SOFTWARE

Building applications for interactive data exploration in systems biology

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

Abstract

Background: With the advent of high-throughput genomics, there is a growing need for interactive data exploration tools as the systems biology community collects data at an unprecedented rate. These tools need to combine advanced statistical analyses, relevant prior knowledge, and interactive visualizations in an applicaiton with simple user interfaces. Although data analyses, user interfaces and visualizations are specialized, application developers can share and reuse the underlying components of the application. 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: Our approach for developing data exploration applications in systems biology builds on the microservice architecture. The resulting applications integrate advanced statistical software, up-to-date information from biological databases and modern data visualization libaries. We demonstrate the viability through the MIxT Blood-Tumor web application that explore and compare transcriptional profiles from blood and tumor samples in breast cancer patients. This approach was reused in two other web applications and several command-line tools. Thus our microservice approach building on software container technology enables re-use and sharing of 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`. The web application for exploring transcriptional profiles, MIxT, is available at `mixt-blood-tumor.bci.mcgill.ca` and its source code at `github.com/fjukstad/mixt`.

Keywords:

Background

In recent years the biological community has generated and collected an unprecedented ammount of data. While the cost of data collection 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.

Several tools are now available i various programming languages for data analysis. These include novel statistical/bioinformatics methodologies and graphical analysis tools. In the R statistical programming language developers share software through repositories such as CRAN[1] or Bioconductor[2]. In other languages, libraries for biological bomputation are often availalbe like BioPython[?] and biogo[?] written in Python or Go, respectively.

Although these framework are of tremendous help for scientists with some computing know-how, even computationally savvy researchers can get tangled up when wrestling with software and big data. Novel applications are needed to better integrate statistical packages, biological databases, and interactive visualizations. Different programming languages solve different tasks. For example, new biological data analysis techniques are quickly realeased in R and its package repositories, high performance computer vision tasks

*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`

are performed optimally in C++ and OpenCV, and portable user interfaces more easily built in HTML, CSS and JavaScript. Therefore applications that integrate novel statistical analysis tools, interactive visualizations, and biological databases likely need to include 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 way the most suitable programming language is used for each specific part. To build a microservice application, developers bundle each service in a container. 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. Initiatives such as BioContainers[5] now provide containers preinstalled with different bioinformatics tools.

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. The MIxT web application integrates statistical analysis together with biological databases and interactive visualizations.

Requirements

First, we 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.

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 has an interface to online reference databases to provide meta-data to biological entities.
3. Its components should be easy to develop, maintain, deploy and share between projects.
4. It scales to meet the performance requirements of interactive data exploration applications.

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 offers 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 and OpenCPU has been used to build multiple applications.[6].

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[7] It allows developers to build web applications in R without having to have any knowledge about HTML, CSS or Javascript. Its widget library to provides more advanced Javascript visualizations such as Leaflet[8] for maps or three.js[9] 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.

[3] docker.com
[4] coreos.com/rkt
[5] biocontainers.pro
[6] opencpu.org/apps.html
[7] shiny.rstudio.com
[8] leafletjs.com
[9] threejs.org

Biogo

this one is a bit out of place.

Biogo is a bioinformatics library written in Go. It provides functionality to analyze genomic and metagenomic 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[10], 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[11] 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.

...

[10] `js.cytoscapejs.org`

[11] `nodejs.org`

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. We also enable stratified analyses of the databaset based on breast cancer subtypes. *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 on the application developer.

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.

A key feature that motivated the design of MIxT was that we want to have the flexibility to run the application on any platform. Bundling each component in a software container such as Docker allows us to deploy the application on a wide range of hardware, from local installations to deployments to cloud providers such as Amazon Web Services[12].

[12] `aws.amazon.com`

Kvik

Based on the development of MIxT, we generalized its functionality and underlying systems into a set of design principle guidelines for application developers:

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 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 three main principles we built a set of software packages that provide functionality for microservices that can be used to build a data exploration application in systems biology. We built a compute service for executing statistical analyses and a database service that provides data from biological databases. Using these it is possible to develop specialized data exploration application in any modern programming language.

Building applications

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.

