Title

GIVE: portable genome browsers for personal websites

Permalink

https://escholarship.org/uc/item/3xd465br

Journal

Genome Biology, 19(1)

ISSN

1474-760X

Authors

Cao, Xiaoyi
Yan, Zhangming
Wu, Qiuyang
et al.

Publication Date

2018-12-01

DOI

10.1186/s13059-018-1465-6

Peer reviewed

SOFTWARE

Open Access

GIVE: portable genome browsers for personal websites

Xiaoyi Cao, Zhangming Yan, Qiuyang Wu, Alvin Zheng and Sheng Zhong[*]

Abstract

Growing popularity and diversity of genomic data demand portable and versatile genome browsers. Here, we present an open source programming library called GIVE that facilitates the creation of personalized genome browsers without requiring a system administrator. By inserting HTML tags, one can add to a personal webpage interactive visualization of multiple types of genomics data, including genome annotation, "linear" quantitative data, and genome interaction data. GIVE includes a graphical interface called HUG (HTML Universal Generator) that automatically generates HTML code for displaying user chosen data, which can be copy-pasted into user's personal website or saved and shared with collaborators. GIVE is available at: https://www.givengine.org/.

Background

Genomics data have become increasingly popular and diverse, posing new challenges to personalized data management and visualization [1–4]. On the one hand, people interested in making their genomic data public required "researchers and policymakers [to anticipate] when people share their genome on Facebook" [5]. This movement asks for development of portable, versatile, and easily deployable genome browsers. Ideally, a portable data visualization tool can work like a Google map that can be inserted into personal websites. On the other hand, new data types, especially those representing genome-wide interactions—including genome-interaction data (Hi-C [6], ChIA-PET [7]), transcriptome-genome interaction data (MARGI [8], GRID-seq [9]), and transcriptome interaction data (PARIS [10], MARIO [11], LIGR-seq [12], SPLASH [13])—require compatible visualization tools; ideally, it should be possible to seamlessly display these data in parallel with other data types including RNA sequencing (RNA-seq) [14], chromatin immunoprecipitation sequencing (ChIP-seq) [15], and Assay for Transposase-Accessible Chromatin using sequencing (ATAC-seq) [16].

It was envisioned that future genome browsers could work like Google Maps, of which users with small efforts can insert a customized version into their own websites [17]. Redeployable genome browsers are developed toward this goal [17–20]. Still, releasing websites with interactive visualization of genomic data would generally require systems administration, database, and web programming work. The GIVE project is aimed to automate this work and offer a portable and lightweight genome browser with complementary advantages of genome browser websites [1, 2], desktop executables [21], and personal homepages and blogs.

We created the open source GIVE programming library to meet the diverse needs of users with various levels of sophistication. A feature called GIVE HUG (HTML Universal Generator) provides a graphical interface to interactively generate HTML codes for displaying user chosen datasets. Users can save and share the HTML file with collaborators or copy-paste the HTML codes into their websites, which would lead to embedded interactive data display. Users can use GIVE to create custom genome browsers without hosting a data server, where all the data are retrieved on-demand from public data servers. Users who choose to host data on their own server can do so with commands provided in GIVE-Toolbox. With a few lines of HTML code, GIVE enables a website to retrieve, integrate, and display diverse data types hosted by multiple servers, including large public depositories and custom-built servers. Such simplicity of use comes from encapsulation of new data management, communication, and visualization technologies made available by the GIVE development team. The cores of these technologies are new data structures and a memory management algorithm.

* Correspondence: szhong@ucsd.edu
Department of Bioengineering, University of California San Diego, La Jolla, CA 92093, USA

Cao *et al. Genome Biology* (2018) 19:92

Page 2 of 8

Results

Overview of the GIVE library

GIVE is composed of an HTML tag library and GIVE-Toolbox. The former is a library of HTML tags for data visualization. GIVE-Toolbox is a set of command line commands, which automates all necessary database operations. For any public datasets for which the metadata can be found in GIVE data hub, users can directly use GIVE's HTML tags to display such data, without invoking GIVE-Toolbox.

GIVE's HTML tag library provides flexibility to build a variety of genome browsers, for example a single-cell transcriptome website [22] (https://singlecell.givengine.org/), an epigenome website [23, 24] (https://encode.givengine.org/, Additional file 1: Figure S1), a genome interaction website [25] (https://mcf7.givengine.org/, Fig. 1), and an RNA-chromatin interaction website [26] (https://margi.givengine.org/, Fig. 2). With GIVE, users can build data visualization websites without hosting actual data (data are hosted on public data servers), data hosting websites, or websites that display composite datasets hosted on user servers and public servers. The GIVE-enabled HTML files can also be used and shared as custom software, which encapsulate both data and visualization capability.

Automatic webpage generation with GIVE HUG

GIVE data hub and its embedded feature HUG enable automatic generation of interactive visualization webpages for user chosen datasets. GIVE data hub is a web page for browsing the metadata of genomic datasets hosted on public data servers (Additional file 1: Figure S2). Inside this web page is a database of metadata, including data type, data description, and the web address of the actual dataset. All metadata in GIVE data hub are validated by the GIVE development team to ensure correctness of information. Users are welcome to submit metadata of additional datasets hosted on public data servers through an online metadata submission form.

