Clément *et al.*, "Bioinformatics for precision medicine in oncology: principles and application to the shiva clinical trial," *Frontiers in genetics*, vol. 5, 2014.

[81] A. Sboner and O. Elemento, "A primer on precision medicine informatics," *Briefings in bioinformatics*, vol. 17, no. 1, pp. 145–153, 2015.

[82] Picard, https://broadinstitute.github.io/picard.

[83] S. Andrews *et al.*, "Fastqc: a quality control tool for high throughput sequence data," 2010.

[84] A. M. Bolger, M. Lohse, and B. Usadel, "Trimmomatic: a flexible trimmer for Illumina sequence data," *Bioinformatics*, vol. 30, no. 15, pp. 2114–2120, 2014.

[85] A. McKenna, M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytsky, K. Garimella, D. Altshuler, S. Gabriel, M. Daly *et al.*, "The genome analysis toolkit: a MapReduce framework for analyzing next-generation dna sequencing data," *Genome research*, vol. 20, no. 9, pp. 1297–1303, 2010.

[86] J. Goecks, A. Nekrutenko, and J. Taylor, "Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences," *Genome biology*, vol. 11, no. 8, p. R86, 2010.

[87] Docker, https://www.docker.com.

[88] BioContainers, "Biocontainers," https://biocontainers.pro, 2017.

[89] K. Cibulskis, M. S. Lawrence, S. L. Carter, A. Sivachenko, D. Jaffe, C. Sougnez, S. Gabriel, M. Meyerson, E. S. Lander, and G. Getz, "Sensitive detection of somatic point mutations in impure and heterogeneous cancer samples," *Nature biotechnology*, vol. 31, no. 3, pp. 213–219, 2013.

[90] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull, "Graphviz—open source graph drawing tools," in *International Symposium on Graph Drawing*. Springer, 2001, pp. 483–484.

[91] A. Cornish and C. Guda, "A comparison of variant calling pipelines using genome in a bottle as a reference," *BioMed research international*, vol. 2015, 2015.

[92] Arvados, https://arvados.org.

[93] G. Kaushik, S. Ivkovic, J. Simonovic, N. Tijanic, B. Davis-Dusenbery, and D. Kural, "Rabix: an open-source workflow executor supporting recomputability and interoperability of workflow descriptions," in *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, vol. 22. NIH Public Access, 2016, p. 154.

[94] W. Tang, J. Wilkening, N. Desai, W. Gerlach, A. Wilke, and F. Meyer, "A scalable data analysis platform for metagenomics," in *Big Data, 2013 IEEE International Conference on*. IEEE, 2013, pp. 21–26.

[95] J. W. Lau, E. Lehnert, A. Sethi, R. Malhotra, G. Kaushik, Z. Onder, N. Groves-Kirkby, A. Mihajlovic, J. DiGiovanna, M. Srdic *et al.*, "The cancer genomics cloud: Collaborative, reproducible, and democratized—a new paradigm in large-scale computational research," *Cancer research*, vol. 77, no. 21, pp. e3–e6, 2017.

[96] Kubernetes, https://kubernetes.io.

[97] J. A. Novella, P. Emami Khoonsari, S. Herman, D. Whitenack, M. Capuccini, J. Burman, K. Kultima, and O. Spjuth, "Container-based bioinformatics with pachyderm," *Bioinformatics*, p. bty699, 2018.

[98] G. M. Kurtzer, V. Sochat, and M. W. Bauer, "Singularity: Scientific containers for mobility of compute," *PloS one*, vol. 12, no. 5, p. e0177459, 2017.

[99] P. Di Tommaso, M. Chatzou, E. W. Floden, P. P. Barja, E. Palumbo, and C. Notredame, "Nextflow enables reproducible computational workflows," *Nature biotechnology*, vol. 35, no. 4, p. 316, 2017.

[100] B. Kim, T. A. Ali, C. Lijeron, E. Afgan, and K. Krampis, "Bio-docklets: Virtualization containers for single-step execution of ngs pipelines." *bioRxiv*, p. 116962, 2017.

[101] A. A. Ali, M. El-Kalioby, and M. Abouelhoda, "The case for docker in multicloud enabled bioinformatics applications," in *International Conference on Bioinformatics and Biomedical Engineering*. Springer, 2016, pp. 587–601.

[102] P. Belmann, J. Dröge, A. Bremges, A. C. McHardy, A. Sczyrba, and M. D. Barton, "Bioboxes: standardised containers for interchangeable bioinformatics software," *Gigascience*, vol. 4, no. 1, p. 47, 2015.

[103] P. Di Tommaso, E. Palumbo, M. Chatzou, P. Prieto, M. L. Heuer, and C. Notredame, "The impact of docker containers on the performance of genomic pipelines," *PeerJ*, vol. 3, p. e1273, 2015.

[104] C. Boettiger, "An introduction to docker for reproducible research," *ACM SIGOPS Operating Systems Review*, vol. 49, no. 1, pp. 71–79, 2015.

Paper 1

B. Fjukstad, K. S. Olsen, M. Jareid, E. Lund, and L. A. Bongo, "Kvik: three-tier data exploration tools for flexible analysis of genomic data in epidemiological studies," *F1000Research*, vol. 4, 2015

The following includes 5 of the 15 total pages. The first 5 pages is the paper, while remaining 10 pages are open reviews, responses, and additions to the final version of the paper. These are available online at `f1000research.com/articles/4-81/v2`

SOFTWARE TOOL ARTICLE

REVISED Kvik: three-tier data exploration tools for flexible analysis of genomic data in epidemiological studies [version 2; referees: 1 approved, 2 approved with reservations]

Bjørn Fjukstad[1], Karina Standahl Olsen[2], Mie Jareid[2], Eiliv Lund[2], Lars Ailo Bongo[1]

[1]Department of Computer Science, UiT - The Arctic University of Norway, Tromsø, 9037, Norway
[2]Department of Community Medicine, UiT - The Arctic University of Norway, Tromsø, 9037, Norway

v2 **First published:** 30 Mar 2015, **4**:81 (doi: 10.12688/f1000research.6238.1)
Latest published: 16 Jun 2015, **4**:81 (doi: 10.12688/f1000research.6238.2)

Abstract

Kvik is an open-source framework that we developed for explorative analysis of functional genomics data from large epidemiological studies. Creating such studies requires a significant amount of time and resources. It is therefore usual to reuse the data from one study for several research projects. Often each project requires implementing new analysis code, integration with specific knowledge bases, and specific visualizations. Although existing data exploration tools are available for single study data exploration, no tool provides all the required functionality for multistudy data exploration. We have therefore used the Kvik framework to develop Kvik Pathways, an application for exploring gene expression data in the context of biological pathways. We have used Kvik Pathways to explore data from both a cross-sectional study design and a case-control study within the Norwegian Women and Cancer (NOWAC) cohort. Kvik Pathways follows the three-tier architecture in web applications using a powerful back-end for statistical analyses and retrieval of metadata.In this note, we describe how we used the Kvik framework to develop the Kvik Pathways application. Kvik Pathways was used by our team of epidemiologists toexplore gene expression data from healthy women with high and low plasma ratios of essential fatty acids.

This article is included in the Container Virtualization in Bioinformatics channel.

Open Peer Review

Referee Status: ✔ ? ?

	Invited Referees 1	2	3
REVISED version 2 published 16 Jun 2015	✔ report		? report
version 1 published 30 Mar 2015	? report	? report	

1 **Paul Klemm**, Otto-von-Guericke University Magdeburg Germany

2 **Zhenjun Hu**, Boston University USA

3 **Lilit Nersisyan**, National Academy of Sciences of Armenia Armenia

Discuss this article

Comments (0)

Corresponding author: Lars Ailo Bongo (larsab@cs.uit.no)

How to cite this article: Fjukstad B, Standahl Olsen K, Jareid M *et al.* **Kvik: three-tier data exploration tools for flexible analysis of genomic data in epidemiological studies [version 2; referees: 1 approved, 2 approved with reservations]** *F1000Research* 2015, **4**:81 (doi: 10.12688/f1000research.6238.2)

Grant information: This work was supported by a grant from the European Research Council, under the title "Transcriptomics in cancer epidemiology - TICE".

Competing interests: No competing interests were disclosed.

First published: 30 Mar 2015, **4**:81 (doi: 10.12688/f1000research.6238.1)

REVISED **Amendments from Version 1**

Overall we reduced the implementation details in the note. This was something that both reviewers pointed out and we felt that the note was a bit too technical. We also clarified the difference between Kvik and Kvik Pathways. We have changed the requirements and included a list of contributions. We also revisited the figures to make them more clear to the reader. We also fixed some grammatical errors.

See referee reports

Introduction

Visual explorative analysis is essential for understanding biological functions in large-scale omics' datasets. However, enabling the inclusion of omics' data in large epidemiological studies requires collecting samples from thousands of people at different biological levels over a long period of time. It is therefore usual to reuse the data for different research questions and projects. Although an existing tool may be useful for one project, no tool provides the required functionality for several different projects.

We have designed and implemented Kvik, a framework that makes it easy to develop new applications to explore different research questions and data. The initial version Kvik[1] contained a prototype system for exploring biological pathways and gene expression data. From this prototype we built the Kvik Framework, which provides developers a simple interface to powerful systems for statistical analyses and meta-databases, and Kvik Pathways: a publicly available data exploration application. From our experience in developing a framework for building data exploration applications, we identified four requirements such applications should satisfy:

Interactive The applications should provide interactive exploration of datasets through visualizations and integration with relevant information. To understand the large quantities of heterogeneous data in epidemiological studies, researchers need interactive visualizations that provide different views and presentations of the data. Also, to understand the results it is important to have instant access to existing knowledge from online databases.

Familiar They should use familiar visual representations to present information to researchers. For more efficient data exploration it is effective to use representations that researchers are familiar with both from the literature and from other applications.

Simple to use Researchers should not need to install software to explore their data through the applications. The applications should protect the researcher from the burden of installing and keeping an application up to date.

Lightweight Data presentation and computation should be separated to make it possible for researchers to explore data without having to have the computational power to run the analyses. With the growing rate data is produced at, we cannot expect that researchers have the resources to store and analyze data on their own computers.

There are several tools for exploring biological data in the context of pathways, such as VisANT (available online at visant.bu.edu) by 2, VANTED (available online at vanted.ipk-gatersleben.de)[3], enRoute by 4 or Entourage by 5 (both available online at caleydo.org). However, these tools do not provide the adaptability needed for exploration of multi-study datasets. Many existing tools place the visualization, data analysis and storage on the user's computer, making it necessary to have a powerful computer. In addition, the tools are often standalone applications that require users to install and update the applications. Kvik Pathways satisfies the above requirements as follows:

Interactive Kvik Pathways provides interactive pathway visualizations and information from the popular Kyoto encyclopedia of genes and genomes (KEGG)[6] database (available online at kegg.jp).

Simple to use Kvik Pathways uses HTML5 and modern JavaScript libraries to provide an interactive application that runs in any modern web browser.

Familiar Kvik Pathways uses the familiar pathway representations from KEGG and graphical user interfaces found in modern web applications.

Lightweight Kvik Pathways uses a powerful back-end provided by the Kvik framework to perform statistical analyses.

Both Kvik and Kvik Pathways are open-sourced at github.com/fjukstad/kvik. We provide an online version of Kvik Pathways at kvik.cs.uit.no and to run Kvik Pathways in a local Docker instance or on a cloud service such as Amazon Web Services (aws.amazon.com) or Google Compute Engine (cloud.google.com/compute), we provide a Docker image at registry.hub.docker.com/u/fjukstad/kvik.

In this note we describe how we used Kvik to implement Kvik Pathways, a tool for exploring gene expression in the context of biological pathways. In Kvik Pathways researchers can explore gene expression data from 7 combined with information from online knowledge bases. We provide the following contributions:

- Kvik Pathways, a publicly available web application for exploring gene expression data in the context of biological pathways without any additional applications than a web browser.
- A requirement analysis for interactive exploration tools for epidemiological studies.
- A detailed description of how we have used Kvik Pathways to explore gene expression data from healthy women with high and low plasma ratios of essential fatty acids.

Methods

Kvik Pathways allows users to interactively explore a molecular dataset, such as gene expression, through a web application. It provides pathway visualizations and detailed information about genes and pathways from the KEGG databases (Figure 1). Through pathway visualizations and integration with the KEGG databases, epidemiologists can perform targeted exploration of pathways and genes

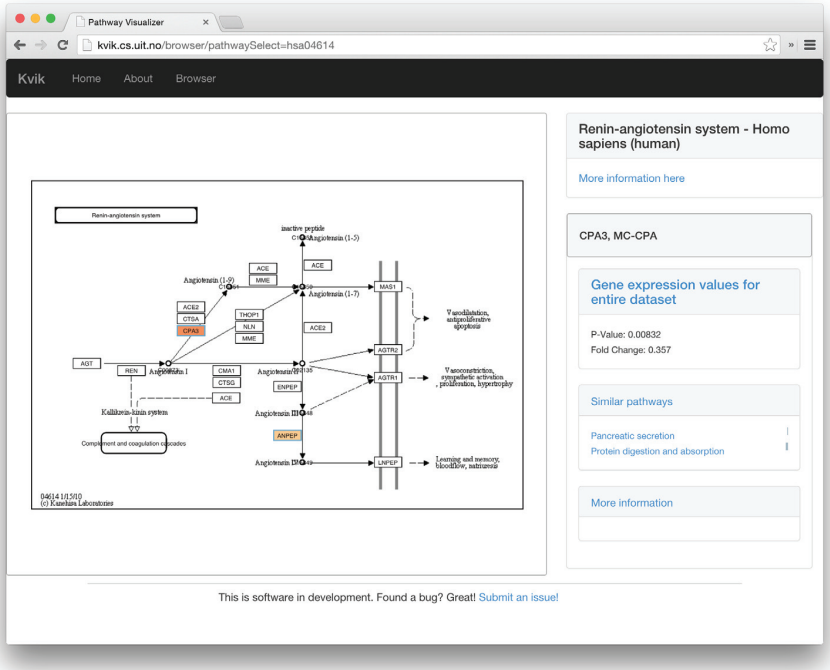


Figure 1. Screenshot of the renin-angiotensin pathway (KEGG pathway id hsa04614) in Kvik Pathways. The user has selected the gene CPA3, which brings up the panel on the right. From here researchers can browse pathways that the gene is a member of, and read relevant information about the gene from KEGG.

to get an overview of the biological functions that are involved with gene expression from the underlying dataset. Kvik Pathways gathers information about related pathways and retrieves relevant information about genes, making it unnecessary for researchers to spend valuable time looking up this information manually. For example, navigating a set of pathways and browsing information about genes in these, requires the researcher to manually query KEGG for each specific gene. Kvik Pathways retrieves information about genes without the researcher having to leave the pathway visualization to retrieve relevant information.

The Kvik framework provides a flexible statistics back-end where researchers can specify the analyses they want to run to generate data for later visualization. For example, in Kvik Pathways we retrieve fold change for single genes every time a pathway is viewed in the application. These analyses are run ad hoc on the back-end servers and generates output that is displayed in the pathways in the client's web browser. The data analyses are implemented in a simple R script and can make use of all available libraries in R, such as Bioconductor (bioconductor.org).

Researchers modify this R script to, for example, select a normalization method, or to tune the false discovery rate (FDR) used to adjust the *p*-values that Kvik Pathways uses to highlight significantly differentially expressed genes. Since Kvik Pathways is implemented as a web application and the analyses are run ad hoc, when the analyses change, researchers get an updated application by simply refreshing the Kvik Pathways webpage.

Implementation

We implemented interactive visualizations using the Cytoscape.js (js.cytoscape.org) library to generate the interactive pathway visualizations, and D3 (d3js.org) for Document Object Model (DOM) manipulation such as generating bar charts with HTML *<svg>* elements. We integrate these with the popular Bootstrap front-end framework (getbootstrap.com) to provide a familiar and aesthetically pleasing user interface.

Kvik Pathways has a three-tiered architecture of independent layers (Figure 2). The browser layer consists of the web application for exploring gene expression data and biological pathways. A front-end layer provides static content such as HTML pages and stylesheets, as well as an interface to the data sources with dynamic content such as gene expression data or pathway maps to the web application. The back-end layer contains information about pathways and genes, as well as computational and storage resources to process genomic data such as the NOWAC data repository. The Kvik framework provides the components in the back-end layer.

In our setup the Data Engine in the back-end layer provides an interface to the NOWAC data repository stored on a secure server on our local supercomputer. In Kvik Pathways all gene expression data is

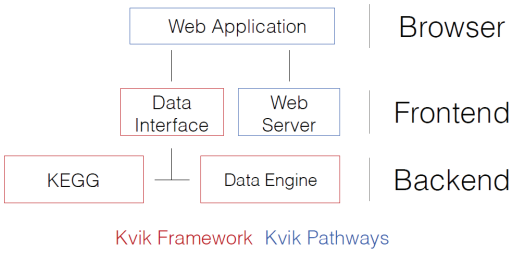


Figure 2. The three-tiered architecture of Kvik Pathways.

stored on the computer that runs the Data Engine. The Data Engine runs an R session accessible over remote procedure calls (RPCs) from the front-end layer using RPy2 (rpy.sourceforge.net) to interface with R. To access data and run analyses the Data Interface exposes a HTTP API to the browser layer (Table 1 provides the interfaces).

To create pathway visualizations the Kvik back-end retrieves and parses the KEGG Markup Language (KGML) representation and pathway image from KEGG databases through its REST API (rest.kegg.jp). This KGML representation of a pathway is an XML file that contains a list of nodes (genes, proteins or compounds) and edges (reactions or relations). Kvik parses this file and generates a JSON representation that Kvik Pathway uses to create pathway visualizations. Kvik Pathways Cytoscape.js to create a pathway visualization from the list of nodes and edges and overlay the nodes on the pathway image. To reduce latency when using the KEGG REST API, we cache every response on our servers. We use the average fold change between the groups (women with high or low plasma ratios of essential fatty acids) in the dataset to color the genes within the pathway maps. To highlight *p*-values, the pathway visualization shows an additional colored frame around genes. We visualize fold change values for individual samples as a bar chart in a side panel. This bar chart gives researchers a global view of the fold change in the entire dataset.

Operation

Kvik Pathways runs in all modern web browsers and does not require any third-party software.

Table 1. The REST interface to the Data Engine. All URLs are relative to the hostname where the Data Engine server runs. On our public installation the Data Engine runs on kvik.cs.uit.no:8888. For example, use kvik.cs.uit.no:8888/genes/ to retrieve all available genes in our dataset. By using a HTTP API we can build different data exploration applications in virtually any programming language.

URL	Description
/fc/[genes...]	Calculate and retrieve fold-change for the specified genes
/pvalues/[genes...]	Calculate and retrieve *p*-values for the specified genes
/exprs/[genes...]	Get the raw gene expression values from the dataset
/genes	Get a list of all genes in the dataset

Use case

We used Kvik Pathways to repeat the analyses in a previous published project (7, doi: 10.1371/journal.pone.0067270) that compared gene expression in blood from healthy women with high and low plasma ratios of essential fatty acids. Gene expression differences between groups were assessed using *t*-tests (*p*-values adjusted with the Benjamini-Hochberg method). There were 184 differentially expressed genes significant on the 5% level. When exploring this gene list originally, functional information was retrieved from GeneCards and other repositories, and the list was analyzed for overlap with known pathways using MSigDB (available online at broadinstitute.org/gsea/msigdb). The researchers had to manually maintain overview of single genes, gene networks or pathways, and gather functional information gene by gene while assessing differences in gene expression levels. With this approach, researchers are limited by manual capacity, and the results may be prone to researcher bias. Kvik Pathways eliminates this researcher bias and does not limit the information retrieval to a researcher's manual capacity.

Initially, Kvik Pathways was implemented to explore gene expression data from a not yet published dataset. To use Kvik Pathways to explore the data from the analyses in 7, we only needed to make small modifications to the analysis R script used by the Data Engine. (The modified R script is found at github.com/fjukstad/kvik/blob/master/dataengine/data-engine.r). Instead of loading the unpublished dataset, we could load the dataset from 7 and use the four functions that are accessible over RPC (Table 1 shows the HTTP API which uses the underlying RPCs). Currently this script is less than 30 lines, consisting of four functions to retrieve data and a simple initialization step that reads the dataset. Researchers only have to modify these four functions to enable exploration of new datasets. As of the current implementation of Kvik Pathways researchers have to modify the analysis script outside the application.

As an example of practical use of Kvik Pathways, we chose one of the significant pathways from the overlap analysis, the renin-angiotensin pathway (Supplementary table S5 in 7). The pathway contains 17 genes, and in the pathway map we could instantly identify the two genes that drive this result. The color of the gene nodes in the pathway map indicates the fold change, and the statistical significance level is indicated by the color of the node's frame. We use this image of a biological process to see how these two genes (and their expression levels) are related to other genes in that pathway, giving a biologically more meaningful context as compared to merely seeing the two genes on a list.

Summary

Kvik Pathways is an open-source system for explorative analyses of functional genomics data from epidemiological studies. It uses R to perform on-demand data analyses providing a flexible back-end that can expand to new analyses and research projects. It uses modern visualization libraries and a powerful back-end for on-demand statistical analyses. Epidemiologists are using Kvik Pathways to analyze gene expression data. Kvik Pathways is open-sourced at github.com/fjukstad/kvik and is available as a Docker image at registry.hub.docker.com/u/fjukstad/kvik.

Data availability

Data used in the use case is available in the Gene Expression Omnibus (ncbi.nlm.nih.gov/geo), under accession number GSE15289.

Software availability

Latest source code

https://github.com/fjukstad/kvik

Source code as at the time of publication

https://github.com/F1000Research/kvik/releases/tag/1.0

Archived source code as at the time of publication

http://dx.doi.org/10.5281/zenodo.16375

Software license

The MIT license.

Author contributions

LAB and BF designed the architecture of the system. BF implemented. All conducted the requirements analysis. EL, MJ, KSO contributed case study. BF drafted manuscript. All authors read, revised and approved the manuscript.

Competing interests

No competing interests were disclosed.

Grant information

This work was supported by a grant from the European Research Council, under the title "Transcriptomics in cancer epidemiology - TICE".

Acknowledgements

Gene expression profiles were analyzed at the Microarray Resource Center Tromsø, UiT – The Arctic university of Norway.

References

1. Fjukstad B, Olsen KS, Jareid M, *et al.*: **Kvik: Interactive exploration of genomic data from the NOWAC postgenome biobank.** Norsk Informatikkonferanse (NIK). 2014.
Reference Source
2. Hu Z, Chang YC, Wang Y, *et al.*: **VisANT 4.0: Integrative network platform to connect genes, drugs, diseases and therapies.** *Nucleic Acids Res.* 2013; **41**(Web Server issue): W225–W231.
PubMed Abstract | Publisher Full Text | Free Full Text
3. Junker BH, Klukas C, Schreiber F: **VANTED: a system for advanced data analysis and visualization in the context of biological networks.** *BMC Bioinformatics.* 2006; **7**(1): 109.
PubMed Abstract | Publisher Full Text | Free Full Text
4. Partl C, Lex A, Streit M, *et al.*: **enRoute: Dynamic path extraction from biological pathway maps for in-depth experimental data analysis**. In *Biological Data Visualization (BioVis), 2012 IEEE Symposium on,* pages 107–114.
Publisher Full Text
5. Lex A, Partl C, Kalkofen D, *et al.*: **Entourage: visualizing relationships between biological pathways using contextual subsets.** *IEEE Trans Vis Comput Graph.* 2013; **19**(12): 2536–2545.
PubMed Abstract | Publisher Full Text | Free Full Text
6. Kanehisa M, Goto S: **KEGG: Kyoto Encyclopedia of Genes and Genomes.** *Nucleic Acids Res.* 2000; **28**(1): 27–30.
PubMed Abstract | Publisher Full Text | Free Full Text
7. Olsen KS, Fenton C, Frøyland L, *et al.*: **Plasma fatty acid ratios affect blood gene expression profiles--a cross-sectional study of the Norwegian Women and Cancer Post-Genome Cohort.** *PLoS One.* 2013; **8**(6): e67270.
PubMed Abstract | Publisher Full Text | Free Full Text

Paper 2

B. Fjukstad, V. Dumeaux, K. S. Olsen, E. Lund, M. Hallett, and L. A. Bongo, "Building applications for interactive data exploration in systems biology," in *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*. ACM, 2017, pp. 556–561

Building Applications for Interactive Data Exploration in Systems Biology

Bjørn Fjukstad
Department of Computer Science
UiT The Arctic University of Norway

Vanessa Dumeaux
Department of Biology
Concordia University

Karina Standahl Olsen
Department of Community Medicine
UiT The Arctic University of Norway

Eiliv Lund
Department of Community Medicine
UiT The Arctic University of Norway

Michael Hallett
Department of Biology
Concordia University

Lars Ailo Bongo
Department of Computer Science
UiT The Arctic University of Norway

ABSTRACT

The significant increase in the rate of data generation by the systems biology community creates a need for interactive exploration tools to explore the resultant datasets. Such tools need to combine advanced statistical analyses, prior knowledge from biological databases, and interactive visualizations with intuitive user interfaces. Each specific research question potentially requires a specialized user interface and visualization methods. Although some features are application-specific, the underlying components of the data analysis tool can be shared and reused.

Our approach for developing data exploration tools in systems biology builds on the microservice architecture that separates an application into smaller components which can communicate using language-agnostic protocols. We show that this design is well suited for bioinformatics applications where different tools written in different languages by different research groups is the norm. Packaging each service in a software container enables re-use and sharing of key components between applications, reducing development, deployment, and maintenance time.

We demonstrate the viability of our approach through a web application, entitled MIxT blood-tumor, for exploring and comparing transcriptional profiles from blood and tumor samples in breast cancer patients. The application integrates advanced statistical software, up-to-date information from biological databases, and modern data visualization libraries.

ACM-BCB'17, August 20-23, 2017, Boston, MA, USA.
 978-1-4503-4722-8/17/08...$15.00
DOI: http://dx.doi.org/10.1145/3107411.3107481

KEYWORDS

Interactive data exploration, software containers, visualization, microservices, systems biology, breast cancer.

INTRODUCTION

In recent years the biological community has generated an unprecedented amount of data. While the cost of data collection has drastically decreased, data analysis continue to represent a large fraction of the total cost of these studies.[9] Data analysis tools, especially those designed specifically for the project at hand, provide clear benefit to the human experts who are interpreting data and deriving results.

Often in systems biology studies, the ability to explore newly generated data relative to prior knowledge located in third-party databases and software systems is key. This includes, for example, entities such as the Gene Ontology (GO),[1] the Kyoto Encyclopedia of Genes and Genomes (KEGG),[2] and the Molecular Signatures Database (MSigDB)[3] that together catalog the function of nearly every gene, gene product, pathway or cellular process. These tools, and most bioinformatics databases in general, offer interfaces for data retrieval.

Data analysis in systems biology is greatly reliant on programming languages especially tailored to these domains, providing easy direct access to specific algorithms and statistical routines. The R statistical programming language provides developers open access to thousands of libraries through repositories such as CRAN[4] or Bioconductor[5]. Similarly, other languages such as Python and Go have bioinformatic extensions including BioPython[2] and biogo[6] respectively, providing domain specific routines. Although tremendously helpful, different tools and languages are used in different domains of systems biology for many reasons. This creates a

[1] geneontology.org.
[2] kegg.jp.
[3] software.broadinstitute.org/gsea/msigdb.
[4] cran.r-project.org.
[5] bioconductor.org.

need for novel approaches to integrate the different libraries between the programming languages and tools.

A microservice architecture structures an application into small reusable, loosely coupled parts. These communicate via lightweight programming language-agnostic protocols such as HTTP, thus making it possible to write single applications in multiple programming languages. This way the most suitable programming language is used for each specific part. To build a microservice application, developers bundle each service in a software container. Containers are built from configuration files which describe the operating system, software packages and their associated versions. Several software container implementations exist including Rkt[6] but Docker,[7] is perhaps the most broadly used. Initiatives such as BioContainers[8] now provide containers pre-installed with different bioinformatics tools. While the enabling technology is available, the microservices approach is not yet widely adopted in bioinformatics.[11]

From our experience we identified a set of components and features that are central to building data exploration applications.

(1) A low-latency language-independent approach for integrating, or embedding, statistical software, such as R, directly into a data exploration application.
(2) Low latency language-independent interface to online reference databases in biology that users can query to explore analyses.
(3) A simple method for deploying and sharing the components of an application between projects.

In this paper, we describe a novel approach for building data exploration applications in systems biology via a sample web application, MIxT (Matched Interactions across Tissues) using high-throughput gene expression profiles of breast cancer tumor data with matched profiles from the patients blood.

METHODS

In this section we first motivate our microservice approach based on our experiences developing the MIxT web application. We describe the process from initial data analysis to the final application, highlighting the importance of language-agnostic services to facilitate the use of different tools in different parts of the application. We then generalize the ideas to a set of principles and services that can be reused and shared between applications, and show their design and implementation.

Motivating Example

The aim of the Matched Interactions Across Tissues (MIxT) study was to identify genes and pathways in the primary breast tumor that are tightly linked to genes and pathways in the patient blood cells.[3] We generated and analyzed expression profiles from blood and matched tumor cells in 173 breast cancer patients included in the Norwegian Women and Cancer (NOWAC) study. The MIxT analysis starts by identifying sets of genes tightly co-expressed across all patients in each tissue. Each group of genes or modules were annotated based on a priori biological knowledge about gene functionality. Focus was placed on the relationships between tissues by asking if specific biologies in one tissue are linked with (possibly distinct) biologies in the second tissue, and this within different subgroup of patients (i.e. subtypes of breast cancer).

We built an R package, mixtR,[9] with the statistical methods and static visualizations for identifying associations between modules across tissues. To make the results more easily accessible we built a web application that interfaces with the R package, but also online databases to retrieve relevant metadata. To make it possible to easily update or re-implement parts of the system without effecting the entire application, it was developed using a microservice architecture. The software containers allowed the application to be deployed on a wide range of hardware, from local installations to cloud systems.

