SOFTWARE

Building applications for interactive data exploration in systems biology

Bjørn Fjukstad[1], Vanessa Dumeaux[2], Karina Standahl Olsen[3], Eiliv Lund[3], Michael Hallett[4] and Lars Ailo Bongo[1*]

Abstract

Background: In scientific fields such as systems biology there is a need for interactive data exploration tools to enable new insights in the fast growing datasets. These tools need to combine advanced statistical analyses, known biology from up-to-date databases, and interactive visualizations. To answer specific research questions tools must provide specialized user interfaces and visualizations. While these are application-specific, the underlying components of a data analysis tool can be shared and reused later. Application developers can therefore compose applications of reusable services rather than implementing a single monolithic application.

Results: We have designed an approach for developing data exploration applications in systems biology that builds on the microservice architecture. We use this approach to build applications that integrate advanced statistical software and up-to-date information from biological databases with modern data visualization libaries. We demonstrate its viability through a web application for exploring and comparing transcriptional profiles from blood and tumor samples, MIxT Blood-Tumor. In addition we have used it to build two other web-applications and several command-line tools. With a microservices approach building on software container technology we can re-use and share key components between application reducing development, deployment and maintenance time.

Conclusions: Our approach and reference implementation Kvik is open-sourced at `github.com/fjukstad/kvik` and the web application for exploring transcriptional profiles, MIxT, is availible at `github.com/fjukstad/mixt`.

Keywords:

Background

In recent years the collection of biological data and curation of biological datasets has been unprecedented. While the cost of collecting data has drastically decreased, data analysis continue to be a larger fraction of the total experiment cost.[1] This calls for novel methods in data analysis and exploration.

To explore the growing number of biological data sets, there are now a number of analysis tools in various programming languages. There are both new methods for analyzing the data as well as presenting them using novel visualization techniques. In the R statistical programming language developers can share software packagesr for exploring high-throughput omics datasets through repositories such as CRAN[1] or Bioconductor[2]. In other languages such as Python or Go, developers can choose bioinformatics libraries such as BioPython[?] and biogo[?] respectively. Such frameworks provide functionality for analyzing data, linking to databases and visualizing the data.

While these frameworks require its users to be proficient at coding, there is a need for applications that allow researchers to explore datasets interactively through simple user interfaces. These applications need to integrate statistical packages, biological databases and interactive visualizations. Unarguably different programming languages are suitable for solving different tasks. E.g. to use R and Bioconductor to analyse biological data, or C++ and OpenCV for optimized high-performance computer vision tasks, or HTML, CSS, and JavaScript to build portable user-interfaces. We argue that to build applications that integrate statistical analyses, interactive visualizations,

*Correspondence: larsab@cs.uit.no
[1]Department of Computer Science, UiT – The Arctic University of Tromsø, 9037 Tromsø, NO
Full list of author information is available at the end of the article

[1]`cran.r-project.org`

[2]`bioconductor.org`

and biological databases it is reasonable to compose the application of several components written in different languages.

A microservice architecture structures an application into small reusable, loosely coupled parts. These communicate via lightweight programming language-agnostic protocols such as HTTP, making it possible to write single applications in multiple different programming languages. This makes it possible to use the most suitable programming language for each specific part. To build a microservice application, developers bundle each service in a container that are deployed. Containers are built from configuration files which describe the operating system, software packages and versions of these. This makes reproducing an application a trivial task. The most popular implementation of a software container is Docker[3], but others such as Rkt[4] exist.

In this paper we describe a novel approach for building data exploration applications in systems biology. We show that by building applications as a set of services it is possible to reuse and share its components between applications. In addition, by packaging the services using container technology we promote reproducible research and simplify application deployment. We have used our approach to build a number of applications, both command-line and web-based. In this paper we describe how we used our approach to develop MIxT, a web application for exploring and comparing transcriptional profiles from blood and tumor samples.

