



Stockholms
universitet

Genome visualization tool for
general use, Input Generated
Genome Browser

Danilo Catalan Canales

Handledare: Lars Arvestad

Examinator: Marc Hellmuth

Inlämningsdatum: January 2, 2024

Abstract

This study presents the development of a gene visualization tool called Input Generated Gene Browser aimed at providing a user friendly gene browser. The tool addresses the limitations of some gene browsers by letting users input their own data for visualization. The focus of the tool is on visualizing genomes, which represents the presence of multiple genes on the same chromosome. The tool utilizes pillars and tracks, where pillars represent homologous genes and tracks represent genome segments. The visualization algorithms employed in the tool allow for the analysis of gene order and synteny. The implementation of the tool utilizes the React framework and Next.js, with deployment facilitated by Vercel. The results demonstrate the successful creation of a gene browser with user friendly features for visualizing genetic information. The tool would offer flexibility and usability for bioinformatics and scientists to take advantage of.

Sammanfattning

Denna studie presenterar utvecklingen av ett genvisualiseringsverktyg som kallas Input Generated Gene Browser. Detta verktyg syftar till att vara ett användarvänligt visualiseringsverktyg. Studien pekar ut begränsningar hos vissa genvisualiseringsverktyg genom att erbjuda användare möjligheten att kunna mata in sina egna data för visualisering. Verktyget fokuserar på att visualisera genomisk synteni, vilket innebär närvaron av flera gener på samma kromosom utan behov av information om relativ order eller avstånd. Verktyget använder så kallade "pillars" och "tracks", där "pillars" representerar homologa gener och "tracks" representerar genomsegment. Visualiseringsverktyget möjliggör analys av genordning och synteni. Implementeringen av verktyget använder sig av React och Next.js samt Vercel som ger möjlighet att framställa webbsidan för visualiseringsverktyget. Resultatet visar att det har skapats en lyckad "genome browser" med användarvänliga funktioner för visualisering av genetisk information. Verktyget erbjuder flexibilitet och användbarhet som forskare och bioinformatiker kan använda sig av.

Contents

1	Introduction	6
1.1	Background	6
1.2	Problem Description	7
1.3	Aim	7
1.4	Delimitations	8
1.5	Outline	8
2	Theory	9
2.1	Synteny	9
2.2	Homologs	10
	2.2.1 Paralogs, whole genome duplication & Ohnologs	10
2.3	Data structure	10
	2.3.1 Pillar & order file	11
2.4	React Framework, Next.js and Vercel	13
2.5	Visualization Algorithms	15
	2.5.1 Ordering by order file	15
	2.5.2 Displaying without order file	16
	2.5.3 Hex coloring	17
3	Method	18
3.1	Web Application	18
3.2	Layout design	18
3.3	Data management	21
	3.3.1 Test Data	22
3.4	Styling the output	22
3.5	Study Limitations	22
4	Results	23
4.1	User input	23
4.2	Gene browser	25
	4.2.1 Filter features & information display	26

4.3	React Development	27
4.4	Design changes	28
5	Conclusion	29
5.1	Discussion	29
5.2	Evaluation	29

1 Introduction

Gene browsers today offer the ability to visualize genetic information, but they often lack compatibility with data from other sources, which could be an important feature missing in some of the gene browsers.

1.1 Background

Bioinformatics involves using computer technology to handle biological data. By organizing and visualizing genes, scientists can potentially improve our comprehension of biological data. [1].

Gene browsers are a type of software that helps visualize sequences of genetic information in a relevant manner to the user. Such software could have a number of applications to assist the user analyze different strains of genetic information [2].

A gene browser created by Byrne and Wolfe [3] focuses on facilitating visual comparison and computational analysis of **synteny relationships** in yeast. It is called **yeast gene order browser**, or **YGOB**, and it's based on a homologous set of genes that have been collected for their respective sequence similarity and synteny. Homologous genes are kept in a matrix called **pillars**. YGOB examines consequences due to an event called **whole genome duplication**, or **WGD**, by visualizing genomes of species pre- and post-WGD. Whole genome duplication implies that a gene copies and attaches itself to the genome. Pre-WGD refers to species that existed before undergoing a whole-genome duplication (WGD) event, while Post-WGD refers to species that have undergone a WGD event in their

evolutionary history.

Even though there are several genome browsers, I will use YGOB as reference as it misses the feature of having user input to their browser. I will also be using their database as input reference for this project.

1.2 Problem Description

While gene browsers today offer valuable visualization capabilities for genetic information, their limited compatibility hampers their widespread usability. This issue prevents researchers and users from efficiently accessing and utilizing these tools for various genetic analysis tasks. Ensuring general compatibility for gene browsers is necessary to enable broader access and utilization of these tools by researchers and scientists from diverse disciplines. It would allow for more efficient and refined workflows, as researchers could seamlessly integrate gene browsers into their existing computational pipelines. Overall, improving compatibility for general use in gene browsers is essential for enhancing their usability and empowering researchers to explore and analyze genetic information more effectively.