HUG automatically generates HTML webpages for any user chosen datasets. To use HUG, users can click "HTML Generator Mode" in data hub website (Additional file 1: Figure S2), select any datasets, and click the "Generate" button (Additional file 1: Figure S3). A separate window will pop up that summarizes the user chosen datasets and provide the generated HTML code (Additional file 1: Figure S4). Like Google Maps, this data-containing genome visualization HTML code can be copy-pasted into a personal website or saved and shared. Users can interactively change a few display
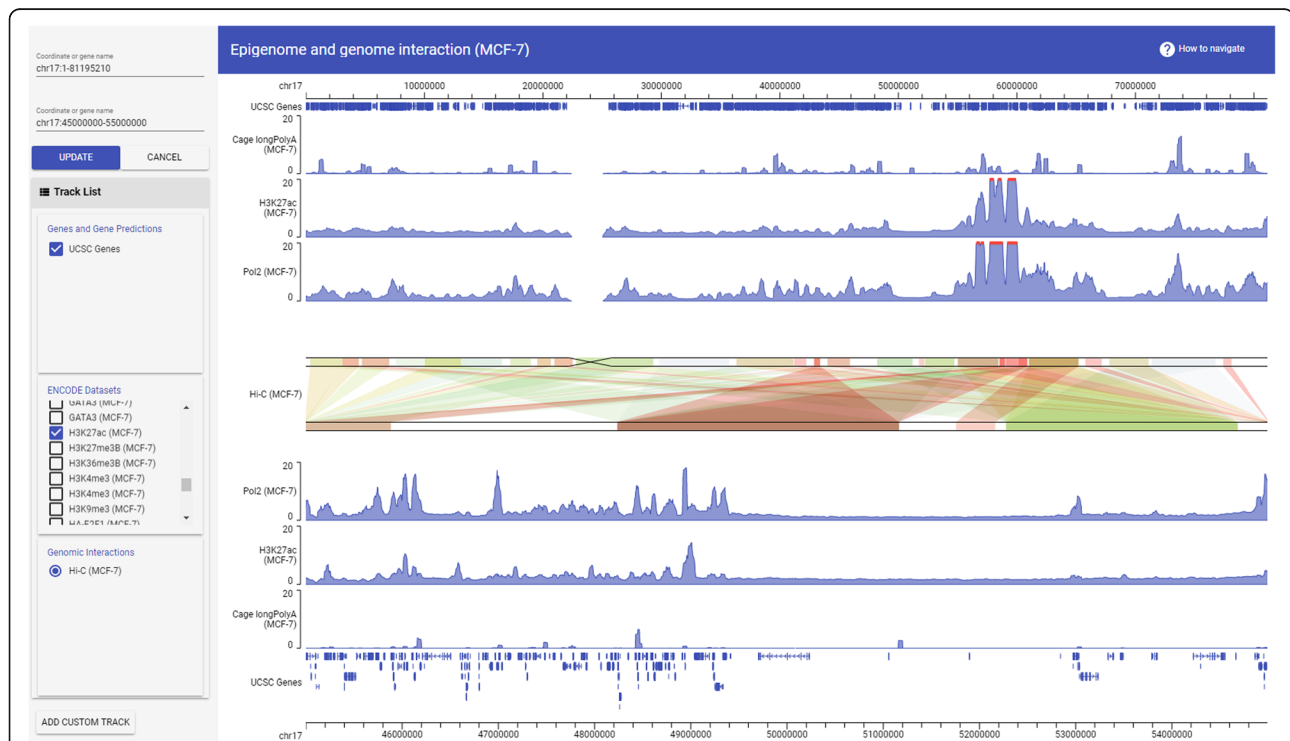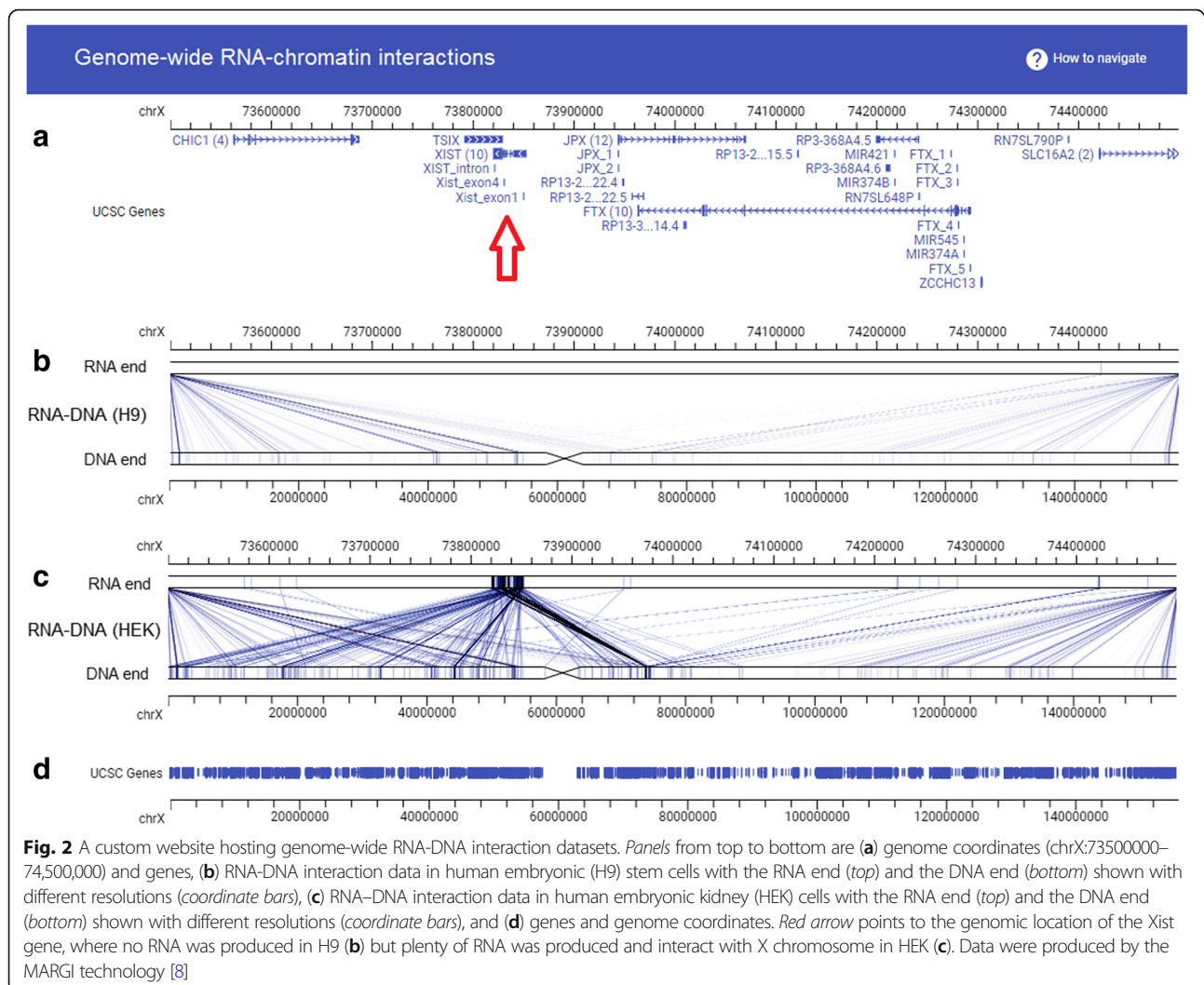