Requirements

From our experience building data exploration applications we have identified a set of reusable services that application developers can use to build a wide range of applications. The key services of a biological data exploration application are i) a compute service for executing statistical analyses in languages such as R, and ii) a database query service for retrieving information from biological databases. On top of these services is possible to build any number applications and these can be reused by different applications.

To build these services we need a framework that fulfills the following requirements:

1 It provides a language-independent approach for integrating, or embedding, statistical software, such as R, directly in interactive data exploration applications.
2 It provides an interface to online databases to provide meta-data to biological entities.
3 Its components should be easy to develop, maintain, deploy and share between projects.

[3] docker.com
[4] coreos.com/rkt

Related Work

In this section we aim to cover some of the existing systems for building interactive data exploration applications in systems biology.

Integrate Statistical Analyses

OpenCPU is a system for embedded scientific computing and reproducible research.[2] It provides an HTTP API to the R programming language to provide an interface with statistical methods. It enables users to make function calls to any R package and retrieve the results in a wide variety of formats such as json or pdf. Users can chose to host their own R server or use public servers, and OpenCPU works in a single-user setting within an R session, or a multi-user setting facilitating multiple parallel requests. This makes OpenCPU suitable for building a service that can run statistical analyses. OpenCPU provides a Javascript library for interfacing with R, as well as Docker containers for easy installation. OpenCPU has been used to build multiple applications.[5]. In Kvik we provide a package to interface with OpenCPU servers from the Go programming language since it provides the same interface to execute and run statistical analyses as we do in our compute service.

Renjin is a JVM-based interpreter for the R programming language.[3] It targets developers who want to integrate the R interpreter in web applications. Since it is built on top of the JVM it allows developers to write data exploration applications that interact directly with R code, both runnin on top of the JVM. Although Renjin supports a large number of CRAN packages it cannot access any R package (i.e. any package from BioConductor or CRAN) without modification. This makes the programming effort to use Renjin as an interface to R higher.

Shiny is a web application framework for R[6] It allows developers to build web applications in R without having to have any knowledge about HTML, CSS or Javascript. It provides a widget library to provide more advanced Javascript visualizations such as Leafle[7] for maps or three.js[8] for WebGL-accellerated graphics. Developers can choose to host their own web server with the user-built Shiny Apps, or host them on public servers. Shiny forces users to implement data exploration applications in R, limiting the functionality to the widgets and libraries in Shiny.

Biogo
this one is a bit out of place.

[5] opencpu.org/apps.html
[6] shiny.rstudio.com
[7] leafletjs.com
[8] threejs.org

Biogo is a bioinformatics library in Go. It provides functionality to analyse genomic and metagenomc datasets in the go programming language.[4] Using the go programming language the developers are able to provide high-performance parallel processing in a clean and simple programming language.

Visualization frameworks

Cytoscape is an open source software platform for visualizing complex networks and integrating these with any type of attribute data[5]. It allows for analysis and visualization in the same platform. Users can add additional features, such as databases connections or new layouts, through Apps. One such app is cyREST which allows external network creation and analysis through a REST API[6]. To bring the visualization and analysis capabilities to the web the creators of Cytoscape have developed Cytoscape.js[9], a Javascript library to create interactive graph visualizations.

Caleydo is a framework for building applications for visualizing and exploring biomolecular data[**?**]. Until 2014 it was a standalone tool that needed to be downloaded, but the Caleydo team are now making the tools web-based. There have been several applications built using Caleydo: StratomeX for exploring stratified heterogeneous datasets for disease subtype analysis[7]; Pathfinder for exploring paths in large multivariate graphs[8]; UpSet to visualize and analyse sets, their intersections and aggregates[9]; Entourage and enRoute to explore and visualize biological pathways [10][11]; LineUp to explore rankings of items based on a set of attributes[12]; and Domino for exploring subsets across multiple tabular datasets[13].

BioJS is an open-source JavaScript framework for biological data visualization.[14] It provides a community-driven online repository with a wide range components for visualizing biological data contributed by the bioinformatics community. BioJS builds on node.js[10] providing both server-side and client-side libraries.