1.3 Aim

The aim of this thesis is to create a gene browser, similar to Byrne and Wolfe's Yeast Gene Order Browser, where the key feature of the browser is for the user to provide data to be visualized. This is to benefit a broader use when it comes to visualizing data which could be made by independent users.

1.4 Delimitations

This study will focus on creating a usable gene browser along with two key features. The genome browser will display genes provided by the user and their gene order. The two features will be for the user to decide the length of the tracks displayed and order the genes after a focused gene. Most of the user experience and visual design will be kept as simple as possible to make it easy to use, maintain a clear focus on visualizing the gene data and overall optimize the user experience.

1.5 Outline

This study contains four sections. Section 2 will present necessary theory required to comprehend the visualization aspects of the browser, including the structure of user input and the components utilized to create the gene browser. Section 3 will discuss the approach of different components and some design choices. The last section will talk about the final result in addition to discussing positive and possible negative outcomes of the created gene browser.

2 Theory

In this chapter I will discuss further on the theory behind what will be visualized, data structures and the reasoning behind it. Lastly, I will briefly explain what React and Vercel is.

2.1 Synteny

Synteny is an important concept due to the context it provides for the visualization tool. **Gene synteny** is defined as the presence of two or more genes on the same chromosome of a given species, where no information of relative order or distance between them is necessary [4]. It's specifically talked about in this manner for bioinformatics, but the term is used slightly differently in genomics where it refers to the conservation of the gene order and organization within a single genome.

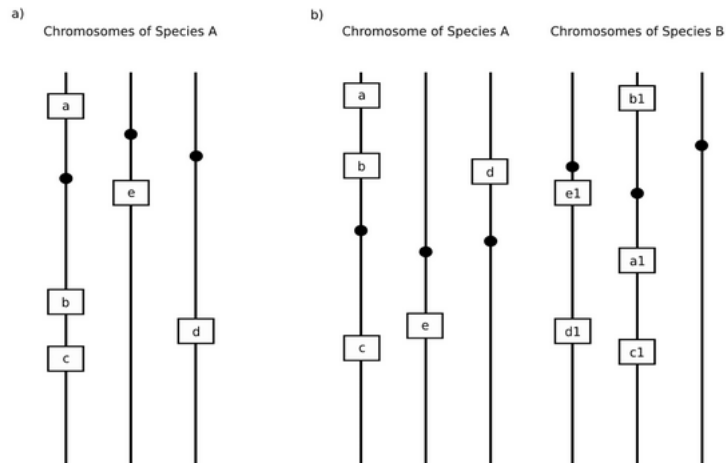


Figure 1: Each line represents a chromosome and each dot separates a region. a) Genes a,b and c are syntenic since they are present in the same chromosome however genes e and d are not. b) Genes a/a1, b/b1 and c/c1 are syntenic in both species A and B.

As can be observed in Figure 1 synteny does not necessarily imply order, but what is important is the synteny relationship shared among species. Synteny relationship is when two species share synteny between genes , as shown in

figure 1 b). In Figure 1 we can observe the different synteny types that will be visualized in the browser that is to be created.

2.2 Homologs

Homologs are genes that share a common ancestor. These gene sequences can be related in two ways. Firstly, when a species splits into two different species through speciation, the related gene sequences are called orthologs. Secondly, gene duplication occurs when a segment, the entire gene, chromosome or genomes duplicates and integrates into the chromosome, resulting in duplicated gene sequences known as paralogs.

2.2.1 Paralogs, whole genome duplication & Ohnologs

When a gene has been duplicated, the resulting two copies are called **paralogs**. Whole genome duplication replicates the complete genetic information [5]. Through evolution, whole genome duplication has played an important role when it comes to the development of organisms. **Ohnologs** represent the paralogous genes as a result from WGD. Ohnologs, as well as paralogs, can provide a genetic basis for rapid evolutionary change by creating redundancy in the genome, allowing for genetic innovations and functional variety.

2.3 Data structure

The study bases its algorithms on the YGOB data structure, therefore, I will first go through the **pillar** data originally used in YGOB before talking about the input format that is used in this study.

The pillar file from YGOB contains a matrix where the rows

are homologous genes shared among the species and the columns represent the species. The homologous genes on the rows are represented by a unique identifier, also called accessions, to the species. The pillar's assignment is to store homology across the different species. The gene occupies the slot for each species if it has it and if it doesn't it stays vacant [3], as shown in Table 1. The columns are called tracks and the rows are called pillars.

- - -	g1	g1	g1
g2	- - -	g2	- - -
g3	- - -	- - -	g3

Table 1: The table displays how the pillars and tracks work. Every row is a pillar which holds a gene identifier and displays a gene across the different columns. The columns are the tracks representing the species. If the species doesn't have the gene it is left blank, represented by the three hyphens.

2.3.1 Pillar & order file

There should be two types of input for the browser: A pillar input and an order input. For every species, or **track**, there should be one order file input with their respective species name as file name to match with the track from the pillar input.