Fig. 1 *Screenshot* of a custom genome browser hosting epigenome and genome interaction datasets. The top genomic coordinate covers the entire chromosome 17 (chr17:1–81,195,210). The first three data tracks from the top are RNA-seq, H3K27ac ChIP-seq, Pol2 ChIP-seq data in MCF-7 cells, shown corresponding to the top genomic coordinate. The genomic coordinate at the bottom shows chr17:45000000–55,000,000. The bottom three data tracks are RNA-seq, H3K27ac ChIP-seq, Pol2 ChIP-seq data shown corresponding to the bottom coordinate. The Hi-C interaction data in the *center panel* shows Hi-C derived links between the genomic regions (top coordinate) and other genomic regions (bottom coordinate). The strengths of the Hi-C derived genomic interactions are plotted in color scale, with *red* being strongest and *green* being weakest

Cao *et al. Genome Biology* (2018) 19:92

Page 3 of 8

Fig. 2 A custom website hosting genome-wide RNA-DNA interaction datasets. *Panels* from top to bottom are (**a**) genome coordinates (chrX:73500000–74,500,000) and genes, (**b**) RNA-DNA interaction data in human embryonic (H9) stem cells with the RNA end (*top*) and the DNA end (*bottom*) shown with different resolutions (*coordinate bars*), (**c**) RNA–DNA interaction data in human embryonic kidney (HEK) cells with the RNA end (*top*) and the DNA end (*bottom*) shown with different resolutions (*coordinate bars*), and (**d**) genes and genome coordinates. *Red arrow* points to the genomic location of the Xist gene, where no RNA was produced in H9 (**b**) but plenty of RNA was produced and interact with X chromosome in HEK (**c**). Data were produced by the MARGI technology [8]

parameters using the top portion of this interactive window and hit the "Update code" button, leading to a new HTML code incorporating user-designated visualization parameters (Additional file 1: Figure S4). HUG offers the simplest way of generating GIVE-powered genome browser websites.

Managing custom data with GIVE-toolbox
To add and manage custom data, users should first download and run GIVE's main executable called GIVE-Docker. GIVE-Docker can be executed on all mainstream operating systems without system-specific configuration. When executed, GIVE-Docker automatically sets up a web server and a database system. Also packaged within this executable is a toolbox (GIVE-Toolbox) that automates all database operations into command line commands (Additional file 1: Table S1), thus relieving the user from working with a database language. Using the website hosting single-cell transcriptomes (https://single cell.givengine.org/) as an example, we provide a line-by-line example of building a website hosting custom data. After downloading and running GIVE-Docker, we will issue GIVE-Toolbox provided commands to initialize a reference genome, add gene annotations, and load custom data (Additional file 1: Table S2), followed by inserting HTML tags to display the data (Last row, Additional file 1: Table S2).