WIP: Biological Databases

Maybe some words here on how to get data out of the different biological databases?

WIP: Microservices, Docker etc.

...

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. Based on our experiences, we then generalize the ideas to a set of principles and services that can be reused and shared between applications.

MIxT

We analyzed profiled RNA in blood and matched tumor from 173 patients in the Norwegian Women and Cancer (NOWAC) study. Each profile measures the expression of 16 782 unique genes. We used Weighted Gene Correlation Network Analysis (WGCNA)[15] to cluster the genes in each tissue based on co-expression. From these clusters of genes, called modules, we investigated their relationship to known biological processes. *More on the methods maybe??*

From the analyses we built an R package[**?**] that implements the different statistical methods developed in the MIxT project. The R package contains functionality for running statistical analyses and generating specialized static visualizations. Based on the results and analyses we wanted to build an application for quickly exploring the results. Since we have a large code base already developed in R we needed a system that can directly interface with the R package. In addition, the interface to R should be accessible through standard protocols, such as HTTP, not enforcing any programming language or platform.

A large part of biological data research is to link the results to known biology from literature or reference databases. In the MIxT project we needed an application that could interface with a set of different databases, keeping the information up-to-date. As with the R interface the interface to the databases should be accessible from any programming language.

Docker ....

Kvik

Based on the development of MIxT, we have generalized our experience into the following design principle guidelines and microservices provided by the Kvik framework:

Principle 1: Build applications as collections of language-agnostic microservices. This makes it possible to re-use key components and build specialized data exploration applications in the most suitable programming language.

Principle 2: Deploy each service using container technology such as Docker. This has a number of benefits. It simplifies deployment itself, it makes it trivial to share services between projects and research groups, and it ensures reproducible services.

Principle 3: Package statistical methods and data as software packages that can be used by power-users

[9] `js.cytoscapejs.org`
[10] `nodejs.org`

and the data exploration tools themselves. An example is to build an application using R packages and OpenCPU or Kvik. This makes it possible to either explore the data and methods through the data exploration application or an R session.

From these principles we developed a set of software packages that provide functionality to build microservices that provides key components to build a data exploration application in systems biology.

WIP: Compute Service

The R microservice and R interface in Kvik is built using a hybrid state pattern[2]. We provide three main operations for interfacing with R: Call, Get and RPC. The Call operation is used to execute and run a function from an R package. It takes as input an R package name, a function name and optional arguments. It returns a unique identifier that later can be used by the Get operation to retrieve results. The Get operation is used to get results in different output formats, e.g. JSON, CSV, PDF, or PNG. The RPC is just a combination of a Call and a subsequent Get.

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. While OpenCPU is implemented on top of R and Apache, Kvik is implemented from the ground up in Go. 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 through the *gopencpu* package.

The R service in Kvik builds on the standard *http* library in Go. On start it launches a user-defined number of R sessions that execute analyses on demand. This allows for parallel execution of analyses. We provide a simple FIFO queue for queuing of requests. The R server also provides the opportunity for users to cache analysis results to speed up subsequent calls.

With this process in mind, we designed the interface to the R programming language in Kvik. We want to make it possible to call any function from an R package and return its results either as plain text, such as comma-separated tables, or binaries such as images. Enforcing that R code is built into R packages ensures that the analysis code can be used by power users through an ordinary R session or in the data exploration application itself.

WIP: Database

We chose to build a database service that interfaces with different online databases to retrieve meta-data on genes and processes. We built our own service so that we could provide caching functionality reducing query time and offload some of the traffic to the various databases.

Similar to how our analysis process shaped the R interface, it also defined how we want to build interfaces to online databases.

Describe the interface to the databases and what we use it for. Could be interesting to talk about provenance/caching.

In its initial state we wanted an interface to interactively query databases such as KEGG or MSigDB for up-to-date information about genes, gene sets or biological pathways. This interface should be available within the data exploration applications to provide valuable metadata, such as gene summaries, for the researchers exploring results.

