DivBrowse – interactive visualization and exploratory data analysis of variant call matrices

Patrick König[1], Sebastian Beier[1,4], Martin Mascher[1,2], Nils Stein[1,3], Matthias Lange[1] and Uwe Scholz[1]

[1]Leibniz Institute of Plant Genetics and Crop Plant Research (IPK) Gatersleben, 06466 Seeland, Germany,
[2]German Centre for Integrative Biodiversity Research (iDiv) Halle-Jena-Leipzig, 04103 Leipzig, Germany,
[3]Center for Integrated Breeding Research, Georg-August University, 37075 Göttingen, Germany
[4]Forschungszentrum Jülich GmbH, Institute of Bio- and Geosciences, Bioinformatics (IBG-4), 52425 Jülich, Germany

Keywords: genomics, data visualization, variation data, variant call format

Abstract

Background

The sequencing of whole genomes is becoming increasingly affordable. In this context large-scale sequencing projects are generating ever larger datasets of species-specific genomic diversity. As a consequence, more and more genomic data needs to be made easily accessible and analyzable to the scientific community.

Results

We present DivBrowse, a web application for interactive visualization and exploratory analysis of genomic diversity data stored in Variant Call Format (VCF) files of any size. By seamlessly combining BLAST as an entry point together with interactive data analysis features such as principal component analysis in one graphical user interface, DivBrowse provides a novel and unique set of exploratory data analysis capabilities for genomic biodiversity datasets. The capability to integrate DivBrowse into existing web applications supports interoperability between different web applications. Built-in interactive computation of principal component analysis allows users to perform ad-hoc analysis of the population structure based on specific

genetic elements such as genes and exons. Data interoperability is supported by the ability to export genomic diversity data in VCF and General Feature Format (GFF3) files.

Conclusion

DivBrowse offers a novel approach for interactive visualization and analysis of genomic diversity data and optionally also gene annotation data by including features like interactive calculation of variant frequencies and principal component analysis. The use of established standard file formats for data input supports interoperability and seamless deployment of application instances based on the data output of established bioinformatics pipelines.

1 Introduction

In times of ever-increasing quantity and quality of sequencing data that is driven by the ongoing reduction of sequencing costs and simultaneously increasing computing power, an ever-increasing amount of data of genomic diversity is being generated, which is almost impossible to keep track of in its entirety [1,2]. Rather, there is a need to reduce complexity through models that aggregate the data in a meaningful way but preserve the information capacity.

One goal of diversity analyses is to find the underlying changes in genomic traits by observing the type of change, frequency and correlations to the phenotype. A prerequisite for this is dense genomic diversity datasets to perform GWAS, which can directly identify related nucleotide polymorphisms, genes and other genetic features like promoters or enhancers [3].

Visualization of such huge genomic datasets as used in diversity analysis remains a challenge for software tools and their developers, as the amount of generated data grows exponentially fast and the development of appropriate visualization and analysis tools is a time-consuming effort [4,5].

For this reason, we developed DivBrowse as a tool specifically for the interactive visualization of very large variant call matrices, which allows users to interactively navigate through variant matrices in the order of hundreds of millions of variants and several thousand to tens of thousands of genotypes. It allows the user to keep a visual overview of very large but also small datasets of genomic diversity, supplemented by interactive analysis features like principle component analysis for ad-hoc investigation of the population structure on the level of genetic features like genes, promotors, enhancers or silencers. DivBrowse can serve as a daily used entry point for exploratory data analysis of data sets of genomic diversity aligned to a reference genome for all species. It uses standardized and established bioinformatics file formats like the Variant Call Format (VCF) [6] for single nucleotide polymorphisms and the Generic Feature Format Version 3 (GFF3) for genome annotation data [7].

DivBrowse was initially developed as part of a web-based information system for visual analytics in the frame of a barley genebank genomics project to serve and visualize VCF-based genotypic data with 22,621 genotypes and 171,263 variants from a GBS approach [8–10]. A new barley reference genome expanded the number of variants of this genotype panel to 775,283 [11]. A VCF file derived from whole genome sequencing data with 223,387,147 variants and 300 barley genotypes was used for the demo instance of DivBrowse available under https://divbrowse.ipk-gatersleben.de/demo [11].

2 System and Methods

DivBrowse combines the approach of genome browsers with the capability to visualize and interactively analyze thousands to millions of genomic variants for thousands of genotypes in the style of an exploratory data analysis [12]. The graphical user interface (GUI) of DivBrowse is accessed via a web browser. It shows genomic features such as nucleotide sequence, associated gene models and genomic variants. Their physical positions in the currently selected chromosome or contig are shown in the upper section of the GUI and the genotypes are listed row-wise in the lower section (see Figure 1). Besides the visualization of the variant calls per variant and genotype, DivBrowse also calculates and displays variant statistics such as minor allele frequencies, proportion of heterozygous calls or missing variant calls for each visualized genomic window. Furthermore, variant effect predictions according to SnpEff [13] can be displayed, provided they are present in the underlying VCF file.

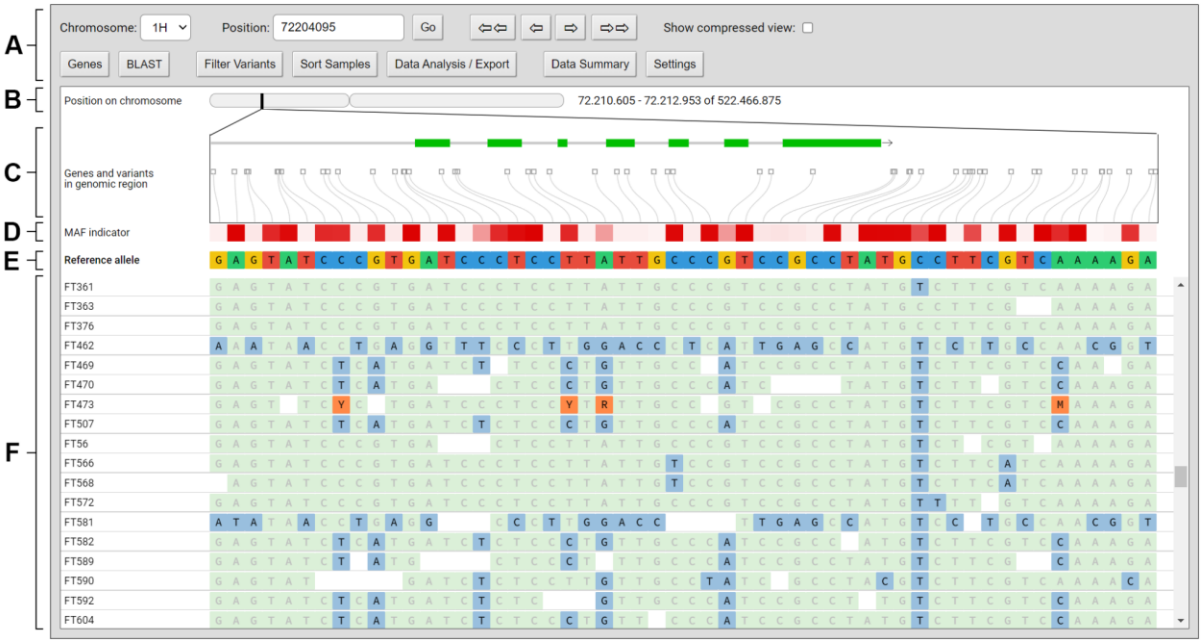


Figure 1: Screenshot of the initial view of the graphical user interface of DivBrowse. It is organized from top to bottom into the following main sections: (A) The navigation and tool menu where users can enter a physical position where to jump to, buttons for navigating left and right through the variant matrix and buttons to open dialogues for data analysis and export features. (B) An overview map of the chromosome with the current visible genomic region highlighted. (C) The track truly scaled in terms of physical positions and distances for genetic traits and variant positions. (D) The track with the visual indicator for the minor allele frequency of each variant where boxes with darker red indicate smaller minor allele frequencies. (E) The track with the reference allele for this variant present in the reference genome. (F) The scrollable track with all variant calls for each combination of variant and genotype.