Without additional coding, the website is automatically equipped with a few interactive features. These features are enabled by JavaScript codes that are encapsulated within the GIVE's HTML tags. Visitors to this website can input new genome coordinates (Additional file 1: Figure S5A), choose any subset of data tracks to display (Additional file 1: Figure S5B), change genome coordinates by dragging the coordinates left or right by mouse (Additional file 1: Figure S5C), or zoom in and out the genome by scrolling the mouse wheel while the mouse pointer is on top of the genome coordinate area (Additional file 1: Figure S5C).

Cao *et al. Genome Biology* (2018) 19:92

Page 4 of 8

Double layer display of genome interaction data

GIVE implements a double layer display strategy for visualization of genome interaction data. In this display format, two genomic coordinates are plotted in parallel (Fig. 1, center, and Fig. 2b, c). Interactions between genomic regions are displayed as links of correspondent genomic regions between the top and bottom coordinates. When intensity values are associated with the links, the intensities are displayed using a red (large) to green (small) color scale (Fig. 1, center). This double layer display strategy has two advantages. First, the top and the bottom coordinates can cover different genomic regions, allowing it to visualize long-range interactions (Fig. 1). Users can shift or zoom the top and the bottom coordinates independently, making it easy to visualize, for example, interactions from the XIST locus (RNA end, Fig. 2b, c) to the entire X chromosome (DNA end, Fig. 2b, c). This double layer design also makes it intuitive to display asymmetric interactions, for example, interactions from RNA (top lanes, Fig. 2) to DNA (bottom lanes, Fig. 2).
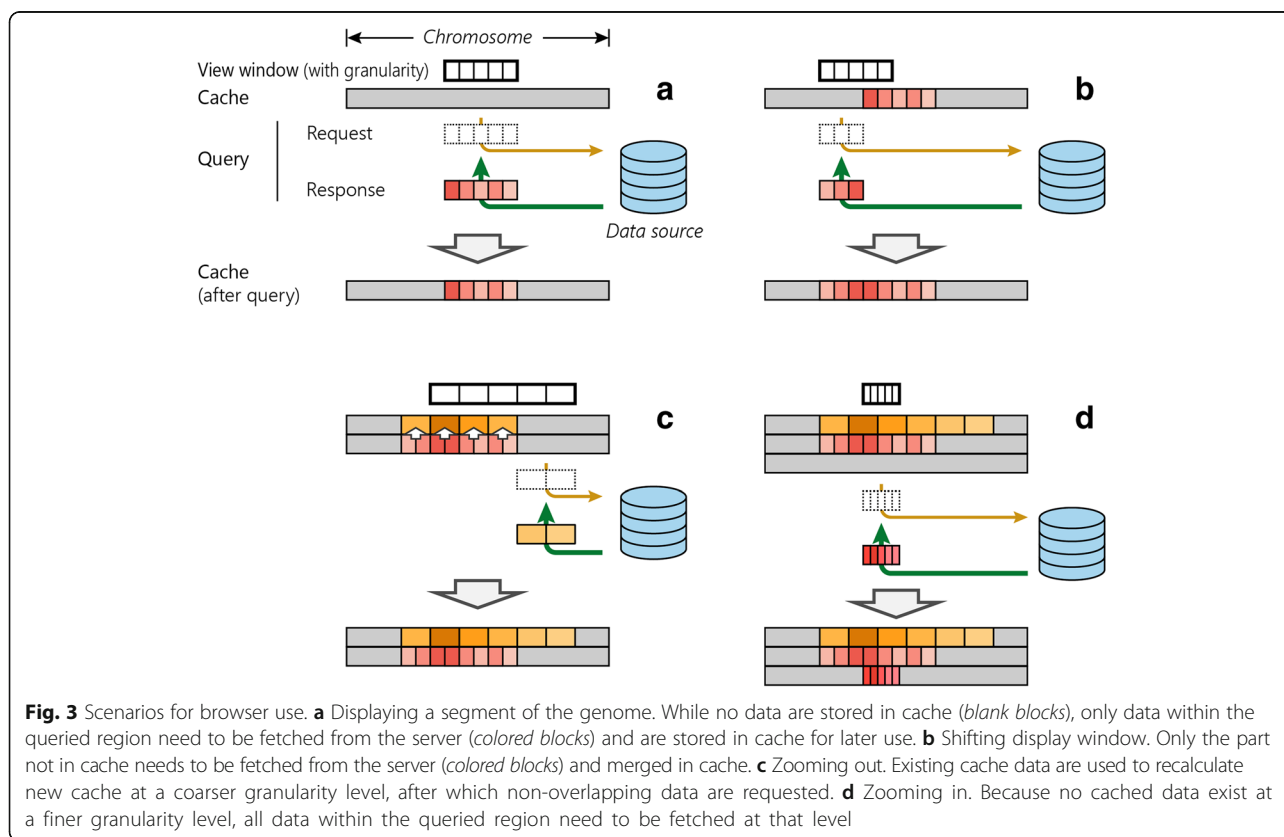
New data structures for transfer and visualization of genomic data

We developed two data structures for optimal speed in transferring and visualizing genomic data. These data structures and their associated technologies are essential to GIVE. However, all the technologies described in this section are behind the scenes. A website developer who uses GIVE does not have to recognize the existence of these data structures.

