# ABSTRACT


| | |
|---|---|
| Title of Dissertation: | Epiviz: INTEGRATIVE VISUAL ANALYSIS SOFTWARE FOR GENOMICS |
| | Florin Chelaru, Doctor of Philosophy, 2015 |
| Directed By: | Professor Héctor Corrada Bravo<br>Department of Computer Science |



Computational and visual data analysis for genomics has traditionally involved a combination of tools and resources, of which the most ubiquitous consist of genome browsers, focused mainly on integrative visualization of large numbers of big datasets, and computational environments, focused on data modeling of a small number of moderately sized datasets. Workflows that involve the integration and exploration of multiple heterogeneous data sources, small and large, public and user specific have been poorly addressed by these tools. Commonly, the data visualized in these tools is the output of analyses performed in powerful computing environments like R/Bioconductor or Python. Two essential aspects of data analysis are usually treated as distinct, in spite of being part of the same exploratory process: algorithmic analysis and interactive visualization. In current technologies these are not integrated within one tool, but rather, one precedes the other. Recent technological advances in web-based data visualization have made it possible for

interactive visualization tools to tightly integrate with powerful algorithmic tools, without being restricted to one such tool in particular.

We introduce Epiviz (http://epiviz.cbcb.umd.edu), an integrative visualization tool that bridges the gap between the two types of tools, simplifying genomic data analysis workflows. Epiviz is the first genomics interactive visualization tool to provide tight-knit integration with computational and statistical modeling and data analysis. We discuss three ways in which Epiviz advances the field of genomic data analysis: 1) it brings code to interactive visualizations at various different levels; 2) takes the first steps in the direction of collaborative data analysis by incorporating user plugins from source control providers, as well as by allowing analysis states to be shared among the scientific community; 3) combines established analysis features that have never before been available simultaneously in a visualization tool for genomics.

Epiviz can be used in multiple branches of genomics data analysis for various types of datasets, of which we detail two: *functional genomics* data, aligned to a continuous coordinate such as the genome, and *metagenomics*, organized according to volatile hierarchical coordinate spaces.

We also present security implications of the current design, performance benchmarks, a series of limitations and future research steps.

# Epiviz: INTEGRATIVE VISUAL ANALYSIS SOFTWARE FOR GENOMICS

By

Florin Chelaru

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park,
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
2015

Advisory Committee:

Professor Héctor Corrada Bravo / Advisor, Chair
Professor Mihai Pop
Professor Amitabh Varshney
Professor Adam Porter
Professor Niklas Elmqvist / Dean's Representative

I dedicate my dissertation work to my brother, Cristian Chelaru,
and my parents, Carmen and George Chelaru.
They have always been my team, my absolute role models in life, my sources of inspiration, love and integrity.

# Acknowledgements

This work would not have been possible without the guidance, support and friendship of a number of people to whom I owe my deepest gratitude.

I want to thank my advisor, Professor Héctor Corrada Bravo, who has been not only a true mentor, but also a wonderful colleague and friend. His guidance, enthusiasm and energy have constituted a fuel for my imagination and a source of motivation when things didn't seem possible.

I would also like to thank Professor Mihai Pop, without whose faith in me and in my skills I might have never worked in this field in the first place. Without his support and feedback this work would not have been possible.

My other committee members, Professor Amitabh Varshney, Professor Niklas Elmqvist and Professor Adam Porter deserve special acknowledgement. They helped me understand the challenges of the problem I set up to solve, and provided support and encouragement.

None would have been possible without my college advisor, Professor Liviu Ciortuz, who set me on this path and never stopped encouraging me throughout the years.

My colleague, Joseph Paulson has my true gratitude for the help, feedback and friendship throughout my work: *dude, you're the best!*

To my other colleagues at CBCB, I also owe thanks. In particular, I want to mention Joyce Hsiao, Kwame Okrah and Mahfuza Sharmin whose friendship made the work environment a wonderful and fun place to go to.

I would also like to thank my friends, both in the US and back in Romania: Susanna Johnson, Alexandru Ariton, Mădălina and Măriuca Morariu, Umesh Logandran and Ethan Ertel. You guys were there for me when I needed you, with love, friendship, advice, uncountable pep talks and occasionally, butt kicks.

My deepest thanks go to my family – my brother, Cristian Chelaru, and my parents, Carmen and George Chelaru. Their constant support and sacrifices have made me the man I am today and anything I have ever accomplished is owed to them. *Vă iubesc, dragilor!*

# Table of contents

# List of tables

# List of figures

# 1 Introduction

In this work we present Epiviz [1]–[3] (http://epiviz.cbcb.umd.edu, **Figure 1**; a short overview video is available at http://youtu.be/099c4wUxozA) an interactive visualization tool for genomics data statistical analysis.

Visual and computational exploration of experimental data from multiple integrated sources is essential to the analysis of high-throughput genomics experimental results. Such analyses often involve gathering data from different sources – third-party experiments, online databases, etc. Unfortunately, the interactive visualization and computational parts are usually done as distinct and disjoint steps in data analysis, because of the lack of tools that treat the analysis as an iterative process that alternates between the two, rather than a static one. For example, genome and epigenome browsers are ubiquitous visualization tools [4]–[14] where the data visualized is commonly the output of algorithmic and statistical analyses performed in a computing environment such as *R/Bioconductor* [15], or *Python*. Yet, there are no effective tools to tightly couple interactive visualization with such environments. Advances in web application frameworks, like AJAX and JQuery, and visualization frameworks, like data-driven documents (d3.js [16]), facilitate development of interactive data visualization tools that are easily deployed through the web [16]–[19]. These tools have gradually moved from interactive visualization of fixed data elements to the integration of algorithmic and analytic capabilities [19], thereby accelerating how insights are derived from data. A number of genome and epigenome browsers have adopted modern web-application technologies to provide more efficient visualization (for example, by implementing

client-side rendering, caching and pre-computation). In addition, these browsers provide a better user interface experience, including zooming and panning [5]–[14]. However, because none of them yet support integration with algorithmic analysis platforms, this limits visualization to presentation and dissemination rather than hybrid analysis where interactive visualization and algorithmic modeling are integrated together. In addition, all of those tools are restricted to the visualization of functional genomics data, usually aligned to a specific genome. Because of this, their visualization and data exploration capabilities cannot be extended to other kinds of data, or different branches of genomics.

Epiviz is an extensible web tool that features a set of interactive and collaborative visualization methods and for the first time in the field, a system that provides open-ended coupling with computational and statistical modeling and data analysis environments. Through Epiviz, we introduce a series of approaches to creating a system that integrates multiple sources of large complex data and visualizing them interactively using different perspectives. We also provide an insight into its design choices. The tool blends together genomic visualizations, computational and statistical analyses, and drastically reduces the time used by researchers trying to put together data from different sources.

The dissertation is structured as follows: in **Chapter 2** we present the current state of the art in the field, specifically research and related attempts at designing visualization tools for genomics data. In **Chapters 3** and **4** we introduce our work, methods used, test results, security aspects and limitations of our approaches.

Finally, **Chapter 5** presents a series of case studies that illustrate essential use cases where Epiviz innovates the field of genomics data visual and statistical analysis.


**Figure 1. Screenshot of Epiviz.** It shows the main elements available within the tool. Visualizations show data within the selected genomic region. 1) the *main toolbar*; 2) a *scatter plot*, showing two *computed measurements*: the mean and difference of colon gene expression for normal and cancer tissues [80]; the code of the plot is customized to show a line at $y$=0; 3) a *heatmap*, showing values from the *Gene Expression Barcode* comparing the normal and cancer expressions for different tissues; 4) a stacked plot, showing two columns for normal and cancer gene expression; it uses the *color by* transformation, to highlight genes with various expression differences; 5) a custom track defined in a plugin hosted on GitHub Gist, showing blocks aligned to the genome, with height corresponding to the expression of the genes; 6) a *stacked track*, showing a *computed measurement* corresponding to the difference between normal and cancer methylation; 7) a *lines track*, showing DNA methylation for normal and cancer colon tissues; the track uses the *group by* transformation to aggregate three normal samples and three tumor samples, and displays error bars to show the variation of methylation for each group at each data point; it uses *basis* (*B-spline*) interpolation to smoothly connect the available data points; 8) a *genes track*, showing human genome genes fetched from the *UCSC database* using a *data provider plugin* stored externally on GitHub Gist; 9) a tooltip showing details on demand for the gene MMP1. The highlighted items correspond to the brushing feature, triggered while hovering over the MMP1 gene in the genes track.

http://epiviz.cbcb.umd.edu/?gist[]=160e8b84795603961b9f&gist[]=5a88f39caa801e58b8ae&ws=GJU2bfURaUd

## 1.1 Motivation

Social media, advertising, economy, biology are witnessing a shift from data generation and gathering to data analysis, whether it is through algorithmic/statistical means, like *machine learning* and *artificial intelligence*, or visualization. These fields are now facing an avalanche of data that awaits organizing and interpretation. In high-throughput genomics in particular, rapid advances in sequencing technologies and their use in large consortium projects like *ENCODE* [20], *1000 Genomes Project* [21], and the *Human Epigenome Roadmap* [22], among others, hold promise for biomedical scientists to posit and test hypotheses on complex mechanisms of development and disease by integrating massive publicly available data as context for their own experimental data. While many existing software tools aid in the integration or visualization of these datasets, there is still a critical need for tools that allow truly interactive, flexible, effective and powerful visual computational exploratory analyses of data. Recent advances in techniques for the exploration of large visual spaces can allow analysts to effectively navigate through a large number of big datasets. These advances are yet to be applied for genomics data, hampering the effectiveness of existing analysis tools.

Many standard analyses in functional genomics, including transcriptome analysis (via microarray and RNAseq [4]), analysis of histone modifications or transcription factor binding using ChIPseq [23], and comprehensive microarray [24] or sequencing assays to profile DNA methylation (DNAm) [9], [25]–[30] employ powerful computational and statistical analysis tools to preprocess and model data at each step of statistical inferences. Visualization of data at each step of these pipelines is

essential for exploratory analysis of the data, characterizing the behavior of the pipeline, and making sense of the biological context of results by comparing to other datasets and genomic features. Interactive and exploratory visualization along with flexible and uniform data integration would make these steps far more efficient.

Integration of data from multiple disparate sources continues to be a significant challenge scientists are faced with. Data analysts often revert to computational platforms commonly used in their fields, like R or Python, which offer quick access to data widely used in their scientific community. In genomics, the R/Bioconductor [15] framework is a state-of-the-art platform for implementation and dissemination of computational and statistical analysis methods for genomics data. Some of the services it is exposed to, like *AnnotationHub* [31], provide uniform access to a large number of data sources to support data integration by providing infrastructure that supports a large variety of data types based on community standards (**Table 1**). Development of interactive and integrative visualization tools based frameworks such as R/Bioconductor is a strong motivation for epiviz, which through them, is assured of high visibility and impact in the scientific community. Tools that directly benefit from our system include a number of frequently used, state-of-the-art methods for a) ChIPseq (DiffBind [32]), where iterative visualization of data and results of peak-calling algorithms is necessary; b) RNAseq analyses using DESeq [33], [34], edgeR [35], [36], or limma [37], [38], and Cufflinks/Cuffdiff [39] through the cummeRbund package, where both location-based coverage and feature-based expression levels are required; c) methylation analyses using BSmooth [40] or minfi [41], where location-based analysis at multiple genomic scales is important.

| **Object type** | **Data Standard** | **Bioconductor Interface** | **Bioconductor Class** |
|---|---|---|---|
| **Sequence Data** | FASTA file | *FaFile* | *DNAStringSet* |
| **Genomic Intervals** | BED/bigBED file | *BEDFile* | *GenomicRanges* |
| **Genomic quantitative data** | WIG/bigWIG file | *BigWigFile* | *RleList* |
| **ShortRead Alignments** | SAM/BAM file | *BamFile* | *GenomicAlignments* |
| **Genomic Variants** | VCF file | *VCF* | *VRanges* |

**Table 1. Partial list of Bioconductor infrastructure data types.** Through this set of datatypes, *AnnotationHub* and Epivizr provide integration of a variety of data through uniform data types, based on community standards, regardless of data source.

The lack of support by existing tools for interactive and integrative visual data analysis using well-curated data sources in the scientific community constitutes the main motivation for the work presented here.

The design of epiviz responds to the need of integrating computational and visual interactive and exploratory analysis for genomics data. Existing tools usually treat these steps: 1) computational and statistical analysis, and 2) interactive integrative visualization, as distinct (**Table 2**), while they are more effective when used iteratively.

| Analysis Step | Common Tasks at this Step | Epiviz Support |
|---|---|---|
| **Algorithmic Analysis (Data Modeling)** | - simple data transformation<br><br><br><br>- searching<br><br>- normalization, aggregation, smoothing, filtering, sorting, mapping, overlapping, dimensionality reduction, feature extraction | - through *computed measurements*<br>- through *UI code data transformations*<br>- through the *gene search* feature<br>- in R/Bioconductor via Epivizr [31], [118], [119], [120], [121]<br>- in Python via Epivizpy[122], [123], [124] |
| **Interactive Visualization (Exploration, Validation and Presentation)** | - overview (low res), detailed view (high res), navigate, side-by-side view for comparison of multiple types of data, different views for the same data in different dimensions<br>- jump between regions of interest<br>- linking between data sets using metadata or genomic location<br>- use visualizations for presentation (paper figures, etc.) | - multi-view, pan/zoom<br><br><br><br>- Epivizr's *slideshow* feature<br><br>- brushing<br>- *save as static image* feature<br>- workspace sharing |

**Table 2. Partial list of genomic data analysis tasks.**

*Motivation by example*

Consider the case of finding differentially methylated regions (DMRs) associated with cancer. Statistical inferences from smoothing methods built on base-pair level measurements of DNA methylation (DNAm) [40], [41] are used to carry out this task. In a general sense, these methods use a statistical model of expected methylation $m_{ljk}$ at genomic locus $l$ for replicate $j$ belonging to class $k$ (say, normal or tumor): $m_{ljk} = f(l) + g_k(l) + e_{ljk}$ where $f(l)$ is a smooth function over genomic loci and $g_k(l)$ is another smooth function that measures deviation in methylation for class $k$, so that contiguous regions where $|g_k(l)|$ is large enough are considered differentially methylated and a level of significance can be determined by parametric or non-parametric statistical methods. This stipulates a chain of measurements and estimates that lead to the final list of regions found: the base-pair level DNAm measured for each sample, the smooth functions used in the above model, and the

regions where methylation differences are determined to be significantly different (**Figure 2**).

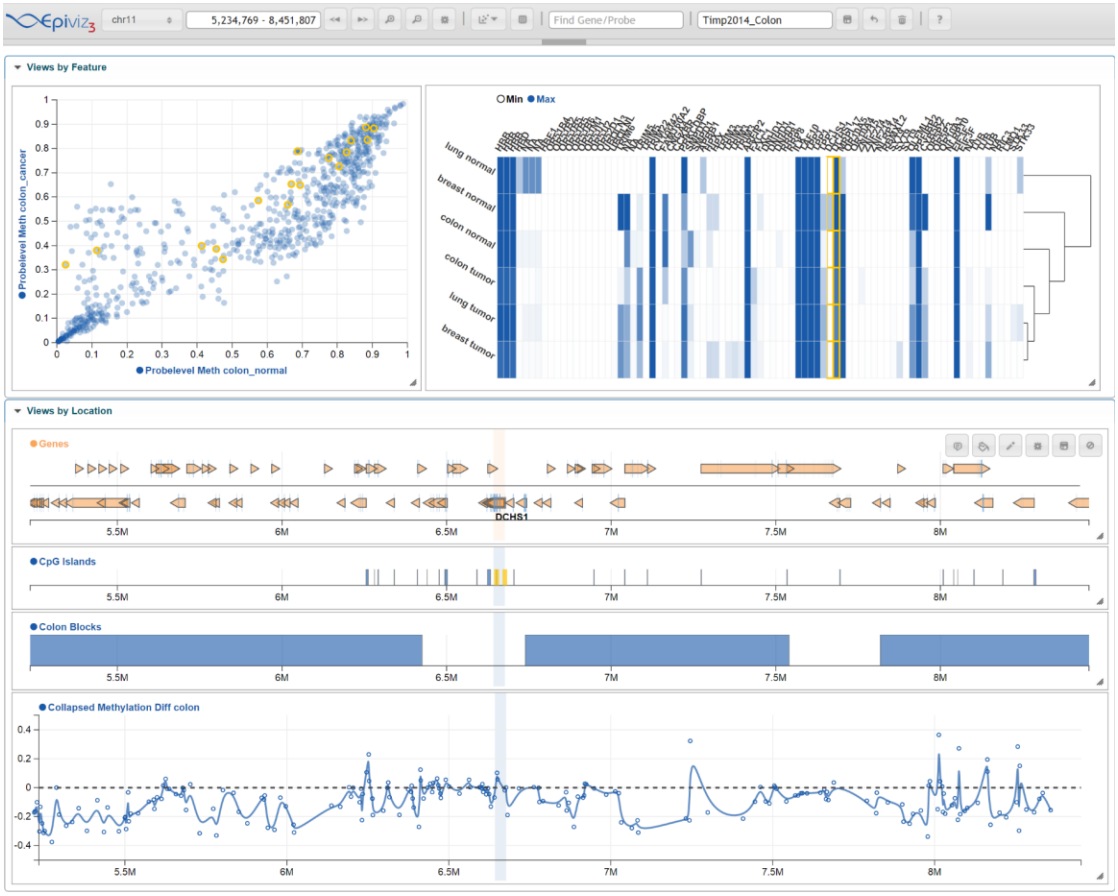


**Figure 2. Statistical analysis of colon cancer methylome.** Top-left displays CpG-level methylation measurements for colon normal and tumor tissue for a sample of TCGA [60] data; top-right displays expression data from the Gene Expression Barcode Project across multiple cancer types as a *heatmap*, the hierarchical clustering is dynamically updated as the user navigates across the genome; the bottom display shows smooth function $\boldsymbol{g_k(l)}$ , described in the main text, corresponding to differences in methylation between normal and tumor; the *colon blocks* track displays statistically significant DMRs inferred from the smoothed methylation difference function. The brushing interaction (in yellow) links data from Bioconductor objects produced at different stages of a statistical analysis pipeline: from measurement preprocessing to statistically significant regions of interest allowing effective exploration of the statistical properties of these genomic findings. http://epiviz.cbcb.umd.edu/?ws=sp9ShCJdS3c

Note that these smooth functions are parameterized by bandwidth to adjust their smoothness, which in turn determines the size of DMRs found: e.g., large smoothing windows leading to longer DMRs detected. For data derived from Illumina

HumanMethylation 450k beadarrays this methodology is implemented in the minfi [41] Bioconductor package.

Once these regions are determined, they would be exported, as BED file for example, and visualized on a genome browser to integrate with other annotations (for example, the location of CpG islands and genes), providing context and interpretation based on the relationship between estimated DMRs and other genome annotations. Unfortunately, at this point most information pertinent to the statistical analysis driving these inferences is lost (for example, the choice of smoothing parameters that determine the width of DMRs and therefore their overlap with other genomic features). Furthermore, interaction in the genome browser is not informed by statistical properties of the inferred regions (for example, navigating through regions in order of statistical significance).

Current tools make it difficult to simultaneously perform *statistically informed* visualization by allowing exploratory analysis of measurements and estimates across this multi-stage statistical procedures and *visualization informed* statistical analysis where parameters of these multi-stage procedures are explored based on integrative visualization.

**Figure 2** shows how the design of Epiviz is based on supporting this type of workflow. Analysts can simultaneously visualize data across multiple stages of statistical analyses (at CpG level, the smooth functions in statistical model, and inferred DMRs) directly from data, through the Epivizr Bioconductor package. Changes in parameters of this statistical pipeline would be reflected immediately in the visualization. Analysts can also integrate annotation data (gene and CpG island

location and expression data from the Gene Expression Barcode [42] project for multiple tissues). Throughout this dissertation, we describe in detail the design of our tool to support this type of analysis workflow.

## 1.2 Contributions

Our contributions can be summarized as:

1. Introducing the first genomics software that brings code to interactive visualization. Epiviz does this in three ways: a) by allowing computational environments such as R or Python to use it as an interactive display device; b) by allowing scripts stored externally (i.e. on source control providers) to be dynamically integrated into the framework; these can be custom visualizations or data providers, but can also extend or change the code base of the framework, for example by altering default settings; c) by providing support for code customizations over existing visualizations, as well as data transformations straight into the UI.
2. Introducing the concept of community-contributed plugins for web applications through JavaScript dynamic extension. Epiviz is the first web-based visualization tool whose code base can be extended by actively incorporating third-party scripts. The Caja [43] sanitizing library that Epiviz uses, guarantees the integrity of user data by executing this code in sandbox mode. Combined with the *workspaces* feature, which allows users to persistently save, replicate and share analysis steps – including code customizations done in the UI – this opens the

door to social collaboration within the genomic community, which has never been done before.

3. Creating the first genomics visualization tool that makes use of the following set of concepts and features *simultaneously*: brushing and linking, binning, supporting data transformations in the UI, predictive caching based on navigation, aggregating data from multiple sources – both cloud-based and local – and persistently saving and sharing data analysis steps as *workspaces*.
4. Applying and extending information visualization concepts to both *functional genomics*, and *metagenomics*, thus becoming the first visualization tool in the field whose scope spans across multiple branches of genomics.

Epiviz is the first system to provide tight integration between a state-of-the-art analytics platform and powerful integrative genomics visualization system. The system is designed around a powerful and flexible extension system where new data types can be defined in both the JavaScript Epiviz framework and its Bioconductor and Python interfaces. New visualizations for existing data types can be plugged in as d3.js visualizations. The *n-tier* architecture of Epiviz allows easy integration of external data streams by using asynchronous data requests thus de-centralizing data storage. Dissemination of visual analyses is easy through Epiviz as permanent URLs are provided which replicate both the underlying data and the visualization components of shared workspaces.

# 2 Background

## 2.1 Existing genomics visualization tools

A number of existing tools approach some of the capabilities targeted by Epiviz, but fall short. For example, the Human Epigenome Browser [44] provides powerful visualization and data integration capability, but is not directly integrated with a computational environment. This implies that users are limited in how they manipulate and transform data during exploratory analyses to features provided by the tool itself. Our project, on the other hand, features a *data provider* plugin system that permits tight coupling with fully capable computational frameworks; the two supported computational environments, R and Python allow integrating data over sets of uniform data types. In R, these are defined by the Bioconductor [15] infrastructure, which provides seamless transitioning between visual and computational analyses. Our design goal is to make all data capable of both being visualized as well as modeled. The AnnotationHub resource provided by Bioconductor also features integration of curated data sources in a uniform manner supporting interactive exploratory workflows not offered by the Human Epigenome Browser.

Galaxy [45]–[47] is another tool that provides excellent support for pipeline workflows, but providing a limited set of visualization options. Our project targets workflows that allow exploratory interaction with data, allowing users to quickly generate hypotheses and derive insights from both experimental and annotation data. In addition, users of Epiviz have the option to create new visualizations that fit their data analysis needs, and immediately use them alongside the existing ones,

automatically getting benefits like brushing or being able to save particular views as vectorial static images. Also, Epiviz employs visualizations that are feature and genomic location-oriented, as well as ones that target data structures directly. Users can customize the code of visualizations directly in the UI, or create new ones that can be immediately used.

A related tool, cBio Portal [48] allows querying and visualizing data in a uniform manner, but is targeted just towards cancer genomics. Epiviz provides a similar framework which works for generalized workflows and an extensible querying and visualization infrastructure that allows the creation of similar tools easier. In addition, our visualization framework also incorporates a number of engineering features like *predictive caching*, to improve smooth interactivity and performance that users and developers can exploit to create web-based data applications for large genomics data.

Current tools for the analysis of large genomics datasets usually target one of two audiences: either programmers and data analysts, who interact with data mostly through code and scripting, or biomedical scientists, who usually interact with data through graphical user interfaces (GUIs). Conversely, there has been a significant push from educational and scientific institutions, including the NIH, to increase the computational literacy of biomedical scientists in preparation for work in a data-intensive field like high-throughput genomics [49], [50]. By continuing to target graphical user interfaces as the *only* way in which biomedical scientists can interact with data, current tools are not in tune with this trend. As a core design component

of Epiviz, we added *code* as an exploration interface to genomics data by providing tight coupling of visualization and computational environments.

## 2.2 Visualization systems with similar functionality

Harger et al. [51] and Zhang et al. [52] provide two comprehensive surveys and comparisons of the state-of-the-art visual analytics open-source toolkits and commercial systems respectively. The comparisons take into account a wide variety of features, from visualizations offered, to analysis functionality and portability. Based on these studies, the Titan [53] toolkit and the system Spotfire [54] stand out through their extensive combination of *visualization* and *statistical analysis* functionality. All other tools analyzed in these surveys lack a feature which we consider essential for genomic data analysis workflows: *multivariate statistical analysis* – the statistical analysis of data in three or more dimensions (for example, *dimensionality reduction*, *clustering*, etc.).

*Titan*

Titan is not in itself a visualization tool, but rather a toolkit providing an environment for data visualization and analysis. Unlike most other desktop-based visualization toolkits, it is OS-flexible, offers a good set of multiple coordinated interactive visualizations, as well as a computational facet for C++, Python and TCL. The architecture is also extensible by allowing users to create custom plugins for data transformations.

There are several reasons why we chose not to use this toolkit, in the creation of our software, and looked towards web-based visualization with an open-ended

computational environment connection. The first has to do with community standards: one of the goals of our efforts in developing Epiviz was to provide support for frequently used state-of-the-art methods for genomic data analysis. This is currently best addressed by R/Bioconductor, through an extensive suite of libraries and packages that are able to both manipulate different types of genomic data (**Table 2**), and facilitate workflows for a number of established domain-specific analysis methods, as outlined in **Section 2.3**. The Titan toolkit is bound to C++, offering limited ways in which a connection to the R/Bioconductor infrastructure could be established. The second motivation for opting for the web-based architecture has to do with light-weight of user extension and interaction. Customizing the code of Epiviz is an easy task partially because of our choice of JavaScript as the main programming environment for the framework. With Titan, in order to create a plugin, one needs to instantiate and build the entire framework on the local machine. In contrast, the current design of Epiviz requires no download, being able to interpret and execute third-party code from cloud-based locations such as GitHub. Our option also implicitly solves a security concern having to do with the fact that JavaScript is given limited access to the user file system, which is ensured by web browser vendors. Desktop-based software on the other hand, is vulnerable to a wide variety of security vulnerabilities which require special attention.

*Spotfire*

Spotfire is a commercial desktop-based data analysis and visualization tool which also comes with a wide variety of analytic features, like brushing and linking, multivariate statistical functions, as well as a feature which allows users to define custom data transformations by writing IronPython code directly into the UI. As powerful as it is, the tool comes with a number of limitations which make it difficult to use for genomic data analysis workflows. The most evident is the fact that the tool is OS-dependent, being able to run solely on Windows. This is a serious drawback for the genomics community, where more than half of the users work on either Macintosh or Linux; in **Figure 3** we show a statistic of the operating systems used in Epiviz sessions since its launch at the end of 2014, revealing that approximately 53% of all users logged in from either Macintosh or Linux, while only about 42% belonged to Windows users. But Spotfire has a few other drawbacks as well, among which the fact that it does not natively support common genomic data types, and also, it relies on loading the entire data in memory, making it unfeasible to use for extremely large datasets.

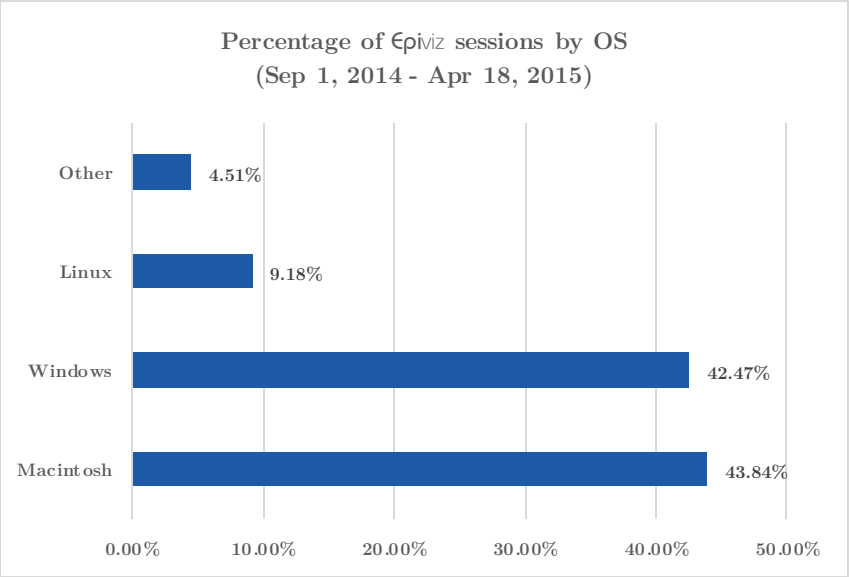


**Figure 3. Percentage of Epiviz sessions by OS since the time of its release, measured using Google Analytics.**

## 2.3 Considerations about genomic data

Data used by bioinformaticians in data analysis is complex in its variety, heterogeneity, and size. Sometimes, this data consists of records mapped to a genome – for example, DNA methylation, or read coverage, which are commonly displayed in genome browsers [6], [11], [55], [56]. In other cases, the main coordinate is replaced by time or some other user-defined dimension. (For more examples, see **Sections 3.4**, **3.5** and **5.5**.)

The size of genomics data also tends to vary greatly. Usually, a single dataset corresponding to the genomic data for a single sample will range from a few hundred records to a few hundred million [57], all of which can easily fit into the memory of a modern day computer. However, in the workflow of a single data analysis task, a scientist may involve a large enough number of datasets for it not to be feasible to load all data in the memory of a workstation at once, or even store it locally. For

this reason, there now exist several large databases aggregating and making available various types of genomic datasets for a multitude of use cases [42], [58], [59]. There has been a trend in the field of storing efficiently and providing large selections of these datasets to bioinformaticians [60]. Often these sit behind genome browsers, which offer, to some extent, help with the dissemination of subsets of these datasets. However, in the past, no tool has been able to interactively visualize data from a multitude of these different sources simultaneously, and at the same time facilitate dynamic statistical analysis of new datasets derived from user experiments, residing locally.