General usability concept

The main concept for the visualization of the variant matrices is a tabular-like gapless display with the variants as columns and genotypes as rows alongside a genome track that visualizes the physical positions of the variants together with physical positions of genes, exons and other

genetic features in the current visible genomic range. Whereas the horizontal axis of the genome track at the top is scaled with respect to the physical positions and distances, the variant calls at the bottom are visualized gapless. Since variants generally do not cover all physical positions, this creates a positional divergence between the variant calls and the corresponding physical positions of the variants in the genome track. As a solution for this divergence bezier curves have been implemented that connect the physical positions of the variants with the corresponding columns in the variant calls matrix below. With this approach more variant calls can be visualized at the same time on the available screen width while maintaining the physical coordinates of each variant along the reference genome. Each variant column is connected to the physical coordinate of the variant through a bezier curve.

We envision that common use of DivBrowse proceeds in three successive stages, as illustrated and explained in Figure 2.

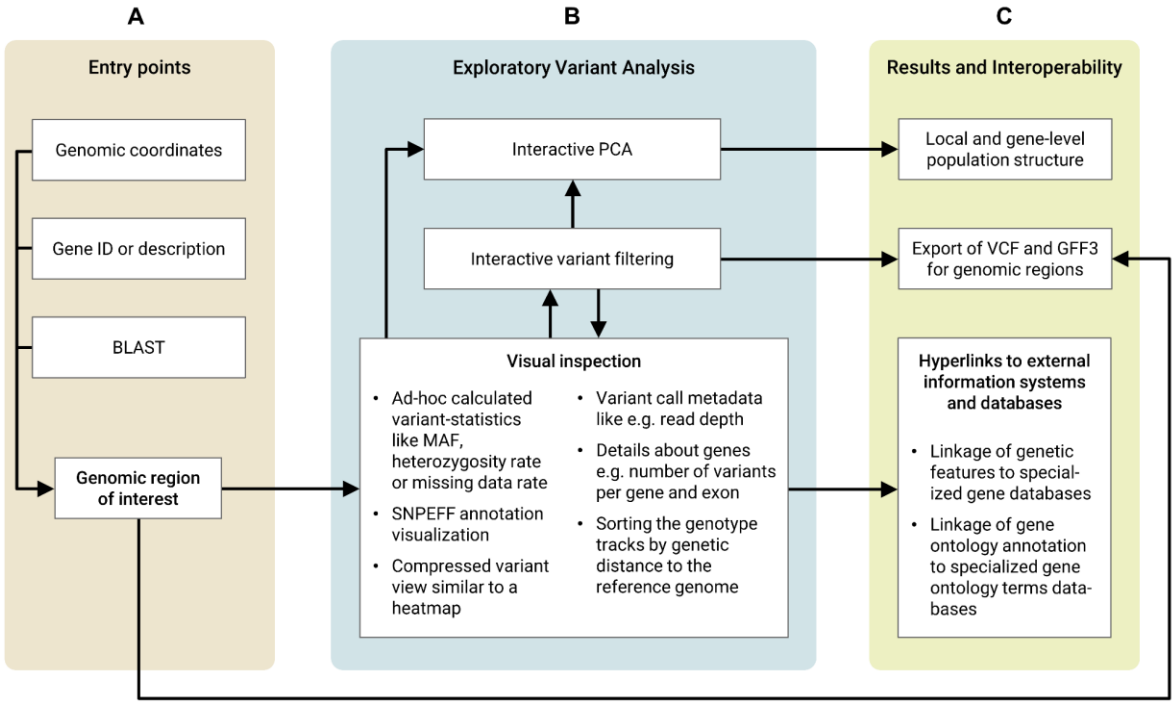


Figure 2: The workflow concept of DivBrowse can be divided into three successive workflow stages: (A) As an initial step users can access the variant data by three independent entry points: a) by providing a genomic coordinate consisting of the chromosome label and the

physical position, b) by searching for a gene by its annotation ID or description and then jumping to the genomic coordinate of this gene, c) by performing a BLAST search with a user-defined nucleotide or protein sequence and then jumping to the genomic coordinates of the BLAST results. (B) After the user has found a genomic region of interest, they can now visually inspect and exploratively analyze the variants. Variant statistics like minor allele frequency or heterozygosity frequency are visualized by a heatmap-style track. The user is able to filter the variants based on variables of the variant statistics. It is also possible to perform a principle component analysis for a genomic window or a gene with or without application of the variant filter settings. We call this workflow "Exploratory Variant Analysis". (C) As a result of the iterative and interactive Exploratory Variant Analysis workflow, the user is able to get a better understanding of local (e.g. promoter or enhancer/silencer regions) or gene-level population structure. The user can also export the variation data together with genome annotation data for a genomic window or gene in VCF and GFF3 files to use those in further downstream analysis steps. It is also possible to integrate data and knowledge from other specialized databases like gene or gene ontology databases that are linked via hyperlinks.

Efficient handling of large variation data matrices

DivBrowse is able to efficiently handle and visualize very large variant matrices in the VCF file format containing variant calls for thousands to tens of thousands of genotypes and thousands to hundreds of millions of variants. Typically, these VCF files can exceed file sizes of 100 gigabytes, even when compressed with GZip or BGZip. Uncompressed file sizes can exceed 1 terabyte easily. It is therefore important to efficiently handle those large amounts of data to enable interactive access to slices of the variation data matrix with request-response-cycle times below one second for a pleasant user experience. DivBrowse is able to do so by using the Zarr Python package for storing the variant calls as n-dimensional compressed

chunked arrays [14]. The VCF file to be visualized is converted to the Zarr format with the vcf_to_zarr() method of the Python package *scikit-allel* [15] beforehand.

Command-line interface (CLI)

The DivBrowse CLI provides a way to start an ad-hoc session of an instance of DivBrowse, that uses VCF and GFF3 data files present in the current working directory of the user's shell. The CLI automatically detects multiple VCF and GFF3 files and provides an interactive dialogue where the user can select the files to be used to start an instance of DivBrowse. The CLI automatically infers basic configuration settings from the VCF and GFF3 files, e.g. the number of chromosomes and its labels and a mapping from chromosome labels in the VCF file to those in the GFF3 file. It is also possible to provide a manually created DivBrowse configuration file (divbrowse.config.yml) in the YAML format [16]. An automatically inferred configuration can also be saved as a configuration YAML-file in the current working directory. Such an inferred configuration file can be then used as a skeleton for a more advanced configuration file to use all features of DivBrowse. Since not all functions of DivBrowse can be derived from the underlying files, some functions have to be activated or configured specifically via the YAML configuration file. Please refer to the official documentation available under https://divbrowse.ipk-gatersleben.de/ for more information.

Gene search

If a gene annotation of the genome exists as a GFF3 file and has been loaded into a DivBrowse instance, it is possible to use the "Gene Search" feature. The "Gene search" dialogue allows searching all genes by their ID or annotated label. It is also possible to list all genes within a user-defined genomic range on a specific chromosome. Both search capabilities can be used in combination to search for genes by ID or annotation on a specific chromosome or within a user-defined genomic range on a specific chromosome. In addition to the number of variants

on the entire gene, the number of variants located on exons of a gene is also displayed in the search results table when using the configuration setting "count_exon_variants: true".