We will introduce the rationales for developing the new data structures with a usage scenario. When a user browses a genomic region, all genome annotation and data tracks within this genomic region should be transferred from the web server to the user's computer. At this moment, only the data within this genomic region require transfer and display (Fig. 3a). Next, the user shifts the genomic region to the left or right. Ideally, the previous data in the user's computer should be re-utilized without transferring again and only the new data in the additional genomic region should be transferred. After data transfer, the previous data and the new data in the user's computer should be combined (Fig. 3b).

Next, the user zooms out. This action changes the resolution of the genome. It is unnecessary to transfer and infeasible to display data at the previous granularity. At this point, the program should adjust the granularity of the already transferred data and then transfer additional data at the new granularity (Fig. 3c). When the user zooms in, the program will adjust to finer granularity and transfer data at this resolution (Fig. 3d). In summary, what is needed is a multiscale data

Fig. 3 Scenarios for browser use. **a** Displaying a segment of the genome. While no data are stored in cache (*blank blocks*), only data within the queried region need to be fetched from the server (*colored blocks*) and are stored in cache for later use. **b** Shifting display window. Only the part not in cache needs to be fetched from the server (*colored blocks*) and merged in cache. **c** Zooming out. Existing cache data are used to recalculate new cache at a coarser granularity level, after which non-overlapping data are requested. **d** Zooming in. Because no cached data exist at a finer granularity level, all data within the queried region need to be fetched at that level

Cao *et al. Genome Biology* (2018) 19:92

Page 5 of 8

container that can add or remove data from both sides of a genomic window.

To substantiate the multiscale data container described above, we developed two data structures named Oak and Pine. Oak handles sparse data tracks such as genome annotation, gene tracks, peak tracks, and interaction regions (BED, interaction data). Pine deals with dense data tracks in bigWig format [27]). Once the user changes the viewing area, Oak and Pine automatically adjust to the optimal tree structure for holding the data in the viewing area, which may involve change of data granularity, change of tree depths, adding or merging nodes, and rearranging node assignments to branches. These operations minimize data transfer over the Internet as well as the amount of data loaded in computer memory.

To optimize the use of memory, we developed an algorithm for removing obsolete data from the memory ("withering"). When the data stored in Oak or Pine nodes have not been accessed by the user for a long time, data in these nodes will be dumped and the memory recycled.

Methods
Using HTML tag library
Use of GIVE's HTML tags does not require any downloading or installation. The simplest way to try out GIVE's HTML tags is to use HUG, a graphical interface that will generate an HTML file for user chosen datasets.

Instead of using HUG, a web developer can import the entire GIVE library to a web page by inserting the following two lines (Lines 1, 2).

```
<script src="https://www.givengine.org/bower_components
/webcomponentsjs/webcomponents-lite.min.js"></script>     (Line 1)

<link rel="import" href="https://www.givengine.org
/components/chart-controller/chart-controller.html">         (Line 2)
```

To display genomics data, the web developer can use either the <chart-controller> tag or the <chart-area > tag. The <chart-controller> tag will display genomic data as well as genome navigation features such as shifting and zooming (Additional file 1: Figure S5C). For example, adding the following line in addition to the two lines above would create a website similar to that in Additional file 1: Figure S5 (Line 3).

```
<chart-controller title-text="Single-cell RNA-Seq" group-id
-list='["genes", "singleCell", "customTracks"]' num-of-
subs="1"></chart-controller>                                  (Line 3)
```

Here, the title-text attribute sets the title text of a website. The <chart-area> tag will display the track data without metadata controls such as data selection buttons and input box for genomic coordinates, while retaining some interactive capacities including dragging and zooming. This option provides the developer greater flexibility for website design. In addition, the <chart-area> tag is compatible with mobile apps.

Using GIVE-toolbox
GIVE-Toolbox is a set of command line tools offered to manage custom data (Additional file 1: Table S1). These command line tools automate data-related operations and relieve website developers from directly programming with a database language (MySQL). In addition to comprehensive documentation and tutorials (Additional file 1: Table S3), executing each tool with –h argument will output usage instruction. GIVE-Toolbox is our recommended option; however, developers can choose to directly work MySQL instead.

Running GIVE-Docker as a standalone executable
Utilizing Docker's container technology (https://www.docker.com), we encapsulated GIVE's codes and all the environmental requirements and database including Apache, MySQL, and PHP into a fully packaged executable called GIVE-Docker. This standardized executable can be deployed without system specific configuration to all mainstream operating systems and cloud computing services, including Linux, macOS, Windows 10, AWS, and Azure. This standalone executable does not require system administration or installation of any prerequisite compiler or database and therefore is the recommended option. Use of the GIVE HTML tag library does not require running GIVE-Docker.

Experienced programmers can choose custom installation instead of using GIVE-Docker. A step-by-step guide of custom installation is provided in GIVE's online manual.

Backstage technologies
The following technologies are wrapped inside the GIVE library. Website developers who use GIVE do not have to understand them or even know their existence.

Query
A query is issued when the user views any genomic region (query region). A new query is issued when the user changes the genomic region. A query induces two actions, which are data retrieval and display of data.

Oak, a data structure
A data structure called Oak is developed to effectively load and transfer a subset of data in BED format. The subset is defined as a continuous genomic region within a chromosome. Oak is a type of tree data structure, with nodes defined below.