Design Principles

Our experience can be generalized into the following design principles for building applications in bioinformatics:

Principle 1: Build applications as collections of language-agnostic microservices. This enables re-use of components and does not enforce any specific programming language on the user interfaces or the underlying components of the application.

Principle 2: Use software containers to package each service. This has a number of benefits: it simplifies deployment, ensures that dependencies and libraries are installed, and simplifies sharing of services between developers.

Microservice Design and Implementation

In the rest of the section we describe how we designed and implemented two microservices in Kvik[4] which we later used to build the MIxT web application.

Compute Service. The main goal of a data exploration application in systems biology is to help users discover interesting patterns in a biological dataset. Because of the complexity of biological data and analyses, we need specialized software to help find these patterns. Because these tools are built to provide specialized analyses, they often don't provide a reusable interface outside the programming environment they are built in.

We have built a compute service that provides an open interface directly to the R programming language, thus providing access to a wealth of algorithm and statistical analysis packages that exists within the R ecosystem. Application developers can use the compute service to execute specialized analyses and retrieve results either as plain text or binary data such as plots. By interfacing directly with R, developers

[6] coreos.com/rkt.
[7] docker.com.
[8] biocontainers.pro.
[9] Available online at github.com/vdumeaux/mixtR.

can modify input parameters to statistical methods directly from the user-facing application.

The compute service offers three main operations to interface with R: i) to call a function with one or more input parameters from an R package, ii) to get the results from a previous function call, and iii) a catch-all term that both calls a function and returns the results. We use the same terminology as OpenCPU[8] and have named the three operations Call, Get, and RPC respectively. These three operations provide the necessary interface for applications to include statistical analyses in the applications.

The compute service is implemented as an HTTP server that communicates with a pre-set number of R processes to execute statistical analyses. At initiation of the compute service, a user-defined number of R worker sessions are launched for executing analyses (default is 5). The compute service uses a round-robin scheduling scheme to distribute incoming requests to the workers. We provide a simple FIFO queue for queuing of requests. The compute service also provides the opportunity for applications to cache analysis results to speed up subsequent calls.

Database Service. To interpret data, experts regularly exploit prior knowledge via database queries and the primary scientific literature. There are a wealth of online databases, some of which provide open APIs in addition to web user interfaces that application developers can make use of. While the databases can provide helpful information, there are some limitations associated with their integration into interactive data exploration applications: i) the APIs are not fast enough to use in interactive applications where the application has to perform multiple database queries, ii) some databases put restrictions on the number of database queries, and iii) there is no uniform way for storing additional database metadata to identify database versions and query parameters.

To alleviate application developers of these challenges, we built an database service that provides a solution to the three. The service provides low latency, minimizes the number of queries to remote databases, and stores additional metadata to capture query parameters and database information. The database service provides an open HTTP interface to biological databases for retrieving meta-data on genes and processes. We currently have packages for interfacing with E-utilities,[10] MSigDB, HGNC, and KEGG.

Both the compute and the databases service in Kvik build on the standard *net/http* package in the Go programming language.[11] The database service use the *gocache*[12] package to cache any query to an online database. In addition we deploy each service as Docker containers.[13]

[10]eutils.ncbi.nlm.nih.gov.
[11]golang.org
[12]github.com/fjukstad/gocache.
[13]Available at hub.docker.com/r/fjukstad/kvik-r and hub.docker.com/r/fjukstad/db.

MATCHED INTERACTIONS ACROSS TISSUES (MIXT)

We show the viability of the microservices approach in Kvik by describing the MIxT web application for exploring and comparing transcriptional profiles from blood and tumor samples. We conduct an initial evaluation to illustrate that we can built interactive applications using the microservices provided by Kvik.

Analysis Tasks

The web application provides functionality to perform six data analysis tasks (A1-A6):

A1: Explore co-expression gene sets in tumor and blood tissue. Users can explore gene expression patterns together with clinicopathological variables (e.g. patient or tumor grade, stage, age) for each module. In addition we enable users to study the underlying biological functions of each module by including gene set analyses between the module genes and known gene sets.

A2: Explore co-expression relationships between genes. Users can explore the co-expression relationship as a graph visualization. Here genes are represented in the network with nodes and edges represent statistically significant correlation in expression between the two end-points.

A3: Explore relationships between modules from each tissue. We provide two different metrics to compare modules, and the web application enables users to interactively browse these relationships. In addition to providing visualizations the compare modules from each tissue, users can explore the relationships, but for different breast cancer patient groups.

A4: Explore relationships between clinical variables and modules. In addition to comparing the association between modules from both tissues, users also have the possibility to explore the association with a module and a specific clinical variable. It is also possible to explore the associations after first stratifying the tumors by breast cancer subtype (an operation that is common in cancer related studies to deal with molecular heterogeneity).

A5: Explore association between user-submitted gene lists and computed modules. We want to enable users to explore their own gene lists to explore them in context of the co-expression gene sets. The web application must handle uploads of gene lists and compute association between the gene list and the MIxT modules on demand.

A6: Search for genes or gene lists of interest. To facilitate faster lookup of genes and biological processes, the web application provides a search functionality that lets users locate genes or gene lists and show association to the co-expression gene sets.

Design and Implementation

From these six analysis tasks we designed and implemented MIxT as a web application that integrates statistical analyses and information from biological databases together with interactive visualizations. Figure 1 shows the system architecture of MIxT which consists of three parts i) the web application

itself containing the user-interface and visualizations; ii) the compute service performing the MIxT analyses developed in an R package, delivering data to the web application; and iii) the database service providing up-to-date information from biological databases. Each of these components run within Docker containers making the process of deploying the application simple.

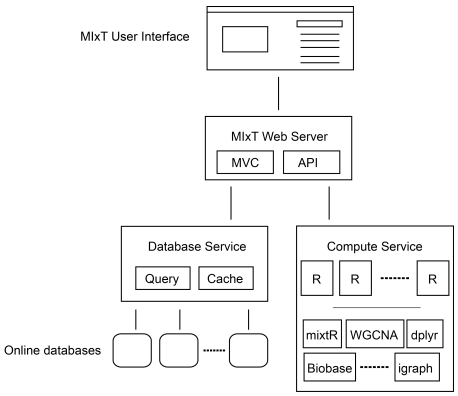


Figure 1: The architecture of the MIxT system. It consists of a web application, the hosting web server, a database service for retrieving metadata and a compute service for performing statistical analysis. Note that only the web application and the R package are specific to MIxT, the rest of the components can be reused in other applications.

We structured the MIxT application with a separate view for each analysis task. To explore the co-expression gene sets (**A1**), we built a view that combines both static visualizations from R together with interactive tables for gene overlap analyses. Figure 2 shows the web page presented to users when they access the co-expression gene set 'darkturquoise' from blood. To explore the co-expression relationship between genes (**A2**) we use an interactive graph visualization build with Sigmajs[14]. We have built visualization for both tissues, with graph sizes of 2705 nodes and 90 348 edges for the blood network, and 2066 nodes and 50 563 edges for the biopsy network. To visualize relationships between modules from different tissues (**A3**), or their relationship to clinical variables (**A4**) we built a heatmap visualization using the d3[15] library. We built a simple upload page where users can specify their gene sets (**A5**). The file is uploaded to the web application which redirects it to the compute service that runs the analyses. Similarly we can take user input to search for genes and processes (**A6**).

The web application is hosted by a custom web server. This web server is responsible for dynamically generating the different views based on data from the statistical analyses and biological databases, and serve these to users. It also

[14]sigmajs.org.
[15]d3js.org.

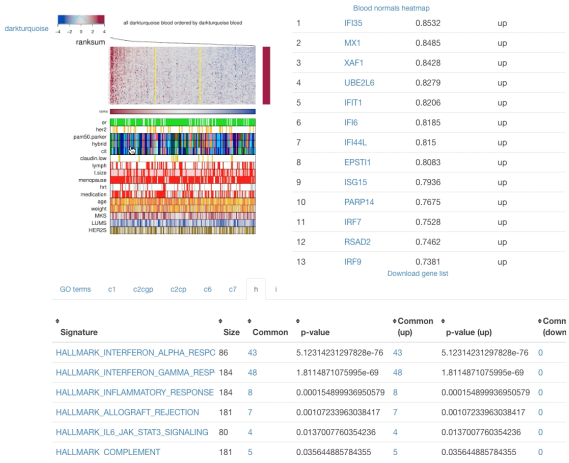


Figure 2: MIxT module overview page. The top left panel contains the gene expression heatmap for the module genes. The top right panel contains a table of the genes found in the module. The bottom panel contains the results of gene overlap analyses from the module genes and known gene sets from MSigDB.

serves the different JavaScript visualization libraries and style sheets.

Evaluation

To investigate if it is feasible to implement parts of an application as separate services, we evaluate the response times for a set of queries to each of the two supporting services.

To evaluate the database service we measure the query time for retrieving information about a specific gene with and without caching.[16] This illustrates how we can improve performance in an application by using a database service rather than accessing the database directly. We use a AWS EC2 *t2.micro*[17] instance to host and evaluate the database service. The results in Table 1 confirm a significant improvement in response time when the database service caches the results from the database lookups. In addition by serving the results out of cache we reduce the number of queries to the online database down to one.

Table 1: Time to retrieve a gene summary for a single gene, comparing different number of concurrent requests.

	1	2	5	10	15
No cache	956ms	1123ms	1499ms	2147ms	2958ms
Cache	64ms	64ms	130ms	137ms	154ms

We evaluate the compute service by running a benchmark consisting of two operations: first generate a set of 100

[16]More details online at github.com/fjukstad/kvik/tree/master/db/benchmark.
[17]See aws.amazon.com/ec2/instance-types for more information about AWS EC2 instance types.

random numbers, then plot them and return the resulting visualization.[18] We use two *c4.large* instances on AWS EC2 running the Kvik compute service and OpenCPU base docker containers. The servers have caching disabled. Table 2 shows the time to complete the benchmark for different number of concurrent connections. We see that the compute service in Kvik performs better than the OpenCPU[19] alternative. We believe that speedup is because we keep a pool of R processes that handle requests. In OpenCPU a new R process is forked upon every request that results in any computation executed in R. Other requests such as retrieving previous results do not fork new R processes.

Table 2: Time to complete the benchmark with different number of concurrent connections.

	1	2	5	10	15
Kvik	274ms	278ms	352ms	374ms	390ms
OpenCPU	500ms	635ms	984ms	1876ms	2700ms

RELATED WORK

In this section we discuss different methods that facilitates building applications using a microservices approach.

Integrate Statistical Analyses

OpenCPU is a system for embedded scientific computing and reproducible research.[8] Similar to the compute service in Kvik, it offers an HTTP API to the R programming language to provide an interface with statistical methods. It allows users to make function calls to any R package and retrieve the results in a wide variety of formats such as JSON or PDF. OpenCPU provides a JavaScript library for interfacing with R, as well as Docker containers for easy installation, and has been used to build multiple applications.[20]. The compute service in Kvik follows many of the design patterns in OpenCPU. Both systems interface with R packages using a hybrid state pattern over HTTP. Both systems provide the same interface to execute analyses and retrieve results. Because of the similarities in the interface to R in Kvik we provide packages for interfacing with our own R server or OpenCPU R servers.

Shiny is a web application framework for R[21] It allows developers to build web applications in R without having to have any knowledge about HTML, CSS, or Javascript. While it provides an easy alternative to build web applications on top of R, it cannot be used as a service in an application that implements the user-interface outside R.

Renjin is a JVM-based interpreter for the R programming language.[1] It allows developers to write applications in Java that interact directly with R code. This makes it possible to use Renjin to build a service for running statistical analyses on top of R. One serious drawback is that existing R packages must be re-built specifically for use in Renjin.

Visualization

Cytoscape is an open source software platform for visualizing complex networks and integrating these with any type of attribute data.[10] Through a Cytoscape App, cyREST, it allows external network creation and analysis through a REST API[7], making it possible to use Cytoscape as a service. To bring the visualization and analysis capabilities to the web applications the creators of Cytoscape have developed Cytoscape.js[22], a JavaScript library to create interactive graph visualizations. Another alternative for biological data visualization in the web browser is BioJS It provides a community-driven online repository with a wide range components for visualizing biological data contributed by the bioinformatics community.[5] BioJS builds on node.js[23] providing both server-side and client-side libraries. In MIxT we have opted to build the visualizations from scratch using sigma.js and d3 to have full control over the appearance and functionality of the visualizations.

Kvik and Kvik Pathwys

We have previously built a system for interactively exploring gene expression data in context of biological pathways.[4] Kvik Pathways is a web application that integrates gene expression data from the Norwegian Women and Cancer (NOWAC) cohort together with pathway images from the Kyoto Encyclopedia of Genes and Genomes (KEGG). We used the experience building Kvik Pathways to completely redesign and re-implement the R interface in Kvik. From having an R server that can run a set of functions from an R script, it now has a clean interface to call any function from any R package, not just retrieving data as a text string but in a wide range of formats. We also re-built the database interface, which is now a separate service. This makes it possible to leverage its caching capabilities to improve latency. This transformed the application from being a single monolithic application into a system that consists of a web application for visualizing biological pathways, a database service to retrieve pathway images and other metadata, and a compute service for interfacing with the gene expression data in the NOWAC cohort. We could then re-use the database and the compute service in the MIxT application.

DISCUSSION

There are different arguments for reusing and sharing microservices over libraries in bioinformatics applications, that would justify the cost of hosting an maintaining a set of distributed microservices. We argue that applications that require large computational or storage resources can benefit from the microservices approach because the applications can share the underlying compute infrastructure between

[18] More details at github.com/fjukstad/kvik/tree/master/r/benchmarks.
[19] Built using the *opencpu-server* Docker image.
[20] opencpu.org/apps.html.
[21] shiny.rstudio.com.
[22] js.cytoscapejs.org.
[23] nodejs.org.

multiple applications and users. This makes it possible to deploy an application on a lightweight system that uses a common service for computation and storage. In addition, benefits such as using different programming languages for a single application, and packaging a microservice as a software container, help to outweigh the operational burden related to using microservices to build applications.

We have used this approach to build different web applications and command line tools, but out of space constraints we only showcase one application in this paper. We have reused the microservices for running statistical analyses and fetch biological metadata, and share these between applications. This makes it possible for multiple applications to use one or more powerful servers for hosting the services. In the case of statistical analyses we simply install the necessary R packages for each application on the compute service and run it as we would for one single application.

Future work

We intend to address few points we aim to address in future work, both in the MIxT web application as well as the supporting microservices. The first issue is to improve the user experience in the MIxT web application. Since it is executing many of the analyses on demand, the user interface may seem unresponsive. We are working on mechanisms that gives the user feedback when the computations are taking a long time, but also reducing analysis time by optimizing the underlying R package. The database service provides a sufficient interface for the MIxT web application. While we have developed the software packages for interfacing with more databases, these haven't been included in the database service yet. In future versions we aim to make the database service an interface for all our applications. We also aim to improve how we capture data provenance. We aim to provide database versions and meta-data about when a specific item was retrieved from the database. One large concern that we haven't addressed in this paper is security. In particular one security concern that we aim to address in Kvik is the restrictions on the execution of code in the compute service. We aim to address this in the next version of the compute service, using methods such as AppArmor[24] that can restrict a program's resource access. In addition to code security we will address data access, specifically put constraints on who can access data from the compute service. We also aim to explore different alternatives for scaling up the compute service. Since we already interface with R we can use the Sparklyr[25] or SparkR[26] packages to run analyses on top of Spark.[12] Using Spark as an execution engine for data analyses will enable applications to explore even larger datasets.

CONCLUSIONS

We have designed an approach for building data exploration applications in systems biology that is based on a microservice architecture. Using this approach we have built a web application that leverages this architecture to integrate statistical analyses, interactive visualizations, and data from biological databases. While we have used our approach to build an application in systems biology, we believe that the microservice architecture can be used to build data exploration systems in other disciplines as well.

[24] wiki.ubuntu.com/AppArmor.
[25] spark.rstudio.com.
[26] spark.apache.org/docs/latest/sparkr.html.

ACKNOWLEDGMENTS

We would like to thank Andrew Bogecho and the System Staff at the School of Computer Science at McGill University for maintaining the compute infrastructure used to run the MIxT system.

This work has been funded by The European Research Council (ERC-AdG 232997 TICE), and The Canadian Cancer Society Research Institute (INNOV2-2014-702940).

REFERENCES

[1] Alexander Bertram. 2013. Renjin: The new R interpreter built on the JVM. In *The R User Conference, useR! 2013 July 10-12 2013 University of Castilla-La Mancha, Albacete, Spain*, Vol. 10. 105.

[2] Peter JA Cock, Tiago Antao, Jeffrey T Chang, Brad A Chapman, Cymon J Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, and others. 2009. Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics* 25, 11 (2009), 1422–1423.

[3] Vanessa Dumeaux, Bjørn Fjukstad, Hans Fjosne E, Jan-Ole Frantzen, Marit Muri Holmen, Enno Rodegerdts, Ellen Schlichting, Anne-Lise Børresen-Dale, Lars Ailo Bongo, Eiliv Lund, and Michael T. Hallett. 2017. Interactions between the tumor and the blood systemic response of breast cancer patients. *Under review* (2017).

[4] Bjørn Fjukstad, Karina Standahl Olsen, Mie Jareid, Eiliv Lund, and Lars Ailo Bongo. 2015. Kvik: three-tier data exploration tools for flexible analysis of genomic data in epidemiological studies. *F1000Research* 4 (2015).

[5] John Gómez, Leyla J García, Gustavo A Salazar, Jose Villaveces, Swanand Gore, Alexander García, Maria J Martín, Guillaume Launay, Rafael Alcántara, Noemi Del Toro Ayllón, and others. 2013. BioJS: an open source JavaScript framework for biological data visualization. *Bioinformatics* (2013), btt100.

[6] R Daniel Kortschak and David L Adelson. 2014. bíogo: a simple high-performance bioinformatics toolkit for the Go language. *bioRxiv* (2014). DOI:https://doi.org/10.1101/005033 arXiv:http://biorxiv.org/content/early/2014/05/12/005033.full.pdf

[7] Keiichiro Ono, Tanja Muetze, Georgi Kolishovski, Paul Shannon, and Barry Demchak. 2015. CyREST: Turbocharging Cytoscape Access for External Tools via a RESTful API. *F1000Research* 4 (2015).

[8] Jeroen Ooms. 2014. The OpenCPU System: Towards a Universal Interface for Scientific Computing through Separation of Concerns. *arXiv preprint arXiv:1406.4806* (2014).

[9] Andrea Sboner, Xinmeng Jasmine Mu, Dov Greenbaum, Raymond K Auerbach, and Mark B Gerstein. 2011. The real cost of sequencing: higher than you think! *Genome biology* 12, 8 (2011), 125.

[10] Paul Shannon, Andrew Markiel, Owen Ozier, Nitin S Baliga, Jonathan T Wang, Daniel Ramage, Nada Amin, Benno Schwikowski, and Trey Ideker. 2003. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research* 13, 11 (2003), 2498–2504.

[11] Christopher L Williams, Jeffrey C Sica, Robert T Killen, and Ulysses GJ Balis. 2016. The growing need for microservices in bioinformatics. *Journal of Pathology Informatics* 7 (2016).

[12] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. 2010. Spark: Cluster Computing with Working Sets. *HotCloud* 10, 10-10 (2010), 95.

Paper 3

V. Dumeaux, B. Fjukstad, H. E. Fjosne, J.-O. Frantzen, M. M. Holmen, E. Rodegerdts, E. Schlichting, A.-L. Børresen-Dale, L. A. Bongo, E. Lund *et al.*, "Interactions between the tumor and the blood systemic response of breast cancer patients," *PLoS Computational Biology*, vol. 13, no. 9, p. e1005680, 2017

PLOS | COMPUTATIONAL BIOLOGY

RESEARCH ARTICLE

Interactions between the tumor and the blood systemic response of breast cancer patients

Vanessa Dumeaux[1,2]*, Bjørn Fjukstad[3], Hans E. Fjosne[4,5], Jan-Ole Frantzen[6], Marit Muri Holmen[7], Enno Rodegerdts[8], Ellen Schlichting[9], Anne-Lise Børresen-Dale[10], Lars Ailo Bongo[3], Eiliv Lund[11☯], Michael Hallett[1,2☯]

1 Department of Biology, Concordia University, Montreal, QC, Canada, **2** School of Computer Science, McGill University, Montreal, QC, Canada, **3** Department of Computer Science, UiT the Arctic University of Norway, Tromsø, Norway, **4** Department of Surgery, St. Olavs University Hospital, Trondheim, Norway, **5** Faculty of Medicine, The Norwegian University of Technology and Science, Trondheim, Norway, **6** University Hospital of North-Norway, Narvik, Norway, **7** Department of Radiology and Nuclear Medicine, Oslo University Hospital, Oslo, Norway, **8** Nordland Central Hospital, Bodø, Norway, **9** Department of Cancer, Oslo University Hospital, Oslo, Norway, **10** Department of Cancer Genetics, Oslo University Hospital, Oslo, Norway, **11** Institute of Community Medicine, UiT the Arctic University of Norway, Tromsø, Norway

☯ These authors contributed equally to this work.
* vanessadumeaux@gmail.com

OPEN ACCESS

Citation: Dumeaux V, Fjukstad B, Fjosne HE, Frantzen J-O, Holmen MM, Rodegerdts E, et al. (2017) Interactions between the tumor and the blood systemic response of breast cancer patients. PLoS Comput Biol 13(9): e1005680. https://doi.org/10.1371/journal.pcbi.1005680

Editor: Florian Markowetz, University of Cambridge, UNITED KINGDOM

Received: March 7, 2017

Accepted: July 7, 2017

Published: September 28, 2017

Data Availability Statement: Data have been deposited at the European Genome-phenome Archive [56] (EGA; https://www.ebi.ac.uk/ega/; accession number EGAS00001001804). Data are available from the NOWAC Data Access Committee for researchers who meet the criteria for access to confidential data based on patient consent and Research Ethics terms.

Funding: We acknowledge funding from the European Research Council (ERC-2008-AdG-232997); the Canadian Cancer Society Research

Abstract

Although systemic immunity is critical to the process of tumor rejection, cancer research has largely focused on immune cells in the tumor microenvironment. To understand molecular changes in the patient systemic response (SR) to the presence of BC, we profiled RNA in blood and matched tumor from 173 patients. We designed a system (MIxT, Matched Interactions Across Tissues) to systematically explore and link molecular processes expressed in each tissue. MIxT confirmed that processes active in the patient SR are especially relevant to BC immunogenicity. The nature of interactions across tissues (i.e. which biological processes are associated and their patterns of expression) varies highly with tumor subtype. For example, aspects of the immune SR are underexpressed proportionally to the level of expression of defined molecular processes specific to basal tumors. The catalog of subtype-specific interactions across tissues from BC patients provides promising new ways to tackle or monitor the disease by exploiting the patient SR.

Author summary

We present a novel system (MIxT) to identify genes and pathways in the primary tumor that are tightly linked to genes and pathways in the patient systemic response (SR). These results suggest new ways to tackle and monitor the disease by looking outside the tumor and exploiting the patient SR.

Institute (INNOV2-2014-702940). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

Introduction

Breast cancer (BC) research has largely focused on understanding the intrinsic properties of the primary tumor in order to therapeutically target key molecular components that drive progression within the tumor epithelial cells [1]. For example, tamoxifen and trastuzumab target the estrogen and human epidermal growth factor receptors (ER, HER2) whose expression levels in tumors define the traditional clinical subtypes of BC. The vast majority of BC-related genomic studies have focused on bulk tumor samples that are expected to be enriched for neoplastic epithelial cells [2]. These efforts have produced subtyping schemes that classify patients into groups based on the similarity of expression of diverse molecular markers and processes [3–9] and generated gene signatures that can predict patient prognosis and benefit from therapy [10–13].

Cancers however are much more than an autonomous mass of epithelial cells. They constitute multicellular systems capable of bidirectional interactions with neighboring non-malignant cells and extracellular components i.e. the tumor microenvironment [14–16]. Tumor-microenvironmental interactions are necessary for tumor progression and drug sensitivity [16, 17] and are becoming better understood [18–21]. In fact, several genomics studies of the BC microenvironment, including our efforts, show that the microenvironment reflects its tumor and harbors prognostic information [22–24]. However, we also recently established that the primary tumor and its microenvironment does not harbor accurate prognostic signals in approximately 20% of BC patients [9]. Specifically, these patients are consistently misclassified by all hallmarks of breast tumors defining tumor epithelial cells (such as proliferation and ER status) and their microenvironment (such as the infiltration of immune cells, angiogenesis and fibroblast activation).

The systemic response (SR) in cancer patients refers here to the perturbations that occur in peripheral blood cells, which include immune effector cells and circulate throughout the body. The fact that a tumor exerts systemic effects (via eg soluble or exosomal factors) may provide an explanation for the clinical observation that patients with one tumor have an increased risk of developing several independent tumors, and that removal of primary cancer improves the survival of patients with distant metastases at the time of diagnosis [25]. In addition, since ER positive (ER+) BC tends to recur as long as 10–15 years after surgical removal of the tumor, it is important to understand systemic factors governing late recurrence and therapeutic approaches that target beyond the tumor site. In fact, there is a rapidly increasing understanding of the various means a tumor employs to favor metastasis in distant organs [26, 27]. For example, an "instigating" BC can exploit the patient SR so that otherwise-indolent disseminated tumor cells become activated [27–32]. The SR has also been investigated in BC at time of diagnosis. Specifically, our recent comparison of blood profiles of BC patients and matched controls yielded a gene signature that reports the presence of BC [33]. This diagnostic signature is specific to BC (i.e. the test classifies women with carcinoma other than breast as negative), and the composition of genes and enriched pathways in the signature suggest that a cytostatic immune-related signal in the SR of patients is associated with the presence of a tumor. Finally, recent evidence demonstrates that engagement of systemic immunity is critical to the process of tumor rejection in genetically engineered mouse models [34].

This study is the first large-scale genomics effort to study the molecular relationships between patient SR and primary tumor. We generated and analyzed expression profiles from peripheral blood and matched tumor cells in 173 BC patients. First, our results highlight how the patient SR is especially relevant to BC immunogenicity. Second, we present a novel tool entitled Matched Interactions across Tissues (MIxT) that starts by identifying sets of genes tightly co-expressed across all patients in each tissue. Then, MIxT identifies which of these

gene sets and pathways expressed in one tissue are associated with gene sets and pathways in the second tissue by determining if their expression patterns in tumor and in the patient SR are tightly correlated. We find that there are very few such associations when all BC are considered. However, we do identify biological processes with significant associations between tumor and patient SR when we stratify our analysis by BC subtype. That is, we identify molecular processes in the tumor that are tightly co-expressed with (different) molecular processes in the SR across patients of a specific subtype. In particular, we detail how several tumor-permissive signals are associated between the tumor and SR of basal BC patients.

Results

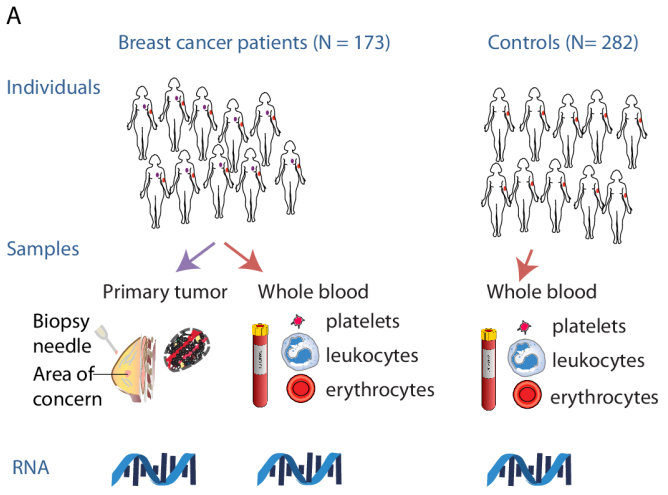
A population genomic resource of blood and matched tumor cells from BC patients

The Norwegian Women and Cancer (NOWAC) is a prospective population-based cohort that tracks 34% of all Norwegian women born between 1943–57. In collaboration with all major hospitals in Norway, we collected blood samples and matched tumor from women with an abnormal lesion, at the time of the diagnostic biopsy or at surgery, before surgery and any treatment (N ~ 300, S1 Text). RNA preservation for blood samples obtained followed our methodology previously described [33, 35] and detailed in S1 Text. RNA profiles from blood and tumor cells were generated using Illumina Beadarrays and data were processed following careful procedures (S1 Text, S1A Fig). After quality control, our study retained matched blood (SR) and tumor profiles of 173 BC patients diagnosed with invasive ductal carcinoma, and blood profiles of 282 control women (ie. women with no history of cancer with the exception of basal-cell and cervical carcinoma, which are both very common; Fig 1A). The controls are used to determine what constitutes a "normal" SR. BC patients and controls are comparable in terms of age, weight and menopausal status (Fig 1B). Several groups including ours have defined intra- and inter- individual variability of blood gene expression in healthy individuals [35–38]. All together, these studies demonstrate that intra-individual changes that can occur between blood draws are strikingly smaller than the variation observed among samples collected from different individuals. In this study, most women were 50 year-old or older and postmenopausal at time of sampling. Each profile measures the expression of 16,782 unique genes (S1 Text, S1A Fig). Almost all BC (95.4%) are early-stage disease (stage I or II).