The pillar file for the pillar input should resemble the pillar file from YGOB, but distinct itself in the first row. The first row should hold the name of every species to tell the algorithm which species is in what track, or column, as shown in Table 2. The remaining rows represent a pillar, as shown in Table 1.

Species 1	Species 2	Species 3	Species N
- - -	gene 1	- - -	gene 1
gene 2	- - -	gene 2	- - -
gene N	gene N	gene N	gene N

Table 2: An example showing how a pillar file should look like. First row is the name of each species. The following rows display the genes one at a time. If the species has the gene it will be displayed in the same row. If it doesn't have the gene it will be replaced by three hyphens.

The order file should contain the gene order for the species, accompanied by additional information about each gene. This information is limited to three different slots. The first slot is the name of the gene identifier, the second tells us which **gene assembly identifier** (chromosome, scaffold or contig) it's has and the third tells about the gene orientation, as shown in Table 3.

Gene Name	Chromosome, Scaffold or Contig Name	Orientation (W=Watson, C=Crick)
-----------	-------------------------------------------	---------------------------------------

Table 3: The table shows how the information should be placed on a order file for every gene.

Chromosomes are a type of structure made of DNA that carries genetic information. A chromosome is sequenced and assembled into chromosome assemblies. The assemblies are made up of smaller sequences called **contigs** and supporting structures called **scaffolds**. A contig is a contiguous DNA sequence which is assembled from smaller DNA fragments, or **reads**, obtained during DNA sequencing. Scaffolds support the assembly by organizing and connecting contigs to create higher-level representations of genomes. Both contigs and scaffolds are essential to form a genome assembly.

In DNA sequencing the orientation refers to the direction or alignment of DNA strands. The two orientations are **forward orientation**, also called **Watson Orientation**

or sense, and reverse orientation, also called **Crick Orientation** or **antisense**. These two strands are complementary DNA strands that make up the double helix structure [6].

Gene 1	Chr 1	W
Gene 2	Chr 1	C
Gene 3	Chr 2	—

Table 4: An example showing how an order file could look like. First column is the name of the gene, the second column is which section of the DNA the gene is positioned and the last column tells us the orientation.

The order file should have the same structure as Table 4. The genes should be placed in order and each column should be filled out respectively and if the information isn't available it should be filled out by three hyphens.

2.4 React Framework, Next.js and Vercel

Next.js is an open-source framework for building server-side rendered and statically generated React applications. It's built upon React and Node.js, which is an open-source and server-side runtime environment. React is a JavaScript library used for building user interfaces in **web applications**. It has a component-based structure, which allows for reusable UI components and building sophisticated user interfaces. Some of the key features that will be used for the study are components and hooks.

React applications are made up of components, which are self-contained and reusable. Components have their own states and properties which allow them to manage and pass along data. **Hooks** allow the use of different features from the react components. It provides a way to handle states, timed methods and side effects.

I will use Next.js to facilitate the process of making the gene order browser. Since Next.js is built upon React, I can

utilize its library to create and reuse components to avoid redundancy. By using components the browser could also store user data on the client side of the application to reduce unnecessary calls and storing data in a server. Next.js also facilitates in making the application available to the public by using Vercel.

Vercel is a cloud platform that aids in deploying web applications, with focus on React dependent projects, aiming to make deployment as easy as possible. This is done by connecting Vercel directly to a repository, specifically a repository created with GitHub. Since GitHub and Vercel have an affiliation they allow for deployment directly from the GitHub account to Vercel.

2.5 Visualization Algorithms

In this section I will discuss definitions and algorithms which will explain how the data is used for the visualization.

As mentioned in section 2.3, the pillar file is described as a matrix where each column is a species and rows are homologous genes shared among the species. I will denote the pillar **matrix P**, where the total amount of rows and columns are **M** and **N**, respectively. Users will be able to select which species, or columns, they would like to display. The chosen species would create a subset matrix of **P**, which I will denote **P'**. Since the algorithm will only display a section of the homologous genes, or a certain amount of rows, the user will be able to choose the maximum rows that they would like to display for each species creating a new subset **D** of the chosen species with rows **before** and **after** the focused gene. I will denote this constant **b**. Lastly, the user will be able to choose the species and gene which I will denote **s** and **g** respectively. The focused gene and species are represented by indices that indicates a specific column and row of the chosen species matrix, more on these different user selections on section 3.2.

2.5.1 Ordering by order file

The order relies on the order file input by the user, which I will denote **O**. The algorithm matches gene names with those in order file **O** and collect their indices. These collected indices will be used to order the rows of the sectioned matrix of the selected species in an ascending order. When the section has been successfully ordered the algorithm will display them as a new **matrix D'**. The genes of the ordered gene matrix get colored and then displayed. The genes get colored the same

if they are placed in the same gene assembly identifier in their respective species, mentioned in section 2.3.1. In other words, if a gene in species A is in chromosome 4 and a gene in species B is also in chromosome 4 they will be colored the same.