In our work we recognize the need to analyze, model, navigate over, correlate and interactively visualize data that is both varied in terms of format and semantics, as well as source and size. Current data analysis workflows of individual researchers as well as research groups usually involve the usage of a multitude of tools as part of the same process, including a number of genome browsers and statistical environments such as R/Bioconductor or Python. To reduce the time spent by users switching between tools in the course of one data analysis process, genome browsers take one of two paths. The first is to duplicate data already available in other databases [61]–[63]; this approach is unfeasible when taking into account the rate at which new data becomes available [59], [64], the sizes of some such databases [42], [65], as well as the efforts aimed at making these databases efficient in terms of query time and data format [60]. The other path is to serve data directly from the cloud [56]. We find the latter preferable, given the data considerations made earlier. But, in addition to genome browsers, data analysts often revert to computational

platforms, for quick access to data widely used in their scientific community as well as quick manipulation and complex modelling of small datasets. For genomics data, the R/Bioconductor framework is a state-of-the-art platform for implementation and dissemination of computational and statistical analysis methods.

However, before Epiviz, no existing tool by itself was able to accomplish both of the following: 1) integrate simultaneously data from different genome browsers without replicating it on the local database, and 2) bridge the gap between genome browsers and computational environments, by coupling interactive visualization with statistical data modelling. Genome browsers sit in front of large databases and are highly interactive, but have little to no data transformation capability. Computational environments on the other hand, are powerful data modelling tools, but do not expose interactive visualizations, nor are they capable of easily manipulating extremely large datasets, being limited to the physical memory of the local machine.

One of the purposes of Epiviz is to make different datasets in various locations and formats easy to access and manipulate in the same tool. Our system is able to visualize datasets directly from their existing locations, as well as custom user data residing on the local machine, side by side. Not only this, but the users are now able to model and change their custom data in a computing environment of their choice and immediately visually explore these changes, all in the same view, without the need to switch between tools or to constantly upload new versions of transformed data to a web server. Epiviz introduces a framework that features a data provider API which can be used to integrate both data available through online services, as

well as data loaded in the memory of a computing environment. In addition, users of Epiviz gain the benefits of new data sources and formats available only to services within these environments [31], [66], [67]. These services provide uniform access to a large number of data sources to support data integration by providing infrastructure that supports a large variety of data types based on community standards (**Table 1**).

## 2.4 Challenges

There are several challenges associated with designing a system like Epiviz, which facilitates workflows for statistical analysis, exploration and dissemination of genomic data. Of all, we consider the most important the following:

1. Standardizing the data format such that various types of data from different sources, domains and with different semantics can be represented uniformly in memory and displayed graphically.
2. Making the software extensible on two levels: a) visualization – so users can easily customize the look and feel of existing visualizations, or create and immediately be able to use new ones for their experimental data, and b) data management – so data can be plugged in from custom sources, whether those are publicly available databases or local computational environments.
3. Coupling computational environments as back-ends to the user interface in a manner that permits two-way communication without imposing a restriction to one computational environment in particular.

To address the first challenge, Epiviz introduces a data standard described in **Section 3.6** that takes into account most known genomic data types. Having a generic unique data format is essential to being able to represent different measurements in the same visualization, or the same measurement in different visualizations. It is also one of the key architecture features at the base of the Epiviz plugin mechanisms, both at the visualization and data provider levels.

Having addressed the first challenge simplifies the second significantly. Epiviz can incorporate third-party code dynamically, as illustrated in **Chapter 3**. This is coupled with an API that exposes utility classes for both visualization and data retrieval to facilitate the dynamic creation and customization of charts and data providers alike.

To address the third challenge, we created a *data provider* (**Section 3.5**) on top of the data management API, which acts as a proxy between Epiviz and computational environments. The computational environment opens a WebSockets server that listens to requests, while the proxy creates a client that connects to the server. Through this mechanism, Epiviz is able to communicate with virtually any programming environment, as long as it implements the server part of the protocol.

In our work we provide two such plugins, one for R and one for Python. The server component is implemented in corresponding packages for each environment – Epivizr [2] and Epivizpy [3] respectively.

# 3 Methods

Epiviz is a web-based visualization software that prototypes a new concept of integrative, interactive and exploratory analysis of genomic data. It tightly couples modern visualization technologies with computational platforms. It also offers multiple visualization methods for location-based (e.g. genomic regions of interest) and feature-based (e.g. exon or transcript-level expression) data using fundamental, well-established, interactive data visualization techniques, of which some are not available in most web-based genome browsers. For example, since visualization elements are implemented using the SVG standard, data objects are mapped directly to vectorial display objects. This allows Epiviz to expose user operations directly on objects within visualizations – for example brushing, which instantly gives users visual insights of the spatial correlation of multiple datasets. All data-displaying containers are resizable, colors can be mapped dynamically to display objects, and charts can be exported as static vectorial image files.

We also introduce a data abstraction model, used throughout the Epiviz framework for storing virtually any type of genomic data, as well as for the communication protocol with *data providers*. Standardizing the data format is the key to our software's flexibility in terms of supported data types as well as being able to visualize any of the data sets using any of the available visualizations. Furthermore, in conjunction with an open-ended plugin system, it permits treating different computational back-ends the same as far as user interaction goes.

## 3.1 Framework architecture

Epiviz is a JavaScript JQuery HTML5 *n-tier* web application. Its architecture is organized following a three-tier deployment pattern: *presentation*, *visualization* and *data management*, whose interaction is event-based, through a controller [68] (**Figure 4**). Tiers are designed to be independent, which facilitates software extension through plugins without the need of re-implementing or modifying parts of the tool. The independence of these tiers is essential to our tool's ability to represent the same data using different visualizations, and reusing the same visualization for different types of data integrated from different sources. Also, this architecture is created with the idea of allowing future extension or entire replacement of any of the tiers independently, without affecting the functionality of the other two, thus making a promise for easy evolution in future versions of the platform.

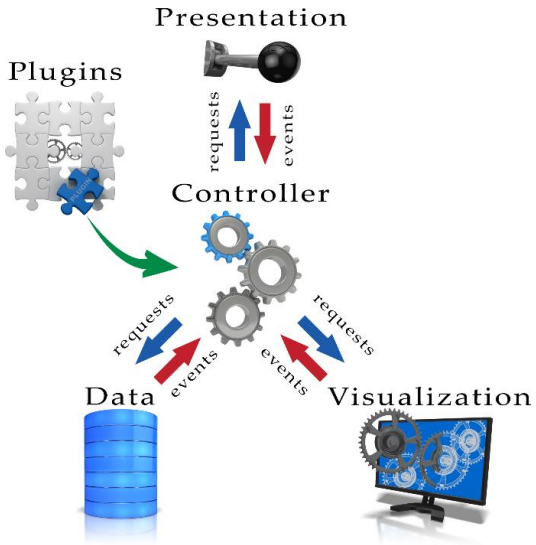


**Figure 4. Epiviz architecture.** The three tiers, *presentation*, *visualization* and *data management* are independent, communicating to each other through a *controller*. The controller sends direct messages to and listens to events triggered by each of the tiers to propagate them further to the either tiers. The controller also processes plugins which extend the visualization and data tiers.

Functional genomics data can be referenced in two distinct domains: by *genomic location*, and by *genomic feature*. In the former, data is organized and referenced by location: for example, regions of interest (ROI) defined by start and end genomic locations, or continuous measurements like DNAm obtained at base-pair resolution. Otherwise, data is organized and referenced by feature: for example, expression for a gene, transcript or exon. Organizing and referencing data in multiple domains requires support for different types of data visualizations. This is a central aspect of the Epiviz design. Data in the location domain is displayed in tracks where visualization appropriate for spatial data is used while data in the feature domain is displayed in charts where other visualization approaches, like scatterplots, heatmaps, etc., are more appropriate.

The data tier uses JQuery, AJAX and WebSockets to merge and processes genomic data in its two generic formats (feature-oriented and location-oriented), sent to Epiviz simultaneously from a number of data providers, such as PHP web servers or interactive R/Bioconductor sessions through the Epivizr package described below. For an improved user experience, this tier implements a predictive caching strategy to accelerate system response to user-initiated data requests (**Section 3.8**).

Organizing and referencing data in multiple domains requires support for different types of data visualizations. Therefore, the visualizations tier presents an extensible framework with a number of predefined visualizations as well as a mechanism for users to define and plug in custom visualizations on-the-fly. The use of JQuery and d3.js in this tier facilitates the implementation of fundamental, well-established, interactive data visualization techniques [69]–[71], not available in classic

web-based genome browsers. For example, since display objects are mapped directly to data elements in the data-driven document framework of d3.js, we implemented a *brushing* feature that instantly gives users visual insights of the spatial correlation of multiple datasets. Also, all data-displaying containers are resizable, colors can be chosen dynamically for display objects, and charts can be exported as static image files (PDF, SVG, PNG or PostScript) all due to the fact that d3.js elements are implemented using the SVG standard.

The Epiviz *presentation tier* provides standard genome browsing functionality along with other features to facilitate discovery and sharing of insights. One example is the ability to define new data measurements based on existing measurements loaded in the data tier using a simple expression language (*computed measurements* – **Section 3.5**). Another example is persistent storage of *workspaces* including genomic location information along with both data measurements and UI elements displayed in a session (**Section 3.7**). These allow users to easily share visual analyses for presentation or as part of a collaborative project.

The *n-tier* design used in Epiviz allows logical separation of *datatypes* – measurements, such as genomic regions of interest, continuous measurements along the genome, and feature-based measurements such as exon, transcript or gene-level expression – from their visualization. These are represented as distinct object classes in the JavaScript Epiviz framework, which allows for an extension mechanism in which visualizations can be implemented independently from data types. This logical separation stresses the fact that multiple views of the same data can be created simultaneously in Epiviz.

We have also integrated data from the Gene Expression Barcode project [42] allowing immediate access to the complete human transcriptome while visual analysis and data explorations are performed. Like all other UI components in Epiviz, brushing is accessible in barcode data displays providing immediate visual cues on transcriptome state with respect to other genomic features derived from new data (**Figure 1**).

The *visualization* and the *data management* tiers belong to an API that facilitates their dynamic extension. One way in which users can dynamically extend these tiers is by plugging in external scripts referred through their URLs or GitHub Gist ids (**Section 3.5**). The two main base classes that can be derived and extended are *Visualization*, for new tracks and plots, and *DataProvider*, for new data sources. This design encourages the scientific community to actively contribute to the Epiviz framework without the bottleneck of a production pipeline.

*Discussion*

In the *design* stage of the development of our tool, we were faced with the question of which development platform to choose to base its implementation, and whether it should be either *desktop*-based or *web*-based. In this section we discuss the motivation behind our choice of the HTML5/JavaScript combination as the development environment for our framework.

1. Portability. One of the great features of HTML5/JavaScript is that they are interpreted languages, OS-independent. All major web browsers support and

interpret these successfully, although with minor differences in behavior and rendering, which we needed to address to provide a uniform user experience.

2. Lightweight. HTML5/JavaScript do not require any additional installation to standard tools used in genomic data analysis workflows. A web browser is sufficient to run an instance of Epiviz on any local machine. Users that already analyze their data using R/Bioconductor or Python can immediately use our framework to interactively visualize it.
3. Advanced features for web-based operations. One of the goals of our tool is to be able to efficiently retrieve cloud-based data sources. JavaScript is optimized for such operations making it easy to develop modules on top of this existing functionality. This also allowed us to implement features through which scripts for entire visualizations or data providers can be fetched directly from their source control cloud-based locations (GitHub Gist).
4. Dynamically interpreted language. The fact that JavaScript is dynamically interpreted opened a door into the type of features that allow users to plug in custom code either directly into the UI or fetch it from source control providers and immediately execute it within our framework.
5. Security. Web browser vendors have taken great efforts to ensure the security of JavaScript code, through limiting its access to the file system. Desktop-based tools that attempt to provide the same level of extensibility through user-defined plugins need to dedicate a lot of effort to the hard problem of ensuring that custom code does not have malicious content. In our case, the only sensitive information accessible through JavaScript plugins was strictly connected to our

tool – user session and workspace information. We were able to address this concern by sanitizing the user script before execution (**Section 3.9**).

6. Richness of visualization resources. As the web-based visualization community increases, so do resources corresponding to this field. In our work we made use of the SVG vector rendering format, which is naturally supported in web browsers. Libraries such as D3.js, JQuery, as well as the WebSocket protocol used to communicate to computational environment sessions facilitated the quick and seamless development of our tool, and account for the richness of visualization features.

The main drawbacks of choosing HTML5/JavaScript as the environment for Epiviz have to do with performance limitations: 1) single thread of execution, 2) limited available memory. In **Section 6.1** we discuss these limitations in detail, as well as future research directions aimed at addressing them.

## 3.2 Bringing code to interactive visualizations

Coupling between computational environments and visualizations has been done before, to some extent, either in areas of computer science unrelated to genomics, or for different audiences than those targeted by Epiviz. Here we briefly describe two categories of tools that attempt to do this, underlining aspects that we considered worth replicating for our purposes, and ideas we decided to take a step further. Combined with the interactive features of a genome browser, these make Epiviz the first tool of its kind.

*Web-based computational environments*

One category of tools that has gained a lot of traction in the past years is represented by web-based computational environments, such as IPython Notebook [72] and RCloud [73]. They are much similar to regular computational environments, but in addition, they combine code execution in browser, plots, and rich media. What we find particularly useful about these tools is the ability of users to create custom notebooks, and share them with each other. This is done either via individual files that can be viewed using an online notebook viewer, or using cloud-based source control providers such as GitHub. This collaborative approach has been proven extremely effective for the audiences of these tools, consisting mainly of programmers and data analysts.

But neither of these tools can substitute a genome browser, because they are not specialized on interactive visual exploration of genomic datasets. They feature a set of static charts, the same as regular computational environments, but operations specific to genome browsers, such as navigation, or brushing, to link data across different charts are not naturally supported.

*Shiny*

Shiny [74] is an interactive web application framework for R, which features a set of predefined web widgets which can be used to visualize custom sets of data loaded in memory. What is especially useful about this tool is a feature called *reactivity*, which consists of binding web controls to R functionality that responds to user actions on the web interface. This opens the door for R to visualizations that

are more interactive than conventional static plots. This, along with the fact that its native environment is R, makes the tool extremely popular within the computational genomics community.

But although it is a great addition to the R computing environment, Shiny cannot be used to replace genome browsers in a data analysis workflow. It is not built to link data across charts using brushing, nor does it support navigation over subsets of the same data. In addition, custom visualizations and visual optimizations cannot be built directly in R – knowledge of JavaScript and visual libraries built on top of it is required to create advanced complex views. But the main reason Shiny is not suitable for genomics data analysis is that it works under the assumption that all resources are loaded in the memory of the computing environment, even when only a subset of those resources is needed by the user. This makes the tool impractical for workflows where large datasets need to be analyzed in an iterative fashion. It also makes it hard to put together subsets of data from a multitude of large databases.

### Epiviz

Epiviz brings code to visualizations in ways similar to those described in the previous sections.

First of all, it features a WebSockets API similar to that used by Shiny, which allows communication between the web framework and any environment that implements the corresponding endpoint protocol. This API is used by the Epivizr and Epivizpy packages described below. In this regard, the way in which Epiviz advances the field of genomic data visualization is that it extends the power of

computational environments to browsing/navigation, allowing the entire data analysis workflow to take place in the same tool. Using libraries like Epivizr or Epivizpy, Epiviz users can now use R or Python and all data modelling functionalities available in these environments to process their experiment results, and interactively visualize each step of their analyses alongside datasets from various online sources, with all the capabilities of a genome browser.

Secondly, Epiviz exposes a plugin API for both data providers and visualizations, so that new data sources can be easily added as needed, and new visualizations can be defined to display the same data from different perspectives. Custom JavaScript code for new visualizations or data providers, can be plugged in on-the-fly using GitHub Gists, similar to the way IPython and RCloud incorporate custom user notebooks. But the way the technology we introduce in Epiviz is entirely novel, even compared to these tools, is that it not only allows users to share and run custom code, but the same API also provides a way in which the code base of the framework itself can be both changed and extended, which has never been done before. This eliminates a limitation shared by all previous genomic visualization tools, and even most scientific visualization tools in general, which consists of restricting the functionality of the tool to a set of predefined views and widgets, with no easy extension capability.

Finally, a set of code features provides a gateway into the parts of JavaScript code that matters for the effective transformation of visualizations and data depicted in them. These features come in two flavors: 1) the first allows the customization of the code corresponding to *individual instances* of visualizations, while 2) the second

consists of a set of data transformation functions such as user-defined filters, coloring, grouping and ordering. Each visualization features a *code button*; when clicked, a dialog pops up with multiple tabs, each corresponding to one of these features, showing an embedded JavaScript code editor. Users can edit and modify the code for any subset of these features, which immediately takes effect. What is particularly useful to this set of features is that they all affect just the visualization instance, and not its entire class. This means that an analysis can contain any number of variations over the same visualization, all starting from the same code base, and yet, all different. In addition, all custom code created by the user can be saved and shared as part of a persistent state *workspace* (**Section 3.7**) transforming it into a social resource that can be shared and reused among the scientific community.

## 3.3 Epivizr

The design of the Epiviz web application was centered on providing powerful interactivity by its tight-knit integration with computational environments. This is implemented in the Epivizr Bioconductor package that uses WebSockets connections between the Epiviz browser client and interactive R sessions to support two-way communication. Data loaded dynamically in R objects is served in response to requests made by Epiviz. This is one of the tools built on top of the *data provider* API.

All data sources catalogued by the AnnotationHub [31] Bioconductor resource are available for integration as measurements via Epivizr: the UCSC genome browser database [61], Ensembl [59], and BioMart [66] are all catalogued by AnnotationHub.

It currently serves 10,780 datasets for 309 different species. 2,238 of these are genome sequences, 302 are variant sets (VCF files) and 7,677 of these are datasets defined over genomic regions. It consists of a web service hosted using Amazon Web Services (AWS) where each dataset (or resource) is stored as a serialized R object of one of the data types defined by the Bioconductor infrastructure as a file on Amazon's S3 storage platform. As part of its curated structure, AnnotationHub uses a versioning system where each dataset is versioned and the service itself is versioned into snapshots. This allows the creation of reproducible integrative analyses since users can request that data integrated is of a specific version.

Infrastructure from the core Bioconductor team and hundreds of contributed packages are used in a large number of projects analyzing data that ranges from expression microarrays to next-generation sequencing. Due to Epivizr, users of epiviz immediately benefit from the fundamental data structures exposed by Bioconductor in their analyses. Conversely, developers of new methods in R and Bioconductor have now access to an interactive way of visualizing data at each step of development.

Epivizr provides support for interactive visualization of fundamental data types in Bioconductor including GenomicRanges that encapsulate data about genomic regions, and SummarizedExperiment objects that encapsulate quantitative measurements over genomic regions from a set of samples [75]. Many software packages in Bioconductor extend these fundamental classes [25], [33], [76], so Epivizr provides interactive visualization support directly for packages that extend these data types. For instance, by providing support for these objects we allow users to

use existing Bioconductor infrastructure[77] to integrate data in SAM or BAM files in their visualizations (**Table 1**).

The package also features updating, filtering and subsetting operations on R objects that trigger updates in their corresponding visualizations in Epiviz. One of its most important capabilities is that it supports interactive exploratory browsing by, for example, allowing users to navigate in order through a set of genomic regions defined in R, using a feature called *slideshow*. Thereby, users can rank regions of interest according to some predefined, or computed attribute. A canonical example is navigation through regions of differentially expressed genes from an RNAseq experiment obtained from packages like DESeq or EdgeR.

*Discussion*

The *Bioconductor* framework is a state-of-the-art platform for the implementation and dissemination of computational and statistical analysis methods for functional genomics data. Infrastructure from the core Bioconductor team and hundreds of contributed packages are used in a large number of projects analyzing data that ranges from expression microarrays to next-generation sequencing. The development of interactive visualization tools based on the Bioconductor infrastructure immediately supports a number of widely used, state-of-the-art methods for ChIPseq, RNAseq and methylation analyses. By supporting interactive visualization of fundamental data structures provided by Bioconductor, developers of new methods using this framework can immediately benefit from powerful, extensible interactive visualization.

The design of the architecture and interactive visualization features in Epiviz and its *Bioconductor* and *Python* counterparts, Epivizr and Epivizpy are guided by three specific principles underlying visual analytics of functional genomics data: a) *context iteration:* where context is provided for data by integrating existing data and visualizing multiple measurements at a time. Existing genome browsers are designed with context iteration in mind where data from multiple sources is aligned by genomic location and navigation is supported. Epiviz provides similar functionality while implementing fundamental well-characterized techniques in the interactive visualization field [69]–[71]. Some examples of this include: multiple data sets are *connected* using brushing, users can *encode* data features as visual elements by assigning color on-the-fly, and users can *reconfigure* visualization by looking at different types of plots provided by Epiviz; b) *data iteration,* where data can be *selected* and *filtered* to concentrate on relevant subsets of data. While existing browsers allow this to some degree, the two-way communication allowed by *Epivizr* and *Epivizpy* are powerful implementations of this principle; c) *model iteration,* where new data can be defined as the result of computational or statistical modeling on-the-fly. This is exactly what we are targeting with the development of *Epivizr* and *Epivizpy* to support two-way communication between *Bioconductor* and *Python* respectively, and Epiviz.

New technology to support interactive data-rich web applications like Epiviz is a very active area of development in the data analytics community. As a result of our efforts, we have added support for repeated querying of data by implementing *IntervalForests* both in *R* and *Python*, a collection of interval trees [78] (one for each

sequence in an assembly) for efficient querying of data by genomic location. The *R IntervalForests* extension is now part of the *Bioconductor* code base and available for general efficient overlap operations over genomic intervals.

## 3.4 Epivizpy

Epivizpy is a library that implements the Epiviz WebSockets API, enabling two-way communication between Python and Epiviz. Its functionality is similar to that of Epivizr, but adapted to the Python environment. For example, the main data structure used in the package is the *Pandas data frame*. **Figure 5** shows an example of read coverage data loaded from a SAM file and displayed in a stacked track using Epivizpy. Data is parsed into a data frame and sent to Epiviz using the WebSockets protocol.

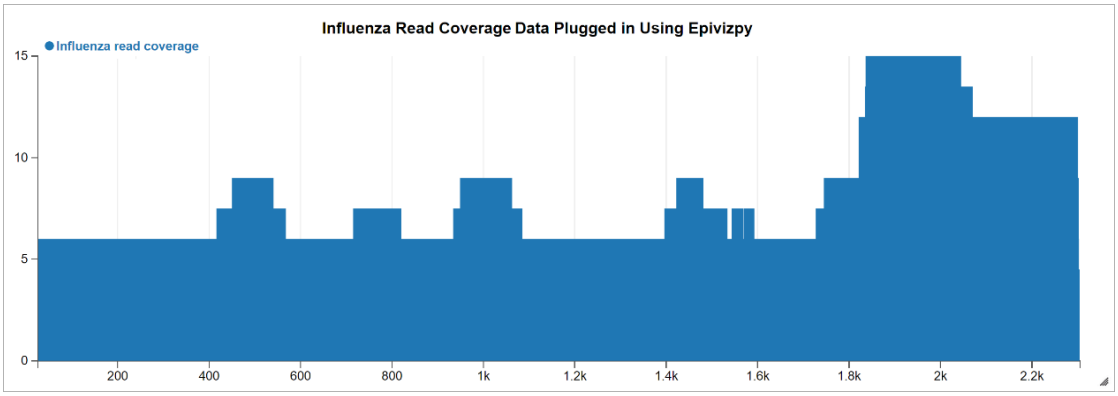


**Figure 5. Influenza read coverage via Epivizpy.** A stacked track showing *influenza read coverage* data loaded from a SAM file, using the Epivizpy Python package.

## 3.5 Software extension using JavaScript dynamic code interpretation

One powerful feature of JavaScript is its ability to evaluate strings of text into runnable code, which Epiviz makes use of, to dynamically incorporate custom user logic into the framework. In this section we expand over the different kinds of

functionalities based on this feature available in Epiviz. Also, in **Section 3.9** we discuss the security implications of taking this path, as well as our approaches to addressing them.

*External scripts*

One way in which we make use of this capability is by providing an extension in the API which permits automatically incorporating user specified external scripts that can override existing visualizations, data providers and settings, or define new ones. The new functionality is ready to use immediately. On launch, Epiviz first loads the base framework logic and searches for scripts specified using user-provided parameters. Epiviz then executes the code in these scripts in sandbox mode and UI elements like menus or measurement lists are immediately updated. Epiviz supports the GitHub Gist API, which makes it possible for users to specify code stored on this source control provider. Using this functionality, users can collaborate on a set of scripts simultaneously, sharing their work while using a common set of *workspaces* as the functionality contained in the scripts evolves.

*Visualization plugins*

The Epiviz help page (http://epiviz.cbcb.umd.edu/help), contains examples of custom visualizations different from the set that comes out-of-the-box with our software. Visualization plugins can be easily created using the Epiviz visualization API, which exposes a series of interfaces and base classes. All visualizations in Epiviz are built on top of this API. The API classes implement basic functionality, like drawing axes, creating a main SVG canvas where drawing is done, drawing legends

etc., which can in turn be used by plugins that only need to implement a *draw* method, greatly simplifying the complexity of plugin code.

**Figure 6** and **Figure 1** show examples of an externally defined visualization: a track, similar to the *blocks track*, where the height of blocks corresponds to *feature values*. The code for this new track is stored on GitHub Gist at http://gist.github.com/160e8b84795603961b9f; it contains three scripts: 1) one that overrides the default Epiviz settings by registering a new visualization type; 2) a subclass of *TrackType*, which defines a set of basic track properties, such as the name of the track to be displayed in Epiviz menus, and 3) a subclass of *Track*, that implements the *draw* method, defining how objects are represented on screen. When plugging in these scripts, the *Track Menu* on the UI is populated automatically with the new visualization, which can be used straight away for all of the available data. A *chart factory* iterates through all registered visualizations, and populates the UI menus accordingly.

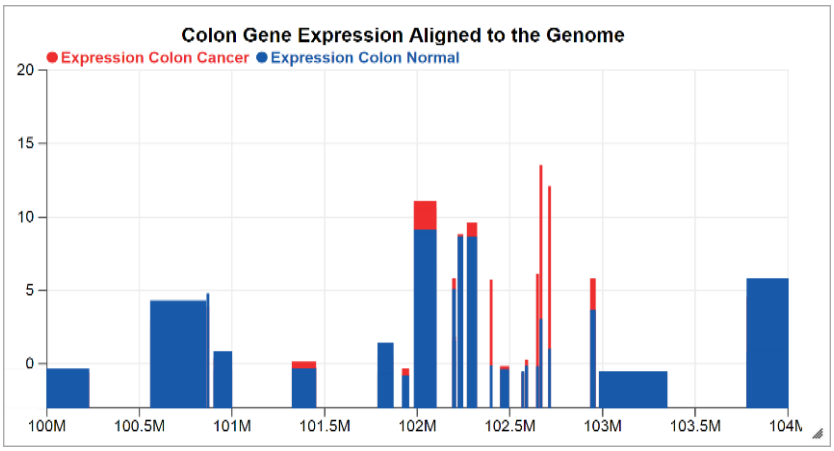


**Figure 6. Custom visualization plugin.** New visualizations can be created and added in Epiviz on-the-fly, by specifying the *script* or *gist* URL argument. In conjunction with the *workspaces* mechanism, this feature allows users to create entire views from scratch and share them in a collaborative project. The visualization plugin is stored via GitHub Gist and is available at http://gist.github.com/160e8b84795603961b9f.

http://epiviz.cbcb.umd.edu/?gist[]=160e8b84795603961b9f&ws=C5ApuwXJlmb

*Data provider plugins*

New *data providers* can be defined in Epiviz using the same mechanism, and a data provider API, designed to allow users to dynamically plug in custom sources of data. Using *data provider* plugins, Epiviz can display data located remotely or on the local machine. Conceptually, Epiviz *data providers* represent proxies to real data sources. For example, the WebSockets *data provider* is used to establish connections with R/Bioconductor through the Epivizr package. Different data providing services are interfaced through an API that de-centralizes data storage by allowing users to easily integrate external data sources. **Figure 7** and **Figure 1** contain a tracks that display human genome genes from the *refGenes* MySQL table, in the UCSC database [61]. This is done through a custom data provider, plugged into Epiviz using the *gist* feature.

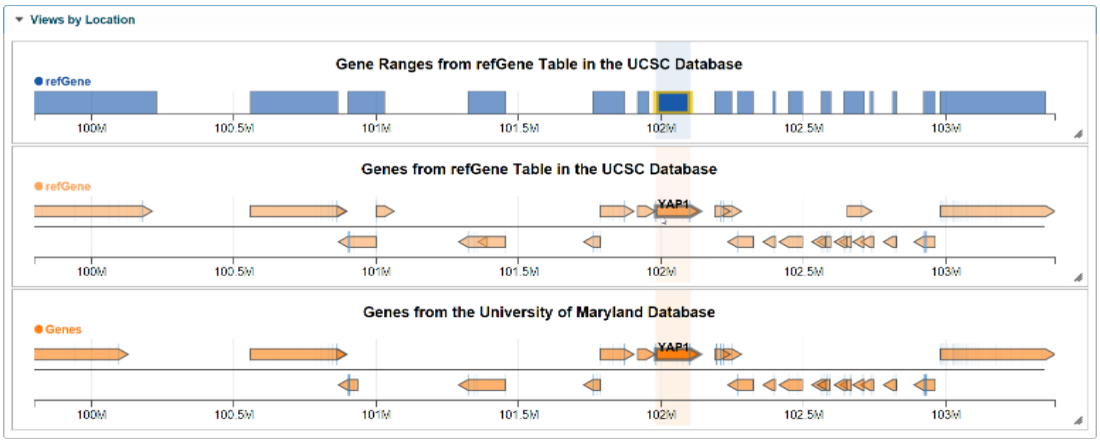


**Figure 7. Data provider plugin for genes from the UCSC database.** Two Epiviz tracks show data retrieved using *data provider* that fetches genes from the UCSC Browser database. Next to them, a track that fetches data from the Epiviz web server. The *data provider* plugin is stored on GitHub Gist at http://gist.github.com/5a88f39caa801e58b8ae.

http://epiviz.cbcb.umd.edu/?ws=yQqwkLWfhip&gist[]=5a88f39caa801e58b8ae

The standardized data format introduced in Epiviz breaks a limitation shared by most genome browsers, which restricts visualization to functional genomics data aligned to a predefined genome. In Epiviz, the coordinate space is defined by *data sources*, through *data providers* (**Section 3.6**). To illustrate this better, we built an example outside the scope of genomics, consisting of a *data provider* for *stock market* data, shown in **Figure 8**. It uses the Yahoo! Finance web service [79] to retrieve information about two stock symbols: AAPL and AMZN.