WIP: Building applications

With Kvik there are multiple ways developers can build data exploration applications. Either bundle analysis and database lookup on a single computer, or separate computational tasks to more powerful compute clusters to improve performance. In this paper we discuss how to develop applications that follow a microservices architecture where data analysis and storage is simply a service.

We do not wish to enforce any style of programming or set of tools to build user-facing apps.

In Kvik we use R packages as the fundamental building block for data exploration applications. They provide an interface to data and analyses, and especially in the field of systems biology, the R programming language provide the largest collection of data analysis packages. We discovered that the most sensible way to build applications on top of our existing code base was to build a system that could interface with our analysis code directly. In Kvik we built an HTTP interface on top of R that allows users to call functions and get results using any programming language with an HTTP library. This allows developers to build data exploration applications in the programming language that is most suitable, or has the best support, for presenting that specific data type.

WIP: Implementation

In ths section we describe the implementation details in Kvik and the microservices required to build the MIxT web application.

Kvik is implemented as a collection of Go packages with the functionality required to build services that can integrate statistical software in a data exploration and provide an interface to up-to-date biological databases. To integrate R we provide two packages *gopencpu* and *r*, that interface with OpenCPU and Kvik R servers respectively. To interface with biological databases we provide the packages *eutils*, *gsea*, *genenames*, and *kegg* that interface with The Entrez Programming Utilities (E-utilities)[11], MSigDB[12], Hugo Gene Nomenclature Committe[13], and Kyoto Encyclopedia of Genes and Genomes (KEGG)[14] respectively. In addition to these packages we provide Docker images that implement the two required microservices.

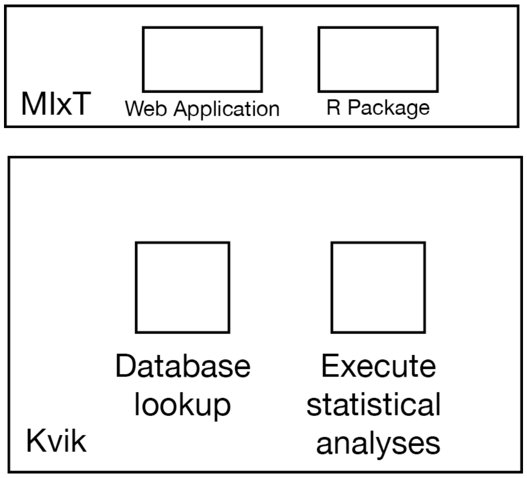


Figure 1 An overview of the relationship between the MIxT application and Kvik. MIxT contains a web application (online at mixt-blood-tumor.bci.mcgill.ca) and the R Package that provides analyses and data to the web application. Kvik provides the services for running the statistical analyses from the R package, and the database lookups found in the web application.

WIP: Results

We show the viability and need for Kvik by describing the MIxT application for exploring and comparing transcriptional profiles from blood and tumor samples. We describe its functionality, implementation and performance requirements.

WIP: Matched Interaction Across Tissues (MIxT)

For the web application we defined six analysis tasks:

Explore co-expression relationships between genes. Create an interactive network visualization that visualizes each gene as a node and significan co-expression relationship as an edge.

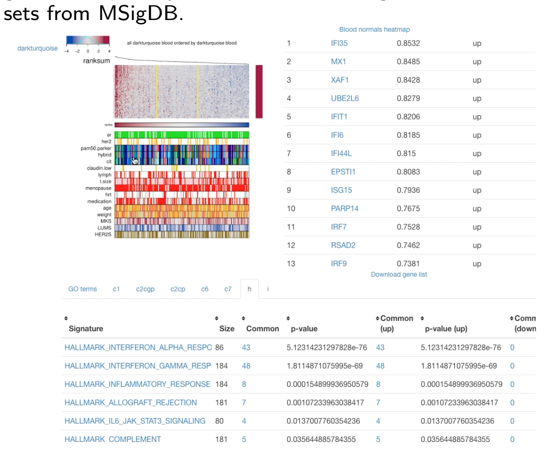
Explore co-expression gene sets in tumor and blood tissue. Visualize gene expression together with clinicopathological variables associated with each module. Include results of gene set analyses that describe the underlying biological functions of the modules.

