RESEARCH

Building systems for interactive data exploration in systems biology

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

Abstract

Background: In scientific fields such as systems biology there is a need for data exploration tools that can enable new insights. Such tools need to integrate advanced statistical analyses with known biology from up-to-date databases to leverage the wealth of existing knowledge. However there are few systems that enable the development of new tools that integrate these.

Results: We have designed an approach for developing data exploration applications in systems biology, and demonstrated its viability through a web application for exploring and comparing transcriptional profiles from blood and tumor samples. Our approach makes it possible to explore data from advanced statistical software packages in any modern programming language, together with up-to-date online databases to create applications that integrates data with known biology.

Conclusions: Our approach and reference implementation Kvik, enables easy development of data exploration tools that provide reproducible analyses using efficient processing. 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 the past decade the generation of biological datasets has been unprecedented, and the famous "$1000k genome, $1M analysis"[] has become more apparent. To decrease both time and cost of analysing biological data, there is now a growing number of analysis framework in various programming languages.[]

In R, there are popular package repositories such as CRAN cran.r-project.org or Bioconductor bioconductor.org where developers can share software packages and keep them up-to-date. In these repositories researchers can find software for exploring high-throughput genomic data in one environment. This includes both pre-processing, e.g. cleaning, removing liers, and analysing it with known statistical methods.

Interpreting the analysis results require integration of known biology, either from biological databases such as MSigDB[] or KEGG[], or through scientific publications from databases such as PubMed[]. Performing manual lookup into these databases is often tedious and error-prone, making it necessary to automate the task.

Especially large datasets in biology require sophisticated methods for visualizing and interpreting the results, as well as communicating and sharing the findings.

Requirements

1. A language-independent approach for integrating statistical software, such as R, directly in interactive data exploration applications.
2. Up-to-date interfaces to online databases providing meta-data for understanding results from statistical analyses.

Contributions

Our contributions are:

1. An approach for developing data exploration applications in systems biology that combine statistical analyses with online databases.
2. A demonstration of its viability through N different applications.
3. Performance evaluation of its central data analysis component.

Related Work

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

*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

OpenCPU
OpenCPU is a system for embedded scientific computing and reproducible research.[**?**] It provides an HTTP API to the R programming language, which can be used from any modern programming language. Users can chose to host their own R servers or use public servers. 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.[1]

Renjin
Renjin is a JVM-based interpreter for the R programming language.[**?**] 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. Although Renjin supports a large number of CRAN packages it cannot access any R package (e.g. from BioConductor) without modification. This makes it less

Shiny
Shiny is a web application framework for R.[**?**] 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 Leaflet for maps or threejs for WebGL-accellerated graphics. Developers can choose to host their own web server for

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

Methods

Data analysis

In this section we describe our typical approach for doing analysing gene expression data in systems biology, and how it shaped the design of Kvik's R interface.

We typically start off with a messy dataset that needs to go through several stages of clean-up and preprocessing before we can analyze it. After the preprocessing we typically develop some simple visualizations that help discover simple patterns in the data. After this quick dirty data exploration we start to apply more advanced statistical methods to look for more intricate patterns in the data. After this analysis we often end up with genes or lists of genes of interest that we can use to guide database lookup.

In terms of data analysis code, the preprocessing steps typically consist of one or more R scripts that we knit [**?**] into PDF reports that we can revisit later. From these scripts we end up with analysis-ready datasets that can be shared within the group. The remaining downstream analysis often starts out in scripts, that are built into R packages with analysis code that can be shared between researchers.

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.

Databases

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

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 for the researchers exploring results.

Building applications

With Kvik there are multiple aveues developers can take to 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 multitier architectures.

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.

[1] opencpu.org/apps.html

Statistical analyses

Describe how we've designed the interface with R: Build an R-package and call functions from it, we provide four different output formats to the user (json, csv, pdf, png), as well as four different http endpoints (call, get and rpc).

The R interface in Kvik follows many of the design patterns in OpenCPU. Both systems interface with R packages in a stateless pattern using an HTTP interface. 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.

Databases

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

Implementation

Applications

Results and Discussion

Describe the MIxT application. Also talk about how our R interface scales and what makes it better than opencpu/renji.

Motivating example

We motivate the 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. Then we describe how MIxT is designed to separate concerns and allow for a layered implementation. We use this to motivate the need and opportunities to abstract away common functionality of these type of applications.

Matched Interaction Across Tissues (MIxT)

We have built a system to identify genes and pathways in the primary tumor that are tightly linked to genes and pathways in the patient systemic response[**?**]. MIxT blood-tumor is an open-source web application for exploring the molecular processes expressed in each tissue.

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.

Design and Implementation

The MIxT application is designed as a multitier web application that consists of a compute backend and a webserver frontend. From our initial analyses we built an R package with functions to provide data to perform the different analysis tasks. Using the

Conclusions

List of abbreviations used

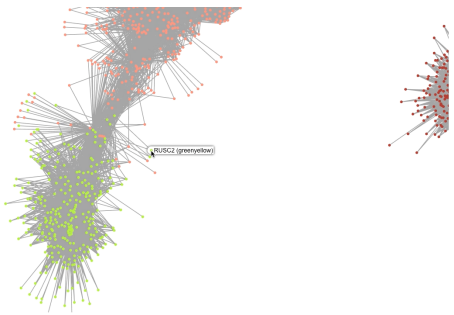
Competing interests

The authors declare that they have no competing interests.

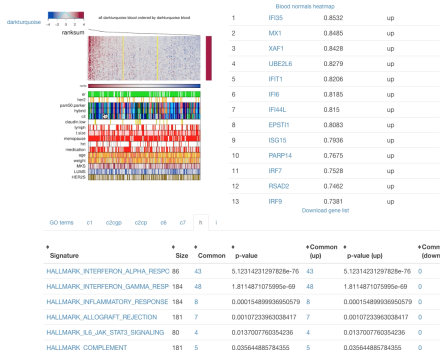
Author details

[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.

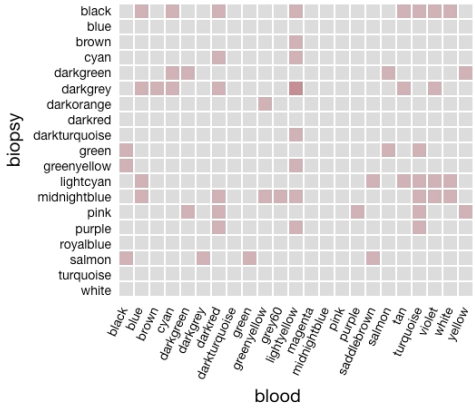
References



(a) A1: Network visualization of gene co-expression



(b) A2: Module visualization.



(c) A3: Visualization of ranksum.

(d) A4: Visualization of significant clinical variable association

Figure 1: Screenshots of the user interface in four of the six analysis tasks. Note that we combine visualization frameworks for both JavaScript and R to generate the visualizations. Specifically, sigmajs (a, sigmajs.org), ggplot2(b, ggplot2.org), and D3 (c and d, d3js.org)