BLAST as an entry point

As an additional entry point besides the entry points "genomic position" and "search genes by ID/description", an optional BLAST entry point has also been implemented in DivBrowse [17]. It allows to perform a BLAST search for a given DNA (blastn) or protein sequence (tblastn) of the user's choice. The BLAST search is implemented by using the Galaxy API of an existing Galaxy instance via the Python package *BioBlend* [18]. The Galaxy instance to be used must have the NCBI BLAST+ tools [19] and the corresponding BLAST databases for the reference genome used in the original VCF installed. In order to use a Galaxy instance for a BLAST search by DivBrowse, the user must have an API key or user account credentials for the Galaxy instance to be used. After the BLAST search has been conducted by the Galaxy server, results are then displayed in a table that consists of a column that reports about the number of variants within the aligned target sequence for each BLAST hit. A DivBrowse user can directly see whether his BLAST hits contain variants or not. The BLAST result table also includes a column with links that can be used to jump directly to the genomic region of each BLAST hit. The BLAST results are stored within the current web browser session so that a DivBrowse user can review all BLAST hits without repeating the whole BLAST search again. In addition, multiple BLAST search results are stored in a session-based cache and can be loaded within microseconds without repeating the BLAST search.

Variant filtering

The "Filter variants" dialogue allows to filter the variants by different variant-level attributes. It is possible to filter the variants by ad-hoc calculated variant statistics like minor allele

frequency/fraction, heterozygosity fraction and missing data fraction. These ad-hoc calculated variant statistics are dependent on the currently selected genotype panel. It is also possible to filter by variant-level attributes that have been derived from the VCF input file, e.g. by the QUAL- or MQ-attribute of a variant. As those metadata attributes are derived from the variant calling process it is not possible to calculate those metrics in an ad-hoc manner. The user-defined filter settings can also be used in the interactive PCA to exclude certain variants from the PCA calculation. This allows users of DivBrowse to interactively study PCA results based on different variant filter criteria. Furthermore, it is also possible to use the user-defined variant filtering in the VCF export to export only those variants that match the specific variant filter criteria to a newly created VCF file. This enables DivBrowse users to export custom filtered variants to a VCF file for their personal downstream analysis scenarios.

Sample sorting

The "Sort samples" feature allows sorting the list of genotypes according to different criteria. On the one hand, the genotype tracks can be sorted alphabetically either in ascending or descending order. This could be helpful if the ID's or labels of the genotypes can be sorted alphabetically in a meaningful way, e.g. if they are actual names of plants or animals and the names can infer ancestry or different traits. Another option is to sort the genotype tracks based on the respective genetic distance from the reference genome. For the calculation of genetic distances, the pairwise Euclidean distances between all genotype vectors is used as the simplest estimator for the genetic distance [20]. The genotype vectors used for the pairwise distance calculation contain the number of alternative alleles for each variant. Here, the user can select which genomic region should be used to calculate the genetic distance for sorting the tracks. It is possible to calculate the genetic distance a) within the currently visible genomic range in the viewport, b) within the boundaries of a currently visible gene or sub-feature of a gene (e.g.

specific exon) or c) within a user-defined genomic window, which could be useful to sort by the genetic distances of an intragenic region, gene flanking region or regulatory region.

Interactive Principal Component Analysis

The built-in principal component analysis (PCA) functionality allows for interactive principle component analysis of a) the variants that are currently visible in the viewport, b) the variants on the complete gene or sub-features of the gene (e.g. a specific exon) that is currently visible in the viewport or c) the variants within a user-defined genomic range. Interactive in this context refers to the PCA being computed on-demand with a runtime in the timeframe of seconds. Due to possible hardware limitations on the server-side compute power, the PCA calculation can be limited to a maximum number of included variants by the configuration setting "pca_max_variants" in the configuration YAML-file of DivBrowse. This allows the user to adjust the amount of variants to be included in the PCA calculation depending on the computational capacity of the web server. For smaller slices up to 10,000 genotypes and 100 variants of the whole variant matrix, the PCA calculation can be done within under one second on average contemporary server hardware.

Export of VCF and GFF3 files

DivBrowse allows exporting data in standard formats, namely VCF and GFF3. The VCF export feature allows the export of variation data of a given genomic range to standard-compliant VCF files [6]. The user can export a) the genomic region that is displayed in the current viewport of DivBrowse, b) the genomic region encompassed by a currently visible gene or sub-feature of that gene (e.g. exon) or c) a genomic window defined by a start and end position within a single chromosome by the user. The GFF3 export feature works similarly to the VCF export feature, but instead exports a GFF3 file containing the gene annotation.

Linkage of genetic features to other web-based information systems

DivBrowse allows setting up URLs to external websites for each gene of a given GFF3 file by a configuration setting in the divbrowse.config.yml file. Furthermore, external links for each available gene ontology term can be set up. This allows users of DivBrowse to get specific and more detailed information about a gene and its ontology terms on specialized external websites and databases, e.g. by linking to species-specific gene expression databases or by linking to the QuickGO service of the European Bioinformatics Institute (EBI) for the ontology terms provided in the GFF3 file for each gene [21].

Standalone or plugin usage

DivBrowse can be used either standalone or as a Javascript plugin to complement existing functionality in an already existing web application. Using it as a plugin makes sense if there is a web application with existing omics data, e.g. phenotypic data, which should be supplemented with a visualization of genotypic data that has a relational connection to the already existing omics data. If only genotypic data is available, the standalone use is sufficient and advised. The usage of DivBrowse as a plugin opens additional possibilities concerning the interaction between the hosting web application and the DivBrowse instance that are described more in detail in the use case “Plugin in existing web-based information systems” in the discussion section of this manuscript. In the case of plugin usage, the communication between the hosting web application and the DivBrowse instance can be realized by using the DivBrowse Javascript-API, which is described in the section “Implementation” of this work.

3 Implementation

General Application Architecture

DivBrowse is implemented in a client-server-architecture that uses a Python-based stack for the server-side part and a stack based on JavaScript in combination with the Svelte framework [22], HTML and CSS for the client-side component (GUI) (see Figure 3). The server-side part uses multiple third-party packages available on the Python Package Index (PyPI, www.pypi.org) that are listed in Supplementary Table 1. The DivBrowse server-side component is configurable by a YAML-format based configuration file named "divbrowse.config.yml" per default. It is also possible to use alternative file names for the configuration file.

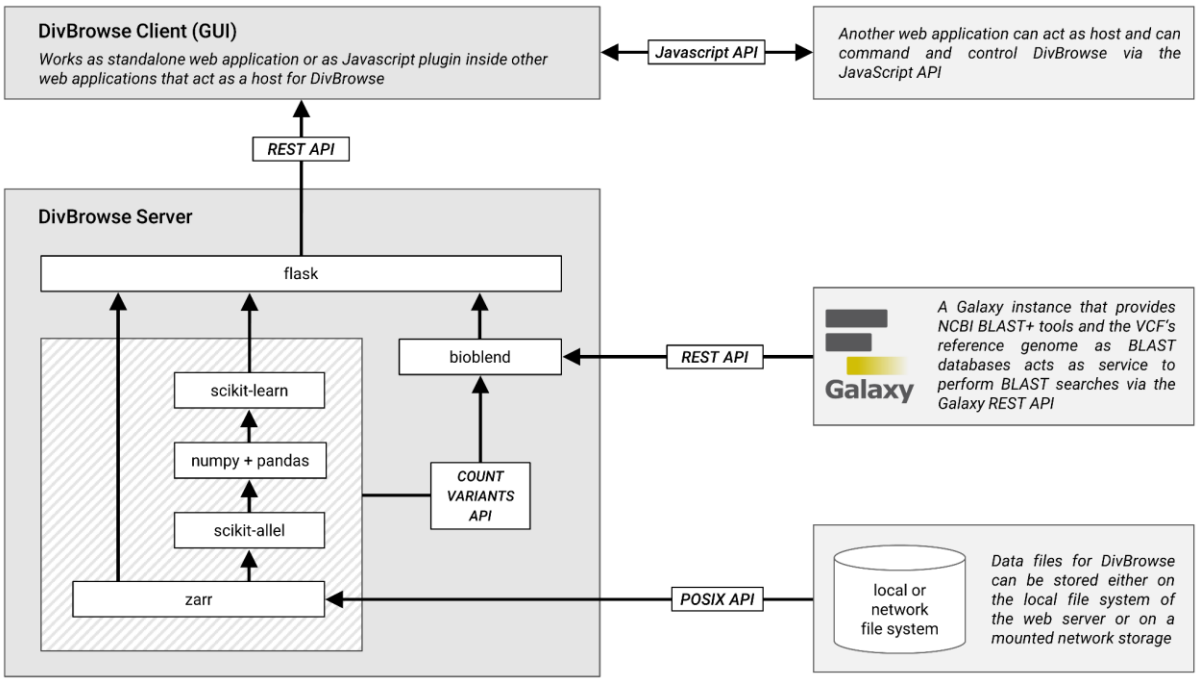


Figure 3: The general architecture of DivBrowse showing the flow of data between the different components and between the server component and the storage layer.