An application starts as one or more R packages with the datasets, analysis code, and optional other utilities for generating static plots in R. Developers can then run the compute service which makes the functions from the R package accessible through an HTTP endpoint. With the data analysis and resulting datasets available, developers can focus on MANGLER NOE HER

It is important to note that since the end application interfaces directly with R, developers can leverage this to produce dynamic visualizations. For example, if an application uses a clustering method to color nodes in a graph, end-users can tweak parameters MANGLER NOE HER

In Kvik we do not specify what programming language or set of tools to use to build an application. We do not believe that there is any one language or framework suitable for every application. We believe that by orchestrating an application as a set of services communicating over standard protocols

Compute Service

The compute service 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.

Database Service

The database service provides an interface to biological databases for retrieving meta-data on genes and processes. In Kvik we have built packages for interfacing with The Entrez Programming Utilities (E-utilities)[13], MSigDB[14], Hugo Gene Nomenclature Committe (HGNC)[15], and Kyoto Encyclopedia of Genes and Genomes (KEGG)[16]

The database service uses a caching mechanism to reduce the load on the online databases. It will also speed up subsequent queries for a cached object, since the query can be served out of cache and not having to be fetched from a remote database. We allow application developers to specify the cache eviction policy, but on default the database service does not evict anything from its cache before the service is restarted. This can be modified by the application developer.

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. We chose the Go programming language because of its performance, ease of development, and simple deployment. 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

[13] eutils.ncbi.nlm.nih.gov

[14] software.broadinstitute.org/gsea/msigdb

[15] genenames.org

[16] kegg.jp

E-utils, MsigDB, HGNC and KEGG respectively. In addition to these packages we provide Docker images that implement the two required microservices.

Both the compute and the databases service in Kvik builds on the standard *http* library in Go. On start the compute service 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 compute service also provides the opportunity for users to cache analysis results to speed up subsequent calls. The database service use the *gocache*[17] package to cache any query to an online database.

Results

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.

Figure 1 shows the relationship between MIxT and Kvik. MIxT consists of a web application and an R package, while Kvik provides the services for doing database lookup of genes and processes, as well as executing statistical analyses.

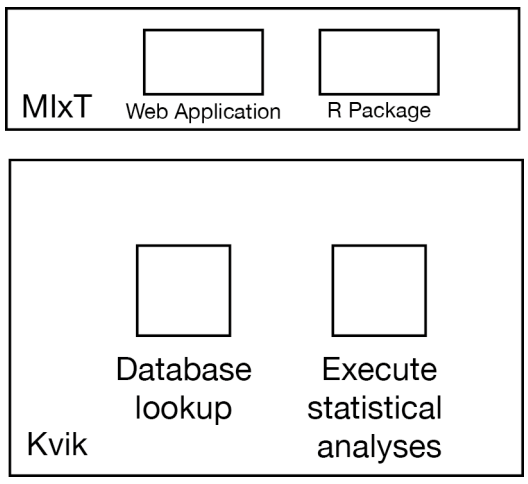


Figure 1 An overview of the relationship between the MIxT application and Kvik. MIxT contains a web application (online at mixt-blood-tumor.bci.mcgill.ca) and the R Package that provides analyses and data to the web application. Kvik provides the services for running the statistical analyses from the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

We define six data exploration tasks that the web application should help users perform:

Explore co-expression gene sets in tumor and blood tissue. We want to simplify the process of exploring the computed co-expression gene sets, or modules, through the web-application. The application should visualize gene expression together with clinicopathological variables associated with each module. In addition we want to enable users to study the underlying biological functions of the modules by including gene set analyses between the module genes and known gene sets.

Explore co-expression relationships between genes. Users should be able to explore the co-expression relationship as a graph visualization. The network should visualize each gene as a node and a significant co-expression relationship as an edge.

Explore relationships between modules from each tissue. Users should be able to explore the relationship between modules from different tissues. We provide two different metrics to compare modules, and the web application should enable users to interactively browse these relationships. In addition to providing visualizations the compare modules from each tissue, users should also be able to explore the relationships, but for different breast cancer subtypes.

Explore relationships between clinical variables and modules. In addition to comparing the association between modules from both tissues, users also have the possibility of exploring the association with a module and a specific clinical variable. This should also be possible for the different breast cancer subtypes.