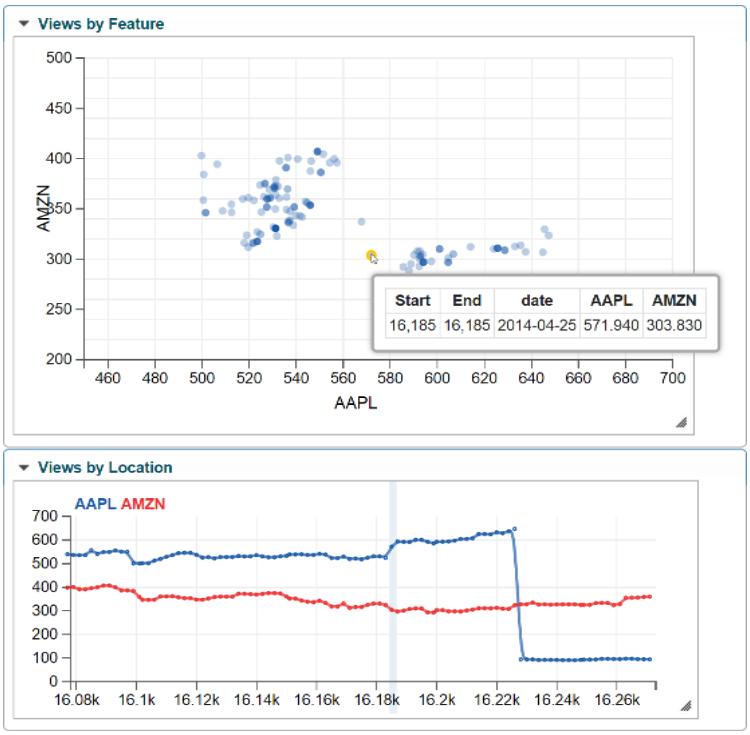


**Figure 8. Stock data via *data provider* plugins.** The scatter plot displays the price of AAPL and AMZN stocks for a given set of days. The lines track shows their evolution over time. The data provider plugin is stored on GitHub Gist: http://gist.github.com/a82a998817564ce3fe48.

http://epiviz.cbcb.umd.edu/2/?ws=SRHZlWRRAPd&gist[]=a82a998817564ce3fe48

*Visualization customization and data transformation in the UI*

Epiviz also introduces a mechanism through which users can customize charts' code directly in the UI, as well as define simple data transformations. Using this, users can alter individual visualizations in place to match their needs. The code of chart *instances* can be modified such as to incorporate additional functionality per user needs. For example, the scatter plot in **Figure 1** contains a line at $y$=0, separating positive and negative gene expression differences. In **Figure 9**, we show

the dialog where users can edit the visualization code, and the code necessary to apply this particular transformation to the scatter plot.

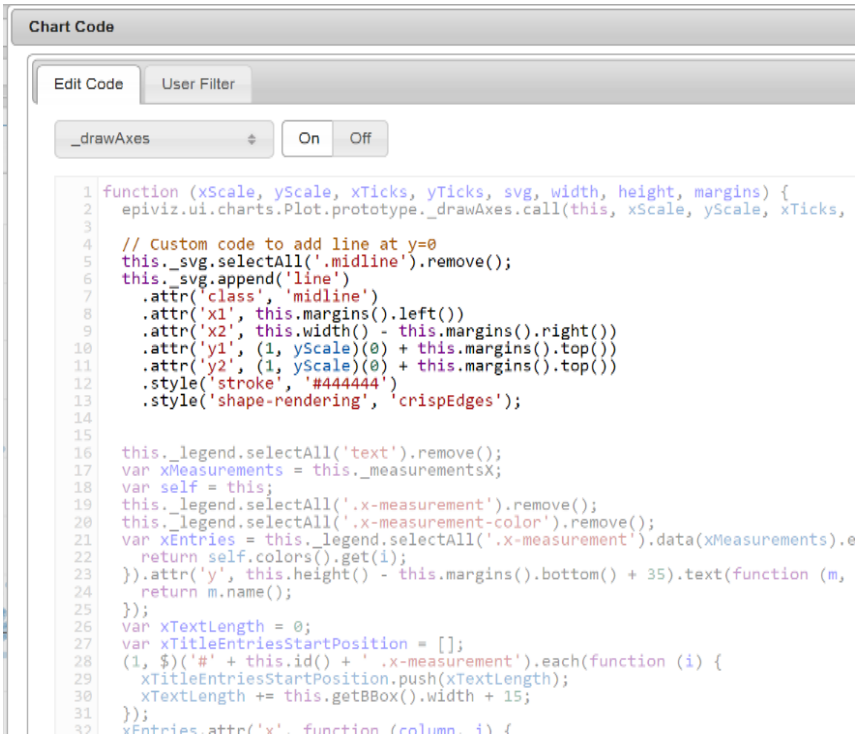


**Figure 9. The *chart code* dialog.** Users of Epiviz can customize the code of individual visualizations directly in the UI, using this embedded JavaScript editor. In this example, we demonstrate the code necessary for adding a line at $y$=0 in the *scatter plot*.

The same type of functionality is used to apply simple data transformations for individual visualizations. These define ways in which the data that comes into a visualization should be transformed prior to rendering. Currently, Epiviz provides support for the following transformations: 1) *filter by* data object properties; 2) *color by* measurement or coordinate properties; 3) *group by* measurement properties; 4) *order by* measurement properties.

In the following lines we expand on these transformations. Each of them exposes two functions that users can implement using the code editor controls in the *code dialog* (**Figure 9**, **Figure 10**). The first is called before any transformation, and is

used to define and initialize variables to be used throughout the transformation; the second function corresponds to the actual transformation, and is called for each object, measurement, or feature coordinate in the selected genomic region, depending on the transformation.

In the *filter by* transformation, the inputs correspond to records in a *data source* (*data source* structures are discussed in detail in **Section 3.6**, and depicted in **Figure 12**). Each input contains coordinate information, as well as a feature value. The *filter by* function yields a *Boolean* with the following semantics: 1) a returned value of *true* signifies that the item should be drawn in the visualization, while 2) *false* means that it should be hidden.

The *color by*, *group by* and *order by* transformations, all have the same signature (**Figure 10**): they can be set up to take as inputs either *data source* records, *measurements*, or *feature coordinates*. Based on the input, the function returns a *label*, in the form of a text string or a number. Labels are used by each transformation accordingly. For example, *color by* will use them to color the objects with the same label with the same color. *Group by* uses a user-selected aggregation function to aggregate all objects with the same label into one visualization object. Finally, *order by* sorts objects in the visualization according to the lexicographic order of their corresponding labels.

For example, the stacked plot in **Figure 1** uses a *color by* transformation, which is used to highlight genes with various expression differences – *0-4*, light blue, *4-8*, dark blue, *8-12*, orange, and *>12*, red. The code necessary to apply this transformation is depicted in **Figure 10**, which shows a screenshot of the contents

of a code transformation dialog. The transformation function returns the same label for all expression differences within an interval of size *4*: assuming *dif* is the difference in expression between normal and cancer genes, then the corresponding label is "Dif: *i*", where $i = \left\lceil \frac{dif}{4} \right\rceil \cdot 4$. For example, for $dif \in [0, 4)$, the resulted label is "Dif: 0"; for $dif \in [4, 8)$, it is "Dif: 4" etc.

Another example is the *lines track* depicted in **Figure 1**, which uses a *group by* transformation, where the two labels are *tumor*, if the measurement corresponds to a tumor sample, and *normal* otherwise.

```
/**
 * This method is called for every data object. If it returns fa
 * @param {epiviz.datatypes.GenomicData.RowItem} [row]
 * @param {epiviz.datatypes.GenomicData} [data]
 * @param {InitialVars} [preMarkResult]
 * @returns {string}
 * @template InitialVars
 */
function(row, data, preMarkResult) {

  var measurements = data.measurements();
  var seriesNormal = data.getSeries(measurements[0]);
  var seriesTumor = data.getSeries(measurements[1]);
  var dif = Math.abs(
    seriesNormal.getByGlobalIndex(row.globalIndex()).value -
    seriesTumor.getByGlobalIndex(row.globalIndex()).value);
  return 'Dif: '+(Math.round(dif / 4) * 4);
}

```

**Figure 10. Custom *color by* transformation for the stacked plot.** This code computes the absolute difference between the two measurements – for example gene expression normal and cancer – in the plot, and splits it in increments of 4. The resulted plot will colour genes with different colours, each corresponding to its expression difference.

The Epiviz API also implements a simple computing language for creation of new measurements from combining existing ones. We call the result *computed measurements*. These differ from the previously mentioned transformations in that they act like regular measurements in the framework, and are available globally for all visualizations to use. The new measurements are computed entirely on the UI side. This permits users to apply simple transformations to data without the need of using a computing environment. For example, based on the gene expression values

for normal and cancer colon tissues, we generated a particular kind of scatter plot also known in the field as an *MA plot* [80] – the $x$ axis shows the average gene expression for normal and cancer colon tissues, while $y$ corresponds to the difference. This plot is used in the use case presented in **Section 5.1** and shown in **Figure 1**. In addition, **Figure 11** shows the EpiViz dialog that facilitates the creation of computed measurements, exemplifying the code necessary to define the *Mean Colon Expression* measurement, as the average value between normal and cancer colon gene expression. Computed measurements values are calculated lazily, as needed by visualizations. This means that no computation is performed at creation time: the values for the new measurements are computed on the fly, just for data represented on the screen.

In the use cases illustrated in **Chapter 5**, we provide several examples of each of these features – chart code customization, data transformations, as well as computed measurements.

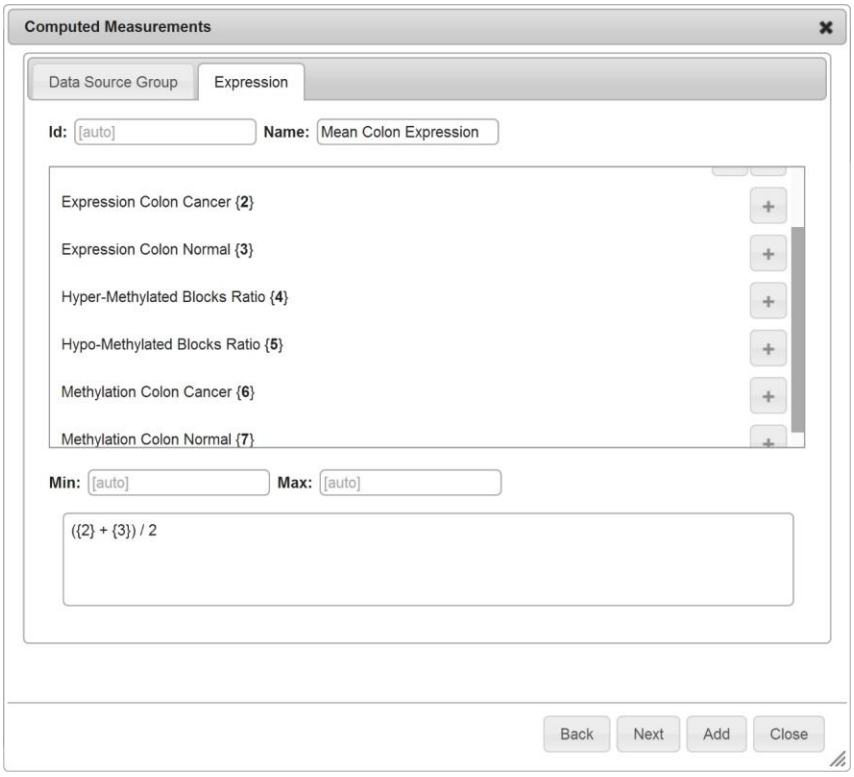


**Figure 11. Screenshot of the *computed measurements* dialog.** New measurements can be computed from existing ones by referencing them in a formula. In this example, the *Mean Colon Expression* measurement computes the average gene expression of colon and cancer samples stored on the University of Maryland web server.

## 3.6 System concepts put together for the first time in genomics

Apart from bringing code to visualizations on various levels, Epiviz also uses a series of design choices and features of which some have been used before for genome browsing, others for other various types of systems. What makes Epiviz stand out is that it is the first software to put all of them together in an integrative genomics interactive visual software. In this section we present some of the most important of these choices, underlining, where necessary, the motivation that led to their development, as well as benefits that follow their implementation.

*Data abstraction and standardization*

One of the most important design decisions in Epiviz was to introduce a series of data structures built on top of an open-ended generic data standard, used to represent a most common known genomic data types. The necessity for this design decision stems from the heterogeneity of genomic data types and the rate at which new ones are released. The unique data format permits different modules of Epiviz to interact with one another without predefining a protocol specific to just those modules. This simplifies the creation of new components, plugging them into Epiviz, and extending existing ones. The standardized data format is the key to our tool's customizability and extensibility, yielding benefits at four different levels: a) allows integrating and aggregating data from different sources, b) allows representing different measurements into the same visualization, for comparison analysis, c) allows representing the same measurement in different visualizations simultaneously in order to explore different aspects of its features, and d) allows Epiviz to expose the API that contains both the visualization and data provider set of base classes used for plugins. In this subsection we briefly present the data standard and some of the more important structures built on top of it.

The data format draws from the *three-table design for genomic data* [15] (**Figure 12**). This design is capable of modelling the vast majority of data present in genomics experiments. Based on this design, we derived a data structure abstraction called, generically, *data source*. The *data source* acts as a table with metadata annotating both rows and columns. Each row in a *data source* corresponds to a *feature*, having associated a specific coordinate; each column corresponds to a measurement or

sample. Thus, each cell in the table represents the measurement value at a particular coordinate or for a particular feature. *Data source* tables tend to be large enough for it not to be feasible to store them entirely in memory. For this reason, Epiviz retrieves only chunks of these tables in memory as needed. *Data sources* are treated as single tables in the Epiviz logic, although in practice, their corresponding data often comes from different physical tables or even different *data providers*. For this reason, coordinate information, common to all measurements in a *data source*, is separated from the actual measurements and retrieved independently, so redundant operations are avoided and no more data than needed is requested or loaded in memory at any given time.

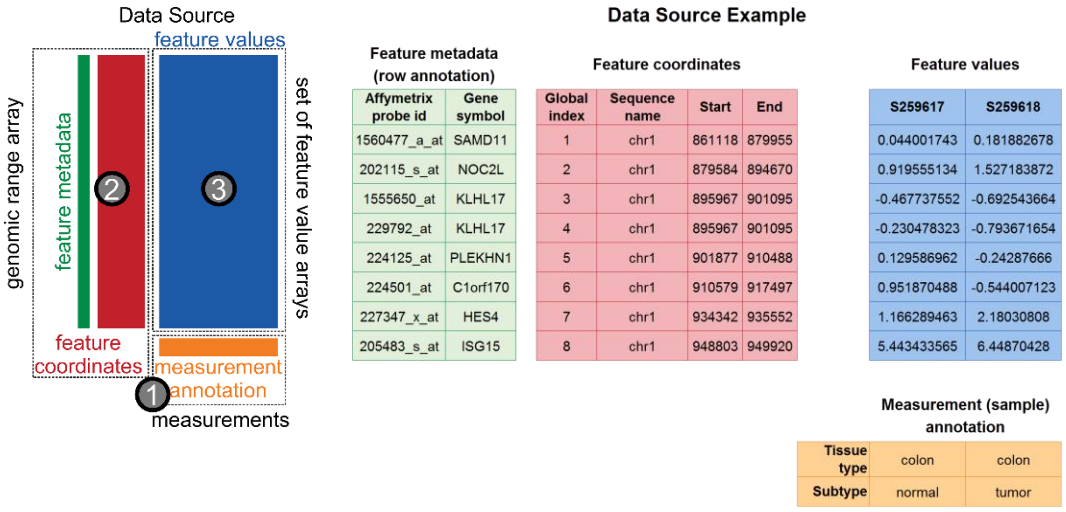


**Figure 12. Epiviz *data sources*.** A standardized data format based on the *three-table* design. The schematic on the left shows the general structure, while on the right we have an example for microarray gene expression data. The three tables are 1) *measurements* (samples), 2) *feature coordinates* stored as *genomic range arrays*, and 3) *feature values*, stored as sets of *feature value arrays*. Each data point is the value for a particular feature for a specific measurement or sample. *Features* occupy a *coordinate space* and have associated metadata. Samples have annotations associated as well.

An important substructure of *data sources* is constituted by *genomic arrays*. These encode all information corresponding to measurements' data objects. There

are two types of genomic arrays in Epiviz – *genomic range arrays*, used to store coordinate information, and *feature value arrays*, used to store feature values for individual measurements. Genomic arrays are designed to store fragments of the data sets they represent, so that although not all data is loaded in memory at once, Epiviz modules can treat them as if they did. For this purpose, each row in a data source table has an identifier denoted as *global index*, which represents the index of a row in the entire data source, ordered by its coordinate. Global indices are passed in increasing consecutive order between modules, which simplifies operations on the UI that only require logarithmic time for searches and constant time for merging fragments of data.

*Serialization optimizations*

To improve user response time and performance, Epiviz implements some data serialization optimization heuristics, to reduce its representation size. Some such heuristics consist of reducing the precision of floating point numbers, omitting data that is redundant (like *end* genomic locations for data objects that consist of just one base pair), or offsetting genomic locations using the lower bound of the *genomic array* in which they are stored. The Epiviz communication protocol allows data providers to specify which of these optimizations have been applied when sending it to the UI. The Epiviz web server *data provider*, for example, implements a few of these. In conjunction, these optimizations reduce data representation size up to ~30% of its original size. The time overhead on the UI side, where data is reconstituted upon retrieval, is usually negligible compared to that of waiting for the decompressed

data to be transmitted over a slow network. For data providers residing on the same machine as the UI though (like Epivizr), best results are obtained when these optimizations are turned off.

*Visual encodings*

The Epiviz UI offers data scientists a combination of *multiple coordinated views* and *overlays*, featuring *brushing* and *linking*. The main goal behind this is to enable both data comparison and visual validation in order to help users extract insights and gain both an overall and detailed understanding of the data. Epiviz provides out-of-the-box visualizations that are both *feature-* and *location-oriented*, to help provide a multidimensional comprehension of the explored domain. In addition, a class of visualizations target the *structure of data sources* directly. All types of visualizations offer in turn different kinds of graphical representations; for example some of the available feature-oriented views are the *heatmap* and *stacked plot*, while some of the available location-oriented views are the *genes* and *lines tracks* (**Figure 1**). One example of data structure visualization is the *icicle*, which we discuss in detail in **Section 4.2**. This differentiates Epiviz from most other genome browsers, which usually feature only *genomic location-oriented* visualizations.

All graphics are rendered using Scalable Vector Graphics (SVG), an XML vector format that all modern browsers can interpret. Choosing this format allows users to treat objects in charts independently, as direct representations of data [16], as well as to perform specific operations on them, by customizing their properties – shape, color, size, stroke, transparency, etc. This opens the door to a wide variety of options

available directly to visualizations, of which perhaps the most important are *brushing and linking*, *object tooltips*, and the ability to save views as both vector and raster static images.

*Navigation*

Like other genome browsers, Epiviz facilitates dynamic navigation across the genome, using controls like the *chromosome selector*, the *location text box*, *left/right navigation buttons* and the *zoom buttons*. In addition, Epiviz permits searching for and automatically navigating to the location of known genes and Affymetrix [81] probes. Using these, users can analyze and compare measurements side by side in different parts of the genome, jumping from one location to another. Used in conjunction with workspaces, these controls make analyses of regions of interest an easy task, the user being able to switch between such regions by switching between active *workspaces*, specifying genes of interest, or simply typing the desired locations.

Epiviz also allows users to conveniently alternate between general views of larger portions of the genome, at low resolution, with zoomed views of specific small regions at high resolution. In **Section 5.1** we show a use case that illustrates this in a real world experiment. All visualizations implement mechanisms of synthesizing the displayed data, and grouping similar objects together, thus not cluttering the screen, while also providing meaningful summaries of the data.

*Generalized coordinate space*

The Epiviz framework is defined with genomic coordinates in mind – the space is partitioned into sequences (e.g. chromosomes) that define strict boundaries within

which any data can be represented. However, a remarkable side-effect of standardizing the data is that coordinate spaces can be defined in *data sources*. **Sections 3.4**, **3.5**, as well as **5.5** present several such examples, where visualized data belongs to different coordinate spaces than that of the human genome. In the first, we used the Epivizpy plugin to visualize genome *read coverage* data from the *influenza* virus, requesting the presentation tier to hide any information about the human genome, irrelevant in this particular experiment. Not being bound to a particular coordinate space, Epiviz has no trouble displaying data from this new genome. In the second, as a proof of concept, we used Epiviz to display stock data provided by a financial web service [79]. In this case, data is not partitioned, and the coordinate space corresponds to time (i.e. each coordinate unit is a day since December 12, 1980). Regardless of the domain, given data formatted appropriately, Epiviz is able to visualize and explore it the same way it does with genomic data. In the third case, the main coordinate is given by the list of available *operational taxonomic units* in the dataset. In **Chapter 4** we show how we used this feature to extend the functionality of Epiviz to *metagenomics*.

*Brushing*

Through brushing, users have the ability to visually link data from all visualizations on the screen (**Figure 1**, **Figure 2**, **Figure 32**, **Figure 37** and **Figure 38**). By hovering over/selecting a particular object in one chart, related objects are automatically highlighted in all other charts as well. The unified data types used in Epiviz include identifiers for *data source groups* which declare keys for

each set of observations, to establish data relationships used in the brushing feature. Therefore, all *data sources* from the same group are assumed to have the same keys. In the absence of keys, we use feature coordinate overlap to establish these relationships as well. Notice that this design is extremely flexible since keys defining data relationships are defined dynamically. Brushing is available in Epiviz due to the choice of SVG as the rendering mechanism, since each object on the screen corresponds to an HTML element in the DOM. Hovering or clicking on an object thus triggers events that all visualizations listen to in order to decide which objects will be highlighted at their end.

*Tooltips*

Along with brushing, cursor hovering also produces a tooltip (that can be turned on and off on demand) which contains information about the group of objects hovered on the screen. This includes *start* and *end* chromosome locations, and all metadata fields associated with the objects. If the visual object corresponds to a group of size larger than a constant predefined in Epiviz, only data belonging to a subsample of the group is displayed (**Figure 13**, **Figure 1**).

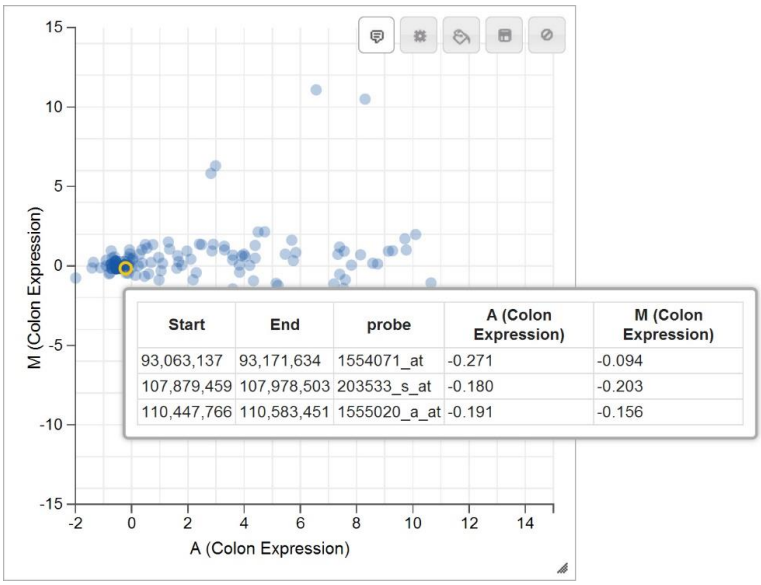


**Figure 13. Tooltip.** When hovering the mouse over data objects in visualizations, a tooltip is displayed, showing information about the objects in the hovered region. The displayed information includes genomic coordinates of data objects and metadata associated to them. In the example, the tooltip contains a column for Affymetrix [81] probe identifiers.

*From visualizations to static images*

Representing all views in a standard vectorial format (SVG) enables Epiviz to save the contents of charts as static image files, both vector types – PDF, PS and EPS – and raster types, like PNG. The PHP/MySQL web server *data provider* features a script that converts the standard SVG type into any of these formats, as chosen by the user. This feature is particularly useful in generating figures for publications, where converting between images formats often gives authors a hard time.

*Visualization customizable features*

The Epiviz *Chart API* implements a mechanism that allows visualizations to define custom settings and features. For example, the *lines track* has an option to

choose the interpolation method between consecutive data points (in **Figure 1**, the lines track uses *B-spline* to interpolate methylation levels). A more compelling example is the *clustering* feature implemented in the *Heatmap* visualization. This feature allows users to use hierarchical clustering in order to reorganize measurements in the heatmap locally, and derive insights (in **Figure 1** for instance, using this feature reveals that cancer samples tend to cluster separately from healthy ones, across tissue types – breast, kidney, etc. – with respect to their gene expression. In other words, two cancer samples from different tissue types are more similar than one cancer and one normal sample from the same tissue). More compelling examples are presented in **Chapter 5**.

## 3.7 Collaboration through sharing of analysis steps

One important functionality essential to scientific data analysis, and yet inexistent in current genomics visualization tools, is that of persistently saving and sharing steps of an analysis within the scientific community. Often times, operations that should be straightforward, such as replicating results presented in publications pose big challenges and come with heavy overhead. Through Epiviz, we take the first steps in the direction of simplifying this process, through a feature called *workspaces*. *Workspaces* store entire analysis states, including all visualizations on the screen, their customizations (screen coordinates, code changes, data transformations and colors), and the current coordinate range in view, so that any user action within the software could be tracked and reproduced.

*Workspaces* are stored in a database on the Epiviz server hosted at the University of Maryland. To save analysis steps, users need to authenticate using an OpenID (OAuth [82]) account. Once logged in, users can create new workspaces and replicate existing ones shared with them. Storage management for persistent *workspaces* is part of the data provider API. Once created, a *workspace* is associated with a unique id that can be used to share individual work with other users, through permanent hyperlinks. Using the *workspace* id, any user can view a particular *workspace*, and copy its contents to their own account. This mechanism can be used for either sharing data analyses between users or even for referencing figures in publications. Most figures in this dissertation contain a hyperlink pointing to the corresponding workspace in Epiviz; **Figure 2** for instance, can be accessed at http://epiviz.cbcb.umd.edu/?ws=sp9ShCJdS3c.

Persistent workspaces for sessions hosted at http://epiviz.cbcb.umd.edu are stored in the servers hosted at the University of Maryland where OAuth is used for authentication so only users that create workspaces can edit them. Storage management for persistent workspaces is part of the *data provider* API. The source code for this type of server is publicly available on the Epiviz project page and can be installed on a standard PHP/MySQL system to provide the same functionality if users desire to keep their workspaces private on a local server. Users can create new workspaces by authenticating and using the *workspace* UI controls.

## 3.8 Performance optimizations and benchmarks

Epiviz implements a number of simple optimizations meant to improve user experience. These constitute our preliminary steps in this direction. The most important optimization consists of a predictive caching mechanism depicted in **Figure 14** and described below. The other type of optimization is individual to each visualization, and consists of *binned aggregation* [83]. Together, they are designed to enhance the user experience by minimizing both wait time and cluttering on the screen.

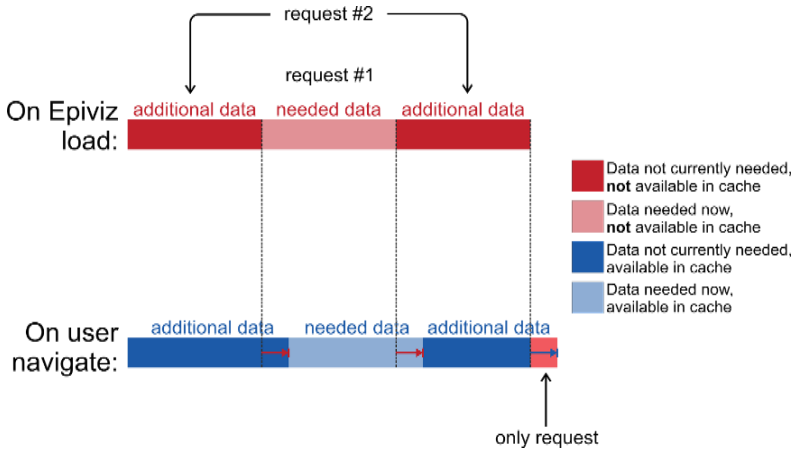


**Figure 14. Predictive caching.** On load or navigation to a new region in the genome, the data manager makes two requests: first for the data needed to update the visualizations on screen, and second for data in its vicinity. Once this data is loaded, subsequent pan and zoom operations are executed immediately, as all data necessary to fulfill them is already stored in memory. These subsequent operations also trigger a small additional request per operation, to keep the data in the cache consistent.

As part of our performance analysis, we explored the effects of these lines of optimization over the overall responsiveness of the browser. We developed a JavaScript test suite on top of the Epiviz base code that measures the load and draw time of our visualizations with caching turned on or off, and with different values for the individual custom parameters of charts. The tests were conducted on an Intel

Core i7-3740QM CPU at 2.7 GHz, system with 16 GB of RAM, running Windows 8.1. Epiviz was loaded in the Google Chrome web browser. The back-end MySQL database and PHP server were running on the same machine, so network latency for the tests was negligible.

*Adaptive and predictive data management for responsive data querying*

We have designed a methodology to ensure responsiveness of visual and computational exploratory data analysis systems. We implemented a simple predictive caching strategy that requests data in advance for simple predicted user actions (**Figure 14**). The requests from the UI first stop at the *cache*, which, if the requested data is already available, serves it back without sending further requests to the *data providers*. If all or part of the requested data is not available in the cache, the cache sends multiple requests to the corresponding *data provider*: one for the requested data, and two for an equal amount of data for the regions in coordinate space immediately before and after the requested region. These requests pre-empt requests that would be made by navigating to neighboring genomic regions, and when zooming out of the current genomic region. The resulting user experience is that the user only has to wait once for the initial request to be fulfilled, all other requests being done in advance, taking advantage of user interaction idle periods. In preliminary tests, we observed that this simple strategy results in substantial performance benefits (**Figure 15**).