Architecture of the server-side component

The DivBrowse server-side component is a Python application that utilizes Flask as a micro web framework to provide and expose the DivBrowse REST-API used for communication between itself and the graphical user interface that runs client-side in the user's web browser. The server-side component uses third-party packages available on the Python Package Index (PyPI, www.pypi.org) like bioblend, numpy, scikit-learn, zarr, scikit-allel, flask, pandas, pyyaml and click (see Supplementary Figure 1). The server-side component can also serve all the static files necessary for the GUI like HTML, Javascript and CSS files. In a productive setup, we recommend using specialized web server software such as Nginx or Apache HTTP Server to serve the static files of the GUI and to act as a proxy server for the WSGI-based Python processes. For instructions on how to set up such a productive web server environment with Nginx, see the official documentation. An overview of all endpoints of the DivBrowse REST-API can be found in supplementary table S-T1.

Architecture of the graphical user interface

The graphical user interface (GUI) of DivBrowse is implemented in Javascript, HTML and CSS utilizing the Svelte framework for a modular and component-based application architecture (see Supplementary Figure 2). In order to maximize compatibility in the case of a plugin usage of DivBrowse inside of an existing web portal we decided not to use a CSS framework like Bootstrap to lower the risk of CSS conflicts due to the cascading nature of CSS. All CSS rules were created individually by hand. In addition, CSS rules were completely avoided to be applied globally at HTML tag level. Instead, rules are applied only based on ID or class attributes with a common prefix "divbrowse" to simulate a CSS namespace. The DivBrowse Javascript component provides a JavaScript API that allows interaction and communication between a DivBrowse instance and a hosting web application. In such a

scenario, DivBrowse acts as a plugin or subcomponent inside of another web application. The Javascript API provided methods to control certain aspects of DivBrowse, e.g. to subset the list of genotypes or to jump to a specific genomic position. Furthermore, callback functions are provided that are called after specific events or user interactions in the DivBrowse GUI. For example when the user performs an interactive PCA and then makes a lasso selection of some genotype samples in the PCA plot, a callback function "selectedSamples()" is called with an array of sample-IDs that have been selected as the first function argument. In this way selection of genotype samples can be transferred from DivBrowse to the host web application where the sample selection can be processed further.

JavaScript-API

The JavaScript-API allows controlling the GUI of DivBrowse from a web application that itself hosts a DivBrowse instance. This allows an interactive coupling of existing web applications for omics data with one or many DivBrowse instances. For example, it is then possible to force a DivBrowse instance to jump to a specific genomic range in its viewport. Communication and data transfer from a DivBrowse instance to the hosting portal is also possible. For example, if a user selects some genotypes of interest in a PCA result, the sample-IDs of those genotypes can be transferred back to the hosting portal via a Javascript callback function. A detailed list of all available JavaScript-API commands is given in Supplementary Table 2.

4 Discussion

Since the visualization is performed by DivBrowse in the web browser, a convenient and low-barrier access to genotypic diversity data is possible even without special bioinformatics knowledge and without time-consuming download of VCF files and their computationally intensive processing on own computer hardware. This supports easy access to the extensive biodiversity data that already exists and will be obtained in the future, making it easily available to a wide audience in the spirit of open data. To realize low latencies for the interactive visualization of variant matrices of the size of several hundred gigabytes, an efficient server-side backend is implemented in Python. For this purpose, the backend uses n-dimensional chunked arrays and is performant enough for a good user experience in contemporary hardware environments like notebooks, standard desktop computers and bare metal server or virtual machine servers. This aspect lowers the financial and technical hurdles for the interactive visualization of genotypic diversity data using DivBrowse for institutions that generate, store, and make genotypic data available for public access. Moreover, the use of species-agnostic bioinformatic file formats, DivBrowse can be used with any species that has a diploid genome. Due to the progressing genetic inventory of plant genetic resources stored in genebanks in the frame of genebank genomics projects, we expect a significantly increasing amount and size of genotypic data sets in the future, whose easy accessibility and visualization is of great importance [23]. We have shown that DivBrowse is capable of delivering diversity matrices with thousands of genotypes and hundreds of millions of variants on standard server hardware or even virtual machines. It thus provides a reusable tool to efficiently deliver the increasing amount of genotypic data to the public in the form of an interactive visualization with easy to use basic analysis functions and multiple entry points.

The Variant Call Format (VCF), that acts as the input file format for genotypic data for DivBrowse, has suffered from a number of indeterminacies in the past with regard to the FAIR

criteria [24,25]. These indeterminacies have made fully automated processing difficult or impossible without manual intervention in terms of interoperability at the level of bioinformatics toolchains. As DivBrowse continues to evolve, we look forward to incorporating recent advances and improvements related to metadata in VCF files to better take advantage of the FAIR data paradigm, such as improved interoperability and reusability [25]. For example, the usage of standardized sample-IDs for the genotypes based on BioSamples-IDs in the VCF metadata section would allow DivBrowse to easily read those BioSamples-IDs and use them to automatically interconnect the genotypes in the GUI of DivBrowse with other omics-related web-based information systems [26]. As another example, the appropriate standardized usage of the VCF metadata field "*Contig*", as recommended in Beier et al. [25], would allow to automatically derive human-readable chromosome labels, the chromosome length and the species from the VCF file. This would make corresponding manual entries for the chromosome labels and length in the DivBrowse configuration file unnecessary in future.

Today, APIs are an integral part in modern data-driven life science [27]. They allow programmatic and automatable access to data of any kind. In terms of smooth interoperability between different institutions and their data-holding systems, standardized and community-accepted APIs are of particularly central importance. One such standardized API in the field of plant breeding is the Breeding API specification project (BrAPI) [28]. BrAPI, as an attempt to establish a widely accepted application programming interface for plant breeding applications and related use cases, is under ongoing development by a global community of scientists of institutes and companies related to plant sciences and plant breeding [28]. Therefore, it will also be a great opportunity to integrate API calls of BrAPI related to genotypic data to support and improve the interoperability between different existing omics-related information systems and data warehouses for germplasm, genebank material and breeding material. Another interesting feature would be to enhance DivBrowse to be capable of reading data from BrAPI

endpoints that serve genotypic data. Thus, DivBrowse could deliver genotypic data through its server-side component via BrAPI, and consume and visualize genotypic data via an external BrAPI endpoint through the client-side GUI.

One major limitation is that only single nucleotide polymorphisms can be visualized. It would also be advantageous to visualize structural variations such as insertions, deletions, copy number variations and translocations. However, at the moment of writing this paper, the VCF specifications regarding structural variations were still under revision by the specification maintainers [29]. A new, stable version of the VCF specification regarding structural variations is planned for version 4.4, which has not yet been released at the time of writing this publication.