Algorithm 1 Order

```
1: P' is a matrix of chosen species.
2: O the order file for the focused species
3: s is the focused species index, indicating the column of matrix P'
4: g is the focused gene index, indicating the row of the matrix P'
5: b is max amount of homologous genes on display.
6: if g is smaller than b/2 then
7:   Collect all rows from 0 to b from P into a new matrix array D
8: else
9:   Collect all rows from g - b/2 to g + b/2 into a new matrix array D
10: end if
11: Organise rows of D based on O indices and return a new organized matrix D'
12: for each column (species gene set) c  $\in$  D' do
13:   if focused species then
14:     highlight species name
15:   end if
16:   for each gene  $\in$  c do
17:     if new assembly identifier then
18:       store color to assembly identifier
19:     end if
20:     Color gene depending on gene assembly component identifier or if
    focused gene
21:     Display gene at species c
22:   end for
23: end for
```

2.5.2 Displaying without order file

If no order file is input for the focused species, the algorithm will not order the homologous genes or color after an identifier, however, it will instead color the genes the same if they are in the same row in matrix **D**, highlighting the homologous genes between the species.

Algorithm 2 Without order

```
1: P' is a matrix of chosen species.
2: s is the focused species index, indicating the column of matrix P'
3: g is the focused gene index, indicating the row of the matrix P'
4: b is max amount of homologous genes on display.
5: if g is smaller than b/2 then
6:   Collect all rows from 0 to b from P into a new matrix array D
7: else
8:   Collect all rows from g - b/2 to g + b/2 into a new matrix array D
9: end if
10: for each column (species gene set) c  $\in$  D do
11:   if focused species then
12:     highlight species name
13:   end if
14:   for each gene  $\in$  c do
15:     if new row then
16:       Store color to specific row
17:     end if
18:     Color gene depending on row index in D or if focused gene
19:     Display gene at species c
20:   end for
21: end for
```

2.5.3 Hex coloring

To differentiate genes visually, I used random hexadecimal colors for homologous genes. Hexadecimal is a base-16 number system, using six letters and numbers (0-9, A-F). Genes contained in the same chromosome, scaffold or contig share the same color. Otherwise, genes are colored based on their pillar. Every new genome assembly component identifier, like chromosome 3 or contig 1c123f, would generate a new color and reserve it for that specific identifier.

3 Method

In this Chapter we will discuss more in depth about the approach and design decisions to get to the final browser application.

3.1 Web Application

The development of the web app followed a systematic process involving several steps. First by focusing on creating the design without any functionality. Once the design was implemented, the next part involved adding functionality and ensuring compatibility with the necessary data. Finally, the gene components were restyled to enhance visual comprehension of the data. Throughout the development process the project was regularly updated in a github repository [7].

3.2 Layout design

The web application will have two pages. One that will handle the inputs from a user and another will display the data provided.

The browser layout is chosen so that the user could get a clear view of the genes as well as having filters to manipulate the data. The filters would have the possibility to be hidden to improve the visibility of the genes displayed.

As shown in Figure 2, I decided to put a navigation bar at the top and a filter at the bottom. The navigation bar routes between the different pages within the browser, where the home-route should be where the genes are visualized, input-route where the user would input their data and about-route potential route where the user would find more information about the gene order browser. To make the design as simple as possible, I have chosen three filter

options for the user which are labeled **Area Size**, **species dropdown** and **gene dropdown**. The Area Size lets the user choose the maximum number of genes displayed on the screen, the species dropdown allows the user to be able to focus on specific species and gene dropdown would allow for the user to focus on a specific gene.

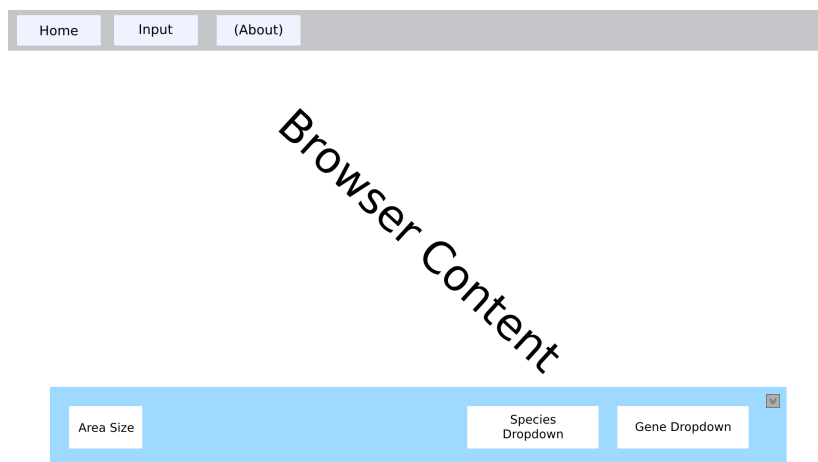


Figure 2: Each line represents a chromosome and each dot separates a region. a) Genes a,b and c are syntenic since they are present in the same chromosome however genes e and d are not. b) Genes a/a1, b/b1 and c/c1 are syntenic in both species A and B.