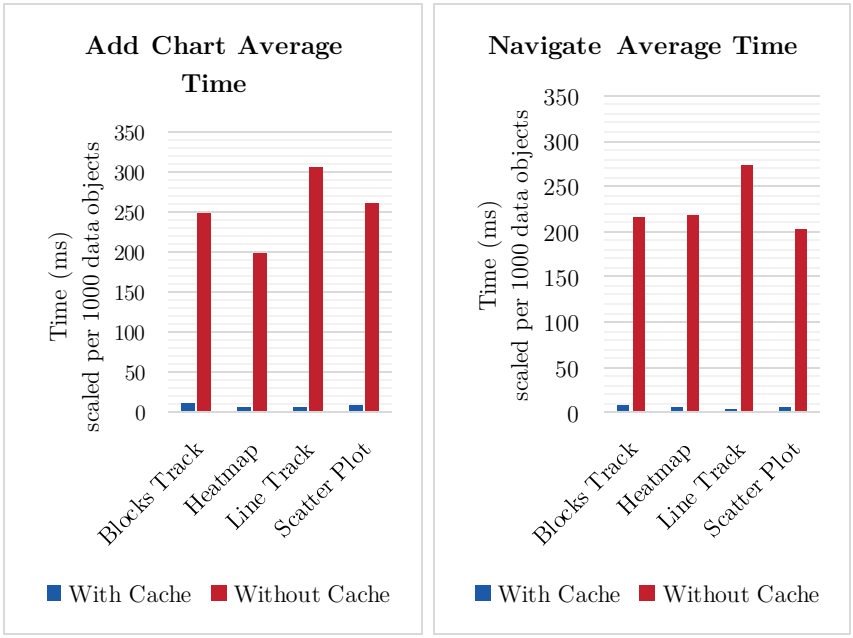


**Figure 15. Comparison of Epiviz operations with and without cache.** Average time comparison of time taken by *add chart* and *navigate* operations per 1,000 data objects with and without using the cache.

The *cache* is a component of the *data manager*, a module that sits between the UI and the data providers. When a user initiates a request – like navigate, zoom, etc. – the UI sends a request for the data corresponding to each of the measurements present in the charts on the screen to the *data manager*. In its turn, for requests that require a lot of time to fulfil, the *data manager* uses the *cache* as an intermediary to retrieve the data. The *cache* computes the adjacent genomic regions to the one currently in view, and if enough data is stored in the cache to cover the entire first region, the cache module immediately sends back a wrapper around the requested data, so the charts can be updated (**Figure 14**). For the regions that don't yet have data in *cache*, requests are made to the appropriate data providers. As the requested data is retrieved, it is merged to data already stored in the *cache*, and every time, a new check is made, whether there is enough data to finally update the charts needing

it. The data request workflow is optimized to avoid requesting and loading redundant data: at all times, only new data is requested from the data providers, as the cache takes into account the pending requests that have not yet been fulfilled, so the same data is not requested two subsequent times.

*Visualization detail vs. performance*

Epiviz also supports the following summarization and binning operations in commonly used visualizations of data from genomics and other biomedical domains [83], [84]:

*Scatter plot*: We parameterized the radius of the drawn points in relation to the minimum of the chart height and width such that if multiple data points overlap a region of this radius, only one point is drawn and transparency is used for underlining locations with higher density of points within location

*Blocks track*: if the distance in screen pixels between two blocks is small, they are merged into one causing less objects to be drawn on the screen while a brushing feature will summarize the number of underlying regions merged.

*Heatmap*: Columns are averaged into a single column based on a parameter determining the maximum number of columns to be drawn.

*Lines track*: points are sampled uniformly across the requested coordinate range to a number depending on the number pixels available in the viewing area.

We have observed that these types of optimizations provide substantial performance benefit for exploratory visualization systems (**Figure 16** and **Figure 17**). In our performance tests suite, we measured the relation between the number

of data objects available to draw, and the number of objects drawn, showing how that affects the total draw time of charts. Tuning the visualization parameters as the requested data increases, offers users the power of adjusting their user experience and also a straightforward way of summarizing data. Our results show how the performance of Epiviz dramatically increases with appropriate values for these parameters.

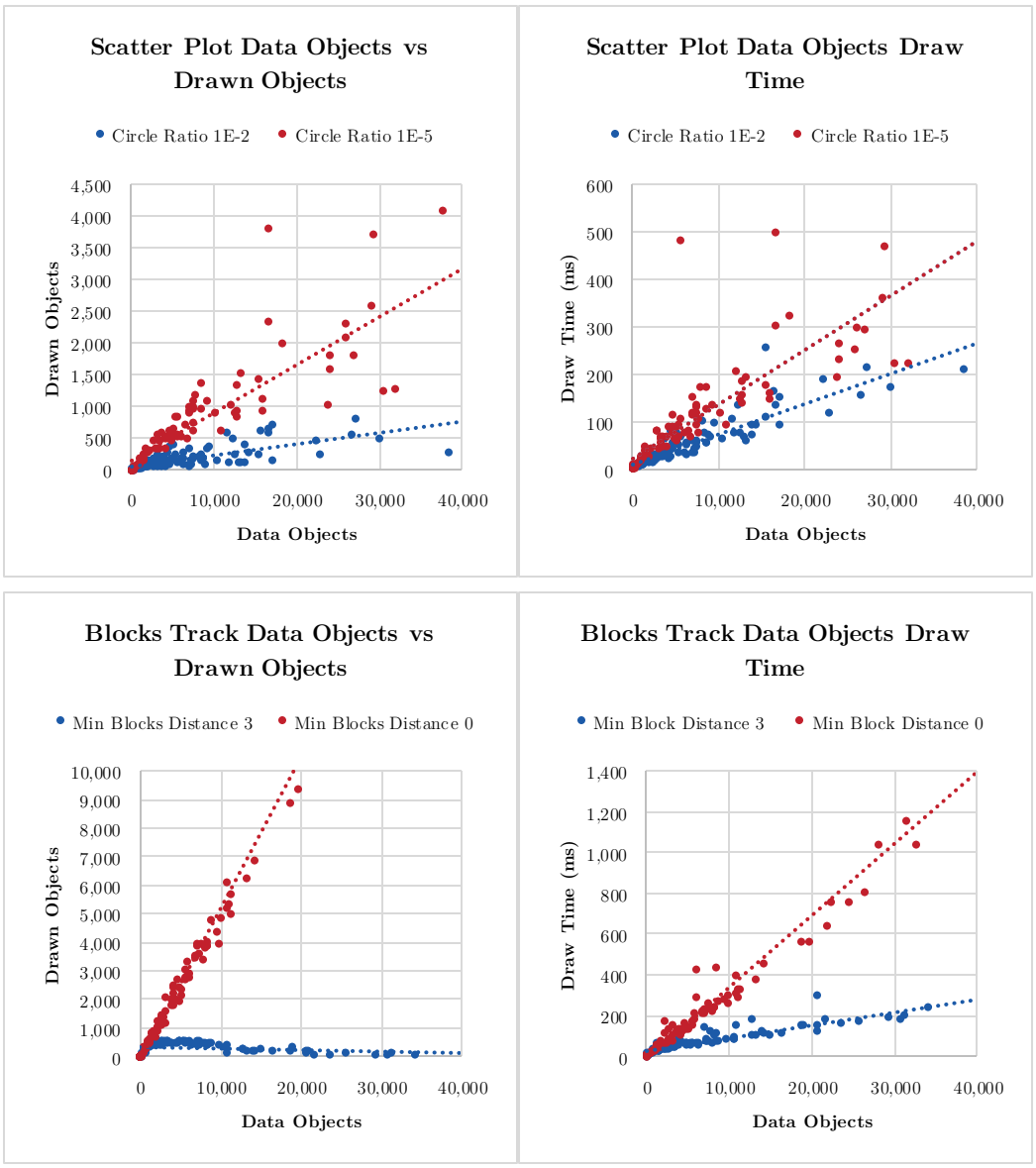


**Figure 16. Draw performance of scatter plot and blocks track.** A comparison of draw times when varying specific chart parameters for *scatter plot* and *blocks track*.

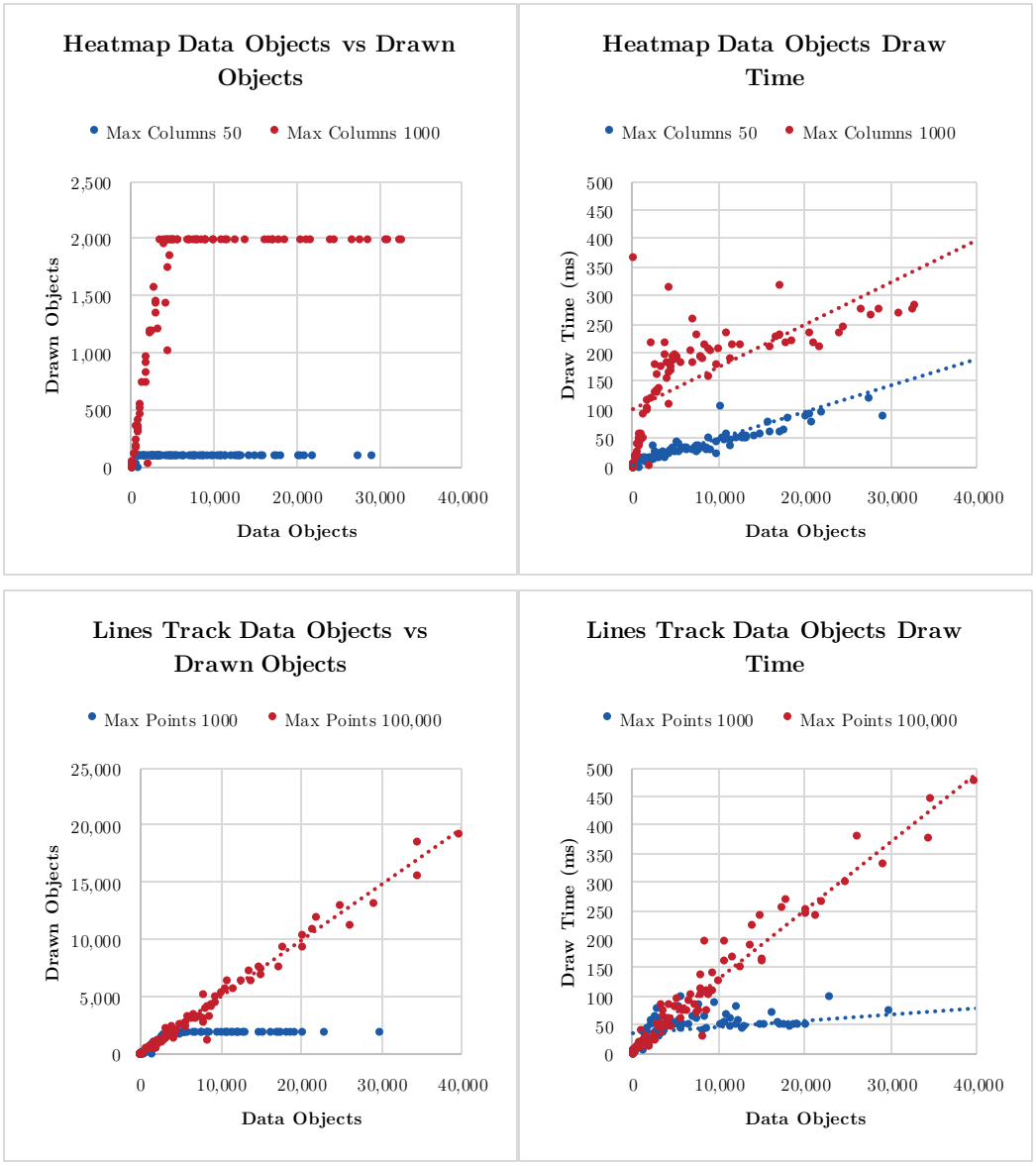


**Figure 17. Draw performance of heatmap and lines track.** A comparison of draw times when varying specific chart parameters for the *heatmap* and *lines track*.

*Performance evaluation framework*

As visualization for functional genomics research is becoming more established, it is becoming increasingly important that the research in this field is validated.

We created an automated test framework, to measure the performance of various operations in Epiviz, the results of which were presented in the previous section. The

test framework is built on top of the Epiviz architecture. The user interface consists of a panel showing the tests loaded into the workspace (**Figure 18**).

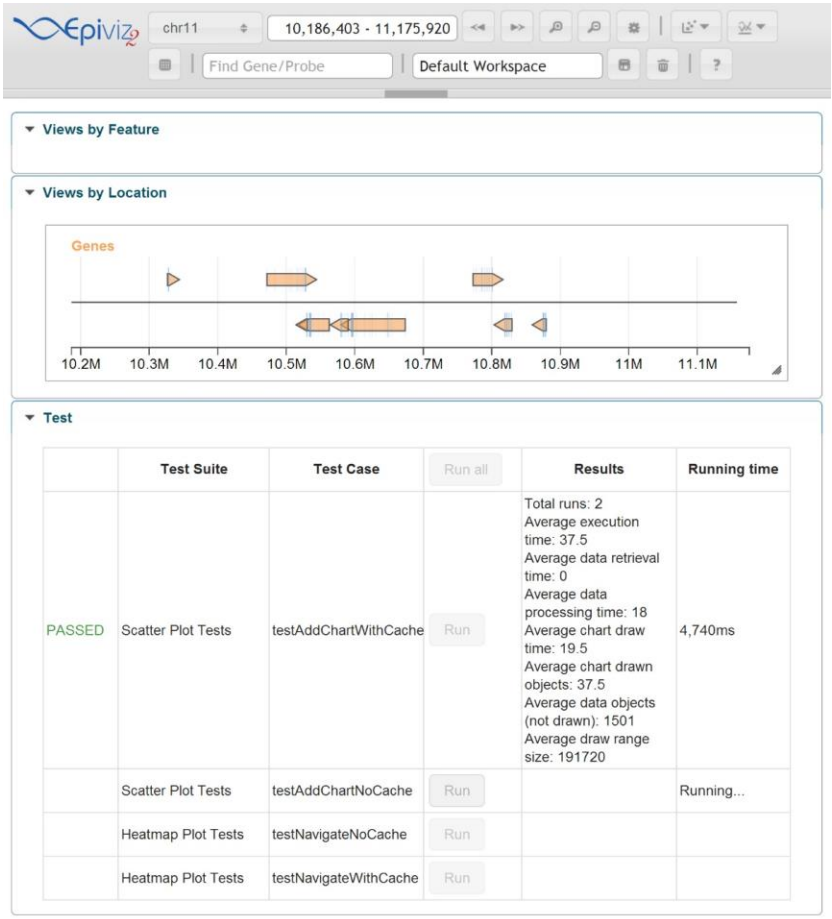


**Figure 18. Test framework user interface.** Tests are grouped in test suites. Each test can be run independently, or alternatively, the entire set of tests can be run sequentially. Each test logs specific execution information and reports outcome and running time.

Behind the scenes, tests are grouped in test suites, customized to the needs of specific modules. Test suites are managed by a service called *test manager*. The test manager connects the test suites to the UI, binding each row in a controller table to a test in a test suite. Tests can be executed either one at a time, all tests in a suite, or the entire list of available tests. The UI reports the outcome of each test (passed

or failed), running status, execution time, and specific log information, used for tracking errors in the code.

Each test suite derives from a base class called *TestSuite*, which is initialized with the active Epiviz settings, as well as handlers to all of the available modules and services (i.e., *LocationManager, MeasurementsManager, ChartFactory, ChartManager, ControlManager, DataProviderFactory, DataManager, UserManager, WebArgsManager* and the main controller).

Test cases consist of methods in TestSuite instances that are prefixed with "test". **Table 3** shows the method that retrieves all tests in a suite. The methods corresponding to test cases have a specific prototype – `function(outputContainer, finishCallback)`, where `outputContainer` points to a HTML element where test logging can outputted (the *Results* column in **Figure 18**), and `finishCallback` is a function called on test completion (**Table 4**).

```
/**
 * @returns {Array.<string>}
 */
epiviztest.TestSuite.prototype.getTestCases = function() {
  var result = [];
  for (var member in this) {
    if (jQuery.isFunction(this[member]) &&
        epiviz.utils.stringStartsWith(member, 'test')) {
      result.push(member);
    }
  }
  return result;
};
```

**Table 3. The method getTestCases().** This method is used to dynamically fetch the list of test cases available in a test suite.

On top of this framework, we built tests that measure the performance of different operations in Epiviz. The test framework is extensible through external plugins, following the same development model used for the data provider and

visualizations plugin mechanism. This way, users that actively contribute to the Epiviz framework with custom plugins have the option to use the test platform to validate their work. **Table 4** provides an example of the general outline of such a test.

```
/**
 * @param {jQuery} resultContainer
 * @param {function} finishCallback
 */
epiviztest.MockTestSuite.prototype.testCheckScatterPlotSVGObjects =
    function(resultContainer, finishCallback) {
  var scatterPlotType = this.epivizFramework.chartFactory.get('Scatter Plot');
  var measurements = ...;
  var chartId = ...;
  var chartProperties = ...;
  var chart = this.epivizFramework.chartManager.addChart(scatterPlotType,
                measurements, chartId, chartProperties);

  var location = ...;
  var data = ...;

  var testResult = { passed: true };

  // Hijack draw method of chart to check drawn objects are correct
  /** @type {function} */
  var chartOwnDraw = chart.draw;
  chart.draw = function() {
    this.draw = chartOwnDraw; // Only do this once

    // Get the list of ui objects drawn by the chart
    /** @type {Array.<epiviz.ui.charts.UiObject>} */
    var uiObjects = chartDraw.apply(this, arguments);

    // Check the actual svg content of the chart
    var drawnPointsSelection = this._svg.selectAll('circle');
    drawnPointsSelection.each(function() {
      var point = d3.select(this);
      var radius = point.attr('r');
      var x = point.attr('cx');
      var y = point.attr('cy');
      // Here check that drawn object is correct
      if (/*point doesn't satisfy test condition*/) {
        testResult = { passed: false, message: "Failure detail message" };
        resultContainer.append("Failure detail message");
      }
    });

    return uiObjects;
  };

  this.epivizFramework.chartManager.updateCharts(location, data, [chartId]);
  finishCallback(testResult);
};
```

**Table 4. A test case outline example for the scatter plot.**

## 3.9 Notes about software security

One challenge that comes with allowing Epiviz to incorporate and run third-party scripts consists of the security risks associated with SQL injection and cross-site scripting (XSS) [85]. The main concern is that, having access to private user information, third-party scripts could be used to compromise user content, privacy, and sensitive data. In this section we underline the ways in which Epiviz addresses these concerns in order to provide a safe data analysis environment.

Epiviz features a server side component, and a client JavaScript component, the latter containing the entire framework functionality, described in this paper. The server side component contains a web service for a number of public epigenetic datasets, similar to those hosted on other genome browser servers, as well as private Epiviz user information, such as their OpenID account data provided when signing up, and *workspaces* associated to it. The public sets of epigenetic data are served in a read-only fashion, no changes to it being permitted to users accessing the service. OpenID information cannot be retrieved using the web services endpoint; in addition, information about a particular user cannot be modified externally after the user has logged in for the first time, having successfully authenticated using the corresponding OpenID provider. The only information that can be both retrieved and modified externally on the server side is that of user workspaces. Changes to workspaces can be made within the session of their owner. All database access for the web server is achieved through PHP prepared statements [86] which are guaranteed safe from SQL injection.

This implementation leaves room for only one type of potential attacks. Malicious external JavaScript code, being incorporated into Epiviz using the dynamic code interpretation feature, might gain access to the user session and potentially compromise workspaces information, the only kind of private information that can be accessed and modified externally. Alternatively, it could extract private information such as that stored in user cookies, and transmit it to a phishing server using, for example, AJAX calls. This constitutes a great vulnerability, and therefore, we needed to find a solution which would allow the execution of third-party scripts in a sandbox mode, with no access to sensitive information or actions that might compromise data integrity.

To address this vulnerability, we decided to use a JavaScript sanitizing library called Caja [43], which allows third-party scripts to be executed by the same JavaScript runtime environment as the framework code, but in sandbox mode, with restrictions defined within the host script.

The Caja library establishes a connection to a Caja server hosted at http://caja.appspot.com. When users of Epiviz supply custom code, by using JavaScript dynamic extension, our framework asks Caja to construct a virtual DOM to which objects in the script are limited to. Epiviz also supplies a series of *defensive objects* to Caja. *Defensive objects* are the only objects within the framework that are accessible to the script and their implementation assumes that their clients *can* be malicious. Caja transforms the code to make it safe to run, by sending a GET request to the Caja server, wrapping the raw user code. The server returns the transformed code. The creators of the library call this process *cajoling*. Cajoling involves adding

inline checks to make sure the code does not break the invariants Caja needs, and ensuring that the code cannot refer to variables in the host page that are not explicitly given to it. It also makes sure that the user code only uses the API published by the Epiviz framework.

From the viewpoint of the user code, it runs with what seems to be a W3C DOM compliant document object and an ECMAScript 5 compliant JavaScript virtual machine. Its document is confined to the boundaries of this virtual DOM, and its JavaScript global types and objects, like Object and Array, are its own and do not affect code outside it. The *defensive objects* are visible to user code as additional global variables in its top-level JavaScript context.

Using this library, Epiviz is able to restrict third-party scripts from any access to sensitive information such as the user cookie, opening pop-up windows or browser tabs, contacting third-party web servers, as well as the entire workspace data. Whether the scripts are stored externally, or they consist of custom code inserted using the web-browser code editor, they are executed in protected mode, which effectively defends against XSS attacks. This security feature allows Epiviz to be the first visualization tool to allow its users the wide range of power that comes with expanding functionality through third-party plugins and code customization.

# 4 Interactive visualization of object hierarchies. Applications in metagenomics

## 4.1 Motivation

The democratization of high throughput DNA sequencers has allowed exploring deeply and simultaneously the entire microbial DNA present in an ecosystem, also known as the *metagenome*. Due to the massive amount of data generated by a single run of these high throughput sequencers a new discipline emerged, called *Metagenomics* [87], with the aim to characterize the microbial composition of an ecosystem and understand the underlying drivers and host-interactions. High throughput sequencing has revolutionized the field through targeted sequencing of the ubiquitous 16S marker-gene and whole metagenomic shotgun (WMS) sequencing. The vast amounts of sequencing data generated have led to public databases of sequence data and summary collections including MG-RAST [88], GOLD [89], GigaDB [90], EBI metagenomics [91], and Bioconductor datasets. However, it is a computational challenge simply dealing with the enormously sized datasets and these rich datasets requiring appropriate normalization and testing methods for downstream analysis.

A number of visualization tools exist that attempt to tackle the problem of metagenomics data visualization, but fail to keep up with the demands of the field. In the past decade, the challenges in all branches of genomics have shifted from data gathering, to data analysis, modeling and dissemination [88], [91]. However, current metagenomic visualization tools are not in tune with this trend, and still use approaches to data analysis that take into account neither the size of data awaiting

interpretation, nor the need to alternate between visualization and data modeling. Instead, they are built around moderately sized datasets, limited to one data source per analysis, and their visualizations are mostly static. For example, both of Phinch [92] and VAMPS [93] are web-based tools that allow users to upload custom data, but do not provide an easy way to visualize it side by side with publicly available data. In addition, these tools do not provide an easy way to analyze and link different dimensions of the same data using multiple coordinated views. Other tools that are desktop-based, such as Krona [94], are even more restrictive, usually coming with OS-dependencies. But the most important drawback of these tools is that none supports the integration with an established statistical and computational environment such as R/Bioconductor, thus providing only limited amount of analysis and exploration capability.

In tackling the problem of metagenomics data analysis, we attempt to use the lessons learned from functional genomics analysis and the principles that apply. First of all, having observed the benefits of a tool that binds visualization and computational environments together in functional genomics, we imported the same idea to metagenomics. Therefore, we extended the Epiviz framework to provide support for hierarchically organized data, such as those prevalent in this field. On the R/Bioconductor side, we built a package called *Metavizr*, as a façade between the established Bioconductor metagenomics package metagenomeSeq [95] and Epiviz. The Epiviz visualization framework assumes all datasets to be large, therefore giving the option of loading and exploring partitions of the data at a time, without loading all in memory at once.

*Metagenomics data*

Metagenomics data usually consists of two components: 1) read counts produced by high-throughput sequencers that target the 16S ribosomal RNA gene from selected samples, and 2) operational taxonomic units (OTUs) which correspond to clusters of similar sequence reads [96]. Representative sequences from each cluster are annotated, a process of labeling the read to a particular organism, against a database of 16S rDNA reference sequences including Ribosomal Database Project [97]. The number of reads assigned to a particular organism is an approximation of the abundance of that organism in the community.

The taxonomy is usually organized in a hierarchy corresponding to the *microbiome phylogenetic tree*, where each leaf corresponds to one OTU. Organism abundances (counts) are organized in a table which contains one *column* for each *sample* in the study, and one *row* (*feature*) for each *OTU*. Therefore, each value in this table corresponds to *the abundance of a specific OTU in a specific sample*. In our work, we store *counts tables* as Epiviz *feature value tables* (**Figure 12**). To be able to represent metagenomics data in Epiviz, we defined the *feature coordinates* as the indices of the *features* in the *counts table*.

In the following sections we expand over the additional architecture decisions we made to allow the analysis, visualization, and manipulation of the hierarchical component of this data, represented by the taxonomy.

*The need for hierarchy analysis and representation in genomics*

The most important addition to the Epiviz architecture required for metagenomics data analysis was a series of components that introduce support for hierarchically organized datasets. These, however, are by no means limited to describing just metagenomics data. In fact, organizing datasets hierarchically represents the key to addressing limitations imposed by assumptions that were at the base of the original design of Epiviz. Epiviz was initially conceived as a genome browser. As such, much of its original interaction model revolved around the assumption that all data is aligned to a primary coordinate, which serves as axis for all measurements, and is fixed for each *data source*. For example, this means that objects that are far away from each other in the genome, will be proportionally far away from each other in visualizations. This assumption does not take into account scenarios where the user needs to visualize side by side and compare two regions of interest that are far away from each other in the genome. Sometimes, analyses require objects or entire user-defined regions to be hidden from view, or aggregated into one object. All of these operations could not be done before without altering the original *data source* in a computational environment. However, as the scope of the tool shifted and expanded to new use cases and other types of genomic data analyses, it became clear that this assumption was no longer valid. Organizing the data hierarchically and allowing users to modify these hierarchies provides a way to address such drawbacks and broaden the reach of our visualization framework.

## 4.2 Methods

*New components to support hierarchy manipulation*

Most genome browsers are designed to display data along a predefined ordered coordinate, usually a genome. We illustrated earlier (**Figure 5** and **Figure 8**) how, by introducing a standardized data format, Epiviz moves the definition of this coordinate away from the architecture of the framework, allowing each *data source* to specify it independently. This is particularly useful when users need to analyze data originating from multiple genomes. Another important benefit of this design is that Epiviz is not restricted to genomic data at all; earlier we showed an example where a *data source* corresponding to stock market prices uses *time* as the main coordinate system and can be used in the exact same way as data aligned to the genome.

This design facilitated the expansion of the Epiviz data standard in order to include support for the analysis, exploration, and manipulation of *data sources* organized hierarchically. We introduced an optional fourth component to the *three table design*, consisting of a *tree* that binds to the main coordinate of the data source. The leaves of the tree correspond to records in the data source (**Figure 19**); subtrees cover entire regions over the main coordinate. This structure can be used as a handle into manipulating the way the other components of the data source can be visualized. The following elements can be altered within the data analysis workflow: 1) *node order* – the order in which the children of each node are traversed, which leads to a well-defined order of the leaves of the tree (**Figure 19**), and 2) *node selection*. *Node selection* can be of three kinds: a) *none* determines the data

corresponding to the sub-tree rooted at the current node to be hidden in all visualizations; b) *aggregate* causes that the leaves of the sub-tree to be aggregated into a single value; and c) *leaves*, which determines all leaves of the sub-tree to be evaluated independently. By manipulating this structure, users can re-order regions that are far away from each other in the data source to visualize side by side. Also, entire regions of the data can be hidden, and others can be aggregated into a single value (**Figure 19** and **Figure 20**).

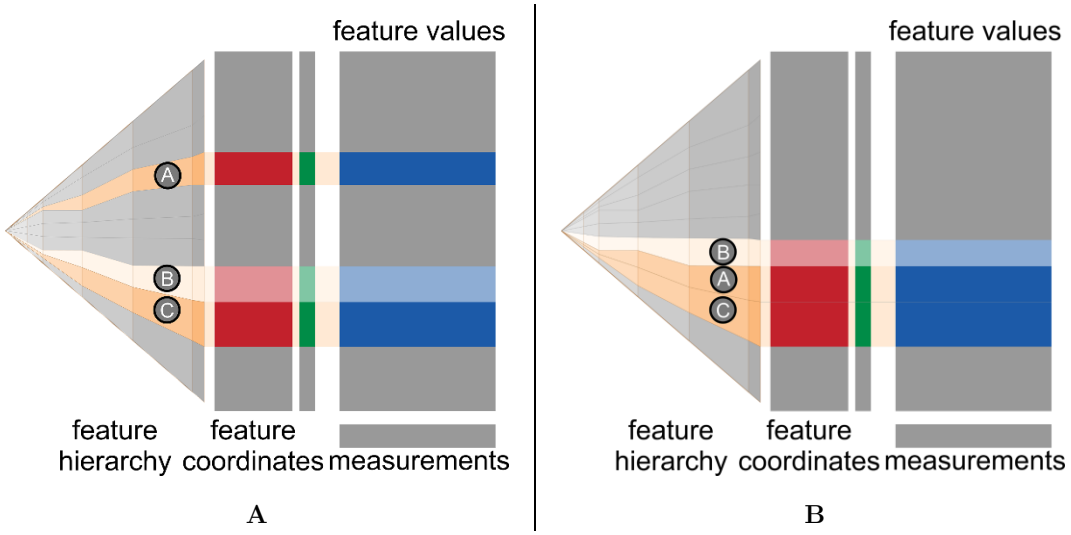


**Figure 19. *Data source* with *feature hierarchy* component.** In addition to the three tables – *measurements*, *feature values*, and *feature coordinates*, we introduced a new component corresponding to *feature hierarchy*. Users of Epiviz can manipulate this component, for example by changing the order of the nodes in the hierarchy. Figure 19**A** shows the four components of a data source, highlighting three regions, (A), (B) and (C), of which (A) and (C) represent regions of interest that are far away on the feature coordinate axis. It is important to notice that each continuous region in the data source corresponds to a sub-tree in the feature hierarchy. By changing the order of the sub-trees (A) and (C), users can visualize these regions side by side – Figure 19**B**.