Another limitation of DivBrowse is that it can currently only visualize haploid and diploid genomes. In the case of allopolyploidy, it is possible in most cases to dissect the genome into diploid subgenomes that can be then visualized as independent chromosomes. For example, the hexaploid wheat genome is such a case, because its genome consists of three diploid subgenomes with 7 chromosomes each [30]. The wheat genome can thus be visualized with DivBrowse by dividing it into 21 diploid chromosomes. But there are, of course, plant cultivars or animal species that have true higher ploidy levels. As a consequence, there is still potential for future research about how genomes with higher levels of ploidy can be visualized in a meaningful and effective way.

A possible future extension regarding integration with existing software could be, for example, to make DivBrowse available as a plugin for Galaxy [31], so that genomic diversity data can be visualized and analyzed directly within a Galaxy workflow.

Use cases and workflow scenarios

Use case "A tool to visualize data of genebank genomics projects"

Genebank genomics projects create a vast amount of genotypic and phenotypic data [9,23]. DivBrowse can be one particular tool to serve and visualize the genotypic diversity stored in genebanks and make this data foundation easily and interactively accessible from every personal computer with an internet connection and an installed web browser. This reduces the barrier to accessing genotypic diversity data in general and makes it easier and faster for non-bioinformaticians to access critical information. As a reusable and configurable application that uses standardized and established bioinformatic file formats like VCF and GFF3 it makes the installation and setup of instances for many different plant species easy and affordable. The deployment of multiple DivBrowse instances for different plant species can be automated by customized Shell scripts or by pipelines for continuous integration and deployment based on GIT repositories [32,33].

Use case "Plugin in existing web-based information systems"

While DivBrowse can be used as a standalone tool, it also plays very well if being integrated in existing web applications that are focused on visualization and analysis of omics-data. One example for such integration is the BRIDGE web portal (https://bridge.ipk-gatersleben.de), where collections of germplasm derived from a search by passport attributes can be directly and seamlessly transferred to the integrated DivBrowse instance, to only visualize the variant calls of those genotypes derived from the previously defined collection of germplasm [10]. Conversely, it is possible to create new germplasm collections from within the DivBrowse plugin by transferring corresponding sample-IDs to the BRIDGE web portal via an API call from DivBrowse. As an example, users are able to create new germplasm collections in the BRIDGE web portal by selecting genotypes in the result of an interactive PCA via a lasso selection in DivBrowse. We can imagine that likewise other existing web applications in the

field of plant genomics, like Gigwa, Germinate, Ensembl or GrainGenes, could be functionally enriched in a similar way using DivBrowse as a plugin for variants visualization and analysis [34–37].

Use case "Fast insights into the genomic diversity of genes and genomic regions"

Geneticists and other scientists interested in available genetic variants of a specific gene, can use DivBrowse to get insights about how many variants for a gene or a genomic range are available and which genotypes are carrying specific alleles. This could be useful to support the understanding of the genetic diversity of specific traits or to get an overview about the genotypic diversity of a specific genomic range. Furthermore, if the genotypic data are of high enough resolution, DivBrowse allows the calculation of a principal component analysis to gain initial insights into the population structure of a gene or genetic feature. The integrated BLAST entry point is useful to find variants based on a given nucleotide or protein sequence [17]. The variants within the genomic region determined by a BLAST can then be quickly and easily exported as a VCF file and are thus available for further processing steps within a genome editing workflow. This feature can support the design of single-guide RNA in a CRISPR/Cas9 genome editing experiment [38,39].

References

1. Christensen K, Dukhovny D, Siebert U, Green R. Assessing the Costs and Cost-Effectiveness of Genomic Sequencing. *J Pers Med*. 2015; doi: 10.3390/jpm5040470.

2. Bayle A, Droin N, Besse B, Zou Z, Boursin Y, Rissel S, et al.. Whole exome sequencing in molecular diagnostics of cancer decreases over time: evidence from a cost analysis in the French setting. *Eur J Health Econ*. 2021; doi: 10.1007/s10198-021-01293-1.

3. Korte A, Farlow A. The advantages and limitations of trait analysis with GWAS: a review. *Plant Methods*. 2013; doi: 10.1186/1746-4811-9-29.

4. Stephens ZD, Lee SY, Faghri F, Campbell RH, Zhai C, Efron MJ, et al.. Big Data: Astronomical or Genomical? *PLOS Biol*. 2015; doi: 10.1371/journal.pbio.1002195.

5. Grüning BA, Lampa S, Vaudel M, Blankenberg D. Software engineering for scientific big data analysis. *GigaScience*. 2019; doi: 10.1093/gigascience/giz054.

6. Danecek P, Auton A, Abecasis G, Albers CA, Banks E, DePristo MA, et al.. The variant call format and VCFtools. *Bioinformatics*. 2011; doi: 10.1093/bioinformatics/btr330.

7. Stein L. Generic Feature Format Version 3 (GFF3).

8. Mascher M. Variant matrices for a global barley diversity panel. 2018; doi: 10.5447/IPK/2018/9.

9. Milner SG, Jost M, Taketa S, Mazón ER, Himmelbach A, Oppermann M, et al.. Genebank genomics highlights the diversity of a global barley collection. *Nat Genet*. 2019; doi: 10.1038/s41588-018-0266-x.

10. König P, Beier S, Basterrechea M, Schüler D, Arend D, Mascher M, et al.. BRIDGE – A Visual Analytics Web Tool for Barley Genebank Genomics. *Front Plant Sci*. 2020; doi: 10.3389/fpls.2020.00701.

11. Monat C, Padmarasu S, Lux T, Wicker T, Gundlach H, Himmelbach A, et al.. TRITEX:

chromosome-scale sequence assembly of Triticeae genomes with open-source tools. *Genome Biol*. 2019; doi: 10.1186/s13059-019-1899-5.

12. de Mast J, Kemper BPH. Principles of Exploratory Data Analysis in Problem Solving: What Can We Learn from a Well-Known Case? *Qual Eng*. 2009; doi: 10.1080/08982110903188276.

13. Cingolani P, Platts A, Wang LL, Coon M, Nguyen T, Wang L, et al.. A program for annotating and predicting the effects of single nucleotide polymorphisms, SnpEff: SNPs in the genome of Drosophila melanogaster strain w ^{1118} ; iso-2; iso-3. *Fly (Austin)*. 2012; doi: 10.4161/fly.19695.

14. Miles A, Jakirkham, Bussonnier M, Moore J, Fulton A, Bourbeau J, et al.. zarr-developers/zarr-python: Zenodo;

15. Miles A, Pyup.Io Bot, Murillo R, Ralph P, Harding N, Pisupati R, et al.. cggh/scikit-allel: v1.3.3. Zenodo;

16. Ben-Kiki O, Evans C, döt Net I: YAML™ Specification Index. https://yaml.org/spec/ (2009). Accessed 2022 Jul 1.

17. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. *J Mol Biol*. 1990; doi: 10.1016/S0022-2836(05)80360-2.

18. Sloggett C, Goonasekera N, Afgan E. BioBlend: automating pipeline analyses within Galaxy and CloudMan. *Bioinformatics*. 2013; doi: 10.1093/bioinformatics/btt199.

19. Cock PJA, Chilton JM, Grüning B, Johnson JE, Soranzo N. NCBI BLAST+ integrated into Galaxy. *GigaScience*. 2015; doi: 10.1186/s13742-015-0080-7.

20. Nei M. Molecular Evolutionary Genetics. Columbia University Press;

21. Binns D, Dimmer E, Huntley R, Barrell D, O'Donovan C, Apweiler R. QuickGO: a web-based tool for Gene Ontology searching. *Bioinformatics*. 2009; doi: 10.1093/bioinformatics/btp536.

22. Harris R. Svelte.

23. Mascher M, Schreiber M, Scholz U, Graner A, Reif JC, Stein N. Genebank genomics bridges the gap between the conservation of crop diversity and plant breeding. *Nat Genet*. 2019; doi: 10.1038/s41588-019-0443-6.