In Figure 3, each row represents a species, or track, and all homologous genes shared among the species, or pillars, are displayed on each column. For genes that match across species, the corresponding cells in each column would contain the gene name. If a gene is not present in a particular species, the corresponding cell would be left blank. Users can slide horizontally through the browser while the columns remain fixed, ensuring that the positional information of the homologous genes remain intact.

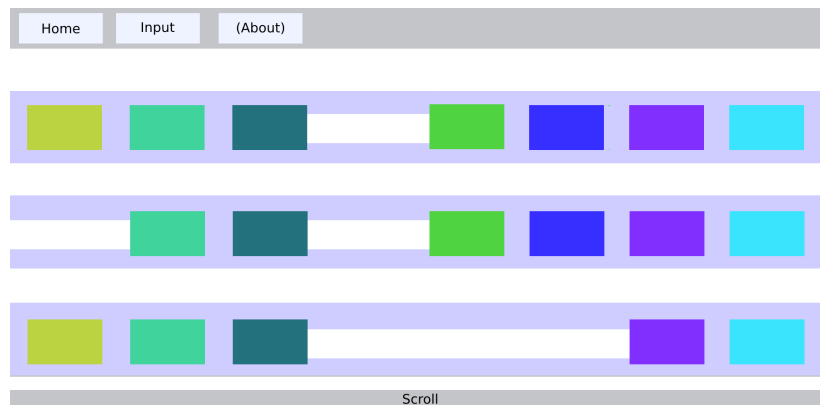


Figure 3: Sketch showing species as every row and columns as homologous genes shared among the species, shown without the filter component. White spaces between genes tell the relative distance between two genes.

Regarding the input design, the pillar and order data can be input separately. After the pillar file has been input the user should be able to see what species are available to be seen. This should be shown in the green section in Figure 4. The user is also able to choose what species to be shown in the browser or not, by having them marked. Once the user has made their selection of active species, they can submit their choices. After submitting, the user will be redirected to the first page to view their inputted data. The order data should be input for every species that is included in the pillar file, however, it is not necessary if it is only to display the pillars, as shown in Figure 3. It should be one file for every species, with the name of the species as file name so the algorithm can match the corresponding order file with species in the pillar file. The species with order file available will be marked on the species displayed on the species Selection box on Figure 4.



Figure 4: User input page showing where the different files go. The green rectangle to the right displays the species from the pillar file.

In all designs, I have utilized different colors to differentiate one element from another. It's important to note that the colors in the designs are not intended to be retained in the final design.

3.3 Data management

As mentioned, the data will be stored in a component which will be accessed by the whole application. Before being visualized, the input data undergoes an algorithm depending on which files the user provides. If the user chooses to only input a pillar file the data would only allow for the browser to display the selected species provided by the pillar file and display the genes from the tracks as they are, without adding any order relevance. However, by also adding the order files for the species the algorithm would be able to use the data to add additional information and relative order to the different genes displayed. Only one pillar should be active at a time. Any new pillar input should override the previous and reset the order input.

3.3.1 Test Data

The test data to use for the new gene browser was extracted from the YGOB and was manipulated to match the data structure presented in section 2.3. The extracted data was taken from a previous version of the YGOB, which matches the data for the order files on section 2.3.1.

3.4 Styling the output

The data output is presented to the user in a clear and simple manner. Depending on the availability of the pillar file and the order file for the species, there are two approaches. If both sources are available, the genes are color-coded based on their corresponding chromosome, providing information about the chromosome to which each gene belongs. Otherwise, if there are no order files, homologous genes are color-coded to indicate their respective track.

3.5 Study Limitations

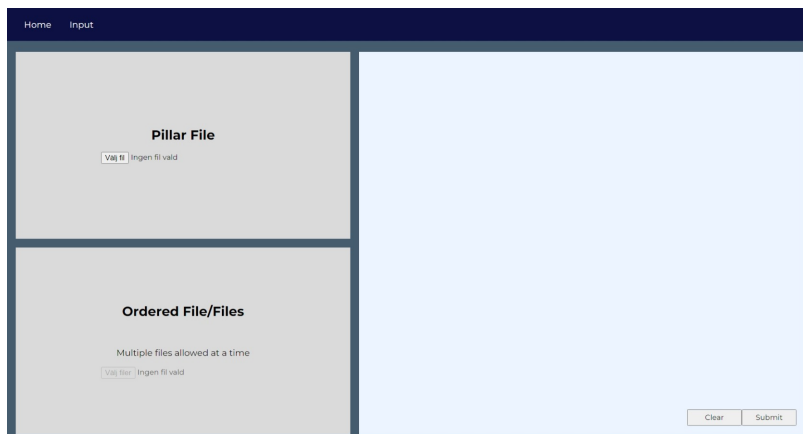
The gene browser is partially limited by the data input, specifically the format outlined in section 2.3.1, Table 2. It does not account for errors in the data, meaning that it will only visualize data that satisfy the correct data format. The genes in the homologous pillars should also match the genes on the order files to be able to give correct visualization and information.