We added support for this new component in the *data provider protocol*. The communication paradigm follows the same principle we used for retrieving feature values and coordinates: only chunks of the entire data are requested at any time, to improve performance and user experience. For hierarchies, this means getting only

sub-trees of the entire hierarchy, by specifying their root and depth. In addition, we extended the WebSockets protocol to include a call through which Epiviz can send requests to back-ends to propagate user changes over the structure of the tree, such as node *order* and *selection*.

The Epiviz API defines the following operations that must be supported by *data providers* in order to facilitate hierarchy visualization: 1) *getRows* – requests the coordinates for all *features* within a specified range. This corresponds to records in the *feature coordinates* table in the *data source* (**Figure 12**), or in the case of hierarchies, to leaves (**Figure 19**); 2) *getValues* – requests *feature values* for a set of *measurements* (*samples*) and *features* within a specified range; 3) *getHierarchy* – requests the sub-tree rooted at a given node in the hierarchy, of a predefined maximum depth, to display in visualizations by *data structure*, such as the *icicle*; 4) *propagateHierarchyChanges* – sends user customizations over the hierarchy (like changes in *node order* or *node selection*) to the *data provider*. This causes the hierarchy to be modified on the *data provider* side. In turn, this causes Epiviz to make new *getRows* and *getValues* requests for the data in all visualizations linked to the updated hierarchy. The *getRows* and *getValues* calls are inherited from the original *data provider* API, while *getHierarchy* and *propagateHierarchyChanges* are, as their names suggest, specifically associated with hierarchically-organized data. The first two are used to update visualizations *by coordinate* and *by feature*, while the last two correspond to visualizations *by data structure*.

*Visualizations for hierarchies*

Two types of visualizations were initially supported by Epiviz. The first display data *by feature* and the second *by coordinate*. Introducing the hierarchy component required a new type of visualization along with a new interaction model inspired by Shiny's Reactivity [74], which would allow the underlying data structure behind a data source to be altered through UI interaction. These are visualizations that display data *by data structure*, allowing the user to perform changes on it, which are propagated to the *data provider*. In the following paragraphs we briefly discuss the motivation behind choosing a suitable UI control for hierarchy visualization and manipulation.

The Epiviz interaction model imposes a number of requirements over the type of visualization suited for this control: 1) it needs to be scalable in a way that allows representation of large trees with minimal information loss; 2) allows easy navigation over different subparts of the tree; 3) each node in the tree needs to have its own representation – this is important for datasets where different nodes and hierarchy levels have distinct semantics (for example, in *phylogenetic trees*, each tree level corresponds to a *taxonomic rank*, and each node to a *taxa*); 4) the order of nodes in the tree has to be well-defined, such that it can be linked to the order of objects in other visualizations (for example, both heatmaps and lines plots show data in a user-defined order, which if altered, changes the meaning of the charts). The order should be easy to modify by a user through interaction with the visualization.

The field of information visualization offers a wide variety of options for hierarchies, among which some of the most common are the *treemap* [98]–[101], the

*sunburst* [102], widely used in *phylogenetic tree* representations [94], [103], [104], and the *icicle* [105], [106], a variation over the sunburst, depicted in an *axis-parallel* coordinate system.

Perhaps the most established type of 2-D visualization for hierarchies in information visualization is the *treemap*. One of its key advantages is that it provides a scalable solution to use limited screen space for representing large tree structures. Moreover, it offers a good intuition over sizes and proportions of subsets of data. With color coding, it can also be used to extract insights from clustered data. For our interaction model however, we found this visualization to be unsuitable, because it comes with two essential drawbacks: 1) only leaves in the tree are explicitly depicted, making it hard to show information about internal nodes; 2) the 2-D layout of the nodes in the structure makes it hard to specify and customize well-defined orders within the tree, in order to link objects to those in other visualizations.

Since the primary usage scenario for this control involved *metagenomics*, we also investigated visualizations for phylogenetic trees, of which the most frequently used is the *sunburst*. This type of display allows easy navigation through the tree, represents each node individually, and preserves order among nodes. There are two aspects of the sunburst that prompted us to look for an alternative: 1) its inefficient use of screen space (for example, if the sunburst is drawn in a square container, then the area of the circle corresponding to the space covered by the sunburst covers less than 80% of the container); but the most important is that 2) although it does preserve order, its layout does not pair well with the other visualizations in Epiviz, making it unclear which part of the radial display corresponds to components within

other displays which are linear, and causing confusion when associated with the brushing/linking feature.

The *icicle* is a variation over the sunburst, depicted in an axis-parallel coordinate system. Unlike the sunburst, it makes use of the entire screen space available. But most importantly, the linear order of nodes corresponds exactly to that of items in other visualizations, which can be easily and intuitively linked by a user. This made the *icicle* the most suitable choice to combine visualization of hierarchies with navigation and manipulation. Because of this, we used it as our first exponent of visualizations *by data structure* (**Figure 20**). Epiviz users can utilize icicles to navigate over the tree corresponding to a data source, and change both the *order* and *selection* of nodes.

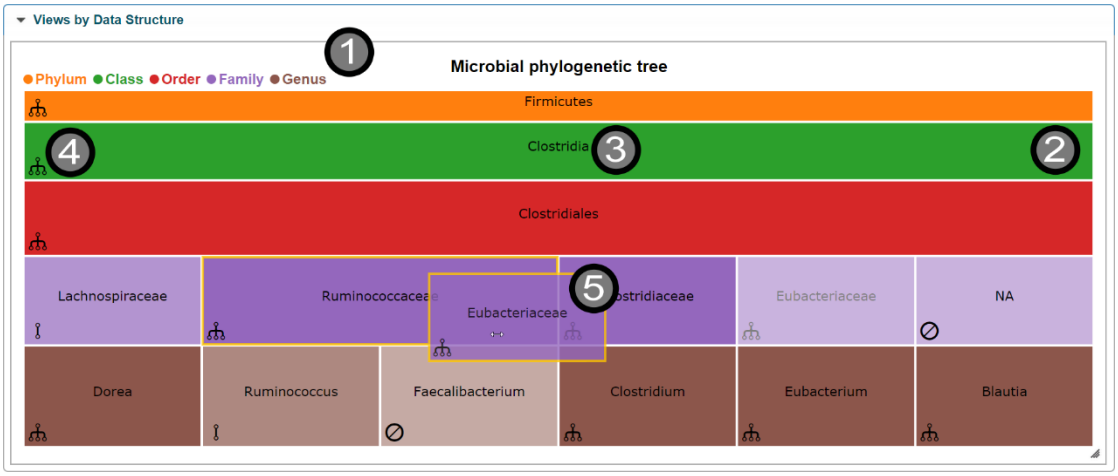


**Figure 20. An *icicle* visualization, of a microbial phylogenetic tree.** The following components can be distinguished: 1) legend – each level in the tree is assigned one color; in the case of bacteria phylogeny, these correspond to *ranks*, i.e. *kingdom*, *family*, *genus*, etc.; 2) nodes, depicted as rectangular cells. All children corresponding to a node are depicted underneath their parent; 3) a label for each of the nodes; 4) node selection icon. Node selection can be dynamically toggled by the user, by clicking on the icon. The order of the children within a node can be altered, through drag-drop 5). In addition, navigating in depth of the structure can be done by clicking on the rectangles corresponding to nodes.

*Metavizr: Integrative visualization for metagenomics*

We created a Bioconductor package called *Metavizr* to address the need for data visualization and exploration in the growing field of Metagenomics. Its architecture builds on top of the metagenomeSeq [95] Bioconductor package for metagenomics data summarization and manipulation, and Epivizr [2], which provides support for the Epiviz WebSockets protocol, and offers an easily extensible mechanism for components of this protocol – either new, or that need overriding.

The package architecture is based on a data type called *EpivizMetagenomicsData*. This is a class that wraps around an *MRexperiment* [95] object, which summarizes the taxonomy and sample abundance of bacteria involved in a metagenomics experiment. The class overrides handlers for the Epiviz WebSockets protocol, defining measurements based on sample abundances, and mapping the data source coordinates to the leaves (operational taxonomic units – OTUs) of the taxonomy hierarchical structure. In addition, it implements request handlers for components of the protocol that are not natively supported by Epivizr, specifically those related to hierarchy manipulation.

*Performance*

The architecture of Epiviz pushes computation-heavy operations away from the visualization framework, to *data providers*, which also applies to the new hierarchy components. In **Section 3.8**, we have discussed the performance associated with the most ubiquitous of our visualizations, including improvements in user experience due to caching for navigation. Therefore, in this section we discuss performance

considerations over hierarchical operations in the Metavizr package, which currently constitutes our main *data provider* for hierarchically organized data.

As stated earlier, the main operations that need to be implemented by *data providers* to support hierarchy visualization are: *getHierarchy*, *propagateHierarchyChanges*, *getRows* and *getValues*.

In Metavizr, the complexity associated with *getHierarchy* is linear in the number of leaves corresponding to the sub-tree requested (**Figure 21**). For optimal experience, users can customize a Metavizr parameter called *maxDepth* which controls the maximum sub-tree depth for visualizations. Depending on the dataset, users should take into account that the number of nodes in a sub-tree grows exponentially with its depth. Thus, the value of the parameter should find a compromise between level of detail (tree depth) and performance. For example, for the study presented in **Section 5.5**, we kept the value of *maxDepth* to *5*, which yields sub-trees of at most *164*, and on average *57* leaves (**Figure 21**).

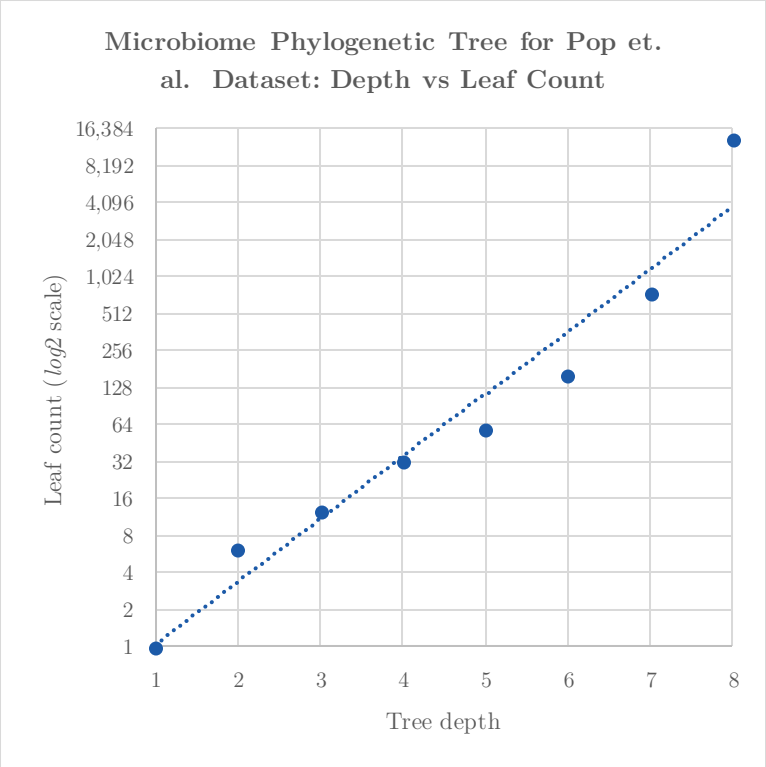


**Figure 21. Exponential growth with depth, of the number of leaves.** The plot shows the average number of leaves in sub-trees of different depths generated from the dataset in Pop et al. [109].

The *propagateHierarchyChanges* operation executed in constant time. But because the operation affects alters the structure of the *data source* hierarchy, it is usually paired with a set of *getRows* and *getValues* requests, aimed to update the visualizations linked to the *data source*.

The *getRows* and *getValues* requests are usually called together, depending on the visualizations on the UI. (For example, the *heatmap* in **Section 5.5**, displaying data for *484* samples and *43* OTUs makes one *getRows* request to retrieve the information about the OTUs and *484* requests for *getValues*, one for each sample.) Both of these operations are executed in time linear to the number of leaves necessary and features respectively, necessary to display all visualizations in the workspace that are linked to the *data source* (**Figure 22** and **Figure 23**). Because these are

usually executed together, we also measured the time performance of Metavizr as a function of the sum of these numbers. The results suggest that for our current implementation both the number of leaves and the number of samples have the same impact over performance (**Figure 24**).

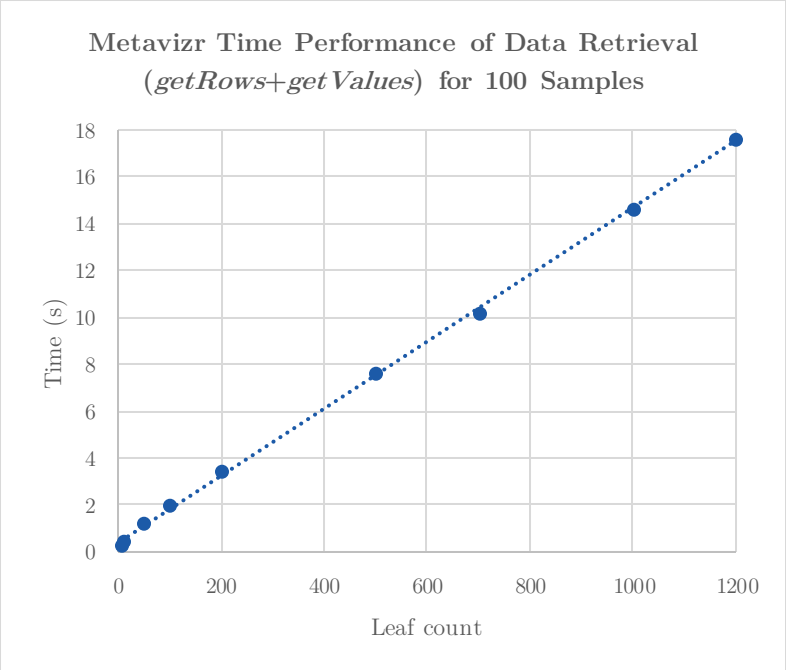


**Figure 22. Time performance for retrieving data for different size hierarchies and fixed sample count.** The plot shows the average execution time required by *getRows* and *getValues* operations together, for a fixed number of samples and different leaf counts.

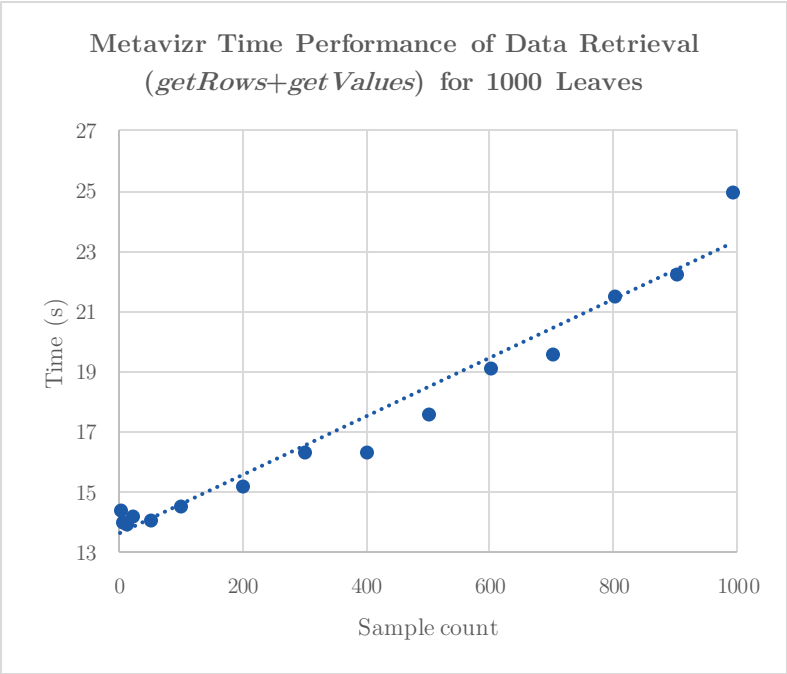


**Figure 23. Time performance for retrieving data for different numbers of samples and fixed hierarchy size.** The plot shows the average execution time required by *getRows* and *getValues* operations together, for a fixed number of leaves and different sample counts.

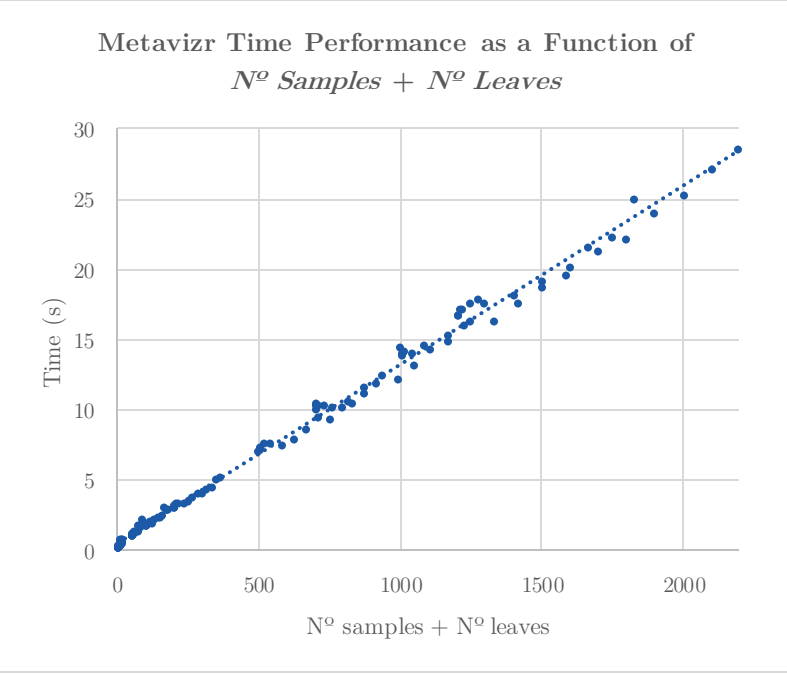


**Figure 24. Time performance as a function of *number of samples + number of leaves*.** The plot shows the average execution time as a linear function of the sum of sample and leaf counts, suggesting that both of them have the same impact over performance.

# 5 Experiments and case studies

## 5.1 Exploring gene expression in relation to DNA methylation

Our first use case highlights the most powerful features of Epiviz in a functional genomics experiment. We used Epiviz to explore the relationship between DNA methylation and gene expression within normal and tumor colon tissues. Our goal was to examine regions in the genome where the difference in gene expression between the two is large.

We started with a *genes track* showing gene models from the UCSC database. To fetch this data, we used a custom *data provider* plugin stored on GitHub Gist. In order to find regions in the genome where expression differences are large, we used two *computed measurements*, corresponding to the *average*, and *difference* gene expression, respectively, for RNAseq data from chromosome 11, to generate an MA plot [80]. We displayed these using a *scatter plot*. In order to better observe expression differences, we dynamically customized the code of the scatter plot to show a line at $y$=0. The dialog and code corresponding to this customization is are depicted in **Figure 9**. We identified two genes with large expression difference by selecting the outliers in this plot from all genes in chromosome 11: *MMP1* and *MMP3*. The brushing feature allowed us to observe that these outliers are also adjacent in the genome. To find the exact genomic locations of these genes, we hovered over them to trigger a tooltip (**Figure 25**, **Figure 26**, **Figure 27** and **Figure 1**).

We zoomed into a smaller genomic region to examine these genes at high resolution. To check whether expression differences are consistent across tissue types,

we added a *heatmap* with aggregated expression data from the Gene Expression Barcode[42], for six different tissue types, both normal and tumor: colon, stomach, breast, kidney, lung and esophagus. We used this visualization's *clustering* feature to group tissues based on gene expression similarity within this genomic region. The high expression differences for the *MMP1* and *MMP3* genes, between normal and tumor tissues, across tissue types, played a decisive role in the clustering result (**Figure 26**).

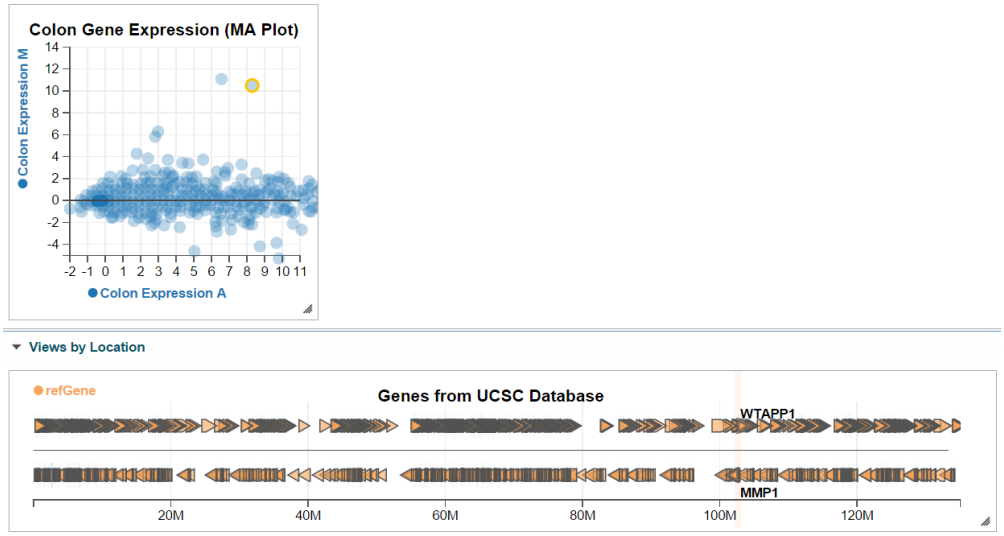


**Figure 25. An overview of gene expression in chromosome 11.** The *scatter plot* shows colon normal and tumour gene expression average on the $x$ axis, and difference on the $y$ axis. The *genes track* shows genes fetched from the UCSC database, using a *data provider* plugin hosted on GitHub Gist (http://gist.github.com/5a88f39caa801e58b8ae). The highlighted data point in the scatter plot corresponds to a gene expression difference outlier. Using the *brushing* feature of Epiviz, we link this outlier to its corresponding gene in the *genes track*.

http://epiviz.cbcb.umd.edu/?gist[]=5a88f39caa801e58b8ae&ws=gdmUH1ANl3m

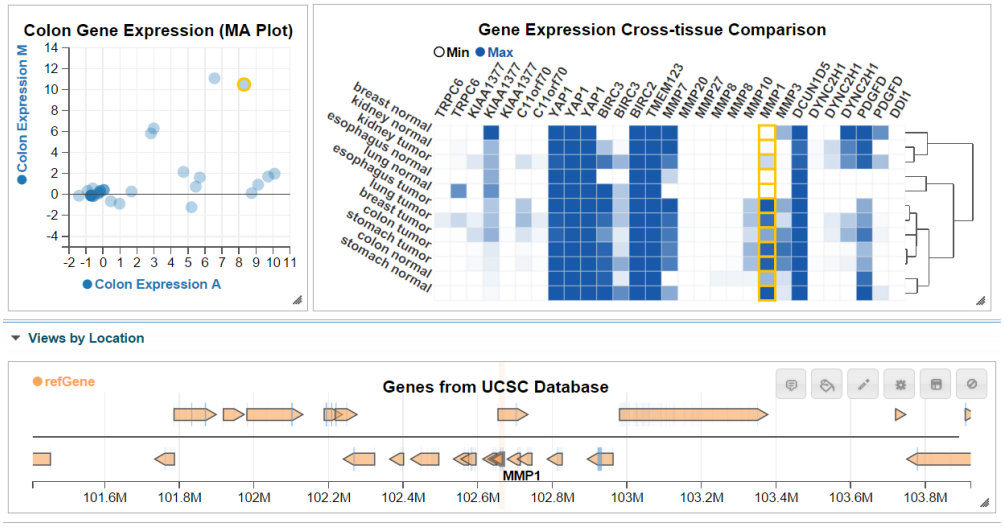


**Figure 26. A comparison of the expression of gene MMP1 across tissue types.** Using the clustering feature of the *heatmap* we notice that the gene MMP1, along with MMP10 and MMP3 determine tumours to separate from healthy tissues, based on gene expression in this genomic region.

http://epiviz.cbcb.umd.edu/?gist[]=5a88f39caa801e58b8ae&ws=S3ppS5e5cei

Next, we added a new visualization called *stacked plot*, showing two columns, corresponding to the summed gene expression for normal and cancer tissues respectively. This visualization stacks values for different genes, one on top of the other, depicting each gene with a different color. Using a *color by* transformation, we customized this plot to color-code different genes according to the expression differences (**Figure 27**, **Figure 10**). A detailed explanation of the code pertaining to this transformation is presented in **Section 3.5**. Analyzing the result yields a couple of insights: first, that overall gene expression tends to be higher for cancer tissues; and second, that genes with high expression difference tend to be collocated in the same region of the genome (**Figure 27**, **Figure 1**).


**Figure 27. Comparison of hypo/hyper-methylated regions and gene expression for normal and cancer colon tissues.** To generate this figure, we created a visualization plugin stored on GitHub Gist (http://gist.github.com/160e8b84795603961b9f). The track shows blocks corresponding to colon gene expression values aligned to the genome, based on gene coordinates (first in the list of tracks in the figure). The second track is a *stacked track* showing a *computed measurement* corresponding to the difference in methylation levels between normal and cancer tissues.

http://epiviz.cbcb.umd.edu/?gist[]=160e8b84795603961b9f&gist[]=5a88f39caa801e58b8ae&ws=u4woieRRaxp

To examine gene expression along the genomic coordinate in relation with DNA methylation, we created a custom track plugin, which we stored on GitHub Gist. The track displays genes as blocks aligned to the genome, whose height corresponds to the gene expression (**Figure 27**, **Figure 6**; **Section 3.5** offers more details about this visualization). We also added a new visualization called *stacked track*, which we used to display a *computed measurement*, corresponding to the difference in methylation levels between normal and cancer samples (**Figure 27**). The data for these measurements corresponds to base-pair-resolution smoothed methylation log ratio resulted from sequencing of bisulfite-converted DNA [25]. The advantage of

this type of visualization over the traditional *blocks track* which we used in the previous version of our work is that this offers not only information about the location of *hypomethylated blocks*, but in addition, it provides a two-dimensional high resolution understanding of the structure of these blocks. For example, examining these two tracks side by side, we were able to extract a number of insights: first, the density of genes increases as methylation difference decreases; second, gene expression for both normal and tumor samples increases in regions where methylation difference is smaller; and third, we were able to differentiate between hypomethylated regions by their level of auto-correlation (smoothness) at base-pair level, which links well in this region with gene expression differences.

Finally, we created a lines track showing the methylation levels for six different samples, three normal and three tumor. We used the *group by* transformation, to aggregate normal and cancer samples together respectively (**Figure 1**). A high resolution view over the data allowed us to further differentiate between the two hypomethylated blocks in view: the first, ranging from 102.3Mbp to 102.9Mbp, and the second, ranging from 103Mbp onward.

We were able to uncover a yet undocumented insight: we discovered that although at the block-level there is an overall difference between cancer and normal methylation, the degree of auto-correlation in the methylation data at base pair resolution varies within the block. Furthermore, we observed that genes that show high differences in expression tend to collocate with regions of low methylation auto-correlation (lower smoothness) while genes that are not differentially expressed collocate with regions of high methylation auto-correlation (high smoothness). This

analysis suggests that understanding the relationship between expression and methylation within long epigenetic domains requires that methylation data is analyzed at multiple genomic resolutions. Using Epiviz, it was easy to switch between low and high resolutions to facilitate these type of multi-resolution analyses.

## 5.2 Interactive visual analysis of the colon cancer methylome

We analyzed the colon cancer methylome as reported in Hansen et al. [25] using Epiviz. Our goal was to interactively visualize large regions in the genome where DNA methylation modifications occur in cancer in the context of other genomic and epigenomic features. **Figure 28** replicates the analysis in Hansen et al. of a region in chromosome 11. We were able to validate the overlap of regions of methylation loss in colon cancer and partially methylated domains (PMDs) reported in fibroblast [28].

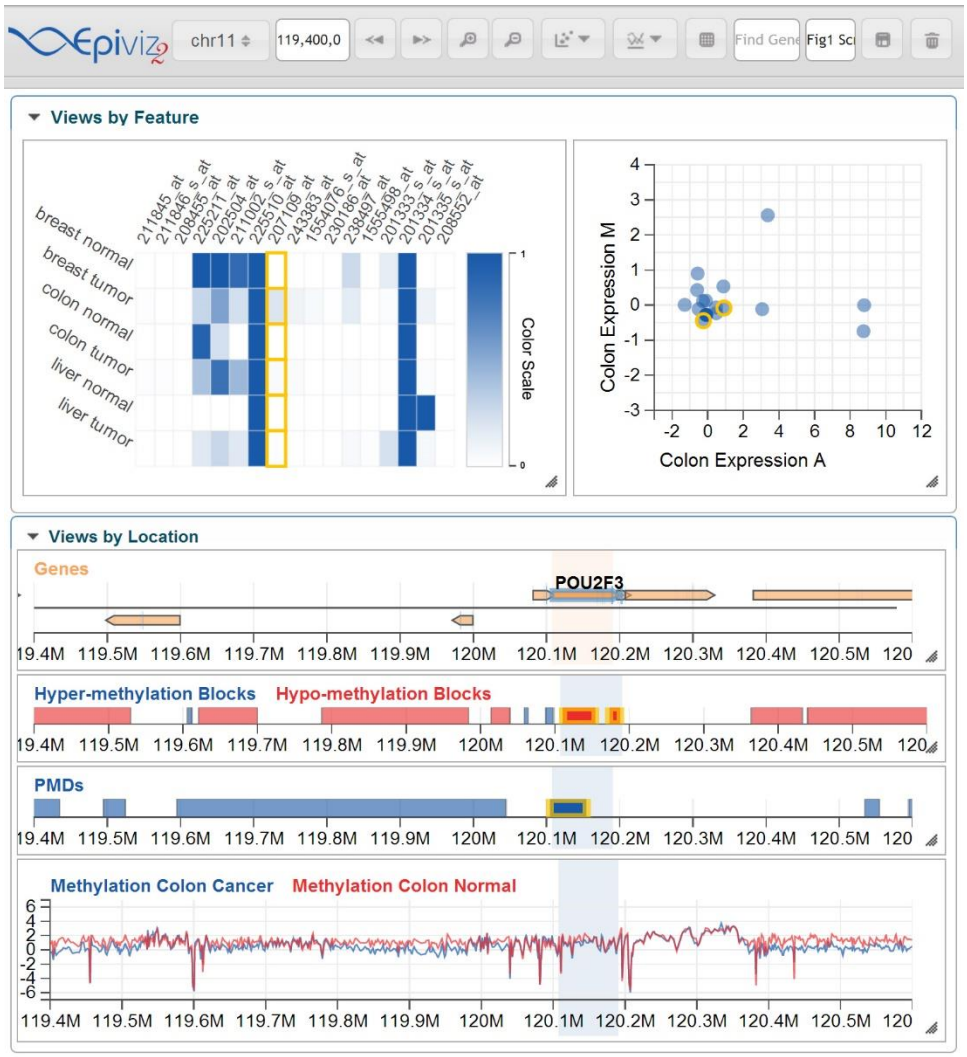


**Figure 28. Analysis of chromosome 11 colon cancer methylome.** Views by feature (top): gene expression across multiple tissues (top left). Each cell of the graph shows the degree of gene expression based on the Gene Expression Barcode project [42]. The highlighted region (yellow boxes) shows the brushing feature linking all charts by spatial location. Difference in gene expression between colon tumor and normal (M) versus average expression (A) for colon normal and colon tumor as an MA plot (top right) shows genes in view region that are differentially expressed. Views by location (bottom): UCSC genome browser gene models (genes), hypermethylation and hypomethylation blocks showing long regions of methylation difference in colon cancer [25], partially methylated domains in fibroblast (PMDs) [28], and methylation colon cancer and normal showing base-pair-resolution smoothed methylation log ratio from sequencing of bisulfite-converted DNA [25]. http://epiviz.cbcb.umd.edu/?ws=2AvhtzACF3p

We analyzed gene expression data from colon normal and tumor tissues and investigated the regulatory effects of large hypo-methylation regions in colon cancer. We transformed expression measurements to display an MA plot, using two

computed measurements. This approach revealed genes that exhibit differential expression within hypo-methylation blocks. Using the integrated gene expression barcode data we observed that expression patterns occurring in colon cancer within these long regions of DNA methylation loss also occurred in other cancer types, which is consistent with previous results [107].