24. Wilkinson MD, Dumontier M, Aalbersberg IjJ, Appleton G, Axton M, Baak A, et al.. The FAIR Guiding Principles for scientific data management and stewardship. *Sci Data*. 2016; doi: 10.1038/sdata.2016.18.

25. Beier S, Fiebig A, Pommier C, Liyanage I, Lange M, Kersey PJ, et al.. Recommendations for the formatting of Variant Call Format (VCF) files to make plant genotyping data FAIR. *F1000Research*. 2022; doi: 10.12688/f1000research.109080.2.

26. Courtot M, Gupta D, Liyanage I, Xu F, Burdett T. BioSamples database: FAIRer samples metadata to accelerate research data management. *Nucleic Acids Res*. 2022; doi: 10.1093/nar/gkab1046.

27. Woody SK, Burdick D, Lapp H, Huang ES. Application programming interfaces for knowledge transfer and generation in the life sciences and healthcare. *Npj Digit Med*. 2020; doi: 10.1038/s41746-020-0235-5.

28. Selby P, Abbeloos R, Backlund JE, Basterrechea Salido M, Bauchet G, Benites-Alfaro OE, et al.. BrAPI—an application programming interface for plant breeding applications. Wren J, editor. *Bioinformatics*. 2019; doi: 10.1093/bioinformatics/btz190.

29. Cameron D: Improved Structural Variant Support by d-cameron · Pull Request #465 · samtools/hts-specs. https://github.com/samtools/hts-specs/pull/465 (2019). Accessed 2022 Jul 8.

30. The International Wheat Genome Sequencing Consortium (IWGSC), Appels R, Eversole K, Stein N, Feuillet C, Keller B, et al.. Shifting the limits in wheat research and breeding using a fully annotated reference genome. *Science*. 2018; doi: 10.1126/science.aar7191.

31. Goecks J, Nekrutenko A, Taylor J, Galaxy Team T. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biol*. 2010; doi: 10.1186/gb-2010-11-8-r86.

32. Perez-Riverol Y, Gatto L, Wang R, Sachsenberg T, Uszkoreit J, Leprevost F da V, et al.. Ten Simple Rules for Taking Advantage of Git and GitHub. Markel S, editor. *PLOS Comput Biol*. 2016; doi: 10.1371/journal.pcbi.1004947.

33. Gruening B, Sallou O, Moreno P, da Veiga Leprevost F, Ménager H, Søndergaard D, et al.. Recommendations for the packaging and containerizing of bioinformatics software. *F1000Research*. 2019; doi: 10.12688/f1000research.15140.2.

34. Sempéré G, Pétel A, Rouard M, Frouin J, Hueber Y, De Bellis F, et al.. Gigwa v2—Extended and improved genotype investigator. *GigaScience*. 2019; doi: 10.1093/gigascience/giz051.

35. Raubach S, Kilian B, Dreher K, Amri A, Bassi FM, Boukar O, et al.. From bits to bites: Advancement of the Germinate platform to support prebreeding informatics for crop wild relatives. *Crop Sci*. 2021; doi: 10.1002/csc2.20248.

36. Cunningham F, Allen JE, Allen J, Alvarez-Jarreta J, Amode MR, Armean IM, et al.. Ensembl 2022. *Nucleic Acids Res*. 2022; doi: 10.1093/nar/gkab1049.

37. Yao E, Blake VC, Cooper L, Wight CP, Michel S, Cagirici HB, et al.. GrainGenes: a data-rich repository for small grains genetics and genomics. *Database*. 2022; doi: 10.1093/database/baac034.

38. Wang H, La Russa M, Qi LS. CRISPR/Cas9 in Genome Editing and Beyond. *Annu Rev Biochem*. 2016; doi: 10.1146/annurev-biochem-060815-014607.

39. Jiang F, Doudna JA. CRISPR–Cas9 Structures and Mechanisms. *Annu Rev Biophys*. 2017; doi: 10.1146/annurev-biophys-062215-010822.

Availability of Source Code and Requirements

The source code and documentation of DivBrowse is available under the MIT license at https://github.com/IPK-BIT/divbrowse. A ready-to-install Python package is available on the Python Package Index at https://pypi.org/project/divbrowse/. An OCI-compliant container image is available on DockerHub at https://hub.docker.com/r/ipkbit/divbrowse. Several demo instances with human, mouse and barley data sets are listed at https://divbrowse.ipk-gatersleben.de.

- Project name: DivBrowse
- Project home page: https://divbrowse.ipk-gatersleben.de
- Code repository: https://github.com/IPK-BIT/divbrowse
- Operating system: Linux, macOS, Windows
- Programming language: Python 3.9, JavaScript
- Other optional requirements: conda, docker or podman
- License: MIT
- biotools ID: divbrowse

Acknowledgements

We thank J. Bauernfeind, T. Münch and H. Miehe for the administration of the IT infrastructure which is used to host the demo instances.

Author Contributions

N.S., M.M., U.S. designed the study. N.S. supervised the experiments and M.M. the data analysis for the barley data sets. P.K. designed and developed application software. P.K. wrote the manuscript with contributions from all co-authors. All authors read and approved the final manuscript.

Funding

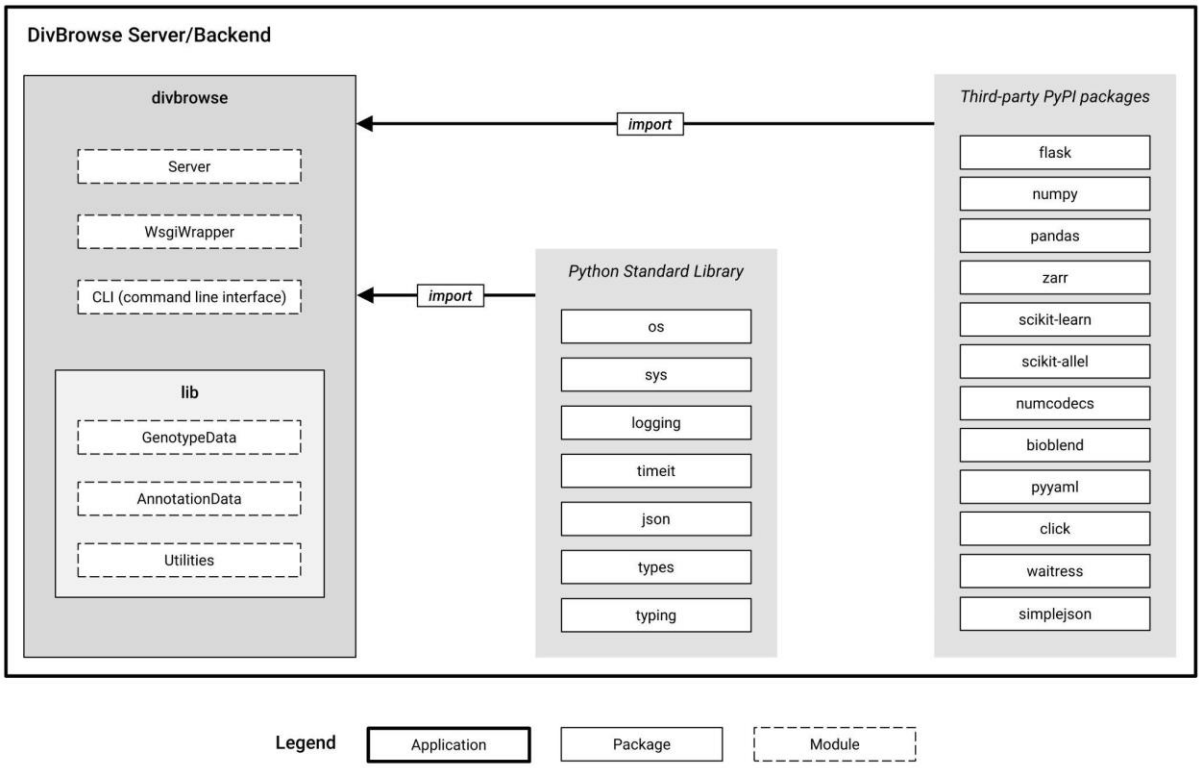
This work was supported by the Leibniz Association [Pakt für Forschung und Innovation: SAW-2015-IPK-1 'BRIDGE' to U.S., M.M., N.S.] and the German Ministry of Education and

Research (BMBF) [grant 031A536A ‘de.NBI’ to U.S., grant 031B0884A ‘SHAPE-II’ to N.S., U.S. and M.M.].