A node is composed of a list of key-value pairs and a set of attributes. A key is a pair of starting and ending genomic coordinates, termed left key and right key,

Cao *et al. Genome Biology* (2018) 19:92

Page 6 of 8

respectively. When populated with data, a node keeps the data for a genomic region defined by the first left key and the last right key. The keys in a node partition the genomic region into non-overlapping sub-regions. A node can be either a branch node or a leaf node. The difference between a branch and a leaf lies in their values. A branch node is a node where the values are other nodes. A leaf node is a node where each value is a set of two lists of data points (Additional file 1: Figure S6). Each data point is a row of a BED file. When populated with data, the first list contains all the rows in the BED file where the start position matches the left key. The second list contains all the rows where the start and the end positions cover (span across) the left key. A value in a leaf node can also be empty. Leaf nodes with empty values are used to mark the genomic regions outside the query region.

Creating an oak instance, populating data, and updating oak

An Oak instance will be created, populated with data, or get updated in response to a query. These actions accomplish data transfer from the server to a user's computer. Only the data within the queried region will be transferred. Hereafter we will refer to an Oak instance as an Oak.

When the query region is on a new chromosome, an Oak will be created as follows. Every unique start position in the BED file that is contained within the query region is used to create a leaf node. The genomic regions on the queried chromosome but outside the query region are inserted as pairs of keys and empty values (placeholders) to the nodes with the nearest keys. The leaf nodes are ordered by their first left keys and sequentially linked by their pointers. A root node is created with all the leaf nodes are its children. This initial tree is fed into a self-balancing algorithm [28, 29] to construct a weight balanced tree, thus finishing the construction of an Oak.

When the query region is on a previously queried chromosome, the query region will be compared with the Oak of that chromosome and the overlapping region will be identified. The data of the overlapping region are therefore already loaded in the Oak and for the purpose of saving time; this should not be loaded again. The data in the rest of the query region will be loaded to the Oak. This is done by first creating a leaf node for every additional unique start position, removing the placeholder key-value pairs, and adding new placeholder key-value pairs for the rest of the chromosome. The weight balancing algorithm [28] is invoked again to re-balance this Oak. The weight balancing step prepares the Oak for efficient response to future queries.

Pine, a data structure

A data structure called Pine is developed to effectively load and transfer a subset of data in bigWig format. The subset is defined as a continuous genomic region within a chromosome. Pine can automatically determine the data granularity, which avoids transferring data at a higher than necessary resolution. The resolution of displayed data is limited by the number of pixels on the screen. Pine instances are always constructed to the appropriate depth and match the limit of the resolution.

A node consists of a list of key-value pairs and a set of attributes. The attributes are the same as those of Oak nodes, except there is an additional attribute, called data summary. The data summary includes the following metrics for a given node (the genomic region defined by the first left key and the last right key of the node): the number of bases; sum of values (summing over every base); sum of squares of the values; maximum value; and minimum value. A key is a pair of starting and ending genomic coordinates termed left key and right key, respectively. The keys in a node divide the genomic region into non-overlapping sub-regions. A node can be either a branch node or a leaf node. Their differences lie in the values. A branch node is a node where the values are other nodes (Additional file 1: Figure S7A). A leaf node is a node where each value is a list of data points (Additional file 1: Figure S7B). Each data point is a row of a bigWig file (binary format).

A node in Pine can have an empty key-value list and an empty data summary. If this is the case, we call it a placeholder node.

Creating a pine instance, populating data, and updating pine

A Pine is created when a query to a new chromosome is issued. A Pine is created with the following steps. First, the depth of the Pine tree is calculated as:

$$Tree\ depth = Ceiling\ (\ log_n(chromosome\ length) - log_n(resolution))$$

(1)

The limit of the resolution (length of genomic region per pixel) is the total length of the queried genomic region (viewing area) divided by the number of horizontal pixels, namely the width of the SVG element in JavaScript.

Next, a root node is created with keys covering the entire chromosome where the query region is contained within. Until reaching the calculated depth, for any node that overlaps with the query region, create a fixed number (n, $n = 20$ in the current release) of child nodes by equal partitioning its genomic region. If any of the created child nodes do not overlap with the query region, use a placeholder node. For each node, point the pointer

Cao *et al. Genome Biology* (2018) 19:92

Page 7 of 8

to the "right hand" node at the same depth. Thus, a Pine is created. This Pine has not loaded with actual data.

To load data, every leaf node issues a request to retrieve the summary data of its covered region (between the first left key and the last right key), which will be responded to by a PHP function wrapped within GIVE. This function returns summary data between the input coordinates from the bigWig file. After filling the summary data for all nodes at the deepest level, all parent nodes will be filled, where the summary data are calculated from the summary data of their child nodes. This process continues until reaching the root node.

A Pine will be updated when a new query partially overlaps with a previous query. In this case, the new depth (d2) is calculated using Eq. 1. This depth (d2) reflects the new data granularity. If d2 is greater than the previous depth, extend the Pine by adding placeholder nodes until d2 is reached. From root to depth d2–1, if any placeholder node overlaps with the query region, partition it by creating n child nodes. If any of the newly created child nodes does not overlap with the query region, use a placeholder node. For any newly created node, point the pointer to the "right hand" node at the same depth. At this step, the Pine structure is updated into proper depth. Finally, at depth d2, retrieve summary data for every non-placeholder node that has not had summary data. Update the summary data of their parent nodes until reaching the root. In this way, only the new data within the query region that had not been transferred before will get transferred.