## 5.3 Integration of Illumina HumanMethylation450k beadarray measurements and exon-level RNAseq expression data using Epivizr

We obtained Illumina HumanMethylation450k beadarray data for 34 colon tumor and 17 normal colon samples from the Cancer Genome Atlas project [57]. We then inferred long blocks of methylation difference in colon cancer using the *minfi* package in Bioconductor. We then used Epivizr to visually analyze the overlap of detected blocks in the TCGA samples using the 450k beadarray and the colon cancer blocks reported by Hansen et al. [25] using BSmooth [40].

We were able to discover some yet undocumented insights. We found that the 450k blocks displayed high overlap with sequencing blocks (**Figure 29**). The method used in *minfi* for the 450k array ignores methylation measurements in CpG islands by design, so that long blocks of methylation change would span across CpG islands. The algorithm in Hansen et al. did not use this design, so blocks are frequently punctuated by CpG islands. Using Epivizr we were able to confirm that the *minfi* procedure works as expected.

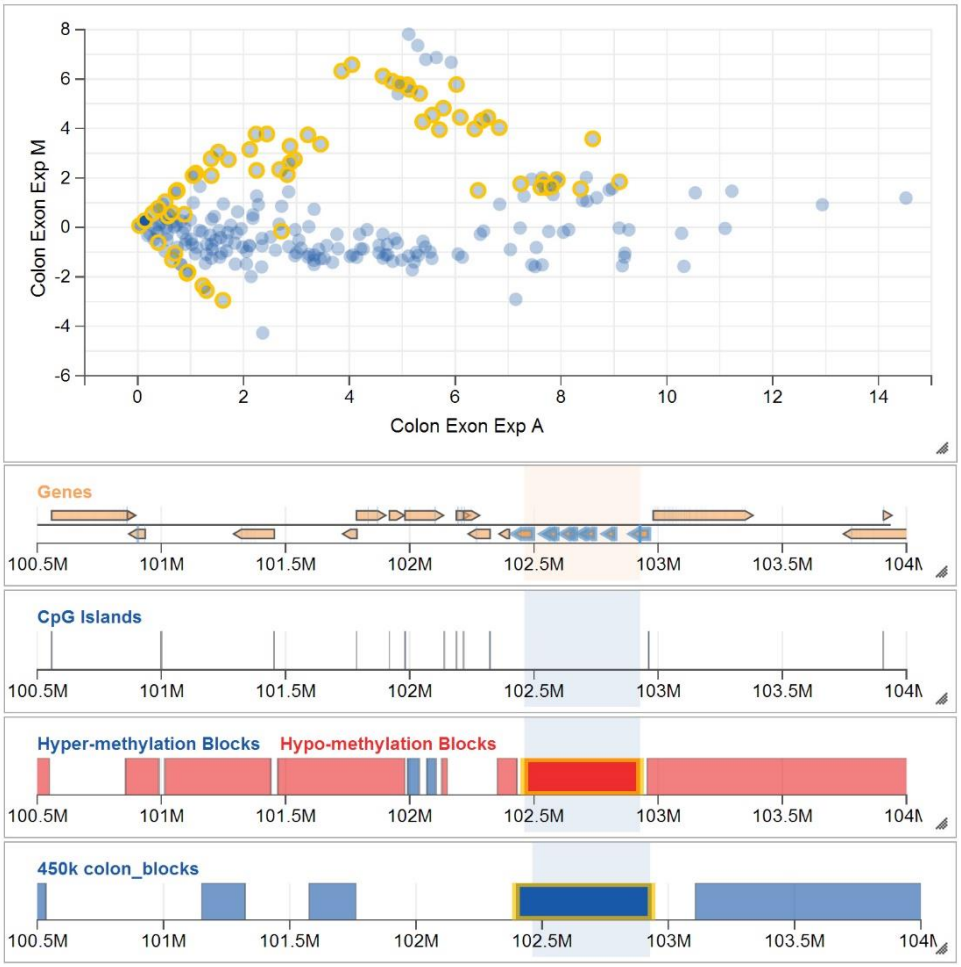


**Figure 29. Integrative analysis of Illumina HumanMethylation450k data and exon-level RNAseq data using Epivizr.** View by feature (top): Difference between colon tumor and normal exon-level expression (M) vs. average colon tumor and normal exon-level expression as an MA plot of RNA-seq data from the TCGA project. View by location (bottom): (Genes and CpG Islands) annotation tracks from UCSC genome browser. (Hyper-methylation Blocks, Hypo-methylation blocks) long regions of methylation difference obtained from sequencing data [25]. (450k colon_blocks) methylation difference regions obtained from TCGA data using the 450k array.

We next obtained exon-level RNAseq data from the Cancer Genome Atlas (TCGA) project [57] to perform an integrative analysis of the colon cancer methylome. RNAseq data can be referenced by genomic location (exon-level coverage, counting the number of fragments aligned to a specific exon), and by feature, e.g., transcript, or gene expression. This makes the multi-perspective

organization of Epiviz useful for this type of analyses. We designed and implemented an Epivizr data type where measurements are specifically indexed by exon, transcript or gene. In this case, we used UCSC exon identifiers for these features. We integrated this data from an R session through Epivizr. We then used the computed measurements facility on the Epiviz web application to again define an MA plot based on exon-level expression (**Figure 29**) and observe the association between higher expression in cancer, now at exon-level, and hypo-methylation blocks. Note that the MA transformation could also be applied on the R session, demonstrating the flexibility and power of a hybrid statistical analysis environment integrated with a modern, powerful visualization tool.

A standard workflow in these types of analyses is to identify regions of interest and navigate through them in a genome browser. The task of successively navigating through a set of specified regions, especially when they are the result of a statistical analysis, is not supported natively by any existing genome browser. However, since Epiviz is tightly integrated with R/Bioconductor interactive sessions, this is straightforward. The *slideshow* command in Epivizr allows users to successively navigate through an arbitrary set of genomic of regions of interest in the Epiviz browser. The vignette included in the Epivizr package includes an example of how this feature is used to navigate through regions containing exons that are significantly over-expressed in colon cancer.

## 5.4 Visualizing exon-level expression data using track-based visualization with Epiviz plugins

We further analyzed the correlation between exon-level expression and DNA methylation. Following the findings in Hansen et al. [25], we expected that exons in highly methylated regions would be silenced, and exons in hypo-methylated blocks would be differentially expressed. To support visualization for this analysis, we created a track-based visualization for continuous measurements of exon-level expression. We used the Epiviz plugin mechanism to include JavaScript files defining the d3.js visualization on the fly. Since Epiviz allows users to define custom visualizations and data types and immediately use them by providing a URL for the JavaScript code, we hosted our scripts on GitHub Gist, which define metadata and rendering code for the new exon-level expression visualization track.

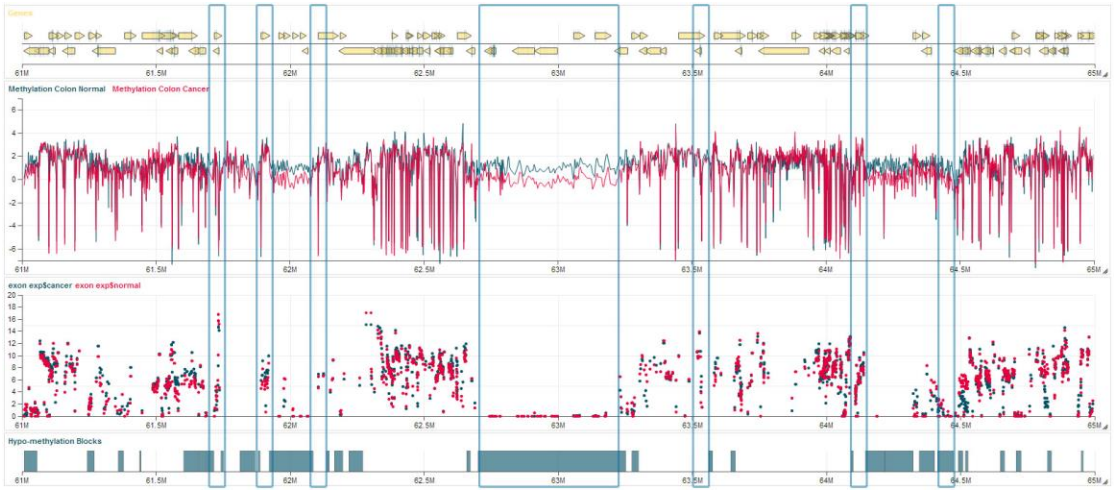


**Figure 30. Exon-level expression in differentially methylated regions.** The track-based visualization of exon-level expression data, side by side with a view of DNA methylation and one of differentially methylated blocks reveals that at low resolution, exons tend to be silenced within blocks, and highly expressed outside.

We analyzed the spatial level correlation of exon-level expression and base-pair level DNA methylation using this newly defined visualization for exon-level

expression as a track in Epiviz with the Epivizr package serving the expression data from an interactive *R* session (**Figure 31** and **Figure 30**). An overview visualization of the data confirmed the observation that hypo-methylated blocks are gene-poor [25]. The same view gave us another insight over the data, suggesting that exons in both normal and cancer tissues tend to be silenced within blocks, while the exons outside blocks tend to be expressed (**Figure 30**).

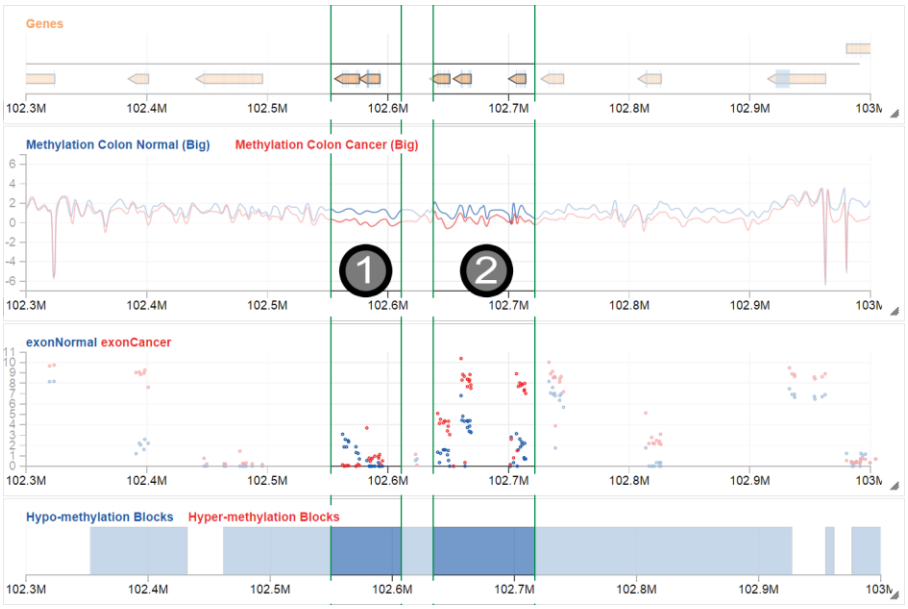


**Figure 31. Using Epiviz plugins to analyze base-pair resolution methylation data and exon-level expression data.** Using a track-based visualization of exon-level expression data we observe the relationship between base-pair resolution methylation and expression: the region marked 1) exhibits high methylation autocorrelation and small expression differences, while the region marked 2) shows low methylation autocorrelation and large expression differences. In particular, this analysis confirms that the methylome displays difference at distinct levels of genomic resolution (smooth vs. non-smooth methylation measurement) and that while gene expression tends to be silenced within long regions of methylation loss, gene expression differences coincide with high-resolution features of the methylome.

We next analyzed the data at high resolution, to obtain a granular understanding of the relationship between basepair-level methylation data and exon-level expression. (**Figure 31**). We were able to confirm our findings in **Section 5.1** where we used *gene expression* align well with our discoveries for *exon expression*:

although at block-level there is an overall difference between cancer and normal methylation, the degree of auto-correlation in the methylation data at basepair resolution varies within the block. Exons featuring high differences in expression tend to collocate with regions of low methylation auto-correlation while exons that are not differentially expressed collocate with regions of high methylation auto-correlation; the same observation we made for gene expression and constitutes a novel epigenetic finding. This analysis strengthens our belief that understanding the relationship between expression and methylation within long epigenetic domains requires that methylation data is analyzed at multiple genomic resolutions.

## 5.5 Interactive visual analysis of metagenomics data

The majority of metagenomic studies are exploratory in nature [87], [108]–[110]. There may not exist an established microbial community for the community of interest. As such, the majority of studies and papers typically use *stacked bar plots* to highlight the composition at a particular level of the tree [109], [111] (though in some cases, for example the vaginal microbiome where the *Lactobacillus* genus dominates these will be separated out at a lower level), *scatter plots* for PCA/MDS analyses [76], [110], [111], and *heatmaps* [108], [112] to cluster subsets of sample populations or OTUs. In this section we present two studies which illustrate usage scenarios and ways in which Epiviz and Metavizr can help in the process of exploring and analyzing metagenomic datasets and lead to the discovery of important scientific insights.

*The effect of diarrhea over the microbiome of young children from low-income countries*

In the first study, samples come from two 16S marker-gene surveys, where the number of reads is measured and annotated for a particular clustering within a given sample. The surveys consist of a series of gut 16S marker-gene targeted sequencing data comprising 484 healthy and 508 moderate-to-severe diarrheic samples from Mali, Bangladesh, Kenya and The Gambia as originally published in [109]. Sequencing and further information can also be found in [109]. In analyzing this dataset we first filtered to only OTUs present in at least 100 samples.

We generated a *stacked plot* showing proportional abundance of dominant genera in non-diarrhea Gambian *controls*. To do this, we used R to select the Gambian *control* subset of the samples, stored in an *MRexperiment* [95] object. The same object also contains information about the phylogeny of the microbial OTUs. The Metavizr package exposes functions that manipulate this information and adapt it to the Epiviz data provider protocol. We used the *metavizControl* function from this package to aggregate the OTUs at the *genus* level. Next, we used the *metavizStack* function to plot the abundance of the OTU groups in a stacked plot. Using the Epiviz *group by* feature within the stacked plot, we grouped samples together into 6-month groups (**Figure 32**). We also added an *icicle* visualization to explore the phylogenetic tree. To better examine the trends of different genera abundances over time, we applied a *basis* (*B-spline*) *interpolation* [113] to the stacked plot, as shown in **Figure 33**. We observed two trends in the resulted figure, corresponding to *Prevotella* and *Escherichia* respectively. The first shows a steady increase in

abundance over time, while the latter shows the opposite. This is consistent with the findings presented in [109].

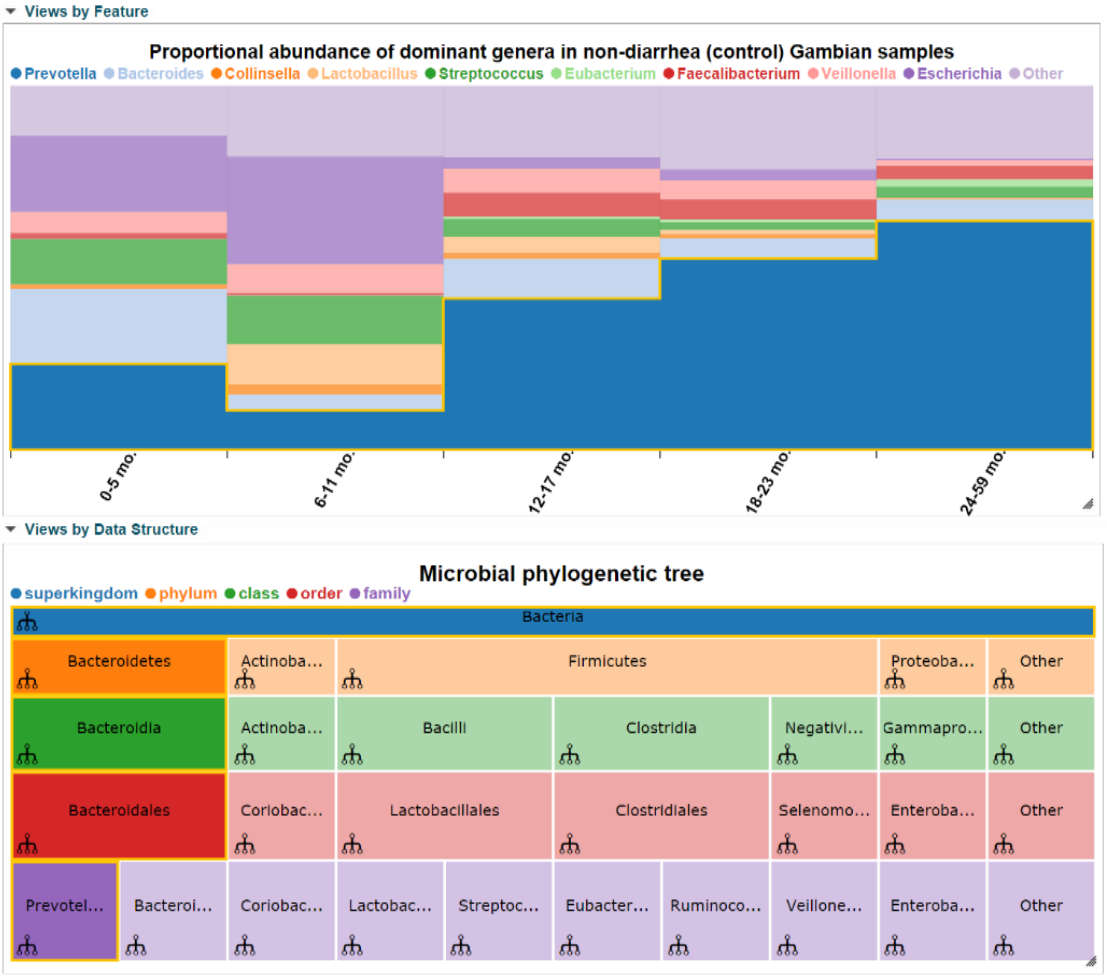


**Figure 32. Abundance of dominant genera in *control* samples.** The stacked plot shows proportional abundance of bacteria grouped by genera for different age groups. The highlighted series corresponds to *Prevotella*, which exhibits a visible trend to increase with age. In the icicle plot we see highlighted the path in the phylogenetic tree that leads to the *Prevotella* genus, due to the brushing/linking feature.

Epiviz allowed to further explore these results: using the brushing and linking feature, we were able to highlight the paths in the phylogenetic tree corresponding to each genera abundance trend (**Figure 32**, **Figure 33**).

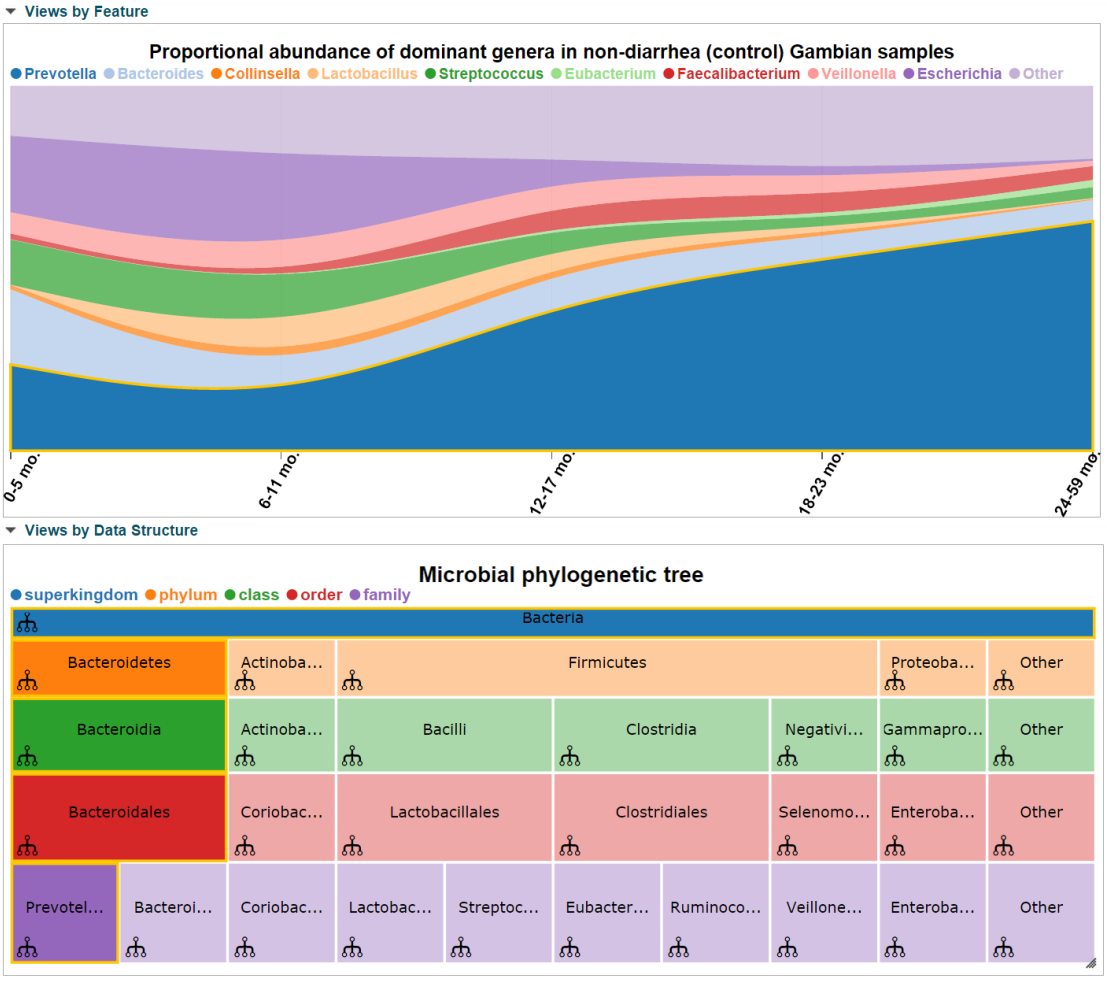


**Figure 33. Smoothed abundance of dominant genera in *control* samples.** The stacked plot shows the same data as that in **Figure 32**, but here the values are interpolated using basis (*B-spline*) interpolation. This display type makes trends in the data apparent, facilitating the formulation of hypotheses.

Also, using the *icicle* de-aggregation, we expanded the sub-tree rooted at *Prevotella* (**Figure 34**). We noticed that not all *Prevotella* species feature the same trend over time; for example, *Prevotella sp. DJF_B116* seems to maintain its abundance. Also, we were able to detect the species that dominates *Prevotella* abundances, determining its trend – *Prevotella sp. BI-42*.

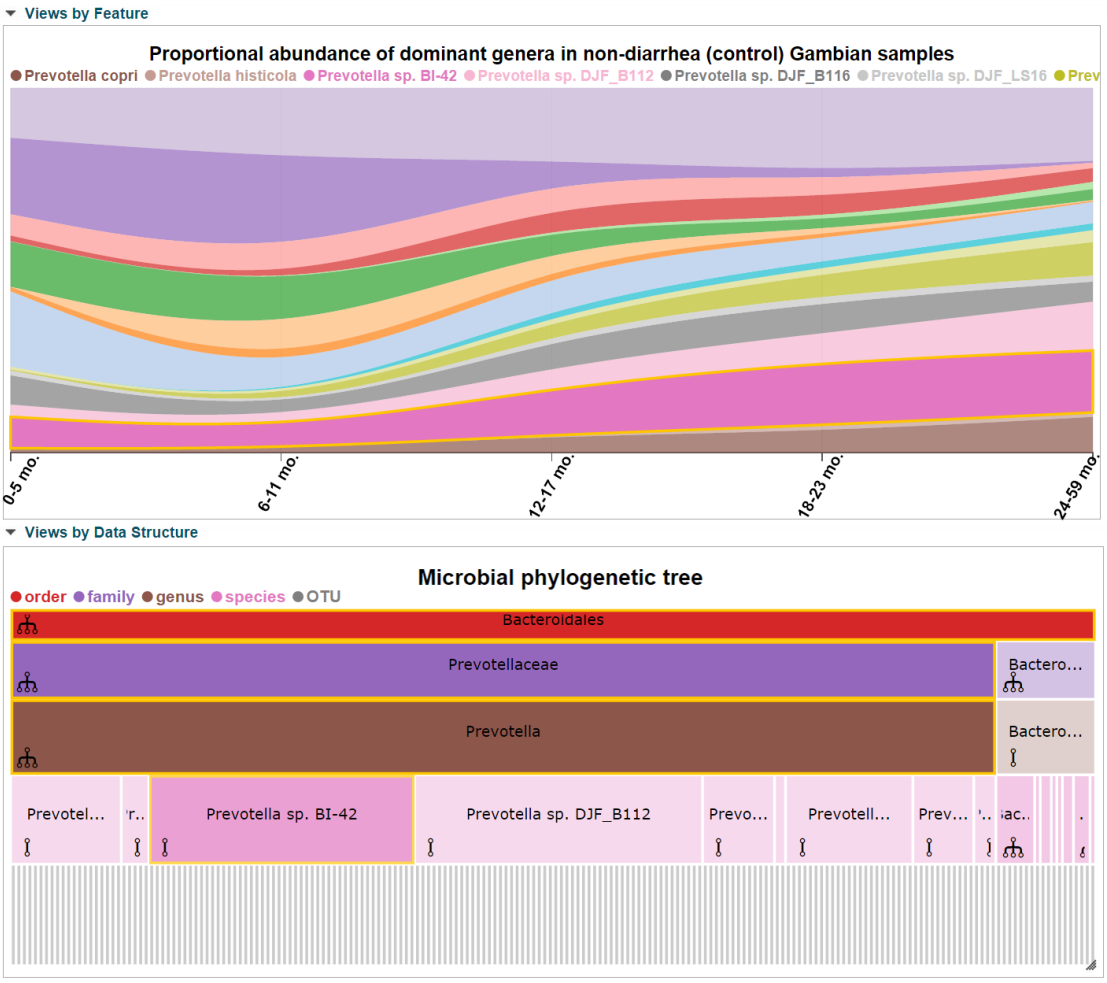


**Figure 34. Smoothed abundance of species belonging to *Prevotella* in *control* samples.** Here we changed the node selection for *Prevotella*, so its sub-nodes can be further analyzed. As a result, the stacked plot is now showing one data series for each species under *Prevotella*. We also navigated in the phylogenetic tree to *order* level. The highlighted path in the tree corresponds to the species *Prevotella sp. BI-42*, which appears to dominate the abundance of the *Prevotella* genus.

We took the study a step further, and plotted normalized [76] the abundance values for the 484 healthy samples from the infant gut cohort [109] in a *heatmap*. We plotted the 43 most variable OTUs, and used the Epiviz *heatmap hierarchical clustering* feature to re-order both samples and OTUs according to abundance similarity. The resulted clustering matches closely the five age range sample groups, due to the two genera observed earlier – *Prevotella* and *Escherichia*. Thus, samples

corresponding to young ages feature high abundances of *Escherichia*. The OTUs more abundant in the older age categories appear to belong to *Prevotella* (**Figure 35**).