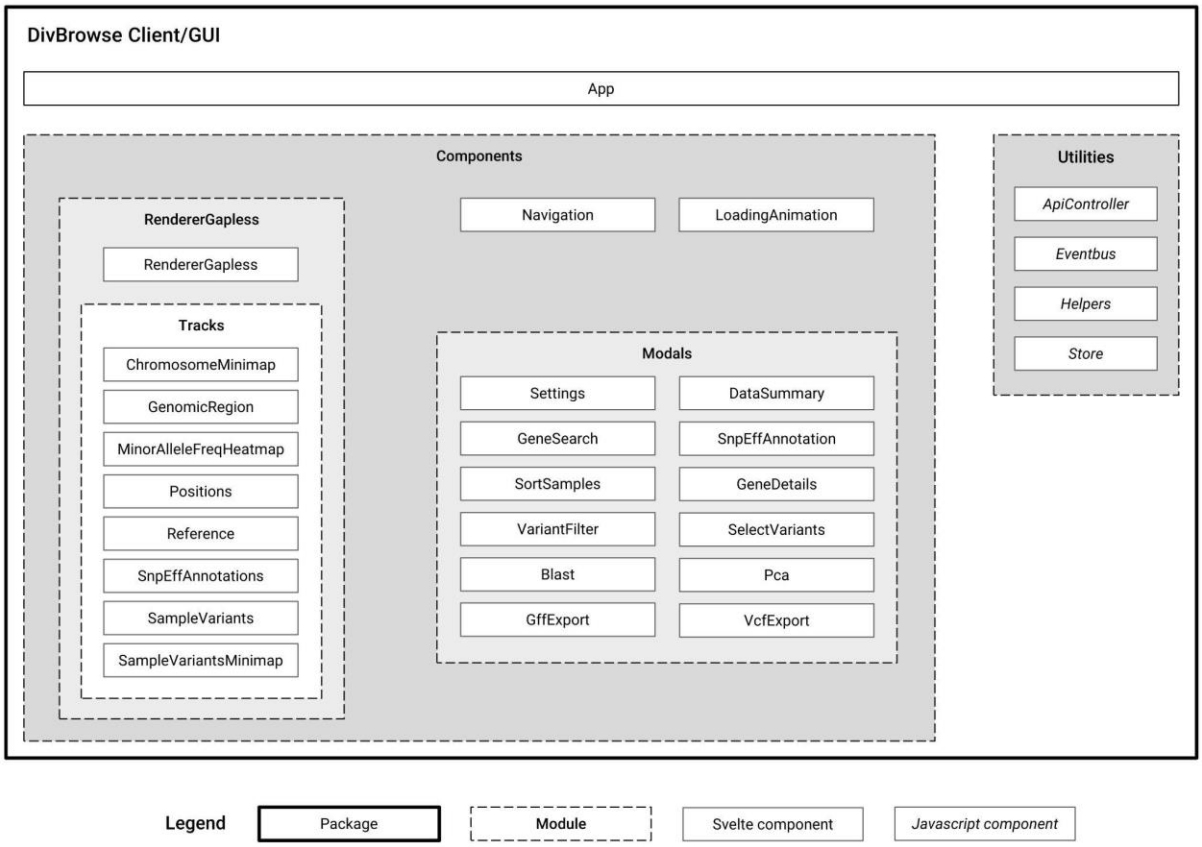
Conflict of Interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Supplementary Material



Supplementary Figure 1: The architecture of the DivBrowse server. The divbrowse package imports and uses numerous packages from the Python Standard Library as well as third-party packages that are published on the Python Package Index (PyPI / www.pypi.org)



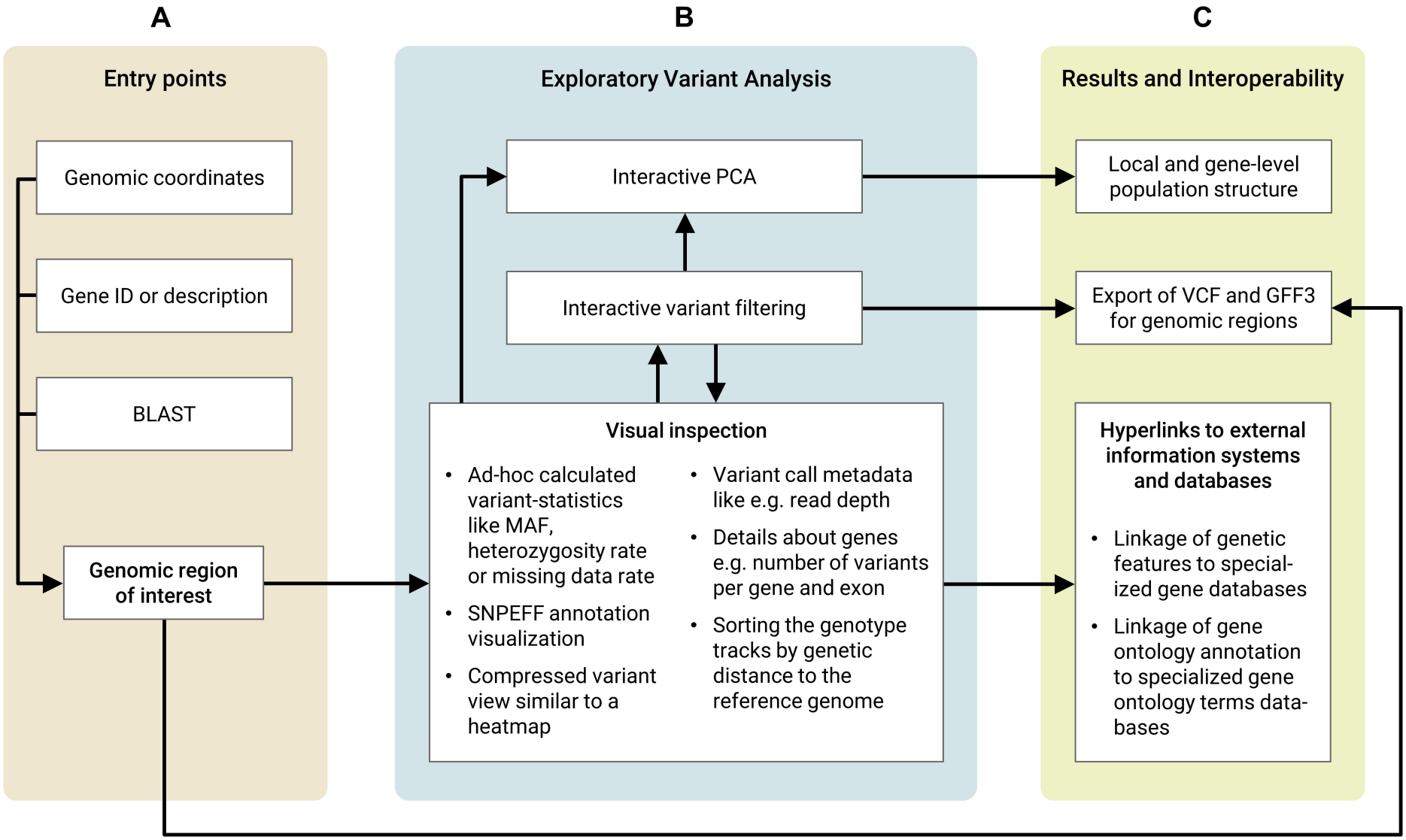
Supplementary Figure 2: The architecture of the DivBrowse client. The main entrypoint is the Svelte component "App" which holds the basic layout of the GUI which itself is divided into multiple modules and components. There are two main top-level modules: Components and Utilities. Components inside of the Components-module are all Svelte components that consist of logic directly responsible for the presentation layer of the GUI. Components inside of the Utilities-module are plain Javascript components that consist of cross-component business logic (like the ApiController) and helper functions.