Memory management

We developed a memory management algorithm called "withering." Every time a query is issued, this algorithm is invoked to dump the obsolete data, which have not been used in the previous ten queries. "Withering" works as follows: all nodes are added with a new integer attribute called "life span." When a node is created, its life span is set to 10. Every time a query is issued, all nodes overlapping with the query region as well as all their ancestral nodes get their life span reset to 10. The other nodes that do not overlap with the query region get their life span reduced by 1. All the nodes with life span equals 0 are replaced by placeholder nodes.

Discussion

The GIVE library is designed to reduce the need for specialized knowledge and programming time for building web-based genome browsers. GIVE is open source software. The open source nature allows the community at large to contribute to enhancing GIVE. The name GIVE (Genome Interaction Visualization Engine) was given when this project started with a smaller goal. Although it has grown into a more general-purpose library, we have decided to keep the acronym.

An important technical consideration is efficient data transfer between the server and users' computers. This is because users typically wish to get an instant response when browsing data. To this end, we developed several technologies to optimize the speed of data transfer. The central idea is threefold, including: (1) only transferring the data in the query region; (2) minimizing repeated data transfer by reusing previously transferred data; and (3) only transferring data at the necessary resolution. To implement these ideas, we developed two new approaches to index the genome and formalized these approaches with two new data structures, named Oak and Pine.

The Oak and Pine are indexing systems for sparse data (BED) and dense data (bigWig), respectively. BED data typically store genomic segments that have variable lengths. Given this particular feature, we did not index the genome base-by-base but rather developed a new strategy (Oak) to index variable-size segments. The bigWig files contain base-by-base data, which for a large genomic region can become too slow for web browsing. We therefore designed the Pine data structure that can automatically assess and adjust data granularity, which exponentially cut down unnecessary data transfer.

Conclusions

GIVE provides portable visualization components to personal websites. GIVE provides new data structures for efficient query, transmission, and visualization of functional genomic data. GIVE's double layer display format offers an alternative approach for visualizing genomic interaction data. GIVE-toolbox relieves web developers from programming with database languages and offers an easier approach for managing custom data. GIVE HUG helps users to generate HTML-based web pages. Custom datasets can be packaged with GIVE into interactive graphical formats and sent to designated collaborators.

Additional file

> **Additional file 1:** **Figure S1.** Screenshot of a website hosting ENCODE datasets. **Figure S2.** Screenshots of GIVE data hub. **Figure S3.** Selection of datasets in GIVE data hub. **Figure S4.** HUG generated HTML code. **Figure S5.** Screenshot of a custom genome browser. **Figure S6.** Oak data structure and operations. **Figure S7.** Pine data structure and operations. **Table S1.** Summary of GIVE Toolbox. **Table S2.** Related to Fig. 1. Line-by-line commands and codes for creating a genome browser loaded with custom data. **Table S3.** Templates with real codes and complete instructions. (PDF 743 kb)

Cao *et al. Genome Biology* (2018) 19:92

Page 8 of 8

Availability of data and materials
The GIVE website is at https://www.givengine.org, which provides samples websites, tutorials, manual, and GIVE executables. A mirror website is at: https://zhong-lab-ucsd.github.io/GIVE_homepage/. Source codes are available at GitHub (under project name "Genomic Interactive Visualization Engine", at https://github.com/Zhong-Lab-UCSD/Genomic-Interactive-Visualization-Engine) [31] and at Zenodo (https://doi.org/10.5281/zenodo.1134907) [32]. GIVE HTML tag library is programmed in HTML, JavaScript and PHP, GIVE-Toolbox is programmed in BASH scripts. GIVE HTML tag library is supported on all major OS platforms (Linux, Windows and macOS) and all major browsers (Chrome, Firefox, Safari and Edge), GIVE-Docker and GIVE-Toolbox are supported on Linux.
GIVE is licensed under the Apache License, Version 2.0 (the "License"). A copy of the License is available at https://www.apache.org/licenses/LICENSE-2.0. A total of nine demos and tutorials with real codes and complete instructions are provided in the GIVE tutorials (Additional file 1: Table S3) (https://github.com/Zhong-Lab-UCSD/Genomic-Interactive-Visualization-Engine/tree/master/tutorials). Supplementary figures and tables are provided in Additional file 1.