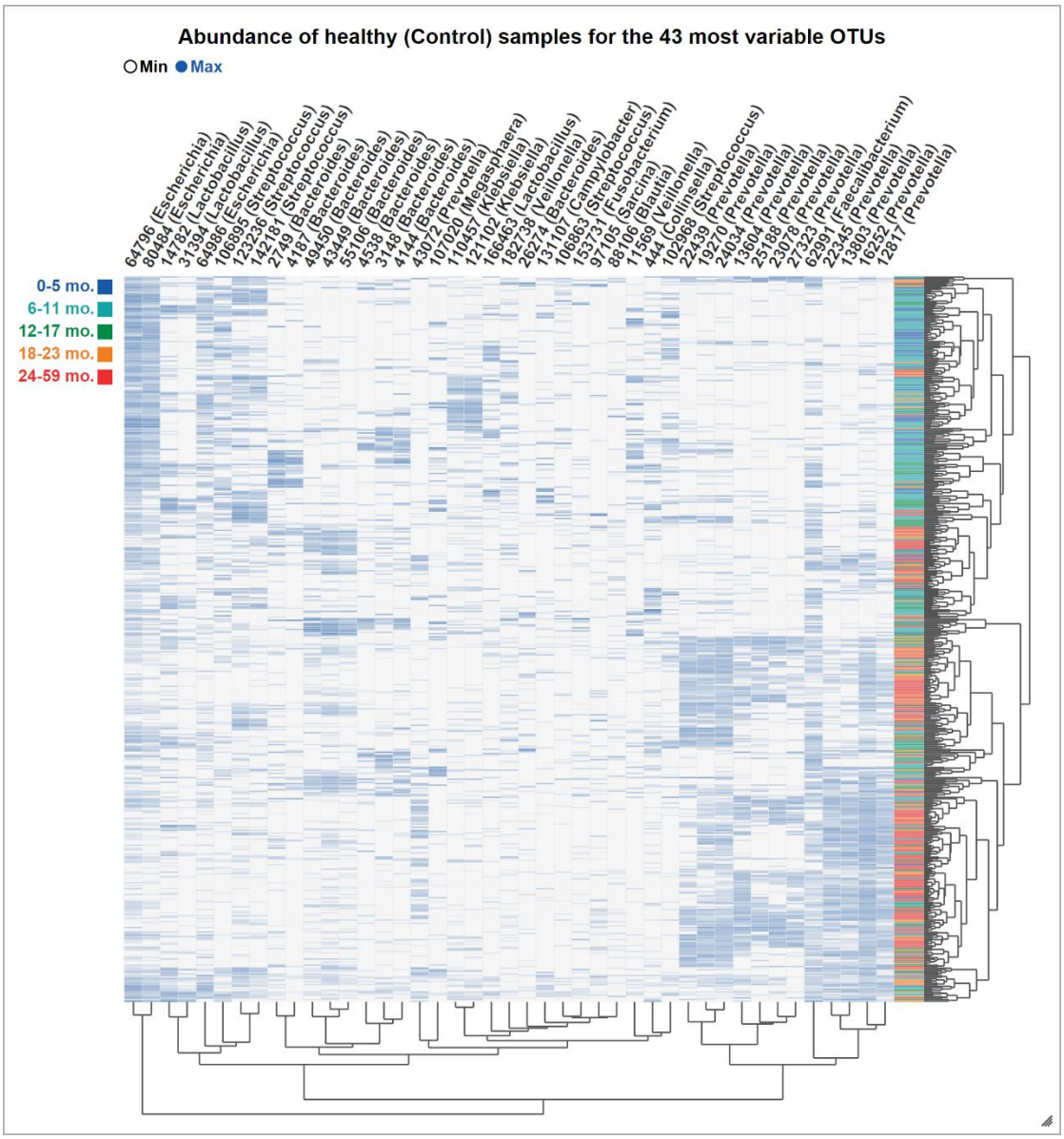


**Figure 35. Normalized abundance in *control* samples for the 43 most variable OTUs.** An Epiviz heatmap showing normalized abundance values for the 484 healthy samples from the infant gut cohort. Samples' abundance clustering matches age progression. The OTUs more abundant in the younger age categories appear to be *Escherichia* and *Bacteroides*.

Following these observations, we generated another view, depicted in **Figure 36**, where we grouped all samples by health status – *case* and *control* – as well as

age category. This view allowed us to visualize and compare the *cases* with the *controls* and extract additional insights. It confirmed that *Escherichia* abundances decreases with age, and *Prevotella* abundances increase. In addition, we noticed that in *cases*, *Escherichia* OTUs tend to be more abundant in older children, while *Prevotella* OTUs are more abundant in controls.

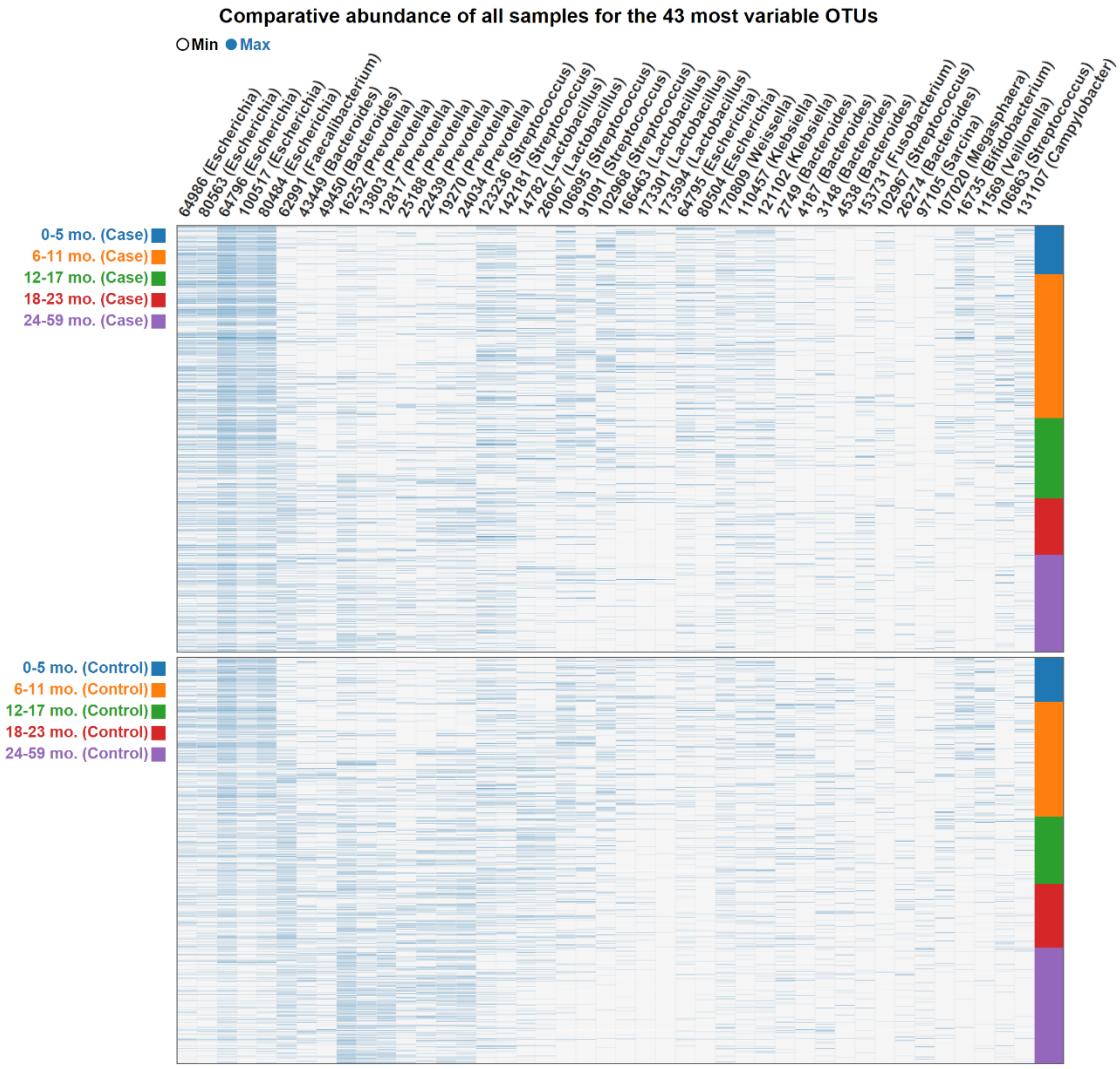


**Figure 36. Normalized abundance comparison of all *case*/*control* samples for the 43 most variable OTUs.** An Epiviz heatmap showing normalized abundance values for all 992 samples, both *case* and *control* from the infant gut cohort. The samples are grouped by age category and health status. This view allows us to compare *cases*/*controls* side by side, to observe differences, and also to observe progression of OTU abundances over time in both health categories.

*The effect of high fat and high sugar diet on the mouse gut microbiome*

The second study expands on experiments introduced in [76], [111]. It comprises twelve germ-free adult male C57BL/6J mice, which were fed a low-fat, plant polysaccharide-rich diet (LF/PP). Each mouse was gavaged with healthy adult human fecal material. Following the fecal transplant, mice remained on the low-fat, plant polysaccharide-rich diet for four weeks, following which a subset of 6 were switched to a high-fat and high-sugar (Western) diet for eight weeks. Fecal samples for each mouse went through PCR amplification of the bacterial 16S rRNA gene V2 region weekly. Details of experimental protocols and further details of the data can be found in [111].

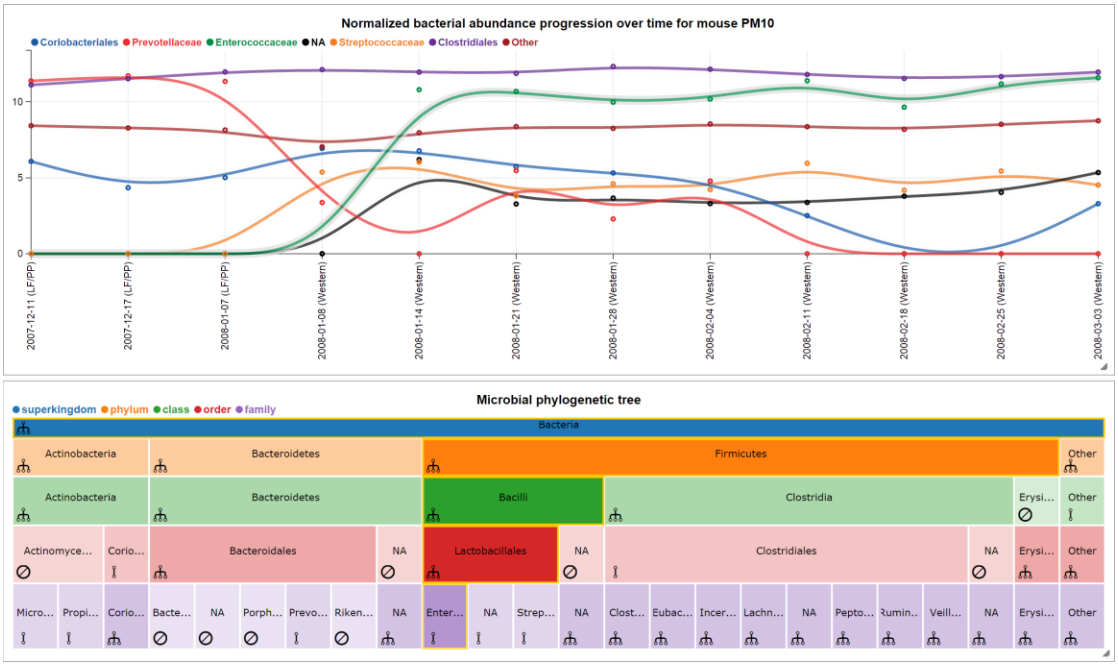


**Figure 37. Normalized bacterial abundance progression over time for mouse PM10.** We organized samples according to time and plotted one individual mouse's normalized bacterial abundance progression through time at different levels in the phylogenetic tree. The mouse, *PM10*, underwent a shift in diet following the third sampling. We can see a marked increase in the abundance of all families of *Bacilli* and a decrease in *Prevotellae*.

We organized samples according to time and used a lines plot to visualize one individual mouse's normalized bacterial abundance progression through time at the class level (**Figure 37**). The mouse, labeled *PM10* underwent a shift in diet following the third sampling. Using Metavizr, we generated a lines plot that displays the abundance of OTUs observed in this mouse over time. We used the icicle plot in Epiviz to customize the aggregation at different levels in the phylogenetic tree, hide groups of OTUs that expose no insight, and highlight the ones that do. In particular, we were interested in bacteria that would characterize the change in diet from LF/PP to Western. We noticed that the shift in diet was bound to an increase in the abundance of all the different *families* of *Bacilli*, in particular *Enterococcaceae*, and a decrease in both *Bacteroidetes*, especially *Prevotellaceae*, and *Actinobacteria*, especially *Coriobacteriales*. This is consistent with the insights presented in [111].

High-throughput sequencing data suffers from the curse of dimensionality. PCA or multidimensional scaling allows for a deeper understanding of the variability due to varying phenotypes or batches in a dataset. Most marker-gene survey studies include a PCA/PCoA(MDS) plot to highlight that the first few components separate samples in a meaningful manner following appropriate normalization [76], [95]. In our study, we plotted all 139 samples from the longitudinal mouse marker-gene survey, using two separate scatter plots. We used the *color by* Epiviz feature to color the data points by *diet*, in the first, and *time* in the second. For this part of the experiment, we organized the mouse experiment data in a hierarchy which we plotted using an Epiviz icicle visualization. For each mouse in the experiment, we have a corresponding sub-tree in the hierarchy. At each subsequent level in the tree, we

have a different experimental feature, such as diet, date, relative time, etc. (**Figure 38**). We noticed that MDS separates the samples into multiple groups, linking noticeably to diet, as well as time.

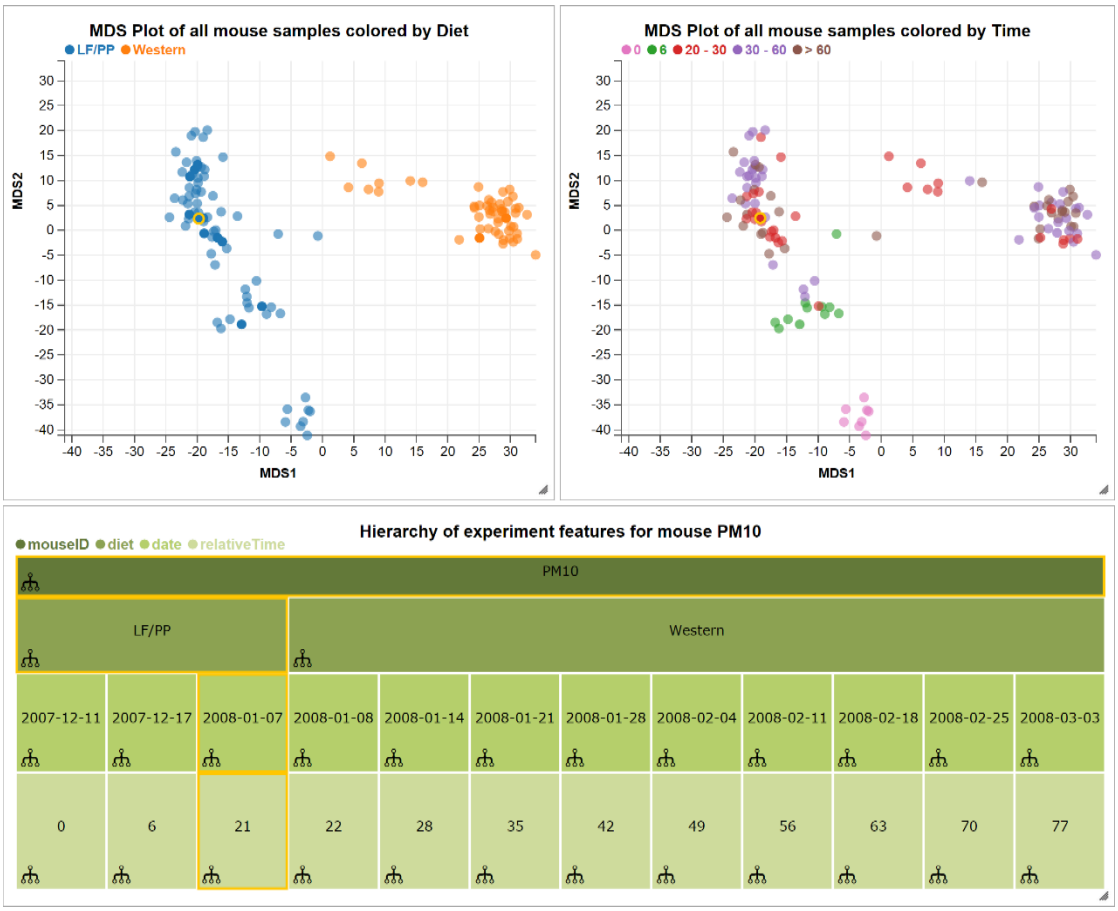


**Figure 38. MDS analysis of all 139 samples from the longitudinal mouse marker-gene survey.** We used two scatter plots in which we show the same data, but apply different *color by* transformations. We notice that MDS separates the samples into multiple groups, most noticeably by diet as well as time. In the left scatter plot, we notice that the first principal component groups correspond to different diets. On the right, we use a coloring scheme that highlights various time intervals, and groups corresponding to it. The icicle shows a hierarchy corresponding to experiment features, such as the mouse id, different diets and time.

# 6 Next steps

## 6.1 Performance limitations

There are a number of performance limitations brought by the design choices of Epiviz. Some reflect the target use cases we built the software for. Others however, are issues we mean to address in future releases of the tool. In this section we underline some which we find more important.

First of all, the design of Epiviz restricts all optimization decisions to those that can be done on the client side. This is because our software, apart from a small server component that stores user data and workspaces information, runs entirely on the client machine, and proposes to aggregate data from any number of external sources. As we have no control over these data sources, optimization on the server side is not an option. This puts any complex data optimization techniques such as those discussed in [114] out of the scope of our current work.

Choosing JavaScript as the main programming language for the Epiviz framework comes with a number of advantages, but also a few limitations. The main advantage is that JavaScript runs natively in all major web browsers, on all operating systems, requiring no special installation. In addition, being optimized for online applications, it has a number of useful features that make it convenient to access online data sources. The fact that it is an interpreted language that can evaluate strings of text into executable code is also an important feature, which in Epiviz we make great use of. Finally, over the years, all major browsers have taken steps in isolating JavaScript, so that it is natively prevented from making changes to the local file

system. This is essential for Epiviz in its intent to allow users incorporate third-party code to extend software functionality, while maintaining the software secure.

With this in mind, the main drawback of using JavaScript for Epiviz is its restriction to running into a single thread of execution. Although HTML5 introduces web workers [115] as an alternative to multi-threaded computing, in practice, because they are treated as separate processes and do not share the context of the main program, they are hard to use for optimizations closely related to the visualizations. For this consideration, operations on relatively large amounts of data tend to impact the performance of Epiviz and lower user experience. Workspaces with a large amount of visualizations, each showing a large number of data objects present latencies in responsiveness, as all objects in all visualizations are drawn using a single processing thread. There are three ways of addressing this issue, which we are looking to implement in our future work: 1) a clearer separation of data processing operations from visualizations, to better use with web workers; 2) the use of the WebGL technology, which comes with all the power of gaining direct access to the GPU [83]; 3) as the new technology WebCL [116] becomes available, and execution threads are introduced into the language, move some of the most CPU-consuming operations of Epiviz into secondary execution threads.

Another design decision that adds to the performance overhead is the use of vector graphics (SVG) for rendering visualizations. This is essential for some of the most important Epiviz features, such as brushing, tooltips, etc. At the same time, vectorial displays do not benefit from the limitation over the number of objects on the screen corresponding to the number of pixels available in the screen resolution,

which comes with raster images. This means that the actual number of objects rendered on the screen will be proportional to the number of data records behind the visualization. In other words, even when objects overlap, each of them is still rendered and has a corresponding element in the HTML document. This, combined with the limitation of single-thread processing in JavaScript, leads to performance hits when a lot of visualizations with many objects are in view. In addition, because of the diverse nature of visualizations, it is impossible to predict which records will correspond to overlapping visual objects at the visualization API level, which moves the burden of visual optimization to each individual chart. In **Section 3.8** we discussed a number of optimizations for individual charts, as well as their effects over render latencies. What is not said there is that these optimizations cannot, because of the design decisions of Epiviz, be generalized so that new visualizations can take advantage of them. A solution for this drawback, which we are looking at implementing in future versions of our software, is to alternate between raster views and vector views, depending on the amount of data loaded in memory. We anticipate this approach to significantly improve user experience for the visualization of extremely large datasets.

## 6.2 Future research directions in collaboration

Epiviz has taken a few preliminary steps in the direction of providing a collaboration-friendly environment. Two features in particular are aimed to help teams that work on a joint data analysis project: 1) *workspaces*, which provide a way of storing analysis steps persistently, and sharing them among the community;

2) an API that has support for custom software plugins created within the scientific community, stored on GitHub Gist, which can be used to share visualizations and data providers among users. In addition, as the entire code base of Epiviz is open-source, users from the scientific community are able to report errors or new ideas, as well as contribute to the development of the project. However, these represent only our preliminary efforts in a long-term plan of creating a truly collaborative data analysis environment and by no means do we consider them complete. In this section we briefly introduce the next three steps we mean to take in this direction, based on previous research on this topic [117] as well as related examples [72], [73] where approaches have proved to provide a high level of collaboration, enhancing usability, transparency and interaction among users.

A first step is to centralize *workspaces*, as well as custom user code used for both plugins and data transformations, in a webpage similar to those provided by IPython and RCloud for user notebooks. We intend to allow our users to browse existing *workspaces* and plugins, copy and extend them, as well as rank them using a starring system. The goal for this feature is to introduce transparency and awareness within the Epiviz community, allowing users to dynamically interact with each other and expand the usability of the framework.

The second step is to introduce *same-workspace concurrent collaboration*. The current functionality already allows the same Epiviz instance to pull data from multiple sources, of which several can be either R or Python sessions. These sessions can be located on any network-accessible machine. This opens the door to a peer-to-peer type of collaboration similar to that of Google Docs, where multiple users can

connect to the same data sources and computational environment sessions, and interact from different workstations over the same joint workspace simultaneously. Changes made to the workspace by one user will be immediately propagated and become visible to all other users connected to the same R session. The current architecture of Epivizr is one step away from permitting this kind of functionality. The only thing missing is an awareness component within the Epivizr package which will track the activity of each connected user, and propagate it to all the others. The peer-to-peer approach also implies that no centralized server would be necessary for this architecture, delegating that functionality to the R/Epivizr sessions instead.

A third step will be to introduce a mechanism for annotating findings directly in visualizations, and storing them persistently for collaborative work. For example, the conclusions presented in **Section 5.1** could be inserted as a comment for the genomic region *102.3M-103M*, in *chromosome 11*, and linked to the user workspace. The current design of the standard data format already allows this type of functionality – the *data source* structure corresponding to the annotated measurement would simply need a new *feature metadata* column, storing user annotations. On the visualization side, users could opt for either showing or hiding the annotations corresponding to one particular measurement and workspace. This will simplify collaborative projects and enhance reproducibility, allowing users to provide detailed explanation as to the findings, insights and conclusions related to a particular persistently saved analysis step (workspace).

# 7 Conclusions

In this thesis we have presented our research involving a new visual analytic tool for genomics, called Epiviz. Although our use case scenario examples are specific to genomics, the concepts introduced in our work are far from being domain-specific. Our work not only introduces a novel software, but also proposes directions towards the architecture of systems for exploratory and interactive visual and computational analysis. Traditional approaches treat data as a given static source of information. In contrast, a lot of our effort was focused on the idea of building a system around facilitating data integration while treating immediate, experimental data differently from static contextual data.

Epiviz puts together existing visualization and system ideas in original ways, of which some have not been tried before. For example, interactive visualization tools such as *Tableau* or *Spotfire* could benefit from interaction with computational environments like R or Python. Web-based analytic tools can use plugin mechanisms similar to that introduced by Epiviz to extend their functionality by incorporating third-party scripts and running them in sandbox mode through code sanitizing libraries. On the collaboration side, data analysis tools can extend the Epiviz *workspaces* concept, in which users are able to store persistently and share interactive views of analyses, including code customizations and visualizations. The lessons learned through the development of this tool, such as the performance implications of visualizing large amounts of data using SVG, or the improvements brought by adding a layer of caching on the UI also constitute good starting points for new endeavors in the same direction.

We gave an overview of the motivations that led to the development of Epiviz as well as a series of design decisions and features that have never been put together before in the field. Epiviz is the first genomic data analysis software that brings code to interactive visualization, bridging the gap between computational environments and genome browsers. The software also sets a precedent for genomic data analysis collaborative workflows by enabling reproducible and shareable steps, and allowing custom user code to be dynamically incorporated, while guaranteeing the security and integrity of user data.

We have described three ways in which Epiviz couples code and visualization. 1) First, our software exposes an API featuring different ways in which back-ends such as computational environments, as well as static web servers can feed data into the UI for a multiple view user interaction model. Epiviz is the first genomics visualization tool that can be used as a display device for R and Python, thus benefiting from all of their analytical features. 2) In addition, third-party JavaScript code stored externally, for example on GitHub Gist, is dynamically integrated into our software in a way that only affects the owners of the scripts. Custom visualizations or data providers all can be plugged into the software using this simple mechanism. 3) Epiviz completes the cycle by allowing users to customize the code of visualizations directly in the UI through controls that dive straight into the code of the framework. The same means are used to also expose simple data transformations, such as filtering, sorting, or creating new measurements as combinations of existing ones.

We also introduced the concept of community-contributed plugins for web applications through JavaScript dynamic extension. Epiviz is the first web-based

visualization tool whose code base can be extended by actively incorporating third-party scripts. One reason for the novelty of this feature is the *security concern* of allowing potentially malicious code into the application. We tackle this concern by isolating user code through a *sanitizing library*. Epiviz guarantees the integrity of user data by executing this code in *sandbox mode*. This feature constitutes a good starting point for collaboration in data analysis web applications in general, which can combine this feature with the concept of *workspaces* that can be persistently stored and shared among the scientific community.

In genomics, Epiviz is the first visualization tool that makes use of the following set of concepts and features *simultaneously*: brushing and linking, binning, supporting data transformations directly in the UI, introducing predictive caching on the UI level based on navigation, aggregating data from multiple sources – both cloud-based and local – and persistently saving and sharing data analysis steps as *workspaces*.

Finally, by coupling our visualization framework with the Epivizr and Metavizr R/Bioconductor packages, we created a statistical and visual analysis tool for *metagenomics*. This is the first tool to couple interactive visualization, multiple coordinated views with linking and brushing, navigation, as well as statistical analysis features to this branch of genomics. In addition, this serves as an example case study for other scientific fields with similar analytical and visualization needs.

# Bibliography


[1] F. Chelaru, L. Smith, N. Goldstein, and H. C. Bravo, "Epiviz: interactive visual analytics for functional genomics data," *Nat. Methods*, vol. 11, no. 9, pp. 938–940, Aug. 2014.

[2] H. C. Bravo, F. Chelaru, L. Smith, and N. Goldstein, "epivizr: R Interface to epiviz web app." .

[3] F. Chelaru and H. C. Bravo, "Epivizpy," 2014. [Online]. Available: http://github.com/epiviz/epivizpy.

[4] Z. Wang, M. Gerstein, and M. Snyder, "RNA-Seq: a revolutionary tool for transcriptomics.," *Nat. Rev. Genet.*, vol. 10, no. 1, pp. 57–63, Jan. 2009.

[5] L. Kong, J. Wang, S. Zhao, X. Gu, J. Luo, and G. Gao, "ABrowse--a customizable next-generation genome browser framework.," *BMC Bioinformatics*, vol. 13, no. 1, p. 2, Jan. 2012.

[6] T. R. Pak and F. P. Roth, "ChromoZoom: a flexible, fluid, web-based genome browser.," *Bioinformatics*, vol. 29, no. 3, pp. 384–6, Feb. 2013.

[7] I. Medina, F. Salavert, R. Sanchez, A. de Maria, R. Alonso, P. Escobar, M. Bleda, and J. Dopazo, "Genome Maps, a new generation genome browser.," *Nucleic Acids Res.*, vol. 41, no. Web Server issue, pp. W41–6, Jul. 2013.

[8] K. Arakawa, S. Tamaki, N. Kono, N. Kido, K. Ikegami, R. Ogawa, and M. Tomita, "Genome Projector: zoomable genome map with multiple views.," *BMC Bioinformatics*, vol. 10, no. 1, p. 31, Jan. 2009.

[9] R. Lister, R. C. O'Malley, J. Tonti-Filippini, B. D. Gregory, C. C. Berry, A. H. Millar, and J. R. Ecker, "Highly integrated single-base resolution maps of the epigenome in Arabidopsis.," *Cell*, vol. 133, no. 3, pp. 523–36, May 2008.

[10] M. E. Skinner, A. V Uzilov, L. D. Stein, C. J. Mungall, and I. H. Holmes, "JBrowse: a next-generation genome browser.," *Genome Res.*, vol. 19, no. 9, pp. 1630–8, Oct. 2009.

[11] M. Fiume, V. Williams, A. Brook, and M. Brudno, "Savant: genome browser for high-throughput sequencing data.," *Bioinformatics*, vol. 26, no. 16, pp. 1938–44, Aug. 2010.

[12] X. Zhou, B. Maricque, M. Xie, D. Li, V. Sundaram, E. A. Martin, B. C. Koebbe, C. Nielsen, M. Hirst, P. Farnham, R. M. Kuhn, J. Zhu, I. Smirnov, W. J. Kent, D. Haussler, P. A. F. Madden, J. F. Costello, and T. Wang, "The Human Epigenome Browser at Washington University.," *Nat. Methods*, vol. 8, no. 12, pp. 989–90, Dec. 2011.

[13] M. Goldman, B. Craft, T. Swatloski, K. Ellrott, M. Cline, M. Diekhans, S. Ma, C. Wilks, J. Stuart, D. Haussler, and J. Zhu, "The UCSC Cancer Genomics Browser: update 2013.," *Nucleic Acids Res.*, vol. 41, no. Database issue, pp. D949–54, Jan. 2013.

[14] J. Zhu, J. Z. Sanborn, S. Benz, C. Szeto, F. Hsu, R. M. Kuhn, D. Karolchik, J. Archie, M. E. Lenburg, L. J. Esserman, W. J. Kent, D. Haussler, and T. Wang, "The UCSC Cancer Genomics Browser.," *Nat. Methods*, vol. 6, no. 4, pp. 239–40, Apr. 2009.

[15] R. C. Gentleman, V. J. Carey, D. M. Bates, B. Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, J. Gentry, K. Hornik, T. Hothorn, W. Huber, S. Iacus, R. Irizarry, F. Leisch, C. Li, M. Maechler, A. J. Rossini, G. Sawitzki, C. Smith, G. Smyth, L. Tierney, J. Y. H. Yang, and J. Zhang, "Bioconductor: open software development for computational biology and bioinformatics.," *Genome Biol.*, vol. 5, no. 10, p. R80, Jan. 2004.

[16] M. Bostock, V. Ogievetsky, and J. Heer, "$D^3$: Data-Driven Documents.," *IEEE Trans. Vis. Comput. Graph.*, vol. 17, no. 12, pp. 2301–9, Dec. 2011.

[17] J. Heer and S. Kandel, "Interactive analysis of big data," *XRDS Crossroads, ACM Mag. Students*, vol. 19, no. 1, p. 50, Sep. 2012.

[18] S. Kandel, R. Parikh, A. Paepcke, J. M. Hellerstein, and J. Heer, "Profiler," in *Proceedings of the International Working Conference on Advanced Visual Interfaces - AVI '12*, 2012, p. 547.

[19] C. Stolte, D. Tang, and P. Hanrahan, "Polaris: a system for query, analysis, and visualization of multidimensional databases," *IEEE Trans. Vis. Comput. Graph.*, vol. 8, no. 1, pp. 52–65, Nov. 2002.