Transcriptional fingerprint of BC subtypes is not the predominant signal in the patient SR

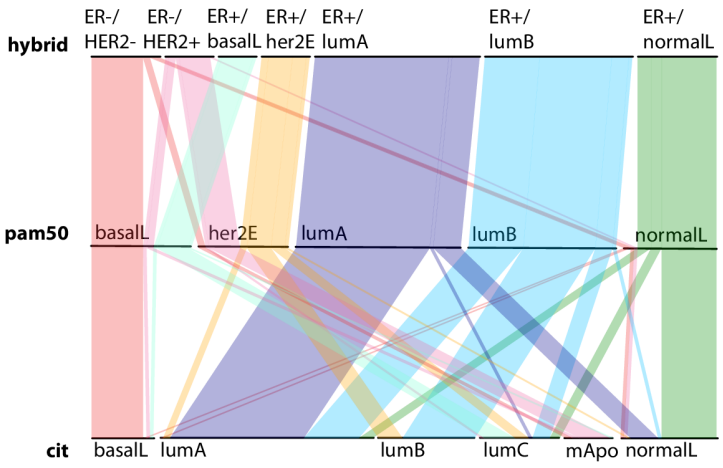
Several tumor RNA-based subtyping tools were applied including PAM50 [5] that defines the intrinsic subtypes including luminal A (lumA), luminal B (lumB), normal-like (normalL), basal-like (basalL), and her2-enriched (her2E). The hybrid subtyping scheme partitions ER + tumors according to their intrinsic subtype and partitions ER- tumors according to their HER2 status [9] (S1 Text, S1B and S1C Fig). In our dataset, all intrinsic luminals (lumA and lumB) and most normalL tumors (85.2%) are ER+; however, ~40% of basalL and ~50% of her2E BC are ER+ (Fig 1C, S1 Table). We also applied the Cartes d'Identité des Tumeurs (CIT) [8] subtyping scheme, which includes a 'molecular-\ apocrine' (mApo) subtype enriched for ER-/HER2+ tumors (78.6%) and the highly immunogenic ER+ luminal C (lumC) subtype enriched for ER+/basalL (39.1%). Fig 1C and S1 Table depict the relationships between these three schemes.

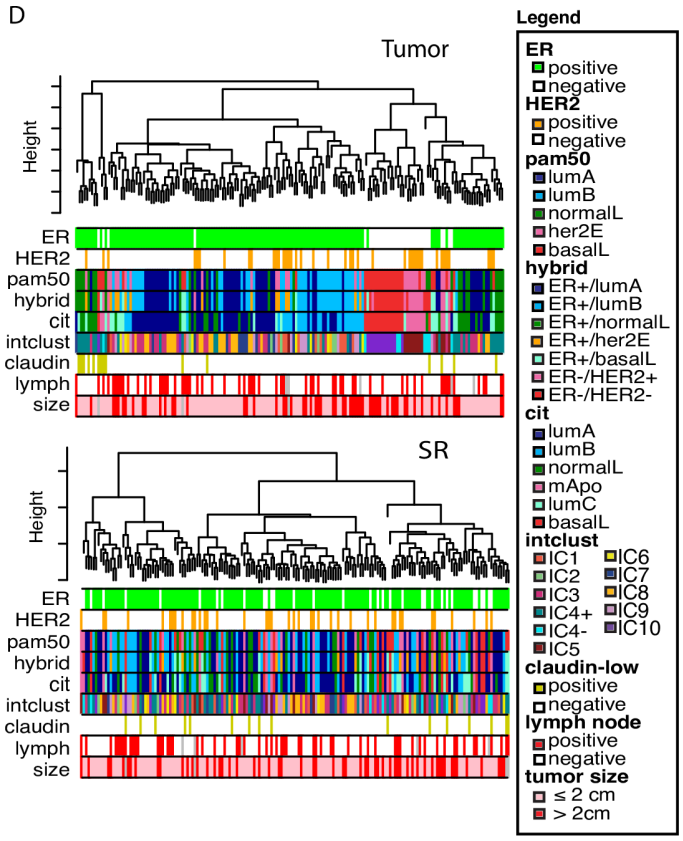
Although the IntClust (IC) subtyping scheme [6] is based on gene expression and DNA copy number profiles simultaneously, subtypes can be inferred using a reported RNA-based



B

	Breast cancer patients	Controls
N samples	173	282
Age (< 50)	20 (11.6 %)	41 (14.5 %)
Age (> 50)	153 (88.4 %)	241 (85.5 %)
Age: mean (range)	56.4 (43-104)	56.1 (43-67)
Weight (< 70)	75 (43.3 %)	159 (56.4 %)
Weight (> 70)	71 (41.0 %)	120 (42.5 %)
Weight: mean (range)	71.6 (50-150)	70.2 (41-120)
Postmenopausal	138 (79.8 %)	269 (95.4%)
Premenopausal	13 (7.5%)	13 (4.6 %)





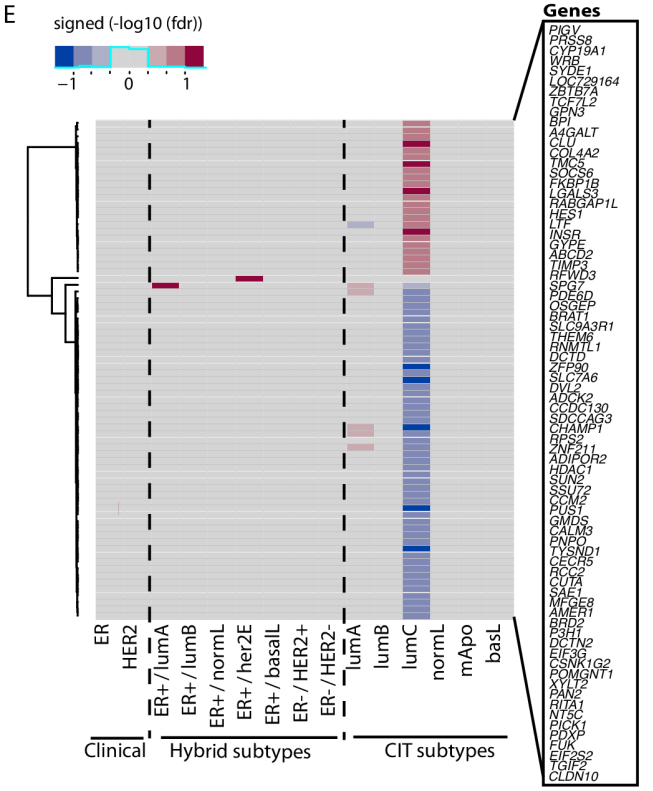


Fig 1. Individual characteristics and SR markers of BC subtypes. (A) Collection of biospecimen from BC patients and controls. (B) Individual characteristics of BC patients and controls. (C) Parallel plot displaying the repartition of BC patients across RNA-based subtyping schemes. (D) Sparse hierarchical clustering of BC patients based on genes expressed in tumor (upper) and the patient SR (lower). Clinicopathological and subtypes attributes are presented below the dendrogram. (E) Significant gene markers of subtypes in SR (false discovery rate, fdr $\leq$ 0.2). Blue and red shade correspond to under- and over- expression of the marker in a given subtype vs the others, respectively. Shading is proportional to the level of significance of the gene marker.

https://doi.org/10.1371/journal.pcbi.1005680.g001

surrogate algorithm [7, 39]. S1 Table reports when subtypes from other schemes are enriched in each IC subtype. Most notably, IC1 and IC9 are enriched for CIT lumB; IC3, IC7 and IC8 are enriched for lumA; IC4+ is enriched for normalL and at lesser extent CIT lumC, IC5 enriched for mApo-her2E-HER2+, and IC10 enriched for basalL and ER-/HER2-. IC2, IC4-, and IC6 include very few patients (n $<$ 10) and were therefore not further considered in our downstream analyses.

Restricting our attention to tumor profiles, we performed sparse hierarchical clustering with complete linkage using a permutation approach to select the tuning parameter that weights each gene to compute the dissimilarity matrix [40]. The resulting clusters were strongly associated with BC subtypes for all three RNA-based schemes (Fig 1D upper), which confirms that the transcriptional fingerprint of BC subtypes are also ubiquitous in our tumor samples. When restricting our attention to SR profiles, this unsupervised analysis does not identify patient clusters enriched for any given subtype across the three schemes (Fig 1D lower), suggesting that the transcriptional fingerprint of BC subtypes is not the predominant signal in the patient SR.

Univariate gene markers are identified in the patient SR for one immunogenic BC subtype

We then asked if there are genes in the patient SR whose expression covaries with the state of the pathological variables ER and HER2 measured in the primary tumor. Although both are key drivers in BC, neither was found to be associated with individual gene expression changes in the patient SR (limma, linear models for microarray data, false discovery rate, fdr $\leq$ 0.2, Fig 1E; S1 Text). Similarly, we asked if there are genes in the SR that are markers of tumor subtype (n patients $>$ 10). For the intrinsic, hybrid, and IntClust subtypes, only the ubiquitin ligase *RFWD3* is highly expressed uniquely in the SR of lumA patients, and *TIMP3*, an inhibitor of matrix metalloproteinases, is highly expressed uniquely in ER+/her2E patients (Fig 1E, S2 Fig). For the CIT subtypes [8], we found 70 univariate gene markers in the SR of patients of the lumC subtype. The genes are primarily involved in general cellular processes such as protein processing or transcription in blood cells (fdr $\leq$ 0.2, Fig 1E, S3 Fig). The lumC subtype is defined by strong activation of several immune pathways at the site of ER+ tumor (i.e. antigen presentation and processing pathway, hematopoietic cell lineage, NK cell mediated cytotoxicity, T-cell receptor signaling and Toll-like receptor signaling) [8], suggesting that the SR is informative in cases where the primary tumor exhibits strong immune properties.

Systems-level analysis reveals tissue-specific molecular processes

To compare genome-wide molecular changes in tumor and SR across patients, we used WGCNA-based clustering to define sets of tightly co-expressed genes (termed modules) in tumor and blood, respectively [41] (S1 Text). Briefly, we opted for a distance measure based on topological overlap, which considers the correlation between two genes and their respective correlations with neighboring genes [42] (S1 Text). The WGCNA cut and merge routine [43] after clustering identified 19 and 23 modules in the patient tumor and SR, respectively (S4 Fig;

S1 Text). Each of these modules can be considered as a unique and stable pattern of expression shared by a significant number of genes.

Modules of the primary tumor are enriched for genes from a broad range of BC hallmarks including angiogenesis (salmon module), extracellular matrix reorganization (greenyellow), proliferation (green), and immune response (brown and darkturquoise) (S2 and S3 Tables, S1 Text). For example, the proliferation tumor module is enriched for mitotic cell cycle-related genes (green, n = 1064 genes; weight01 Fisher test [44], p-value < 2e-17) including the well-known marker of proliferation MKI67, 12 serine/threonine kinases that are used in the calculation of the mitotic kinase score (MKS) [45], and several components of the Minichromosome Maintenance Complex (MCM).

Modules of the patient SR are often enriched for genes involved in either general cellular processes such as translation (black) and transcription (grey60), or immune-related processes such as inflammatory response (brown, green), B-cell response (saddlebrown), innate immune response (greenyellow) (S4 and S5 Tables). Thus, seven SR modules are enriched in genes that are specifically expressed in immune cells [46] ("iris" signature set in S5 Table; Fisher's Exact Test FET fdr < 0.05).

We constructed a web-based system to visualize gene expression networks, heatmaps and pathway analyses of the modules in each tissue at http://mixt-blood-tumor.bci.mcgill.ca. In a network, genes are represented by nodes (colored by their module membership) that are connected by edges whose length corresponds to their level of co-expression across patients [47]. When selecting only strong gene-gene correlations (topological overlap > 0.1) and removing isolated nodes, the SR network has ~20% more genes than the tumor network (Fig 2A and 2B). Moreover, the SR network has approximately twice as many edges (89,465 connections between genes) than the tumor network (50,617 connections between genes). Thus, the underlying patterns of expression of the tumor genes (and modules) are more dissimilar from each other than the patterns of expression of the SR genes (and modules). In both tissues, the edges that span between modules reflect natural overlaps between cellular process (Fig 2A and 2B). For example in tumors, angiogenesis-related genes of the salmon module are strongly co-expressed with genes of the greenyellow module involved in extracellular matrix remodeling. In blood, modules enriched for genes involved in general cellular processes such as translation (black), RNA processing (violet), and RNA splicing (darkred) are also heavily connected to each other.

Several processes in the SR are differentially expressed in patients with HER2+, lumC or large tumors

We first investigated the relationships between the expression pattern of each module and patient clinicopathological attributes. Towards this end, each gene of a module is used to rank the patient samples (S1 Text). In particular, the sum of gene ranks (ranksum) for each patient provides a linear ordering of the patient samples. Association tests then compare the ranksum values of patients with the attribute of interest eg tumor subtype (S1 Text).

When we consider tumor modules, the expression pattern of the green module (S5A Fig), previously established to be enriched for proliferation-related genes (S2 Table), ranks basalL, her2E and lumB tumors significantly higher than lumA and normalL tumors (ANOVA p-value < 1e-34, S5B Fig). In fact, we observe that the expression pattern of nearly every module is associated with BC subtype (15 of 19 modules, Fig 2C, fdr ≤ 0.15). Moreover, many tumor modules are associated with the proliferative state of the tumor encoded into the MKS score [45] (Pearson correlation, fdr ≤ 0.15) or with ER status (ER+ vs ER-, t-test, fdr ≤ 0.15), two variables that are strongly embedded in the definition of BC subtypes (Fig 2C). These results are consistent with our previous claim that patient subtype is a predominant signal in the

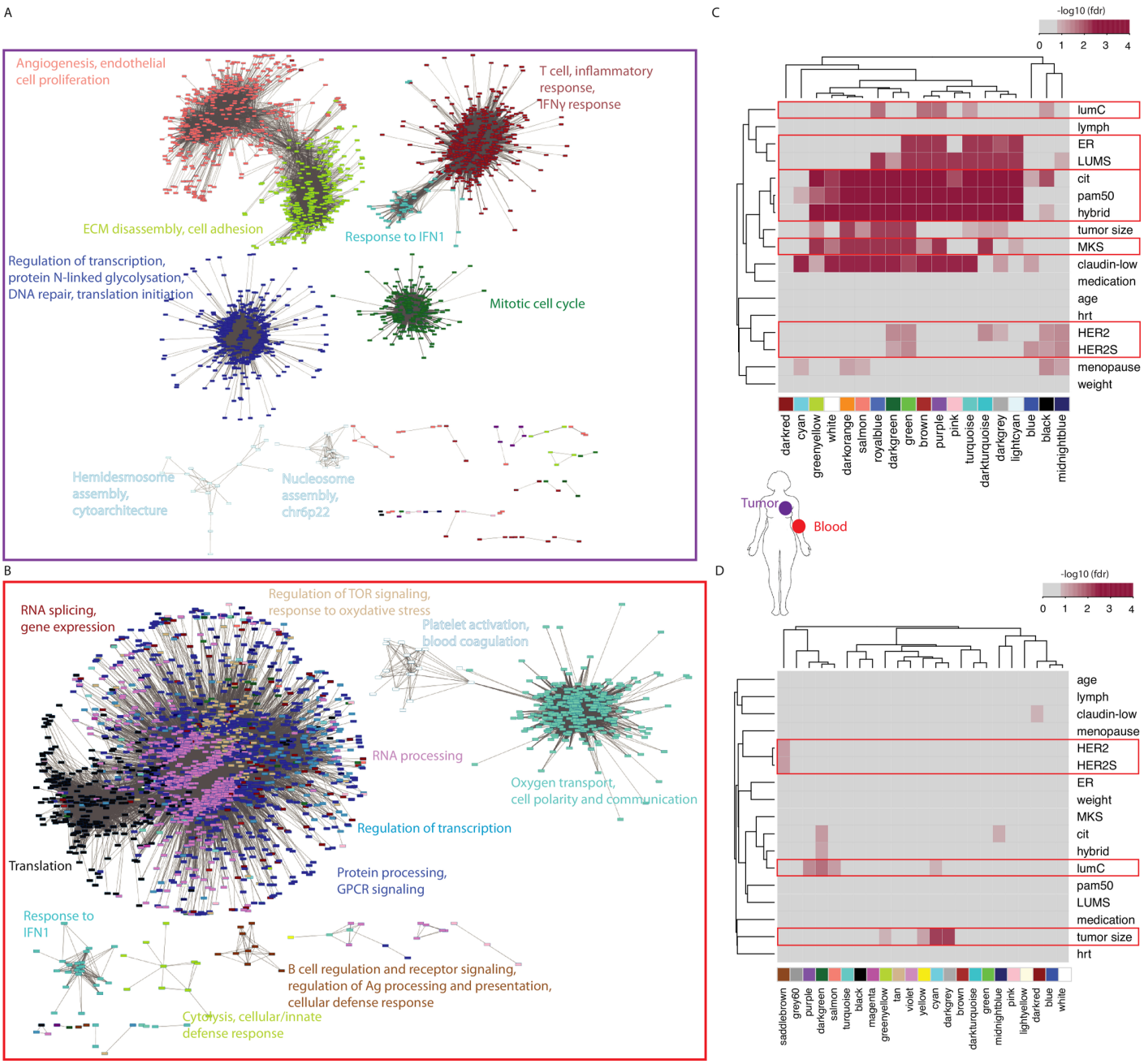


Fig 2. Gene co-expression networks, modules and associations with clinicopathological attributes of BC patients. (A) Network visualization using the edge-weighted spring embedded layout from Cytoscape (v3.2.1) including the top gene connections (topological overlap > 0.1) in tumor. Each node (gene) is color-coded by the module to which it belongs, Keywords representing top pathway enrichments (biological processes) are indicated for each module. (B) Network visualization including the top gene connections in the patient SR. The legend follows Fig 2A. (C) Associations between tumor modules and clinicopathological attributes of patients. Associations were estimated using Pearson correlation (Student's p) or ANOVA. Shading is proportional to $-\log_{10}$(fdr) of the associations (fdr $\leq$ 0.15). HER2S: HER2 score; LUMS: luminal score; MKS: Mitotic kinase gene expression score; hrt: hormone replacement therapy (D) Associations between SR modules and clinicopathological attributes of patients. The legend follows Fig 2C.

https://doi.org/10.1371/journal.pcbi.1005680.g002

primary tumor. Several tumor modules are associated with HER2 status of the tumor, however there are fewer such modules (n = 6) when compared with the proliferative state or ER status

of the tumor (Fig 2C), suggesting that transcriptional fingerprint of HER2 is not as ubiquitous in tumor samples. A small number of modules are associated with the lumC subtype, including the brown module enriched for T-cell and inflammatory response genes (S2 Table). This is again consistent with the fact that this is a highly immunogenic subtype [8] (lumC versus not lumC, t-test, fdr ≤ 0.15, Fig 2C).

HER2 status, the lumC subtype and tumor size are all associated with modules of the patient SR (Fig 2D, t-test fdr ≤ 0.15). Although we did not find univariate gene markers in blood associated with HER2 status, the saddlebrown SR module is significantly underexpressed in patients with HER2+ tumors compared to other BC subtypes and controls (fdr = 0.07, S6A Fig) and is enriched for genes involved in B-cell receptor signaling and proliferation (including *BLK*, *CXCR5*, *CD19*, *CD79A*, *CD79B* and *FCRL5*; S4 and S5 Tables). Four SR modules are associated with the immunogenic lumC subtype; one of these modules are also associated with tumor size (Fig 2D, S6B and S6C Fig). Among the 70 univariate gene markers in blood of lumC tumors identified earlier, 31 are included in the darkgreen SR module predominantly underexpressed in lumC patients in comparison to other BC subtypes (fdr = 0.02, S6B Fig). In fact, all four SR modules associated with the lumC subtype are underexpressed compared to other BC subtypes and control samples (S6B and S6C Fig). This includes the purple module highly enriched for genes involved in T-cell (thymus) homing (*CCR7*, *LTA*, *LTB*, *VEGFB*, *HAPLN3*, *SLC7A6*, *SIRPG*, *BCL11B0*) and activation (*CD47*, *TNFRSF25*, *MAL*, *LDLRAP1*, *CD40LG*) which are underexpressed in lumC patients (fdr = 0.04, S6B Fig). Genes in the cyan modules are also found underexpressed in patients with large ($>$ 2cm) tumors compared to other BC patients and controls (Fig 2D, S6C Fig). Finally, specifically for patients with large tumors, both the darkgrey module, which is enriched for *MYC* target genes, and the greenyellow module, which is enriched for genes involved in the lymphoid cell-mediated immunity (including *GZMH*, *GZMB*, *GZMM*, *KLRD1*, *PRF1*, *KLRG1*, and *GNLY*; S4 and S5 Tables), are underexpressed compared to the remaining BC patients and controls.

Together these results indicate that distinct SR are detected in BC patients with HER2+, lumC and/or large tumors, and that overall the patient immune response is underexpressed compared to patients of other subtypes and controls. These results also highlight the importance of distinct immune components for each of these disease groups. In particular, patients with HER2+ tumors exhibit low expression of genes specifically expressed in B-cell compared to patients with other BC subtypes. Patients with lumC tumors exhibit low expression of genes involved in T-cell homing and function compared to patients with other BC subtypes. Patients with large tumors (>2cm) exhibit low expression of genes involved in lymphoid cell-mediated immunity compared to patients with smaller tumors.

Our Matched Interactions across Tissues (MIxT) approach explores biological processes that interact between tissues

Our analysis to this point identified modules within each tissue independently. Our focus here is on the relationships between tissues by asking if specific biologies in one tissue are correlated with (possibly distinct) biologies in the second tissue. To do this, we constructed a software entitled MIxT (Matched Interactions across Tissues) that contains the computational and statistical methods for identifying and exploring associations between modules across tissues (http://mixt-blood-tumor.bci.mcgill.ca).

Using MIxT, we first ask if genes that are tightly co-expressed in the primary tumor are also tightly co-expressed in the SR, and vice versa (Fig 3A, S1 Text) by investigating the gene overlap between tumor and SR modules (Fisher's Exact Test FET, fdr < 0.01). Genes that retain strong co-expression across patients regardless of tissue type are likely to be involved in the

same biological functions in both tissues as a "system-wide" response to the presence of the disease (even if patterns of gene expression across tissues might differ).

Most modules, regardless of tissue, have significant overlap with three to five modules in the other tissue (Fig 3A). In some cases, it appears that a single (large) module in one tissue is in large part the union of several smaller modules from the other tissue. For example, the brown tumor module has 2765 genes including many involved in immune-related processes (T-cell costimulation, the IFN-gamma pathway and inflammation, S2 and S3 Tables). All of these genes/processes show very strong co-expression in the tumor however, in the SR, these genes divide into four distinct patterns of co-expression (Fig 3A), captured by four different modules: brown (inflammation), greenyellow (cytolysis and innate immune response), saddlebrown (B-cell) and pink (TNFA inflammatory response) (S4 and S5 Tables).

Of note, MIxT identifies three modules in each tissue (SR and tumor) that do not have significant overlap with any module in the other tissue (Fig 3A). For tumors, this includes the purple module enriched for genes involved in estrogen response, the lightcyan module enriched for genes involved in hemidesmosome assembly and cytoarchitecture, and the greenyellow module enriched for genes involved in ECM organization (Fig 3A, S2 and S3 Tables). For the SR, this includes the turquoise module enriched for genes expressed in erythrocytes and involved in hemoglobin production, the purple module enriched for genes in translational termination, and the green module enriched for genes involved in inflammation and specifically expressed in myeloid cells (Fig 3A, S4 Table). This suggests that these processes and responses are either specific to a tissue type (eg ECM organization specific to tumor, and hemoglobin production specific to blood cells) or that the co-expression of genes involved in a defined process is unique to a particular tissue (eg genes specifically co-expressed in peripheral myeloid cells).

There is only one instance where a single tumor module has significant overlap with only a single SR module: darkturquoise modules of size = 86 and 97 genes in SR and tumor, respectively with 50 common genes, including 20 involved in the type 1 IFN signaling pathway (S2 and S4 Tables). Although these two "mirrored" modules share many genes, their patterns of expression are significantly different between the two matched tissues (Fig 3B, correlation between ranksums p-value > 0.05; S1 Text), hinting at a non-concordant expression of the local (in tumor) and systemic (in blood) IFN-1 mediated signals.

MIxT identifies novel interactions between processes across tissues within specific subtypes

Whereas the previous section considers interactions defined by a large number of shared genes between a tumor and a SR module, we also examined more general notions of interactions in MIxT. Here we identify tumor and SR modules that have similar expression patterns (ie both modules linearly order the patients in very similar manner in both tissues) but do not necessarily share any genes in common. More specifically, MIxT derives estimates of significance for interactions using a random permutation approach based on the Pearson correlation between ranksums of gene expression in modules across tissues (S1 Text). This type of interaction detects a biological process or response in the primary tumor that is tightly correlated (or anti-correlated) with a (possibly distinct) biological process or response in the SR, and vice versa. The specific expression pattern in the tissues allows us to then postulate the functional nature of the interaction across tissues.

MIxT identified only one tumor module (of 19) that interacts with only a single SR module (of 23) across all patients (MIxT statistic; p-value < 0.005). The paucity of pan-BC interactions across tissues suggest the need to stratify by patient subtype. After stratification for each of the

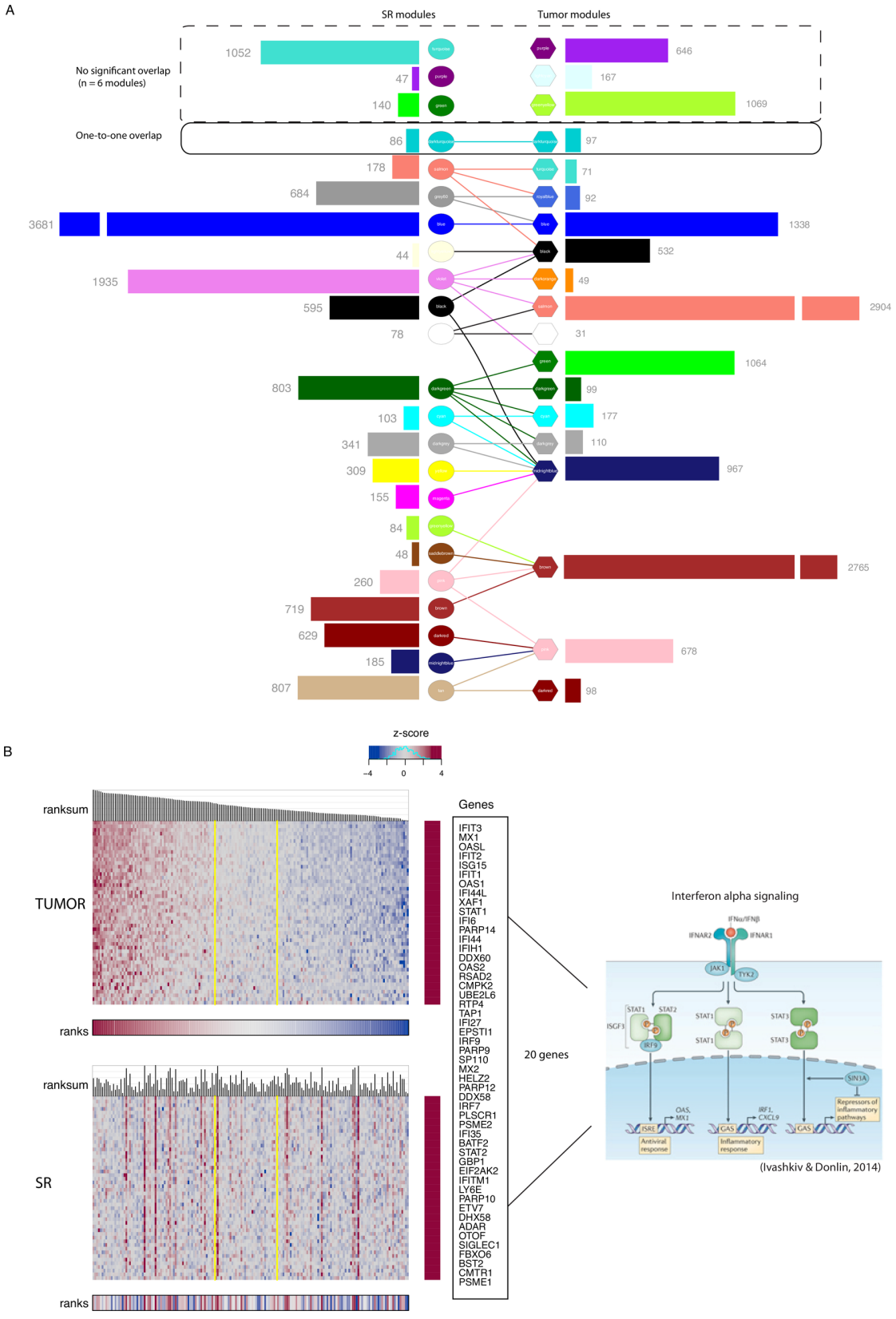

A
SR modules
Tumor modules
No significant overlap
(n = 6 modules)
One-to-one overlap
1052
646
47
167
140
1069
86
97
178
71
684
92
3681
1338
44
532
1935
49
595
2904
78
31
1064
803
99
103
177
341
110
309
967
155
84
48
2765
260
719
629
678
185
807
98
B
z-score
−4
0
4
ranksum
TUMOR
ranks
ranksum
SR
ranks
Genes
IFIT3
MX1
OASL
IFIT2
ISG15
IFIT1
OAS1
IFI44L
XAF1
STAT1
IFI6
PARP14
IFI44
IFIH1
DDX60
OAS2
RSAD2
CMPK2
UBE2L6
RTP4
TAP1
IFI27
EPSTI1
IRF9
PARP9
SP110
MX2
HELZ2
PARP12
DDX58
IRF7
PLSCR1
PSME2
IFI35
BATF2
STAT2
GBP1
EIF2AK2
IFITM1
LY6E
PARP10
ETV7
DHX58
ADAR
OTOF
SIGLEC1
FBXO6
BST2
CMTR1
PSME1
20 genes
Interferon alpha signaling
IFNα/IFNβ
IFNAR2
IFNAR1
JAK1
TYK2
STAT1
STAT2
ISGF3
IRF9
STAT1
STAT1
STAT3
STAT3
SIN1A
Repressors of inflammatory pathways
OAS, MX1
IRF1, CXCL9
ISRE
GAS
GAS
Antiviral response
Inflammatory response
(Ivashkiv & Donlin, 2014)

Fig 3. Modules size and overlap in their gene composition across tissues. (A) Histograms depicting number of genes composing modules in each tissue. Edges between modules indicate significant overlaps in gene composition (Fisher exact test, fdr < 0.01). (B) Expression heatmaps of the 47 genes included in both darkturquoise modules in tumor (upper) and SR (lower). Patients in both heatmaps are linearly ordered based on their ranksum of gene expression in tumors. Yellow vertical lines delimit the region of Independence (ROI_{95}) in tumor that contains 95% of randomly generated samples. Twenty genes out of the 47 common genes are involved in the type 1 IFN signaling pathway (IFN alpha signaling pathway is depicted on the right).

https://doi.org/10.1371/journal.pcbi.1005680.g003

five subtyping schemes (clinical, PAM50, hybrid, CIT, and Intclust) (Fig 4A), we identified 53 interactions involving 15 tumor modules and 19 SR modules (MIxT statistic; p-value < 0.005; Fig 4B, S7 Fig). Tumor and SR modules are indicated in columns and rows of Fig 4B, respectively. A non-empty cell corresponds to a significant interaction with color used to indicate in which subtype the association is found, grouping together similar subtypes across schemes (eg basalL tumors of the pam50 and CIT schemes). Nearly all interactions are significant in only a single subtype (four exceptions indicated by orange arrows, Fig 4B). For some subtypes, a single stimulus in the tumor affects several biological processes in the patient SR. For example, within the ER+/HER2- subtype and only within this subtype, the pink tumor module, enriched for genes involved in alternative splicing, is associated with three SR modules, enriched for a diverse range of biological processes (orange rectangle in Fig 4B).