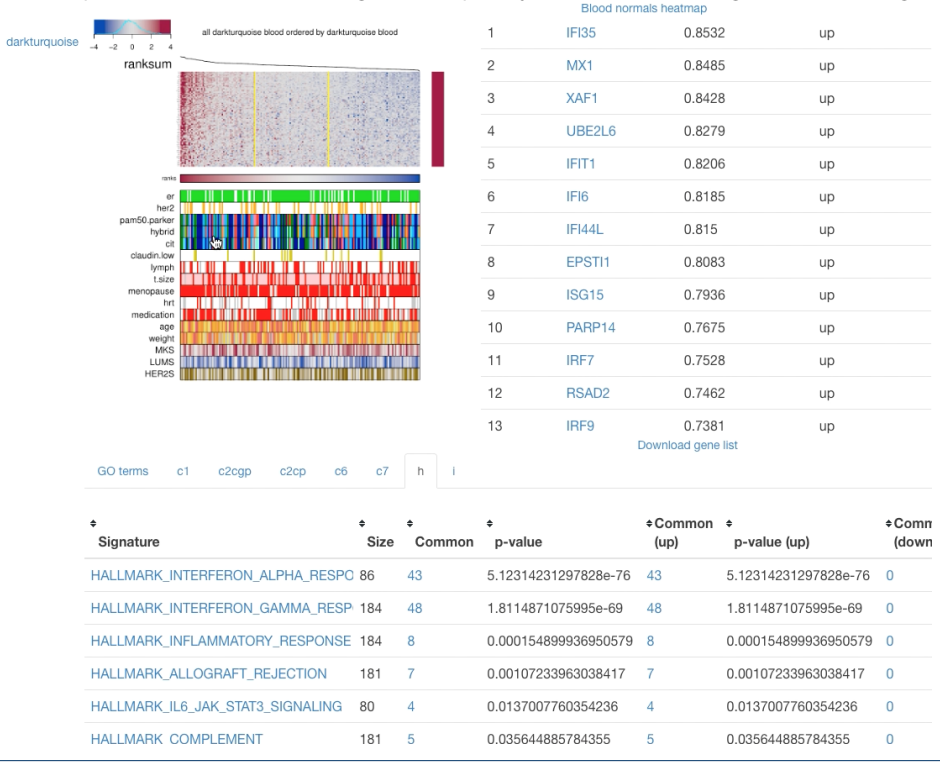
Explore association between user-submitted gene lists and computed modules. We want to enable users to explore their own gene lists to explore them in context of the co-expression gene sets. The web application must handle uploads of gene lists and compute association on demand.

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

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. The MIxT web application consists of three services: i) the web application itself containing the user-interface and visualizations; ii) the compute service performing the MIxT analyses delivering data to the web application; and iii) the database service providing up-to-date information from biological databases. Each of these services run within Docker containers making the process of deploying the application simple. By composing the application of a set of services we can substitute parts of the application without re-writing the entire application. For example if we want to use OpenCPU to interface with data analysis we can do so by simply

[17] github.com/fjukstad/gocache

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.



exchanging the Kvik compute service with OpenCPU. Both services communicate over HTTP and their interfaces are the same.

We structured the MIxT application with a separate view for each analysis task.

To explore the co-expression gene sets we built a view that combines both static visualizations from R together with interactive tables with gene overlap analyses. Figure 2 shows the web page presented to users when they access the co-expression gene set 'darkturquoise' from blood. Using the Kvik compute service we can generate plots on demand and provide users with high-resolution PDFs or PNG files.

To explore the co-expression relationship between genes we use an interactive graph visualization build with Sigmajs[18]. 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[19]. By generating the network layout using the compute service we relieve the clients.

To visualize relationships between modules from different tissues, or their relationship to clinical variables we built a heatmap visualization using the d3[20] library. Figure 3 shows an example of this heatmap visualization, showing the association between the clinical variables and the modules from biopsy for all samples.

WIP: Discussion

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.[21]

WIP: Future work

While we have buit

[18] sigmajs.org

[19] cran.r-project.org/web/packages/GGally

[20] d3js.org

[21] github.com/fjukstad/kvik/tree/master/gopencpu

Figure 3 Heatmap visualization of the association between clinical variables and the modules in biosy. The visualization is built using the d3 JavaScript library.

We have built a database service that 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.

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

Docker compose

This includes both access restrictions to parts of an application, e.g. access to a database, and security concerns regarding remote code execution within the compute service.

to the biological databases that are

Security. Documentation. Scale.

WIP: Conclusions

We have designed an approach for building data exploration applications in systems biology that builds 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.

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]Department of Community Medicine, UiT – The Arctic University of Tromsø, 9037 Tromsø, NO. [3]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)