[20] B. E. Bernstein, E. Birney, I. Dunham, E. D. Green, C. Gunter, and M. Snyder, "An integrated encyclopedia of DNA elements in the human genome.," *Nature*, vol. 489, no. 7414, pp. 57–74, Sep. 2012.

[21] G. R. Abecasis, D. Altshuler, A. Auton, L. D. Brooks, R. M. Durbin, R. A. Gibbs, M. E. Hurles, and G. A. McVean, "A map of human genome variation from population-scale sequencing.," *Nature*, vol. 467, no. 7319, pp. 1061–73, Oct. 2010.

[22] B. E. Bernstein, J. A. Stamatoyannopoulos, J. F. Costello, B. Ren, A. Milosavljevic, A. Meissner, M. Kellis, M. A. Marra, A. L. Beaudet, J. R. Ecker, P. J. Farnham, M. Hirst, E. S. Lander, T. S. Mikkelsen, and J. A. Thomson, "The NIH Roadmap Epigenomics Mapping Consortium.," *Nat. Biotechnol.*, vol. 28, no. 10, pp. 1045–8, Oct. 2010.

[23] T. S. Furey, "ChIP-seq and beyond: new and improved methodologies to detect and characterize protein-DNA interactions.," *Nat. Rev. Genet.*, vol. 13, no. 12, pp. 840–52, Dec. 2012.

[24] R. A. Irizarry, C. Ladd-Acosta, B. Carvalho, H. Wu, S. A. Brandenburg, J. A. Jeddeloh, B. Wen, and A. P. Feinberg, "Comprehensive high-throughput arrays for relative methylation (CHARM).," *Genome Res.*, vol. 18, no. 5, pp. 780–90, May 2008.

[25] K. D. Hansen, W. Timp, H. C. Bravo, S. Sabunciyan, B. Langmead, O. G. McDonald, B. Wen, H. Wu, Y. Liu, D. Diep, E. Briem, K. Zhang, R. A. Irizarry, and A. P. Feinberg, "Increased methylation variation in epigenetic domains across cancer types.," *Nat. Genet.*, vol. 43, no. 8, pp. 768–75, Aug. 2011.

[26] R. D. Hawkins, G. C. Hon, L. K. Lee, Q. Ngo, R. Lister, M. Pelizzola, L. E. Edsall, S. Kuan, Y. Luu, S. Klugman, J. Antosiewicz-Bourget, Z. Ye, C. Espinoza, S. Agarwahl, L. Shen, V. Ruotti, W. Wang, R. Stewart, J. A. Thomson, J. R. Ecker, and B. Ren, "Distinct epigenomic landscapes of pluripotent and lineage-committed human cells.," *Cell Stem Cell*, vol. 6, no. 5, pp. 479–91, May 2010.

[27] R. Lister, M. Pelizzola, Y. S. Kida, R. D. Hawkins, J. R. Nery, G. Hon, J. Antosiewicz-Bourget, R. O'Malley, R. Castanon, S. Klugman, M. Downes, R. Yu, R. Stewart, B. Ren, J. A. Thomson, R. M. Evans, and J. R. Ecker,

"Hotspots of aberrant epigenomic reprogramming in human induced pluripotent stem cells.," *Nature*, vol. 471, no. 7336, pp. 68–73, Mar. 2011.

[28] R. Lister, M. Pelizzola, R. H. Dowen, R. D. Hawkins, G. Hon, J. Tonti-Filippini, J. R. Nery, L. Lee, Z. Ye, Q.-M. Ngo, L. Edsall, J. Antosiewicz-Bourget, R. Stewart, V. Ruotti, A. H. Millar, J. A. Thomson, B. Ren, and J. R. Ecker, "Human DNA methylomes at base resolution show widespread epigenomic differences.," *Nature*, vol. 462, no. 7271, pp. 315–22, Nov. 2009.

[29] A. Meissner, T. S. Mikkelsen, H. Gu, M. Wernig, J. Hanna, A. Sivachenko, X. Zhang, B. E. Bernstein, C. Nusbaum, D. B. Jaffe, A. Gnirke, R. Jaenisch, and E. S. Lander, "Genome-scale DNA methylation maps of pluripotent and differentiated cells.," *Nature*, vol. 454, no. 7205, pp. 766–70, Aug. 2008.

[30] T. S. Mikkelsen, M. Ku, D. B. Jaffe, B. Issac, E. Lieberman, G. Giannoukos, P. Alvarez, W. Brockman, T.-K. Kim, R. P. Koche, W. Lee, E. Mendenhall, A. O'Donovan, A. Presser, C. Russ, X. Xie, A. Meissner, M. Wernig, R. Jaenisch, C. Nusbaum, E. S. Lander, and B. E. Bernstein, "Genome-wide maps of chromatin state in pluripotent and lineage-committed cells.," *Nature*, vol. 448, no. 7153, pp. 553–60, Aug. 2007.

[31] M. Carlson and S. Arora, "A client for retrieving Bioconductor objects from AnnotationHub." [Online]. Available: http://www.bioconductor.org/packages/release/bioc/html/AnnotationHub.html. [Accessed: 20-Aug-2014].

[32] C. S. Ross-Innes, R. Stark, A. E. Teschendorff, K. A. Holmes, H. R. Ali, M. J. Dunning, G. D. Brown, O. Gojis, I. O. Ellis, A. R. Green, S. Ali, S.-F. Chin, C. Palmieri, C. Caldas, and J. S. Carroll, "Differential oestrogen receptor binding is associated with clinical outcome in breast cancer.," *Nature*, vol. 481, no. 7381, pp. 389–93, Jan. 2012.

[33] S. Anders and W. Huber, "Differential expression analysis for sequence count data.," *Genome Biol.*, vol. 11, no. 10, p. R106, Jan. 2010.

[34] S. Anders, A. Reyes, and W. Huber, "Detecting differential usage of exons from RNA-seq data.," *Genome Res.*, vol. 22, no. 10, pp. 2008–17, Oct. 2012.

[35] D. J. McCarthy, Y. Chen, and G. K. Smyth, "Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation.," *Nucleic Acids Res.*, vol. 40, no. 10, pp. 4288–97, May 2012.

[36] M. D. Robinson, D. J. McCarthy, and G. K. Smyth, "edgeR: a Bioconductor package for differential expression analysis of digital gene expression data.," *Bioinformatics*, vol. 26, no. 1, pp. 139–40, Jan. 2010.

[37] G. K. Smyth, "Linear models and empirical bayes methods for assessing differential expression in microarray experiments.," *Stat. Appl. Genet. Mol. Biol.*, vol. 3, p. Article3, Jan. 2004.

[38] C. W. Law, Y. Chen, W. Shi, and G. K. Smyth, "Voom: precision weights unlock linear model analysis tools for RNA-seq read counts.," *Genome Biol.*, vol. 15, no. 2, p. R29, Feb. 2014.

[39] C. Trapnell, A. Roberts, L. Goff, G. Pertea, D. Kim, D. R. Kelley, H. Pimentel, S. L. Salzberg, J. L. Rinn, and L. Pachter, "Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks.," *Nat. Protoc.*, vol. 7, no. 3, pp. 562–78, Mar. 2012.

[40] K. D. Hansen, B. Langmead, and R. A. Irizarry, "BSmooth: from whole genome bisulfite sequencing reads to differentially methylated regions.," *Genome Biol.*, vol. 13, no. 10, p. R83, Oct. 2012.

[41] M. J. Aryee, A. E. Jaffe, H. Corrada-Bravo, C. Ladd-Acosta, A. P. Feinberg, K. D. Hansen, and R. A. Irizarry, "Minfi: a flexible and comprehensive Bioconductor package for the analysis of Infinium DNA methylation microarrays.," *Bioinformatics*, vol. 30, no. 10, pp. 1363–9, May 2014.

[42] M. N. McCall, K. Uppal, H. A. Jaffee, M. J. Zilliox, and R. A. Irizarry, "The Gene Expression Barcode: leveraging public data repositories to begin cataloging the human and murine transcriptomes.," *Nucleic Acids Res.*, vol. 39, no. Database issue, pp. D1011–5, Jan. 2011.

[43] M. Miller, M. Samuel, B. Laurie, I. Awad, and M. Stay, "Caja: Safe active content in sanitized JavaScript," *Google white Pap.*, 2008.

[44] X. Zhou, B. Maricque, M. Xie, D. Li, V. Sundaram, E. A. Martin, B. C. Koebbe, C. Nielsen, M. Hirst, P. Farnham, R. M. Kuhn, J. Zhu, I. Smirnov,

W. J. Kent, D. Haussler, P. A. F. Madden, J. F. Costello, and T. Wang, "The Human Epigenome Browser at Washington University.," *Nat. Methods*, vol. 8, no. 12, pp. 989–90, Dec. 2011.

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

[46] J. Goecks, N. Coraor, A. Nekrutenko, and J. Taylor, "NGS analyses by visualization with Trackster.," *Nat. Biotechnol.*, vol. 30, no. 11, pp. 1036–9, Dec. 2012.

[47] J. Goecks, C. Eberhard, T. Too, A. Nekrutenko, and J. Taylor, "Web-based visual analysis for high-throughput genomics.," *BMC Genomics*, vol. 14, no. 1, p. 397, Jan. 2013.

[48] E. Cerami, J. Gao, U. Dogrusoz, B. E. Gross, S. O. Sumer, B. A. Aksoy, A. Jacobsen, C. J. Byrne, M. L. Heuer, E. Larsson, Y. Antipin, B. Reva, A. P. Goldberg, C. Sander, and N. Schultz, "The cBio cancer genomics portal: an open platform for exploring multidimensional cancer genomics data.," *Cancer Discov.*, vol. 2, no. 5, pp. 401–4, May 2012.

[49] E. D. Green and M. S. Guyer, "Charting a course for genomic medicine from base pairs to bedside.," *Nature*, vol. 470, no. 7333, pp. 204–13, Feb. 2011.

[50] "BD2K:NIH Big Data to Knowledge." [Online]. Available: http://bd2k.nih.gov/#sthash.z6k1PpND.dpbs. [Accessed: 20-Aug-2014].

[51] J. R. Harger and P. J. Crossno, "Comparison of Open Source Visual Analytics Toolkits," *Proc. SPIE-IS&T Electron. Imaging*, vol. 8294, pp. 389–400, Jan. 2012.

[52] L. Zhang, A. Stoffel, M. Behrisch, S. Mittelstadt, T. Schreck, R. Pompl, S. Weber, H. Last, and D. Keim, "Visual analytics for the big data era — A comparative review of state-of-the-art commercial systems," in *2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*, 2012, pp. 173–182.

[53] B. Wylie and J. Baumes, "A unified toolkit for information and scientific visualization," *Proc. SPIE*, vol. 7243, no. August, p. 72430H–72430H–16, 2009.

[54] C. Ahlberg, "Spotfire: an information exploration environment," *ACM SIGMOD Rec.*, vol. 25, no. 4, pp. 25–29, 1996.

[55] W. J. Kent, C. W. Sugnet, T. S. Furey, K. M. Roskin, T. H. Pringle, A. M. Zahler, and D. Haussler, "The human genome browser at UCSC.," *Genome Res.*, vol. 12, no. 6, pp. 996–1006, Jun. 2002.

[56] J. T. Robinson, H. Thorvaldsdóttir, W. Winckler, M. Guttman, E. S. Lander, G. Getz, and J. P. Mesirov, "Integrative genomics viewer.," *Nat. Biotechnol.*, vol. 29, no. 1, pp. 24–6, Jan. 2011.

[57] Cancer Genome Atlas Network, "Comprehensive molecular characterization of human colon and rectal cancer.," *Nature*, vol. 487, no. 7407, pp. 330–7, Jul. 2012.

[58] D. Karolchik, "The UCSC Genome Browser Database," *Nucleic Acids Res.*, vol. 31, no. 1, pp. 51–54, Jan. 2003.

[59] T. J. P. Hubbard, B. L. Aken, S. Ayling, B. Ballester, K. Beal, E. Bragin, S. Brent, Y. Chen, P. Clapham, L. Clarke, G. Coates, S. Fairley, S. Fitzgerald, J. Fernandez-Banet, L. Gordon, S. Graf, S. Haider, M. Hammond, R. Holland, K. Howe, A. Jenkinson, N. Johnson, A. Kahari, D. Keefe, S. Keenan, R. Kinsella, F. Kokocinski, E. Kulesha, D. Lawson, I. Longden, K. Megy, P. Meidl, B. Overduin, A. Parker, B. Pritchard, D. Rios, M. Schuster, G. Slater, D. Smedley, W. Spooner, G. Spudich, S. Trevanion, A. Vilella, J. Vogel, S. White, S. Wilder, A. Zadissa, E. Birney, F. Cunningham, V. Curwen, R. Durbin, X. M. Fernandez-Suarez, J. Herrero, A. Kasprzyk, G. Proctor, J. Smith, S. Searle, and P. Flicek, "Ensembl 2009.," *Nucleic Acids Res.*, vol. 37, no. Database issue, pp. D690–7, Jan. 2009.

[60] A. McKenna, M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytsky, K. Garimella, D. Altshuler, S. Gabriel, M. Daly, and M. A. DePristo, "The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data.," *Genome Res.*, vol. 20, no. 9, pp. 1297–303, Sep. 2010.

[61] D. Karolchik, R. M. Kuhn, R. Baertsch, G. P. Barber, H. Clawson, M. Diekhans, B. Giardine, R. A. Harte, A. S. Hinrichs, F. Hsu, K. M. Kober, W. Miller, J. S. Pedersen, A. Pohl, B. J. Raney, B. Rhead, K. R. Rosenbloom, K. E. Smith, M. Stanke, A. Thakkapallayil, H. Trumbower, T. Wang, A. S. Zweig, D. Haussler, and W. J. Kent, "The UCSC Genome Browser Database: 2008 update.," *Nucleic Acids Res.*, vol. 36, no. Database issue, pp. D773–9, Jan. 2008.

[62] P. A. Fujita, B. Rhead, A. S. Zweig, A. S. Hinrichs, D. Karolchik, M. S. Cline, M. Goldman, G. P. Barber, H. Clawson, A. Coelho, M. Diekhans, T. R. Dreszer, B. M. Giardine, R. A. Harte, J. Hillman-Jackson, F. Hsu, V. Kirkup, R. M. Kuhn, K. Learned, C. H. Li, L. R. Meyer, A. Pohl, B. J. Raney, K. R. Rosenbloom, K. E. Smith, D. Haussler, and W. J. Kent, "The UCSC Genome Browser database: update 2011.," *Nucleic Acids Res.*, vol. 39, no. Database issue, pp. D876–82, Jan. 2011.

[63] M. Clamp, "Ensembl 2002: accommodating comparative genomics," *Nucleic Acids Res.*, vol. 31, no. 1, pp. 38–42, Jan. 2003.

[64] The ENCODE Project Consortium, "The ENCODE (ENCyclopedia Of DNA Elements) Project.," *Science*, vol. 306, no. 5696, pp. 636–40, Oct. 2004.

[65] N. Siva, "1000 Genomes project.," *Nat. Biotechnol.*, vol. 26, no. 3, p. 256, Mar. 2008.

[66] S. Durinck, Y. Moreau, A. Kasprzyk, S. Davis, B. De Moor, A. Brazma, and W. Huber, "BioMart and Bioconductor: a powerful link between biological databases and microarray data analysis.," *Bioinformatics*, vol. 21, no. 16, pp. 3439–40, Aug. 2005.

[67] R. J. Kinsella, A. Kähäri, S. Haider, J. Zamora, G. Proctor, G. Spudich, J. Almeida-King, D. Staines, P. Derwent, A. Kerhornou, P. Kersey, and P. Flicek, "Ensembl BioMarts: a hub for data retrieval across taxonomic space.," *Database (Oxford).*, vol. 2011, no. 0, p. bar030, Jan. 2011.

[68] W. Eckerson, "Three Tier Client/Server Architecture: Achieving Scalability, Performance, and Efficiency in Client Server Applications," 1995.

[69] A. Perer and B. Shneiderman, "Integrating Statistics and Visualization for Exploratory Power: From Long-Term Case Studies to Design Guidelines," *IEEE Comput. Graph. Appl.*, vol. 29, no. 3, pp. 39–51, May 2009.

[70] B. Shneiderman, "The eyes have it: a task by data type taxonomy for information visualizations," in *Proceedings 1996 IEEE Symposium on Visual Languages*, pp. 336–343.

[71] J. S. Yi, Y. A. Kang, J. Stasko, J. Jacko, and J. A. J. Ji Soo Yi, Youn Ah Kang, John T. Stasko, "Toward a Deeper Understanding of the Role of Interaction in Information Visualization," *IEEE Trans. Vis. Comput. Graph.*, vol. 13, no. 6, pp. 1224–31, 2007.

[72] F. Pérez and B. E. Granger, "IPython: A system for interactive scientific computing," *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 21–29, 2007.

[73] "RCloud: A Social Coding Environment for the Big Data Era." [Online]. Available: http://attinnovationspace.com/innovation/story/a7795827. [Accessed: 11-Feb-2015].

[74] RStudio, "Shiny, A web application framework for R." [Online]. Available: http://shiny.rstudio.com/. [Accessed: 11-Feb-2015].

[75] M. Lawrence, W. Huber, H. Pagès, P. Aboyoun, M. Carlson, R. Gentleman, M. T. Morgan, and V. J. Carey, "Software for computing and annotating genomic ranges.," *PLoS Comput. Biol.*, vol. 9, no. 8, p. e1003118, Jan. 2013.

[76] J. N. Paulson, O. C. Stine, H. C. Bravo, and M. Pop, "Differential abundance analysis for microbial marker-gene surveys.," *Nat. Methods*, vol. 10, no. 12, pp. 1200–2, Dec. 2013.

[77] M. Morgan, S. Anders, M. Lawrence, P. Aboyoun, H. Pagès, and R. Gentleman, "ShortRead: a bioconductor package for input, quality assessment and exploration of high-throughput sequence data.," *Bioinformatics*, vol. 25, no. 19, pp. 2607–8, Oct. 2009.

[78] J. F. Allen, "Maintaining knowledge about temporal intervals," *Commun. ACM*, vol. 26, no. 11, pp. 832–843, Nov. 1983.

[79] "Yahoo! Finance RSS Feeds - YDN." [Online]. Available: https://developer.yahoo.com/finance/. [Accessed: 23-Aug-2014].

[80] S. Dudoit, Y. H. Yang, M. J. Callow, and T. P. Speed, "Statistical methods for identifying differentially expressed genes in replicated c{DNA} microarray experiments," *Stat. Sin.*, vol. 12, pp. 111–139, 2002.

[81] G. Liu, A. E. Loraine, R. Shigeta, M. Cline, J. Cheng, V. Valmeekam, S. Sun, D. Kulp, and M. A. Siani-Rose, "NetAffx: Affymetrix probesets and annotations.," *Nucleic Acids Res.*, vol. 31, no. 1, pp. 82–6, Jan. 2003.

[82] B. Leiba, "OAuth web authorization protocol," *IEEE Internet Comput.*, vol. 16, no. 1, pp. 74–77, 2012.

[83] Z. Liu, B. Jiang, and J. Heer, "ImMens: Real-time visual querying of big data," *Comput. Graph. Forum*, vol. 32, no. 3, pp. 421–430, 2013.

[84] N. Elmqvist and J.-D. Fekete, "Hierarchical aggregation for information visualization: overview, techniques, and design guidelines.," *IEEE Trans. Vis. Comput. Graph.*, vol. 16, no. 3, pp. 439–54, Jan. .

[85] Y. Nadji, P. Saxena, and D. Song, "Document structure integrity: A robust basis for cross-site scripting defense," *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2009.

[86] "PHP: Prepared statements and stored procedures - Manual." [Online]. Available: http://php.net/manual/en/pdo.prepared-statements.php. [Accessed: 12-Feb-2015].

[87] J. Handelsman, J. Tiedje, National Research Council (US) Committee on Metagenomics: Challenges and Functional, N. R. C. (Us) C. on M. C. and Functional, and N. R. C. (Us) C. on M. C. and Functional, *THE NEW SCIENCE OF METAGENOMICS: Revealing the Secrets of Our Microbial Planet*. 2007, p. 170.

[88] F. Meyer, D. Paarmann, M. D'Souza, R. Olson, E. M. Glass, M. Kubal, T. Paczian, A. Rodriguez, R. Stevens, A. Wilke, J. Wilkening, and R. A. Edwards, "The metagenomics RAST server - a public resource for the automatic phylogenetic and functional analysis of metagenomes.," *BMC Bioinformatics*, vol. 9, no. 1, p. 386, Jan. 2008.

[89] I. Pagani, K. Liolios, J. Jansson, I. M. A. Chen, T. Smirnova, B. Nosrat, V. M. Markowitz, and N. C. Kyrpides, "The Genomes OnLine Database (GOLD) v.4: Status of genomic and metagenomic projects and their associated metadata," *Nucleic Acids Res.*, vol. 40, 2012.

[90] T. P. Sneddon, P. Li, and S. C. Edmunds, "GigaDB: announcing the GigaScience database," *Gigascience*, vol. 1, no. 1, p. 11, 2012.

[91] S. Hunter, M. Corbett, H. Denise, M. Fraser, A. Gonzalez-Beltran, C. Hunter, P. Jones, R. Leinonen, C. McAnulla, E. Maguire, J. Maslen, A. Mitchell, G. Nuka, A. Oisel, S. Pesseat, R. Radhakrishnan, P. Rocca-Serra, M. Scheremetjew, P. Sterk, D. Vaughan, G. Cochrane, D. Field, and S.-A. Sansone, "EBI metagenomics--a new resource for the analysis and archiving of metagenomic data.," *Nucleic Acids Res.*, vol. 42, no. Database issue, pp. D600–6, Jan. 2014.

[92] H. M. Bik and P. Interactive, "Phinch: An interactive, exploratory data visualization framework for -Omic datasets," Cold Spring Harbor Labs Journals, Oct. 2014.

[93] S. M. Huse, D. B. Mark Welch, A. Voorhis, A. Shipunova, H. G. Morrison, A. M. Eren, and M. L. Sogin, "VAMPS: a website for visualization and analysis of microbial population structures.," *BMC Bioinformatics*, vol. 15, no. 1, p. 41, Jan. 2014.

[94] B. D. Ondov, N. H. Bergman, and A. M. Phillippy, "Interactive metagenomic visualization in a Web browser.," *BMC Bioinformatics*, vol. 12, no. 1, p. 385, Jan. 2011.

[95] J. N. Paulson, H. Talukder, M. Pop, and H. C. Bravo, "metagenomeSeq: Statistical analysis for sparse high-throughput sequencing." Bioconductor, 2014.

[96] M. Ghodsi, B. Liu, and M. Pop, "DNACLUST: accurate and efficient clustering of phylogenetic marker genes.," *BMC Bioinformatics*, vol. 12, no. 1, p. 271, 2011.

[97] B. L. Maidak, G. J. Olsen, N. Larsen, R. Overbeek, M. J. McCaughey, and C. R. Woese, "The RDP (Ribosomal Database Project).," *Nucleic Acids Res.*, vol. 25, no. 1, pp. 109–111, 1997.

[98] B. Shneiderman, "Tree visualization with tree-maps: 2-d space-filling approach," *ACM Trans. Graph.*, vol. 11, no. 1, pp. 92–99, Jan. 1992.

[99] B. S. Johnson, "Treemaps: visualizing hierarchical and categorical data," Jan. 1993.

[100] B. Shneiderman and M. Wattenberg, "Ordered Treemap Layouts," p. 73, Oct. 2001.

[101] B. B. Bederson, B. Shneiderman, and M. Wattenberg, "Ordered and quantum treemaps: Making effective use of 2D space to display hierarchies," *ACM Trans. Graph.*, vol. 21, no. 4, pp. 833–854, Oct. 2002.

[102] J. Stasko and E. Zhang, "Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations," in *IEEE Symposium on Information Visualization 2000. INFOVIS 2000. Proceedings*, 2000, pp. 57–65.

[103] I. Letunic and P. Bork, "Interactive Tree Of Life (iTOL): an online tool for phylogenetic tree display and annotation.," *Bioinformatics*, vol. 23, no. 1, pp. 127–8, Jan. 2007.

[104] B. Song, X. Su, J. Xu, and K. Ning, "MetaSee: an interactive and extendable visualization toolbox for metagenomic sample analysis and comparison.," *PLoS One*, vol. 7, no. 11, p. e48998, Jan. 2012.

[105] J. B. Kruskal and J. M. Landwehr, "Icicle Plots: Better Displays for Hierarchical Clustering," *Am. Stat.*, vol. 37, no. 2, pp. 162–168, May 1983.

[106] A. Taylor, K. McLeod, C. Armit, R. Baldock, and A. Burger, "Visualization of gene expression information within the context of the mouse anatomy," Jul. 2014.

[107] H. C. Bravo, V. Pihur, M. McCall, R. A. Irizarry, and J. T. Leek, "Gene expression anti-profiles as a basis for accurate universal cancer signatures.," *BMC Bioinformatics*, vol. 13, no. 1, p. 272, Jan. 2012.

[108] F. H. Karlsson, V. Tremaroli, I. Nookaew, G. Bergström, C. J. Behre, B. Fagerberg, J. Nielsen, and F. Bäckhed, "Gut metagenome in European

women with normal, impaired and diabetic glucose control.," *Nature*, vol. 498, pp. 99–103, 2013.

[109] M. Pop, A. W. Walker, J. Paulson, B. Lindsay, M. Antonio, M. A. Hossain, J. Oundo, B. Tamboura, V. Mai, I. Astrovskaya, H. Corrada Bravo, R. Rance, M. Stares, M. M. Levine, S. Panchalingam, K. Kotloff, U. N. Ikumapayi, C. Ebruke, M. Adeyemi, D. Ahmed, F. Ahmed, M. T. Alam, R. Amin, S. Siddiqui, J. B. Ochieng, E. Ouma, J. Juma, E. Mailu, R. Omore, J. G. Morris, R. F. Breiman, D. Saha, J. Parkhill, J. P. Nataro, O. C. Stine, H. Bravo, R. Rance, M. Stares, M. M. Levine, S. Panchalingam, K. Kotloff, U. N. Ikumapayi, C. Ebruke, M. Adeyemi, D. Ahmed, F. Ahmed, M. T. Alam, R. Amin, S. Siddiqui, J. B. Ochieng, E. Ouma, J. Juma, E. Mailu, R. Omore, J. G. Morris, R. F. Breiman, D. Saha, J. Parkhill, J. P. Nataro, and O. C. Stine, "Diarrhea in young children from low-income countries leads to large-scale alterations in intestinal microbiota composition.," *Genome Biol.*, vol. 15, no. 6, p. R76, Jan. 2014.

[110] J. Ravel, P. Gajer, Z. Abdo, G. M. Schneider, S. S. K. Koenig, S. L. McCulle, S. Karlebach, R. Gorle, J. Russell, C. O. Tacket, R. M. Brotman, C. C. Davis, K. Ault, L. Peralta, and L. J. Forney, "Vaginal microbiome of reproductive-age women.," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 108 Suppl, pp. 4680–4687, Mar. 2011.

[111] P. J. Turnbaugh, V. K. Ridaura, J. J. Faith, F. E. Rey, R. Knight, and J. I. Gordon, "The effect of diet on the human gut microbiome: a metagenomic analysis in humanized gnotobiotic mice.," *Sci. Transl. Med.*, vol. 1, no. 39857, p. 6ra14, 2009.

[112] B. Liu, L. L. Faller, N. Klitgord, V. Mazumdar, M. Ghodsi, D. D. Sommer, T. R. Gibbons, T. J. Treangen, Y. C. Chang, S. Li, O. C. Stine, H. Hasturk, S. Kasif, D. Segrè, M. Pop, and S. Amar, "Deep sequencing of the oral microbiome reveals signatures of periodontal disease," *PLoS One*, vol. 7, no. 6, 2012.

[113] M. Unser, A. Aldroubi, and M. Eden, "Fast B-spline transforms for continuous image representation and interpolation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1991. [Online]. Available: http://www.computer.org/csdl/trans/tp/1991/03/i0277.pdf. [Accessed: 28-Mar-2015].

[114] L. Lins, J. T. Klosowski, and C. Scheidegger, "Nanocubes for real-time exploration of spatiotemporal datasets," *IEEE Trans. Vis. Comput. Graph.*, vol. 19, no. August, pp. 2456–2465, 2013.

[115] "HTML5 Web Workers." [Online]. Available: http://www.w3schools.com/html/html5_webworkers.asp. [Accessed: 12-Feb-2015].

[116] "WebCL - Heterogeneous parallel computing in HTML5 web browsers." [Online]. Available: https://www.khronos.org/webcl/. [Accessed: 12-Feb-2015].

[117] J. Heer and M. Agrawala, "Design considerations for collaborative visual analytics," *VAST IEEE Symp. Vis. Anal. Sci. Technol. 2007, Proc.*, pp. 171–178, 2007.

[118] H. Li, "Using the BioSeqClass Package," 2014. [Online]. Available: http://www.bioconductor.org/packages/release/bioc/vignettes/BioSeqClass/inst/doc/BioSeqClass.pdf. [Accessed: 16-Oct-2014].

[119] C. Bartenhagen, "RDRToolbox: A package for nonlinear dimension reduction with Isomap and," 2014. [Online]. Available: http://www.bioconductor.org/packages/release/bioc/vignettes/RDRToolbox/inst/doc/vignette.pdf. [Accessed: 16-Oct-2014].

[120] Y. H. Yang and S. Dudoit, "Normalization: Bioconductor's marray package," 2014. [Online]. Available: http://www.bioconductor.org/packages/release/bioc/vignettes/marray/inst/doc/marrayNorm.pdf. [Accessed: 16-Oct-2014].

[121] P. Aboyoun, H. Pages, and M. Lawrence, "Representation and Manipulation of Genomic Intervals," 2014. [Online]. Available: http://bioconductor.org/packages/release/bioc/manuals/GenomicRanges/man/GenomicRanges.pdf. [Accessed: 16-Oct-2014].

[122] S. Anders, P. T. Pyl, and W. Huber, "HTSeq - A Python framework to work with high-throughput sequencing data.," *Bioinformatics*, Sep. 2014.

[123] F. Pedregosa, R. Weiss, and M. Brucher, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[124] I. Baber, J. P. Tamby, N. C. Manoukis, D. Sangaré, S. Doumbia, S. F. Traoré, M. S. Maiga, and D. Dembélé, "A python module to normalize microarray data by the quantile adjustment method.," *Infect. Genet. Evol.*, vol. 11, no. 4, pp. 765–8, Jun. 2011.