Immune activity at the tumor site is associated with inflammatory SR in opposite ways for two distinct subtypes

The brown tumor module, which is enriched for genes involved in immune processes (S2 Table), has several interactions with SR modules across several subtypes (orange rectangle in Fig 4B). This includes interactions specific to normalL, lumB and IC9 but also several distinct interactions within the ER-/HER2- and basal subtypes. This suggests that immune signals expressed in tumor are associated with changes in expression of different molecular processes in the patient SR for a broad range of subtypes.

As alluded to earlier, only a few interactions are significant in two distinct subtypes simultaneously. For example, the brown tumor module is associated with green SR module in both ER-/HER2- and lumB although the directionality of the association differs between the two cases. More specifically, patients with high ranksums in the brown tumor module have low ranksums according to the green SR module, if the patient is of the ER-/HER2- subtype (Fig 5A, 5C and 5E, MIxT statistic, p-value = 0.004). At the same time, patients with high ranksums in the brown tumor module have high ranksum with respect to the green SR module, if the patient is of the lumB subtype (Fig 5B, 5D and 5F MIxT statistic, p-value < 0.004). In this manner the direction of correlation between the biological processes of the brown tumor module and of the green SR module is determined by the subtype of the patient.

For the brown tumor module in both subtypes, patients with a high ranksum (on the left of the ordering in Fig 5B or 5C for both subtypes) have the strongest immune signals in the tumors. This is because most of the immune-related genes in this brown module (within the red sidebar in Fig 5B and 5C, S3 Table) have highest expression in these patients. This includes genes involved in T-cell stimulation (incl. *CD3*, *CD4*, *CD5*, *ICOS*, several *HLA-DR*, *-DP*, *-DQ*), IFNɣ signaling (*IFNG*, *IRF1-5*, *ICAM1*, *IFI30*, *HLA-A -B -C*) and inflammation (incl. several interleukins, chemokines). For the green SR module in both subtypes, a high ranksum indicates an inflammatory SR (patients on the right in Fig 5E for ER-/HER2-, and patients on the left in Fig 5F for lumB). This is because almost every inflammation-related genes (incl. I*FNAR1*, *IL15*, *TLR2*, *IL18RAP*, *RNF144B)*, and B-cell proliferation genes (incl.

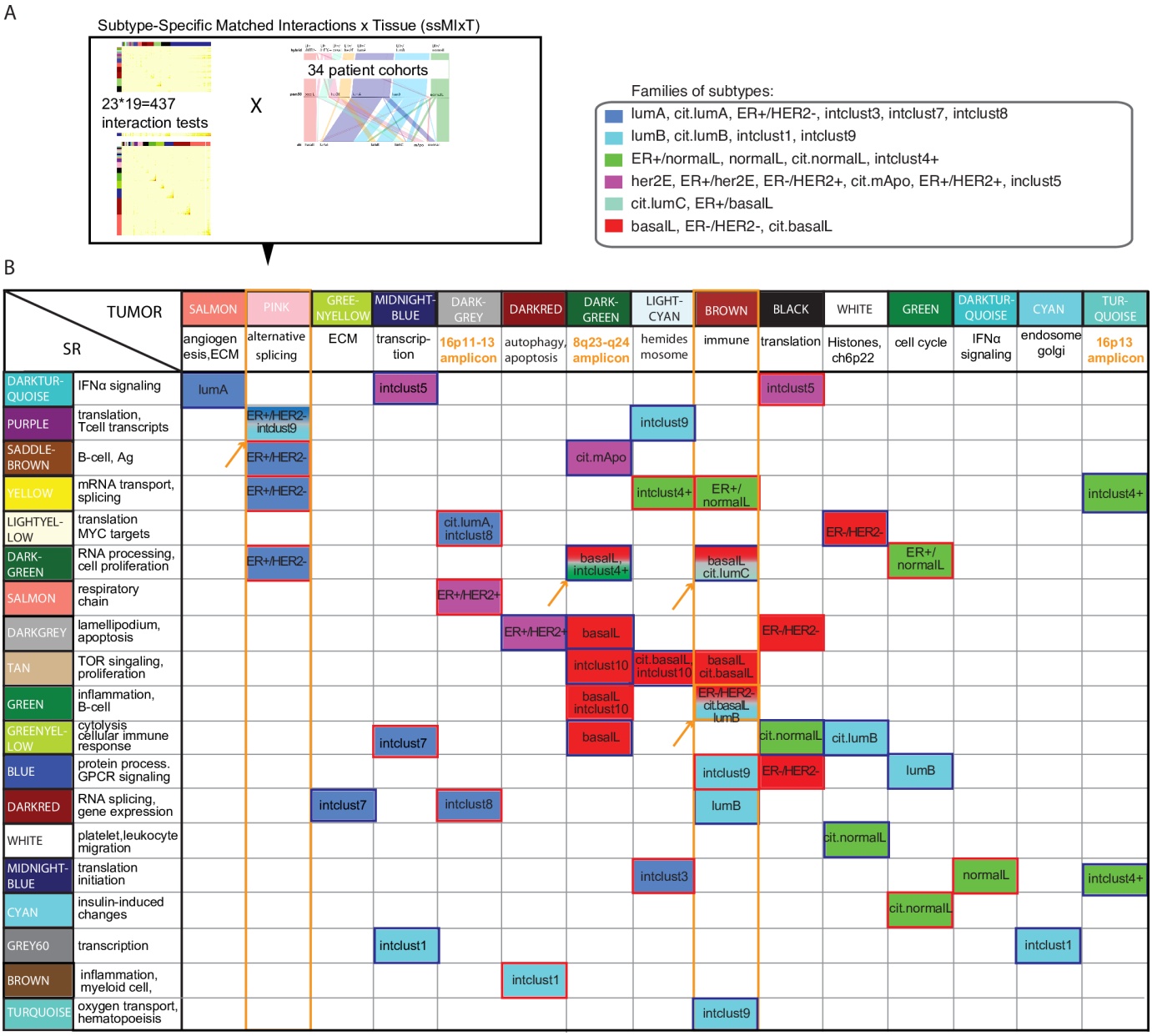


Fig 4. Subtype-Specific Matched Interactions across Tissue (ssMIxT). (A) Schematic of ssMIxT analysis (B) Significant associations between modules in SR and tumor from BC patients by subtype (MIxT statistic, p-value < 0.005). SR and tumor modules with top pathway enrichment keywords are presented in rows and columns, respectively. Subtype(s) in which the significant associations are found are indicated in the table. Blue and red borders correspond to negative and positive correlations between ranksums, respectively. Findings discussed in the text are highlighted in orange.

https://doi.org/10.1371/journal.pcbi.1005680.g004

BCL6, *IL13RA1*, *MIF*, *IRS2*) (within the red sidebar in Fig 5E and 5F, S5 Table) have highest expression in these patients.

Thus, ER-/HER2- patients with low immune activity at the tumor site have a high inflammatory SR (right side of Fig 5C and 5E). In fact, the level of the inflammatory response in these BC patients is higher than healthy controls (Fig 5I, t-test, $p < 0.001$). However, for the lumB subtype, the relationship between tumor and SR is reversed. Here, it is the patients that

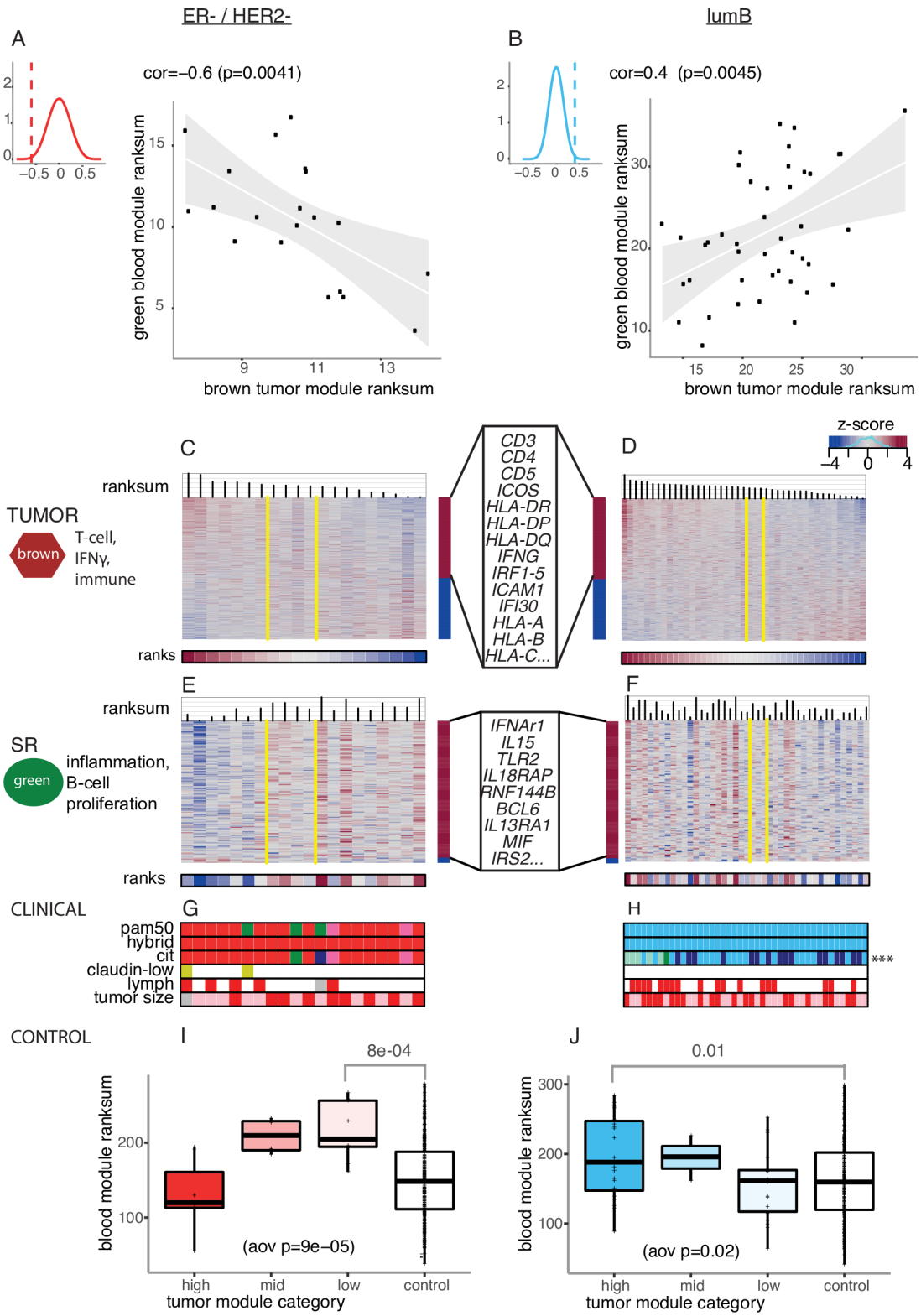


Fig 5. Association between the brown tumor and green SR module for two distinct subtypes. (A) Scatter plot of ranksums of the brown tumor module and the green SR module in ER-/HER2- patients. The top corner depicts the background distributions of the correlations coefficients between ranksums of every modules pairs across tissues in ER-/

HER2- patients. (B) Scatter plot of ranksums of the brown tumor module and the green SR module in ER+/lumB patients. Legend follows Fig 5A (C) Expression heatmap of genes in the brown tumor module in ER-/HER2- patients. Patients are linearly ordered based on the ranksum of gene expression in the brown tumor module. Yellow vertical lines delimit the ROI_{95} in tumor that contains 95% of the randomly generated samples. Genes that are positively and negatively correlated with the ranksum are represented in the right sidebar colored in red and blue, respectively. Top pathway enrichment keywords and representative genes are indicated on the left and right of the heatmap, respectively). (D) Expression heatmap of genes in the brown tumor module in ER+/lumB patients. Legend follows Fig 5C. (E) Expression heatmap of genes in the green SR module in ER-/HER2- patients. Legend follows Fig 5C. Top pathway enrichment keywords and representative genes are indicated on the left and right of the heatmap, respectively. (F) Expression heatmap of genes in the green SR module in ER+/lumB patients. Legend follows Fig 5E. (G) Clinical characteristics of ER-/HER2- patients ordered by the ranksum of gene expression in the brown tumor module. Legend follows Fig 1D. (H) Clinical characteristics of ER+/lumB patients ordered by the ranksum of gene expression in the brown tumor module. Legend follows Fig 1D. Asterisks represent the level of significance of the associations between the gene ranksums for the brown tumor module and clinicopathological attributes of patients. Associations were estimated using ANOVA (fdr < ***0.01). (I) Distribution of ranksums for ER-/HER2- patients and controls induced by the expression of genes in the green SR module. Patients are grouped according to the ROI_{95} brown tumor module category as defined in Fig 5C. aov: analysis of variance (J) Distribution of ranksums for ER+/lumB patients and controls induced by the expression of genes in the green SR module. Patients are grouped according to the ROI_{95} brown tumor module category as defined in Fig 5D.

https://doi.org/10.1371/journal.pcbi.1005680.g005

have high immune activity at the tumor site that have a high inflammatory SR (left side Fig 5D and 5F). In fact, the CIT subtyping scheme calls these patients on the left side as belonging to the lumC subtype (Fig 5H), the highly immunogenic ER+ subtype. In these lumB patients the inflammatory response is also higher than in healthy controls (t-test, p-value < 0.01; Fig 5J).

Altogether these results indicate that a high inflammatory SR is observed in both ER-/HER2- and ER+/lumB patients but increase in systemic inflammation is associated with distinct immune activity at the tumor site depending on subtype.

Expression of genes in known BC amplicons is associated with concomitant changes in the patient SR for defined subtypes

Three tumor modules are enriched for genes within amplicons prevalent in BC [48] (highlighted in orange in Fig 4B, S3 Table). Two modules, the darkgrey and turquoise tumor modules, contain 68 genes (of 110) and 48 genes (on 71) located within the 16p11-13 amplicon highly prevalent in luminal tumors [48], respectively (S3 Table). The darkgrey module interacts with two distinct SR modules for the lumA and ER+/HER2+ subtype, respectively (S8A and S8B Fig). Tumors of both subtypes that over-express genes in the darkgrey module (left hand side S8C and S8D Fig) are likely amplified in 16p13. In these patients, the presence of this amplification is correlated with changes in expression of specific processes within the patient SR and these processes are distinct depending on subtype (S8E and S8F Fig, $p < 0.005$ in both cases). S8G and S8H Fig depicts associations between the presence of this amplification and patient clinico-pathological attributes. For example, in ER+/HER2+ patients (S8H Fig), the presence of 16p13 amplification is correlated with the luminal score of the tumor. In the lumA subtype, patients with the highest expression of the lightyellow SR module are significantly different than healthy controls (S8I Fig), and in the ER+/HER2+ subtype, patients with the lowest expression of the salmon module are significantly different than healthy controls (S8J Fig).

The third module enriched for genes involved in BC amplifications is the darkgreen tumor module. This module contains 43 (of 99) genes within the 8q23-24 amplicon prevalent in basal and her2E tumors [48] (S3 Table). Most associations with patient SR modules are specific to the basalL subtype (Fig 4B) and again suggest that basalL tumors that harbor this amplification have concomitant changes in expression of specific molecular processes in patient SR.

A fully integrated view of molecular changes correlated between tumor and SR in basalL patients

Approximately one-fourth of the interactions identified by MIxT are specific to ER-/HER2-, IC10 and basalL subtypes, indicating that the tumor and SR interact strongly in this family of BCs (Fig 4B). We study two tumor modules in greater depth here: the brown immune-enriched module and the darkgreen 8q-enriched module, and their interactions with SR modules in basalL patients (Fig 6A–6C). Here the brown tumor module interacts with one (tan) SR module enriched for genes involved in TOR signaling and cell proliferation (Fig 6A and 6B). BasalL patients with low immune activity at their tumor site (right side of brown tumor module) have low expression of the tan SR module, and this expression is significantly lower than healthy controls (boxplots in Fig 6B, t-test $p < 0.0005$).

The darkgreen tumor module interacts with four SR modules in basalL patients (Fig 6A and 6C). High expression of genes in 8q is associated with high expression of the green SR module. This module is enriched for genes involved in inflammation. For the remaining three SR modules associated with the 8q-enriched tumor module, almost all genes in these modules are underexpressed when 8q genes are highly expressed (ie. the patient orderings are reversed compared to the darkgreen tumor module). These SR modules contain genes involved in general cellular processes of blood cells (RNA/protein processing, cell proliferation; darkgreen module), genes involved in cytolysis and lymphoid cell-mediated immunity (greenyellow module), and MYC and CD5 target genes (darkgrey module) (Fig 6A–6C, S5 Table). The increase in inflammatory SR and the decrease in the three other molecular processes in the SR of basalL patients whose tumor is amplified on 8q are all significantly different from how these processes are expressed in healthy controls (boxplots in Fig 6C). Overall, we identified one distinct signature in the SR of basalL patients with low immune activity at their tumor site and several immuno-suppressive signals in the SR of basalL patients whose tumor is amplified on 8q.

Discussion

Molecular profiles of peripheral blood cells and matched tumors were generated and compared for a large cohort of BC patients part of the NOWAC study. The NOWAC consortium provides a highly curated population-based study with extensive gene expression profiling across several tissues from BC patients and controls [35, 49]. A careful design and our extensive experience in blood-based expression profiles enable a detailed molecular description of the patient SR to the presence of BC where blood molecular profiles represent effectively an "averaging" over the transcriptional programs of the different types of cells in blood.

We first asked if the SR could provide accurate univariate markers of tumoral properties such as ER status or subtype. Although thousands of transcripts are differentially expressed in tumors between ER+ and ER- BC [9, 50], there is no gene in SR that can reliably predict ER status of the primary tumor. Moreover, the SR does not inform on the intrinsic BC subtype of the tumor such as lumA, lumB or basalL subtype or on IntClust subtypes. Interestingly, univariate markers in the patient SR were only identified for the CIT lumC subtype defined as particularly immunogenic ER+ tumors [8], suggesting that the SR is informative in cases where the primary tumor exhibits strong immune properties. This is consistent with previous reports that uses blood transcriptomics as a gateway into the patient immune system [51–53] and which is extensively used in the context of autoimmune and infectious diseases [54–56]. This result suggests that it is also applicable in cancer such as BC.

To further investigate the molecular changes in the patient SR, we extended our analyses to multivariate approaches where genes are combined into sets or "modules". In particular, we

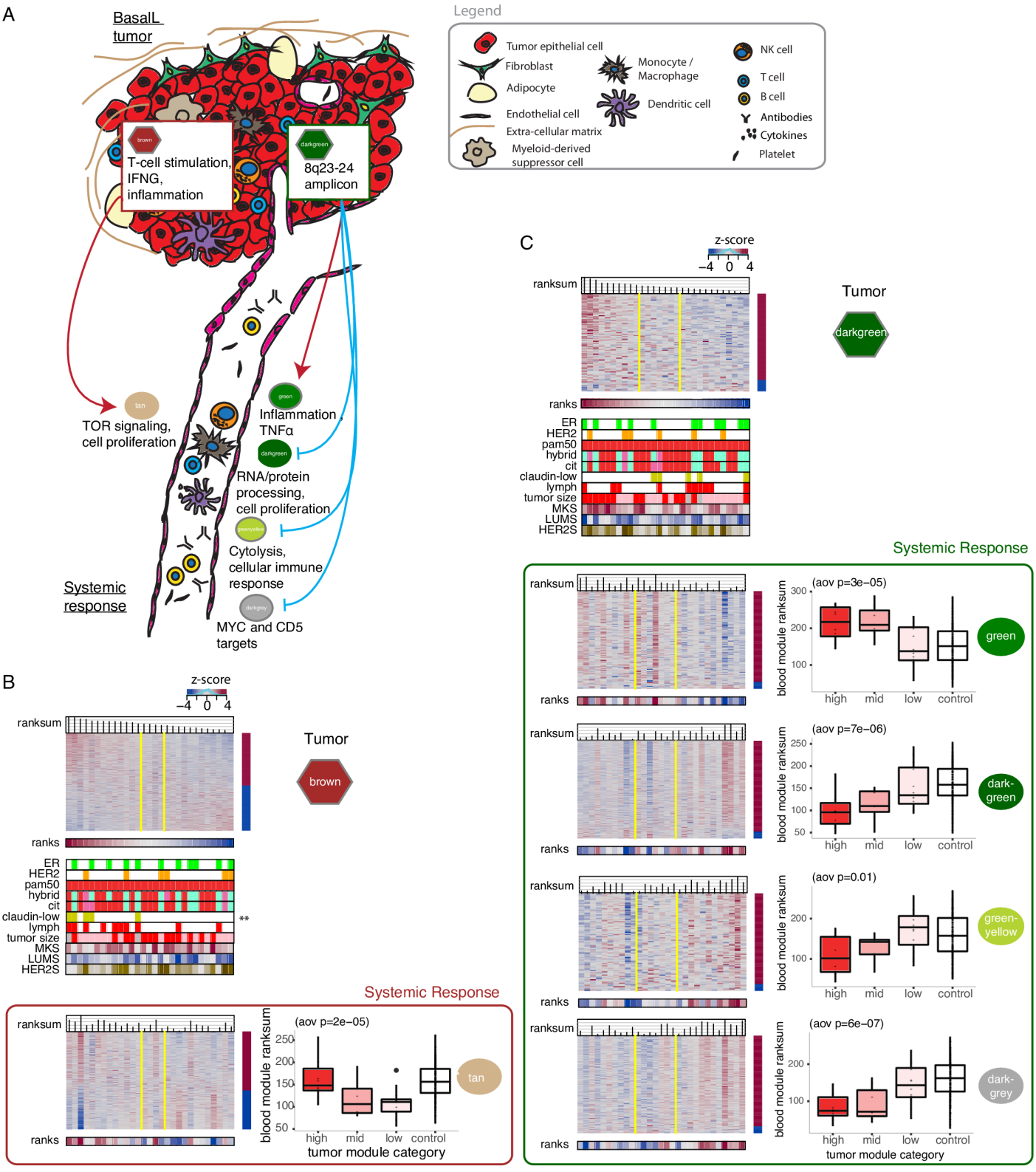


Fig 6. Significant Matched Interactions across Tissue (MIxT) in basalL patients. (A) The figure summarizes the two sets of significant MIxT in basalL patients detailed in Fig 6A and 6C. Top pathway enrichment keywords are presented for each module. Red and blue arrows correspond to negative and

positive correlations between ranksums, respectively. (B) MIxT in basalL patients between the brown tumor module and the darkgreen SR module. Heatmaps are ordered by ranksum of gene expression in the brown tumor module. Asterisks represent the level of significance of the associations between the gene ranksums for the brown tumor module and clinicopathological attributes of patients (fdr < **0.05). Associations were estimated using ANOVA and Pearson correlation for categorical and continuous variable, respectively. Boxplots show the distribution of ranksums for the SR module in patients classified according to their ROI_{95} tumor module category and controls. (C) The second set of MIxT in basalL patients between the darkgreen tumor module and four SR modules (darkgreen, green, greenyellow, darkgrey). Legend follows Fig 6B.

https://doi.org/10.1371/journal.pcbi.1005680.g006

performed cluster analysis to partition the genes of both tumor and SR profiles into modules with each module representing a distinct pattern of expression across patients. Our user-friendly website (www.mixt-blood-tumor.bci.mcgill.ca) provides access to these modules built in each tissue, enables investigation of their expression profiles in each tissue and allow user-defined queries of gene, gene sets, and pathway of interest. Further, our MIxT approach estimates gene module expression in both tissues and find significant associations between modules across tissues in a representative cohort of BC patients.

In our dataset, the primary tumor and SR have approximately the same number of modules (19 and 23, respectively) but their gene composition is qualitatively different. Not surprisingly, many modules in tumors were enriched for genes involved in hallmarks of cancer, while SR modules were enriched for either general cellular processes or specific immune responses. Only one module involved in the IFN-I pathway is highly conserved in both tumor and SR, although the common genes had markedly different expression patterns between the two tissues. This is important as it establishes that genes, whose expression patterns may act as good markers in the primary tumor, are not necessarily expressed in the same manner within blood cells.

Our multivariate approach was able to identify modules from the patient SR that could reliably identify not only lumC but also HER2+ and large (> 2cm) tumors. These three cases are among the most immunogenic subtypes of BC and are of relatively poor prognosis. For these patients, gene expression in blood cells is mostly decreased compared to other BC and controls. This result also highlights the importance of distinct immune components of the SR for each of these disease groups: B-cells for HER2+ tumors, T-cells for lumC, and aspects of the cellular immune response for large tumors. Interestingly, a previous study showed that her2E tumors have the highest B-cell infiltration and expression of B-cell receptor gene segments, although this was not predictive of improved patient survival [57]. Our study finds an impaired systemic B-cell response specifically in HER2+ patients, consistent with an inefficient anti-tumoral response in these patients, potentially due to a dysfunctional antigen receptor response and cell development. We could also speculate that the dysfunctional thymic T-cell homing signature in lumC patients reflects the well-documented effect of estrogen on thymic T lymphopoiesis [58–61] in patients diagnosed with a highly immunogenic ER+ tumor. These associations would certainly require validation in follow-up studies.

Finally, MIxT focuses on molecular associations between tissues and provides a holistic view of molecular changes in BC patients. Although the focus here is towards gene expression of blood and matched tumor, our approach could be extended to multiple tissues (eg. blood-microenvironment-tumor) or other levels of molecular data (eg. DNA level somatic aberrations, gene and miRNA expression, epigenetic profiles).

Interestingly, associations between BC tumor and patient SR are heavily dependent on subtype. Only one interaction between tumor and patient SR is identified when all BC patients are considered in the analysis but many are identified when we first stratify patients by BC subtype. This is perhaps not surprising given that there is a great deal of molecular heterogeneity between BC subtypes making "one SR fitting all" highly unlikely. We identified molecular stimuli in tumors that change patient SR in multiple ways only for patients within a particular

subtype. For example, expression of genes involved in alternative splicing in ER+/HER2- tumors is associated with changes in expression of multiple processes in SR of patients and those associations are observed only within this specific subtype.

Of note, immune signals measured at the tumor site are associated with distinct SR across a broad range of subtypes. Immune-related processes are known to be more or less expressed within every subtypes and have prognostic capacity in almost all subtypes [9]. Here we show that a change in immune activity at the tumor site is not associated with equal SR across subtypes. Furthermore, high immune signals in tumor is associated with the patient inflammatory SR in opposite ways depending if the patient is ER-/HER2- or lumB. The high inflammatory SR in ER-/HER2- patients (with low immune activity at the tumor site) and in lumB patients (with high immune activity at the tumor site) were both significantly different from how systemic inflammation is "normally" expressed in controls.

Finally, we identify other examples of interactions between tumor and patient SR that occur in subtype-specific fashions. In particular, three tumor modules were enriched for genes in known large-scale BC amplicons (16p11-13, 8q23-24). The expression of these genes changes in a coordinated manner from high to low, suggesting that these genes measure amplification of the corresponding region in BC tumors. In turn, these patterns of expression were associated with distinct SR depending on subtypes highlighting the significance of each amplicon in defining patient SR for particular BC subtypes (eg 16p13 in lumA and ER+/HER2+, and 8q23-24 in basalL and her2E). Of note, these patterns of expression also define patients with particular clinico-pathological characteristics. For example, ER+/HER2+ tumors that do not highly express the genes on 16p have a lower luminal score than ER+/HER2+ tumors that highly express the genes on 16p.

When we restrict our attention to basalL patients, we observe that both the immune-related module and the presence of a 8q23-24 amplification is associated with the patient SR. In fact, the subset of basal patients with 8q23-24 amplification exhibit high inflammatory SR and underexpress genes involved in general cellular proliferation of blood cells, in immune cytolysis, and in MYC and CD5 targets. Together, our matched profiles offer a detailed map of tumor-permissive SR particularly relevant for basalL tumors amplified on 8q and highlight a signature in the SR of basalL patients with low immune activity at their tumor site. This is especially interesting in the context of BC-immunotherapy combination or for monitoring response to these therapies. Overall, our study set the groundwork for further investigation of promising new ways to tackle and monitor the disease by looking outside the tumor and exploiting the patient SR.