4 Results

In this chapter we will discuss the results, comparing the designs with the outcome, the different features and how it compares to YGOB as reference. From this chapter onwards the created order browser will be mentioned as IGGB which stands for input generated gene browser.

4.1 User input

The user input displays clear instructions on where to add the input to their respective parts. The design has prioritized the size of the display window to the right of the display as shown on Figure 5. This was to allow more room for larger lists.



The screenshot shows a web interface with a dark blue header containing 'Home' and 'Input' links. The main content area is divided into three sections. On the left, there are two stacked grey boxes. The top box is titled 'Pillar File' and contains a file input field with a 'Välj fil' button and the text 'Ingen fil vald'. The bottom box is titled 'Ordered File/Files' and contains the text 'Multiple files allowed at a time' and another file input field with a 'Välj fil' button and the text 'Ingen fil vald'. On the right, there is a large, empty light blue rectangular area. At the bottom right of this area, there are two buttons: 'Clear' and 'Submit'.

Figure 5: User input page

After the user adds the pillars, the right-hand window will automatically display the first row containing the names of the species. These names will be highlighted to indicate that they have been selected to appear on the gene browser, as shown in Figure 6.

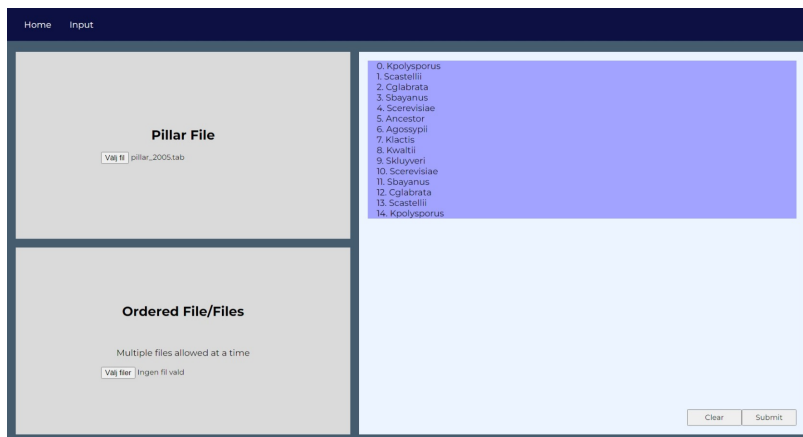


Figure 6: User input page.

For the gene order files I chose to put an asterisk next to the names to indicate which pillars have their gene order file available. In Figure 7 we can see a visualization when a pillar has an available gene order file and when species are deselected.

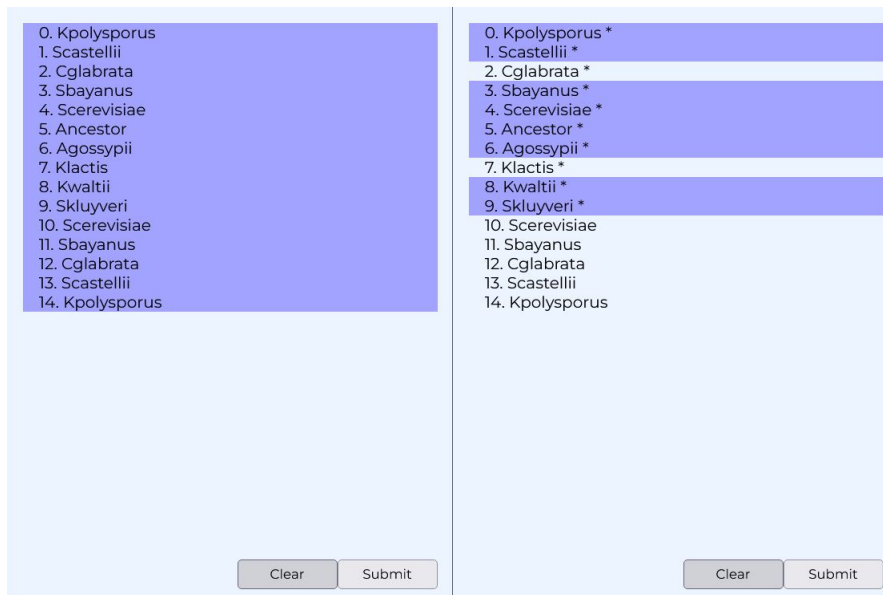


Figure 7: user input with pillar files and species selection.

After the user has made their selection of species, they may proceed with submitting their choices.

4.2 Gene browser

When entering the gene browser, users will be presented with a visual representation of homologous genes, colored according to their corresponding pillar. The genes are displayed as colored boxes, accompanied by their gene identifier and, if available, the source sequence identifier from the order file, positioned just above the name. The filters will show an initialized state, but more about the filter features in section 4.2.1 and how it will affect the browser.



Figure 8: Gene browser with pillar only file, Sbayanus is the species being focused and gene Sbay 23.47 is being focused.