Explore relationships between modules from each tissue. Visualize how modules from each tissue are related using two different metrics, ranksum and gene overlap. Also enable subtype selection, enabling users to investigate relationships within a particular subtype.

Explore relationships between clinical variables and modules. Visualize significant associations between module expression and clinical variables.

Explore association between user-submitted gene lists and computed modules. Allow users to upload own gene lists and have the application compute modules which the gene list is enriched for.

Search for genes or gene lists of intrest. Allow users to search for specific genes or genelists of intrest and show what modules they are associated with.



1	IFI35	0.8532	up
2	MX1	0.8485	up
3	XAF1	0.8428	up
4	UBE2L6	0.8279	up
5	IFIT1	0.8206	up
6	IFI6	0.8185	up
7	IFI44L	0.815	up
8	EPSTI1	0.8083	up
9	ISG15	0.7936	up
10	PARP14	0.7675	up
11	IRF7	0.7528	up
12	RSAD2	0.7462	up
13	IRF9	0.7381	up

Signature	Size	Common	p-value	Common (up)	p-value (up)	Comn (down
HALLMARK_INTERFERON_ALPHA_RESPC	86	43	5.12314231297828e-76	43	5.12314231297828e-76	0
HALLMARK_INTERFERON_GAMMA_RESP	184	48	1.8114871075995e-69	48	1.8114871075995e-69	0
HALLMARK_INFLAMMATORY_RESPONSE	184	8	0.000154899936950579	8	0.000154899936950579	0
HALLMARK_ALLOGRAFT_REJECTION	181	7	0.00107233963038417	7	0.00107233963038417	0
HALLMARK_IL6_JAK_STAT3_SIGNALING	80	4	0.0137007760354236	4	0.0137007760354236	0
HALLMARK_COMPLEMENT	181	5	0.035644885784355	5	0.035644885784355	0

Figure 2 MIxT module overview page. The screenshot show the user interface for exploring a single module. It consists of three panels. The top left panel contains the gene expression heatmap. 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.

WIP: MIxT design and Implementation

The MIxT application is designed as a modern application consisting of multiple services that together provide an interactive web application. By composing an application of a set of services we can substitute parts of the application without re-writing the entire application. This type of architectural style is called a microservices architecture and is popular in 'web-scale' systems. For example if we want to use OpenCPU to interface with data analysis we can do so by simply exchanging the Kvik R service with OpenCPU. Both services communicate over HTTP and their interface is the same.

From our initial analyses we built an R package with functions to provide data and analysis to the different

[11]eutils.ncbi.nlm.nih.gov
[12]software.broadinstitute.org/gsea/msigdb
[13]genenames.org
[14]kegg.jp

analysis tasks. Using this design it is possible to either explore the data through the web site or a local R session.

To explore the co-expression relationship between genes we use an interactive graph visualization build with Sigmajs[15]. 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. The sigmajs visualization library has functionality for generating a layout for large networks, but we generate this layout server-side to reduce the computational load on the client. To generate this layout we use the GGally package[16].

We have built modules for each tissue, and to explore gene sets associated with genes in these modules, we provide module overview pages that show gene expression visualized together with clinicopathological variables and gene set analyses that describe the underlying functions of the module.

We have used different metrics to link the modules from each tissue, ranksum and gene overlap. To visualize the associations we use the d3[17] library to build an interactive heatmap visualization.

To allow users to explore the relationship between clinical variables and the computed modules, we built an interctive heatmap visualization that visualizes the association between different metrics and each module.

WIP: Future work

Write it.

Security. Documentation. Scale.

WIP: Discussion
WIP: Conclusions

List of abbreviations used

Competing interests

The authors declare that they have no competing interests.