Materials and methods

Gene expression data

Tumor and blood samples were obtained as part of the NOWAC study [49, 62] with approval from Regional Committees for Medical and Health Research Ethics in Norway. Between 2006–10, we collected blood and biopsy samples from BC cases at time of diagnosis, and blood samples from selected age-matched blood controls together with associated lifestyle and clinicopathologic data (S1 Text). In total, and after data preprocessing, profiles include 16,792 unique genes expressed in primary tumors and blood from 173 BC patients, and in blood from 290 controls (S1A Fig).

Subtypes and gene markers of subtypes

We used ER status as measured by IHC and HER2 status measured by FISH or IHC where available. When unavailable, ER and HER2 status was determined using gene expression of the

ESR1 gene and 6 gene members of the HER2 amplicon, respectively [9, 63] (S1 Text, S1B and S1C Fig). In addition, we calculated the HER2 score (HER2S) and the luminal score (LUMS) as the average expression of the HER2 amplicon gene members and the pam50 luminal genes, respectively. A proliferation score was calculated similarly using 12 mitotic kinases to produce the Mitotic kinase gene expression score (MKS) [45]. Samples were labeled according to our subtyping schemes from the literature: PAM50 [5], hybrid [9], CIT [8], IntClust [7, 39] (S1 Text).

Lists of differentially expressed genes in SR according to subtypes were obtained using the R/Bioconductor package Limma [64]. Whenever p-values were adjusted for multiple testing, the false discovery rate [65] was controlled at the reported level (S1 Text).

Weighted gene co-expression analysis (WGCNA) and gene modules

An unsigned weighted co-expression network was constructed independently in each tissue (SR and tumor) using the R/Bioconductor package WGCNA [41] (S1 Text). First, a matrix of pairwise correlations between all pairs of genes is constructed across blood and tumor samples, respectively. Next, the adjacency matrix is obtained by raising the co-expression measure to the power $\beta = 6$ (default value). Based on the resulting adjacency matrix, we calculate the topological overlap, which is a robust and biologically meaningful measure of network interconnectedness [42] (that is, the strength of two genes' co-expression relationship with respect to all other genes in the network). Genes with highly similar co-expression relationships are grouped together by performing average linkage hierarchical clustering on the topological overlap. The Dynamic Hybrid Tree Cut algorithm [43] cuts the hierarchal clustering tree, and modules are defined as branches from the tree cutting. Modules in each network were annotated based on Gene Ontology biological processes (weight01 Fisher test [44]), MSigDB [66] and other curated signatures relevant to immune and blood cell responses [33, 46, 52] (S1 Text)

Gene ranksum and linear ordering of patients

Our approach maps samples to a linear ordering based on expression of genes within a given module or signature of interest (S1 Text). In an univariate fashion, each gene within a given module/signature is used to rank all patients based on their expression. For each patient, the ranks of all k genes from the signature are summed and patients are then linearly ordered from right to left according to this ranksum vector. To identify the left and right boundaries of the low and high regions within the observed linear ordering, we delimit the region of independance (ROI_{95}) for each module. Briefly, we compute (n = 10K times) the position of an artificial patient within the observed linear ordering by summing the randomized ranks over all k genes in the module (S1 Text). The ROI_{95} is defined as the region that contains 95% of the randomly generated samples. The three defined categories of patients correspond to those patients that have high ranskums of the module/signature (high category), low ranksums of the module/signature (low category), and a set of patients where the expression of the genes within the module/signature lose their pattern of pairwise correlation (mid category).

Module association tests

Using gene ranksums to capture module expression, we asked how modules are associated with patients' clinical attributes and how they are associated across tissues. Pearson correlation and Analysis of Variance (ANOVA) was used to test association between a given module and continuous patient attributes (eg. age, weight, MKS, LUMS) and between a given module and categorical patient attributes (eg. ER, HER2, subtypes, lymph node status), respectively (S1

Text). For each variable. we computed empirical p-values after permuting clinical labels 1000 times. For each variable, we perform a total of 42 association tests (23 blood modules + 19 tumor modules) and used false discovery rate [65] to correct for multiple testing for each variable independently or for each "family" of tests when dependent variables are very similar (S1 Text).

Interactions between modules across tissues are identified using a random permutation approach based on the Pearson correlation between ranksums of gene expression in modules across tissues (S1 Text). ANOVA was used to compare SR module expression between BC patients (assigned to a given tumor module ROI_{95} categories) and controls.

Data and software availability

Data resource. Microarray data have been deposited at the European Genome-phenome Archive [67] (EGA; https://www.ebi.ac.uk/ega/; accession number EGAS00001001804).

Software. The MIxT web application (http://mixt-blood-tumor.bci.mcgill.ca/) is written in the Go programming language to provide an interface to statistical analyses in R and link to online databases. Users can browse through all the results generated for this study, visualize gene co-expression networks and expression heatmaps, and search for genes, gene lists, and pathways. We use Bootstrap (http://getbootstrap.com) to build the user interface and Javascript libraries D3 (http://d3js.org) and Sigma (http://sigmajs.org) to build interactive visualizations. The web application framework is open sourced at http://github.com/fjukstad/mixt.

Supporting information

S1 Table. Enrichment of clinicopathological and tumor subtypes attributes across subtyping schemes. The table shows statistically significant associations between tumor attributes (columns) and subtypes (rows). For columns representing binary variables (ER, HER2, LN, as well as subtype/cohorts), the table shows the number of patients and the level of significance computed using Fisher's exact test (FET). Enrichment is indicated using "+" symbols, while for depletion "-" symbols are used. The number of symbols in each entry correspond to significance levels of 0.05, 0.01, 0.001, and < 0.0001. For example, the entry in row "her2E" and column "HER2+" contains the symbol "++++" indicating that herE patients are more likely to be HER2+ than non-her2E patients. In contrast, the entry in row "her2E" and column "ER+" contains the symbol "—" indicating that her2E patients are less likely to be ER+ than non-her2E patients. Grey indicates cases where enrichment cannot be calculated.
(XLSX)

S2 Table. Top GO terms enriched in tumor modules. Top 5 GO terms that overlap with each module. "Annotated" indicates the number of genes in the GO term, "Significant" indicates the number of overlapping genes. "Expected" indicates the number of genes that we would expect by chance to be overlapping with the GO term. "classicFisher" presents the p-value from a classic fisher exact test and "weight01Fisher" presents the p-value from the weight01 algorithm and fisher exact test from [44].
(XLSX)

S3 Table. Top 5 enrichments among each of the following signature sets. i) c1, c2.cgp, c2.cp, c6, c7 and h gene set collections from MSigDB signatures (v5.1) [66]. ii) peripheral-blood mononuclear cell (PBMC) transcriptional modules (sig.set = i) from [52]. iii) our blood-based gene expression signatures (341- and 50-gene; sig.set = d) for BC [33] iv) immune-specific gene sets (sig.set = iris) from [46]. Enrichment for each gene signature was estimated for all genes in the modules and for genes that are positively (red genes up) or negatively (blue genes

dn) correlated with the patient ranksum only using the hypergeometric minimum-likelihood p-values, computed with the function 'dhyper' (equivalent to one-sided Fisher exact test). P-values were then adjusted for multiple testing using false discovery rate [65].
(XLSX)

S4 Table. Top GO terms enriched in SR modules. Legend follows S2 Table.
(XLSX)

S5 Table. Top 5 gene sets of each signature set enriched in SR modules. Legend follows S3 Table.
(XLSX)

S1 Fig. Gene expression and clinical data processing. (A) Preprocessing of the microarray data was performed identically in each of the five datasets: blood (bl) 1–4 and tumor (t. 1) datasets. Steps that trim samples and probes/genes are presented horizontally and vertically, respectively. In total, we investigated blood and tumor profiles from 173 BC patients and blood profiles from 282 controls. Profiles include 16,782 unique genes. (B) Imputation of missing ER status based on expression of *ESR1* gene. Receiver operating characteristic (ROC) curve setting on the right using IHC/FISH assignment as true label. False positive rate threshold was set to < 0.2 with regard to the true label. (C) Imputation of missing HER2 status based on expression of genes included in the HER2 amplicon (*ERBB2*, *GRB7*, *PGAP3*, *PNMT*, *MIEN1*, *TCAP*).
(TIF)

S2 Fig. Significant univariate gene markers of subtypes in SR (false discovery rate, fdr $\leq$ 0.2). Blue and red shade correspond to under- and over- expression of the marker in a given subtype vs the others, respectively. Shading is proportional to the level of significance of the gene marker.
(TIF)

S3 Fig. Gene expression heatmap of the 70 blood markers of lumC tumors. Rows correspond to genes and columns correspond to samples. Gene expression are scaled by row. Patients are linearly ordered based on their ranksum of gene expression. Genes are ordered by their correlation to the observed patient ordering. Genes that are positively and negatively correlated with the patient ranksum are represented in the right sidebar colored in red and blue, respectively. Yellow vertical lines delimit the Region Of Independence (ROI_{95}) that contains 95% of the randomly generated samples. A green tick in 'lumC' refers to a patient with a luminal C tumor according to the CIT scheme [8].
(TIF)

S4 Fig. Gene co-expression networks in each tissue. (A) Heatmap of the topological overlap between genes expressed in tumors. Each row and column represent a gene, light color indicates low topological overlap and progressively darker red indicates higher topological overlap. Module assignment is displayed along the left and the top of the heatmap. (B) Heatmap of the topological overlap between genes expressed in SR. The legend follows S4A Fig.
(TIF)

S5 Fig. Expression patterns of the green tumor module. (A) Expression heatmap of genes in the green tumor module. Legend follows S3 Fig. Color coding for ER, HER2, pam50, hybrid, cit, claudin-low and lymph follows Fig 1D. In general, a tick for a binary clinical variable refers to a positive value (eg. a red tick in 'basalL' refers to patients with basalL tumors). For continuous variables such as Mitosis Kinase Score (MKS), Luminal Score (LUMS), HER2 score

(HER2S), age, and weight, dark and light shades represent high and low values, respectively. Asterisks represent the level of significance of the associations between the gene ranksums for the green tumor module and clinicopathological attributes of patients. Associations were estimated using ANOVA or Pearson correlation for categorical and continuous variable, respectively (p-value $< {}^{*}0.05, {}^{**}0.01, {}^{***}0.001$). (B) Distribution of ranksums for the green tumor module according to pam50 subtypes. aov: analysis of variance.
(TIF)

S6 Fig. SR modules associated with clinico-pathological variable. (A) One (saddlebrown) modules in the patient SR is associated with HER2+ BC. (B) Three modules in the patient SR are associated with lumC BC. (C) One module in the patient SR are associated to both lumC and large (>2cm) tumors. (D) Three modules in the patient SR are associated with large (>2cm) tumors. The legend for expression heatmaps (left) follows S3 Fig. Boxplots (right) compare module expression in SR from patients with lumC, HER2+, or large tumors with other BC patients, and controls. aov: analysis of variance.
(TIF)

S7 Fig. Background distributions of the correlations coefficients between ranksums of gene expression in modules across tissues within each subtype. The dotted lines represent the lower and higher bounds that were used to call significant associations between modules across tissues. Curves are colored according to the families of subtypes as listed in Fig 4.
(TIF)

S8 Fig. Associations between the darkgrey tumor module and distinct SR by subtypes. (A) Scatter plot of ranksums of the darkgrey tumor module and the lightyellow SR module in CIT lumA patients. The top corner depicts the background distributions of the correlations coefficients between ranksums of every modules pairs across tissues in CIT lumA patients. (B) Scatter plot of ranksums of the darkgrey tumor module and the salmon SR module in ER+HER2 + patients. Legend follows S8A Fig (C) Expression heatmap of genes in the darkgrey tumor module in luminal A patients under the CIT scheme [8]. Patients are linearly ordered based on the ranksum of gene expression of the darkgrey tumor module. Yellow vertical lines delimit the ROI_{95} in tumor that contains 95% of the randomly generated samples. Genes that are positively and negatively correlated with the patient ranksum are represented in the right sidebar colored in red and blue, respectively. Top enrichment keywords are indicated on the left of the heatmap (S2 and S3 Tables). (D) Expression heatmap of genes in the darkgrey tumor module in ER+/HER2+ patients. Legend follows S8C Fig. (E) Expression heatmap of genes in the lightyellow SR module luminal A patients under the CIT scheme [8]. Legend follows S8C Fig. Top enrichment keywords are indicated on the left of the heatmap (S4 and S5 Tables). (F) Expression heatmap of genes in the salmon SR module in ER+/HER2+ patients. Legend follows S8C Fig. Top enrichment keywords are indicated on the left of the heatmap (S4 and S5 Tables). (G) Clinical characteristics of luminal A patients under the CIT scheme [8] ordered by gene ranksums derived from the darkgrey tumor module. Legend follows Fig 1D. Asterisks represent the level of significance of the associations between the gene ranksums for the darkgrey tumor module and clinicopathological attributes of patients. Associations were estimated using ANOVA (fdr $< {}^{*}0.1, {}^{**}0.05, {}^{***}0.01$). (H) Clinical characteristics of ER+/HER2+ patients ordered by gene ranksums derived from the darkgrey tumor module. Legend follows S8G Fig. (I) Distribution of ranksums for luminal A patients under the CIT scheme [8] and controls induced by the expression of genes in the lightyellow SR module. Patients are grouped according to the ROI_{95} darkgrey tumor module category as defined in S8C Fig. aov: analysis of variance (H) Distribution of ranksums for ER+HER2+ patients and controls induced by the

expression of genes in the salmon SR module. Patients are grouped according to the ROI_{95} darkgrey tumor module category as defined in S8D Fig.
(TIF)

S1 Text. Supporting methods.
(PDF)

Acknowledgments

The authors acknowledge the clinical/pathological assistance provided by S. Dahl, T. Sauer, T. Cappelen, B. Naume and R. Mortensen members of the Norwegian Breast Cancer Group; the technical assistance from M. Melhus and B. Augdal; and preliminary work on MIxT from A. Tofigh.

Author Contributions

Conceptualization: Vanessa Dumeaux, Eiliv Lund.

Formal analysis: Vanessa Dumeaux.

Funding acquisition: Vanessa Dumeaux, Eiliv Lund, Michael Hallett.

Methodology: Vanessa Dumeaux, Michael Hallett.

Resources: Hans E. Fjosne, Jan-Ole Frantzen, Marit Muri Holmen, Enno Rodegerdts, Ellen Schlichting, Anne-Lise Børresen-Dale.

Software: Vanessa Dumeaux, Bjørn Fjukstad, Lars Ailo Bongo.

Writing – original draft: Vanessa Dumeaux, Michael Hallett.

Writing – review & editing: Vanessa Dumeaux, Bjørn Fjukstad, Lars Ailo Bongo, Eiliv Lund, Michael Hallett.

References

1. Yarden Y. The biological framework: translational research from bench to clinic. Oncologist. 2010; 15 Suppl 5:1–7. https://doi.org/10.1634/theoncologist.2010-S5-01 PMID: 21138950.
2. Weigelt B, Baehner FL, Reis-Filho JS. The contribution of gene expression profiling to breast cancer classification, prognostication and prediction: a retrospective of the last decade. J Pathol. 2010; 220 (2):263–80. Epub 2009/11/21. https://doi.org/10.1002/path.2648 PMID: 19927298.
3. Perou CM, Sorlie T, Eisen MB, van de Rijn M, Jeffrey SS, Rees CA, et al. Molecular portraits of human breast tumours. Nature. 2000; 406(6797):747–52. Epub 2000/08/30. https://doi.org/10.1038/35021093 PMID: 10963602.
4. Sorlie T, Perou CM, Tibshirani R, Aas T, Geisler S, Johnsen H, et al. Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications. Proceedings of the National Academy of Sciences of the United States of America. 2001; 98(19):10869–74. Epub 2001/09/13. https://doi.org/10.1073/pnas.191367098 PMID: 11553815; PubMed Central PMCID: PMC58566.
5. Parker JS, Mullins M, Cheang MC, Leung S, Voduc D, Vickery T, et al. Supervised risk predictor of breast cancer based on intrinsic subtypes. Journal of clinical oncology: official journal of the American Society of Clinical Oncology. 2009; 27(8):1160–7. Epub 2009/02/11. https://doi.org/10.1200/JCO.2008.18.1370 PMID: 19204204; PubMed Central PMCID: PMC2667820.
6. Curtis C, Shah SP, Chin SF, Turashvili G, Rueda OM, Dunning MJ, et al. The genomic and transcriptomic architecture of 2,000 breast tumours reveals novel subgroups. Nature. 2012; 486(7403):346–52. Epub 2012/04/24. https://doi.org/10.1038/nature10983 PMID: 22522925; PubMed Central PMCID: PMC3440846.
7. Ali HR, Rueda OM, Chin SF, Curtis C, Dunning MJ, Aparicio SA, et al. Genome-driven integrated classification of breast cancer validated in over 7,500 samples. Genome biology. 2014; 15(8):431. https://doi.org/10.1186/s13059-014-0431-1 PMID: 25164602; PubMed Central PMCID: PMCPMC4166472.

8. Guedj M, Marisa L, de Reynies A, Orsetti B, Schiappa R, Bibeau F, et al. A refined molecular taxonomy of breast cancer. Oncogene. 2012; 31(9):1196–206. Epub 2011/07/26. https://doi.org/10.1038/onc.2011.301 PMID: 21785460; PubMed Central PMCID: PMC3307061.
9. Tofigh A, Suderman M, Paquet ER, Livingstone J, Bertos N, Saleh SM, et al. The prognostic ease and difficulty of invasive breast carcinoma. Cell Rep. 2014; 9(1):129–42. https://doi.org/10.1016/j.celrep.2014.08.073 PMID: 25284793.
10. Hornberger J, Alvarado MD, Rebecca C, Gutierrez HR, Yu TM, Gradishar WJ. Clinical validity/utility, change in practice patterns, and economic implications of risk stratifiers to predict outcomes for early-stage breast cancer: a systematic review. Journal of the National Cancer Institute. 2012; 104(14):1068–79. Epub 2012/07/07. https://doi.org/10.1093/jnci/djs261 PMID: 22767204.
11. Killock D. Breast cancer: Genetic signature might spare 100,000 women annually from chemotherapy. Nature reviews Clinical oncology. 2016; 13(10):589. https://doi.org/10.1038/nrclinonc.2016.150 PMID: 27620708.
12. Paik S, Shak S, Tang G, Kim C, Baker J, Cronin M, et al. A multigene assay to predict recurrence of tamoxifen-treated, node-negative breast cancer. The New England journal of medicine. 2004; 351(27):2817–26. Epub 2004/12/14. https://doi.org/10.1056/NEJMoa041588 PMID: 15591335.
13. van 't Veer LJ, Dai H, van de Vijver MJ, He YD, Hart AA, Mao M, et al. Gene expression profiling predicts clinical outcome of breast cancer. Nature. 2002; 415(6871):530–6. Epub 2002/02/02. https://doi.org/10.1038/415530a PMID: 11823860.
14. Bissell MJ, Radisky D. Putting tumours in context. Nat Rev Cancer. 2001; 1(1):46–54. Epub 2002/03/20. https://doi.org/10.1038/35094059 PMID: 11900251; PubMed Central PMCID: PMC2975572.
15. Cichon MA, Degnim AC, Visscher DW, Radisky DC. Microenvironmental influences that drive progression from benign breast disease to invasive breast cancer. J Mammary Gland Biol Neoplasia. 2010; 15(4):389–97. Epub 2010/12/17. https://doi.org/10.1007/s10911-010-9195-8 PMID: 21161341; PubMed Central PMCID: PMC3011086.
16. Hanahan D, Coussens LM. Accessories to the crime: functions of cells recruited to the tumor microenvironment. Cancer Cell. 2012; 21(3):309–22. Epub 2012/03/24. https://doi.org/10.1016/j.ccr.2012.02.022 PMID: 22439926.
17. McMillin DW, Negri JM, Mitsiades CS. The role of tumour-stromal interactions in modifying drug response: challenges and opportunities. Nature reviews Drug discovery. 2013; 12(3):217–28. Epub 2013/03/02. https://doi.org/10.1038/nrd3870 PMID: 23449307.
18. Pylayeva-Gupta Y, Grabocka E, Bar-Sagi D. RAS oncogenes: weaving a tumorigenic web. Nat Rev Cancer. 2011; 11(11):761–74. Epub 2011/10/14. https://doi.org/10.1038/nrc3106 PMID: 21993244; PubMed Central PMCID: PMC3632399.
19. Santarpia L, Lippman SM, El-Naggar AK. Targeting the MAPK-RAS-RAF signaling pathway in cancer therapy. Expert opinion on therapeutic targets. 2012; 16(1):103–19. Epub 2012/01/14. https://doi.org/10.1517/14728222.2011.645805 PMID: 22239440; PubMed Central PMCID: PMC3457779.
20. Scherz-Shouval R, Santagata S, Mendillo ML, Sholl LM, Ben-Aharon I, Beck AH, et al. The reprogramming of tumor stroma by HSF1 is a potent enabler of malignancy. Cell. 2014; 158(3):564–78. https://doi.org/10.1016/j.cell.2014.05.045 PMID: 25083868; PubMed Central PMCID: PMCPMC4249939.
21. Wendt MK, Smith JA, Schiemann WP. Transforming growth factor-beta-induced epithelial-mesenchymal transition facilitates epidermal growth factor-dependent breast cancer progression. Oncogene. 2010; 29(49):6485–98. Epub 2010/08/31. https://doi.org/10.1038/onc.2010.377 PMID: 20802523; PubMed Central PMCID: PMC3076082.
22. Finak G, Bertos N, Pepin F, Sadekova S, Souleimanova M, Zhao H, et al. Stromal gene expression predicts clinical outcome in breast cancer. Nature medicine. 2008; 14(5):518–27. Epub 2008/04/29. https://doi.org/10.1038/nm1764 PMID: 18438415.
23. Hu M, Polyak K. Molecular characterisation of the tumour microenvironment in breast cancer. European journal of cancer. 2008; 44(18):2760–5. Epub 2008/11/26. https://doi.org/10.1016/j.ejca.2008.09.038 PMID: 19026532; PubMed Central PMCID: PMC2729518.
24. Pepin F, Bertos N, Laferriere J, Sadekova S, Souleimanova M, Zhao H, et al. Gene expression profiling of microdissected breast cancer microvasculature identifies distinct tumor vascular subtypes. Breast cancer research: BCR. 2012; 14(4):R120. Epub 2012/08/22. https://doi.org/10.1186/bcr3246 PMID: 22906178.
25. Gnerlich J, Jeffe DB, Deshpande AD, Beers C, Zander C, Margenthaler JA. Surgical removal of the primary tumor increases overall survival in patients with metastatic breast cancer: analysis of the 1988–2003 SEER data. Annals of surgical oncology. 2007; 14(8):2187–94. Epub 2007/05/25. https://doi.org/10.1245/s10434-007-9438-0 PMID: 17522944.

26. Egeblad M, Nakasone ES, Werb Z. Tumors as organs: complex tissues that interface with the entire organism. Developmental cell. 2010; 18(6):884–901. Epub 2010/07/16. https://doi.org/10.1016/j.devcel.2010.05.012 PMID: 20627072; PubMed Central PMCID: PMC2905377.

27. McAllister SS, Weinberg RA. Tumor-host interactions: a far-reaching relationship. Journal of clinical oncology: official journal of the American Society of Clinical Oncology. 2010; 28(26):4022–8. Epub 2010/07/21. https://doi.org/10.1200/JCO.2010.28.4257 PMID: 20644094.

28. Castano Z, Tracy K, McAllister SS. The tumor macroenvironment and systemic regulation of breast cancer progression. The International journal of developmental biology. 2011; 55(7–9):889–97. Epub 2011/12/14. https://doi.org/10.1387/ijdb.113366zc PMID: 22161844.

29. DeFilippis RA, Tlsty TD. Hello out there. . .is anybody listening? Cancer discovery. 2012; 2(12):1084–6. Epub 2012/12/12. https://doi.org/10.1158/2159-8290.CD-12-0434 PMID: 23230186.

30. Elkabets M, Gifford AM, Scheel C, Nilsson B, Reinhardt F, Bray MA, et al. Human tumors instigate granulin-expressing hematopoietic cells that promote malignancy by activating stromal fibroblasts in mice. The Journal of clinical investigation. 2011; 121(2):784–99. Epub 2011/01/27. https://doi.org/10.1172/JCI43757 PMID: 21266779; PubMed Central PMCID: PMC3026724.

31. Kuznetsov HS, Marsh T, Markens BA, Castano Z, Greene-Colozzi A, Hay SA, et al. Identification of luminal breast cancers that establish a tumor-supportive macroenvironment defined by proangiogenic platelets and bone marrow-derived cells. Cancer discovery. 2012; 2(12):1150–65. Epub 2012/08/17. https://doi.org/10.1158/2159-8290.CD-12-0216 PMID: 22896036; PubMed Central PMCID: PMC3517696.

32. McAllister SS, Gifford AM, Greiner AL, Kelleher SP, Saelzler MP, Ince TA, et al. Systemic endocrine instigation of indolent tumor growth requires osteopontin. Cell. 2008; 133(6):994–1005. Epub 2008/06/17. https://doi.org/10.1016/j.cell.2008.04.045 PMID: 18555776.

33. Dumeaux V, Ursini-Siegel J, Flatberg A, Fjosne HE, Frantzen JO, Holmen MM, et al. Peripheral blood cells inform on the presence of breast cancer: A population-based case-control study. Int J Cancer. 2014. Epub 2014/06/17. https://doi.org/10.1002/ijc.29030 PMID: 24931809.

34. Spitzer MH, Carmi Y, Reticker-Flynn NE, Kwek SS, Madhireddy D, Martins MM, et al. Systemic Immunity Is Required for Effective Cancer Immunotherapy. Cell. 2017; 168(3):487–502 e15. https://doi.org/10.1016/j.cell.2016.12.022 PMID: 28111070.

35. Dumeaux V, Olsen KS, Nuel G, Paulssen RH, Borresen-Dale AL, Lund E. Deciphering normal blood gene expression variation—The NOWAC postgenome study. PLoS Genet. 2010; 6(3):e1000873. Epub 2010/03/20. https://doi.org/10.1371/journal.pgen.1000873 PMID: 20300640; PubMed Central PMCID: PMC2837385.

36. Debey S, Schoenbeck U, Hellmich M, Gathof BS, Pillai R, Zander T, et al. Comparison of different isolation techniques prior gene expression profiling of blood derived cells: impact on physiological responses, on overall expression and the role of different cell types. The pharmacogenomics journal. 2004; 4(3):193–207. Epub 2004/03/24. https://doi.org/10.1038/sj.tpj.6500240 PMID: 15037859.

37. Radich JP, Mao M, Stepaniants S, Biery M, Castle J, Ward T, et al. Individual-specific variation of gene expression in peripheral blood leukocytes. Genomics. 2004; 83(6):980–8. Epub 2004/06/05. https://doi.org/10.1016/j.ygeno.2003.12.013 PMID: 15177552.

38. Whitney AR, Diehn M, Popper SJ, Alizadeh AA, Boldrick JC, Relman DA, et al. Individuality and variation in gene expression patterns in human blood. Proceedings of the National Academy of Sciences of the United States of America. 2003; 100(4):1896–901. Epub 2003/02/13. https://doi.org/10.1073/pnas.252784499 PMID: 12578971; PubMed Central PMCID: PMC149930.

39. Ali HR, Chlon L, Pharoah PD, Markowetz F, Caldas C. Patterns of Immune Infiltration in Breast Cancer and Their Clinical Implications: A Gene-Expression-Based Retrospective Study. PLoS Med. 2016; 13(12):e1002194. https://doi.org/10.1371/journal.pmed.1002194 PMID: 27959923; PubMed Central PMCID: PMCPMC5154505.

40. Witten DM, Tibshirani R. A framework for feature selection in clustering. J Am Stat Assoc. 2010; 105(490):713–26. https://doi.org/10.1198/jasa.2010.tm09415 PMID: 20811510; PubMed Central PMCID: PMCPMC2930825.

41. Langfelder P, Horvath S. WGCNA: an R package for weighted correlation network analysis. BMC bioinformatics. 2008; 9:559. Epub 2008/12/31. https://doi.org/10.1186/1471-2105-9-559 PMID: 19114008; PubMed Central PMCID: PMC2631488.

42. Yip AM, Horvath S. Gene network interconnectedness and the generalized topological overlap measure. BMC bioinformatics. 2007; 8:22. https://doi.org/10.1186/1471-2105-8-22 PMID: 17250769; PubMed Central PMCID: PMCPMC1797055.

43. Langfelder P, Zhang B, Horvath S. Defining clusters from a hierarchical cluster tree: the Dynamic Tree Cut package for R. Bioinformatics. 2008; 24(5):719–20. https://doi.org/10.1093/bioinformatics/btm563 PMID: 18024473.

44. Alexa A, Rahnenfuhrer J, Lengauer T. Improved scoring of functional groups from gene expression data by decorrelating GO graph structure. Bioinformatics. 2006; 22(13):1600–7. Epub 2006/04/12. https://doi.org/10.1093/bioinformatics/btl140 PMID: 16606683.

45. Bianchini G, Iwamoto T, Qi Y, Coutant C, Shiang CY, Wang B, et al. Prognostic and therapeutic implications of distinct kinase expression patterns in different subtypes of breast cancer. Cancer research. 2010; 70(21):8852–62. https://doi.org/10.1158/0008-5472.CAN-10-1039 PMID: 20959472.

46. Abbas AR, Baldwin D, Ma Y, Ouyang W, Gurney A, Martin F, et al. Immune response in silico (IRIS): immune-specific genes identified from a compendium of microarray expression data. Genes Immun. 2005; 6(4):319–31. https://doi.org/10.1038/sj.gene.6364173 PMID: 15789058.

47. Shannon P, Markiel A, Ozier O, Baliga NS, Wang JT, Ramage D, et al. Cytoscape: a software environment for integrated models of biomolecular interaction networks. Genome research. 2003; 13 (11):2498–504. https://doi.org/10.1101/gr.1239303 PMID: 14597658; PubMed Central PMCID: PMCPMC403769.