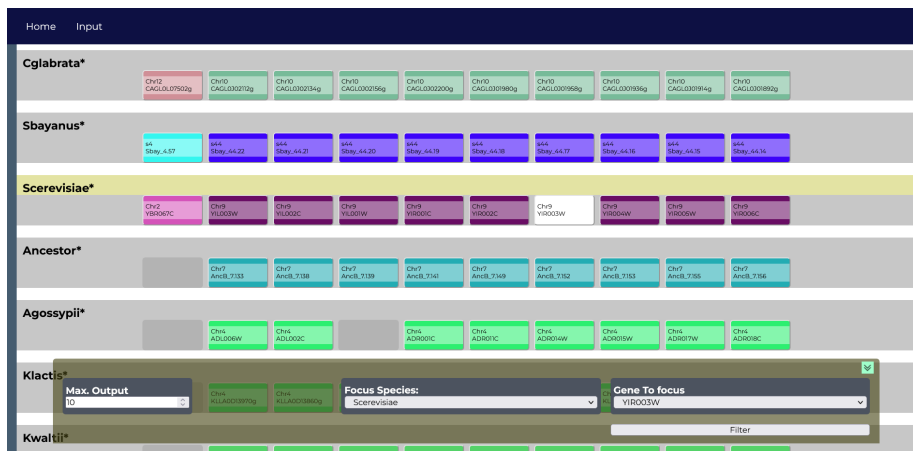


Figure 9: Gene browser with order file, Scerevisiae is the species being focused and YIR003W is the gene in focus.

We can observe how the genes are displayed depending on which files have been input from the user. Figure 8 shows what the browser would look like with only the pillar file and figure 9 shows what the browser would look like when the order file for the different species is also added.

4.2.1 Filter features & information display

At the bottom of the window, a filter module is visible, which has all the design features outlined in Figure 2. A latent React hook is utilized to track changes in the filters. This hook ensures that the pillars are regenerated and updated accordingly. The browsers start the gene list at the focused gene and end at the maximum allowed distance from the focused gene.

To maximize visibility of the genome, the filter module can be hidden by clicking the arrow located in the right corner of the module.

When a gene is clicked in the gene browser, a small window will appear, providing available information about the gene, as shown in Figure 9. Additionally, the colors assigned to the genes correspond to the ID numbers of their respective source

sequences. Any genes displayed without color, represented by a black color, indicate pillars that lack an available gene order file.

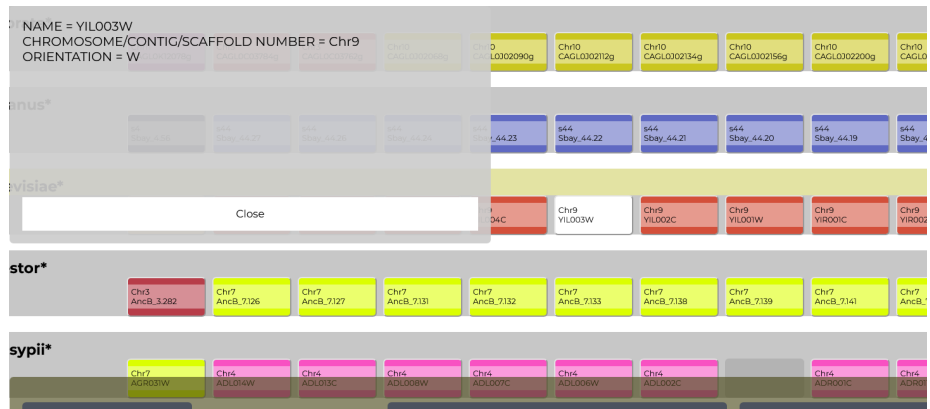


Figure 10: Information window and ordered genes.

4.3 React Development

React proved to be a valuable tool in the development process, offering a dynamic workflow through the reuse of multiple components and functions. The use of hooks introduced latent functionality, allowing components like the filter component to efficiently update other components in a queue-like fashion. The useContext feature, as mentioned in section 3.3, played a crucial role by providing easy access to global data within necessary components, eliminating the need for external storage. This dynamic approach to back-end development greatly contributed to the creation of the gene browser, enabling seamless data handling and enhancing overall flexibility. To demonstrate the project I linked my github account to Vercel. From Vercel I set up an environment for the repository to go online. Finally, Vercel built the project and launched the IGGB [8].

4.4 Design changes

During the process of making the gene browser, some changes had to be made to the initial design. This was partly due to time constraints and the browser becoming more advanced than initially intended. Most notably is the feature to see the relative distance in the chromosome segments between homologous genes as shown in Figure 3. The design of the file upload section displayed in Figure 4 changed from being drag and drop to a simple file upload button. However, since the input allowed users to add multiple files simultaneously, all the separate files could still be added at once, simplifying the process for users.

There were some missing elements with the design, which will be further discussed in section 5. These elements didn't affect the browser directly, but could have had some effect on the user experience. Some of these flaws include not giving the user the possibility to write their own information columns. This could potentially expand the information slots further to provide more information for every gene. Which could give more information for the browser to display.