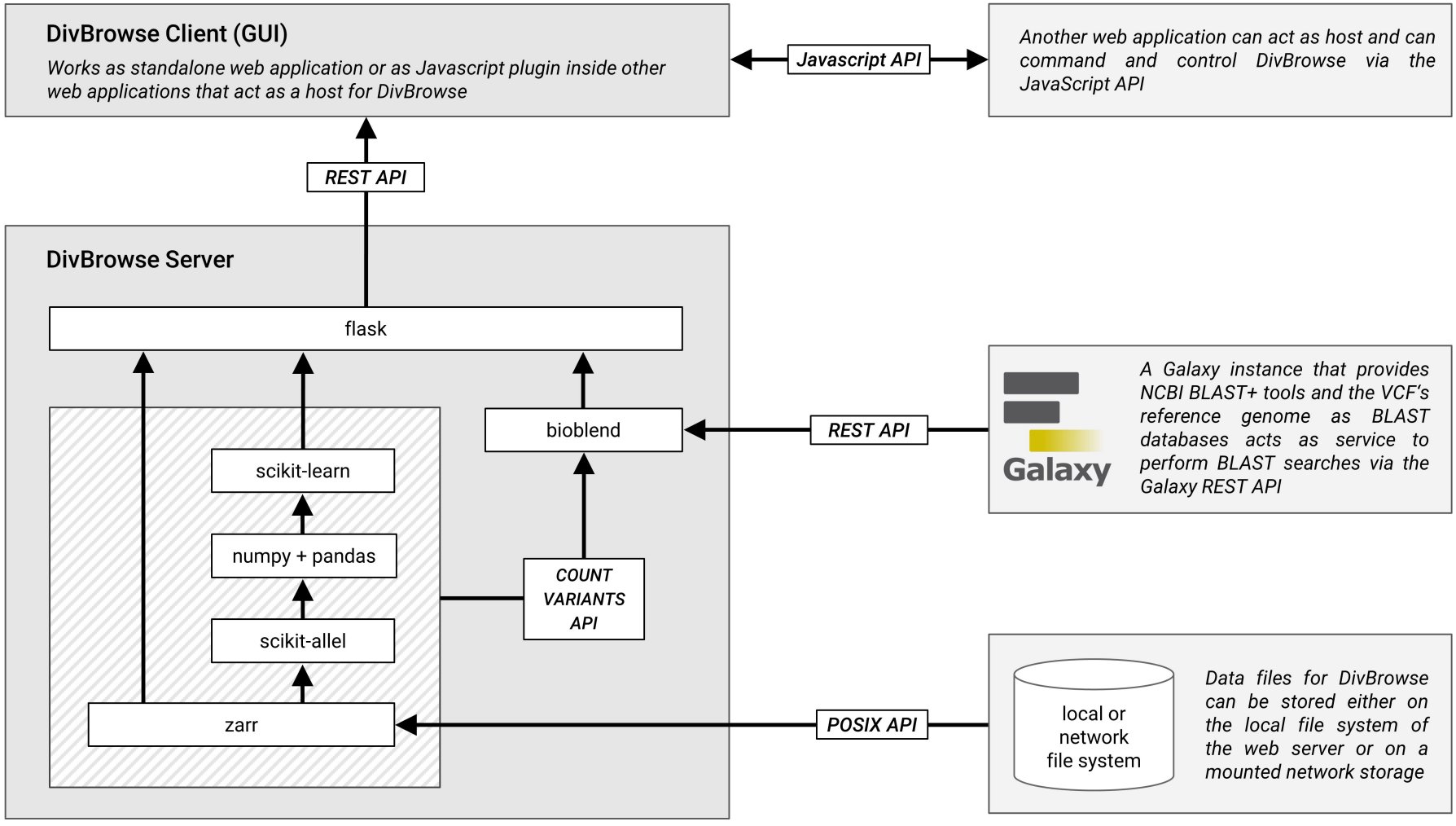
REST Resource URL	HTTP method	Purpose
/configuration	GET	The GUI which itself is implemented agnostically derives important metadata and configuration settings to setup itself automatically to the corresponding DivBrowse server component.
/genes	GET	This API-call delivers all genes and genetic features. The data is loaded only once at the start of the user's session.
/variants	POST	Delivers variants and metadata
/variant_calls	POST	Delivers variant calls and metadata on a per call level
/genomic_window_summary	POST	Statistical summary about the variants within a user defined genetic region/window
/pca	POST	Performs a PCA according the user's input and returns the calculation result
/vcf_export_check	POST	Checks whether the VCF export with the user's input parameters is possible or not

Supplementary Table 1: Information about the resources of the REST-API

API Javascript method	Purpose
AppInstanceObj.setSamples(*arg*)	Set a list of sample-IDs that should be visible in the genotypes track. The function argument "*arg*" can hold either: ● An array of sample-IDs that are then also used as genotype labels in the genotype tracks. ● An array of objects where each object is a map with two entries "id" and "link". In this case "id" holds the sample-ID of a genotype and "link" holds an "<a>"-HTML-tag that acts as the genotype label in the genotype tracks. This way, genotype labels can act as linked labels to e.g. link to another separate web application. ● An array of objects where each object is a map with two entries "id" and "displayName". In this case "id" holds the sample-ID of a genotype and "displayName" holds the corresponding genotype label in the genotype tracks. This way, the visible genotype labels can differ from the sample-IDs.
ConfigObj.samplesSelectedCallback(sampleIds)	Callback function, that will be called after a lasso selection in a PCA result. The callback function gets called with an argument which holds an array of the sample-IDs of the lasso-selected genotypes.

Supplementary Table 2: Information about the methods of the Javascript-API


A
B
C
Entry points
Genomic coordinates
Gene ID or description
BLAST
Genomic region of interest
Exploratory Variant Analysis
Interactive PCA
Interactive variant filtering
Visual inspection
Ad-hoc calculated variant-statistics like MAF, heterozygosity rate or missing data rate
SNPEFF annotation visualization
Compressed variant view similar to a heatmap
Variant call metadata like e.g. read depth
Details about genes e.g. number of variants per gene and exon
Sorting the genotype tracks by genetic distance to the reference genome
Results and Interoperability
Local and gene-level population structure
Export of VCF and GFF3 for genomic regions
Hyperlinks to external information systems and databases
Linkage of genetic features to specialized gene databases
Linkage of gene ontology annotation to specialized gene ontology terms databases


DivBrowse Client (GUI)
Works as standalone web application or as Javascript plugin inside other web applications that act as a host for DivBrowse
Javascript API
Another web application can act as host and can command and control DivBrowse via the JavaScript API
REST API
DivBrowse Server
flask
scikit-learn
numpy + pandas
scikit-allel
zarr
bioblend
COUNT VARIANTS API
REST API
Galaxy
A Galaxy instance that provides NCBI BLAST+ tools and the VCF's reference genome as BLAST databases acts as service to perform BLAST searches via the Galaxy REST API
POSIX API
local or network file system
Data files for DivBrowse can be stored either on the local file system of the web server or on a mounted network storage