48. Nikolsky Y, Sviridov E, Yao J, Dosymbekov D, Ustyansky V, Kaznacheev V, et al. Genome-wide functional synergy between amplified and mutated genes in human breast cancer. Cancer research. 2008; 68(22):9532–40. https://doi.org/10.1158/0008-5472.CAN-08-3082 PMID: 19010930.

49. Dumeaux V, Borresen-Dale AL, Frantzen JO, Kumle M, Kristensen VN, Lund E. Gene expression analyses in breast cancer epidemiology: the Norwegian Women and Cancer postgenome cohort study. Breast cancer research: BCR. 2008; 10(1):R13. Epub 2008/02/15. https://doi.org/10.1186/bcr1859 PMID: 18271962; PubMed Central PMCID: PMC2374969.

50. Gruvberger S, Ringner M, Chen Y, Panavally S, Saal LH, Borg A, et al. Estrogen receptor status in breast cancer is associated with remarkably distinct gene expression patterns. Cancer research. 2001; 61(16):5979–84. Epub 2001/08/17. PMID: 11507038.

51. Chaussabel D, Baldwin N. Democratizing systems immunology with modular transcriptional repertoire analyses. Nat Rev Immunol. 2014; 14(4):271–80. https://doi.org/10.1038/nri3642 PMID: 24662387; PubMed Central PMCID: PMCPMC4118927.

52. Chaussabel D, Pascual V, Banchereau J. Assessing the human immune system through blood transcriptomics. BMC biology. 2010; 8:84. Epub 2010/07/14. https://doi.org/10.1186/1741-7007-8-84 PMID: 20619006; PubMed Central PMCID: PMC2895587.

53. Li S, Rouphael N, Duraisingham S, Romero-Steiner S, Presnell S, Davis C, et al. Molecular signatures of antibody responses derived from a systems biology study of five human vaccines. Nat Immunol. 2014; 15(2):195–204. https://doi.org/10.1038/ni.2789 PMID: 24336226; PubMed Central PMCID: PMCPMC3946932.

54. Banchereau R, Hong S, Cantarel B, Baldwin N, Baisch J, Edens M, et al. Personalized Immunomonitoring Uncovers Molecular Networks that Stratify Lupus Patients. Cell. 2016; 165(3):551–65. https://doi.org/10.1016/j.cell.2016.03.008 PMID: 27040498.

55. Berry MP, Graham CM, McNab FW, Xu Z, Bloch SA, Oni T, et al. An interferon-inducible neutrophil-driven blood transcriptional signature in human tuberculosis. Nature. 2010; 466(7309):973–7. https://doi.org/10.1038/nature09247 PMID: 20725040; PubMed Central PMCID: PMCPMC3492754.

56. Mejias A, Dimo B, Suarez NM, Garcia C, Suarez-Arrabal MC, Jartti T, et al. Whole blood gene expression profiles to assess pathogenesis and disease severity in infants with respiratory syncytial virus infection. PLoS Med. 2013; 10(11):e1001549. https://doi.org/10.1371/journal.pmed.1001549 PMID: 24265599; PubMed Central PMCID: PMCPMC3825655.

57. Iglesia MD, Vincent BG, Parker JS, Hoadley KA, Carey LA, Perou CM, et al. Prognostic B-cell signatures using mRNA-seq in patients with subtype-specific breast and ovarian cancer. Clinical cancer research: an official journal of the American Association for Cancer Research. 2014; 20(14):3818–29. https://doi.org/10.1158/1078-0432.CCR-13-3368 PMID: 24916698; PubMed Central PMCID: PMCPMC4102637.

58. Rijhsinghani AG, Thompson K, Bhatia SK, Waldschmidt TJ. Estrogen blocks early T cell development in the thymus. Am J Reprod Immunol. 1996; 36(5):269–77. PMID: 8955504.

59. Zoller AL, Kersh GJ. Estrogen induces thymic atrophy by eliminating early thymic progenitors and inhibiting proliferation of beta-selected thymocytes. J Immunol. 2006; 176(12):7371–8. PMID: 16751381.

60. Shi Y, Wu W, Chai Q, Li Q, Hou Y, Xia H, et al. LTbetaR controls thymic portal endothelial cells for haematopoietic progenitor cell homing and T-cell regeneration. Nature communications. 2016; 7:12369. https://doi.org/10.1038/ncomms12369 PMID: 27493002; PubMed Central PMCID: PMCPMC4980457.

61. Forster R, Davalos-Misslitz AC, Rot A. CCR7 and its ligands: balancing immunity and tolerance. Nat Rev Immunol. 2008; 8(5):362–71. https://doi.org/10.1038/nri2297 PMID: 18379575.

62. Lund E, Dumeaux V, Braaten T, Hjartaker A, Engeset D, Skeie G, et al. Cohort profile: The Norwegian Women and Cancer Study—NOWAC—Kvinner og kreft. Int J Epidemiol. 2008; 37(1):36–41. Epub 2007/07/24. https://doi.org/10.1093/ije/dym137 PMID: 17644530.

63. Staaf J, Jonsson G, Ringner M, Vallon-Christersson J, Grabau D, Arason A, et al. High-resolution genomic and expression analyses of copy number alterations in HER2-amplified breast cancer. Breast cancer research: BCR. 2010; 12(3):R25. Epub 2010/05/13. https://doi.org/10.1186/bcr2568 PMID: 20459607; PubMed Central PMCID: PMC2917012.

64. Smyth GK. Linear models and empirical bayes methods for assessing differential expression in microarray experiments. Statistical applications in genetics and molecular biology. 2004; 3:Article3. Epub 2006/05/02. https://doi.org/10.2202/1544-6115.1027 PMID: 16646809.

65. Benjamini Y, Hochberg Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. Journal of the Royal Statistical Society Series B. 1995; 57:289–300.

66. Subramanian A, Tamayo P, Mootha VK, Mukherjee S, Ebert BL, Gillette MA, et al. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. Proceedings of the National Academy of Sciences of the United States of America. 2005; 102(43):15545–50. Epub 2005/10/04. https://doi.org/10.1073/pnas.0506580102 PMID: 16199517; PubMed Central PMCID: PMC1239896.

67. Lappalainen I, Almeida-King J, Kumanduri V, Senf A, Spalding JD, Ur-Rehman S, et al. The European Genome-phenome Archive of human data consented for biomedical research. Nature genetics. 2015; 47(7):692–5. https://doi.org/10.1038/ng.3312 PMID: 26111507.

Paper 4

B. Fjukstad and L. A. Bongo, "A review of scalable bioinformatics pipelines," *Data Science and Engineering*, vol. 2, no. 3, pp. 245–251, 2017

Data Sci. Eng. (2017) 2:245–251
https://doi.org/10.1007/s41019-017-0047-z

A Review of Scalable Bioinformatics Pipelines

Bjørn Fjukstad[1] · **Lars Ailo Bongo**[1]

Received: 28 May 2017 / Revised: 29 September 2017 / Accepted: 2 October 2017 / Published online: 23 October 2017

Abstract Scalability is increasingly important for bioinformatics analysis services, since these must handle larger datasets, more jobs, and more users. The pipelines used to implement analyses must therefore scale with respect to the resources on a single compute node, the number of nodes on a cluster, and also to cost-performance. Here, we survey several scalable bioinformatics pipelines and compare their design and their use of underlying frameworks and infrastructures. We also discuss current trends for bioinformatics pipeline development.

Keywords Pipeline · Bioinformatics · Scalable · Infrastructure · Analysis services

1 Introduction

Bioinformatics analyses are increasingly provided as services that end users access through a web interface that has a powerful backend that executes the analyses. The services may be generic, such as those provided by research institutes such as EMBL-EBI (http://www.ebi.ac.uk/services), commercial companies such as Illumina (https://basespace.illumina.com/home/index), and research projects such as Galaxy (https://usegalaxy.org/). However, they can also be specialized and targeted, for example, to marine metagenomics as our marine metagenomics portal (https://mmp.sfb.uit.no/).

✉ Lars Ailo Bongo
larsab@cs.uit.no

[1] Department of Computer Science, UiT The Arctic University of Norway, 9037 Tromsø, Norway

Scalability is increasingly important for these analysis services, since the cost of instruments such as next-generation sequencing machines is rapidly decreasing [1]. The reduced costs have made the machines more available which has caused an increase in dataset size, the number of datasets, and hence the number of users [2]. The backend executing the analyses must therefore scale up (vertically) with respect to the resources on a single compute node, since the resource usage of some analyses increases with dataset size. For example, short sequence read assemblers [3] may require TBs of memory for big datasets and tens of CPU cores [4]. The analysis must also scale out (horizontally) to take advantage of compute clusters and clouds. For example, the widely used BLAST [5] is computationally intensive but scales linearly with respect to the number of CPU cores. Finally, to efficiently support many users it is important that the analyses scale with respect to cost-performance [6].

The data analysis is typically implemented as a pipeline (workflow) with third-party tools that each processes input files and produces output files. The pipelines are often deep, with 10 or more tools [7]. The tools are usually implemented in a pipeline framework ranging from simple R scripts to full workbenches with large collections of tools (such as the Galaxy [8] or Apache Taverna [9]). A review of pipeline frameworks is in [10], but it does not focus on scalability. Here, we survey several scalable bioinformatics pipelines and compare their design and deployment. We describe how these scale to larger datasets or more users, how they use infrastructure systems for scalable data processing, and how they are deployed and maintained. Finally, we discuss current trends in large-scale bioinformatics analyses including containers, standardization, reproducible research, and large-scale analysis-as-a-service infrastructures.

2 Scalable Pipelines

We focus our review on scalable pipelines described in published papers. Many of the pipelines are configured and executed using a pipeline framework. It is difficult to differentiate between the scalability of a pipeline framework and the scalability of individual tools in a pipeline. If a pipeline tool does not scale efficiently, it may be necessary to replace it with a more scalable tool. However, an important factor for pipeline tool scalability is the infrastructure service used by the pipeline framework for data storage and job execution (Fig. 1). For example, a columnar storage system may improve I/O performance, but many analysis tools are implemented to read and write regular files and hence cannot directly benefit from columnar storage. We therefore structure our description of each pipeline as follows:

1. We describe the compute, storage, and memory requirements of the pipeline tools. These influence the choice of the framework and infrastructure systems.
2. We describe how the pipelines are used. A pipeline used interactively to process data submitted by end users has different requirements than a pipeline used to batch process data from a sequencing machine.
3. We describe the pipeline framework used by the pipeline, how the pipeline tools are executed, how the pipeline data are stored, and the execution environment.
4. We describe how the pipeline tools scale out or up, how the pipeline framework supports multiple users or jobs, and whether the execution environment provides elasticity to adjust the resources allocated for the service.
5. We discuss limitations and provide comparisons to other pipelines.

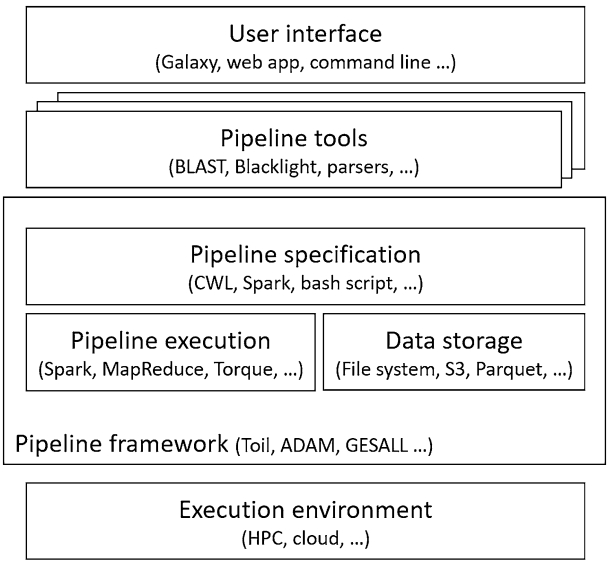


Fig. 1 Scalable pipeline components. A pipeline consists of third-party tools, data parsers, and data transformations. The pipeline tools and their dependencies are specified using a workflow language or implemented as a program or script. A pipeline framework executes the pipeline tools on a cluster or cloud using a big data processing engine or a supercomputer job scheduler. The pipeline framework stores the data as files, objects, or matrices in a columnar storage. The execution environment allocates the resources needed for the pipeline, and a user interface provides access for end users to the pipeline

2.1 META-Pipe 1.0 Metagenomics Pipeline

Our META-pipe pipeline [11, 12] provides preprocessing, assembly, taxonomic classification, and functional analysis for metagenomics samples. It takes as input short reads from a next-generation sequencing instrument and outputs the organisms found in the metagenomics sample, predicted genes, and their corresponding functional annotations. The different pipeline tools have different resource requirements. Assembly requires a machine with at least 256 GB RAM, and it cannot run efficiently on distributed resources. Functional analysis requires many cores and has parts that are I/O intensive, but it can be run efficiently distributed on a cluster with thin nodes. Taxonomical classification has low resource requirements and can be run on a single node. A typical dataset is 650 MB in size and takes about 6 h to assemble on 12 cores and 20 h for functional annotation on 384 cores.

A Galaxy [13] interface provides META-pipe 1.0 to Norwegian academic and industry users (https://nels.bioinfo.no/). The pipeline is specified in a custom Perl-script-based framework [14]. It is executed on the Stallo supercomputer, which is a traditional HPC cluster with one job queue optimized for long-executing batch jobs. A shared global file system provides data storage. We manually install and maintain the pipeline tools and associated database versions on a shared file system on Stallo.

The job script submitted to the Stallo job scheduler describes the resources requested on a node (scale up) and the number of nodes requested for the job (scale out). Both Galaxy and the job scheduler allow multiple job submissions from multiple users at the same time, but whether the jobs run simultaneously depends on the load of the cluster. HPC clusters are typically run with a high utilization, so jobs are often queued for a long time and therefore jobs submitted at the same time may not run at the same time. HPC clusters are not designed for elastic resource provision, so it is difficult to efficiently scale the backend to support the resource requirement variations of multi-user workloads.

META-pipe 1.0 has several limitations as a scalable bioinformatics service. First, the use of a highly loaded supercomputer causes long wait times and limits elastic adjustment of resources for multi-user workloads. We manually deploy the service on Galaxy and Stallo, which makes updates time-consuming and prone to errors. Finally, our custom pipeline framework has no support for provenance data maintenance nor failure handling. For these reasons, we have re-implemented the backend in META-pipe 2.0 using Spark [15] so that it can take advantage of the same features as the pipelines described below do.

2.2 Genome Analysis Toolkit (GATK) Variant Calling Reference Pipeline

The GATK [16] best practices pipeline for germline SNP and indel discovery in whole-genome and whole-exome sequence (https://software.broadinstitute.org/gatk/best-practices/bp_3step.php?case=GermShortWGS) is often used as reference for scalable genomics data analysis pipelines. This pipeline provides preprocessing, variant calling, and callset refinement. (The latter usually is not included in benchmarking.) It takes as input short reads and outputs annotated variants. Some tools have high CPU utilization (BWA and HaplotypeCaller), but most steps are I/O bound. An Intel white paper [17] recommends using a server with 256 GB RAM and 36 cores for the pipeline, and they achieved the best resource utilization by running analysis jobs for multiple datasets at the same time and configuring the jobs to only use a subset of the resources. The pipeline is well suited for parallel execution as demonstrated by the MapReduce programming models used in [16] and the Halvade [18] Hadoop MapReduce implementation that analyzes a 86 GB (compressed) WGS dataset in less than 3 h on Amazon Elastic MapReduce (EMR) using 16 workers with a total of 512 cores.

The first three versions of GATK are implemented in Java and optimized for use on local compute infrastructures. Version 4 of GATK (at the time of writing in Beta) uses Spark to improve I/O performance and scalability (https://software.broadinstitute.org/gatk/blog?id=9644). It uses GenomicsDB (https://github.com/Intel-HLS/GenomicsDB) for efficiently storing, querying, and accessing (sparse matrix) variant data. GenomicsDB is built on top of Intel's TileDB (http://istc-bigdata.org/tiledb/index.html) which is designed for scalable storage and processing of sparse matrices. To support tertiary (downstream) analysis of the data produced by GATK, the Hail framework (https://hail.is/) provides interactive analyses. It optimizes storage and access of variant data (sparse matrices) and provides built-in analysis functions. Hail is implemented using Spark and Parquet.

Tools in the GATK can be run manually through the command line, specified in the workflow definition language (WDL) and run in Cromwell, or use written in Scala and run on Queue (https://software.broadinstitute.org/gatk/documentation/pipelines). GATK provides multiple approaches to parallelize tasks: multi-threading and scatter–gather. Users enable multi-threading mode by specifying command-line flags and use Queue or Cromwell to run GATK tools using a scatter–gather approach. It is also possible to combine these approaches (https://software.broadinstitute.org/gatk/documentation/article.php?id=1988).

2.3 ADAM Variant Calling Pipeline

ADAM [6] is a genomics pipeline that is built on top of the Apache Spark big data processing engine [15], Avro (https://avro.apache.org/) data serialization system, and Parquet (https://parquet.apache.org/) columnar storage system to improve the performance and reduce the cost of variant calling. It takes as input next-generation sequencing (NGS) short reads and outputs sites in the input genome where an individual differs from the reference genome. ADAM provides tools to sort reads, remove duplicates, do local realignment, and do base quality score recalibration. The pipeline includes both compute and I/O-intensive tasks. A typical dataset is 234 GB (gzip compressed) and takes about 74 min to run on 128 Amazon EC2 r3.2xlarge (4 cores, 30.5 GB RAM, 80 GB SSD) instances with 1024 cores in total.

ADAM focuses on backend processing, and hence, user-facing applications need to be implemented as, for example, Scala or Python scripts. ADAM uses Spark to scale out parallel processing. The data are stored in Parquet, a columnar data storage using Avro serialized file formats that reduce I/O load by providing in-memory data access for the Spark pipeline implementation. The pipeline is implemented as a Spark program.

Spark is widely supported on commercial clouds such as Amazon EC2, Microsoft Azure HDInsight, and increasingly in smaller academic clouds. It can therefore exploit the scale and elasticity of these clouds. There are also Spark job schedulers that can run multiple jobs simultaneously.

ADAM improves on MapReduce-based pipeline frameworks by using Spark. Spark solves some of the limitations of the MapReduce programming model and runtime system. It provides a more flexible programming model than just the map-sort-reduce in MapReduce, better I/O performance by better use of in-memory data structures between pipeline stages and data streaming, and reduced job startup time for small jobs. Spark is therefore becoming the de facto standard for big data processing, and pipelines implemented in Spark can take advantage of Spark libraries

such as GraphX [19] for graph processing and MLlib [20] for machine learning.

ADAM has two main limitations. First, it implemented as part of research projects that may not have the long-term support and developer efforts required to achieve the quality and trust required for production services. Second, it requires re-implementing the pipeline tools to run in Spark and access data in Parquet, which is often not possible for analysis services with multiple pipelines with tens of tools each.

2.4 GESALL Variant Calling Pipeline

GESALL [21] is a genomic analysis platform for unmodified analysis tools that use the POSIX file system interface. An example pipeline implemented with GESALL is their implementation of the GATK variant calling reference pipeline that was used as an example in the ADAM paper [6]. GESALL is evaluated on fewer but more powerful nodes (15, each with 24 cores, 64 GB RAM, and 3 TB disk) than the ADAM pipeline. A 243 GB compressed dataset takes about 1.5 h to analyze.

GESALL pipelines are implemented and run as MapReduce programs on resources allocated by YARN (https://hadoop.apache.org/). The pipeline can run unmodified analysis tools by wrapping these using their genome data parallel toolkit. The tools access their data using the standard file system interface, but GESALL optimizes data access patterns and enables correct distributed execution. It stores data in HDFS (https://hadoop.apache.org/) and provides a layer on top of HDFS that optimizes storage of genomics data type, including custom partitioning and block placement.

Like Spark, MapReduce is widely used in both commercial and academic clouds and GESALL can therefore use the horizontal scalability, elasticity, and multi-job support features of these infrastructures. The unmodified tools executed by a GESALL pipeline may also be multi-threaded. A challenge is therefore to find the right mix of MapReduce tasks and per-tool multi-threading.

2.5 Toil: TCGA RNA-Seq Reference Pipeline

Toil is a workflow software to run scientific workflows on a large scale in cloud or high-performance computing (HPC) environments [22]. It is designed for large-scale analysis pipelines such as The Cancer Genome Atlas (TCGA) [23] best practices pipeline for calculating gene- and isoform-level expression values from RNA-seq data. The memory-intensive STAR [24] aligner requires 40 GB of memory. As with other pipelines, the job has a mix of I/O- and CPU-intensive tasks. In [22], the pipeline runs on a cluster of AWS c3.8xlarge (32 cores, 60 GB RAM, 640 GB SSD storage) nodes. Using about 32.000 cores, they processed a 108 TB with 19,952 samples in 4 days.

Toil can execute workflows written in both the Common Workflow Language (CWL, http://www.commonwl.org/), the Workflow Definition Language (WDL, https://github.com/broadinstitute/wdl), or Python. Toil is written in Python, so it is possible to interface it from any Python application using the Toil Application Programming Interface (API). Toil can be used to implement any type of data analysis pipeline, but it is optimized for I/O-bound NGS pipelines. Toil uses file caching and data streaming, and it schedules work on the same portions of a dataset to the same compute node. Toil can run workflows on commercial cloud platforms, such as AWS, and private cloud platforms, such as OpenStack (https://www.openstack.org/), and it can execute individual pipeline jobs on Spark. Users interface with Toil through a command-line tool that orchestrates and deploys a data analysis pipeline. Toil uses different storage solutions depending on platform: S3 buckets on AWS, the local file system on a desktop computer, network file systems on a high-performance cluster, and so on.

3 Current Trends

In addition to the scalability considerations discussed above, we see several other trends in pipelines developed for scalable bioinformatics services.

Containers are increasingly used to address the challenges of sharing bioinformatics tools and enabling reproducible analyses in projects such as BioContainers (http://biocontainers.pro/). A bioinformatics pipeline is often deep, with more than 15 tools [7]. Each tool typically has many dependencies on libraries and especially reference databases. In addition, some tools are seldom updated. Pipelines therefore often require a large effort to install, configure, and run bioinformatics tools. Software containerization packages an application and its dependencies in an isolated execution environment. One popular implementation of software container is Docker [25]. With Docker, developers can build a container from a configuration file (Dockerfile) that includes machine and human-readable instructions to install the necessary dependencies and the tool itself. Both the Dockerfile and the resulting container can be moved between machines without installing additional software, and the container can be rerun later with the exact same libraries. Containers can be orchestrated for parallel execution using, for example, Kubernetes (https://kubernetes.io/) or Docker Swarm (https://github.com/docker/swarm), and there are now multiple pipelining tools that use Docker or provide Docker container support including Nextflow [26], Toil [22], Pachyderm (http://www.pachyderm.io/), Luigi (https://

github.com/spotify/luigi) [27], Rabix/bunny [28], and our own walrus system (http://github.com/fjukstad/walrus).

There are several efforts to standardize pipeline specifications to make it easier to port pipelines across frameworks and execution environments (including Toil described above). For example, the Common Workflow Language (CWL) is an effort supported by many of the developers of the most popular pipeline frameworks. CWL is a standard for describing data analysis pipelines. Developers can describe a data analysis pipeline in YAML or JSON files that contain a clear description of tools, input parameters, input and output data, and how the tools are connected. There are multiple systems that implement the CWL standard, including Galaxy [13], Toil, Arvados (https://arvados.org/), and AWE [29], making it possible to write a single description of a pipeline and run it in the most suitable pipeline execution environment. It is an open challenge to implement support for the standardized pipeline descriptions on execution environments such as Spark.

The needs and challenges for reproducible analyses [30] require a standardized way to specify and document pipelines and all their dependencies, in addition to maintaining all provenance information of pipeline executions [31]. Specifications such as CWL can be used to standardize the specification, and for example, Spark has built-in data lineage recording. However, there is not yet an analysis standard that describes the minimum information required to recreate bioinformatics analyses [32].

Finally, there are several large infrastructures and platforms that provide scalable bioinformatics services. The European ELIXIR (https://www.elixir-europe.org/) distributed infrastructure for life science data resources, analysis tools, compute resources, interoperability standards, and training. The META-pipe pipelines described above are developed as part of the ELIXIR project. Another example is the Illumina BaseSpace Sequence Hub (https://basespace.illumina.com/home/index), which is a cloud-based genomics analysis and storage platform provided by the producer of the currently most popular sequencing machines. Other commercial cloud platforms for bioinformatics analyses are DNAnexus (https://www.dnanexus.com/), Agave (https://agaveapi.co), and SevenBridges (https://www.sevenbridges.com/platform/). We believe the efforts required to maintain and provide the resources needed for future bioinformatics analysis services will further consolidate such services in larger infrastructures and platforms.

4 Summary and Discussion

We have provided a survey of scalable bioinformatics pipelines. We compared their design and use of underlying infrastructures. We observe several trends (Table 1). First, there are few papers that describe the design, implementation, and evaluation of scalable pipeline frameworks and pipelines, especially compared to the number of papers describing bioinformatics tools. Of those papers, most focus on a specific type of analysis (variant calling) using mostly the same tools. This suggests that there is a need to address the scalability and cost-effectiveness of other types of bioinformatics analysis.

Most papers focus on the scalability of a single job. Only our META-pipe paper evaluates the scalability of the pipeline with respect to multiple users and simultaneous jobs. With analyses provided increasingly as a service, we believe multi-user job optimizations will become increasingly important. Staggered execution of multiple pipeline jobs can also improve resource utilization as shown in [17, 27].

It is becoming common to standardize pipeline descriptions and use existing pipeline frameworks rather than implementing custom job execution scripts. An open challenge is how to optimize the execution of pipelines specified in, for example, CWL. Frameworks such as GESALL provide genomic dataset optimized storage which can be difficult to utilize from a generic pipeline specification. ADAM uses an alternative approach where the pipeline is a Spark program like for data analyses in many other domains.

In addition to standardizing pipeline description, there is a move to standardize and enable completely reproducible execution environments through software containers such as Docker. Although not yet widely adopted, containerized bioinformatics tools simplify deployment, sharing, and reusing of tools between research groups. We believe that standardizing the execution environment, together with standardizing the pipeline descriptions, is a key feature for reproducible research in bioinformatics.

Most pipelines save data in a traditional file system since most analysis tools are implemented to read and write files in POSIX file systems. GESALL provides a layer that enables using HDFS for data storage by wrapping tools and providing optimized genomic data-specific mapping between POSIX and HDFS. ADAM uses a different, data-oriented approach, with a layered architecture for data storage and analysis that exploits recent advancement in big data analysis systems. Like ADAM, GATK4 is also built on top of Spark and a columnar data storage system. ADAM requires re-implementing the analysis tools, which may be practical for the most commonly used tools and pipelines such as the GATK reference pipelines, but is often considered impractical for the many other tools and hence pipelines.

Pipeline frameworks such as GESALL and ADAM use MapReduce and Spark to execute pipeline jobs on clouds or dedicated clusters, and Toil supports job execution on

Table 1 Classification of scalable bioinformatics pipelines

Pipeline	META-pipe 1.0 [11, 12]	GATK [16]	GESALL [21]	ADAM [6]	TOIL [22]
Application	Metagenomics	Variant calling	Variant calling	Variant calling	RNA-seq
User interface	Galaxy	Command line, REST API	Command line	Scala or Python scripts	Command line
Pipeline specification	Perl script	WDL or bash scripts	MapReduce program	Spark program	CWL or Python
Data storage	File system	File system	Layer on top of HDFS	Parquet columnar storage	S3 buckets or file system
Pipeline execution	Torque job scheduler	JVM, Cromwell, Spark	MapReduce (YARN)	Spark	Toil
Execution environment	HPC cluster	HPC Cluster, Cloud	HPC cluster, cloud	Cloud	HPC cluster, cloud
Scalability	Parallel processes and multi-threaded programs	Multi-threading and scatter–gather	MapReduce tasks and multi-threaded programs	Spark tasks	Distributed and parallel workers

HPC clusters with a job scheduler, which is important since most bioinformatics analysis pipelines are not implemented in MapReduce or Spark. HPC job schedulers are also provided on commercial and private clouds, so also these pipelines can take advantage of the elasticity provided by these infrastructures.

In addition to enabling and evaluating horizontal scalability, the cost of an analysis and the choice of virtual machine flavors are becoming increasingly important for efficient execution of bioinformatics analysis, since pipelines are increasingly deployed and evaluated on commercial clouds [6, 21, 22]. However, even on dedicated clusters it is important to understand how to scale a pipeline up and out on the available resources to improve the utilization of the resources. However, with the exception of [17], none of the reviewed papers have evaluated multiple pipeline job executions from the cluster provider's point of view.

We believe deployment, provenance data recording, and standardized pipeline descriptions are necessary to provide easy-to-maintain and reproducible bioinformatics pipelines in infrastructures such as ELIXIR or platforms such as BaseSpace. These three areas are typically not addressed in the reviewed papers, suggesting that more research is required to address these areas in the context of scalable bioinformatics pipelines.

Summarized, we have described many scalability problems and their solutions in the reviewed papers. These include: scaling up nodes to run tools with large memory requirements (META-pipe), scale out for parallel execution (all reviewed pipelines), use of optimized data structures and storage systems to improve I/O performance (GATK 4.0, ADAM, GESALL), and the choice of machine flavor to optimize either the execution or cost (GATK, ADAM, GESALL). Although many of the pipelines have the same scalability issues, such as I/O performance for variant calling, the infrastructure system and optimizations differ depending on overall design choices (e.g., the use of unmodified vs modified analysis tools) and the software stack (e.g., Spark vs HPC schedulers and file systems). We therefore believe there is no right solution or platform that solves all scalability problems, and that more research is needed to scale up and cost-optimize the many types of bioinformatics data analyses. The increasing use of standardized layers, standards, and interfaces to implement these analyses should allow reusing the developed solutions across pipelines and pipeline frameworks.