Author details

[1]Department of Computer Science, UiT – The Arctic University of Tromsø, 9037 Tromsø, NO. [2]University of ...., , . [3]Department of Community Medicine, UiT – The Arctic University of Tromsø, 9037 Tromsø, NO. [4]Department of Biology, Concordia University, H4B 1R6 Montréal, CA.

References

1. Sboner, A., Mu, X.J., Greenbaum, D., Auerbach, R.K., Gerstein, M.B.: The real cost of sequencing: higher than you think! Genome biology **12**(8), 125 (2011)
2. Ooms, J.: The opencpu system: Towards a universal interface for scientific computing through separation of concerns. arXiv preprint arXiv:1406.4806 (2014)
3. Bertram, A.: 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, p. 105 (2013)
4. Kortschak, R.D., Adelson, D.L.: bíogo: a simple high-performance bioinformatics toolkit for the go language. bioRxiv (2014). doi:10.1101/005033. http://biorxiv.org/content/early/2014/05/12/005033.full.pdf
5. Shannon, P., Markiel, A., Ozier, O., Baliga, N.S., Wang, J.T., Ramage, D., Amin, N., Schwikowski, B., Ideker, T.: Cytoscape: a software environment for integrated models of biomolecular interaction networks. Genome research **13**(11), 2498–2504 (2003)
6. Ono, K., Muetze, T., Kolishovski, G., Shannon, P., Demchak, B.: Cyrest: Turbocharging cytoscape access for external tools via a restful api. F1000Research **4** (2015)
7. Lex, A., Streit, M., Schulz, H.-J., Partl, C., Schmalstieg, D., Park, P.J., Gehlenborg, N.: Stratomex: Visual analysis of large-scale heterogeneous genomics data for cancer subtype characterization. In: Computer Graphics Forum, vol. 31, pp. 1175–1184 (2012). Wiley Online Library
8. Partl, C., Gratzl, S., Streit, M., Wassermann, A.M., Pfister, H., Schmalstieg, D., Lex, A.: Pathfinder: Visual analysis of paths in graphs. In: Computer Graphics Forum, vol. 35, pp. 71–80 (2016). Wiley Online Library
9. Lex, A., Gehlenborg, N., Strobelt, H., Vuillemot, R., Pfister, H.: Upset: visualization of intersecting sets. IEEE transactions on visualization and computer graphics **20**(12), 1983–1992 (2014)
10. Lex, A., Partl, C., Kalkofen, D., Streit, M., Gratzl, S., Wassermann, A.M., Schmalstieg, D., Pfister, H.: Entourage: Visualizing relationships between biological pathways using contextual subsets. IEEE transactions on visualization and computer graphics **19**(12), 2536–2545 (2013)
11. Partl, C., Lex, A., Streit, M., Kalkofen, D., Kashofer, K., Schmalstieg, D.: enroute: Dynamic path extraction from biological pathway maps for in-depth experimental data analysis. In: Biological Data Visualization (BioVis), 2012 IEEE Symposium On, pp. 107–114 (2012). IEEE
12. Gratzl, S., Lex, A., Gehlenborg, N., Pfister, H., Streit, M.: Lineup: Visual analysis of multi-attribute rankings. IEEE transactions on visualization and computer graphics **19**(12), 2277–2286 (2013)
13. Gratzl, S., Gehlenborg, N., Lex, A., Pfister, H., Streit, M.: Domino: Extracting, comparing, and manipulating subsets across multiple tabular datasets. IEEE transactions on visualization and computer graphics **20**(12), 2023–2032 (2014)
14. Gómez, J., García, L.J., Salazar, G.A., Villaveces, J., Gore, S., García, A., Martín, M.J., Launay, G., Alcántara, R., Ayllón, N.D.T., et al.: Biojs: an open source javascript framework for biological data visualization. Bioinformatics, 100 (2013)
15. Langfelder, P., Horvath, S.: Wgcna: an r package for weighted correlation network analysis. BMC bioinformatics **9**(1), 559 (2008)

[15] `sigmajs.org`
[16] `cran.r-project.org/web/packages/GGally`
[17] `d3js.org`