5 Conclusion

In this chapter we'll discuss the result and compare the YGOB and IGGB browser to then evaluate.

5.1 Discussion

The IGGB could show promising results in terms of its user focused design and functionality. By allowing users to upload their own data, the IGGB offers more flexibility compared to gene browsers with missing input features, like YGOB. By incorporating interactive features, the browser enables users to simplify the selection process for their own data, creating an easy and engaging browsing experience. Users may efficiently navigate through their data and provide relevant information about each gene. This could make the IGGB a capable tool with user-friendly traits making analysis easier. Figure 7 illustrates the effective display of gene order data availability and species selection within the IGGB.

The gene browser component of the IGGB presents users with a visually appealing and informative representation of homologous genes. The color-coded tracks and accompanying gene names and source sequence identifiers make it easier to identify and interpret the data. The filter module at the bottom of the gene browser window offers users the ability to customize their viewing experience by ordering the synteny after a specific gene.

5.2 Evaluation

The comparison between the YGOB and the IGGB reveals distinct strengths and weaknesses for each platform. While YGOB relies on its own data for accurate synteny analysis,

IGGB allows users to manipulate and interpret their own data within the framework of its functionalities.

One of the key strengths of the YGOB lies in its use of carefully curated data on synteny relationships. By relying on their own data, the YGOB ensures more accuracy and consistency in synteny analysis by consistently updating and revalidating their own data for public use [3]. Researchers and users who prioritize reliability and validated data may find YGOB to be a valuable resource. However, this strength can also be viewed as a limitation since the synteny analysis is restricted to the data included in YGOB. Users who have specific genomic datasets outside the scope of YGOB may face challenges in using its capabilities fully.

On the other hand, IGGB provides a flexible and user focused approach to gene data analysis. The ability to manipulate and interpret their own data within the IGGB framework allows users to explore resemblance and relationships specific to their research interests. This flexibility allows for a broader range of applications and enables users to investigate gene orders and synteny patterns beyond the constraints of a predefined dataset. By offering a user-friendly interface and interactive visualization options, IGGB promotes a more exploratory and customizable experience.

In summary, YGOB and IGGB have distinct advantages and trade-offs. YGOB's strength lies in its reliance on a curated dataset, ensuring accuracy and consistency, while IGGB empowers users to analyze their own data and interpret it within a flexible framework. Both platforms serve different user needs and research contexts, highlighting the

importance of understanding the strengths and limitations of each approach.

Future advancements could involve integrating the strengths of both platforms, combining curated datasets with user driven analysis capabilities, to provide a comprehensive gene browser solution for the scientific community.

References

- [1] David Adams. "Definition and use of Bioinformatics". In: (). URL: <https://www.genome.gov/genetics-glossary/Bioinformatics>.
- [2] Sanjida H Rangwala and Anatoliy Kuznetsov et al. "Accessing NCBI data using the NCBI Sequence Viewer and Genome Data Viewer (GDV)". In: (2021 jan), pp. 159–169.
- [3] Kevin P. Byrne and Kenneth H. Wolfe. "The Yeast Gene Order Browser: Combining curated homology and syntenic context reveals gene fate in polyploid species". In: *Genome Res* (October 2005). DOI: 10.1101/gr.3672305.
- [4] N. Stein. "Synteny (Syntenic Genes)". In: *Brenner's Encyclopedia of Genetics (Second Edition)*. Ed. by Stanley Maloy and Kelly Hughes. Second Edition. San Diego: Academic Press, 2013, pp. 623–626. ISBN: 978-0-08-096156-9. DOI: <https://doi.org/10.1016/B978-0-12-374984-0.01508-4>. URL: <https://www.sciencedirect.com/science/article/pii/B9780123749840015084>.
- [5] *Homologs, Orthologs, and Paralogs*. [Online; accessed 2023-08-04]. Dec. 2022. URL: [https://bio.libretexts.org/Bookshelves/Microbiology/Microbiology_\(Boundless\)/07%3A_Microbial_Genetics/7.13%3A_Bioinformatics/7.13C%3A_Homologs_Orthologs_and_Paralogs](https://bio.libretexts.org/Bookshelves/Microbiology/Microbiology_(Boundless)/07%3A_Microbial_Genetics/7.13%3A_Bioinformatics/7.13C%3A_Homologs_Orthologs_and_Paralogs).
- [6] Giudicelli V. and Lefranc M.P. *Ontology for Immunogenetics: IMGT-ONTOLOGY Bioinformatics*. 1999.
- [7] Danilo Catalan Canales. *Open source github repository*. Jan. 2022. URL: <https://github.com/DCatCan/IGGB>.
- [8] Danilo Catalan Canales. *Working website for the IGGB*. Jan. 2023. URL: <https://iggb-dcatcan.vercel.app/>.

Datalogi
www.math.su.se

Beräkningsmatematik
Matematiska institutionen
Stockholms universitet
106 91 Stockholm