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Sboner A, Mu XJ, Greenbaum D et al (2011) The real cost of sequencing: higher than you think! Genome Biol 12:125. doi:10.1186/gb-2011-12-8-125
2. Schuster SC (2008) Next-generation sequencing transforms today's biology. Nat Methods 5:16–18. doi:10.1038/nmeth1156
3. Vollmers J, Wiegand S, Kaster A-K (2017) Comparing and evaluating metagenome assembly tools from a microbiologist's

perspective—not only size matters! PLoS ONE 12:e0169662. doi:10.1371/journal.pone.0169662
4. Couger MB, Pipes L, Squina F et al (2014) Enabling large-scale next-generation sequence assembly with Blacklight. Concurr Comput Pract Exp 26:2157–2166. doi:10.1002/cpe.3231
5. Altschul SF, Gish W, Miller W et al (1990) Basic local alignment search tool. J Mol Biol 215:403–410. doi:10.1016/S0022-2836(05)80360-2
6. Nothaft FA, Massie M, Danford T et al (2015) Rethinking data-intensive science using scalable analytics systems. In: Proceedings of 2015 ACM SIGMOD international conference on management of data. ACM, New York, pp 631–646
7. Diao Y, Abhishek R, Bloom T (2015) Building highly-optimized, low-latency pipelines for genomic data analysis. In: Proceedings of the 7th biennial Conference on Innovative Data Systems Research (CIDR 2015)
8. Blankenberg D, Von Kuster G, Bouvier E et al (2014) Dissemination of scientific software with Galaxy ToolShed. Genome Biol 15:403. doi:10.1186/gb4161
9. Wolstencroft K, Haines R, Fellows D et al (2013) The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. Nucleic Acids Res 41:W557–W561. doi:10.1093/nar/gkt328
10. Leipzig J (2016) A review of bioinformatic pipeline frameworks. Br Bioinform. doi:10.1093/bib/bbw020
11. Robertsen EM, Kahlke T, Raknes IA et al (2016) META-pipe—pipeline annotation, analysis and visualization of marine metagenomic sequence data. ArXiv160404103 Cs
12. Robertsen EM, Denise H, Mitchell A et al (2017) ELIXIR pilot action: marine metagenomics—towards a domain specific set of sustainable services. F1000Research 6:70. doi:10.12688/f1000research.10443.1
13. Afgan E, Baker D, van den Beek M et al (2016) The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update. Nucleic Acids Res 44:W3–W10. doi:10.1093/nar/gkw343
14. Pedersen E, Raknes IA, Ernstsen M, Bongo LA (2015) Integrating data-intensive computing systems with biological data analysis frameworks. In: 2015 23rd Euromicro international conference on parallel, distributed and network-based processing (PDP). IEEE Computer Society, Los Alamitos, pp 733–740
15. Zaharia M, Franklin MJ, Ghodsi A et al (2016) Apache Spark: a unified engine for big data processing. Commun ACM 59:56–65. doi:10.1145/2934664
16. McKenna A, Hanna M, Banks E et al (2010) The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. Genome Res 20:1297–1303. doi:10.1101/gr.107524.110
17. Prabhakaran A, Shifaw B, Naik M et al (2015) Infrastructure for GATK* best practices pipeline deployment. Intel, Santa Clara
18. Decap D, Reumers J, Herzeel C et al (2017) Halvade-RNA: parallel variant calling from transcriptomic data using MapReduce. PLoS ONE 12:e0174575. doi:10.1371/journal.pone.0174575
19. Gonzalez JE, Xin RS, Dave A et al (2014) GraphX: graph processing in a distributed dataflow framework. In: Proceedings of 11th USENIX conference on operating systems design and implementation. USENIX Association, Berkeley, pp 599–613
20. Meng X, Bradley J, Yavuz B et al (2016) MLlib: machine learning in Apache Spark. J Mach Learn Res 17:1235–1241
21. Roy A, Diao Y, Evani U et al (2017) Massively parallel processing of whole genome sequence data: an in-depth performance study. In: Proceedings of 2017 ACM international conference on management of data. ACM, New York, pp 187–202
22. Vivian J, Rao AA, Nothaft FA et al (2017) Toil enables reproducible, open source, big biomedical data analyses. Nat Biotechnol 35:314–316. doi:10.1038/nbt.3772
23. The Cancer Genome Atlas Research Network, Weinstein JN, Collisson EA et al (2013) The cancer genome atlas pan-cancer analysis project. Nat Genet 45:1113–1120. doi:10.1038/ng.2764
24. Dobin A, Davis CA, Schlesinger F et al (2013) STAR: ultrafast universal RNA-seq aligner. Bioinformatics 29:15–21. doi:10.1093/bioinformatics/bts635
25. Merkel D (2014) Docker: lightweight linux containers for consistent development and deployment. Linux J 239:2
26. Di Tommaso P, Chatzou M, Floden EW et al (2017) Nextflow enables reproducible computational workflows. Nat Biotechnol 35:316–319. doi:10.1038/nbt.3820
27. Schulz WL, Durant T, Siddon AJ, Torres R (2016) Use of application containers and workflows for genomic data analysis. J Pathol Inform 7:53. doi:10.4103/2153-3539.197197
28. Kaushik G, Ivkovic S, Simonovic J et al (2016) Rabix: an open-source workflow executor supporting recomputability and interoperability of workflow descriptions. Pac Symp Biocomput Pac Symp Biocomput 22:154–165
29. Gerlach W, Tang W, Keegan K et al (2014) Skyport: container-based execution environment management for multi-cloud scientific workflows. In: Proceedings of 5th international workshop on data-intensive computing in the clouds. IEEE Press, Piscataway, pp 25–32
30. Peng RD (2011) Reproducible research in computational science. Science 334:1226–1227. doi:10.1126/science.1213847
31. Huntemann M, Ivanova NN, Mavromatis K et al (2016) The standard operating procedure of the DOE-JGI Metagenome Annotation Pipeline (MAP v.4). Stand Genomic Sci 11:17. doi:10.1186/s40793-016-0138-x
32. ten Hoopen P, Finn RD, Bongo LA et al (2017) The metagenomics data life-cycle: standards and best practices. GigaScience. doi:10.1093/gigascience/gix047

Paper 5

I. A. Raknes, B. Fjukstad, and L. Bongo, “nsroot: Minimalist process isolation tool implemented with linux namespaces,” *Norsk Informatikkonferanse*, 2017

nsroot: Minimalist Process Isolation Tool Implemented with Linux Namespaces

Inge Alexander Raknes[1], Bjørn Fjukstad[2], Lars Ailo Bongo[2]

[1]Dept. of Chemistry, UiT – The Arctic University of Norway
[2]Dept. of Computer Science, UiT – The Arctic University of Norway
{inge.a.raknes, bjorn.fjukstad, lars.ailo.bongo}@uit.no

Abstract

Data analyses in the life sciences are moving from tools run on a personal computer to services run on large computing platforms. This creates a need to package tools and dependencies for easy installation, configuration and deployment on distributed platforms. In addition, for secure execution there is a need for process isolation on a shared platform. Existing virtual machine and container technologies are often more complex than traditional Unix utilities, like chroot, and often require root privileges in order to set up or use. This is especially challenging on HPC systems where users typically do not have root access. We therefore present nsroot, a lightweight Linux namespaces based process isolation tool. It allows restricting the runtime environment of data analysis tools that may not have been designed with security as a top priority, in order to reduce the risk and consequences of security breaches, without requiring any special privileges. The codebase of nsroot is small, and it provides a command line interface similar to chroot. It can be used on all Linux kernels that implement user namespaces. In addition, we propose combining nsroot with the AppImage format for secure execution of packaged applications. nsroot is open sourced and available at: https://github.com/uit-no/nsroot.

Introduction

Container and virtual machine technologies are vital for secure and portable execution on computing platforms such as clouds IaaS. For scientific computing the use of container technologies, such as Docker [1], rkt (https://coreos.com/rkt/) and Snapcraft (http://snapcraft.io/), have the potential to make data analysis applications easier to install, port, and use. Specifically, containers allow packaging the application with its dependencies in a self-contained container that can easily be executed on a wide variety of computing platforms. However, container technologies rely on operating system services to provide the application with a name space and protection from other application. Hence, these must rely on operating system features that may not be available in all kernels and distributions, or they may require root access.

Our work is motivated by our experiences developing, deploying, and maintaining the META-pipe metagenomics data analysis service [2]. Such genomic data analysis is typically done through a collection of tools arranged in a pipeline where the output of one tool is the input to the next tool. The data transformations include file conversion, data cleaning, normalization, and data integration. A specific biological data analysis project often requires a deep workflow that combines 10-20 tools [3]. There are many libraries [4]–[6] with hundreds of tools, ranging from small, user-created scripts to large, complex applications [7]. Each tool has dependencies in form of libraries, but also data files such as specific versions of reference databases. These dependencies make it difficult to install, configure, and deploy a pipeline on a new computing platform. There are therefore recent initiatives, such as BioDocker (https://github.com/BioDocker) that aim to provide packaged genomics data analysis tools in containers.

The cost of producing genomics data is rapidly decreasing (http://www.genome.gov/sequencingcosts), so it has become necessary to move the

This paper was presented at the NIK 2017 conference. For more information see http://www.nik.no/

analyses from the single server typically used to high performance clusters or clouds [2], [3], [8]. Since porting and maintaining a deep data analysis pipeline on a cluster or cloud is time-consuming and requires access to a large amount of compute and storage resources, it is increasingly common to provide a pipeline as a data analysis service. Recent examples of such services include Illumina Basespace (https://basespace.illumina.com/home/index), EBI Metagenomics [9], our META-pipe [2] and other pipelines provided by the Norwegian eInfrastructure for Life Sciences (https://nels.bioinfo.no/). For such services security is important since the service may be used to process sensitive, scientifically valuable, or commercially valuable data on behalf of a user. This is further complicated since the pipeline consists of third party tools that may not have high code quality, that have not been properly tested, and that are executed as the same user on the cloud or HPC backend. So, it is especially important to isolate the individual pipeline tool executions from each other. Another challenge is that, unlike IaaS services, HPC systems are often provided as a traditional shared UNIX system where users typically do not have root access. This complicates the task of installing complex application dependencies.

An ideal container solution would be one that would let us package all of our application dependencies inside the same Jar-file that we submit to our Spark cluster while at the same time provide full process isolation for the 3rd party tools that are run within the workflow. This would also make it easy to know which tools (and versions) were involved in processing a dataset since they would all be part of the application bundle. However, to our knowledge no existing solution existed that satisfies all needs. Existing virtual machine and container technologies are often more complex than traditional Unix utilities, like chroot, and often require root privileges.

In this paper we discuss our solution for minimalistic process isolation. It solves the problem of isolating an application from the rest of the filesystem and it also lets us restrict an application from communicating with other applications via network or IPC by using the network- and IPC namespaces. Our solution, called *nsroot* provides process isolation by making use of Linux namespaces in order to provide an isolated root file system for a process (similar to chroot) while at the same time cutting it off the network. This enables us to run applications with all their dependencies is a single directory while at the same time providing extra security through extra process isolation. In addition, we can combine *nsroot* with AppImage container to provide a simple, packaged bundle for an application. Finally, we propose using *nsroot* with AppImage and package those inside a Java Jar file for easy submission to a Spark cluster.

nsroot works on Linux kernels newer than 3.10, and we have successfully tested it on the Fedora 22 and Ubuntu 14.04 LTS.

nsroot

nsroot (https://github.com/uit-no/nsroot) is a process isolation tool that makes use of namespaces to allow an application to change the root directory. It is designed to be lightweight and minimalistic to ensure wide portability across Linux kernel versions and distributions. It can be used to run processes within their own virtual root file system, which in turn allows the caller full control over absolute paths that are referenced by the program.

Unlike other container tools, like Docker, nsroot does not require any installation. Instead it can be packaged as part of an application bundle.

Usage

To use *nsroot*, we first create a new root file system containing the application and all its dependencies. The root filesystem can be created by for example setting up the application in a docker container and then using docker to export the application files and dependencies to the new root directory. Alternatively, the root file system can be manually created, or it may consist of a statically linked self-contained application. We also describe below how nsroot can be combined with AppImage to provide extra isolation for containers.

Second, we create a new directory for the old root filesystem in a subdirectory of the new root. This is needed by the PIVOT_ROOT system call (http://man7.org/linux/man-pages/man2/pivot_root.2.html) that changes the root directory.

Third, we call *nsroot* with the same parameters as chroot to change into the new filesystem. We can also specify bind-mounts to mount existing directories into this new root file system. To further restrict which processes the contained process can communicate with we can apply the network and IPC namespaces. We can validate that the network namespace is used by calling ifconfig inside the container.

Namespaces

We implemented *nsroot* by using namespaces as described in the NAMESPACES man page (http://man7.org/linux/man-pages/man7/namespaces.7.html). In particular, we are using mount namespaces in order to change the mount table for a single process tree and we are using the user namespace in order to do so without requiring root access (https://www.toptal.com/linux/separation-anxiety-isolating-your-system-with-linux-namespaces). We also found a tutorial about mount namespaces useful for this project (https://www.ibm.com/developerworks/library/l-mount-namespaces).

As described in the namespaces man page: “User namespaces isolate security-related identifiers and attributes, in particular, user IDs and group IDs […], the root directory, keys […], and capabilities […]. A process's user and group IDs can be different inside and outside a user namespace. In particular, a process can have a normal unprivileged user ID outside a user namespace while at the same time having a user ID of 0 inside the namespace; in other words, the process has full privileges for operations inside the user namespace, but is unprivileged for operations outside the namespace.”

In other words, the process can run with root privileges in a user namespace without actually having root privileges elsewhere. Summarized, our code does the following:

1. Clone the process using the appropriate namespaces. This includes at least the user and mount namespaces (CLONE_NEWUSER, CLONE_NEWNS)
2. Subprocess bind-mounts the new root filesystem into a subdirectory in the new root (for example /path/to/newroot/mnt).
3. pivot_root is called in order to make newroot the new root filesystem. The old root filesystem becomes available under /mnt
4. User-specified shared directories are bind mounted from the old root filesystem (available under /mnt) into the new root file system.
5. The old root filesystem (/mnt) is unmounted
6. execvp is called in order to replace the current process image with an executable specified by the user.

Implementation

nsroot is implemented in C and it consists of less than 500 lines of code. It has no dependencies other than the kernel. User namespaces were implemented in the Linux kernel version 3.8. We have tested *nsroot* with Linux kernels newer than 3.10 (Fedora 22 and Ubuntu 14.04). However, it does not currently work on CentOS since it does not yet support user namespaces (http://rhelblog.redhat.com/2015/07/07/whats-next-for-containers-user-namespaces/).

Related Work

In this section, we describe related work in process isolation tools based on Linux user namespaces and container technologies. In addition, we discuss how containers are used for genomics data analysis.

nsjail and firejail

At the same time when we developed nsroot, other projects developed similar tools. nsjail (http://google.github.io/nsjail/) was started the day before nsroot, and firejail (https://firejail.wordpress.com/) had its first beta release in April 2014. Both *nsjail* and *firejail* uses Linux user namespaces, similarly to *nsroot*, for process isolation, and in addition they use seccomp-bpf to filter system calls. *nsjail* uses resource limits and cgroups to limit the CPU, memory usage, and address space sizes of applications. *nsjail* also provides virtual network inside the application (although this requires root access to setup). *firejail* natively supports the AppImage format.

The main advantage of *nsroot* over *nsjail* and *firejail* is that *nsroot* maintains compatibility with the chroot command and it has a smaller codebase (since it has fewer features). We believe this makes the command line interface easier to understand and use.

Containers and Virtual Machines

Both Virtual machines (VMs) and containers are technologies used to provide an isolated environment for applications. They are both binaries that can be transferred between hosts. While a VM includes an application, its dependencies and an entire OS, containers share the underlying host OS and contain only the application and dependencies making them smaller in size. Containers are isolated processes in user space on the host OS (as described in https://www.docker.com/what-docker).

In [10] Docker containers are shown to have better than, or equal performance as VMs. Both forms of virtualization techniques introduce overhead in I/O-intensive workloads, especially in VMs, but introduce negligible CPU and memory overhead.

For genomics pipelines the overhead of Docker containers will be negligible since these tend to be compute intensive and they typically run for several hours [11].

Containers in Life Science Research

Recent projects propose to use containers for life science research. The BioContainers (https://github.com/biocontainers) and Bioboxes [12] projects addresses the challenge of installing bioinformatics data analysis tools by maintaining a repository of docker containers for commonly used data analysis tools.

Containers have also been proposed as a solution to improve experiment reproducibility, by ensuring that the data analysis tools are installed with the same responsibilities [13].

Future Directions

We believe nsroot can be used to provide extra isolation for minimalistic container technologies. These can then be used to package and execute for example data analysis tools as Spark jobs. This combination makes it easier to port legacy genomics data analysis pipelines to the Spark framework. In this section we present our proposed solution.

AppImage

AppImage (http://appimage.org/) is a technology for packaging applications for Linux. An AppImage application is a single ELF executable file that contains all the application's dependencies and it is self-contained. *nsroot* can be used in an AppImage to provide extra isolation via Linux namespaces.

Using *nsroot* within an AppImage has several advantages. First, it provides software packaging to nsroot. Second, it allows to run less-than-secure AppImages in isolation. Third, *nsroot* removes the problem of absolute paths. Fourth, it increases security by stricter isolation, which is especially beneficial if the installed AppImage becomes old or unpatched.

The above observations are also shared by the *firejail* (https://github.com/netblue30/firejail) tool that is implemented similarly to *nsroot* and that can already be used with the AppImage format.

Spark Pipelines

Bioinformatics workflows running on Apache Spark could benefit from using *nsroot* for isolating the execution of legacy 3rd party tools. In particular, we could package all of our application dependencies inside the same Jar-file that we submit to our Spark cluster using AppImage, and then use *nsroot* to provide full process isolation for the 3rd party tools that are run within the workflow.

Teaching Tool

We believe that the minimalistic design, and the 420 lines code, of *nsroot* makes it useful as a teaching tool. For example, *nsroot* could be used in a project about containers in an operating systems course.

Conclusion

We have presented our *nsroot* process isolation tool. It uses the Linux user namespaces to allow application processes to run with root privileges without actually having root privileges outside of the user namespace. We propose using nsroot for secure packaged application execution, and for Spark job distribution.

Our work was motivated by our experiences developing a genomics data analysis service that we have designed for cloud execution. We plan to use *nsroot*, and the proposed Spark solution to install, configure, deploy, and securely execute the data analysis tools for user submitted data using a single cloud user. Although we focused on genomics data analysis, we believe our solution can also be used for data analysis pipelines in other fields. *nsroot* and the proposed solutions are especially suited for deep pipelines with potentially insecure third-party tools.

As future work we intend to implement the proposed integration with AppImage and the Java jar file deployment. We also plan to explore the possibility of providing versioned

datasets to ensure complete reproducibility of a pipeline using for example Pachyderm (http://pachyderm.io/), which is a project that provides version control for data and a containerized pipeline system.

nsroot is open sourced and available at: https://github.com/uit-no/nsroot.

Acknowledgment

This work was done as part of the Norwegian E-infrastructure for Life Sciences (NeLS) project. This work was funded by the Research Council of Norway.

References

[1] D. Merkel, "Docker: Lightweight Linux Containers for Consistent Development and Deployment," *Linux J*, vol. 2014, no. 239, Mar. 2014.

[2] E. M. Robertsen *et al.*, "META-pipe - Pipeline Annotation, Analysis and Visualization of Marine Metagenomic Sequence Data," *ArXiv160404103 Cs*, Apr. 2016.

[3] Y. Diao, R. Abhishek, and T. Bloom, "Building Highly-Optimized, Low-Latency Pipelines for Genomic Data Analysis," presented at the CIDR, 2015.

[4] J. Hillman-Jackson, D. Clements, D. Blankenberg, J. Taylor, A. Nekrutenko, and G. Team, "Using Galaxy to Perform Large-Scale Interactive Data Analyses," in *Current Protocols in Bioinformatics*, John Wiley & Sons, Inc., 2002.

[5] R. C. Gentleman *et al.*, "Bioconductor: open software development for computational biology and bioinformatics," *Genome Biol.*, vol. 5, no. 10, p. R80, 2004.

[6] J. E. Stajich *et al.*, "The Bioperl Toolkit: Perl Modules for the Life Sciences," *Genome Res.*, vol. 12, no. 10, pp. 1611–1618, Oct. 2002.

[7] J. Leipzig, "A review of bioinformatic pipeline frameworks," *Brief. Bioinform.*, p. bbw020, Mar. 2016.

[8] F. A. Nothaft *et al.*, "Rethinking Data-Intensive Science Using Scalable Analytics Systems," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, 2015, pp. 631–646.

[9] A. Mitchell *et al.*, "EBI metagenomics in 2016 - an expanding and evolving resource for the analysis and archiving of metagenomic data," *Nucleic Acids Res.*, p. gkv1195, Nov. 2015.

[10] P. Di Tommaso, E. Palumbo, M. Chatzou, P. Prieto, M. L. Heuer, and C. Notredame, "The impact of Docker containers on the performance of genomic pipelines," *PeerJ*, vol. 3, p. e1273, Sep. 2015.

[11] P. Belmann, J. Dröge, A. Bremges, A. C. McHardy, A. Sczyrba, and M. D. Barton, "Bioboxes: standardised containers for interchangeable bioinformatics software," *GigaScience*, vol. 4, p. 47, 2015.

[12] C. Boettiger, "An Introduction to Docker for Reproducible Research," *SIGOPS Oper Syst Rev*, vol. 49, no. 1, pp. 71–79, Jan. 2015.

Paper 6

B. Fjukstad, V. Dumeaux, M. Hallett, and L. A. Bongo, "Reproducible data analysis pipelines for precision medicine," To appear in the proceedings of 2019 27th Euromicro International Conference On Parallel, Distributed and Network-based Processing (PDP). IEEE, 2019

Reproducible Data Analysis Pipelines for Precision Medicine

Bjørn Fjukstad*, Vanessa Dumeaux‡, Michael Hallett§, and Lars Ailo Bongo*
* Department of Computer Science
UiT The Arctic University of Norway
‡PERFORM Centre
Concordia University
§Department of Biology
Concordia University

Abstract

Precision medicine brings the promise of more precise diagnosis and individualized therapeutic strategies from analyzing a cancer's genomic signature. Technologies such as high-throughput sequencing enable cheaper data collection at higher speed, but rely on modern data analysis platforms to extract knowledge from these high dimensional datasets. Since this is a rapidly advancing field, new diagnoses and therapies often require tailoring of the analysis. These pipelines are therefore developed iteratively, continuously modifying analysis parameters before arriving at the final results. To enable reproducible results it is important to record all these modifications and decisions made during the analysis process.

We built a system, `walrus`, to support reproducible analyses for iteratively developed analysis pipelines. The approach is based on our experiences developing and using deep analysis pipelines to provide insights and recommendations for treatment in an actual breast cancer case. We designed `walrus` for the single servers or small compute clusters typically available for novel treatments in the clinical setting. `walrus` leverages software containers to provide reproducible execution environments, and integrates with modern version control systems to capture provenance of data and pipeline parameters.

We have used `walrus` to analyze a patient's primary tumor and adjacent normal tissue, including subsequent metastatic lesions. Although we have used `walrus` for specialized analyses of whole-exome sequencing datasets, it is a general data analysis tool that can be applied in a variety of scientific disciplines.

I. Introduction

Precision medicine uses patient-specific molecular information to diagnose and categorize disease to tailor treatment to improve health outcome.[1] Important goals in precision medicine are to learn about the variability of the molecular characteristics of individual tumors, their relationship to outcome, and to improve diagnosis and therapy.[2] International cancer institutions are therefore offering dedicated personalized medicine programs.

Bjørn Fjukstad is now at DIPS AS.

For cancer, high throughput sequencing is an emerging technology to facilitate personalized diagnosis and treatment since it enables collecting high quality genomic data from patients at a low cost. Data collection is becoming cheaper, but the downstream computational analysis is still time consuming and thereby a costly part of the experiment. This is because of the manual efforts to set up, analyze, and maintain the analysis pipelines. These pipelines consist of a large number of steps that transform raw data into interpretable results.[3] These pipelines often consists of in-house or third party tools and scripts that each transform input files and produce some output. Although different tools exist, it is necessary to carefully explore different tools and parameters to choose the most efficient to apply for a dedicated question.[4] The complexity of the tools vary from toolkits such as the Genome Analysis Toolkit (GATK) to small custom *bash* or *R* scripts. In addition some tools interface with databases whose versions and content will impact the overall result.[5]

Improperly developed analysis pipelines for precision medicine may generate inaccurate results, which may have negative consequences for patient care.[6] When developing analysis pipelines for use in precision medicine it is therefore necessary to track pipeline tool versions, their input parameters, and data. Both to thoroughly document what produced the final clinical reports, but also for iteratively improving the quality of the pipeline during development. Because of the iterative process of developing the analysis pipeline, it is necessary to use analysis tools that facilitate modifying pipeline steps and adding new ones with little developer effort.

A. Breast Cancer Diagnosis and Treatment

We have previously analyzed DNA sequence data from a breast cancer patient's primary tumor and adjacent normal cells to identify the molecular signature of the patient's tumor and germline. When the patient later relapsed we analyzed sequence data from the patient's metastasis to provide an extensive comparison against the primary and to identify the molecular drivers of the patient's tumor.

We used Whole-Genome Sequencing (WGS) to sequence the primary tumor and adjacent normal cells at an average depth of 20, and Whole-Exome Sequencing (WES) at an average depth of 300. The biological samples were sequenced at

the Genome Quebec Innovation Centre and we stored the raw datasets on our in-house server. From the analysis pipelines we generated reports with end results, such as detected somatic mutations, that was distributed to both the patient and the treating oncologists. These could be used to guide diagnosis and treatment, and give more detailed insight into both the primary and metastasis. When the patient relapsed we analyzed WES data using our own pipeline manager, `walrus`, to investigate the metastasis and compare it to the primary tumor.

For the initial WGS analysis we developed a pipeline to investigate somatic and germline mutations based on Broad Institute's best practices. We developed the analysis pipeline on our in-house compute server using a *bash* script version controlled with *git* to track changes as we developed the analysis pipeline. The pipeline consisted of tools including picard,[1] fastqc,[2] trimmomatic,[3] and the GATK.[4] While the analysis tools themselves provide the necessary functionality to give insights in the disease, ensuring that the analyses could be fully reproduced later left areas in need of improvement.

We chose a command-line script over more complex pipeline tools or workbenches such as Galaxy[7] because of its fast setup time on our available compute infrastructure, and familiar interface. More complex systems could be beneficial in larger research groups with more resources to compute infrastructure maintenance, whereas command-line scripting languages require little infrastructure maintenance over normal use. In addition, while there are off-site solutions for executing scientific workflows, analyzing sensitive data often put hard restrictions on where the data can be stored and analyzed.

After we completed the first round of analyses we summarized our efforts and noted some lessons learned.

- Datasets and databases should be version controlled and stored along with the pipeline description. In the analysis script we referenced to datasets and databases by their physical location on a storage system, but these were later moved without updating the pipeline description causing extra work. A solution would be to add the data to the same version control repository hosting the pipeline description.
- The specific pipeline tools should also be kept available for later use. Since installing many bioinformatics tools require a long list of dependencies, it is beneficial to store the pipeline tools to reduce the time to start analyzing new data or re-run analyses.
- It should be easy to add new tools to an existing pipeline and execution environment. This includes installing the specific tool and adding to an existing pipeline. Bundling tools within software containers, such as Docker, and hosting them on an online registry simplifies the tool installation process since the only requirement is the container runtime.
- While bash scripts have their limitations, using a well-known format that closely resembles the normal command-line use clearly have its advantages. It is easy to understand what tools were used, their input parameters, and the data flow. However, from our experience when these analysis scripts grow too large they become too complex to modify and maintain.
- While there are new and promising state-of-the art pipeline managers, many of these also require state-of-the-art computing infrastructure to run. This may not be the case for the current research and hospital environments.

The above problem areas are not just applicable to our research group, but common to other research and precision medicine projects as well. Especially when hospitals and research groups aim to apply personalized medicine efforts to guide therapeutic strategies and diagnosis, the analyses will have to be able to be easily reproducible later. We used the lessons learned to design and implement `walrus`, a command line tool for developing and running data analysis pipelines. It automatically orchestrates the execution of different tools, and tracks tool versions and parameters, as well as datasets through the analysis pipeline. It provides users a simple interface to inspect differences in pipeline runs, and retrieve previous analysis results and configurations. In the remainder of the paper we describe the design and implementation of `walrus`, its clinical use, its performance, and how it relates to other pipeline managers.

II. WALRUS

`walrus` is a tool for developing and executing data analysis pipelines. It stores information about tool versions, tool parameters, input data, intermediate data, output data, as well as execution environments to simplify the process of reproducing data analyses. Users write descriptions of their analysis pipelines using a familiar syntax and `walrus` uses this description to orchestrate the execution of the pipeline. In `walrus` we package all tools in software containers to capture the details of the different execution environments. While we have used `walrus` to analyse high-throughput datasets in precision medicine, it is a general tool that can analyze any type of data, e.g. image datasets for machine learning. It has few dependencies and runs on on any platform that supports Docker containers. While other popular pipeline managers require the use of cluster computers or cloud environment, we focus on single compute nodes often found in clinical environments such as hospitals.

`walrus` is implemented as a command-line tool in the Go programming language. We use the popular software container implementation Docker[5] to provide reproducible execution environments, and interface with git together with git-lfs[6] to version control datasets and pipeline descriptions. By choosing Docker and git we have built a tool that easily integrates with current bioinformatic tools and workflows. It runs both natively or within its own Docker container to simplify its installation process.