Publisher's Note
Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References
1. Tyner C, Barber GP, Casper J, Clawson H, Diekhans M, Eisenhart C, et al. The UCSC genome browser database: 2017 update. Nucleic Acids Res. 2017;45:D626–34.
2. Zhou X, Maricque B, Xie M, Li D, Sundaram V, Martin EA, et al. The human epigenome browser at Washington University. Nat Meth. 2011;8:989–90.
3. Li R, Liu Y, Li T, Li C. 3Disease browser: a web server for integrating 3D genome and disease-associated chromosome rearrangement data. Sci Rep. 2016;6:34651.
4. Wang Y, Zhang B, Zhang L, An L, Xu J, Li D, et al. The 3D Genome Browser: a web-based browser for visualizing 3D genome organization and long-range chromatin interactions. bioRxiv; 2017. https://doi.org/10.1101/112268.
5. 23andMe. When People Share their Genome on Facebook; 2011. https://blog.23andme.com/23andme-and-you/when-people-share-their-genome-on-facebook/. Accessed 12 Mar 2018.
6. Lieberman-Aiden E, van Berkum NL, Williams L, Imakaev M, Ragoczy T, Telling A, et al. Comprehensive mapping of long-range interactions reveals folding principles of the human genome. Science. 2009;326:289–93.
7. Fullwood MJ, Liu MH, Pan YF, Liu J, Xu H, Mohamed YB, et al. An oestrogen-receptor-alpha-bound human chromatin interactome. Nature. 2009;462:58–64.
8. Sridhar B, Rivas-Astroza M, Nguyen TC, Chen W, Yan Z, Cao X, et al. Systematic mapping of RNA-chromatin interactions in vivo. Curr Biol. 2017;27:610–2.
9. Li X, Zhou B, Chen L, Gou LT, Li H, Fu XD. GRID-seq reveals the global RNA-chromatin interactome. Nat Biotechnol. 2017;35:940–50.
10. Lu Z, Zhang QC, Lee B, Flynn RA, Smith MA, Robinson JT, et al. RNA duplex map in living cells reveals higher-order transcriptome structure. Cell. 2016;165:1267–79.
11. Nguyen TC, Cao X, Yu P, Xiao S, Lu J, Biase FH, et al. Mapping RNA-RNA interactome and RNA structure in vivo by MARIO. Nat Commun. 2016;7:12023.
12. Sharma E, Sterne-Weiler T, O'Hanlon D, Blencowe BJ. Global mapping of human RNA-RNA interactions. Mol Cell. 2016;62:618–26.
13. Aw JG, Shen Y, Wilm A, Sun M, Lim XN, Boon KL, et al. In vivo mapping of eukaryotic RNA Interactomes reveals principles of higher-order organization and regulation. Mol Cell. 2016;62:603–17.
14. Ozsolak F, Milos PM. RNA sequencing: advances, challenges and opportunities. Nat Rev Genet. 2011;12:87–98.
15. Zhou VW, Goren A, Bernstein BE. Charting histone modifications and the functional organization of mammalian genomes. Nat Rev Genet. 2011;12:7–18.
16. Buenrostro JD, Giresi PG, Zaba LC, Chang HY, Greenleaf WJ. Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, DNA-binding proteins and nucleosome position. Nat Methods. 2013;10:1213–8.
17. Skinner ME, Uzilov AV, Stein LD, Mungall CJ, Holmes IH. JBrowse: a next-generation genome browser. Genome Res. 2009;19:1630–8.
18. Stein LD. Using GBrowse 2.0 to visualize and share next-generation sequence data. Brief Bioinform. 2013;14:162–71.
19. Barrios D, Prieto C. D3GB: an interactive genome browser for R, Python, and WordPress. J Comput Biol. 2017;24:447–9.
20. Carrere S, Gouzy J. myGenomeBrowser: building and sharing your own genome browser. Bioinformatics. 2017;33:1255–7.
21. Robinson JT, Thorvaldsdottir H, Winckler W, Guttman M, Lander ES, Getz G, et al. Integrative genomics viewer. Nat Biotechnol. 2011;29:24–6.
22. Biase FH, Cao X, Zhong S. Cell fate inclination within 2-cell and 4-cell mouse embryos revealed by single-cell RNA sequencing. Genome Res. 2014;24:1787–96.
23. The ENCODE Project Consortium. An integrated encyclopedia of DNA elements in the human genome. Nature. 2012;489:57–74.
24. Yue F, Cheng Y, Breschi A, Vierstra J, Wu W, Ryba T, et al. A comparative encyclopedia of DNA elements in the mouse genome. Nature. 2014;515:355–64.
25. Mourad R, Hsu PY, Juan L, Shen C, Koneru P, Lin H, et al. Estrogen induces global reorganization of chromatin structure in human breast cancer cells. PLoS One. 2014;9:e113354.
26. Sridhar B, Rivas-Astroza M, Nguyen TC, Chen W, Yan Z, Cao X, et al. Systematic mapping of RNA-chromatin interactions in vivo. Curr Biol. 2017;27:602–9.
27. Kent WJ, Zweig AS, Barber G, Hinrichs AS, Karolchik D. BigWig and BigBed: enabling browsing of large distributed datasets. Bioinformatics. 2010;26:2204–7.
28. Bayer R, McCreight EM. Organization and maintenance of large ordered indexes. Acta Informatica. 1972;1:173–89.
29. Comer D. Ubiquitous B-tree. ACM Comput Surv. 1979;11:121–37.
30. Polymer Authors. Polymer. 1.9.3 edition. 2017. https://www.polymer-project.org/.
31. Cao X, Yan Z, Wu Q, Zheng A, Zhong S. GIVE: portable genome browsers for personal websites. Github; 2018. https://github.com/Zhong-Lab-UCSD/Genomic-Interactive-Visualization-Engine. Accessed 7 July 2018.
32. Cao X, Yan Z, Wu Q, Zheng A, Zhong S. GIVE: portable genome browsers for personal websites. Zenodo; 2018. https://doi.org/10.5281/zenodo.1134907. Accessed 12 Mar 2018.