[1] broadinstitute.github.io/picard
[2] bioinformatics.babraham.ac.uk/projects/fastqc
[3] usadellab.org/cms/?page=trimmomatic
[4] software.broadinstitute.com/gatk
[5] docker.com
[6] git-lfs.github.com

With walrus we target pipeline developers that use command-line tools and scripting languages to build and run analysis pipelines. Users can use existing Docker containers from sources such as BioContainers,[8] or build containers with their own tools. We integrate with the current workflow using git to version control analysis scripts, and use git-lfs for versioning of datasets as well. The pipeline description format in walrus resembles standard command line syntax. In addition, walrus automatically track and version input, intermediate, and output files without users having to explicitly declare these in the description.

A. Pipeline Configuration

Users configure analysis pipelines by writing pipeline description files in a human readable format such as JavaScript Object Notation (JSON) or YAML Ain't Markup Language (YAML). A pipeline description contains a list of stages, each with inputs and outputs, along with optional information such as comments or configuration parameters such as caching rules for intermediate results. Listing 1 shows an example pipeline stage that uses MuTect[9] to detect somatic point mutations. Users can also specify the tool versions by selecting a specific Docker image, for example using MuTect version 1.1.7 as in Listing 1, line 3.

Users specify the flow of data in the pipeline within the pipeline description, as well as the dependencies between the steps. Since pipeline configurations can become complex, users can view their pipelines using an interactive web-based tool, or export their pipeline as a DOT file for visualization in tools such as Graphviz.[7]

Listing 1. Example pipeline stage for a tool that detects somatic point mutations. It reads a reference sequence file together with both tumor and normal sequences, and produces an output file with the detected mutations.

```
{
  "Name": "mutect",
  "Image": "fjukstad/mutect:1.1.7",
  "Cmd": [
    "--analysis_type","MuTect",
    "--reference_sequence","/walrus/input/reference.fasta",
    "--input_file:normal","/walrus/input/normal.bam",
    "--input_file:tumor","/walrus/input/tumor.bam",
    "-L","/walrus/input/targets.bed",
    "--out","/walrus/mutect/mutect-stats-txt",
    "--vcf","/walrus/mutect/mutect.vcf"
  ],
  "Inputs":[
     "input"
  ]
}
```

Users add data to an analysis pipeline by specifying the location of the input data in the pipeline description, and walrus automatically mounts it to the container running the analysis. The location of the input files can either be local or remote locations such as an FTP server. When the pipeline is completed, walrus will store all the input, intermediate and output data to a user-specified location.

B. Pipeline Execution

When users have written a pipeline description for their analyses, they can use the command-line interface of walrus to run the analysis pipeline. walrus builds an execution plan from the pipeline description and runs it for the user. It uses the input and output fields of each pipeline stage to construct a Directed Acyclic Graph (DAG) where each node is a pipeline stage and the links are input/output data to the stages. From this graph walrus can determine parallel stages and coordinate the execution of the pipeline.

In walrus, each pipeline stage is run in a separate container, and users can specify container versions in the pipeline description to specify the correct version of a tool. We treat a container as a single executable and users specify tool input arguments in the pipeline description file using standard command line syntax. walrus will automatically build or download the container images with the analysis tools, and start these with the user-defined input parameters and mount the appropriate input datasets. While the pipeline is running, walrus monitors running stages and schedules the execution of subsequent pipeline stages when their respective input data become available. We have designed walrus to execute an analysis pipeline on a single large server, but since the tools are run within containers, these can easily be orchestrated across a range of servers in future versions.

Users can select from containers pre-installed with bioinformatics tools, or build their own using a standard Dockerfile. Through software containers walrus can provide a reproducible execution environment for the pipeline, and containers provide simple execution on a wide range of software and hardware platforms. With initiatives such as BioContainers, researchers can make use of already existing containers without having to re-write their own. Data in each pipeline step is automatically mounted and made available within each Docker container. By simply relying on Docker walrus requires little software setup to run different bioinformatics tools.

While walrus executes a single pipeline on one physical server, it supports both data and tool parallelism, as well as any parallelization strategies within each tool, e.g. multi-threading. To enable data and tool parallelism, e.g. run the same analyses to analyse a set of samples, users list the samples in the pipeline description and walrus will automatically run each sample through the pipeline in parallel. While we can parallelize the independent pipeline steps, the performance of an analysis pipeline relies on each of the independent tools and available compute power. Techniques such as multithreading can improve the performance of a tool, and walrus users can make use of these techniques if their are available through the tools command line interface.

Upon successful completion of a pipeline run, walrus will write a verbose pipeline description file to the output directory. This file contains information on the runtime of each step, which steps were parallelized, and provenance related information to the output data from each step. Users can investigate this file to get a more detailed look on the completed pipeline. In addition to this output file walrus will return a unique version ID for the pipeline run, which later can be used to investigate a previous pipeline run.

[7] graphviz.org

C. Data Management

In `walrus` we provide an interface for users to track their analysis data through a version control system. This allows users to inspect data from previous pipeline runs without having to recompute all the data. `walrus` stores all intermediate and output data in an output directory specified by the user, which is version controlled automatically by `walrus` when new data is produced by the pipeline. We track changes at file granularity.

In `walrus` we interface with `git` to track any output file from the analysis pipeline. When users execute a pipeline, `walrus` will automatically add and commit output data to a git repository using `git-lfs`. Users typically use a single repository per pipeline, but can share the same repository to version multiple pipelines as well. Instead of writing large blobs to a repository, git-lfs writes small pointer files with the hash of the original file, the size of the file, and the version of git-lfs used. The files themselves are stored separately which makes the size of the repository small and manageable with git. The main reason why we chose git and git-lfs for version control is that git is the de facto standard for versioning source code, and we want to include versioning of datasets without altering the typical development workflow.

Since we are working with potentially sensitive datasets `walrus` is targeted at users that use a local compute and storage servers. It is up to users to configure a remote tracker for their repositories, but we provide command-line functionality in `walrus` to run a git-lfs server that can store users' contents. They can use their default remotes, such as Github, for hosting source code but they must themselves provide the remote server to host their data.

D. Pipeline Reconfiguration and Re-execution

Reconfiguring a pipeline is common practice in precision medicine, e.g. to ensure that genomic variants are called with a desired sensitivity and specificity. To reconfigure an existing pipeline users make the applicable changes to the pipeline description and re-run it using `walrus`. `walrus` will then recompute the necessary steps and return a version ID for the newly run pipeline. This ID can be used to compare pipeline runs, the changes made, and optionally restore the data and configuration from a previous run. Reconfiguring the pipeline to use updated tools or reference genomes will alter the pipeline configuration and force `walrus` to recompute the applicable pipeline stages.

The command-line interface of `walrus` provides functionality to restore results from a previous run, as well as printing information about a completed pipeline. To restore a previous pipeline run, users use the `restore` command line flag in `walrus` together with the version ID of the respective pipeline run. `walrus` will interface with git to restore the files to their state at the necessary point in time.

III. Results

To evaluate the usefulness of `walrus` we demonstrate its use in a clinical setting, and the low computational time and storage overhead to support reproducible analyses.

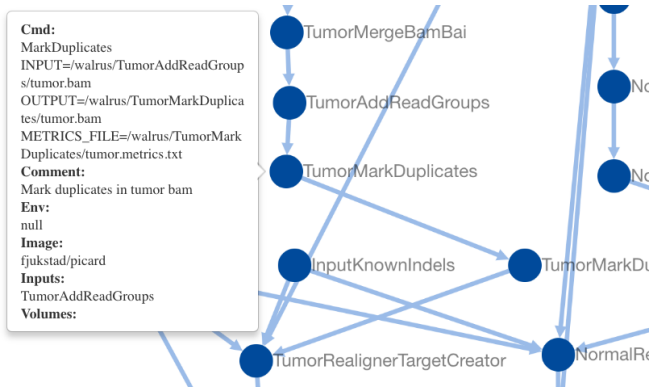


Fig. 1. Screenshot of the web-based visualization in `walrus`. The user has zoomed in to inspect the pipeline step which marks duplicate reads in the tumor sequence data.

A. Clinical Application

We have used `walrus` to analyze a whole-exome data from a sample in the McGill Genome Quebec [MGGQ] dataset (GSE58644)[10] to discover Single Nucleotide Polymorphisms (SNPs), genomic variants and somatic mutations. We interactively developed a pipeline description that follows the best-practices of The Broad Institute[8] and generated reports that summarized the findings to share the results. Figure 1 shows a screenshot from the web-based visualization in `walrus` of the pipeline.

From the analyses we discovered inherited germline mutations that are recognized to be among the top 50 mutations associated with an increased risk of familial breast cancer. We also discovered a germline deletion which has been associated with an increased risk of breast cancer. We also discovered mutations in a specific gene that might explain why specific drug had not been effective in the treatment of the primary tumor. From the profile of the primary tumor we discovered many somatic events (around 30 000) across the whole genome with about 1000 in coding regions, and 500 of these were coding for non-synonymous mutations. We did not see amplification or constituent activation of growth factors like HER2, EGFR or other players in breast cancer. Because of the germline mutation, early recurrence, and lack of DNA events, we suspect that the patient's primary tumor was highly immunogenic. We have also identified several mutations and copy number changes in key driver genes. This includes a mutation in a gene that creates a premature stop codon, truncating one copy of the gene.

While we cannot share the results in details or the sensitive dataset, we have made the pipeline description available at github.com/uit-bdps/walrus along with other example pipelines.

B. Example Dataset

To demonstrate the performance of `walrus` and the ability to track and detect changes in an analysis pipeline, we have implemented one of the variant calling pipelines from [11]

[8] software.broadinstitute.org/gatk/best-practices

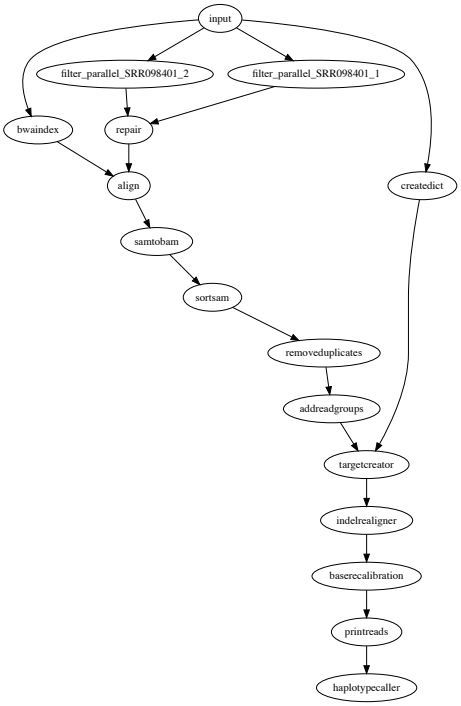


Fig. 2. In addition to the web-based interactive pipeline visualization, `walrus` can also generate DOT representations of pipelines. The figure shows the example variant calling pipeline.

using tools from Picard and the GATK. We show the storage and computational overhead of our approach, and the benefit of capturing the pipeline specification using a pipeline manager rather than a methods section in a paper. The pipeline description and code is available along with `walrus` at github.com/uit-bdps/walrus. Figure 2 shows a simple graphical representation of the pipeline.

1) Performance and Resource Usage: We first run the variant calling pipeline without any additional provenance tracking or storing of output or intermediate datasets. This is to get a baseline performance measurement for how long we expect the pipeline to run. We then run a second experiment to measure the overhead of versioning output and intermediate data. Then we introduce a parameter change in one of the pipeline steps which results in new intermediate and output datasets. Specifically we change the `--maxReadsForRealignment` parameter in the indel realigner step back to its default (See the online pipeline description for more details). This forces `walrus` to recompute the indel realigner step and any subsequent steps. We then use the `restore` flag in `walrus` to illustrate what the parameter change had on the pipeline output. To illustrate how `walrus` can restore old pipeline configurations and results, we restore the pipeline to the initial configuration and results. We show the computational overhead and storage usage of restoring a previous pipeline configuration.

Reproducing results from a scientific publication can be a difficult task. For example, troublesome formatting of the online version of [11] led to some pipeline tools failing. The parameters prefixed with two consecutive hyphens (`--`) are converted to single em dashes (—). PDF versions of the paper lists the parameters correctly. In addition, the input filenames in the variant calling step do not correspond to any output files in previous steps, but because of their similarity to previous output files we assume that this is just a typo. These issues in addition to missing commands for e.g. the filtering step highlights the clear benefit of writing and reporting the analysis pipeline using a tool such as `walrus`.

Table I shows the runtime and storage use of the different experiments. In the second experiment we can see the added overhead of adding version control to the dataset. In total, an hour is added to the runtime and the data size is doubled. The doubling comes from git-lfs hard copying the data into a subdirectory of the `.git` folder in the repository. With git-lfs users can move all datasets to a remote server reducing the local storage requirements. In the third experiment we can see that only the downstream analyses from configuring the indel realignment parameter is executed. It generates 30GB of additional data, but the execution time is limited to the applicable stages. Restoring the pipeline to a previous configuration is almost instantaneous since the data is already available locally and git only has to modify the pointers to the correct files in the `.git` subdirectory.

TABLE I
RUNTIME AND STORAGE USAGE FOR A VARIANT-CALLING PIPELINE DEVELOPED WITH `WALRUS`.

Experiment	Task	Runtime	Storage Use
1	Run pipeline with default configuration	21 hours 50 minutes	235 GB
2	Run the default pipeline with version control of data	23 hours 9 minutes	470 GB
3	Re-run the pipeline with modified indel realignment parameter	13 hours	500 GB
4	Restoring pipeline back to the default configuration	$<$ 1 second	500GB

IV. Related Work

There are a wealth of pipeline specification formats and workflow managers available. Some are targeted at users with programming experience while others provide simple Graphical User Interfaces (GUIs).

We have previously conducted a survey of different specialized bioinformatics pipelines.[12] The pipelines were selected to show how analysis pipelines for different applications use different technologies for configuring, executing and storing intermediate and output data. In the review, we targeted specialized analysis pipelines that support scaling out the pipelines to run on High-Performance Computing (HPC) or cloud computing platforms.

Here we describe general systems for developing data analysis pipelines, not just specialized bioinformatics pipelines. While most provide viable options for genomic analyses, we

have found many of these pipeline systems require complex compute infrastructure beyond the smaller clinical research institutions. We discuss tools that use the common Common Workflow Language (CWL) pipeline specification and systems that provide versioning of data.

CWL is a specification for describing analysis workflows and tools.[13] A pipeline is written as a JSON or YAML file, or a mix of the two, and describes each step in detail, e.g. what tool to run, its input parameters, input data and output data. The pipeline descriptions are text files that can be under version control and shared between projects. There are multiple implementations of CWL workflow platforms, e.g. the reference implementation cwl_runner[13], Arvados[14], Rabix[15], Toil[16], Galaxy[7], and AWE.[17] It is no requirement to run tools within containers, but implementations can support it. There are few of these tools that support versioning of the data. Galaxy is an open web-based platform for reproducible analysis of large high-throughput datasets.[7] It is possible to run Galaxy on local compute clusters, in the cloud, or using the online Galaxy site.[9] In Galaxy users set up an analysis pipeline using a web-based graphical interface, and it is also possible to export or import an existing workflow to an Extensible Markup Language (XML) file.[10] We chose not to use Galaxy because of missing command-line and scripting support, along with little support for running workflows with different configurations.[18] Rabix provides checksums of output data to verify it against the actual output from the pipeline. This is similar to the checksums found in the git-lfs pointer files, but they do not store the original files for later. An interesting project that uses CWL in production is The Cancer Genomics Cloud[19]. They currently support CWL version 1.0 and are planning on integrating Rabix as its CWL executor. Arvados stores the data in a distributed storage system, Keep, that provides both storage and versioning of data. We chose not to use CWL and its implementations because of its relaxed restrictions on having to use containers, its verbose pipeline descriptions, and the complex compute architecture required for some implementations. We are however experimenting with an extension to `walrus` that translates pipeline descriptions written in `walrus` to CWL pipeline descriptions.

Pachyderm is a system for running big data analysis pipelines. It provides complete version control for data and leverages the container ecosystem to provide reproducible data processing.[20] Pachyderm consists of a file system (Pachyderm File System (PFS)) and a processing system (Pachyderm Processing System (PPS)). PFS is a file system with git-like semantics for storing data used in data analysis pipelines. Pachyderm ensures complete analysis reproducibility by providing version control for datasets in addition to the containerized execution environments. Both PFS and PPS is implemented on top of Kubernetes.[21] There are now recent efforts to develop bioinformatics workflows with Pachyderm that show great promise. In [22], the authors show the potential performance improvements of single workflow steps, not the full pipeline, when executing a pipeline in Pachyderm. They unfortunately do not show the time to import data into PFS, run the full pipeline, and optionally investigate different versions of the intermediate, or output datasets.

We believe that the approach in Pachyderm with version controlling datasets and containerizing each pipeline step is, along with walrus, the correct approach to truly reproducible data analysis pipelines. The reason we did not use Kubernetes and Pachyderm was because our compute infrastructure did not support it. In addition, we did not want to use a separate tool, PFS, for data versioning, we wanted to integrate it with our current practice of using git for versioning.

Snakemake is a long-running project for analyzing bioinformatic datasets.[23] It uses a Python-based language to describe pipelines, similar to the familiar Makefile syntax, and can execute these pipelines on local machines, compute clusters or in the cloud. To ensure reproducible workflows, Snakemake integrates with Bioconda to provide the correct versions of the different tools used in the workflows. It integrates with Docker and Singularity containers[24] to provide isolated execution, and in later versions Snakemake allows pipeline execution on a Kubernetes cluster. Because Snakemake did not provide necessary integration with software containers at the time we developing our analysis pipeline, we did not find it to be a viable alternative. For example, support for pipelines consisting of Docker containers pre-installed with bioinformatics tools came a year later than walrus.

Another alternative to develop analysis pipelines is Nextflow.[25] Nextflow uses its own language to describe analysis pipelines and supports execution within Docker and Singularity containers.

As discussed in [26, 12], recent projects propose to use containers for life science research. The BioContainers and Bioboxes[27] projects address the challenge of installing bioinformatics data analysis tools by maintaining a repository of Docker containers for commonly used data analysis tools. Docker containers are shown to have better than, or equal performance as Virtual Machines (VMs), and introduce negligible overhead opposed to executing on bare metal.[28] While Docker containers require a bootstrapping phase before executing any code, this phase is negligible in the compute-intensive precision medicine pipelines that run for several hours. Containers have also been proposed as a solution to improve experiment reproducibility, by ensuring that the data analysis tools are installed with the same responsibilities.[29]

V. Discussion

`walrus` is a general tool for analyzing any type of dataset from different scientific disciplines, not just genomic datasets in bioinformatics. Users specify a workflow using either a YAML or JSON format, and each step in the workflow is run within a Docker container. `walrus` tracks input, intermediate, and output datasets with git to ensure transparency and reproducibility of the analyses. Through these features `walrus` helps to ensure repeatability of the computation analyses of a research project.

Precision medicine requires flexible analysis pipelines that allow researchers to explore different tools and parameters to

[9] Available at usegalaxy.org.

[10] An alpha version of Galaxy with CWL support is available at github.com/common-workflow-language/galaxy.

analyze their data. While there are best practices to develop analysis pipelines for genomic datasets, e.g. to discover genomic variants, there is still no de-facto standard for sharing the detailed descriptions to simplify re-using and reproducing existing work. With `walrus` we provide one alternative to develop and share pipeline descriptions.

Pipelines typically need to be tailored to fit each project and patient, and different patients will typically elicit different molecular patterns that require individual investigation. In our WES analysis pipeline we followed the best practices, and explored different combinations of tools and parameters before we arrived at the final analysis pipeline. For example, we ran several rounds of preprocessing (trimming reads and quality control) before we were sure that the data was ready for analysis. `walrus` allowed us to keep track of different intermediate datasets, along with the pipeline specification, simplifies the task of comparing the results from pipeline tools and input parameters.

`walrus` is a very simple tool to set up and start using. Since we only target users with single large compute nodes, `walrus` can run within a Docker container making Docker its only dependency. Systems such as Nextflow, Galaxy or Pachyderm all require users to set up and manage complex compute infrastructures. The simplicity of `walrus` enables repeatable computational analyses without any of these obstacles, and is one of the strengths of our tool.

Unlike other proposed solutions for executing data analysis pipelines, `walrus` is the only system we have discovered that explicitly uses git, and git-lfs, to store output datasets. Other systems either use a specialized storage system, or ignore data versioning at all. We believe that using a system that bioinformaticians already use for source control management is the simplest way to allow users version their data alongside their analysis code. The alternative of using a new data storage platform that provides data versioning requires extra time and effort for researchers both to learn and integrate in their current workflow.

We have seen that there are other systems to develop, share, and run analysis pipelines in both bioinformatics and other disciplines. Like `walrus`, many of these use textual representations in JSON or other languages to describe the analysis pipeline, and Docker to provide reproducible and isolated execution environments. In `warlus` we provide pipeline descriptions that allows users to reuse the familiar command-line syntax. The only new additional information they have to add is the dependencies between tasks. Systems such as CWL requires that users also describe the input and output data verbosely. We believe that the tool, `walrus`, can detect these, and will handle this for the user. This will in turn make the pipeline descriptions of `walrus` shorter in terms of lines of code.

While systems such as Galaxy provide a graphical user interface, `walrus` requires that its users know how to navigate the command line and use systems such as git and Docker, to analyze a dataset. For some users this may an obstacle, but we believe that it provides a more hands-on and transparent view of the whole data analysis process.

While we provide one approach to version control datasets, there are still some drawbacks. `git-lfs` supports large files, but in our results it added 5% in runtime. This makes the entire analysis pipeline slower, but we argue that having the files under version control outweigh the runtime. In addition, there are only a few public `gif-lfs` hosting platforms for datasets larger than a few gigabytes, making it necessary to host these in-house. In-house hosting may also be a requirement at different medical institutions.

We aim to investigate the performance of running analysis pipelines with `walrus`, and the potential benefit of its built-in data parallelism. While our WES analysis pipeline successfully run steps in parallel for the tumor and adjacent normal tissue, we have not demonstrated the benefit of doing so. This includes benchmarking and analyzing the system requirements for doing precision medicine analyses. We are also planning on exploring parallelism strategies where we can split an input dataset into chromosomes and run some steps in parallel for each chromosome, before merging the data again.

We believe that future data analysis systems for precision medicine will follow the lines of our proposed approach. Software container solutions provide valuable information in the reporting of the analyses, and they impose little performance overhead. Further, the development of container orchestration systems such as Kubernetes is getting wide adoption nowadays, especially in web-scale internet companies. However, the adoption of such systems in a clinical setting depend on support from more tools, and also the addition of new compute infrastructure.

VI. Conclusions

We have designed and implemented `walrus`, a tool for developing reproducible data analysis pipelines for use in precision medicine. Precision medicine requires that analyses are run on hospital compute infrastructures and results are fully reproducible. By packaging analysis tools in software containers, and tracking both intermediate and output data, `walrus` provides the foundation for reproducible data analyses in the clinical setting. We have used `walrus` to analyze a patient's metastatic lesions and adjacent normal tissue to provide insights and recommendations for cancer treatment.

VII. Acknowledgements

We would like to thank Daniel Del Balso for his work implementing the initial WGS analysis pipeline. This work has been funded by The European Research Council (ERC-AdG 232997 TICE), and The Canadian Cancer Society Research Institute (INNOV2-2014-702940).

References

[1] National Research Council et al. *Toward precision medicine: building a knowledge network for biomedical research and a new taxonomy of disease*. National Academies Press, 2011.

[2] Ian F Tannock and John A Hickman. Limits to personalized cancer medicine. *New England Journal of Medicine*, 375(13):1289–1294, 2016.

[3] Yanlei Diao, Abhishek Roy, and Toby Bloom. Building highly-optimized, low-latency pipelines for genomic data analysis. In *CIDR*, 2015.

[4] Nicolas Servant, Julien Roméjon, Pierre Gestraud, Philippe La Rosa, Georges Lucotte, Séverine Lair, Virginie Bernard, Bruno Zeitouni, Fanny Coffin, Gérôme Jules-Clément, et al. Bioinformatics for precision medicine in oncology: principles and application to the shiva clinical trial. *Frontiers in genetics*, 5, 2014.

[5] Andrea Sboner and Olivier Elemento. A primer on precision medicine informatics. *Briefings in bioinformatics*, 17(1):145–153, 2015.

[6] Somak Roy, Christopher Coldren, Arivarasan Karunamurthy, Nefize S Kip, Eric W Klee, Stephen E Lincoln, Annette Leon, Mrudula Pullambhatla, Robyn L Temple-Smolkin, Karl V Voelkerding, et al. Standards and guidelines for validating next-generation sequencing bioinformatics pipelines: A joint recommendation of the association for molecular pathology and the college of american pathologists. *The Journal of Molecular Diagnostics*, 2017.

[7] Jeremy Goecks, Anton Nekrutenko, and James Taylor. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome biology*, 11(8):R86, 2010.

[8] BioContainers. Biocontainers. https://biocontainers.pro, 2017. [Online; Accesssed: 16.08.2017].

[9] Kristian Cibulskis, Michael S Lawrence, Scott L Carter, Andrey Sivachenko, David Jaffe, Carrie Sougnez, Stacey Gabriel, Matthew Meyerson, Eric S Lander, and Gad Getz. Sensitive detection of somatic point mutations in impure and heterogeneous cancer samples. *Nature biotechnology*, 31(3):213–219, 2013.

[10] Ali Tofigh, Matthew Suderman, Eric R Paquet, Julie Livingstone, Nicholas Bertos, Sadiq M Saleh, Hong Zhao, Margarita Souleimanova, Sean Cory, Robert Lesurf, et al. The prognostic ease and difficulty of invasive breast carcinoma. *Cell reports*, 9(1):129–142, 2014.

[11] Adam Cornish and Chittibabu Guda. A comparison of variant calling pipelines using genome in a bottle as a reference. *BioMed research international*, 2015, 2015.

[12] Bjørn Fjukstad and Lars Ailo Bongo. A review of scalable bioinformatics pipelines. *Data Science and Engineering*, 2(3):245–251, 2017.

[13] Peter Amstutz, Michael R. Crusoe, Nebojša Tijanić, Brad Chapman, John Chilton, Michael Heuer, Andrey Kartashov, Dan Leehr, Hervé Ménager, Maya Nedeljkovich, and et al. https://figshare.com/articles/Common_Workflow_Language_draft_3/3115156/2, Jul 2016.

[14] Arvados. Arvados — open source big data processing and bioinformatics. https://arvados.org, 2017. [Online; Accesssed: 16.08.2017].

[15] Gaurav Kaushik, Sinisa Ivkovic, Janko Simonovic, Nebojsa Tijanic, Brandi Davis-Dusenbery, and Deniz Kural. Rabix: an open-source workflow executor supporting recomputability and interoperability of workflow descriptions. In *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, volume 22, page 154. NIH Public Access, 2016.

[16] John Vivian, Arjun Arkal Rao, Frank Austin Nothaft, Christopher Ketchum, Joel Armstrong, Adam Novak, Jacob Pfeil, Jake Narkizian, Alden D Deran, Audrey Musselman-Brown, et al. Toil enables reproducible, open source, big biomedical data analyses. *Nature Biotechnology*, 35(4):314–316, 2017.

[17] Wei Tang, Jared Wilkening, Narayan Desai, Wolfgang Gerlach, Andreas Wilke, and Folker Meyer. A scalable data analysis platform for metagenomics. In *Big Data, 2013 IEEE International Conference on*, pages 21–26. IEEE, 2013.

[18] Ola Spjuth, Erik Bongcam-Rudloff, Guillermo Carrasco Hernández, Lukas Forer, Mario Giovacchini, Roman Valls Guimera, Aleksi Kallio, Eija Korpelainen, Maciej M Kańduła, Milko Krachunov, et al. Experiences with workflows for automating data-intensive bioinformatics. *Biology direct*, 10(1):43, 2015.

[19] Jessica W Lau, Erik Lehnert, Anurag Sethi, Raunaq Malhotra, Gaurav Kaushik, Zeynep Onder, Nick Groves-Kirkby, Aleksandar Mihajlovic, Jack DiGiovanna, Mladen Srdic, et al. The cancer genomics cloud: Collaborative, reproducible, and democratized—a new paradigm in large-scale computational research. *Cancer research*, 77(21):e3–e6, 2017.

[20] Pachyderm. http://pachyderm.io.

[21] Kubernetes. https://kubernetes.io.

[22] Jon Ander Novella, Payam Emami Khoonsari, Stephanie Herman, Daniel Whitenack, Marco Capuccini, Joachim Burman, Kim Kultima, and Ola Spjuth. Container-based bioinformatics with pachyderm. *Bioinformatics*, page bty699, 2018.

[23] Johannes Köster and Sven Rahmann. Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, 28(19):2520–2522, 2012.

[24] Gregory M Kurtzer, Vanessa Sochat, and Michael W Bauer. Singularity: Scientific containers for mobility of compute. *PloS one*, 12(5):e0177459, 2017.

[25] Paolo Di Tommaso, Maria Chatzou, Evan W Floden, Pablo Prieto Barja, Emilio Palumbo, and Cedric Notredame. Nextflow enables reproducible computational workflows. *Nature biotechnology*, 35(4):316, 2017.

[26] Inge Alexander Raknes, Bjørn Fjukstad, and Lars Bongo. nsroot: Minimalist process isolation tool implemented with linux namespaces. *Norsk Informatikkonferanse*, 2017.

[27] Peter Belmann, Johannes Dröge, Andreas Bremges, Alice C McHardy, Alexander Sczyrba, and Michael D Barton. Bioboxes: standardised containers for interchangeable bioinformatics software. *Gigascience*, 4(1):47, 2015.

[28] Paolo Di Tommaso, Emilio Palumbo, Maria Chatzou, Pablo Prieto, Michael L Heuer, and Cedric Notredame. The impact of docker containers on the performance of genomic pipelines. *PeerJ*, 3:e1273, 2015.

[29] Carl Boettiger. An introduction to docker for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1):71–79, 2015.